Expert Systems With Applications 184 (2021) 115486

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

On the relevance of the metadata used in the semantic segmentation of indoor image spaces

Luis Vasquez-Espinoza [a], Manuel Castillo-Cara [b,*], Luis Orozco-Barbosa [c]

[a] Center of Information and Communication Technologies, Universidad Nacional de Ingenieria, Lima, Peru
[b] Universidad de Lima, Lima, Peru
[c] Albacete Research Institute of Informatics, Universidad de Castilla-La Mancha, Albacete, Spain

ARTICLE INFO

Keywords:
Deep learning
U-net
Semantic segmentation
Metadata preprocessing
Fully convolutional network
Indoor scenes

ABSTRACT

The study of artificial learning processes in the area of computer vision context has mainly focused on achieving a fixed output target rather than on identifying the underlying processes as a means to develop solutions capable of performing as good as or better than the human brain. This work reviews the well-known segmentation efforts in computer vision. However, our primary focus is on the quantitative evaluation of the amount of contextual information provided to the neural network. In particular, the information used to mimic the tacit information that a human is capable of using, like a sense of unambiguous order and the capability of improving its estimation by complementing already learned information. Our results show that, after a set of pre and post-processing methods applied to both the training data and the neural network architecture, the predictions made were drastically closer to the expected output in comparison to the cases where no contextual additions were provided. Our results provide evidence that learning systems strongly rely on contextual information for the identification task process.

1. Introduction

In the past few years, the use of artificial intelligence principles and methodologies has greatly contributed to the development of computer vision processing services and applications. Computer vision applications for pixel-wise classification processes have taken advantage of novel deep artificial neural architectures. They have been key on improving the quality of the results on the tasks of detection (Wang, Gao, & Yuan, 2018), positioning (Martinez-Gomez et al., 2016; Lovón-Melgarejo, Castillo-Cara, Huarcaya-Canal, Orozco-Barbosa, & García-Varea, 2019) and membership based on contextual information (Fu et al., 2020; Ding, Jiang, Shuai, Liu, & Wang, 2020).

The scenario to be evaluated in the present investigation involves a computational vision problem called *semantic segmentation* (Feng et al., 2020); problem that seeks to discriminate, section and uniquely classify each pixel belonging to an element of interest within an image (Souly, Spampinato, & Shah, 2017). Great advances have been made in this field of research; from the invention of the architecture known as *U-Net* (Ronneberger, Fischer, & Brox, 2015) to new technologies that seek to apply this intelligent segmentation in real-time 3D scenarios (Pham, Hua, Nguyen, & Yeung, 2019). In such scenarios, the elements of interest are detected and classified based on a visual reference (Hung, Tsai, Liou, Lin, & Yang, 2018).

The application developed in this research uses the set of photographs of the contest "LSUN: Large-scale Scene Understanding Challenge: Room Layout Estimation", whose goal is to classify the limiting walls of an indoor environment. This dataset has been used in numerous research efforts (Badrinarayanan, Kendall, & Cipolla, 2017; Lee, Badrinarayanan, Malisiewicz, & Rabinovich, 2017 and Dasgupta, Fang, Chen, & Savarese, 2016).

The ultimate goal of the present work is to improve the performance of the semantic segmentation of indoor imagery by exploiting the information provided by the contextual information (metadata) used during the training phase of the neural network. Our work has been motivated by the low performance obtained when applying successive convolution operations to identify elements of interest: an approach widely used by the research community. However the use of the contextual information requires some changes in the structure of the neural network.

In order to actually implement our proposal, the introduction of our approach includes the following contributions:

* Corresponding author.
E-mail addresses: luis.vasquez@uni.pe (L. Vasquez-Espinoza), jmcastil@ulima.edu.pe (M. Castillo-Cara), luis.orozco@uclm.es (L. Orozco-Barbosa).

https://doi.org/10.1016/j.eswa.2021.115486
Received 22 March 2020; Received in revised form 14 June 2021; Accepted 24 June 2021
Available online 6 July 2021

- The development of a masking procedure whose implementation required the preprocessing of the dataset to homogenize the information that comprises the elements of interest within the scene, i.e., the elements to be segmented.
- The definition of a univocal representation of contextual data to be used by the neural network.
- The definition of an algorithm including the processing and adaptation of the contextual information and the structure of the neural network.
- A quantitative evaluation of our approach when applied to the segmentation of the contextual information generated during the training phase of a U-Net.

In this sense, we carried out a comparative analysis of the performance of the U-Net architecture. Our results show that the system converges to a more accurate representation of the borders of the walls given the premise that an agent with cognitive abilities would perform better in a recognition task if a referential pivot factor were provided (Yu, Zhang, Song, Seff, & Xiao, 2015). The main question of this paper can therefore be summarized as follows: could an agent, using the contextual information of the metadata managed during the training phase, provide better learning outcomes and more consistent results than the obtained using only a standard representation of the metadata? Fig. 1 shows the general scheme of the research reported in this work.

Section 1 introduces the work and states the main contributions of this work. Related work is discussed in Section 2. Section 3 describes the tools, the dataset used and the problem of using a Fully Convolutional Network (FCN) without a preprocessing of metadata normalization. Section 4 introduces the pipeline used to solve the problem with emphasis on the algorithmic development of: (i) the generation of masking rule; (ii) the calculation of the mask score; and (iii) the application of mask to the target. Section 5 describes the U-Net architecture indicating optimization, the data preprocessing sequence and the phases of training and forecasting. In this context, Section 6 shows the importance and improvement of our proposal with respect to others regarding the metrics and performance visualization. Finally, the article ends with the conclusions and our future work, Section 7.

2. Related work

Most research efforts on semantic image segmentation make use of FCN architectures (Garcia-Garcia, Orts-Escolano, Oprea, Villena-Martinez, & Garcia-Rodriguez, 2017) and innovative solutions such as the ResUnet-a framework used in urban semantic segmentation of high resolution aerial images (Diakogiannis, Waldner, Caccetta, & Wu, 2020.) The main task of the FCN networks in the overall segmentation process is to determine in a discriminatory manner which pixel corresponds to a given class, regardless of its location (Armeni et al., 2016). In this work, our data set consists of a set of indoor environments images. Once the FCN has estimated the classification of the pixels present in its output, a comparative metric with the expected segmentation is applied.

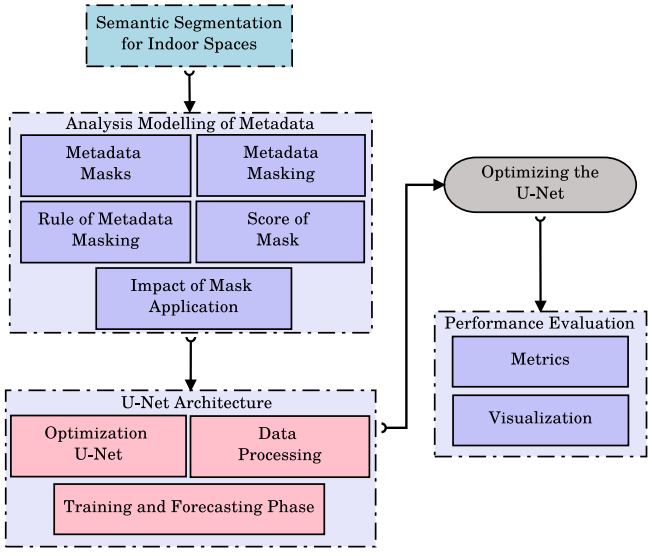


Fig. 1. Overall schema of the Semantic segmentation with U-Net Optimization for Indoor Images Spaces.

2.1. Semantic segmentation

Semantic segmentation has spurred the interest of many researchers working in the field, not only in the processes of image segmentation as presented in this work, but on other related processing mechanisms, such as Indoor Mobile Laser Scanners (IMLS) (Nikoohemat, Peter, Oude Elberink, & Vosselman, 2018; Yi et al., 2019). Such research efforts look at locating robots in indoor environments through semantic location. The ultimate goal of such works is to enable the autonomous navigation of robots by classifying scenes pertaining to the robot trajectories (Romero-González, Martínez-Gómez, García-Varea, & Rodriguez-Ruiz, 2017). This work aims, not only to identify elements of interest in the image, but also to locate them.

The work presented by the authors of Long, Shelhamer, and Darrell (2015) evidences the favorable properties of using FCNs in the semantic segmentation of images. Their study comprises a quantitative comparison to other approaches reported at that time. Their work also details the adaptations made to the neural networks structures to fully operate as FCN architectures. Their results show an amazing performance improvement on the segmentation tasks of around 67%. The research developed in this article owes much to the theoretical-comparative study between CNNs and FCNs, which although it does not use the same architecture as the one used in this work, it does apply the transformation described to pass from a classical classifier to one based purely on convolutions.

Looking to discern biomedical elements from images, Ronneberger et al. (2015) implemented the favorite architecture for semantic segmentation tasks: U-Net. The usefulness of this network is based on the combination of outputs present in layers at the same level of depth, adding even more context by a better defined tensor and thus optimizing the location of features of interest. This architecture qualifies as a type of network *encoder–decoder*, composed of two parts: one for coding, and another for decoding. The first is responsible for simplifying and synthesizing generic characteristics of the context, and the second increases the resolution and retrieves the details of the previously simplified information. This process allows improving the pixel-wise classification in the process of semantic segmentation.

Finally, many of the current efforts are focused on using the contextual information in the training process to improve semantic segmentation process (Ding et al., 2020; Fu et al., 2020). In Fu et al. (2020), the authors have considerably improved the performance of the down-sampling process used in the deconvolution phase. Their results have shown very good results in all accounts: the assigned metrics and the segmentation of the scene objects.

2.2. Room layout estimation

This section reviews the current bibliography specifically oriented to the segmentation of components on indoor images (Garcia-Garcia et al., 2018).

In Lee et al. (2017), Chen-Yu Lee et al. implemented RoomNet, a network encoder–decoder of stacked architecture (different from U-Net). The operation of RoomNet is based on predictions or key points that determine the distribution of corners present in the image of the target indoor environment. This approach seeks to fix the coordinate points of interest; rather than applying a pixel-wise analysis of the image

as other related works have done. In this work, the visual data is processed transforming the expected values or targets to a form based on a pre-established template. This process provides static references to the model for its normalization.

In Badrinarayanan et al. (2017), Badrinarayanan et al. present SegNet: a deep, robust architecture, FCN specialized in pixel-wise segmentation tasks. Such architecture makes use of an encoder/decoder architecture, composed of two well-defined parts of processing: the convolutive phase of summary and the upsampling phase of refinement. The present work takes as reference this technique of processing within the network for segmentation tasks, using the two parts of the processing. However, these parts have not been worked as independent sets, instead the encoder communicates with the decoder to add information to the final layers of the architecture used, providing it with additional contextual robustness.

Finally, reviewing the literature of post-training optimization methods, the work in Dasgupta et al. (2016) introduced a novel network, namely the DeLay network. This network is an estimation network specialized in interiors; which, unlike RoomNet (Badrinarayanan et al., 2017), the Delay network aims to optimize the estimated post-segmentation targets. It starts by separating targets into independent channels. It then links them to a parameterized layout, which allows detecting the intermediate segments located between the walls to finally classifies them by extrapolation. In this way, the idea of solving the problem of wall detection is rescued from independent channels, i.e., by binary classifying the walls found, thus training the network for each class indexed according to the channel to which it belongs.

Taking into account this different approaches and methodologies of the reported in the literature, in this work a U-Net type network has been used as starting point. We then modify and adapt its structure to exploit the contextual information obtained during the training phase. In this process, we have defined the overall process as well as the data model allowing to get the most from the contextual information (Cruz, Rangel, Gomez-Donoso, & Cazorla, 2019).

3. Background: tools, dataset and metadata preprocessing

In this section, we describe the tools, dataset and the relevance of the metadata.

3.1. Tools

During the early stages of our work, different tools were evaluated, among which the programming language with specialized libraries addressing our needs. As a result of this evaluation, it was decided to use Python, since the capabilities and tools available are for immediate use and accessible implementation. Likewise, OpenCV was used as a library specialized in computational vision, to apply high-level operations to images within the set of data manipulated during preprocessing, training and validation, thus saving computational load to the neural network. Moreover, among the alternatives available in the catalog of libraries specialized in deep learning, it was set to use PyTorch.

3.2. LSUN dataset

The LSUN Dataset is a public domain database used in computer-vision competitions motivating the development of novel proposals for classification and detection of elements in selected images. In the category of semantic segmentation, a consideration of limiting walls was required on a series of images of indoor environments (rooms), which did not have more expected objective detail than the manual labeling of its walls. The LSUN Room Layout Estimation consists of 4000 pairs of training data, 394 of validation and 1000 of test. Each pair is composed of an image of arbitrary dimension and two–dimensional counterpart labeled on the walls that limit it.

Fig. 2 shows a pair of sample data consisting of a tuple of two components. The first one, located on the left, is the photograph of the room in question, while the second, located on the right, describes the room from the segmentation of the walls present.

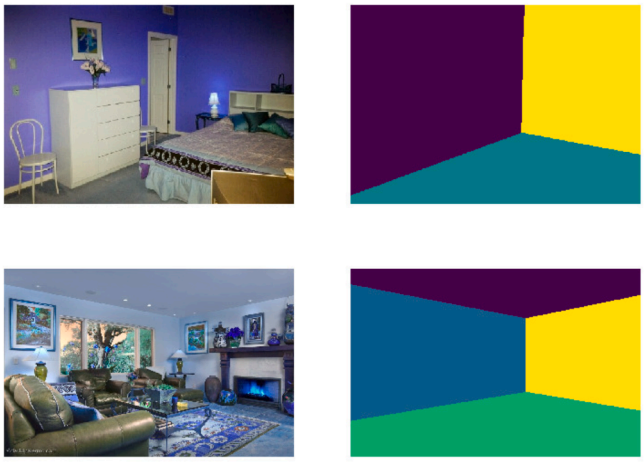


Fig. 2. LSUN Room Layout Estimation data tuples, containing both the image of the indoor location and its segmentation in walls.

Additionally, the LSUN dataset comprises a brief description of its composition. The type of data, shown in the first column groups the entire dataset according to its intended use: data destined to training, validation, testing, of the model to be generated, see Table 1. The second column indicates the number of walls. Finally, the third column quantifies each occurrence of them.

Furthermore, the data is classified based on its dimensions, Table 2. The classification is done once again by considering the type of data according to its intended use, see the first column. The minimum, maximum and average dimensions of the photographs present in the set, in pixels (px), are shown.

3.3. Metadata preprocessing

In the previous section, we provide the details of the dataset being used in our study. From the previous description, it should be clear that although the dataset correctly labels the walls, the data does not provide the means to unequivocally differentiate the walls. Such shortcoming will result on limited learning results when attempting the spatial (location) detection of a wall. Fig. 3 shows the five classification channels corresponding to the five boundaries of the rooms: left, right and central walls, ceiling and floor. For each image, the *target* is plotted in each one of the five channels at the same height as the original image (first, third and fifth row). Rows 2, 4 and 6 show their corresponding *outputs* after using the trained model.

As seen for the second channel, second column of the results, a wall located to the left in the first image is properly detected. However, in the case

Table 1
LSUN Dataset description by number of considered walls.

Designation	Considered walls	Data quantity
Training	1	0
	2	262
	3	1111
	4	1976
	5	650
	6	1
Validation	1	0
	2	18
	3	111
	4	205
	5	60
	6	0

Table 2
LSUN Dataset description by data dimension.

Data designation	Minimum dimension (px)	Maximum dimension (px)	Average dimension (px)
Training	256x200	3744x5616	887.4x1163.0
Validation	256x256	3456x4608	918.7x1185.3
Testing	191x255	4680x4680	881.76x1153.3

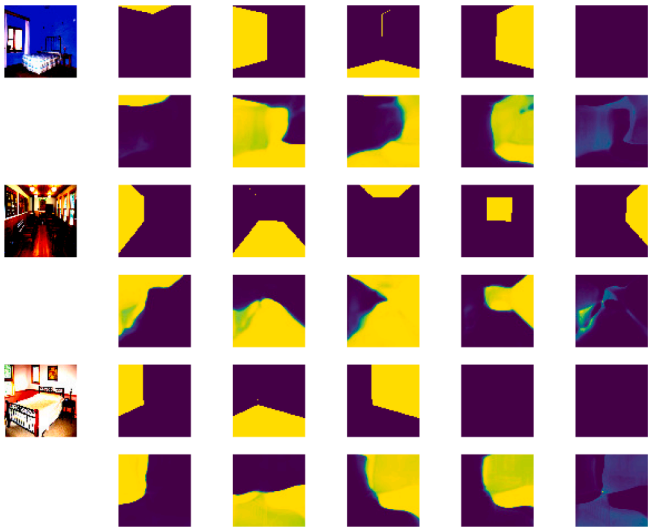


Fig. 3. No-preprocessing model output for three images of the validation set. Having 2 descriptive rows for each, the first one shows the target (expected segmentation); while the second one shows the model output. All five columns represent the walls, floor and ceiling that it may find in an indoor image.

of the other two images, the floor is identified and not the left wall. These results clearly show the need of properly processing the channel labels.

Under this context, we should preprocess the data by assigning each item (wall, floor, ceiling) to a given channel. Such assignment will provide us with the means of easily identifying the position and type of the room boundaries. The results of such process can be a very valuable source of information for future research. This was achieved through the use of masks with preset values to count frequencies over masked values.

4. Metadata modeling analysis

As previously mentioned, it is necessary to normalize the data to add robustness to our neural network. To better illustrate the entire preprocessing, the flowchart shown in Fig. 4 is used as reference.

Starting from the data collection (*Step 1*), the diagram describes an arbitrary data such as the two–component tuple: the image of the target indoor environment and the labeling of its boundaries. Subsequently, the labeling of its walls is performed using the masking process (*Step 2*). As a result of this process, the assigned mask will be obtained based on the scoring system. The dataset is then updated using winner mask (*Step 3*). This preprocessing is applied to all available data. The procedures of the three steps are further detailed in the following sections.

4.1. Definition of metadata masks

The masks to be applied to a given target data (images) should serve as a reference to other processes by homogenizing their representation. It must be taken into account that a target is only the second component of the data tuple, represented by a two–dimensional matrix with the labeling of the case. The criterion for selecting these matrices from the original dataset was based on finding the most generic patterns

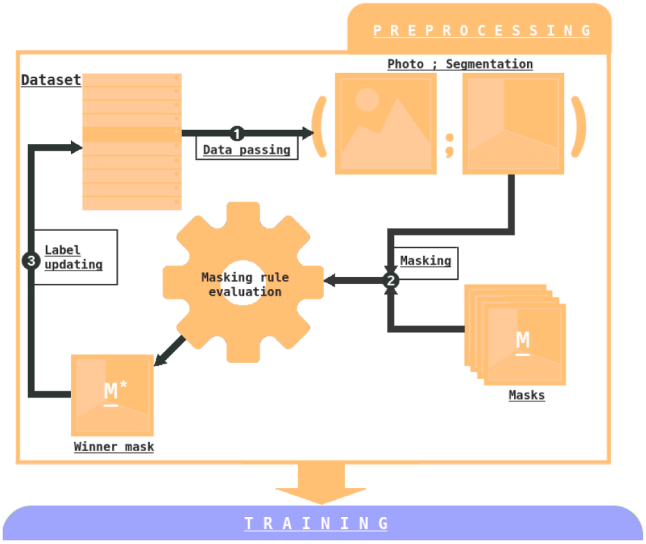


Fig. 4. General scheme of preprocessing aiming to normalize the data. Iterating over the provided dataset, this phase takes each segmentation component (target) and standardizes its labels, applying the masking process from the winner mask's masking rule.

applicable to the photographs. Moreover, the modification of its values was made based on the rules shown in Table 3.

Fig. 5 provides the details of the 17 proposed filters with their labels. For each filter, their labels are assigned clockwise. For instance, for label 1, the standardization mask assigns IDs 1 to the left wall, label 3 to the right one; and label 4 to the floor.

Once having defined the masks to be used, we can proceed with the training process with the main goal of obtaining the quantitative data to be used later in the scoring process. In order to clearly defined the terminology used, we formulate the two following definitions.

Definition 1. Let M and T be a masking matrix (or mask) and a target matrix, respectively, it is said that a pixel $p_M \in M$ masks (::) another pixel $p_T \in T$ if the coordinates relative to their matrices match.

$$p_M = (p_{Mx}, p_{My}) \in M \subset \mathbb{N}^2$$

$$p_T = (p_{Tx}, p_{Ty}) \in T \subset \mathbb{N}^2$$

$$p_M :: p_T = True \Leftrightarrow p_{Mx} = p_{Tx} \wedge p_{My} = p_{Ty}$$

Definition 2. Let M be a matrix, for the context in which the topic is developed, it is said that $r \in \mathbb{N}$ is a label of M if this is present in M.

$$label(M) = r \Leftrightarrow r \in M \subset \mathbb{N}^2$$

These definitions simply redefine the original idea regarding the elements contained in the masking and target matrices used in the masking of a pixel in T by a pixel from M.

Table 3
Standard masking rule that associates each component (walls, floor and ceiling) to an identification (*ID*).

Wall	ID
Left wall	1
Ceiling	2
Right wall	3
Floor	4
Background wall	5

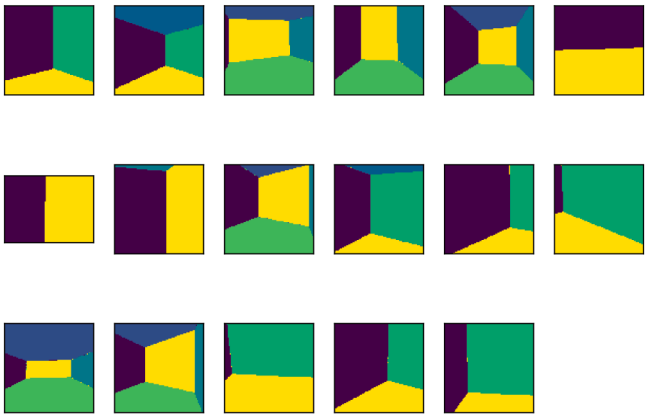


Fig. 5. Standardization masks for the possible wall arrangements, considering the cases in existences within the dataset.

The first task consists on counting, in the target matrix, the number of pixels corresponding to a given label. This process will result on obtaining at most five groups of pixel counts (one for each wall). Thereafter, a mask is taken and it is counted, for each single label r, the percentage of the total unique values of the target matrix masked by a given label. Finally, the complement of this percentage is taken to determine the number of pixels of the target not having been masked by the right label. This percentage represents the masking error. Once this process is finished for each label r, there will be a maximum of five error percentage groups, i.e., a total of 25 percentages in total per mask.

To illustrate the process, consider the case shown in Fig. 6 consisting of a target of three labels (a, b and c) using only two masks. First, the procedure counts the pixels of each label of the target, represented by the histogram shown in the upper right corner. Then, it quantifies the number of pixels identified by each one of the available masks (1, 3 and 4) for each label of the target (a, b or c). This allows us to count the error correspondences according to the percentage not masked.

Take for instance, the result obtained for *mask 1*. As seen from the histogram of *label 3* (*mask 1, label 3*) this one masks quite well the 'b' values of the target. This is confirmed by the low value (height) of the green bar shown in the histogram for this case.

Another enlightening example is the case of *mask 2*, which for its *label 4* (*mask 2 label 4*) counts a high error rate for the values 'a' and 'b'. This result shows that the value '4' does not properly mask them. However, this same value is still capable of masking 'c' satisfactorily.

The example shows that the second mask has the same segmentation as the target. This assignment was intentionally set as we will show later that the masks should be selected as indicated to better homogenize the data. The main goal of this first preprocessing step is to precisely lay the quantitative basis for finding (1) the mask best fitting the target; and (2) the mask unequivocally identifying a single label. This last idea sets the evaluation criteria: to qualify a mask label according to its ability to mask values with a low percentage of error, and to penalize those values that do not specialize in a single masking. Therefore, we must first define the criteria under which compare the usefulness of the masks.

4.2. Rule of metadata masking

Once the masking is finished, the error percentages will be used to generate a rule under which the target labels should be replaced with new ones to homogenize the dataset. The structure of a rule for a mask M based on a label of target r_T will have the form $r_M : (r_T, e)$. Note that this dictionary will have the best performing r_M values as keys, and the values to be stored will be two–dimensional tuples that contain the value to be masked (r_T) and the masking error (e). The masking rules items translate to an r_M label masking the r_T values in the target with a percentage error of e.

The generation of these rules can be implemented under the sequence detailed in Algorithm 1, where mask M and target T are defined as the matrices to be used, in addition to initializing an empty dictionary called *Rule*. Subsequently, and for each T (r_T) label, that M (r_M) label that masks it with a minimum error will be searched. Then both, the newly found optimal label and the corresponding error, should finally be recorded in *Rule*.

Algorithm 1.

Algorithm 1:Masking rule generation.

1: $T \leftarrow Target$
2: $M \leftarrow Mask$
3: $Rule \leftarrow \{\}$
4: **for** label $r_T \in T$ **do**
5: $\quad r_{min} \leftarrow r_M \in M / error(r_M, r_T)$ be minimum
6: $\quad error_{min} \leftarrow error(r_{min}, r_T)$
7: $\quad Rule.append(\{r_{min} : (r_T, error_{min})\})$
8: **end for**
9: **return** *Rule*

Returning to the example set out in Fig. 6 and only using the image, it is stated that the resulting rules are those shown in the Masking Rules column of Table 4. Here, the rules indicate which mask best fits the target value. For instance, for *mask 1*, the value '1' will mask the value 'a' in the target with an error of 0.4 or 40%; while for *mask 2*, the masking value '4' exhibits and error of 0.04 or 4%.

In this sense, knowing in advance that the second mask is better than the first one for the given target, it is necessary to numerically express this difference. This is done by rewarding/penalizing every mask.

4.3. Mask scoring

At this point, it is necessary to identify the best mask for the target to be standardized. The scoring is performed by using two key metrics: (i) the performance of one value to mask another; and (ii) the specialization on said single masked value. Until the previous step the process has dictionaries containing the best performing values. If these values still have a high percentage error, it would indicate that the mask may not be adequately compared to others. To consolidate this rating numerically, it is necessary to consider all the percentage errors. In this way, if the accumulative value of a mask is very high with respect to the ones reported for other masks, we should not only be able to identify the best option, but also to quantify its suitability for our purposes.

Definition 3. The mask p_d score generated from a masking rule d will be calculated as the sum of the errors present in d.

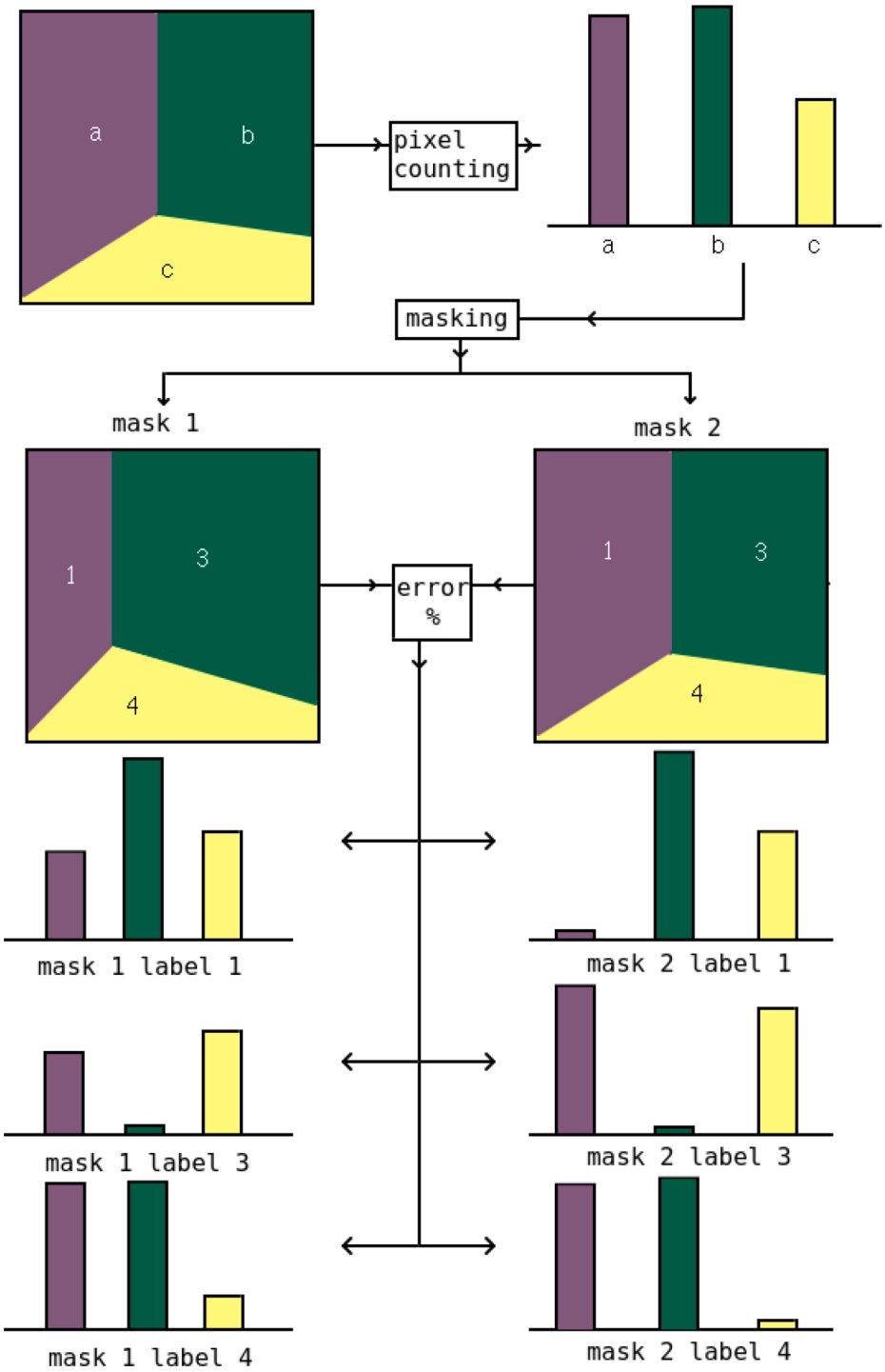


Fig. 6. Masking process example for two masks. Initially, pixel counting is performed over the target matrix, obtaining the unique label frequencies used as input for the masking trial (considering two different masks for this example), where *mask 2*, by simple inspection, should mask the target more accurately compared to *mask 1*. This idea is supported by how, after obtaining the frequency diagrams for each mask, a heterogeneity is noted for a single masked value for each unique label in the target, proving the one-to-one correspondence of the second mask.

Table 4
Rule generation example, in which after applying the algorithm we can see that the second mask is the one with lowest errors.

Mask	Error (%)	Masking Rules
Mask 1	1:{a:40, b:90, c:90} 3:{a:45, b:5, c:90} 4:{a:90, b:85, c:10}	{1: (a,0.4), 3: (b,0.05), 4: (c,0.1)}
Mask 2	1:{a:3, b:95, c:90} 3:{a:99, b:1, c:89} 4:{a:90, b:85, c:4}	{1: (a, 0.03), 3: (b, 0.01), 4: (c, 0.04)}

$$score\left(p_d\right) = \sum_{d_i \in d} error\left(d_i\right)$$

Obtaining this score can be implemented using the Algorithm 2 as a reference, where after defining the target and the mask to be used, the masking rule is generated from these two matrices. This is done by following the procedure detailed in the previous section. Subsequently, it will be iterated for every item present in the resulting dictionary R. The masking errors for each label, located in index 1 of the internal tuple of each item r, will be then accumulative added to acc. This variable represents the mask scores.

Table 5
Mask scoring example, concluding that the winner mask is the second one with an error of 0.08. Best result is shown in bold.

Mask	Masking rules	Mask score
Mask 1	{1: (a,0.4),3: (b,0.05),4: (c,0.1)}	0.55
Mask 2	{1: (a,0.03),3: (b,0.01),4: (c,0.04)}	**0.08**

Algorithm 2.

Algorithm 2: Mask score calculation.

1: $T \leftarrow Target$
2: $M \leftarrow Mask$
3: $R \leftarrow GenerateRule(T, M)$
4: $acc \leftarrow 0$
5: **for** item $r \in R$ **do**
6: $\quad error \leftarrow r.\text{value}[1]$
7: $\quad acc \leftarrow acc + error$
8: **end for**
9: **return** acc

Table 5 shows the score obtained for the sample mask 1 and 2. The mask exhibiting the lowest score represents the best option, e.g., *mask 2*.

4.4. Mask application

Algorithm 3 describes the process of replacing the target values based on the correspondence rule of the winning mask. Here the best masking rule, R, is determined, so that subsequently and for each pixel of the target, T, its relevant value is replaced.

Algorithm 3.

Algorithm 3: Mask application over a target.

1: $T \leftarrow$ Target
2: $\overline{M} \leftarrow$ List of available masks
3: $\overline{R} \leftarrow GenerateRules(T, \overline{M})$
4: $R \leftarrow getBestRule(\overline{R})$
5: **for** pixel $p \in T$ **do**
6: $\quad tuples \leftarrow R.values$
7: $\quad r_T \leftarrow tuples[p].key$
8: $\quad p \leftarrow r_T$
9: **end for**
10: **return** T

5. Neural network architecture

We describe in this section the architecture of the neural network by highlighting the changes required to process the contextual data.

5.1. Optimized U-net architecture

We use as departure an U-Net architecture of identical dimensions to those set forth in Ronneberger et al. (2015), but varying three parameters of its parameters:

- **The dimension of the input tensors**: The U-Net was prepared to be able to accept input tensors of dimension `1x3x200x200` (having originally been `1x3x572x572`), thus reducing the memory used to load the images and optimizing the iteration time over the images.
- **The first downsampling**: The first tests carried out on the architecture made it decisive to reduce the dimensions in order to homogenize the network to our specific problem. This change provides us with better performance and computer consumption results. Hence, the first downsampling performed does have an impact on the processing by applying a *trade-off* between processing time and resource demand. We observe that this change does not have an impact on the segmentation and generation of the network.
- **The depth of the network**: In this second parameter, the fixed depth was extended to 5 levels of downsampling and upsampling, exceeding by one the levels considered in the original architecture and thus generating a more gradual way of resizing (Sherrah, 2016). Understand as a level the group of tensors in the U-Net product of convolutive operations that do not alter its planar dimensions (those with width and height of `200x200`, `100x100`, `50x50`...).
- **Adding a complement layer**: As explained in the next section, it has been deemed necessary to optimize the segmentation of the back wall located in the last channel of the target, from a complement operation on the other four channels, to sharpen better the edges of the walls in the estimate. Note that this last layer is connected to the network, i.e., it is not simply an annexed layer but relates to the classic layers. Therefore, this last layer has its synaptic weights, participates in backpropagation and all CNN operations.

5.2. Data processing sequence

Fig. 7 shows the way the images are processed within the network. Each layer is represented by a rectangular solid, and for each of these the expected dimensions of its output are displayed. Using this reference figure, and tracking the U-Net from left to right and from top to bottom, the procedure is as follows:

1. Starting with the input layer, an arbitrary dimension photograph will be taken and a tensor of dimensions `1x3x200x200` will be returned, depicted by the three squares shown below the input image.
2. Two successive operations of the same convolution will then be applied with 64 filters each, resulting in two adjacent tensors of dimension `1x64x200x200`.
3. Another convolution is applied to reduce the dimensions of the tensor in half: a downsampling convolution. Another convolution follows, resulting in the tensor of dimensions `1x64x100x100`.

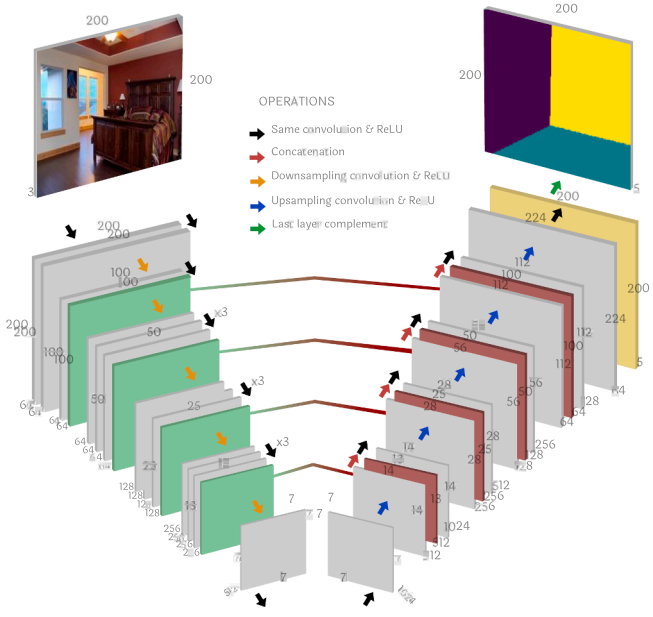


Fig. 7. Custom U-Net architecture diagram. By transversing the U-Net from the input layer (upper left corner), the network applies, to the indoor image, successive downsampling and same convolution operations, until a dimension threshold is reached (middle layers of the U-Net). Eventually, the pertinent walls information is obtained after successive upsamplings, to finally be homogenized into the results, concluding in the estimated segmentation of the room's comprising walls.

4. From here, emphasis is placed on the color markings that connect the last layer to each respective level. They will be then used for concatenating them with the results of subsequent convolutions.
5. Successive convolutions and downsamplings operations will continue to be applied until reaching the deepest layer, where the tensors have a dimension of `1x512x7x7`. As a result of the last convolution, a tensor of `1x1024x7x7` is returned.
6. From this point, the task of the remaining layers is to progressively increase the dimensions of their inputs, until obtaining a tensor of dimensions comparable to those of the original image. For this, a first upsampling convolution will be applied, which will prepare the tensor to use the output of two levels back (indicated by the red connection), thus concatenating the result of this upsampling (of dimensions `1x512x14x14`) with the transferred data (of dimensions `1x256x13x13`).
7. Since the concatenation performed, represented by the dark red arrow, does not present dimensionality discrepancies, an additional convolution is applied to the transferred tensor. The final result corresponds to the concatenation of the tensors of dimensions `1x512x14x14` and `1x512x13x13`.
8. The procedure for concatenation and convolution pairs will be developed for the remaining levels, until a tensor of dimensions `1x64x224x224` is obtained, to which a last upsampling will be applied resulting in the prediction of the label of the room (of dimension `1x5x200x200`).
9. Finally, to sharpen the estimation of edges of the walls to be detected, a last layer will be applied where the last channel (corresponding to the bottom wall) is segmented from the complement of the combination of the other four. As shown in Fig. 8, this last layer takes the parts of the first four channels to integrate them into a single data matrix, to extract the complement of the latter and thus define the bottom wall. As a result of this last step, a tensor of dimensions `1x5x200x200` similar to the input will be obtained, but with sharpened edges, distributing the five walls possibly present in the five available channels of the second component.

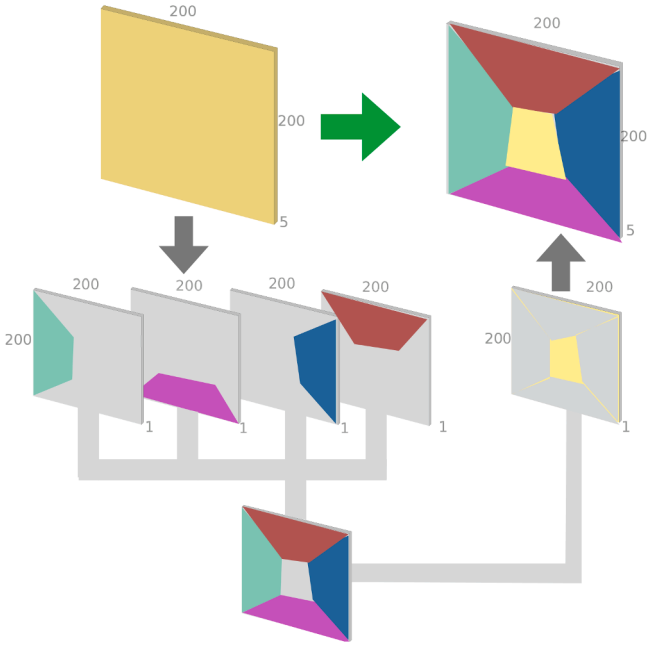


Fig. 8. Last hidden layer diagram, which applies an edge refinement by creating a last depth channel using the other one's complement, thus optimizing the segmentation.

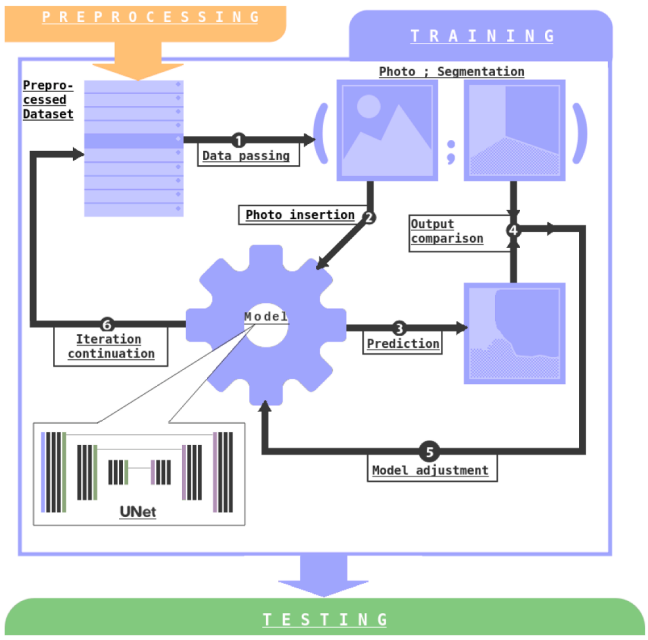


Fig. 9. Scheme of training phase. Taking the preprocessed data from the previous masking operation, the model consisting of a customized U-Net will estimate a prediction of the expected segmentation, and compare it to the recently preprocessed segmentation target in order to adjust the model, continuing the training process with the next input iteration.

The previous procedure can be summarized as follows: The image entered gradually will be resized by successive convolution processes, thus taking advantage of its property of immutability of characteristics and giving depth to the expected final label. Another detail to take into account is that all the convolutions described are attached to a ReLU rectification procedure, which normalizes the data and optimizes the training.

5.3. Training phase

Having defined the U-Net architecture to be used, we now describe the central procedure of the network training, refer to Fig. 9:

1. The sequence initializes from the end of the preprocessing of the data. The data will consist of a pair: photography and labeling (*step 1*). The first element will be taken as input to the model to be trained (*Step 2*). The model is represented by the central gear of the diagram being a U-Net.
2. The result of the processing of the photograph will generate a prediction of the labeling of the present walls (*Step 3*), which will be compared with the expected labeling stored in the second component of the data (*Step 4*). This is done to calculate the difference between the two tensors using cross entropy.
3. Appropriate adjustments to the model on the convolution filters present in the U-Net will be determined (*Step 5*). This is done by using backpropagation on each filter and normalizing the activation parameters for the layers involved.

Table 6
Hyperparameters in the training phase.

Hyperparameter	Value
Epochs	45
Batch size	1
Gamma	0.1
Learning rate	0.0001
Step Size	30

4. Finally, the evaluation of the next set data (*Step 6*) will continue until the convergence criterion is met.

It is appropriate to specify that the training could use several criteria to finish the iteration, such as determining a maximum allowed error between the labels or setting a number of fixed repetitions. For the present investigation the convergence of the model was supported in the amount of available data, number of times and the learning rate defined before training (see Table 6). The values of these and other hyperparameters were determined empirically, seeking to optimize the overall process that includes preprocessing, training and final tests.

5.4. Evaluation phase

Once the modeling with the results validated by the relevant data, the performance of this model will be determined in some supervised cases. From this process, we should obtain a specific label and some others closer to reality.In the case of the latter, we do not have any a priori information. The sequence of this process can be seen in Fig. 10 and it can be summarized as follows:

1. First, a set of test data is set; this data can be specific to the LSUN set or some other independent source.
2. Without loss of generality, it can be assumed that this dataset is made up of the already known two–dimensional tuple (displayed in *Step 1*), from which the photograph of the indoor environment will be taken and evaluated in the trained model (*Step 2*).
3. The metrics corresponding to binary cross entropy and dice loss is then calculated, so that the final loss is weighted from these two and the weight assigned to each one (*Step 3*).
4. Once obtained the aforementioned metrics, the calculation of the next tuple of the set (*Step 4*) should be performed to cover all the data of interest.

These results will be discussed in the next section, where they will be compared with those obtained prior to preprocessing.

6. Experimental results and evaluation

In this section, we should evaluate the performance of our proposals. Taking into account the modifications applied to the original U-Net neural network and having explained the characteristics of the customized version, three semantic segmentation solutions will be analyzed: (i) the one using the architecture described in Fig. 7, i.e., the one not making use of the metadata, from now on referred to as `U-Net`; (ii) the one including the preprocessing of the metadata, from now on referred to as `U-Net + WPP`; and (iii) the one including in addition to preprocessing the metadata, a refinement postprocessing of the complement of the channels, illustrated in Fig. 8, and from now on referred as `U-Net + WPP + RC`.

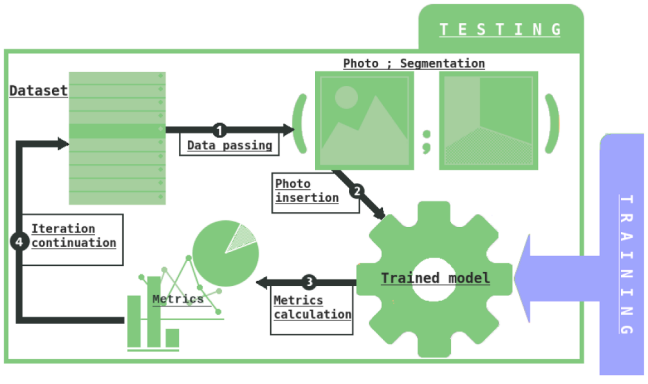


Fig. 10. Scheme of testing phase. After the training phase is concluded, the trained model fulfills the function of estimating the segmentation's accuracy, corroborating the model's efficiency.

Since the main contribution of this work has been the introduction of a metadata preprocessing mechanism offering added services, we should properly choose a metric capable of assessing the performance of the solution in terms of the segmentation results as well as of the added services. In fact, the choice of the most adequate semantic segmentation metrics has been and still is a research topic of great interest (Zhang, Mehta, & Caspi, 2021). In Csurka, Larlus, and Perronnin (2013), the authors have shown that in some instances, a user study should be necessary by taking some sample images in order to be able to assess the performance of semantic segmentation mechanisms.

In our case, we have decided to evaluate our proposals making use of three different metrics. The first metrics aims to evaluate the performance of the learning process of our proposal. A comparative study with the performance offered by the U-Net is also included. We have then conducted a visual analysis, user level, by showing the results obtained when applying our solutions to three images, selected by exhibiting challenging features, as explained in SubSection 6.2.

6.1. Metrics

To be able to monitor the performance of the learning process, we have defined a metric capable of providing us with an assessment of two fundamental features of the error produced when comparing the expected labeling to the estimated one, namely how different and how homogeneous are the sets formed by the pixels of same lettering. To capture the former, we have made use of the cost function known as *Binary Cross Entropy* (BCE). In our case, we should carry the evaluation over the matrices belonging to each of the five channels available in the target. Notice that by individually applying the metric to each channel, the metric will implicitly evaluate the benefits introduced by the preprocessing introduced by our proposal. Second, to evaluate the frequency balance of pixels with the same labeling, the cost function *Dice* is used (also known as the Sorensen-Dice coefficient used for the first time in Dice (1916)). Accordingly, we define the loss metric by giving the same weight to both cost functions as follows:

$$loss = 0.5 \times Dice(y, \hat{y}) + 0.5 \times BCE(y, \hat{y})$$

where:

y: Target expected.

$\hat{y}$: Estimated Target.

Fig. 11 shows the loss metric results during the training and validation phases for all the three models under study. From the results shown in the figure, we derive the following analysis for each one of the three models.

6.1.1. *Case 1: U-Net*

During the training phase (see Fig. 11a), we note that the `U-Net` initially reports the highest error of all the three models. However, its loss metric decreases rapidly and even at a faster rate than the one reported by the `U-Net + WPP` models as a function of the Epoch number. A deep decrease around Epoch 28 allows us to predict a considerable learning rate improvement. It even reports better results that the `U-Net + WPP + RC` model around Epoch 35 with a slight performance degradation around Epoch 40 where it definitely begins to converge.

During the validation phase (see Fig. 11b), it is observed that the loss metric abruptly increases around Epoch 28, i.e., the Epoch where the loss exhibited a considerable improvement during the training phase. This trend clearly shows that the `U-Net` model is overfitting. That is to say, the `U-Net` has not been able to generalize what it has learned during the training phase.

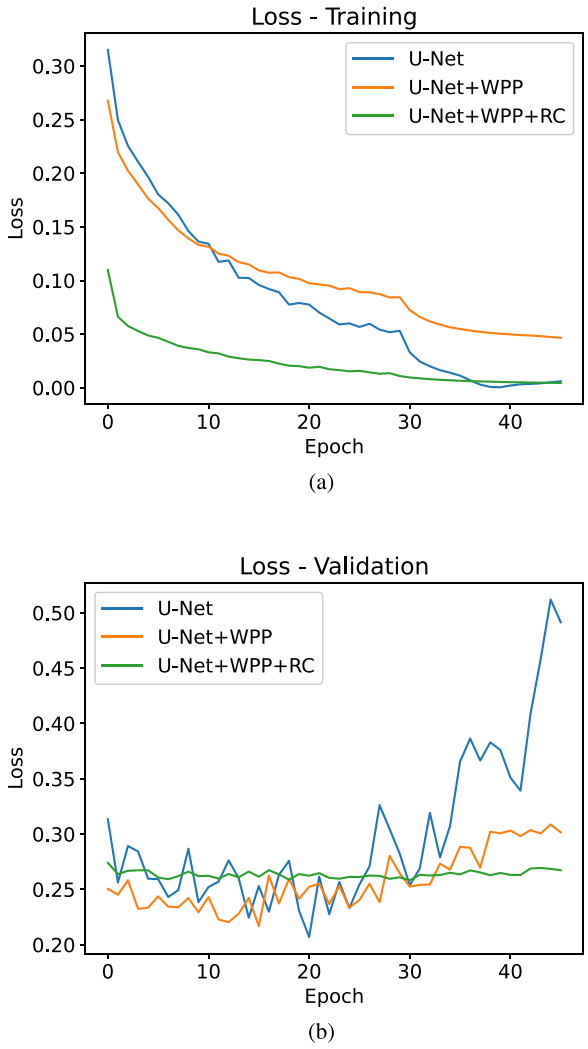


Fig. 11. *Loss* metric obtained by training and validating all three of the used U-Net versions. (**a**) Training; (**b**) Validation.

6.1.2. *Case 2: U-Net + WPP*

In training (see Fig. 11a), the `U-Net + WPP` starts at a point where the error is high but decreases rapidly and even converges in a similar way to the one reported for the `U-Net`. However, the loss reported by `U-Net + WPP` in training as it converges is considerably higher than the one reported by the other two models.

During validation, the `U-Net + WPP` model reports a loss lower than the one reported by `U-Net`. It is particular interesting to observe that different to the Case of the `U-Net`, the loss shows a much lower rise and better convergence around Epoch 40. This result allows us to confirm that the data preprocessing mitigates up to a certain extend the overfitting of the learning process of `U-Net`.

6.1.3. *Case 3: U-Net + WPP + RC*

During the training phase (see 11a), the `U-Net + WPP + RC` model presents a healthier learning curve, starting with a loss value considerably lower than the other two models, and with a soft fall converging to the same value than the one reported by `U-Net`.

In validation (see Fig. 11b), `U-Net + WPP + RC` exhibits a sustained learning constant error. This result validates the robustness of `U-Net + WPP + RC` on validation data that converges from the first evaluations resulting on an effective generalization of the classification learning process.

Fig. 12 shows how the metric is distributed for the three cases studied. If the loss value is analyzed vertically for the datasets for training and validation (located in the first and second column), the improvement is evidenced not only in magnitude, but also in deviation and distribution, minimizing the loss considerably.

Once having evaluated the learning process of the three models, we should visually show the benefits of our proposals. The visualization analysis should allow us to show that the `U-Net + WPP + RC` does not stagnate in the learning phase and that it correctly generalizes the segmentation in the validation phase.

6.2. *Visual Analysis*

In order to provide further insight into the capabilities of our proposal, we include in this section a visual analysis of the segmentation process results of three images. The images have been selected due to the challenging level of difficulty. As seen in Figs. 3, 13 and 14, the first image is characterized by heavy-colored walls, a feature making difficult to distinguish the edge between the two walls. The second presents a long depth of field and lateral window walls while in the third image, the bed prevents the viewer to detect the corner formed by the two walls and the floor. Having made clear the scope and effect of the preprocessing on the implemented neural network, the three figures show the visual results obtained when applying the `U-Net`, `U-Net + WPP` and `U-Net + WPP + RC` approaches to the three validation images.

In this analysis, we should recall that we have made use of the channels distribution described in SubSection 3.3. It should also be clear that this visualization analysis relates to the benefits brought by the preprocessing and postprocessing phases. The former consisting on the use of the metadata allowing us to identify the location and shape of the layout elements of the room: walls, floor and ceiling. The postprocessing phase is included to sharpen the edges of the walls, as explained in SubSection 5.2.

6.2.1. *Case 1: U-Net*

This first method corresponds to the baseline configuration, i.e., the use of the `U-Net` without any extra processing and whose results were reported in SubSection 3.3.

As seen in Fig. 3, `U-Net` is not very robust, evidenced by the poor segmentation of the walls. Instead of resulting in a representation of the walls by polygons of sharp and precise corners, turned them out to be imprecise forms and located in arbitrary channels. As explained in SubSection 3.3, these results suggested that the use of the metadata may prove effective not only to overcome the difficulties of identifying the room layout, but as well to identify their shapes.

6.2.2. *Case 2: U-Net + WPP*

In the Case of the `U-Net + WPP`, (see Fig. 13), an orderly, robust and understandable learning of the network is shown. A significant improvement can be observed when estimating the position and shape of the walls, getting closer to the expected geometric shape and maintaining an order of classification through the five channels. However, an estimation problem is noted for the background wall segmentation, channel 5, located in the last column for all the three images. This estimation error has a very negative impact in the room layout identification process of all three. The worst results are reported for the third image whose channel 2 corresponding to the ceiling exhibits an overlap over channels 1 and 3 (right and left walls). This result motivated us to introduce the postprocessing refinement, Step 9 of the process described in SubSection 5.2, to enhance the learning process of channel 5.

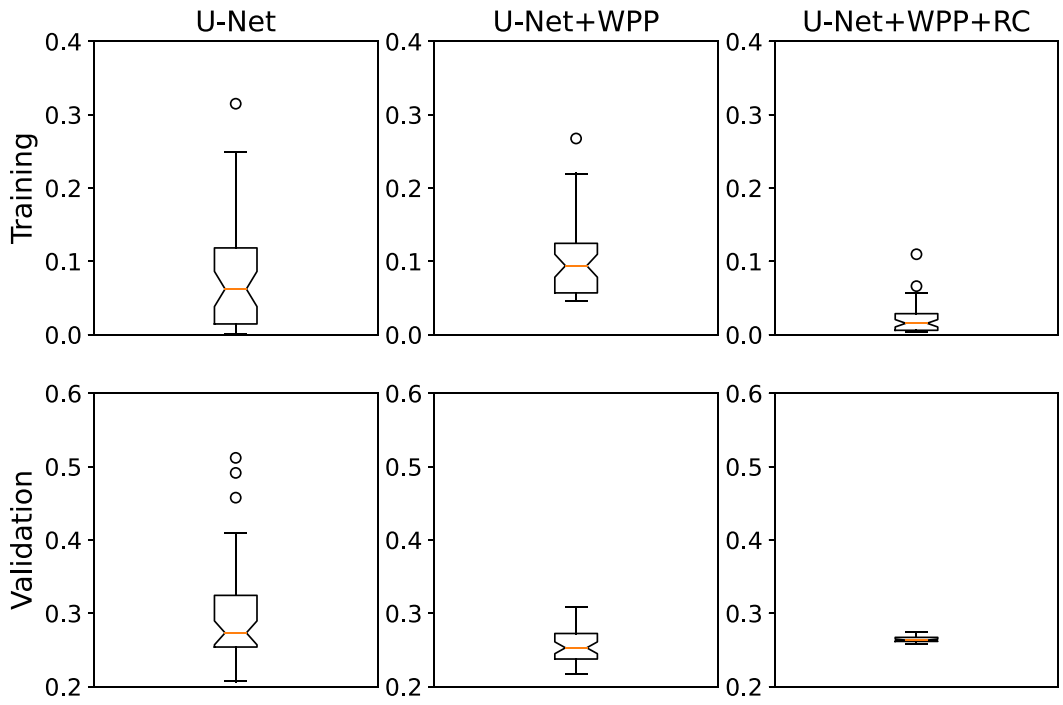


Fig. 12. *Loss* metric distribution by quartiles. It can be seen that using `U-Net + WPP + RC` the loss drops considerably in Training/Validation.

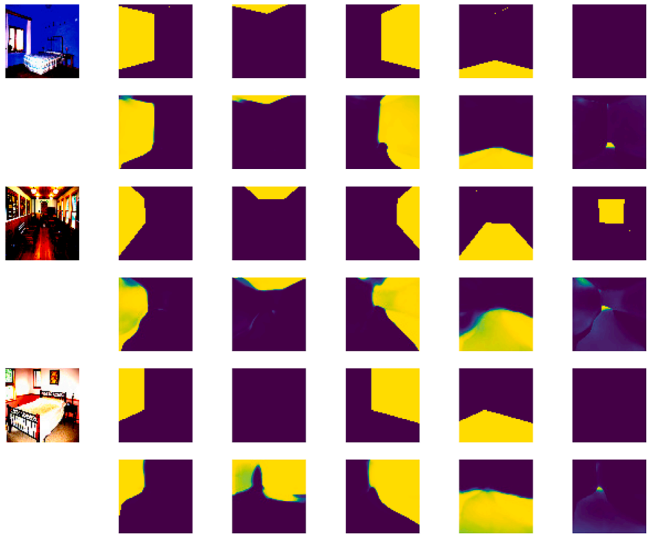


Fig. 13. Semantic segmentation result applying `U-Net + WPP` in the validation dataset.

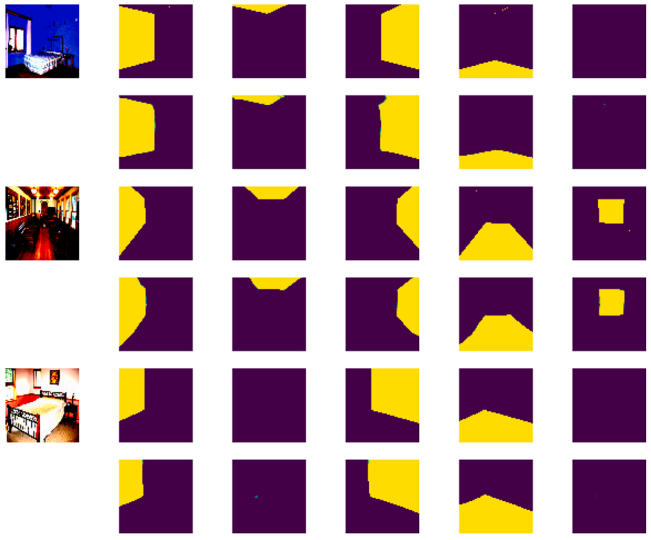


Fig. 14. Semantic segmentation result applying `U-Net + WPP + RC` in the validation dataset.

6.2.3. *Case 3: U-Net + WPP + RC*

Finally, Fig. 14 shows the results when we apply our proposal to all three. Further to the improved results obtained in the Case when the metadata is used, we observe a considerable improvement as we make use of the edge refinement step. The proposed step does not only have a positive impact on identifying the absence and presence of the background wall in the first and second images, but more importantly, it clearly identifies the layout of the room of the third image. As seen in Figs. 3 and 13, the `U-Net` and `U-Net + WPP` models have failed to identify the absence of the ceiling in the third image, the one corresponding to the second channel. By incorporating the edge refinement processing to the structure of `U-Net + WPP + RC`, our proposal is able to identify the room layouts of all the three images.

7. Conclusions and future plans

Having identified the shortcoming of the U-Net, we first undertook the analysis behind its poor performance. Towards this end, we first examine the main reasons why the model did not meet expectations. We then design, implement and evaluate two alternative solutions, namely

U-Net + PP and U-Net + WPP + RC.

Due to the nature, architecture elements and goal of our semantic segmentation proposals, we have carefully selected the metrics providing us an overall evaluation of the learning process, added-value of our proposals and computational and energy requirements.

Our results have shown that the performance of the neural network architecture U-Net can be considerably improved by preprocessing of the metadata generated in the training phase. In addition, and quite relevant to our study, we have found that: (i) a neural network that classifies data into independent channels will improve its estimates if these channels uniquely order the objective information; and (ii) the estimation of a particular channel within a segmented system will optimize its sharpness of edge definition if a refinement is applied to the final product during the training phase. Furthermore, our results have shown, using three different images, that this latter refinement may have a positive impact over the results obtained in other channels leading to an overall improvement, i.e., a more accurate room layout identification.

Finally, an objective that remains pending for future research will be to consider stress tests applied to a continuous image stream. In fact, our work in this area has been motivated by the high usage rate of video material.

CRediT authorship contribution statement

Luis Vasquez-Espinoza: Conceptualization, Data curation, Formal analysis, Writing - original draft, Writing - review & editing. **Manuel Castillo-Cara:** Conceptualization, Data curation, Formal analysis, Writing - review & editing. **Luis Orozco-Barbosa:** Conceptualization, Formal analysis, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work has been partially funded by the Spanish Ministry of Science, Education and Universities, the European Regional DevelopmentFund and the State Research Agency [grant number RTI2018-098156-B-C52], and by FONDECYT / World Bank [grant number 026-2019FONDECYT-BM-INC.INV].

References

Armeni, I., Sener, O., Zamir, A. R., Jiang, H., Brilakis, I., Fischer, M., & Savarese, S. (2016). 3D semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1534–1543).

Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence, 39*, 2481–2495.

Cruz, E., Rangel, J. C., Gomez-Donoso, F., & Cazorla, M. (2019). How to add new knowledge to already trained deep learning models applied to semantic localization. *Applied Intelligence*. https://doi.org/10.1007/s10489-019-01517-1

Csurka, G., Larlus, D., & Perronnin, F., (2013) What is a good evaluation measure for semantic segmentation? In British Machine Vision Conference, BMVC 2013, Bristol, UK, September 9–13, 2013.

Dasgupta, S., Fang, K., Chen, K., & Savarese, S. (2016). Delay: Robust spatial layout estimation for cluttered indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 616–624).

Diakogiannis, F. I., Waldner, F., Caccetta, P., & Wu, C. (2020). Resunet-a: A deep learning framework for semantic segmentation of remotely sensed data. *ISPRS Journal of Photogrammetry and Remote Sensing, 162*, 94–114. https://doi.org/10.1016/j.isprsjprs.2020.01.013

Dice, L.R., (1916) Distribution of the land vertebrates of southeastern washington, by lee raymond dice. DOI: 10.5962/bhl.title.24150.

Ding, H., Jiang, X., Shuai, B., Liu, A. Q., & Wang, G. (2020). Semantic segmentation with context encoding and multi-path decoding. *IEEE Transactions on Image Processing, 29*, 3520–3533. https://doi.org/10.1109/TIP.2019.2962685

Feng, D., Haase-Schütz, C., Rosenbaum, L., Hertlein, H., Gläser, C., Timm, F., Wiesbeck, W., & Dietmayer, K. (2020). Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges. *IEEE Transactions on Intelligent Transportation Systems*, 1–20. https://doi.org/10.1109/TITS.2020.2972974

Fu, J., Liu, J., Li, Y., Bao, Y., Yan, W., Fang, Z., & Lu, H. (2020). Contextual deconvolution network for semantic segmentation. *Pattern Recognition, 101*, Article 107152. https://doi.org/10.1016/j.patcog.2019.107152

Garcia-Garcia, A., Orts-Escolano, S., Oprea, S., Villena-Martinez, V., & Garcia-Rodriguez, J., (2017). A review on deep learning techniques applied to semantic segmentation. arXiv preprint arXiv:1704.06857.

Garcia-Garcia, A., Orts-Escolano, S., Oprea, S., Villena-Martinez, V., Martinez-Gonzalez, P., & Garcia-Rodriguez, J. (2018). A survey on deep learning techniques for image and video semantic segmentation. *Applied Soft Computing, 70*, 41–65.

Hung, W.C., Tsai, Y.H., Liou, Y.T., Lin, Y.Y., & Yang, M.H., (2018). Adversarial learning for semi-supervised semantic segmentation. arXiv:1802.07934.

Lee, C. Y., Badrinarayanan, V., Malisiewicz, T., & Rabinovich, A. (2017). Roomnet: End-to-end room layout estimation. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 4865–4874).

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3431–3440).

Lovón-Melgarejo, J., Castillo-Cara, M., Huarcaya-Canal, O., Orozco-Barbosa, L., & García-Varea, I. (2019). Comparative study of supervised learning and metaheuristic algorithms for the development of bluetooth-based indoor localization mechanisms. *IEEE Access, 7*, 26123–26135. https://doi.org/10.1109/ACCESS.2019.2899736

Martinez-Gomez, J., Martinez del Horno, M., Castillo-Cara, M., Brea Lujan, V. M., Orozco-Barbosa, L., & Garcia-Varea, I. (2016). Spatial statistical analysis for the design of indoor particle-filter-based localization mechanisms. *International Journal of Distributed Sensor Networks, 12*.

Nikoohemat, S., Peter, M., Oude Elberink, S., & Vosselman, G. (2018). Semantic interpretation of mobile laser scanner point clouds in indoor scenes using trajectories. *Remote Sensing, 10*. https://doi.org/10.3390/rs10111754

Pham, Q.H., Hua, B.S., Nguyen, T., & Yeung, S.K., (2019). Real-time progressive 3d semantic segmentation for indoor scenes. 2019 IEEE Winter Conference on Applications of Computer Vision (WACV) 10.1109/wacv.2019.00121.

Romero-González, C., Martínez-Gómez, J., García-Varea, I., & Rodriguez-Ruiz, L. (2017). On robot indoor scene classification based on descriptor quality and efficiency. *Expert Systems with Applications, 79*, 181–193.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *International Conference on Medical image computing and computer-assisted intervention, Springer.*, 234–241.

Sherrah, J., 2016. Fully convolutional networks for dense semantic labelling of high-resolution aerial imagery. CoRR abs/1606.02585. arXiv:1606.02585.

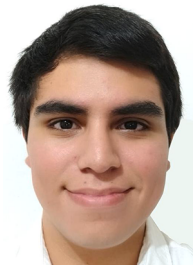
Souly, N., Spampinato, C., & Shah, M., (2017). Semi and weakly supervised semantic segmentation using generative adversarial network. CoRR abs/1703.09695.

Wang, Q., Gao, J., & Yuan, Y. (2018). Embedding structured contour and location prior in siamesed fully convolutional networks for road detection. *IEEE Transactions on Intelligent Transportation Systems, 19*, 230–241. https://doi.org/10.1109/TITS.2017.2749964

Yi, Z., Chang, T., Li, S., Liu, R., Zhang, J., & Hao, A. (2019). Scene-aware deep networks for semantic segmentation of images. *IEEE Access, 7*, 69184–69193. https://doi.org/10.1109/ACCESS.2019.2918700

Yu, F., Zhang, Y., Song, S., Seff, A., & Xiao, J., (2015). LSUN: construction of a large-scale image dataset using deep learning with humans in the loop. CoRR abs/1506.03365. arXiv:1506.03365.

Zhang, Y., Mehta, S., & Caspi, A., (2021). Rethinking semantic segmentation evaluation for explainability and model selection. CoRR abs/2101.08418.



Luis Vasquez-Espinoza, graduated of Computer Science program at Universidad Nacional de Ingenieria (Peru), has been working as a senior investigator at the computing specialized scientific association (ACECOM-UNI) and at Intelligent Ubiquitous Technologies – Smart City (IUT-SCi) Lab. His current research interests include optimization of deep learning architectures and evaluating the similarities with the behavior of the human brain.

Manuel Castillo-Cara received the PhD degree from the University of Castilla-La Mancha (Spain) in July 2018. He has been working on university educational issues at the Computer Science as an Associate Professor and head of Intelligent Ubiquitous Technologies – Smart City (IUT-SCi) Lab at Universidad de Lima (Peru). His current research is focused on Intelligent Ubiquitous Technologies, especially on in Wireless Sensor Networks, Distributed Computing, Pattern Recognition and Artificial Intelligence.

Luis Orozco-Barbosa received the Doctorat de l'Université from the Université e Pierre et Marie Curie, France, in 1987. From 1987 to 2001, he was a faculty member at the Electrical and Computer Engineering Department of the University of Ottawa, Canada. In 2002, he joined the Department of Computer Engineering at the Universidad de Castilla-La Mancha (Spain). His current research interests include Internet protocols, wireless sensor communications, and IoT technologies. He is a member of the IEEE.