A System for Furniture Analysis and Home Scenes Recognition

Cartella Giuseppe, Marchesini Kevin, Sarto Sara
University of Modena and Reggio Emilia
{240498, 249277, 237128}@studenti.unimore.it

Abstract

We present a system for analysing indoor home scenes. The aim of our project is to propose a furniture retrieval system based on the segmentation and extraction of objects present in the environment, followed by a room classification step. To perform these tasks we compared different approaches and algorithms, trying to reach the best possible results.

1. Introduction

We developed a software that, taken an image of a room of the house, analyzes the environment and suggests similar furniture to those present in our dataset that has been created for this purpose. In particular the main tasks are:

1. **Instance Segmentation**: We used an instance segmentation network to detect and segment each object present in the room. We will also discuss about our custom implementation of the architecture in order to improve mask details.

2. **Furniture Retrieval**: Given the bounding boxes from the previous step, they are used as query images and then the most similar objects are retrieved.

3. **Room Classification**: Once we have found all furniture in the room we classify the environment on the basis of a list of rooms that we have decided to focus on.

Instance segmentation (Figure 1) is definitely the most important task. Indeed a bad furniture extraction can seriously affects our retrieval and classification systems.

In order to learn as much as possible from this project, we used different methods illustrated both during lectures and in literature, comparing all results to find the best solution.

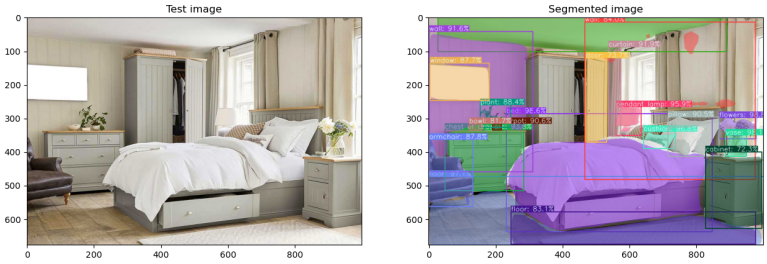


Figure 1: *Instance Segmentation result. Each detected object has its own mask, bounding box and class name.*

1.1. Related Work

Our system must perform an optimal instance segmentation task since we want to obtain good results for the next pipeline steps. In literature, a lot of different approaches have been presented. We decided to focus on Mask R-CNN as base architecture, which represents a very effective instance segmentation approach. Mask branches have a straightforward structure but more complex designs have the potential to improve performance [2].

In the image retrieval part different solutions can be compared, from more classical techniques as SIFT [5], to neural networks, in particular convolutional autoencoder [3], [7], and DHash algorithm. Related to this last part sometimes it is useful to do background suppression, with some methods as Grabcut [6], in order to have better performance. GrabCut [6] is a useful method that we used in order to have retrieval results

that are not influenced by the environment or by other objects nearby. Moreover, we decided to combine it with the results of the mask to be more accurate.

Regarding the classification part it is important to focus on furniture that allows one room to be discriminated from another one, depending on the different degree of importance that shall be attributed to it. A simple but effective model is Random Forest.

2. Datasets

2.1. ADE20K for instance segmentation

As our project aims to segment specific images concerning indoor home scenes, we cannot use a network fully pretrained for instance on COCO or ImageNet, because these datasets are too general and do not contain some important object classes that are essential for our purpose. For this reason, we decided to use a fully annotated dataset called *ADE20K* [8]. It contains both indoor and outdoor scenes for a total of 27.574 images. Therefore, we discarded useless images and we chose 10 different scenes such as bathroom, bedroom, children room, dining room, garage, home office, kitchen, living room, nursery and utility room.

We ended up with 5297 examples (Figure 2) and, as a matter of fact, we only considered objects included in these images. In the code we provide some useful scripts for *ADE20K* filtering.

Figure 2: *Some examples of ADE20K images*

2.2. Retrieval dataset

For the retrieval part of the pipeline we need single objects, therefore, we downloaded them from Google by using a Chrome extension called *Imageye*, which allows a wide range of images to be downloaded from a website. In order to get a larger dataset, mainly needed for the training of the autoencoder, we merged the images with a useful dataset available on Kaggle.

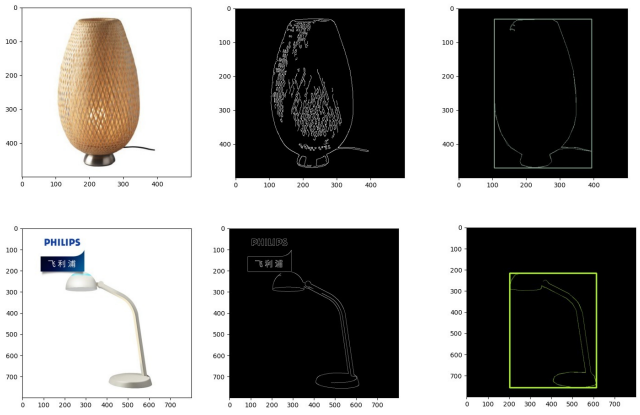
Final dataset contains 1938 images (Figure 3). We decided to focus on 6 type of objects: armchair, bed, bicycle, chair, sofa and lamp. We created a dataset annotation file containing the name of the image, its label and the coordinates of the object bounding box (further details in 2.2.1).

Figure 3: *Some examples of retrieval dataset and application of GrabCut*

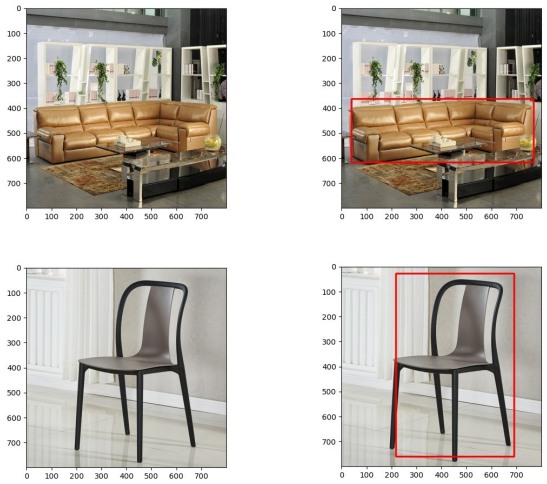
2.2.1 Foreground Extraction

For each image (with known label) of the retrieval dataset our aim is to extract the exact object excluding everything around that can influence the final result. This is a hard task because we don't know the exact position of the object and the application of Grabcut

method on the entire image may not give the desired results. So, in order to obtain the exact bounding box of the object, both for writing it on the annotation file of the retrieval dataset and for applying Grabcut on a more specific area, we defined a particular approach.



(a) *Not COCO class (lamp) images and application of Canny algorithm. For every found contour in the image an approximation is applied to polygons and stating that the curve must be closed. The last image is the bounding box obtained from the polygon that has the biggest area.*



(b) *COCO class (sofa and chair) with bounding boxes found by Mask R-CNN*

Figure 4: *Example of localization of object's bounding box.*

In particular, if the category of the main furniture present in the image is a COCO dataset category, the bounding box to insert in the annotation file is found with a pre-trained Mask R-CNN (Figure 4b), otherwise the bounding box is found using Canny edge detector (Figure 4a). In particular, with Canny, we find the main contours, we approximate them with polygons which enclosed in a bounding box, and then we take the bounding box with the maximum area. If this area is big enough, the image is acceptable. In this way we can also avoid possible unwanted labels present in the image during the download phase. Moreover, this method represents a feasible idea because almost all the images in our retrieval dataset have exactly one main subject, otherwise it would be impossible to extract the exact bounding box of the furniture.

3. The approach

As we mentioned before, our project is divided into three main tasks. First, in order to retrieve all objects present in the image we focused on a solution that is entirely based on deep learning. We propose a comparison between the default version of the Mask-RCNN and a custom implementation, applying small changes to the network structure. We have decided to use Mask R-CNN [2] not only because it is one of the most used neural network in this case, but also because, since it is based on Faster-R-CNN [1], it is already pre-trained on COCO dataset [4]. This last detail is significant especially in order to get quite a good performance in a short time.

Regarding the retrieval part, we developed three different ideas which exploit different concepts and techniques. We started with a more classical approach using SIFT [5], comparing the feature descriptor of the query object with the ones in the dataset. Moreover, we experimented a solution based on deep learning making use of a convolutional autoencoder, taking for each query the latent vector. Lastly, application of DHash algorithm represents our third idea. It is a method that generates an hash for each image and returns the most similar results considering the Hamming Distance.

For classification a lot of methods could have been used. We implemented a simple Random Forest classifier which models in a very effective manner the feature importance that in our case is represented by the presence or absence of a specific object in the room. Then, we compared this method with a simple MLP model.

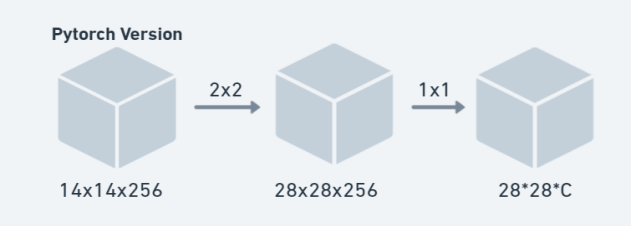


Figure 5: *Original Mask R-CNN PyTorch Version.*

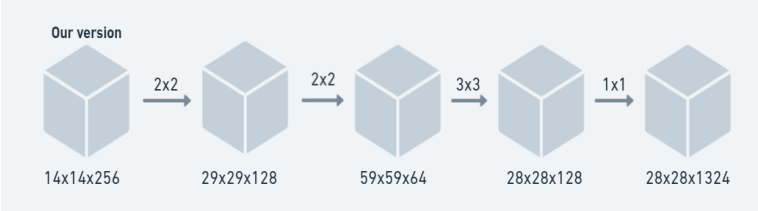


Figure 6: *Modified Mask R-CNN with changes only in the Mask Predictor part. All convs are 2x2, transpose conv is 3x3, except the output conv which is 1x1. All the kernels have dilation and stride 2, and in the hidden layer we use ReLu.*

3.1. Instance Segmentation

In order to carry out a good image retrieval system and classify the room, first of all we need to localize and identify the most significant objects. These tasks have to be performed all together, so we decided to focus on Mask R-CNN, which also provides information about the mask, which will be very helpful during retrieval. At the beginning we adopted a partial pre-trained version available on PyTorch. We say 'partial', because, due to the problems already described in 2.1, we retrained object detector and mask predictor parts on ADE20K, focusing on all the objects present in the last filtered version of the dataset. However, backbone has not been re-trained.

Our main concern is related to the mask output because we will pass it to the next step, for this reason, we compared default version (Figure 5) with our custom implementation (Figure 6) where we tried to slightly change the mask predictor architecture, adding a transpose convolutional layer and the corresponding convolutional one. In order to get a bigger receptive field, dilation is set equal to 2. For computational reasons the number of channels has been reduced compared to [2] where all the channels are 256. We started from 128 channels and after transpose convolution we ended up to 64 channels. Then back to 128 with convolutional layer ending with a number of channel equal to the number of object classes, 1324 in our case. The kernel size and the stride have been changed in the layers in order to get dimension in the first layer of 29x29, in the middle layers of 59x59 and then get back to 28x28 as in the original architecture (Figure 6).

3.2. Furniture Retrieval

By taking into consideration results of the previous step (bounding box, class name and mask for each found object), now two parallel pipelines start: one for furniture retrieval and the other one for room classification.

As for the image retrieval task, we focused on the development of different methods: SIFT [5], DHash and convolutional autoencoder (Figure 9). For all the three methods the query is compared with each object of the given retrieval dataset.

Before applying any retrieval method on the bounding box of the object, a bilater filter is applied in order to smooth the noise and to enhance its edges.

3.2.1 SIFT

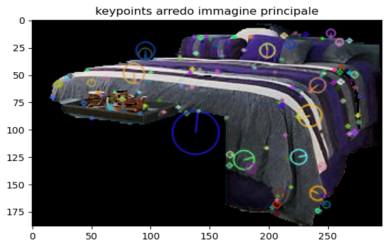


Figure 7: *Sift example*

Our first approach for image retrieval is based on SIFT (Scale-Invariant Feature Transform) method, a feature detection algorithm used in computer vision to detect and describe local feature in images (Figure 7). First of all, we pre-computed all key-points and descriptors of the objects in the retrieval dataset, using the available OpenCV function *sift.detectAndCompute()*. Lastly, in order for the same operation to not be carried out once again, we saved the descriptors in memory. The same is done for each

query image, that in our case is the bounding box extracted by the network that represents one of the retrieval objects we decided to focus on.

Matches beetween feature descriptors are defined using a Brute Force Matcher and then applying KNNMatch. The 5 images with minimum distance from the query are returned.

3.2.2 Convolutional Autoencoder

Convolutional Autoencoder can be really useful when it is a matter of Content-based Image Retrieval (CBIR). The structure is composed of an encoder that tries to increase the receptive field and extract a compressed low dimensional representation of the input, while the decoder part tries to reconstruct it.

Two main steps are required for the proposed Content-based Image Retrieval method. First of all, feature extraction to get representative features, and second of all similarity calculation between query and the other dataset objects.

After the training of the autoencoder, similarly to what we have done with SIFT, we saved an embedding of every object in the dataset.

When retrieval must be performed, the bounding box (got from the previous instance segmentation) is passed through the encoder obtaining a latent vector that must be compared with all the saved ones of the dataset, considering that similar objects should be represented by a similar embedding.

At the end, the 5 most similar images are retrieved using KNN.

The autoencoder architecture is composed by 5 convolutional layers in the encoder and 5 transpose convolutional ones in the decoder part, starting from a image 3x224x224, getting in the latent space in a dimension 256x7x7 and then getting back to 3x224x244 (Figure 9). The encoder has all the filters of dimension 3x3 and to perform the down-sampling part we use stride equal to 2, while doubling the number of channels. The decoder, on the other side, tries to recover the original image using filter 2x2 and increasing the number of channels. Some results can be seen in Figure 8.

3.2.3 DHash Algorithm

The idea of this algorithm is to remove high frequencies and details shrinking the image to 9x8, so with a total number of 72 pixels. The small image, then, is converted to gray-scale for colour reduction. After reducing the scale and the colour of the image, the algorithm computes the difference between adjacent pixels in order to find the gradient direction and, based on that result an hash code is generated for each image. This process enable us to obtain similar hash values even if the image is scaled or the aspect ratio changes, or if the brightness or contrast of the image is altered.

In order to get similar images to the query one, the hash values of all the images in our dataset of single objects are compared to the query one just computing the Hamming distance (counting the number of bits that are different). In our case we kept, amongst all the images of the correct class, the first 5 similar objects found.

Figure 8: *The highlighted images represents queries, next to them the retrieved images by the convolutional autoencoder found within the entire retrieval dataset*

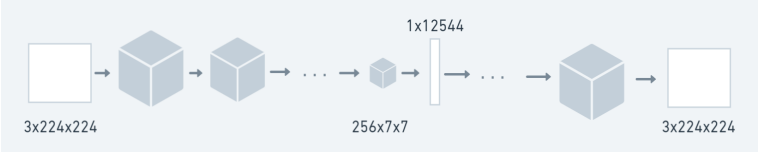


Figure 9: *Convolutional Autoencoder Architecture. The input image and the reconstructed output are 3x224x224. The latent vector dimension is 1x12544. Each conv is 3x3, while in the up-sampling 2x2. The encoder channels are doubled each time 16,32,64,128 and 256 (viceversa for the decoder).*

3.3. Room Classification

In this section we describe the other parallel pipeline that performs classification. The final classification result is entirely based on the objects detected by the network. Therefore, getting a network with a good detection accuracy is extremely important because, otherwise, bad results can negatively affect the classification.

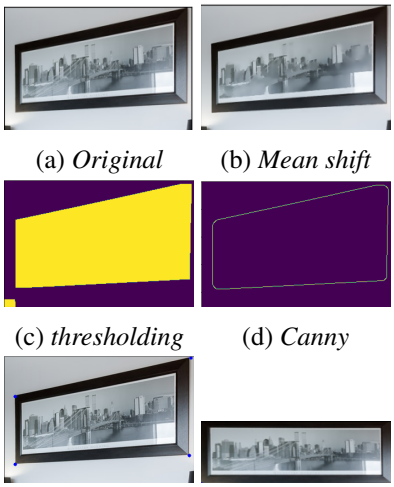
In order to classify the right room among the 10 scenes that we have selected, two approaches are proposed: one based on an ensemble machine learning algorithm and the other on deep learning.

For the first one a Random Forest Classifier is trained using a matrix *IxN* extracted from *ADE20K* dataset where *I* represents all the different images and *N* represents all the possible objects. In each intersection there is a 1 if the object is present in the image, 0 otherwise.

$$matrix[i][j] = \begin{cases} 1, & \text{if the object } j \text{ is in the room i} \\ 0, & \text{otherwise} \end{cases}$$

For the second approach, the same matrix is passed through three fully connected layers.

4. Rectification



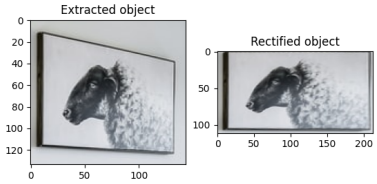
(a) *Original* (b) *Mean shift*

(c) *thresholding* (d) *Canny*

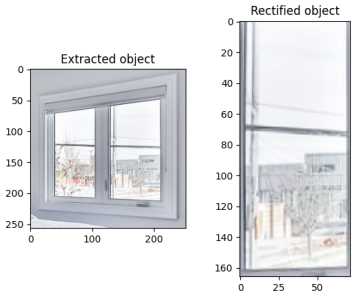
(e) *Contours* (f) *Result*

Figure 10: *Steps for rectification*

In addition to all the tasks previously described, we integrated in our system a rectification step focusing on windows, doors and pictures detected in the image. As for retrieval, the input of this process is the bounding box. The method involves image processing, contours and corners detection and perspective transformation.



(a) *This image represents a good rectification result of a painting. This is an easy task because of the frame of the painting that helps finding the four key-points.*



(b) *The window is a difficult example, and in fact the result is not very accurate, due to the complexity of the image in which it is difficult to differentiate between background and foreground and finding the key-points.*

Figure 11: *Examples of rectification.*

4.1. Image Processing for Rectification

First of all, we applied mean shift filter to the image (Figure 10b), so that edges might be easier detected, followed by a binary thresholding (Figure 10c). Then, we use a series of image operators to close the object contours and to obtain the inverse image. At this point we find the potential contours of the entire object, choosing the boundary enclosing the major area (so the most probable candidate to be the object we want to rectify). Then we blur the image with a me-

dian filter, and apply Canny algorithm to find edges (Figure 10d). From this set of edges the Hough transform is able to find the ones that form straight lines: intersecting these lines we can easily find the corners. Finally we select the 4 more likely corners using the K-means, which simply groups them based on their spatial position (Figure 10e).

4.2. Perspective transformation

At this point, with the 4 corners, we compute the new image dimensions, preserving the aspect ratio, and we obtain the transformation matrix and perform the perspective transformation (Figure 10f). The result is the so-called "birds-eye view" of our image, which has the property to have the object at the center of the image with the contours rectified.

The method is pretty good with pictures and doors, instead it encounters more difficulties with windows, because sometimes the main corners are not founded in the right position but in correspondence of a part of glass of the window (Figure 11). One last thing to point out is that, unfortunately, this approach is not applicable on generic furniture. In fact with irregular shapes, lines and corners can't be easily detected.

5. Experimental Evaluation

5.1. Instance Segmentation

Regarding the instance segmentation part of the pipeline, both the default Mask R-CNN architecture and our modified have been trained for 10 epoches reaching similar results in term of mAP, even if after our changes the mAP gets slightly better (see Table 1). The main difference during training is the time that it took: for the default version around 6 hours, while for the modified one almost an entire day. The main explanation is that, although it has been used dilation equal to 2, we inserted a convolutional layer and its corresponding transpose layer increasing the parameters and the number of operations.

Architecture	mAP	
	Default	Modified
Mask R-CNN	57.5%	61%

Table 1: *Mask R-CNN mAP Comparison*

Figure 12: *Comparison between Default Mask R-CNN and the Modified one, respectively. Some bounding boxes and masks with the modified version are more precise, moreover the Default Mask R-CNN misses an entire object, the painting, while the Modified create a good bounding box.*

5.2. Furniture Retrieval

After changing part of the Mask R-CNN architecture the performances in the retrieval task actually get better, especially in the autoencoder (Table 2). This is due to the fact that in the retrieval part of the pipeline Grabcut is used and as input it receives the details of the mask from the neural network, therefore the more accurate the mask is the more Grabcut will be precise and so the retrieval.

Even for SIFT and Dhash, having a more precise mask helps isolating the object and getting a better performance (Table 3).

5.2.1 SIFT

SIFT computation is performed only on images of the same class of the input object, so for example we compare bed only with other beds of our retrieval dataset, considering the 5 most similar. We have made this choice because SIFT used only with a simple distance is not good enough to discriminate always objects belonging two different classes, because they could have in any case some similar characteristics. The algorithm uses the nearest-neighbour (NN) distance.

The results are shown in Table 3.

5.2.2 Convolutional Autoencoder

The convolutional autoencoder was trained for 50 epochs on the dataset of single objects (1938 images), using to compare the input image and the reconstructed one the Mean Square Error loss.

During test, when an object is found by the instance segmentation neural network, it is passed through the encoder in order to get its latent vector. This last one will be compared with the latent vectors of all the dataset, not only with the images with the same label of the query, and for each query image the 5 most similar images are retrieved thanks to the Nearest Neighbours algorithm.

We decided to compare the retrieval results obtained from the basic mask and from the modified one. The results can be seen in Table 2.

Autoencoder based method was the best in terms of mean average precision (mAP).

Method	mAP	
	Default	Modified
Autoencoder	54.4%	66.3%

Table 2: *Autoencoder Retrieval Comparison*

5.2.3 DHash

In regards to retrieval evaluation using DHash, the most important thing to notice is that the algorithm works great for detecting exact duplicates or near duplicates, for example the same object slightly altered. For this reason, in our case it works well with chairs or sofas where the shape does not drastically change, while on more particular objects like lamps, that can have different shapes, colours and details the accuracy decreases. In this case, as for SIFT, we decided to only retrieve images belonging to the same label class. The results can be seen in Table 3.

Methods	mAP	
	Default	Modified
SIFT	57%	57.8%
DHash	54.9%	55.2%

Table 3: *Retrieval Comparison*

5.3. Room Classification

To evaluate the two proposed methods we simply computed the accuracy value on a test set of images for

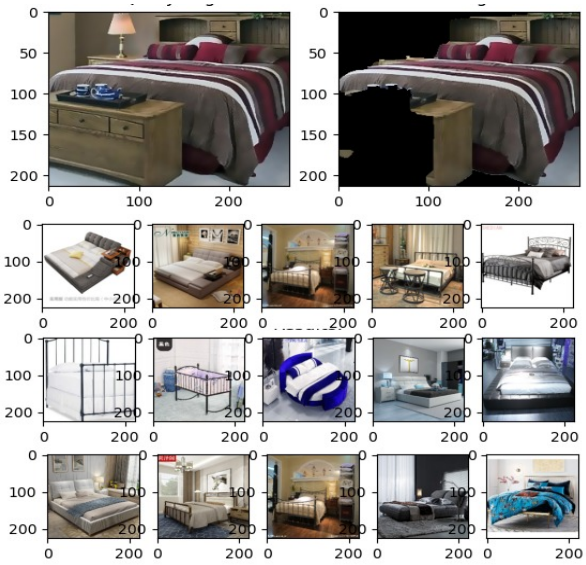


Figure 13: *Comparison between retrieval methods: the upper image is the query, below there are the retrieved images by different retrieval methods. The first one is the result from autoencoder (3.2.2), the middle one from SIFT (3.2.1) and the last one from DHash (3.2.3)*

both the Random Forest and the fully connected. At the beginning we started classifying a room just considering 101 objects (plus the background), the ones that for us were the more discriminant and the more useful in recognizing the room represented in the image. Then, we noticed that the classification was not always so accurate, so we decided to classify using all the objects annotated in the ADE20K dataset. The results in Table 4 show us that in this way accuracy increases.

For Random Forest classifier we tried to find the best parameters applying GridSearch that exhaustively considers all parameter combinations. The best parameters in our case are: {'criterion': 'gini', 'max_depth': 60, 'max_features': 'log2', 'n_estimators': 100}.

For the fully connected version, it is definitely very important the choice of the optimizer, indeed in both cases using Adam gives us better performances than working with SGD. For the Random Forest classifier, on the other side, using GridSearch to get the best parameters actually did not change the results very much.

What actually increased our accuracy in both methods was making the dataset balanced and classifying

with a greater number of objects.

Despite the differences characterising the two used method, what it is important to highlight is that, both with few objects and all, the Random Forest classifier wins over the fully connected methods, especially when there are few objects.

Method	Random Forest		Fully Connected	
	Default	GridSearch	Adam	SGD
101 objs	96.2%	96.4%	92.2%	87.5%
all objs	98.6%	**98.7%**	95.8%	90.0%

Table 4: *Comparison between classification methods*

6. Conclusion

In this paper we propose different approaches for recognizing indoor scenes, combining traditional computer vision algorithms and more novel techniques such as neural networks. At the end we achieved satisfying results in most of the tasks, even when we tried to make changes or combine together different techniques. In particular, in our pipeline the best solutions for each part are: our modified version of Mask R-CNN for instance segmentation, the convolutional autoencoder for retrieval and the Random Forest for classification.

Code is publicly available at: `https://github.com/GiuseppeCartella/Indoor-Scene-Understanding`.

References

[1] Ross Girshick. "Fast R-CNN". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017).

[2] Gkioxari He, Dollar, and Girshick. "Mask R-CNN". In: *IEEE International Conference on Computer Vision (ICCV)* (2017).

[3] Alex Krizhevsky and Geoffrey E. Hinton. "Using Very Deep Autoencoders for Content-Based Image Retrieval". In: *ESANN* (2011).

[4] Tsung-Yi Lin et al. "Microsoft COCO: Common Objects in Context". In: *IEEE International Conference on Computer Vision (ICCV)* (2014).

[5] David G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoin". In: *International Journal of Computer Vision* (2004).

[6] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "GrabCut — Interactive Foreground Extraction using Iterated Graph Cuts". In: *ACM Transactions on Graphics* (2004).

[7] Indah Agustien Siradjuddin, Wrida Adi Wardana, and Mochammad Kautsar Sophan. "Feature Extraction using Self-Supervised Convolutional Autoencoder for Content based Image Retrieval". In: *2019 3rd International Conference on Informatics and Computational Sciences (ICICoS)* (2019).

[8] Bolei Zhou, Hang Zhao, and Xavier Puig. "Scene Parsing through ADE20K Dataset". In: *Computer Vision and Pattern Recognition (CVPR)* (2017).