Door Finder and Buildings Classifier

An application of Computer Vision techniques for building classification and door detection

Emanuele Bianchi
Università di Modena e Reggio Emilia
Via Università, 4, 41121 Modena MO
274208@studenti.unimore.it

Benedetta Fabrizi
Università di Modena e Reggio Emilia
Via Università, 4, 41121 Modena MO
280615@studenti.unimore.it

Gabriele Rosati
Università di Modena e Reggio Emilia
Via Università, 4, 41121 Modena MO
225817@studenti.unimore.it

Abstract

Purpose of this work is the development of a framework for buildings recognition and the detection of their entrance. To this end, two types of neural networks are used: one for the classification of buildings and one for door detection. As regards the classification task, a comparison with other approaches based on image retrieval was also made, always with the goal of being able to classify the building, making use of image processing and, therefore, applying filters and image transformations.

1. Introduction

The use of neural networks can improve everyday life quality of everyone, but mostly important they can be used for helping people with particular deseas. For example, people with reduced mobility that are forced to use wheelchairs can find several problems when entering in buildings. In this context, neural networks for image recognition can be exploited for the development of a smart framework able to recognize buildings and detect doors so that the wheelchair user can easily enter in a building.

2. Buildings Classification

In this Section we describe the dataseta and the chosen Artificial Neural Network for the classification of buildings. Finally, a description of the tests run for choosing the best model are reported.

2.1. Dataset for ResNet

Since the impossibility to find a dataset about buildings that fit the project requirements, we built our personal dataset by picking the images from different stock websites. Then, we have arranged them in five different folders, corresponding to the classes defined in the project: *house*, *flat complex*, *church*, *historical buildings/monument*, *shops*.

After having filled the dataset, we cleaned it from those images not containing useful information. The resulting dataset is balanced, with each class approximately having the same number of photos. Since we built or own dataset, it is not very large. Indeed, the total amount of images is 1300.

During classification, both in training phase and test phase, the artificial neural network will read directly the folders name and take them as classes name.

2.2. Choice of model

For classification purpose, we made use of Residual Network, or **ResNet**[1], which is very popular for image classification. In order to make it compatible with our final goal, we modified the last layer before the fully connected one, also providing it with an additional output: therefore our network returns a matrix $[1, 5]$, where 5 is the number of classes, and the feature vector that we will use in section 4. Furthermore, we did a **fine-tuning** freezing the first layers and retraining just the last 19 layers.

Since ResNet architecture may vary in quantity of layers, we made some tests to find the best model to fit our data. To do so, we used the **early stopping** technique, plotting both train and validation loss and saving the model with the lower loss value. At first we set up a *patience* of 5 epochs, afterwards we increased it to 20 if the analysis of the curve showed that the model was not overfitting and had a decreasing trend.

Figure 1 shows an example of early stopping method with 5 epoches patience in which the validation loss has a downward trend, whereby the patience could be raised to 20.

Figure 2 shows an experiment in which the validation loss doesn't seem to be improvable.

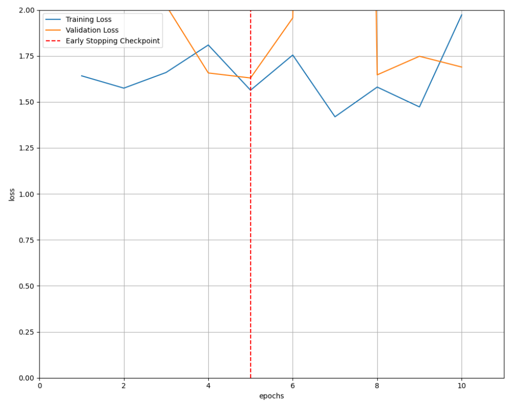

Figure 1. The trend of validation loss is downward. In cases like that, the patience was increased to 20 and the experiment was repeated.

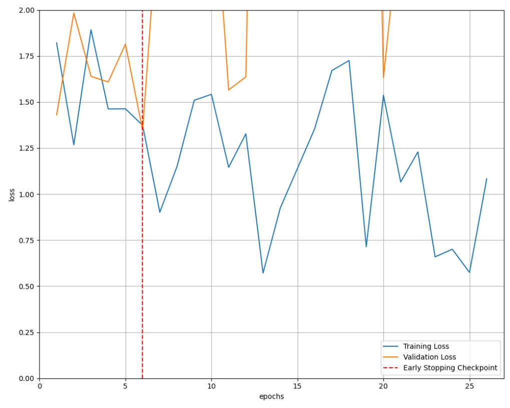

Figure 2. The validation loss is too high and doesn't seem to be improvable, so we could discard the model.

Initially we tried using ResNet152, with a learning rate of 0.001, 0.005, 0.1, 0.2, and 0.5. We were able to see better performances with the first two values, accordingly we decided to apply those learning rates to smaller ResNet, such as ResNet18, ResNet34 and ResNet50. Finally we tried to optimize the neural networks that gave the best two models by applying the **Stochastic Gradient Descent**, while in the first tests we made use of **Adam** method. Table 1 shows the results we achieved at this stage.

As shown here, the best result was achieved with a ResNet34 with 0.005 learning rate value and SGD method. Figure 3 shows the early stopping plot of said model.

ResNet	Learning Rate	Accuracy	Loss	Optimizer
ResNet152	0.001	0.37	1.43	Adam
ResNet152	0.005	0.35	1.39	Adam
ResNet152	0.1	0.26	1.63	Adam
ResNet152	0.2	0.28	1.55	Adam
ResNet152	0.5	0.28	1.55	Adam
ResNet18	0.001	0.38	1.41	Adam
ResNet18	0.005	0.65	0.93	Adam
ResNet34	0.001	0.44	1.31	Adam
ResNet34	0.005	0.49	1.21	Adam
ResNet50	0.001	0.34	1.41	Adam
ResNet50	0.005	0.38	1.38	Adam
ResNet18	0.005	0.77	0.59	SGD
ResNet34	0.005	0.82	0.54	SGD

Table 1. The table shows the results of the experiments on the ResNet. As you can see, ResNet34 with a learning rate value of 0.005 and SGD method achieved the best result.

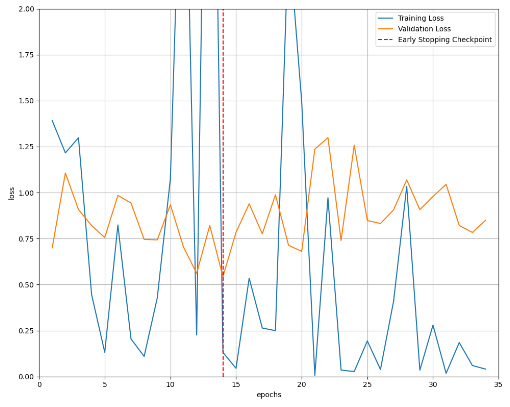

Figure 3. Best result achieved. The red line point to the epoch which get the lower loss value during validation phase

3. Door Detection

3.1. Dataset for Yolo

First we annotated the dataset (the same one built in precedence) identifying the door in each image, to do that we made use of **LabelImg**, a graphical annotation tool. The network was pre-trained on COCO and then fine-tuned using google colaboratory on about a thousand images of doors.

3.2. Detection task

As far as the door detection is concerned we used the **YOLO** network[2], acronym of *You Only Look Once*, in particular YOLOv2, which is an object detection system aimed for real-time processing. YOLOv2 is less accurate than the region proposal based architectures, but faster because it relies on simpler operations.

YOLO divides the input image into an S × S grid. Each cell of the grid:

- has B boundary boxes and each box has a box confidence score, which reflects the probability that the box contains an object (objectivity) and how accurate the boundary box is.
- detects only one object regardless of the number of boxes B
- computes the probabilities of conditional class C (the probability that the detected object belongs to a particular class).

In figure 4 we can see the YOLO model of how the bounding boxes work.

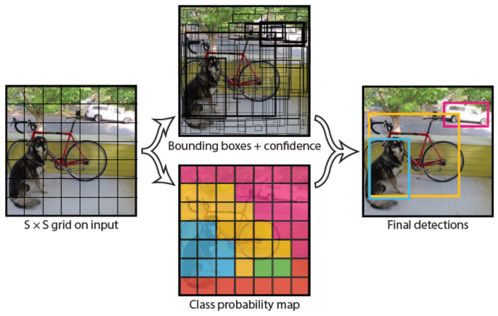

Figure 4. The YOLO Model. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \cdot S \cdot (B \cdot 5 + C)$ tensor.

The class confidence score for each prediction box is computed as follows:

$$CCS = BCS \times CCP \tag{1}$$

where

- CCS = class confidence score
- BCS = box confidence score
- CCP = conditional class probability

measure confidence in both classification and location, which is where an object is located.

$$BCS \equiv P_r(object) \cdot IoU \tag{2}$$

$$CCP \equiv P_r(class_i | object) \tag{3}$$

$$CCS \equiv P_r(class_i \cdot IoU = BCS \times CCP \tag{4}$$

The results obtained are quite satisfactory considering the small size of the dataset that we could have, however the confidence levels could be improved.

4. Image Retrieval

In order to face the Image Retrieval part we tried two possible approaches, which are different in the type of features extracted from images. The descriptors adopted as first solution are the **ORB** (Oriented FAST and rotated BRIEF)[3], while for the second solution the descriptors are the features extracted from the neural network used in classification.

4.1. Dataset for Retrieval

The dataset used for retrieval is composed by the same images of the previous tasks but this time we applied some image processing algorithm over each image: first of all we resized our images to the fixed size of 400×400, then we put them in grey scale, finally we applied an equalized histogram[4] and a bilateral filter[5], whose formula is given by:

$$g(i,j) = \frac{\sum_{k,l} f(k,l) w(i,j,k,l)}{\sum_{k,l} w(i,j,k,l)} \tag{5}$$

$$w(i,j,k,l) = d(i,j,k,l) + r(i,j,k,l) \tag{6}$$

$$d(i,j,k,l) = exp(-\frac{(i-k)^2 + (j-l)^2}{2\sigma_d^2}) \tag{7}$$

$$r(i,j,k,l) = exp(-\frac{||f(i,j) - f(k,l)^2||}{2\sigma_r^2}) \tag{8}$$

where:

- $w(i,j,k,l)$ are the weight
- $d(i,j,k,l)$ is the domain kernel
- $r(i,j,k,l)$ is the range kernel

We used the equalize histogram to improve the contrast in the images, in order to stretch out their intensity range, while we used a bilateral filter in order to suppress the noise.

4.2. Image Retrieval task

In order to extract the features, for this task we modified ResNet. Especially now the net returns features extracted from the last layer before the fully connected one, these features are used as descriptors of the image. In both cases, (ORB descriptor and Neural Network descriptor) as similarity measure, we applied the **MSE** (Mean Square Error) between the query image and the dataset images:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y - f(x))^2 \tag{9}$$

where:

- n is the number of samples
- y is the predicted class
- $f(x)$ is the real class

Before computing a resulting class, there is a query expansion process that consists in some geometric transformations applied to the query image. The geometric transformations are *rotation*, *perspective transformation* and *WARP transformation*.

After that transformation process the average between the features from the original query image and the features that came from transformed images is computed.

Finally we estimated the MSE between the average features and the features extracted by images of the dataset and the output is the class with the smallest value of MSE.

In Figure 5 is possible to see one of best results of two retrieval algorithm.

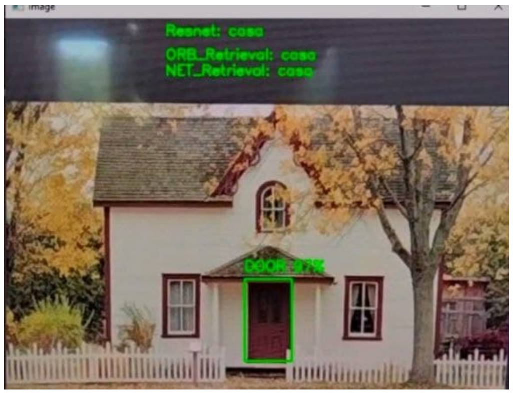

Figure 5. Example of best result of classification and door detection.

5. Results and Conclusions

From the conducted experiments, we can conclude that classification via ResNet34 achieves good performance in testing phase, while the object detection on doors performed by YOLO can still be improved.

For what concern the building recognition, among the three approaches examined, the classification through a neural network proves to be the most satisfactory, since it is also able to give a semantic meaning to the feature it extracts. For the same reason, comparing the other two remaining approaches, the image retrieval carried out by exploiting the feature extracted from the network results more stable then the one carried out via ORB.

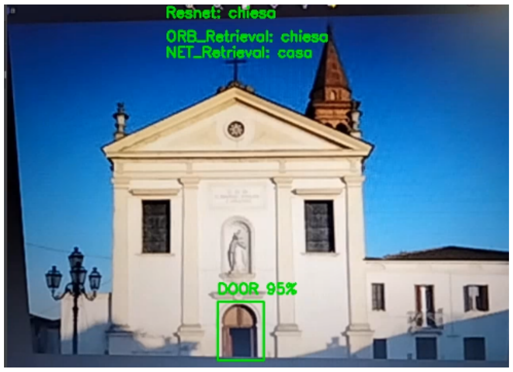

Figure 6. Example of result where image retrieval with network feature does not work.

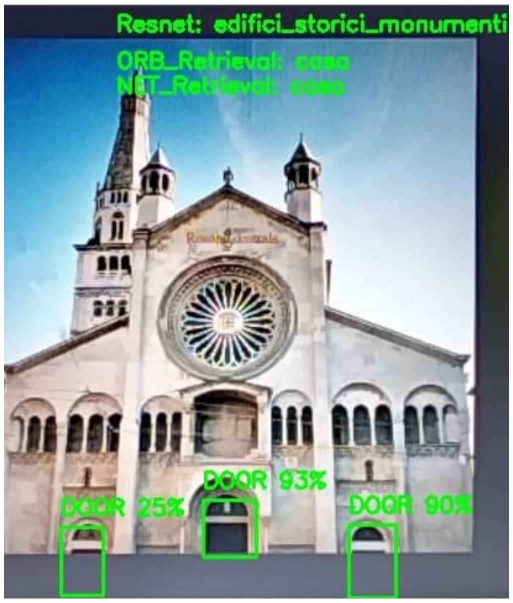

Figure 7. Example of good result on YOLO detection.

In the next pictures we can see different example of results of our system. In particular, in figure 6 the retrieval with network features does not work, while the results are good for ORB, ResNet and YOLO; figure 7 shows a very good result for door detection, while the retrieval is completely wrong. Also, the result of classification should have been *church*, but it is understandable why the network could confuse churches and historical buildings; in figure 8 both classification and retrieval are good, while the door detection has a low percentage, probably because that door is different than a classic door; finally in figure 9 the system cannot detect a door and the retrieval with ORB is wrong, but both classification and retrieval with network feature are good.

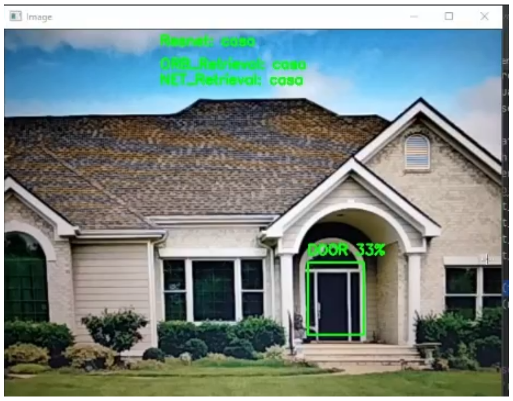

Figure 8. Example of good classification result, but bad door detection.

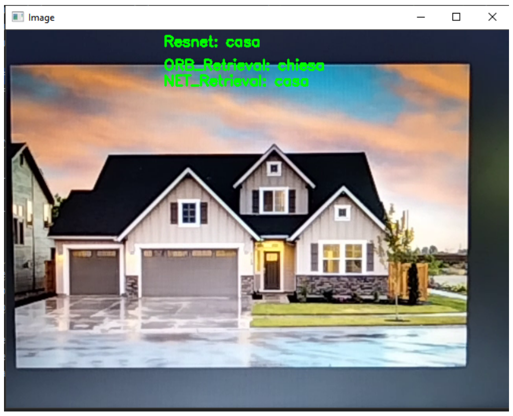

Figure 9. Example of bad retrieval with ORB.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2016.

[3] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International Conference on Computer Vision*, 2011, pp. 2564–2571.

[4] "Histogram equalization," https://docs.opencv.org/3.4/d4/d1b/tutorial_histogram_equalization.html.

[5] R. Szeliski, *Computer Vision: Algorithms and Applications (draft)*, 2010, p. 125–125.