Object Detection with RetinaNet on Aerial Imagery: The Algarve Landscape

C. Coelho(✉), M. Fernanda P. Costa, L. L. Ferrás, and A. J. Soares

Centre of Mathematics, University of Minho, Campus de Gualtar, 4710-057 Braga, Portugal
cmartins@cmat.uminho.pt, {mfc,llima}@math.uminho.pt
https://www.cmat.uminho.pt

Abstract. This work presents a study of the different existing object detection algorithms and the implementation of a Deep Learning model capable of detecting swimming pools from satellite images. In order to obtain the best results for this particular task, the RetinaNet algorithm was chosen. The model was trained using a customised dataset from Kaggle and tested with a newly developed dataset containing aerial images of the Algarve landscape and a random dataset of images obtained from *Google Maps*. The performance of the trained model is discussed using several metrics. The model can be used by the authorities to detect illegal swimming pools in any region, especially in the Algarve region due to the high density of pools there.

Keywords: Computer vision · Neural networks · Deep learning · Object detection · RetinaNet

1 Introduction

The main objective of this work is to detect swimming pools from satellite images, especially in the Algarve region. Due to the high density of pools that exist in this Portuguese region, some may be illegal and actively contribute to tax avoidance. Based on the work developed in [4] and [18], the Deep Learning (DL) methods are the most suitable to perform this type of detection.

In [4], both two-stage and single-stage DL algorithms were studied and compared. The two-stage group includes Convolutional Neural Networks (CNN) [12], Region-based CNN (RCNN) [13], Fast RCNN [14] and Faster RCNN [15]. These algorithms showed very high performance on the proposed task, but very high computational cost and slow prediction speed, due to a first stage that extracts the objects of an image, followed by a second stage in which these objects are classified.

Supported by FCT – Fundação para a Ciência e a Tecnologia, through projects UIDB/00013/2020 and UIDP/00013/2020 of CMAT-UM.

© Springer Nature Switzerland AG 2021
O. Gervasi et al. (Eds.): ICCSA 2021, LNCS 12950, pp. 501–516, 2021.
https://doi.org/10.1007/978-3-030-86960-1_35

The single-stage group includes You Only Look Once (YOLO) [16], Single-Shot Detector (SSD) [17] and RetinaNet [1]. These algorithms classify the entire image and are faster than two-stage detectors. The remarkable speed of YOLO makes it to be preferred for real-time detection at the expense of accuracy [16]. In comparison with YOLO, SSD is slower but its approach improves the detection of objects of different sizes, in an image. The common problem of YOLO and SSD is the presence of class imbalance when the target images have high background presence, resulting in low positive classifications, as in the case of swimming pools [1]. The RetinaNet algorithm has been presented in the literature [1] to address class imbalances and improve detection of smaller objects, achieving a performance comparable with two-stage algorithms, but with the speed of single-stage algorithms.

Since satellite images are characterised by the presence of a massive amount of background, and objects can have different scales, one can conclude, based on the studied DL methods, that the RetinaNet method seems to be the most appropriate to solve the problem proposed in our study.

Having these ideas in mind, in this work we present the development of three RetinaNet models capable of detecting swimming pools in satellite images with high accuracy.

After this Introduction, the paper is organised as follows. The RetinaNet method is briefly described in Sect. 2. Then, to obtain and test the RetinaNet models, we created four datasets that are presented in Sect. 3, one for training and three for testing. The three RetinaNet models are trained and the results obtained with the testing datasets are presented and discussed in Sect. 4. The work ends with the main conclusions presented in Sect. 5.

2 RetinaNet

The RetinaNet algorithm is a single-stage detector that brought two new improvements over YOLO and SSD, namely Focal Loss [1] and Feature Pyramid Network [2], allowing it to match the performance of two-stage detectors without sacrificing speed.

To address possible class imbalances between the background class and the class under investigation, RetinaNet uses a modified version of the cross-entropy function called Focal Loss,

$$FL(p, y) = -\sum_{t} y_t (1 - p_t)^{\gamma} \log p_t \tag{1}$$

where t is the class index, y_t the class label such that $y_t = 1$ if the object belongs to class t and $y_t = 0$ otherwise,

p_t is the predicted probability of the object being of class t and $\gamma \in [0, 5]$. As suggested by the authors [1], a value of $\gamma = 2$ was used in our work.

The Focal Loss function defined in (1) gives lower importance to easily classified examples by using the modulating factor, γ, associated with a new term,

$(1 - p_t)$. In this way, easily classified examples, *i.e.* with high prediction probability (above 0.6), receive a value close to zero, resulting in very little or no learning, and the focus is on hard classified examples [1].

Multiple scale image processing can be used to detect objects with different scales in a single image, but is computationally expensive and time consuming.

As anticipated above, RetinaNet uses a backbone architecture called Feature Pyramid Network. This is a feature extractor that generates multiple multiscale feature maps, focusing on both accuracy and speed [2].

Several algorithms can be used as extractors, and the ResNet was chosen in this work. The ResNet architecture allows to add layers without compromising accuracy [8]. Therefore, different network depths were analysed to determine the one that gives the best results. Specifically, we analysed the following ResNet architectures, namely ResNet50, ResNet101 and ResNet150 with 50, 101 and 150 layers respectively [9].

In summary, the RetinaNet architecture, shown in Fig. 1, consists of: (a) a bottom-up path with a backbone network called Feature Pyramid Network, which computes multiple feature maps at different scales of an entire image; (b) a top-down path that upsamples feature maps from higher pyramid layers and associates equally sized top-down and bottom-up layers through lateral connections; (c) a classification subnetwork responsible for object classification; (d) a regression subnetwork to improve bounding box estimation [2].

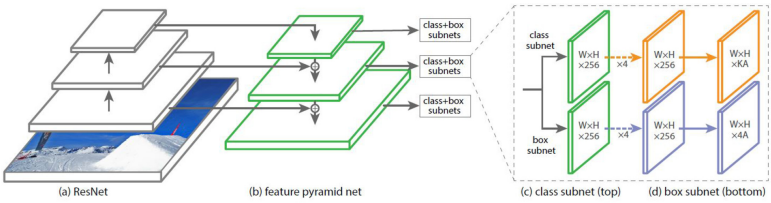

Fig. 1. RetinaNet network architecture composed of four main components: a) bottom-up pathway; b) top-down pathway; c) classification subnetwork; d) regression subnetwork; adapted from [2]. (Color figure online)

The RetinaNet also includes several algorithms to optimise the weights, *i.e.* the parameters, of the neural network, allowing for customisation. By default, RetinaNet uses Adaptive Moment Estimation (Adam) as it has been shown to be quite efficient in training neural networks [3].

3 Datasets

Four datasets were created in this work. One dataset for training, denoted by $\mathcal{D}_{tr}$, and three datasets for testing, denoted by $\mathcal{D}_{test1}$, $\mathcal{D}_{test2}$, $\mathcal{D}_{test3}$, respectively. The proposed testing datasets are composed of images with increasing difficulty for

swimming pool detection. In particular, $\mathcal{D}_{test1}$ is constructed based on a dataset from *Kaggle* and consists of images similar to those used in the training dataset. Moreover, $\mathcal{D}_{test2}$ consists of *Google Maps* images from the Algarve region, in Portugal. This region was chosen because of the high density of swimming pools in this area. Finally, $\mathcal{D}_{test3}$ consists of images with swimming pools and similar looking objects taken from *Google Maps*. These testing datasets are used to illustrate the performance differences among the three RetinaNet models.

3.1 Training Dataset - $\mathcal{D}_{tr}$

A new dataset for training swimming pool detection models was created from a dataset retrieved from *Kaggle*. The original dataset contains 3748 training satellite images with bounding box annotations and labels for classifying cars (labelled 1) and swimming pools (labelled 2). The images are of the RGB type and have 224×224 pixels [11].

For each training image that contains objects of one of the classes, car or swimming pool, there is a corresponding file in XML format (EXtensible Markup Language) written in the PASCAL Visual Object Classes (PASCAL VOC) structure that contains important information, such as the file name of the corresponding image, the size of the image, the recognised object class and the coordinates of the bounding box. An example of an annotation file from the training dataset is the following:

```
<?xml version="1.0"?>
<annotation>
    <filename>000000012.jpg</filename>
    <source>
        <annotation>ArcGIS Pro 2.1</annotation>
    </source>
    <size>
        <width>224</width>
        <height>224</height>
        <depth>3</depth>
    </size>
    <object>
        <name>2</name>
        <bndbox>
            <xmin>149.53</xmin>
            <ymin>196.11</ymin>
            <xmax>193.97</xmax>
            <ymax>224.00</ymax>
        </bndbox>
    </object>
    <object>
        <name>2</name>
        <bndbox>
```

```
            <xmin>120.24</xmin>
            <ymin>212.77</ymin>
            <xmax>158.87</xmax>
            <ymax>224.00</ymax>
        </bndbox>
    </object>
</annotation>
```

From this example of the annotation file, we can see that the image has two pools represented by the two object sections with class label 2 and the coordinates of the bounding boxes.

Since in this work only the detection of swimming pools was of interest, it was necessary to discard the information related to the car class. Therefore, as a first step, a *Python* script (*Py1*) was developed to remove from the dataset the images and the annotation files that contained only cars. The script *Py1* implements the following strategy.

Open each training annotation file and do:

- Count the number of objects of each class, pool (denoted by 2) and car (denoted by 1);
- If there are only objects of class car, then delete the image and annotation file.

For samples with both classes, the car objects annotation data must be removed. Therefore, as a second step, another *Python* script (*Py2*) was developed to implement the following steps.

Open each training annotation file and for each object block do:

- If the label is 1, then exclude the object block.

After applying *Py1* and *Py2* scripts to the complete training dataset of *Kaggle*, the new training dataset D_{tr} was obtained. Dataset D_{tr} contains a total of 1993 training images with their corresponding XML annotation files.

However, the RetinaNet algorithm requires a single annotation file in Comma-Separated Values (CSV) with a specific format. Therefore, in a third step, another *Python* script (*Py3*) was implemented to perform the conversion by extracting the information from the XML files and merging all objects, one per line. The required format for each line is:

```
path/to/image.jpg,x1,y1,x2,y2,class_name
```

After the conversion, an example of some rows from the new CSV annotations file is:

```
000000012.jpg,149.53,196.11,193.97,224.00,pool
000000012.jpg,120.24,212.77,158.87,224.00,pool
000000014.jpg,211.71,156.41,224.00,196.16,pool
```

3.2 Testing Dataset from *Kaggle* - $\mathcal{D}_{test1}$

The dataset $\mathcal{D}_{test1}$ was created based on a testing dataset from *Kaggle* containing 2703 satellite images without any kind of annotation files [11]. Therefore, in order to compare and analyse the performance of the three RetinaNet models, it was mandatory to annotate all testing images. Thus, a *Python* script (*Py4*) was developed to help with this task by reading the coordinates of mouse clicks as bounding box coordinates and extracting the data inherent to the image, *i.e.* size and filename, and writing an XML annotation file for each image. After obtaining the XML files, it is known that the $\mathcal{D}_{test1}$ contains 2703 satellite images with a total of 620 swimming pools distributed over 524 images. These images are similar to those used to create the $\mathcal{D}_{tr}$ training dataset.

3.3 Testing Dataset from Algarve's Region - $\mathcal{D}_{test2}$

A new dataset was created with images from the Algarve region taken from *Google Maps*. We point out again that this region has been chosen essentially for several reasons. First, the main objective of this work is to detect illegal swimming pools, and the Algarve region has a high density of swimming pools. Second, there is a large amount of noise in this area, namely semi-hidden swimming pools and similar looking objects, which makes the problem much more interesting.

To speed up the process of sequentially capturing these sample images, a *Python* script (*Py5*) was also developed in view of automating the process by cutting a large image into smaller images, and allows full-screen printouts of the software to be obtained manually. When using this script, one can choose the number and type of divisions or images, which can be all of the same size or of a random size.

To analyse the performance of the training models in a variety of scenarios mimicking real-world use of satellite imagery, both equal and random size imagery were considered. Therefore, after applying *Py5*, the script *Py4* was applied to create the corresponding XML files for each image.

The new test dataset $\mathcal{D}_{test2}$ is a collection of 289 images of different sizes, zoom percentages and image quality. In total, 298 swimming pools are distributed among 172 examples, the others being negative examples with absence of swimming pools. Some selected examples are shown in Fig. 2. It can be seen that the images of $\mathcal{D}_{test2}$ were intentionally not carefully selected. The goal of using $\mathcal{D}_{test2}$ was to make more difficult the task of finding swimming pools by the training models in real case scenarios.

3.4 Testing Dataset from *Google Maps* - $\mathcal{D}_{test3}$

In order to examine the limitations of the trained models in depth, a few cropped images were carefully selected from *Google Maps*. These images were taken at different zoom percentages and from areas with different image quality. The eight images selected to create this test dataset are shown in Fig. 3.



Fig. 2. Algarve's landscape dataset examples in scale. The various samples have different zoom percentages, sizes and image quality. (Color figure online)



Fig. 3. Eight test images taken from *Google Maps* with different sizes and resolutions. The images are numbered in yellow on the top right corner. (Color figure online)

These images were carefully selected because they have objects that closely resemble swimming pools, and also swimming pools with an unusual appearance. This selection was made with the intention of fooling the detectors and quantifying their possible limitations. Below are the characteristics of each image:

- **image 1** - 433 × 340 pixels, presence of a lake and a blue tag;
- **image 2** - 442×258 pixels, encloses two swimming pools and a pool look-alike glass structure;
- **image 3** - 298 × 241 pixels, a pool without one of the most common characteristics, the blue colour;
- **image 4** - 100 × 118 pixels, very small and high zoom image;
- **image 5** - 366 × 270 pixels, unusual swimming pool body;
- **image 6** - 1351 × 1364 pixels, very high-quality 3-dimensional picture with two pools and a basketball court;
- **image 7** - 1099 × 1333 pixels, the same court as in image 6 and a green swimming pool;
- **image 8** - 732 × 613 pixels, the basketball court seen in images 6 and 7 but with higher zoom percentage.

To generate the XML annotation files for each of these images, the script *Py4* was applied. The new test dataset D_{test3} thus contains a total of 8 images with their corresponding XML annotation files.

4 Training and Testing of RetinaNet Models

In this section, we present the training parameters and metrics used to analyse the performance of the three trained RetinaNet models with different backbone architectures, namely the ResNet50, ResNet101 and ResNet150, using the testing datasets $\mathcal{D}_{test1}$, $\mathcal{D}_{test2}$ and $\mathcal{D}_{test3}$. For each trained model, in order to calculate the metrics *accuracy*, *precision*, *sensitivity* and *specificity*, confusion matrices, with values of True Negative (TN), True Positive (TP), False Negative (FN) and False Positive (FP), had to be created. In addition, Type I and Type II errors were also calculated following the ideas in [5].

- **True Positive (TP):** The expected and the predicted values are positive (meaning that, there is a swimming pool in an image and the model identified it). For each swimming pool correctly detected by the model, 1 is added to the TP value;
- **True Negative (TN):** The expected and the predicted values are negative (meaning that, there are not swimming pools in an image and the model does not detect any in the image). For each image correctly classified as not having pools, 1 is added to the TN value;
- **False Positive (FP):** The predicted value is positive but the expected is negative (meaning that, there is not a swimming pool in the image but the model identifies a part of the image as one). For each pool detected but not being one, 1 is added to the FP value;

– **False Negative (FN):** The predicted value is negative but the expected is positive (meaning that, there are swimming pools in the image but the model is not able to detect them). For each object, wrongly classified as being a swimming pool, 1 is added to the FN value.

4.1 Training

The three RetinaNet models were trained using the training dataset presented in Sect. 3. It is well known that deep neural networks have a large number of unknown parameters that are used to learn different features and map the input to the desired output. Therefore, the task of finding the optimal values for all these parameters, *i.e.* weights, requires a huge labelled training dataset. This can lead to a huge computation time and requires the use of supercomputers. To overcome this problem, one can use the weights of similar pre-trained models as a starting point for training the three RetinaNet models with ResNet50, ResNet101, and ResNet150, see [6] and [7].

Nowadays, there are several neural network models trained using the *ImageNet* dataset (a project that provides a dataset consisting of a large collection of images with human annotations created by academics, containing more than 14 million images of more than 20000 classes and bounding box annotations for 1 million images). The weights learned by some neural networks, *e.g.*, ResNet, InceptionV3, MobileNet, etc., are available to the community for download and use in *Keras*, *Tensorflow*, and in other deep learning libraries.

In this work, we used the weight files of pre-trained ResNets available in *Keras* using *ImageNet* as initial weights for training the three RetinaNet models with ResNet50, ResNet101, and ResNet150. We recall that the total number of weights to be initialised and optimised in these backbone models is 25636712 in ResNet50, 44707176 in ResNet101, and 60419944 in ResNet150 [10]. We used the RetinaNet algorithm options by default, with the following exceptions: i) three different depth ResNet backbone architectures, by default ResNet50; ii) initialisation of the weights using the weight files available in *Keras*; iii) a stack size of 1; iv) a learning rate of 10^{-5} for the optimisation algorithm, by default Adam; v) a maximum number of iterations of 500. We note that using a batch size of 1 means that each time an image traverses the entire network, the weights are updated by the Adam algorithm.

The RetinaNet algorithm requires two Comma-Separated Values (CSV) files with specific formats as input data. A CSV file with the annotations developed with the script *Py3*, and a CSV file in class mapping format with the labels corresponding to the classes. The mapping file must contain the target classes and a corresponding ID number (starts at 0), one per line. In our case, the CSV mapping file looks as follows:

```
pool , 0
```

Thus, after preprocessing the training dataset D_{tr}, the three RetinaNet models were trained with ResNet50, ResNet101 and ResNet150. As with the training dataset, a CSV file containing the annotations, developed using the *Py3* script,

and a CSV file in class mapping format were also used for training. The training was performed in a PC Intel Core i5-8365U Processor, CPU 4.10GHz, with 8GB RAM and integrated graphics. Note that, the usage of a dedicated graphics card would drastically reduce training time, allowing to further train the networks in search of even better weights approximations.

In the next subsections, the validation of the trained models is performed using the developed test datasets $\mathcal{D}_{test1}$, $\mathcal{D}_{test2}$ and $\mathcal{D}_{test3}$.

4.2 Results Obtained with $\mathcal{D}_{test1}$

As expected, the results obtained with the dataset $\mathcal{D}_{test1}$ are excellent for the three considered architectures, see Table 1. It can be seen that the accuracy of each model is high and remarkably close, about 0.95, indicating that the classifiers are correct in most cases. All RetinaNet models have high precision, above 0.95, suggesting that an image labelled as positive is indeed positive, as confirmed by the low number of false positives in the table. Also, type I and II errors are small and very close to each other, indicating low false positive and false negative numbers, respectively. The model using a ResNet152 backbone architecture is better at detecting the class under study, showing lowest number of false negatives, but the performance differences between the three architectures are insignificant.

Table 1. Confusion matrix values and computed evaluation quantities for each of the trained models when using $\mathcal{D}_{test1}$

	TP	TN	FP	FN	Accuracy	Precision	Sensitivity	Specificity	Type I	Type II
ResNet50	471	2200	18	149	0.9412	0.9632	0.7600	0.9919	0.0081	0.2400
ResNet101	480	2285	7	140	0.9495	0.9856	0.7742	0.9969	0.0031	0.2258
ResNet152	503	2241	6	117	0.9571	0.9882	0.8113	0.9973	0.0027	0.1887

The results are good due to the large number of elements in the dataset and also to the fact that the testing images are similar to the training images.

To illustrate the difficulties of each model in detecting swimming pools, two tests are now performed, considering the more complex datasets $\mathcal{D}_{test2}$ and $\mathcal{D}_{test3}$.

4.3 Results Obtained with $\mathcal{D}_{test2}$

The Algarve landscape dataset was given to all RetinaNet models and the obtained results are shown in Table 2.

It is important to keep in mind that there are a total of 298 swimming pools in the dataset and that the images represent the worst case scenario of the models. From Table 2, we can see that none of the three models were able to detect the entirety of the positive examples, see the table column True Positive,

Table 2. Confusion matrix values and computed evaluation quantities for each of the trained models using the Algarve's landscape dataset.

	TP	TN	FP	FN	Accuracy	Precision	Sensitivity	Specificity	Type I	Type II
ResNet50	131	63	161	167	0.3716	0.4486	0.4396	0.2813	0.7187	0.5604
ResNet101	154	97	83	144	0.5251	0.6498	0.5168	0.5389	0.4611	0.9461
ResNet152	276	112	29	22	0.8838	0.9049	0.92617	0.7943	0.2057	0.07383

TP, with the ResNet152 model having by far the highest number of correct detections. Impressively, the ResNet152 model has a high sensitivity value, which means that no swimming pool passes unnoticed, which is also indicated by the low error of type II. This is a very important result, as inspecting an object incorrectly classified as a swimming pool causes less losses than overlooking an illegal construction. The models with a ResNet50 and ResNet101 have such low performance results that they should not be considered for real swimming pool detection. On the other hand, the model with a ResNet152 shows very high accuracy and precision, indicating that the classifier is correct in most cases and if an object is labelled as positive, it is indeed positive. This conclusion is also recognised by the number of false positives reported, corroborated by the lowest type I error.

From this analysis, it can be concluded that the RetinaNet model using ResNet152 backbone architecture outperforms by far the other models using ResNet50 and ResNet101 and shows high performance results. This means that the RetinaNet with a ResNet152 can be used in practise without hesitation.

4.4 Results Obtained with $\mathcal{D}_{test3}$

The results obtained with the three RetinaNet models are detailed and discussed in depth below.

Model Using ResNet50. The images in Fig. 3 were fed to the RetinaNet model using a ResNet50. Objects detected as swimming pools were enclosed by a blue bounding box, as shown in Fig. 4.

Some comments concerning the results obtained for each image are listed below:

- **image 1** - The model did not detect a swimming pool, although the image contains a lake and a blue sign;
- **image 2** - As expected, two swimming pools were detected and the glass structure did not fool the model;
- **image 3** - The model could detect the unclean swimming pool;
- **image 4** - Several swimming pools were detected instead of one. This can be explained by the very low resolution and size of this image;
- **image 5** - Again, three detections were made for a single pool. The confusion can possibly be explained by the unusual shape;

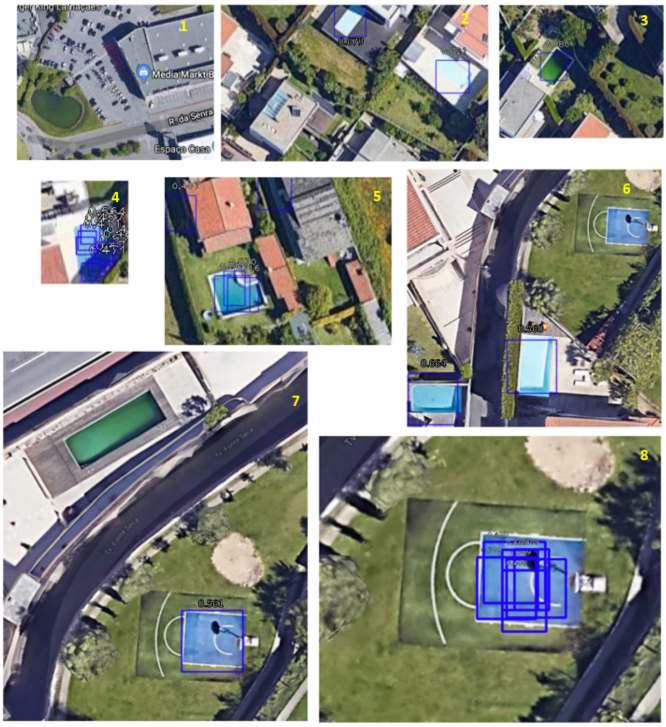

Fig. 4. Eight test images from *Google Maps* with different sizes and resolutions. The objects detected by RetinaNet+ResNet50 are surrounded by blue boxes. The images are numbered in the upper right corner. (Color figure online)

- **image 6** - The model recognised the swimming pools and did not confuse the basketball court with a swimming pool;
- **image 7** - Unlike image 3, the unclean pool was not recognised. This could explain why the lake in image 1 was not detected. Also, the previously unrecognised yard (image 6) was classified as a pool this time. The difference between the images suggests that the model is sensible to the zoom;
- **image 8** - The test performed on this image supports the statement presented in the previous point. The higher zoom helped to fool this model.

Model Using ResNet101. The images in Fig. 3 were fed to the RetinaNet model built on top of a ResNet101. The detected pools were enclosed by a blue bounding box, as shown in Fig. 5.

Some comments concerning the results obtained for each image are listed below:

- **image 1** - The model did not detect a swimming pool, although the image contains a lake and a blue sign;



Fig. 5. Eight test images from *Google Maps* with different sizes and resolutions. The objects detected by RetinaNet+ResNet101 are surrounded by blue boxes. The images are numbered in the upper right corner. (Color figure online)

- **image 2** - The model failed to detect a swimming pool, performing worse than the ResNet50 model for this image;
- **image 3** - The model was unable to detect the unclean swimming pool;
- **image 4** - Multiple swimming pools were detected instead of one. Again, this can be explained by the very low resolution and size of this image;
- **image 5** - Similar to the previous model, three detections were made for a single swimming pool. The confusion could be explained by the unusual shape;
- **image 6** - The model recognised the swimming pools and did not mistake the basketball court by a swimming pool;
- **image 7** - All objects were correctly classified. The only difference between this image and image 3 is the quality. Image 7 has higher quality and 3-dimensional textures;
- **image 8** - The higher zoom percentage had no effect on detection.

Model Using ResNet152. Finally, the images in Fig. 3 were passed to the RetinaNet model built on top of a ResNet152. The detected swimming pools were enclosed by a blue bounding box, as shown in Fig. 6.

Some comments concerning the results obtained for each image are listed below:



Fig. 6. Eight test images from *Google Maps* with different sizes and resolutions. The objects detected by RetinaNet+ResNet152 are surrounded by blue boxes. The images are numbered in the upper right corner. (Color figure online)

- **image 1** - The model did not detect a swimming pool, although the image contains a lake and a blue sign;
- **image 2** - The ResNet152 model was able to correctly classify the objects in this image;
- **image 3** - The unclean pool was correctly detected;
- **image 4** - Unlike the previous models, the swimming pool in this image was not detected;
- **image 5** - Unlike the ResNet50 and ResNet101 models, the ResNet152 model correctly detected a single pool in this image;
- **image 6** - It correctly identified the pools and was not fooled by the blue basketball court;
- **image 7** - Both the pool and the court were correctly classified;
- **image 8** - As seen before, the basketball court did not fool the model.

The results obtained with the three models are now summarised.

- The models with ResNet50 and ResNet101 are not able to detect unusually coloured pools, green as in the tested examples;
- The results show that the model with ResNet50 is sensitive to the zoom of the image, indicating a wrong classification of objects when the zoom percentage is high;

– Classification of exceptionally small images is difficult for the model with ResNet152.

The model with the best performance was RetinaNet with a ResNet152 backbone architecture. In fact, it was able to correctly detect all swimming pools, with the exception of one in the smallest image 4. This result is not surprising, since the superiority of the model with a ResNet152 has already been shown by the results with $\mathcal{D}_{test2}$. Also, this last test data set, $\mathcal{D}_{test2}$, in addition to the results with $\mathcal{D}_{test3}$, proves that the models with a ResNet50 and ResNet101 are highly error-prone.

5 Conclusions

The aim of this work was to develop a model capable of detecting swimming pools in satellite images using a Deep Learning approach. To achieve this, several object detection algorithms were investigated and compared. The concluding remarks of this work are:

– We developed a dataset for training, $\mathcal{D}_{tr}$, and for testing, $\mathcal{D}_{test1}$, by modifying an existing one, filtering swimming pool annotations and creating label files for the testing images. To simulate real-world usage, *e.g.* by government agencies, 289 images of the Algarve region, in Portugal, were extracted from *Google Maps* resulting in a new dataset, $\mathcal{D}_{test2}$. In addition, to closely examine the limitations of the trained networks, eight cropped images from *Google Maps* were carefully selected to fool RetinaNet, $\mathcal{D}_{test3}$.
– Three RetinaNet models were trained with the training samples of the modified dataset, $\mathcal{D}_{tr}$, each one with a different depth backbone architecture: ResNet50, ResNet101, and ResNet152. To evaluate the models, the testing sets ($\mathcal{D}_{test1}$, $\mathcal{D}_{test2}$, $\mathcal{D}_{test3}$) were given to the models and a comparison between the predictions and the expected values was performed using confusion matrices that allowed to compute five performance metrics.
– The confusion matrices along with the values of the performance metrics showed that the RetinaNet model with a ResNet152 backbone architecture is the best at detecting swimming pools and achieved the highest performance in all three testing sets. Furthermore, this model was not fooled by the $\mathcal{D}_{test3}$ samples and only showed difficulty in correctly predicting very small and poor quality images.

References

1. Lin, T., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2980–2988 (2017)
2. Lin, T., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2117–2125 (2017)

3. Kingma, D., Ba, J.: Adam: a method for stochastic optimization. In: International Conference on Learning Representations (2014)
4. Coelho, C.: Machine Learning and Image Processing. Master's thesis, to appear in Repositorium, University of Minho (2020). http://repositorium.sdum.uminho.pt
5. Powers, D.: Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. J. Mach. Learn. Technol. **2**, 37–63 (2011)
6. Zabir, M., Fazira, N., Ibrahim, Z., Sabri, N.: Evaluation of pre-trained convolutional neural network models for object recognition. Int. J. Eng. Technol. **7**(3), 95–98 (2018)
7. Salahat, E., Qasaimeh, M.: Recent advances in features extraction and description algorithms: A comprehensive survey. In: Proceedings of the IEEE International Conference on Industrial Technology, pp. 1059–1063 (2017)
8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778 (2016)
9. Keras RetinaNet. https://github.com/fizyr/keras-retinanet. Accessed 4 May 2021
10. Keras Applications. https://keras.io/api/applications. Accessed 4 May 2021
11. Swimming Pool and Car Detection. https://www.kaggle.com/kbhartiya83/swimming-pool-and-car-detection. Accessed 4 May 2021
12. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge (2016)
13. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 580–587 (2013)
14. Girshick, R.: Fast R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1440–1448 (2015)
15. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans. Pattern Anal. Mach. Intell. **39**(6), 1137–1149 (2015)
16. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779–788 (2016)
17. Liu, W., et al.: SSD: single shot multibox detector. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9905, pp. 21–37. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46448-0_2
18. Coelho, C., Costa M., Ferrás L., Soares A.: Development of a machine learning model and a user interface to detect illegal swimming pools. In: SYMCOMP 2021 5th International Conference on Numerical and Symbolic Computation Developments and Applications, pp. 445–454. Évora (2021)