Object Detection in Aerial Images

Jordi Orihuela Ponsarnau

Resum– Aquest treball estudia l'aplicació de la detecció d'objectes en imatges aèries. Concretament, es realitza una anàlisi de la diferència entre les imatges estàndard i les aèries i s'extreuen els aspectes on es poden aplicar canvis per obtenir millors resultats en aquestes últimes. Es realitza una comparativa de detectors basats en regions aportant vàries solucions tant en la part de generació de regions candidates com en la de classificació. Pel que fa a la generació, es compara analíticament l'ús d'una finestra lliscant amb l'algorisme de cerca selectiva. Per altra banda, a la tasca de classificació, principalment s'ataca el problema de la baixa resolució utilitzant una xarxa neuronal convolucional poc profunda i es presenta un model basat en tècniques de super resolució.

Paraules clau– Detecció d'objectes, Imatges Aèries, Aprenentatge Computacional, Finestra Lliscant, Cerca Selectiva, Xarxes Neuronals Convolucionals, Super Resolució

Abstract– This work studies the application of object detection in aerial images. Specifically, a review of the differences between standard and aerial images is made and the aspects where changes can be applied in the latter to improve results are extracted. A comparison of region-based detectors is made by providing several solutions for both the region proposal part and the classification one. Regarding the region proposal generation, the use of a sliding window is compared analytically with the Selective Search algorithm. On the other hand, in the classification task, the low-resolution problem is mainly tackled using a shallow convolutional neural network and a model based on super resolution techniques is presented.

Keywords– Object Detection, Aerial Images, Deep Learning, Sliding Window, Selective Search, Convolutional Neural Network, Super Resolution

1 Introduction

In the last decade, the field of computer vision has made significant progress in the automatic understanding of images. This has been mainly thanks to the advances that Deep Learning (DL) [1] has brought to the field. The biggest one being the application of Convolutional Neural Networks (CNN) [2] as a solution to traditional problems such as image classification, semantic segmentation or object detection among others. Deep Learning is usually referred as the common name for the different algorithms that use Deep Neural Networks to extract meaning out of data.

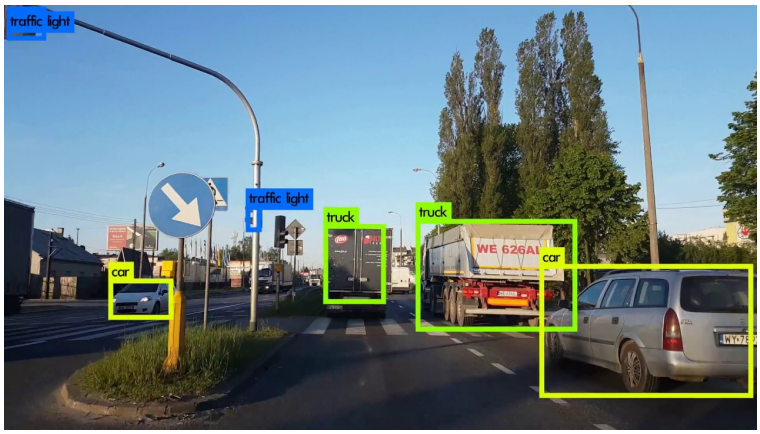
The huge raise in high-quality commercial drones and other type of UAVs allows almost everyone to generate and obtain aerial images of a terrain quite easily and cheaper than ever before. Getting an important amount of quality data is fundamental for Deep Learning and any other data-based learning algorithm. This data can be applied to agriculture management, precision farming, surveillance tasks, etc.

This work combines computer vision with the aerial imagery taken by drones to explore how object detection works in this kind of scenes. Object detection is the method used in order to understand an image by labelling the different objects in it and estimating their location. Traditionally object detection has been applied to standard images (see Figure 1) where the camera is in parallel with the ground.

Applying object detection in aerial images is an interesting and challenging task because these images are quite different from the ones that are commonly used in the field. If we consider those differences and build a more specific detector instead of a generic one we could achieve better performance. The main differences are shown in Table 1.

Object detection has been extensively explored in the literature [5, 6, 7] where different algorithms are proposed that try to improve a particular aspect of the process. The traditional process of object detection can be divided in two parts: find where the objects are located in an image (**object localisation**) and determine which class each ob-

• E-mail de contacte: Jordi.Orihuela@e-campus.uab.cat
• Menció realitzada: Computació
• Treball tutoritzat per: Daniel Ponsa Mussarra
• Curs 2018/19



Standard images

- Different scales.
- High resolution images.
- Medium/Big objects.
- Camera on the plane.

Figure 1: Object detection on a standard image [3].

Aerial images

- Objects at fixed scale.
- Low resolution images.
- Small objects.
- Rotation on camera.

Figure 2: Object detection on an aerial image [4].

Table 1: Main differences between standard and aerial images with an example for both cases.

ject belongs to (**object classification**).

Depending on how those tasks are solved, the object detection methods are divided in two groups:

- **Region based methods:** Uses an algorithm (e.g. Sliding Window) to generate different regions proposals that will be classified using a separate algorithm (e.g. CNN) into the different object categories. They are usually more accurate.

- **Single shot methods:** Unifies both processes to get the result in a single step (e.g. a single Neural Network which has a proposal network followed by a classification one). They are usually faster.

Doing an extensive analysis of the different strategies for each method is completely out of reach for a project like this. For this reason, we will focus in the region based methods because they allow us to experiment in the region proposal and classifying processes independently and research how to improve performance in object detection in aerial images.

2 State of the art

Region-based object detection has been an studied field for several years and nowadays the canonical models are all based in Deep Learning techniques. The most known algorithms that really improved in some way its predecessor have been: R-CNN [8], Fast R-CNN [9] and Faster R-CNN [10].

- **R-CNN:** It uses Selective Search to create 2000 region proposals. Those regions are reshaped into a fixed size and feed into a CNN one by one which outputs a feature map that is then classified by an SVM. It also has a bounding box regressor which helps in getting better localisation of the objects. It is the most similar to our implementation. Even though it can get good results, processing all the regions individually makes the algorithm really slow.

- **Fast R-CNN:** It it design to solve the timing issue in R-CNN. Instead of generating region proposals out of the original images, it first generates the feature map by processing the whole image on a CNN. Then it applies the Selective Search to generate the regions and

classifies them using a Fully-Connected network instead of using a different model (SVM). Not having to generate the feature map for each region causes a great improvement in time.

- **Faster R-CNN:** Fast R-CNN despite being way faster than R-CNN it is still takes a lot of time because of the Selective Search. Faster R-CNN introduces a region proposal network (RPN) which is located in the same place that the Selective Search was in Fast R-CNN. It works by sliding a 3x3 window across the feature map and for each location generating multiple possible regions using different predefined anchor boxes (one taller, one wider, one larger, etc). Finally, it classifies each box into whether it has an object or not, if the confidence on having an object is over a certain threshold, the box is considered a region proposal and it is classified in the same way it was in Fast R-CNN.

The results on Table 2 show the great improvement (especially time-wise) that Faster R-CNN provides.

Method	Time	mAP
R-CNN	50s	66.0
Fast R-CNN	2s	66.9
Faster R-CNN	198ms	69.9

Table 2: Time and mAP results of each method on PASCAL VOC 2007 dataset [11].

Regarding object detection in aerial images specifically, there have been some attempts to solve the problem in different ways [12, 13, 14]. The most used solution is tackling the low resolution issue by applying super resolution techniques or modifying the state of the art models to perform better in this kind of images.

3 GOALS

The main goal of the project has already been defined in the introduction but in this section it is going to be detailed and prioritised with a goal tree (Figure 3).

1. **Main goal:** Compare the performance of different Deep Learning object detection methods applied to aerial images.

In order to achieve this goal, we will also need to:

2. Study the state of the art regarding object detection, aerial images and low resolution image classification.

3. Define a pipeline which allows an easy execution and evaluation of different methods.

4. Analyse and compare the performance of using Sliding Window or Selective Search as the region proposal algorithm. [15].

5. Implement and train a simple CNN as a baseline classifier.

6. Implement and train a Super Resolution (SRCNN) [16] to improve image resolution.

7. Analyse and compare the performance of using Bicubic interpolation [17] or a SRCNN in the classification task.

8. Evaluate the different methods implemented and analyse the results to conclude why the performance is higher in one or another.

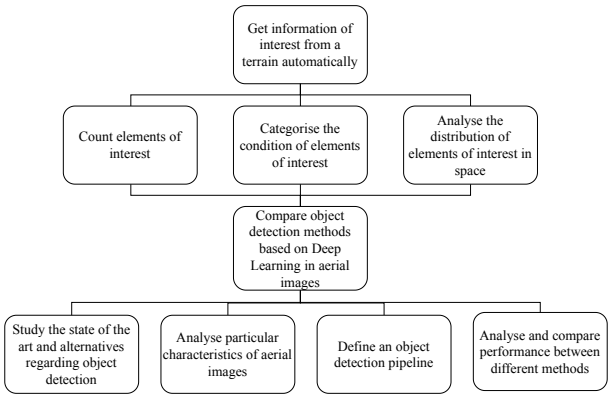


Figure 3: Goal tree of the project.

4 METHODOLOGY

In this section we define how we are going to execute the different elements of the project in order to achieve the goals that we have contemplated.

4.1 Development tools

There are various Deep Learning libraries for implementing and training neural networks. The most used by both the researchers and the industry are Tensorflow [18] and Pytorch [19], both in **Python**. We will use **Tensorflow** along with the high level library Keras [20] which is buit-in and will simplify the whole process.

The main disadvantage of Deep Learning is the huge amount of resources that it needs to train a model. Usually Graphical Processing Units (GPUs) are used to minimise the training time. For this reason, we will use some GPUs granted by the Centre de Visió per Computador (CVC) [21].

The code will be uploaded to a Github repository [22] in order to keep a version history and share progress between the student and the teacher.

4.2 Software architecture

It is important to establish a pipeline that allows us to easily change the method that is applied in order to extract the same kind of metrics and results in each execution.

As it has been already stated in the goals section, various region proposals methods and classifiers will be analysed. This entails that in our pipeline (see Figure 4) we need to be able to modify the region proposal or the classifier at any minute and get the same type of results. That allows us to run a detection using, for example, Selective Search and Super Resolution and obtain the same metrics in equal format as any other combination.

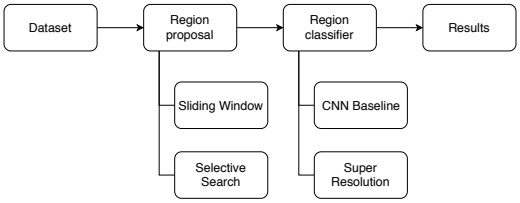


Figure 4: Execution pipeline showing different steps and alternatives for each one.

4.3 Metrics

Since the project is about analysing and comparing different algorithms, evaluating the performance of each one is key to getting good results. This is why choosing the correct metrics for both the region generator as well as the classifier is really important.

In the first step, the region proposal algorithm, we will evaluate it using the **Intersection over Union (IoU)** [23] which measures the overlap between the predicted region and the ground-truth (the real one). The result is a value between 0 and 1 where a higher value implies a better prediction of the region. It will also be important to consider the timing aspect of each algorithm.

$$\textbf{IoU} = \frac{OverlappingRegion}{CombinedRegion}$$

Being (x1, y1) the upper left coordinate and (x2, y2) the opposite one (lower right) of both the predicted region and the real one:

$$(new\_x1, new\_y1) = (max(x1), max(y1))$$

$$(new\_x2, new\_y2) = (max(x2), max(y2))$$

Overlapping Region $= (new\_x2 - new\_x1) \cdot (new\_y2 - new\_y1)$
Combined Region $= area(boxPred) + area(boxGT) - Overlap.Reg.$

On the other hand, we will use the standard metrics in image classification. Those range from the more basic ones such as the accuracy, precision and recall to a confusion matrix which will provide a more visual and detailed result. As some super resolution techniques are going to be applied, we will use the Peak Signal-to- Noise Ratio (PSNR) [24] to evaluate how well an upscaled image resembles the original one.

Once the generator and classifier are evaluated, we can get an idea of how good our object detector could end up being due to the limitations that each part has.

5 IMPLEMENTATION

5.1 Datasets

DOTA [25] is the standard dataset for object detection in aerial images. It contains 2806 images with more than 188.882 objects annotated in total. Each image has a size from 800x800 to 4000x4000 approximately and contains objects from various shapes, orientations and scales. The objects are divided in 15 categories: plane, ship, storage tank, baseball diamond, tennis court, basketball court, ground track field, harbor, bridge, large vehicle, small vehicle, helicopter, roundabout, soccer ball field and swimming pool.

Since we did not want to over-complicate the task and neither spend a huge part of the time we had for the project training the different networks, the first thing we did was to only use the train set of DOTA as our whole dataset. We had 1410 instead of the 2806 images in the original dataset.

We also decided to standardise the scale of the different images by searching for the maximum ground sample distance (the physical size in meters of one image pixel, provided by the dataset) and downscale (using anti-aliasing to lose as minimum quality as possible) the whole dataset by the factor between the gsd of the image and the maximum one. Once this was done, we realised that there a few images which had a GSD that was too low or too high and this produced images which really low quality. At the end, we ended up choosing images with a GSD from 0.09m/px and 0.295m/px (maximum) which reduced our dataset to 884 images. To sum up the process, an example is proposed: With a maximum GSD of 0.295m/px, the process for an image of size 1200 x 1000 at 0.15m/px would be to calculate the factor (0.15/0.295 = 0.5) and then downscale the image by the same factor (1200*0.5 x 1000*0.5 = 600 x 1000). This dataset would be referenced from now as **Dataset-0_29**.

In order to train a Super Resolution algorithm, we need to have the same image in both low and high resolution. This is why we need to generate a lower resolution version of the Dataset-0_29, specifically a version at 25% of the resolution. This implies that the GSD will be also downscaled at 25% which will result in a maximum GSD of 1.18m/px. This dataset would be referenced from now as **Dataset-1_18**.

5.2 Experiments

The different goals that we have set for this work are divided in 3 different experiments:

1. Analyse the performance of using Sliding Window or Selective Search as region proposal generators.
2. Classify regions using a simple CNN.
3. Analyse and compare the classification of low resolution regions using super resolution techniques.

5.2.1 Experiment 1: Region proposal

In this experiment we want to see how well the different algorithms generate a region proposal where it should be. We will use the IoU metric that we have already discussed to see how many proposals we have where a real object is depending on the IoU. We will represent that with an histogram.

Sliding Window
Even though our dataset has been standarized in scale, it is still has a lot of variety regarding the shape and size of the different objects in each category. Therefore, to apply a Sliding Window it would be necessary to have multiple windows with different sizes and aspect ratios. Besides

that, the images are usually quite big so we will end up with lots of regions which means a huge computational cost. For this reason, we have decided to not generate all the possible regions but only those that are closest to the objects. We generate what would be the center points of the sliding window (with a stride of 8) and then check for each object with point is closest to it. Then we apply a similar solution as the one used by the RPN of Faster R-CNN, we center different predefined boxes and see which one of those gets the higher IoU.

Selective Search

Selective Search [15] is an algorithm that has been applied to methods in the state of the art such as RCNN. It works by grouping regions that have similar color, texture and size using a graph-based segmentation based by Felzenszwalb and Huttenlocher [26]. Firstly, it generates an over-segmented image which are the smaller regions and then it continues merging regions until everything is combined together. This means that the first regions generated by the algorithm will be the most specific and small and the last ones will be bigger and more general (see Figure 5). As RCNN does, we will just use the first 2000 regions for each image.

Figure 5: Selective Search's segmented image and region proposals in three different steps [15].

5.2.2 Experiment 2: CNN Baseline

Once we have our regions generated, we need to classify each one to see if there is an object and to which class it belongs. To do so, besides the 15 classes that DOTA has, we need to add an extra one which will be called 'background' and will be used to classify those regions without any object.

Convolutional Neural Networks are used in image classification thanks to their ability to learn the characteristics of an image without requiring any feature hand-engineering. As it is shown in Figure 6 the first layer receives an image with a fixed shape (32x32 in our case) and the following layers apply different functions to the neurons to finally produce the predicted label.

A CNN needs lots of labelled images to learn from, in our case we already have the images for the 15 classes (reshaped to 32x32 which is the input size of the network) of objects but are missing the ones for the background. To generate those, we will check the IoU between each of the 2000 regions that Selective Search provides us and the real regions. If the IoU is 0 (which means that there is not an object there), we add that region (also with size 32x32) to the dataset labelled as background. We will use the Dataset-0_29.

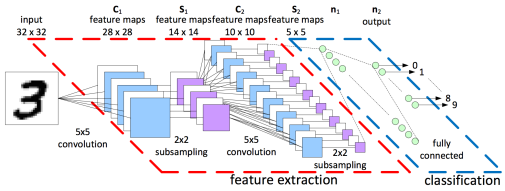


Figure 6: Example of CNN applied to MNIST dataset [27].

One of the biggest problems that we face in aerial images is that, since the objects are so small, the regions end up being in really low resolution. When using a CNN, the higher resolution the better because the convolution layers that the network applies in order to extract features from the image cause a loss of the resolution in the image. So this is why the chosen model for our CNN is not very deep and its quite simple (see Figure 13 in Appendix).

5.2.3 Experiment 3: Super Resolution

This last experiment analyses the performance of applying different super resolution techniques to low resolution images. We will extract regions of 8x8 from the Dataset-1_18 using the same process of Experiment 2. The regions are 8x8 because we want to upscale them by 4 to be able to compare the result with the high resolution ones (Dataset-0_29) and to also use them as input for our classifier which needs the image to be 32x32. The two algorithms that are going to be compared are the bicubic interpolation and a Super Resolution CNN (SRCNN).

Bicubic interpolation

Out of the traditional upscaling methods, bicubic interpolation provides the smoothest and best results. It works by resampling each pixel taking into account its 16 nearest neighbour (a window of 4x4). This is usually slower than other methods but since our images are really small it is not noticeable and the results are much better.

Super Resolution CNN

Using Deep Learning to improve performance on super resolution algorithms has produced great results [28]. This experiment uses a SRCNN which is based on a CNN and is one of the most simple but effective methods. The input of the network is the bicubic interpolation version of the low resolution image, so we first need to do some pre-processing on the image. This means that the input shape of the network is the same as the output (in our case 32x32) but it learns to generate better images.

The architecture of the network is depicted in Figure 7 and the parameters used are the same as the ones that got the best results on the original paper [16].

6 Results

In this section an evaluation and comparison of the different experiments is done.

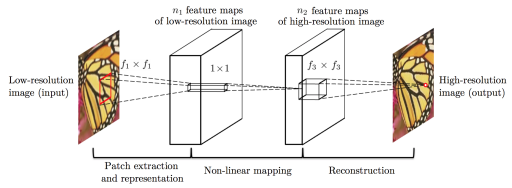


Figure 7: Overview of the SRCNN architecture [16].

6.1 Datasets

It is important to see the final distribution of classes in our dataset before jumping straight into the experiments (Figure 8. This provides us more information of how many objects per class exist and would be useful to help us understand the results in the classification experiments.

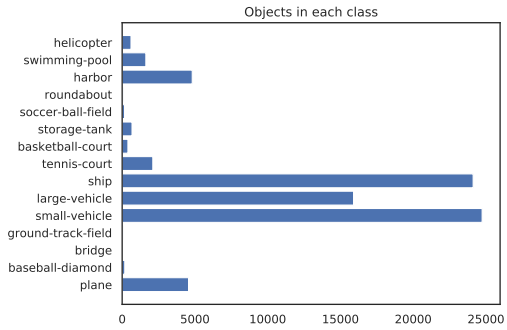


Figure 8: Histogram of the number of samples for each class in our dataset.

It is obvious that the dataset is highly imbalanced with classes like ship or small vehicle having more than 20000 samples and others like bridge only having 13. On the experiments regarding classification we will need to have this in mind in order to get good results.

In both datasets we have the same number of images, 884. We have split those into 3 subsets: train (70%), validation (20%) and test (10%). We end up having the following distribution of images per subset:

- **Train**: 619 images
- **Val**: 177 images
- **Test**: 88 images

6.2 Experiments

Now that we know how our dataset is, it is time to evaluate the different experiments that we have implemented.

6.2.1 Experiment 1

To be able to perform the Sliding Window the way we have designed, we first need to define the default boxes that will be used. We will generate each box by using an aspect ratio and a scale. The ratio is what will define if the box is wider or taller and the scale the size of it (height x width). After studying the size of the different objects in the dataset and doing some testing, the final values are:

Ratios: 0.4, 0.75, 1.0, 1.3, 1.6, 2.0

Scales: [250, 500, 1000, 1500, 2000, 2500, 3500, 4000, 4500, 5500, 7000, 15000, 20000]

In total, for each window there are 66 different boxes generated. The exact shape of the box is defined by:

$$Height = abs(\sqrt{\frac{Scale}{Ratio}})$$

$$Width = abs(\frac{Scale}{Height})$$

For example, if we chose a ratio of 1.3 and a scale of 2500, the final box would be 44x57.

It is now possible to check how many of the regions generated by the Sliding Window are actually where an object is (positive region proposal). Since what matters to us are the results for the test subset, this process is going to be made only on it. In Figure 9 we can see that for lower IoUs (0.5 and 0.6) more than 70% of the regions are detected correctly. When the IoU is 0.7 it still detects half of the objects but when we want more precision the results drop quite heavily. For example, a near perfect detection would be an IoU of 0.9 and we can see that clearly it is only detection with that precision 5.49% of the objects. It is also noticeable that it detects the exact same box (IoU of 1) on 0.02% of objects.

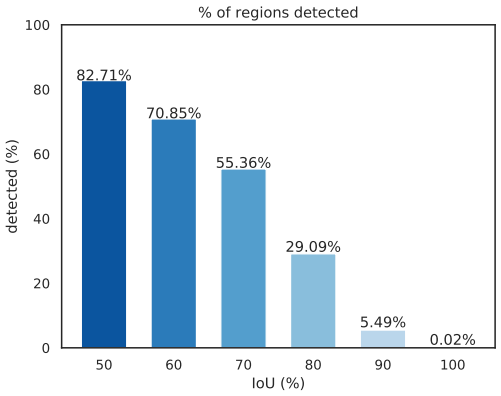


Figure 9: Histogram of the percentage of positive regions proposals by IoU using Sliding Window.

On the other hand, the Selective Search also requires some decisions to be made. The most important one is to decide which is the minimum size of region that you want. In our case we decided to set it at 500 which would be a region of approximately 20x20. Since all the objects in DOTA are quite small and Selective Search sometimes produces really big regions, we have decided to also delete those regions that are bigger than 40000.

The results using Selective Search are quite different (see Figure 10) from the Sliding Window ones. With lower IoUs the percentage of positive region proposals is considerably lower (with IoU 0.5, it has dropped a 22%), this is because, as expected, since the Sliding Window generates way more regions it also gets more correct.

The interesting thing is that Selective Search is much more regular, there is no heavy drop in the percentage detected as it was the case of IoUs 80-90 in the Sliding Window. Furthermore, for the two highest Intersection over Union values, Selective Search outperforms considerably its competitor. This is remarkable because if the algorithm

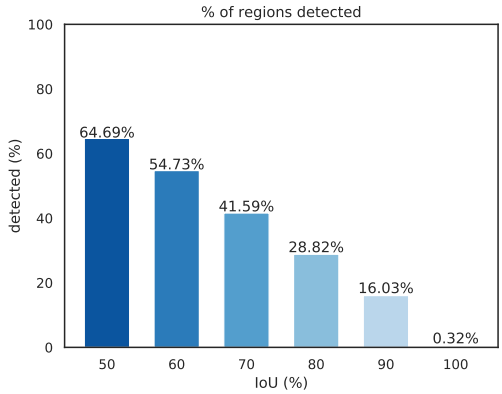


Figure 10: Histogram of the percentage of positive regions proposals by IoU using Selective Search.

detects more precisely a higher number of objects it would facilitate the task of the classifier.

We have already said that Sliding Window is really slow and impossible to use in a real case of object detection. It is possible to verify this by looking at the number of regions that each algorithm produces in a, for example, 1024x1024 image.

Sliding Window: We would generate 1024/8 (stride) x 1024/8 centres (16384) and to each of these centres we would apply 66 different boxes, generating more than 1M of regions.

Selective Search: Even though Selective Search generates a different number of regions depending on the composition of the image (colour, texture, shapes...) but, as we have already said, we only use 2000 regions for each image.

It is now even more clear that using a Sliding Window, even though for some cases it gives more flexibility and better results, with this kind of images is very expensive computationally and not useful for this task. This is why in the following experiments we will use only Selective Search.

6.2.2 Experiment 2

Training any kind of neural network requires a lot of time not only for the training itself but also for adjusting the hyper-parameters of the network in order to get better results. Our CNN uses the model in Figure 13 and after training different models the best one produces the results shown in Table 3. To train the network, as we have already stated, we use the real boxes for the objects and 20k random regions from Selective Search (with IoU equal to 0).

We also evaluate the classifier on two different test datasets. The first one uses the same process has the training one (real objects and Selective Search background) and the second one uses only all the regions generated by Selective Search (we consider it an object if it has an IoU over 0.5). The latter would gives us real information of how the whole object detector works.

	Precision	Recall	Accuracy
GT + SS background	0.16	0.66	0.73
SS regions	0.16	0.52	0.72

Table 3: Classification mean results of test images on Dataset-0_29.

The first thing to highlight is that the accuracy is pretty good, but we need to be careful because our dataset is highly imbalanced (most of the regions are background) and it could be predicting always the same class. Figure 11 shows a confusion matrix which does not use all the background regions for better visualisation. Thanks to the confusion matrix, we can check that it is not predicting every image as the same class and it is working as expected.

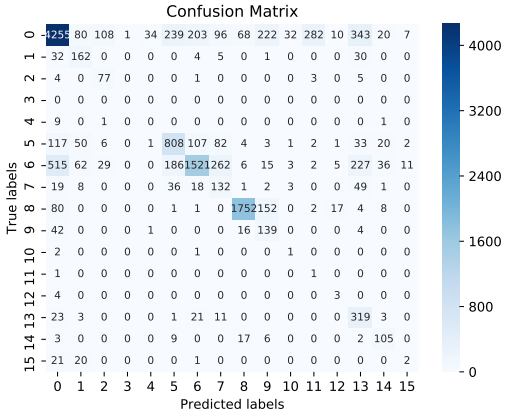


Figure 11: Confusion matrix using 6000 regions of background.

The other thing that is noticeable is that we have a very low precision, we can also check in the confusion matrix the reason behind this. What is happening is that some background images resemble another class for its colour or texture and the network is predicting it as that class instead of background. This does not affect the background precision/recall because it is actually predicting the majority of these correctly.

If we compare the two datasets, as one could expect, the one using the ground-truth boxes gets better results because the regions with objects are perfectly centered. On the other hand, using only the Selective Search regions the results are not that worse which means that the CNN is able to generalise correctly.

With this experiment we have seen that it is possible to more or less classify the regions when the image resolution is still decent.

6.2.3 Experiment 3

We have seen that we can get good results using regions of 32x32, but it is also interesting to study what would happen if we had even images with worse resolution and how could we cope with it.

After training the SRCNN we need to evaluate whether it is actually performing better than the bicubic interpolation. To do so, we use the PSNR which is a metric that gives a higher value when the recomposition of the image is more similar to the original one. The PSNR results of upscaling test images from 8x8 to 32x32 are:

- **Bicubic**: 70.9
- **SRCNN**: 72.2

As we can see, the PSNR is 1.3 higher on average using SRCNN over bicubic interpolation. It is clearer if we visualise what the difference in PSNR on real images. Figure

12 shows the original image and the images generates by both methods and we can perfectly see that while the bicubic image is blurred the SRCNN version is quite similar to the original one.

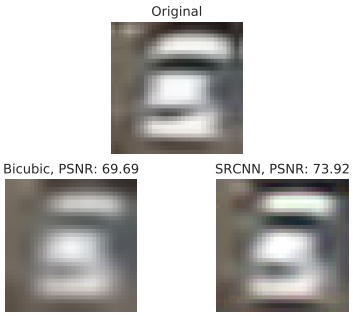


Figure 12: Bicubic and SRCNN applied to an image of the dataset.

Using the same model as in the previous experiment, we will retrain a model applying both the bicubic interpolation and a SRCNN to the Dataset-1_18.

To get a more complete result, two comparisons are going to be made. The first one using the regions from Dataset-0_29 downscaled which will allow us to analyse how worse the predictions get with the lower resolution in comparison to the results of Experiment 2. The second one uses the 8x8 regions directly from Dataset-1_18.

On Table 4 we can see that, as it could be expected, the results for the regions of the Dataset-0_29 are considerably worse than they were on the previous experiment which used the original images. Despite that, with lower resolutions it is also easier to classify the subset which uses the real objects regions images and not only the regions extracted by the Selective Search.

Regarding the comparison between the two methods, it is evident that upscaling images with the SRCNN is better in all the different cases. In general, the SRCNN gets an improvement in accuracy of around 13-15% compared to the traditional method.

This corroborates our original hypothesis which is that when working with low resolution images, choosing the best method for upscaling is key to a great improvement in classification results.

	Precision	**Recall**	**Accuracy**
Bicubic GT+SS (0_29)	0.10	0.58	0.49
Bicubic SS (0_29)	0.13	0.49	0.49
SRCNN GT+SS (0_29)	0.10	0.58	0.58
SRCNN SS (0_29)	0.13	0.43	0.58
Bicubic GT+SS (1_18)	0.24	0.70	0.55
Bicubic SS (1_18)	0.16	0.48	0.51
SRCNN GT+SS (1_18)	0.25	0.71	0.62
SRCNN SS (1_18)	0.18	0.52	0.59

Table 4: Classification mean results using Bicubic interpolation and SRCNN for upscaling test images.

7 Conclusion

In this work, we started by studying the state of the art regarding object detection and aerial images which gave us the information we needed to see what could be done to combine those two fields and improve performance. We decided to use region-based methods in order to experiment with both the region proposal generation and the classification task.

We have developed a pipeline which allows us to experiment and evaluate the methods proposed really easily. All the methods are evaluated by state of the art metrics which allows us to do a comparison between them more effective and precise.

We have analysed two different algorithms for region proposals (Sliding Window and Selective Search) and after various tests we have determined that out of those two the Sliding Window is too slow to use in an object detector. We have also seen that Selective Search with a dataset like DOTA which has a wide variety of shapes and sizes is capable of detecting maximum 65% of the regions correctly.

Classification has been applied to two datasets, one with higher resolution (Dataset-0_29) than the other (Dataset-1_18). A simple CNN has been used on the first one to get a performance baseline. A super resolution network (SRCNN) is proposed to tackle the low resolution problem in aerial images and compared to a traditional upscaling technique (Bicubic interpolation). With this analysis, we prove that getting the highest resolution images is fundamental for a good classification. With the proposed SRCNN we get a substantial improvement compared to bicubic interpolation which confirms that using super resolution techniques is a valid solution to minimise the impact of one of the main problematic characteristics that aerial images have.

In the appendix, Figures 14, 15 show final detection results using the best method (CNN baseline).

7.1 Future work

Now that we have a basic understanding of how to work with aerial images, it would be interesting to continue this study deepening into other particularities of them. Some ideas might be:

- Use Spatial Transformer Networks [29] to tackle the rotation of objects.
- Apply and modify up-to-date state of the art methods such as Faster R-CNN.
- Compare results in terms of speed and accuracy with a single-shot method.

8 Acknowledgements

This project has been made with the support of the BOSSS project (TIN2017-89723-P).

It has also been fundamental the guidance received from Daniel Ponsa Mussarra as the supervisor for the project. He has helped me a lot during the whole process and provided information whenever I needed it.

I would also like to thank the help received from my friends and family and their support during these last months.

REFERENCES

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. URL: http://www.deeplearningbook.org.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, (USA), Curran Associates Inc., 2012.

[3] K. Majek, "4k tiny yolo object detection 2." URL: https://github.com/karolmajek/darknet. [Online; accessed 28/06/2019].

[4] D. Sincha, M. Chervonenkis, and P. Skribtsov, "Vehicle detection and classification in aerial images," *Indian Journal of Science and Technology*, vol. 9, no. 48, 2016.

[5] Z. Zhao, P. Zheng, S. Xu, and X. Wu, "Object detection with deep learning: A review," *CoRR*, vol. abs/1807.05511, 2018.

[6] J. Hui, "Object detection: speed and accuracy comparison," March 2018. URL: https://www.medium.com/@jonathan_hui/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359. [Online; accessed 10/03/2019].

[7] J. Xu, "Deep learning for object detection: A comprehensive review," September 2017. URL: https://www.towardsdatascience.com/deep-learning-for-object-detection-a-comprehensive-review-73930816d8d9. [Online; accessed 10/03/2019].

[8] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *CoRR*, vol. abs/1311.2524, 2013.

[9] R. B. Girshick, "Fast R-CNN," *CoRR*, vol. abs/1504.08083, 2015.

[10] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015.

[11] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results."

[12] Z. Wang, S. Chang, Y. Yang, D. Liu, and T. S. Huang, "Studying very low resolution recognition using deep networks," *CoRR*, vol. abs/1601.04153, 2016.

[13] Y. Ren, C. Zhu, and S. Xiao, "Small object detection in optical remote sensing images via modified faster r-cnn," *Applied Sciences*, vol. 8, p. 813, 05 2018.

[14] M. CHEVALIER, N. Thome, M. Cord, J. Fournier, G. Henaff, and E. Dusch, "Low resolution convolutional neural network for automatic target recognition," in *7th International Symposium on Optronics in Defence and Security*, (Paris, France), Feb. 2016.

[15] J. Uijlings, K. E. A. Sande, T. Gevers, and A. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, pp. 154–171, 09 2013.

[16] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *CoRR*, vol. abs/1501.00092, 2015.

[17] Wikipedia contributors, "Bicubic interpolation — Wikipedia, the free encyclopedia," 2019. [Online; accessed 24/06/2019].

[18] Tensorflow. URL: https://www.tensorflow.org. [Online; accessed 10/03/2019].

[19] PyTorch. URL: https://www.pytorch.org. [Online; accessed 10/03/2019].

[20] Keras. URL: https://www.tensorflow.org/guide/keras. [Online; accessed 10/03/2019].

[21] C. de Visió per Computador (CVC). URL: http://www.cvc.uab.es/. [Online; accessed 18/06/2019].

[22] Github. URL: https://github.com/jordiori/tfg. [Online; accessed 30/06/2019].

[23] J. Hui, "map (mean average precision) for object detection," March 2018. URL: https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173. [Online; accessed 17/05/2019].

[24] Wikipedia contributors, "Peak signal-to-noise ratio — Wikipedia, the free encyclopedia," 2019. [Online; accessed 24/06/2019].

[25] G.-S. Xia, X. Bai, J. Ding, Z. Zhu, S. Belongie, J. Luo, M. Datcu, M. Pelillo, and L. Zhang, "Dota: A large-scale dataset for object detection in aerial images," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[26] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *International Journal of Computer Vision*, Sep 2004.

[27] Y. LeCun and C. Cortes. URL: http://yann.lecun.com/exdb/mnist/. [Online; accessed 18/06/2019].

[28] W. Yang, X. Zhang, Y. Tian, W. Wang, and J. Xue, "Deep learning for single image super-resolution: A brief review," *CoRR*, vol. abs/1808.03344, 2018.

[29] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," *CoRR*, vol. abs/1506.02025, 2015.

APPENDIX

A.1 Definitions, acronyms and abbreviations

Term	Description
DL	Deep Learning
CNN	Convolutional Neural Network
UAV	Unmanned Aerial Vehicle
SOTA	State Of The Art
RCNN	Region-CNN
SRCNN	Super Resolution CNN
GPU	Graphical Processing Unit

A.2 Figures

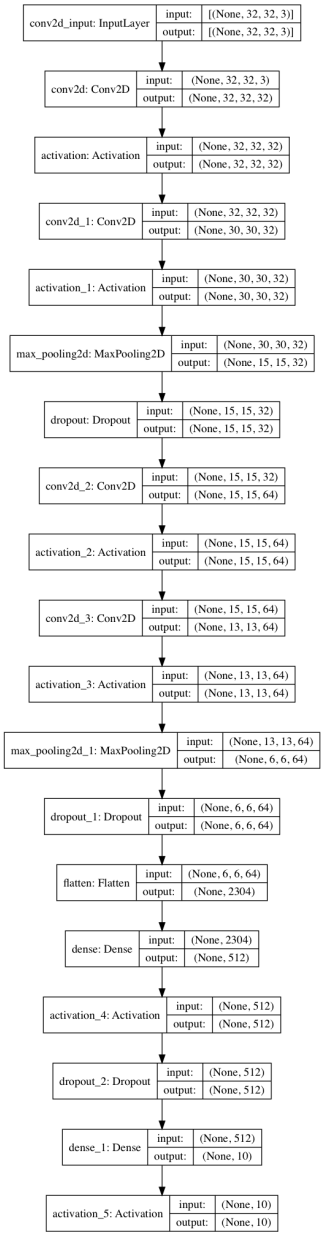


Figure 13: Architecture of our CNN baseline model.

A.3 Detection examples

The green boxes are the ground-truth and the red ones are the detected objects.

Figure 14: Selective Search + CNN baseline on P0005 image.

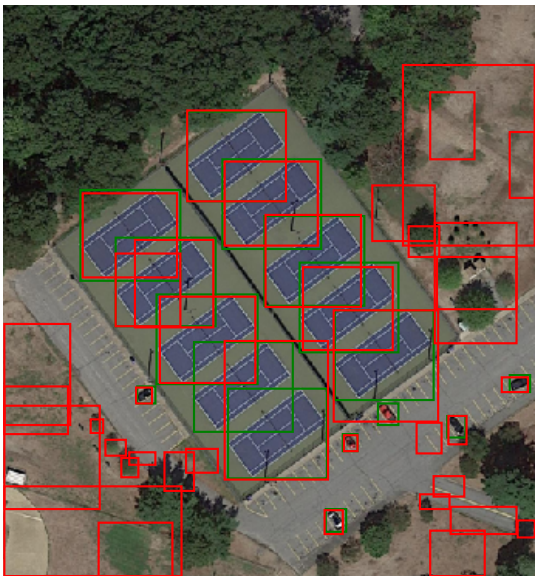


Figure 15: Selective Search + CNN baseline on P0241 image.