Detection of cities vehicle fleet using YOLO V2 and aerial images

H.Lechgar[1], H. Bekkar [1], H. Rhinane [1]

[1] Faculté des Sciences Ain Chock, Université Hassan II, Casablanca, Maroc- (h.lechgar, hajar.bekkar, h.rhinane)@gmail.com

KEY WORDS: Deep Learning, Convolutional neural network (CNN), Yolo, Dataset, Artificial intelligence (AI), Detection, Cars.

ABSTRACT:

Recent progress in deep learning methods has shown that key steps in object detection and recognition can be performed with convolutional neural networks (CNN). In this article, we adapt YOLO (You Only Look Once) to a new approach to perform object detection on satellite imagery. This system uses a single convolutional neural network (CNN) to predict classes and bounding boxes. The network looks at the entire image at the time of the training and testing, which greatly enhances the differentiation of the background since the network encodes the essential information for each object. The high speed of this system combined with its ability to detect and classify multiple objects in the same image makes it a compelling argument for use with satellite imagery.

1. INTRODUCTION

Getting a robust and accurate location of any object in a given input image, combined with a correct label, is a key element in solving many automation tasks. In most cases, the location is only a small part of the entire pipeline, which requires very high precision, in order to minimize the error propagated through the remaining process.

Object detection and recognition is one of the most important areas of computer vision because it is a key step for many applications including smart city, smart home, surveillance and robotics.

In this article, we focus on detecting vehicles from high-resolution satellite imagery. The detection of this object (the vehicles) can be used as part of the modelling and optimization of road networks. When cities prosper and begin to exceed their infrastructure, traffic often stops. This solution can be used to determine the critical density within a city's road network and create computer traffic models that can be simplified to the point where real-time traffic management becomes feasible.

In this work, we use deep learning methods, based on convolutional neural networks (CNN) (Yoon et al., 2015), in particular the YOLO model which is a recent fast and open source CNN ("Common architectures in convolutional neural networks.," n.d.), which will be modified to improve its performance to detect vehicles in satellite imagery. The model sees the entire image at the time of the training and test, which greatly improves the differentiation of the background since the network allows us to detect several objects in a scene. This is described in more detail later.

The rest of this article is organized as follows. Section (2) provides an overview of Deep Learning and Yolo. Section (3) covers the equipment used, and the methodology followed for the training and validation of the model. The results obtained are then discussed in Section (4), followed by a conclusion in Section (5).

2. BACKGROUND

Deep learning is an automatic learning technique that teaches computers to do what comes naturally to humans. In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound (Goodfellow et al., 2016). Deep learning models can achieve state of the art accuracy, sometimes exceeding human performance. The models are formed using a large number of tagged data and neural network architectures that contain many layers. Most deep learning methods use neural network architectures, which is why they are often called deep neural networks. The term "deep" usually refers to the number of hidden layers in the neural network. Traditional neural networks contain only two or three hidden layers, while deep networks can contain up to 150 hidden layers (Bengio, 2009). Deep learning models are formed using large sets of tagged data and neural network architectures that learn features directly from the data without the need to manually extract the features.

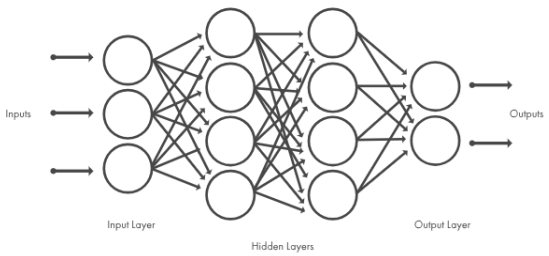

Figure 1: Neural networks, which are organized in layers consisting of a set of interconnected nodes.

Convolutional neural networks (CNN or ConvNet) One of the most popular types of deep neural networks (Khan et al., 2018). CNN convolves learned functions with input data and uses 2D convolutional layers, which makes this Figure (2) architecture well suited for processing 2D data, such as images. CNN contains four components:

a) Convolutional layers: Apply a specified number of convolution filters to the image. For each subregion, the layer performs a set of mathematical operations to produce a unique value in the output feature map. The convolutional layers then apply a ReLU activation function to the output to introduce nonlinearities into the model.

b) Rectified linear unit layer (ReLU) applies an activation function per element, such as the maximum threshold (0, x) to zero, where x is the input of a neuron (Nair and Hinton, 2010). ReLU layers don't change the volume size as convolutional layers do.

c) Pooling layers: Sub-sample the image data extracted by the convolutional layers to reduce the dimensionality of the feature map and processing times.
A commonly used grouping algorithm is maximum grouping, which extracts sub-regions from the map (for example, 2x2 pixel tiles), retains its maximum value, and rejects all other values
d) Dense layers (fully connected): Perform a classification on the entities extracted by the convolutional and subsampled layers by the grouping layers. In a dense layer, each node of the layer is connected to each node of the previous layer. Typically, a CNN consists of a stack of convolutional modules that perform feature extraction. Each module consists of a convolutional layer followed by a grouping layer. The last convolutional module is followed by one or more dense layers that perform the classification. The final dense layer in a CNN contains a single node for each target class in the model (all possible classes that the model can predict), with a softmax activation function (Pancioni, 2018) to generate a value between 0 and 1 for each node (the sum of all these softmax values are equal to 1).

We can interpret the softmax values for a given image as relative measures of the probability that the image falls into each target class.

Yolo: (you only look once) is a state of the art real-time object detection system (Redmon and Farhadi, 2017a) that uses neural networks to detect objects in images.
Earlier detection systems convert classifiers or locators to perform the detection. They apply the model to an image at multiple locations and scales. High-scoring regions of the image are considered detections.

YOLO uses a different approach by applying a single neural network to the complete image. The network divides the image into regions and predicts the boundaries and probabilities for each region.

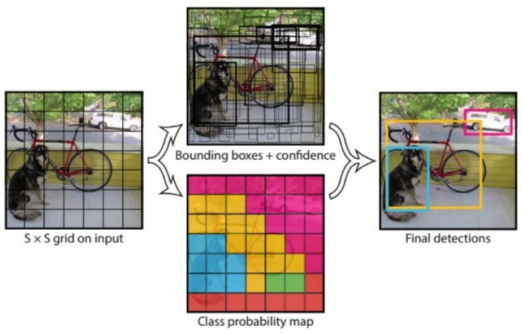

Figure 2: YOLO image processing overview *(Redmon and Farhadi, 2017a)*

3. DATA AND MATERIELS

We propose the YOLO approach for the detection of objects from aerial images. We choose cars as a target because vehicles are ubiquitous and automatic recognition of vehicle models is very useful for monitoring and analysis. Our method uses tagged detection images to learn how to precisely locate vehicles. The images are extracted from the VEDAI dataset (Popov, 2018). This dataset contains the largest number of tagged aerial images and is the one commonly used in detection work similar to our project. We improve the algorithm of YOLO basic detection system to form a fast and robust detector, capable of recognizing fine objects on high-resolution images.

3.1 Dataset

To form a deep neural network object detector requires a large amount of data; hence the coupling of deep learning with big data.

Since there are no large sets of standardized public data containing many car collections on high-resolution satellite images, it has been difficult to find a dataset that meets the criteria of our project:

- The images must be copyright free or at least freely usable within the computer vision community, which is a strong criterion because the production of aerial images is usually expensive.
- The number of images should be large enough to represent the needs of the model,
- High-resolution images.

After reviewing all possible sources of images that would meet the previous requirements, we decided to retain the satellite imagery of VEDAI (Vehicle Detection in Aerial Imagery), a new database to evaluate the detection of small vehicles in aerial images. The dataset includes different categories of vehicles for a total of more than 1500 images. The images are divided into two different categories, one for training and the other for the test, corresponding to two different sizes of images (1024×1024 and 256×256). The dataset is published with precisely defined divisions and image metrics, allowing for reliable evaluation. The dataset and annotations are publicly available.

For convenience, annotations are provided as a set of independent files, one per image, with each file named according to the name of the image (for example, the image annotation file 00000010.png is named 00000010.txt). They contain exactly the same information as the main annotation file. A set of annotations is provided for the 256×256 images contains 2000 images for the training and 120 images for the test, and another for the 1024×1024 images contains 190 images for the training and 37 images for the test. An example of an annotated image is given in Figure (3).

Figure 3: Illustration of the annotation

3.2 Ground truth

Ground Truth is factual data that has been observed or measured, can be analysed objectively and has not been deduced (Baier and Wernecke, 2006). Indeed, in image recognition technologies, field truth is information obtained by direct visual examination, especially when used to verify or calibrate an automatic recognition system (Redmon and Farhadi, 2017a).

In our case, for object detection, Ground Truth is a dataset describing the position, size, and class of objects in our VEDAI dataset. With position and size information, we can generate what's called a bounding box.

An enclosing box is an area defined by two points plus a width and height or four points generating a rectangle.

The shape of the rectangle is somewhat of a limitation for the YOLO detection model because most real objects don't usually have a rectangle shape, but it simplifies the calculation and detection.

3.3 Material descriptions

The YOLO model requires an important computing power. High performance GPUs have an efficient parallel architecture for model learning, combined with clusters or cloud computing, it allows us to reduce network-training time from a few weeks to a few hours, the reason why I opted for the AMAZON AWS service and with a very specific configuration:

- **Environnement** : Amazon AWS
- **Instance type** : p2.xlarge
- **System** : Windows Server 2016 – X64
- **Processor** : 4 CPU Intel Xeon E5-2686 V4 2.30Ghz
- **RAM** : 61Go
- **GPU** : 1 GPU NVIDIA TESLA K80 with 12Go in memory
- **HDD** : 100Go

The training framework we used to train YOLO is Darknet (Redmon and Farhadi, 2017b) which is an open source neural network framework written in C and CUDA that supports CPU and GPU computation.

3.4 Methods

a. Dataset training

The YOLO training algorithm is configured in this way, for each subdivision output the model calculates:

- '**Avg IOU**' is a great metric to determine how accurately our model detected a certain object.
- '**Avg recall**' is defined in the code as *recall/count*, and therefore a metric for how many positives (objects to be detected) YOLO has detected on the total amount of positives in this subdivision.
- '**Count**' is the quantity of positives present in the current subdivision of the images.
- The number of iterations
- The total loss.
- The average loss (error), which should be as low as possible. As a rule of thumb, once this reaches below 0.060730 avg, we can stop the training.
- The total time spent processing this batch.

- The number of iterations
- The total loss.
- The average loss (error), which should be as low as possible. As a rule of thumb, once this reaches below 0.060730 avg, we can stop the training.
- The total time spent processing this batch.

YOLO saves the weights in the backup file for every 100 iterations.

b. Test

The test part consists of choosing some images from the test file that has already been prepared; images that have not been used in the training part, to highlight the performance of our detection configuration

4. RESULTS AND DISCUSSION

In this article, we create a robust and fast detector capable of recognizing fine grain vehicles, from aerial imagery, using the YOLO model architecture.

To study the performance of our detector, we trained and tested the model on two resolutions of different images (256×256) and (1024×1024).

The dataset used contains more than three thousand images divided as follows: **70%** of the images as training data and the remaining **30%** as test data. An exact number of images of the training and test data for each resolution of the images.

We have made changes to the YOLO template configuration settings, namely, the correct setting of the input size of the images in the first convolution layer; the number of classes, as well as the output dimension of the last fully connected layer that has also been modified according to the number of classes.

To simplify the task, each size of the dataset is formed independently.

In the training section, the performance of the YOLO model is evaluated from the value of the (Avg loss). Once the average loss no longer decreases, at many iterations the training can be stopped.

During the training process, we find that for images with (256×256) resolution, the network begins to converge after 3208 iterations and we report the accuracy of the average loss at 20,000 iterations with a value of 0.22 avg, and training lasted three days for the two thousand images; however, for images with (1024×1024) resolution, network convergence started after 802 iterations and accuracy of average loss was reported after 22.000 iterations with values of 0.05 avg, and training lasted one week for the one thousand two hundred images, This may be due to the large size of the images. The evolution of the average loss during the iterations of the learning process, for the two sizes of imageries given in Figure (4).

We can see that the union intersection is better for both imagery sizes since it is greater than 50%, as well as the positive object detection rate for each subdivision exceeds 50%; therefore, the performance of the system in terms of object detection is acceptable.

The Figures (5) and (6) show the results of vehicle detection. In figures (5) left and (6) left, we present the results of the images with resolution (1024×1204), the model seems very powerful since 91% of test vehicles are detected without any ambiguity.

The figures (5) right and (6) right show the detection of images with resolution (256×256). We notice that the model detects only about 75% of test vehicles. We think that is due the fact that YOLO was not able to extract and learn enough features from low resolution as it did for higher resolution images.

The figure (7) shows the accuracy of the detection in terms of true positive rate (TPR), false positive rate (FPR) and false negative rate (FNR). From this figure, we can see that the proposed algorithm can work well in this dataset providing more than 75% of TPR and about 5% of FNR whatever the size of the satellite images.

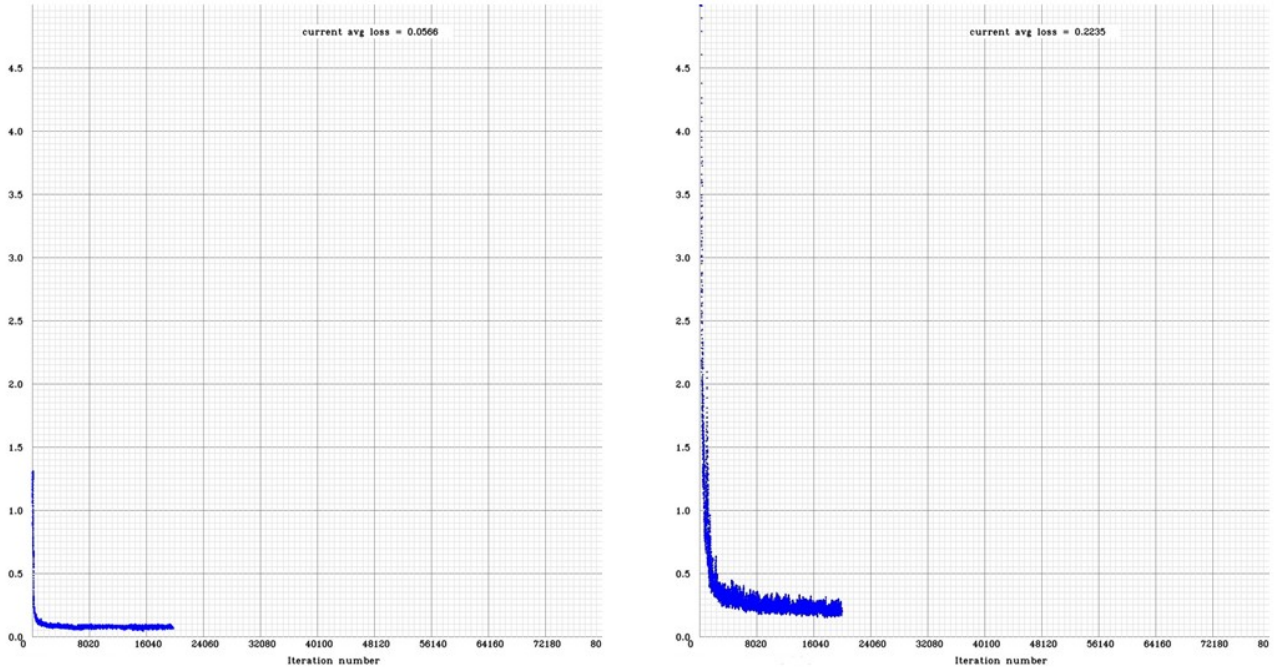

Figure 1 : Left: The evolution of the average loss related to the number of iterations, for the images with resolution (1024 × 1024). Right: The evolution of the average loss related to the number of iterations, for the images with resolution (256 × 256)

Figure 5: Left: Yolo test for Imagery with resolution (1024 × 1024). Right: Yolo test for Imagery with resolution (256 × 256)

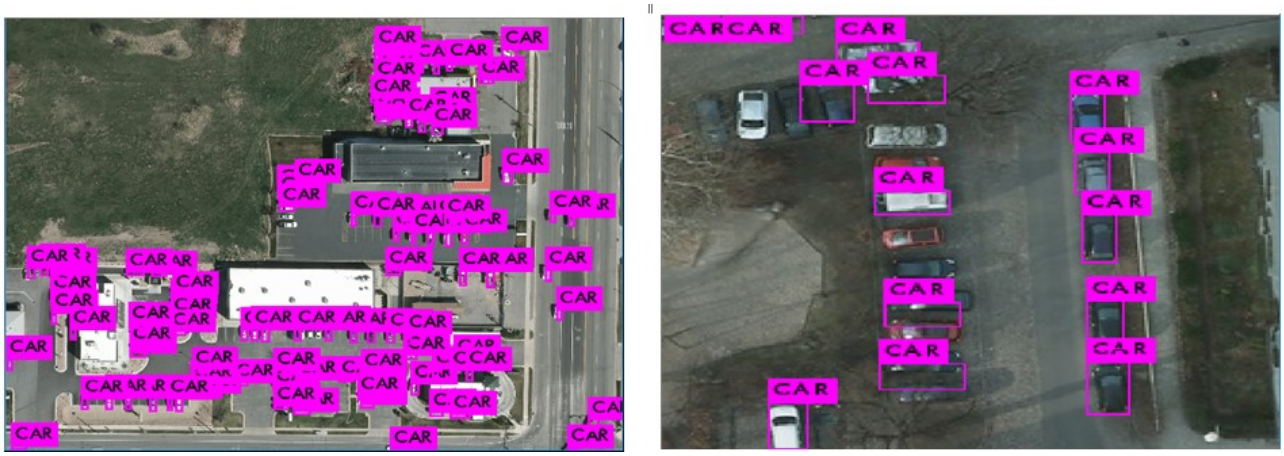

Figure 6: Left: Yolo test for Imagery with resolution (1024 × 1024). Right: Yolo test for Imagery with resolution (256 × 256)

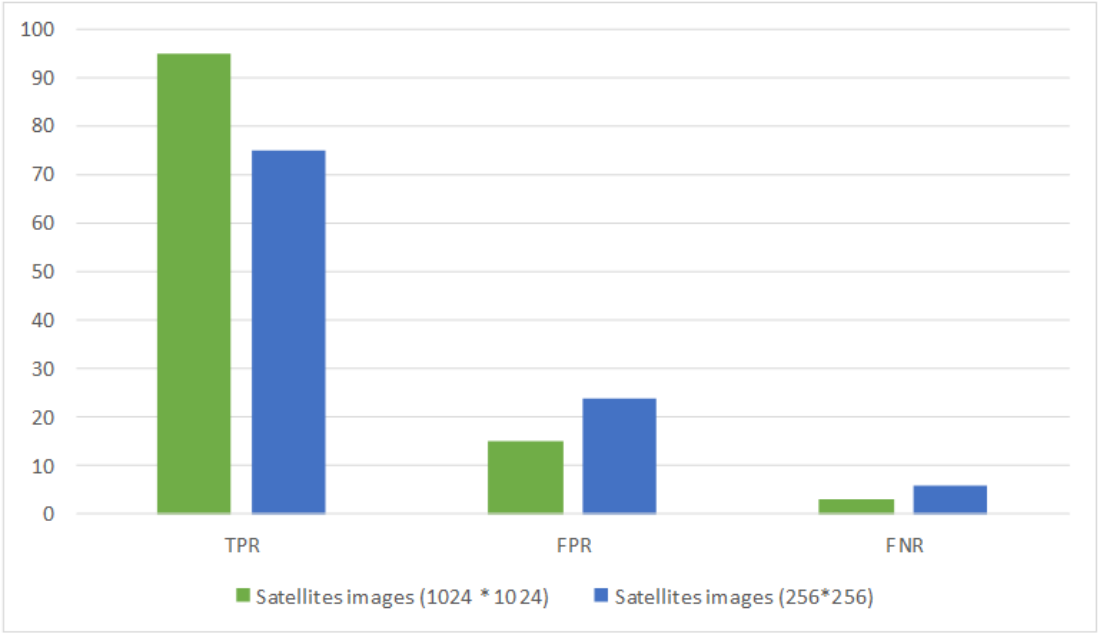

Figure 7: Statistical evolution of the results obtained

5. CONCLUSION

In this article, we present YOLOv2 as a new state of the art detection system, fast compared to other detection systems, it can be run at a variety of image sizes to provide a compromise between speed and precision.

The YOLO model algorithm has been trained and refined for each step so that they are robust under different conditions (e.g, variety of imaging resolution).

The results show that the trained system is capable of detecting very fine objects in challenging environments such as partial occlusion and low illumination, which is a practical candidate for many applications in the field of artificial intelligence.

ACKNOWLEDGEMENTS

Our thanks go to VEDAI publisher for sharing free aerial images of vehicles.

REFERENCES

Baier, D., Wernecke, K.-D., 2006. Innovations in Classification, Data Science, and Information Systems: Proceedings of the 27th Annual Conference of the Gesellschaft für Klassifikation e.V., Brandenburg University of Technology, Cottbus, March 12-14, 2003. Springer Science & Business Media.

Bengio, Y., 2009. Learning Deep Architectures for AI. Now Publishers Inc.

Common architectures in convolutional neural networks. [WWW Document], n.d. URL https://www.jeremyjordan.me/convnet-architectures/ (accessed 11.8.18).

Khan, S., Rahmani, H., Shah, S.A.A., Bennamoun, M., 2018. A Guide to Convolutional Neural Networks for Computer Vision. Morgan & Claypool Publishers.

Nair, V., Hinton, G.E., 2010. Rectified linear units improve restricted boltzmann machines, in: Proceedings of the 27th International Conference on Machine Learning (ICML-10). pp. 807–814.

Pancioni, L., 2018. Artificial Neural Networks in Pattern Recognition. Springer.

Popov, N., 2018. vedai dataset for darknet. Contribute to nikitalpopov/vedai development by creating an account on GitHub.

Redmon, J., Farhadi, A., 2017a. YOLO9000: better, faster, stronger. arXiv preprint.

Redmon, J., Farhadi, A., 2017b. YOLO9000: better, faster, stronger. arXiv preprint.

Yoon, Y., Jeon, H.-G., Yoo, D., Lee, J.-Y., So Kweon, I., 2015. Learning a deep convolutional network for light-field image super-resolution, in: Proceedings of the IEEE International Conference on Computer Vision Workshops. pp. 24–32.