FEDERAL UNIVERSITY OF SANTA MARIA
TECHNOLOGY CENTER
TELECOMMUNICATIONS ENGINEERING

DEEP LEARNING-BASED ORIENTED OBJECT DETECTION IN REMOTE SENSING IMAGERY: YOLOV7-OBB

COURSE CONCLUSION PAPER

Pietro Terra Pizzutti dos Santos

Santa Maria, RS, Brazil

2023

DEEP LEARNING-BASED ORIENTED OBJECT DETECTION IN REMOTE SENSING IMAGERY: YOLOV7-OBB

Pietro Terra Pizzutti dos Santos

Course Conclusion Paper presented to Telecommunications Engineering Undergraduate Course, Telecommunications Concentration Area, Federal University of Santa Maria (UFSM, RS), as partial requirement for obtaining the degree of

Bachelor of Telecommunications Engineering

Advisor: Prof. Dr. Natanael Rodrigues Gomes

Santa Maria, RS, Brazil

2023

Terra Pizzutti dos Santos, Pietro

Deep Learning-Based Oriented Object Detection in Remote Sensing Imagery: YOLOv7-OBB / by Pietro Terra Pizzutti dos Santos. – 2023.

86 f.: il.; 30 cm.

Advisor: Natanael Rodrigues Gomes

Course Conclusion Paper - Federal University of Santa Maria, Technology Center, Telecommunications Engineering, RS, 2023.

1. Course Conclusion Paper. 2. Remote sensing imagery. 3. YOLOv7. 4. Object detection. 5. Deep learning. 6. Oriented bounding-box. 7. DOTA dataset. I. Rodrigues Gomes, Natanael. II. Título.

E-mail: pietro.pio@outlook.com

Federal University of Santa Maria
Technology Center
Telecommunications Engineering

The Examining Committee, signed below,
approves the Course Conclusion Paper

DEEP LEARNING-BASED ORIENTED OBJECT DETECTION IN REMOTE SENSING IMAGERY: YOLOV7-OBB

elaborated by
Pietro Terra Pizzutti dos Santos

as partial requirement for obtaining the degree of
Bachelor of Telecommunications Engineering

EXAMINING COMMISSION:

Natanael Rodrigues Gomes, Dr.
(President / Advisor)

Daniel Fernando Tello Gamarra, Dr. (UFSM)

João Baptista dos Santos Martins, Dr. (UFSM)

Santa Maria, January 2023.

ACKNOWLEDGMENTS

I start my acknowledgments by those responsible for giving me life notions which have built the core of what I have become, decisive for choosing the paths trodden until today. I thank my entire family for supporting me, both emotionally and economically. Giving me the gift of freedom and free will in all my choices in my career as student to become an engineer.

To all my professors in the Federal University of Santa Maria who did an extraordinary job, who far beyond teaching me the basics towards the more specialized science matter, opened my eyes to the vastness, usefulness and amazing capabilities of mathematics and physics applied to real systems and problems. Which has instigated me to keep exploring further and further.

This admiration by the engineering was probably quickly noticed by the commission deciding the students who were going to France through the BRAFITEC exchange program, in 2019, when they chose me to participate in the most magnificent experience of my life. Two and a half years living in France and a double degree achieved. A special thanks to the program advisor Dr. João Baptista dos Santos Martins and the Electronics Department director of my French course, Dr. Nathalie Deltimple, who were always present to help me in all the different steps and transition which needed a ton of papers.

Dr. Natanael Rodrigues Gomes has helped me with all the exoneration of disciplines that I had already done in Bordeaux INP and he was fundamental to the exchange of information with the course. He proposed the subject of this paper and helped me a lot structuring it and choosing the right approach to address the problems on the project.

Finally, I would like to thank my professors at Bordeaux INP, who also did an awesome work instigating on me the challenges and enhancements of my knowledge in electronics, which made me specialize myself in image and signal processing, subject of this very paper.

“We must believe that we are gifted for something, and that this thing, at whatever cost, must be attained.”

— MARIE CURIE

ABSTRACT

Course Conclusion Paper
Bachelor of Telecommunications Engineering
Federal University of Santa Maria

DEEP LEARNING-BASED ORIENTED OBJECT DETECTION IN REMOTE SENSING IMAGERY: YOLOV7-OBB

AUTHOR: PIETRO TERRA PIZZUTTI DOS SANTOS
ADVISOR: NATANAEL RODRIGUES GOMES
Defense Place and Date: Santa Maria, January 25st, 2023.

Remote sensing (RS) is the act of processing and extracting meaningful features about the ground and objects observed at a distance, usually from a much higher position from aircraft and satellites. Due to the large field of coverage in RS imagery, object detection in these images can be really useful, gathering a broad and concise notion of the objects present in certain areas. Due to their great capability of assimilating intricate patterns, Deep Learning (DL) models have achieved state-of-the-art (SOTA) performance in computer vision tasks. In this project, an extensive research is conducted on current DL-based object detection models and a suitable model, YOLOv7, is chosen to serve as a baseline for modifications to enable a high performance oriented bounding-box (OBB) detector in RS imagery. In supervised DL models, their final performance is very dependent on the quality of their training. To improve it, large datasets covering the specific task are pursued, converging to the use of DOTA dataset. Moreover, the concept of transfer learning is employed to allow the use of a pre-trained model on a very large dataset with different tasks. The final model is evaluated on common object detection metrics, such as the confusion matrix, precision, and recall curves. They validate the detector, capable of identifying 16 object classes with SOTA performance: high accuracy, fast and with the latest oriented bounding-box. Comparing the confusion matrices of the developed model and YOLOv5-OBB (KAIXUAN, 2022), for instance, it correctly identifies with a probability of 0.97, 0.89, 0.67 and 0.67% the following classes: plane, baseball diamond, bridge and ground track field. Meanwhile the YOLOv5-OBB obtains 0.96, 0.83, 0.6 and 0.6% for the same respective classes. Another interesting point is the reduction from 0.73 to 0.69% in the probability of mistaking the background for a small-vehicle. The model can further be trained on custom datasets for detection in agriculture, livestock, militarily, etc., bringing implications for many areas and activities. The repository containing all the codes used and developed in this project is available at (SANTOS, 2022).

Keywords: Machine learning; Deep learning; Remote sensing imagery; Object detection; Oriented bounding-box; YOLOv7; DOTA dataset..

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

ML	Machine Learning
RS	Remote Sensing
SOTA	State-of-the-art
DL	Deep Learning
SAR	Synthetic Aperture Radar
InSAR	Interferometric SAR
ANN	Artificial neural network
NN	Neural Network
VAE	Variational Auto-encoder
KL	Kullback-Leibler
RL	Reinforcement Learning
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short Term Memory
GAN	Generative Adversarial Network
ROI	Region of Interest
FC	Fully Connected
RPN	Region Proposal Network
SSD	Single Shot MultiBox Detector
YOLO	You Only Look Once
mAP	mean Average Precision
FPN	Feature Pyramid Network
MS COCO	Microsoft Common Objects in Context
TL	Transfer Learning
IoU	Intersection over Union
PR	Precision-Recall
AP	Average Precision
AUC-PR	Area Under the Precision-Recall Curve
HBB	Horizontal Bounding Box
OBB	Oriented Bounding Box
FCOS	Fully Convolutional One-Stage Object
SkewIoU	Skew Intersection over Union

CSL	Circular Smooth Label
NMS	Non-Maximum Suppression
GPU	Graphics Processing Unit
VM	Virtual Machine
UAV	Unmanned Aerial Vehicle
NDVI	Normalized Difference Vegetation Index
NIR	Near-InfraRed
BFLOP	Billion FLoating-point OPerations
EOE	Exchangeability of Edges

CONTENTS

1 INTRODUCTION

This chapter motivates the reader about this paper's subject by presenting its objectives at section 1.1, followed by the justifications supporting the work done, at section 1.2, and finally, at section 1.3, it exposes how the rest of the content is addressed.

1.1 Objectives

To get started, the objective is to review from initial and more traditional machine learning-based object detectors to state of the art (SOTA) ones, comprehending their evolution and mechanisms of improvement. With the required knowledge, it is then chosen the most adequate architecture and model to be implemented and provide a groundwork that is further modified, based on SOTA methods and customized to work with remote sensing imagery.

With the obtained model, modern training methods, such as transfer learning concept and some data augmentations, are researched and applied to the model training. The appropriated dataset, with remote sensing images and object detection task, is also a target of research in this paper. The most common metrics and losses in the literature, for machine learning-based models, are evaluated throughout its training.

The final results are compared with other models available to the same task and dataset. Their performance and possible constraints are all considered to generate a very detailed comparison between them. It is identified the improvements and issues of the model. The impacts that this project brings and its many applications are then debated. Allowing a scientific regard to keep perfecting models and acquiring even higher performance in future works.

1.2 Justification

Detection of objects in remote sensing images has an enormous variety of application: checking for the presence of varied objects of interest in determined area, such as facilities, villages, airports, the traffic of cars and boats, plantations, and multi-class object counting, etc. Serving many purposes such as surveillance, defense, agriculture, environmental protection, search-and-rescue, etc.

However, as it can be imagined and it is further addressed in this paper, at section 2.6, establishing models to perform well the object detection task in remote sensing imagery is very

complicated. Since the success of AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC-2012) (RUSSAKOVSKY et al., 2015), it has been perceived the deep learning high capacities with respect to object detection.

Since then, the evolution of deep learning-based object detector models has never stopped and more specific applications, such as in remote sensing, started employing them. Due to relative recent development of deep learning approaches, this field is yet replete of uncertainties and possibilities to improvement. By constraining the object detectors to remote sensing imagery, there are many modifications passive of great enhancement.

1.3 Monograph organization

To allow a good development in machine learning (ML)-based object detection in remote sensing, it is essential to address each individual part that constitutes it at first. Chapter 2 introduces the necessary knowledge about remote sensing: what it is, sensor types, different resolutions, object detection in remote sensing, etc. Chapter 3 brings the fundamental concepts of deep learning. The literature review, in chapter 4, addresses early object detectors and their evolution; and SOTA deep learning (DL)-based object detectors in remote sensing imagery.

After creating the foundations from the basic knowledge to move on, chapter 5 presents the proposed methodology. It goes throughout many implementations, mainly driven by the results seen in chapter 6.

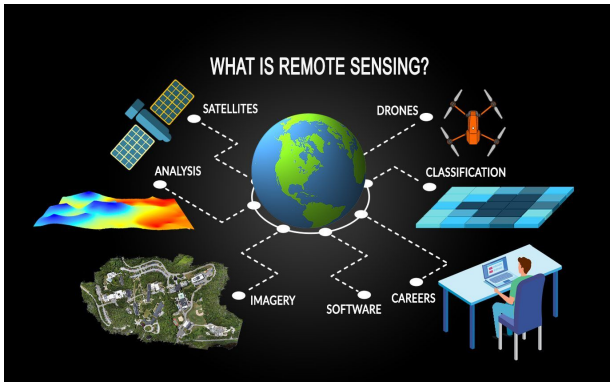
Finally, chapter 7 recapitulates different aspects from this paper and concludes about the achieved results. Bringing to light what was accomplished and opening up many possibilities for future work.

2 REMOTE SENSING

Remote sensing is extracting relevant information about regions and objects through images, i.e., without physically interact with them at some distance. It is demanded in different activity fields, e.g., most Earth science disciplines, such as meteorology, geology, hydrology, etc; commercial and economic prospecting; military intelligence, etc.

The images processed to extract information are called remote sensing images. Usually these images are acquired through satellite or aircraft-based sensor technologies.

Fig. 2.1 – Global vision of remote sensing



Source: Adapted from (GISGEOGRAPHY, 2021).

From the signal received at the sensor and knowing the emitted signal beforehand, it is possible to make different estimates of the surface or the object.

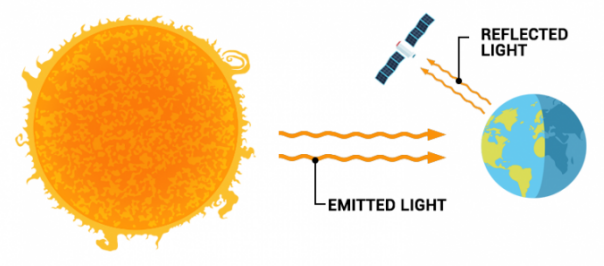
2.1 Sensor Types

Depending how the emitted signal is created, we can define two types of remote sensing sensors (GISGEOGRAPHY, 2021):

- Passive sensor;
- Active sensor.

For passive sensors, the emitted signal that is later reflected and measured by the sensor is not produced by itself, it comes from a different source. The most common radiation source is the sunlight.

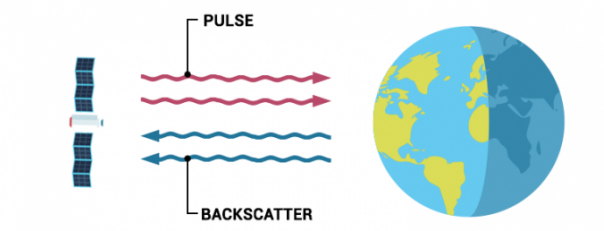
Fig. 2.2 – Principle of operation of passive sensors



Source: Adapted from (GISGEOGRAPHY, 2021).

Active sensors, however, produce their won signal that is back-scattered at the region or object they illuminate.

Fig. 2.3 – Principle of operation of active sensors



Source: Adapted from (GISGEOGRAPHY, 2021).

The active remote sensing is very useful whenever the application needs a synthetic signal due to some desired properties or it wants an independence from the external radiation source. For instance, whenever it uses sunlight it will be subject to the time of the day to gather new data. Since the active sensor emits its own radiation, there is no need of sunlight. Fig. 2.2 and 2.3 show both systems.

2.2 Platforms for sensors

Sensors might be attached at different platforms. The most common are satellites, airplanes and drones. Each of them has some advantages and disadvantages. Drones, UAVs, airplanes and helicopters have very high resolution imagery and are free to choose the most adequate path they will pass. However, they have small coverage extent, which is the advantage of low Earth orbit satellites, but have coverage limited to orbital paths and deal with cloud obstruction.

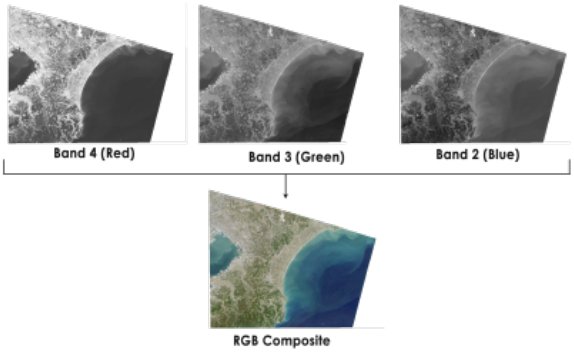
2.3 Remote Sensing Image

Remote sensing images have different resolutions (DIRECT, 2022):

- spectral;
- spatial;
- radiometric and
- temporal.

Spectral resolution refers to the bandwidth and the sampling rate the sensor captures. High spectral resolution has narrower bandwidth, e.g. 10 nm (DIRECT, 2022). Current *Landsat* collection has 7 bands, including several in the infrared spectrum, ranging from a spectral resolution of 0.7 to 2.1 μm. The *Hyperion* sensor on *Earth Observing-1* resolves 220 bands from 0.4 to 2.5 μm, with a spectral resolution of 0.10 to 0.11 μm per band (WIKIPEDIA, 2021a).

Fig. 2.4 – Bands composition



Source: Adapted from (GISGEOGRAPHY, 2021).

Spatial resolution refers to the surface coverage by one single pixel in the image, defining the smallest features that can be distinguished. Typically, pixels correspond to square areas ranging in side length from 50 cm to 1 km (GISGEOGRAPHY, 2021; WIKIPEDIA, 2021a).

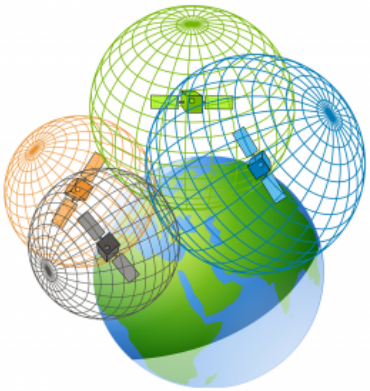
Fig. 2.5 – 3 different resolution from the same area

Source: Adapted from (GISGEOGRAPHY, 2021).

The radiometric resolution refers to the dynamic range or the total number of levels to discretize the signal that the sensor can record. A larger dynamic range for a sensor brings more discernible details in the image. The *Landsat* 7's sensor records 8-bit images, i.e., 256 unique gray values of the reflected energy, while *Ikonos-2* has an 11-bit radiometric resolution, i.e., 2048 grey values (DIRECT, 2022).

The temporal resolution refers to the time elapsed between consecutive images taken of the same area. Satellites depend on their orbits, they may be stationary or need to complete a revolution in their orbit to revisit the same area (DIRECT, 2022). Fig. 2.6 shows multiple satellites with their fields of sight.

Fig. 2.6 – Multiple satellites with their fields of sight



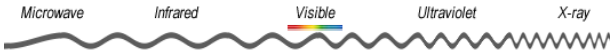
Source: Adapted from (GISGEOGRAPHY, 2021).

The temporal characteristic is relevant in time-series studies or those requiring an averaged or mosaic image as in deforesting and land use change monitoring.

2.4 Types of imagery

The electromagnetic spectrum may be decomposed into different spectral bands, from long wavelengths like microwave to short wavelengths like X-rays, as shown in Fig. 2.7

Fig. 2.7 – Spectral bands from long to shortwaves

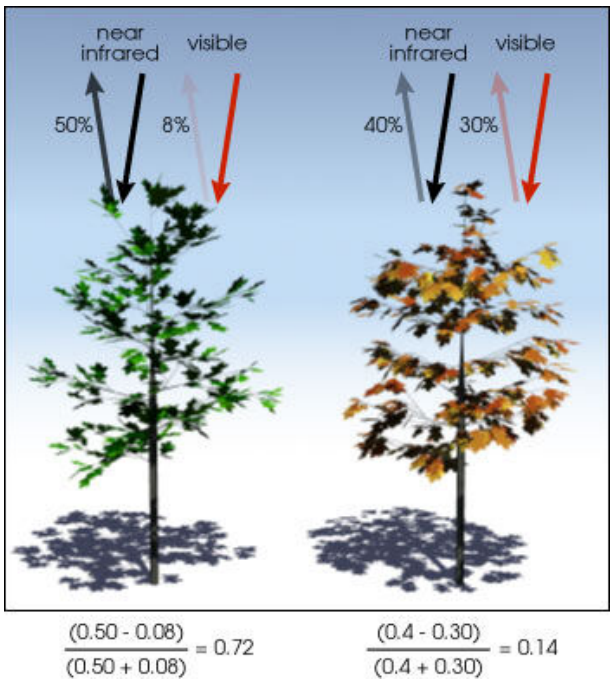


Source: Adapted from (GISGEOGRAPHY, 2021).

Human eyes are electromagnetic sensors at visible wavelengths, raging from red to violet. However, synthetic sensors can capture wavelengths beyond this range and thus, can see signals from different bands that have different interactions (responses) with soils and objects. For instance, the Normalized Difference Vegetation Index (NDVI) uses the response of both visible and near-infrared bands to classify vegetation. NDVI is a standardized way to measure healthy vegetation. The higher it is, the healthier is the vegetation, because the vegetation reflects more near-infrared (NIR) and green light compared to other wavelengths, but absorbs more red and blue light (GISGEOGRAPHY, 2021).

It can be seen at Fig. 2.8.

Fig. 2.8 – NDVI for different vegetation



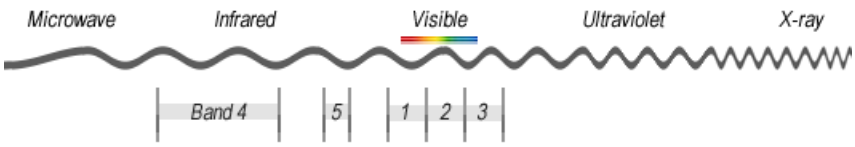
Source: Adapted from (GISGEOGRAPHY, 2021).

Based on the number of operation bands and their narrowness, there are 2 types of imagery:

- Multi-spectral and;
- Hyper-spectral.

Multi-spectral imagery is specified with few wide bands, as shown in Fig. 2.9.

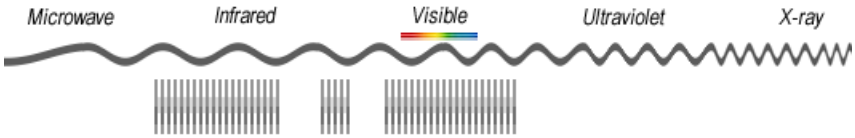
Fig. 2.9 – Multi-spectral imagery with wide bands at infrared, near infrared and visible



Source: Adapted from (GISGEOGRAPHY, 2021).

Hyper-spectral imagery is specified with several narrow bands, as shown in Fig. 2.10.

Fig. 2.10 – Hyper-spectral imagery with several narrow bands at infrared, near infrared and visible

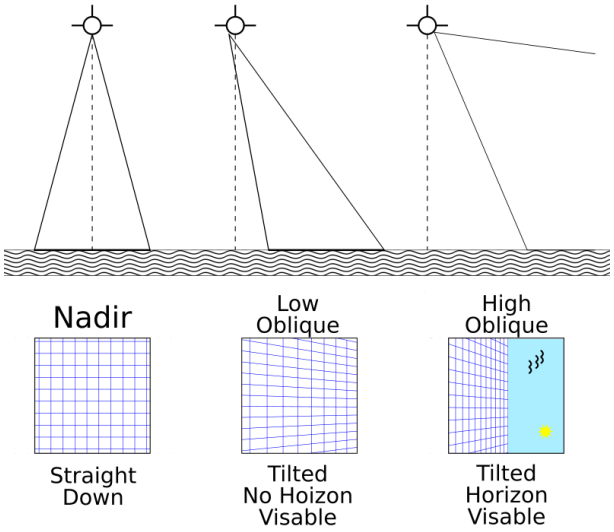


Source: Adapted from (GISGEOGRAPHY, 2021).

2.5 Angle of shooting

Depending on the angle of the optical axis, the image capture can be classified into one of the following three categories: Nadir, High Oblique and Low Oblique. Each of these captures the surface with different perspective deformations, as seen in the Fig. 2.11.

Fig. 2.11 – Nadir, High Oblique and Low Oblique with their respective perspective deformation



Source: Adapted from (DIETRICH, 2016).

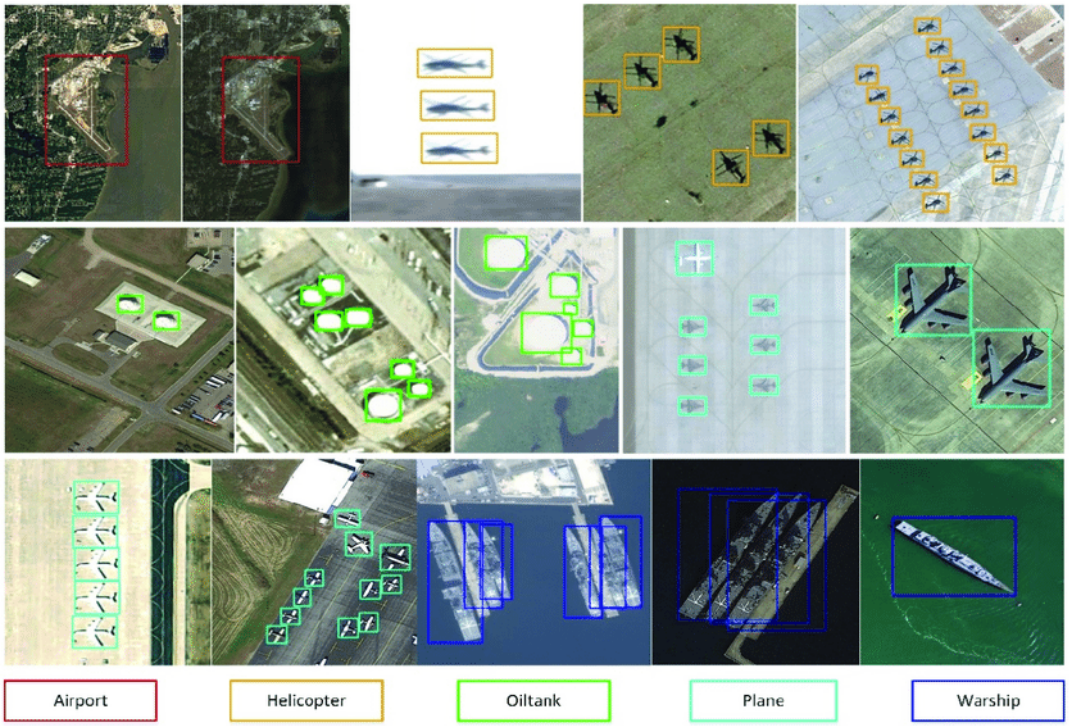
2.6 Object detection

Remote sensing possesses several tasks, e.g., parameter inversion, terrain surface classification, object detection, despeckling, topography of surface, multimodal data fusion, etc. This paper focuses in the object detection task.

Object detection can be unfolded into two sub-tasks: object area delimitation and its classification. Rather than other use cases, such as object analysis, it is not required very exact process to perform object detection. Basically, it is required a model to be attuned to the range of nuances to identify a certain class or category.

The object detection in remote sensing images follows the same principle as in a regular image, constraining the images under remote sensing imagery peculiarities. Fig. 2.12 shows an example of object detection in remote sensing images, where objects are identified with colored bounding boxes around them, and each color indicates the class the object belongs to.

Fig. 2.12 – Object detection in remote sensing images



Source: Adapted from (ZHUANG et al., 2019).

Establishing models capable to perform well this task, however, is not easy. Traditional methods tried to extract hand-crafted features based on expert domain knowledge to characterize objects, relying on template matching or through the use of traditional machine learning approaches, such as Support Vector Machines (SVMs) (MAHONY et al., 2019; SHIN; BALASINGHAM, 2017).

Unfortunately, besides those approaches being thoughly generalizable for different datasets, an object present in a remote sensing image may suffer from many deformations which may completely decharacterize the object from its regular defined view. Fading the heuristically engineered features to be unable to generically describe the objects under different modifications. However, deep learning methods extract discriminative features proved to overcome very well those aforementioned issues.

The success of AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC-2012) (RUSSAKOVSKY et al., 2015), with a large difference from the second position using traditional methods, has worldwide demonstrated that machine learning methods had a huge occult potential for image processing. Nowadays, these methods are becoming the leaders and preferred techniques applied

to the majority of image processing.

2.6.1 Remote sensing imagery peculiarities

As seen in section 2.4, different types of imagery can produce images with different number of channels. However, most applications encountered use the visible spectral band, confined to RGB channels.

The objects of interest are mostly very small and have only small nuances to differentiate them when compared to the whole image. Detection is a difficult task even for the human eye. Since most models work by learning selective filters, by the images that pass through them, the generated features are based on pixels. Thus, just a few pixels of a small object are not enough to allow this process to perform well, when compared to larger objects. Also, small objects are prone to be mislabeled, where their class may be omitted.

Another source of variations found specifically in remote sensing images is the perspective deformation and scaling implied by distance and angle of shooting.

3 DEEP LEARNING

The recent evolution of computational calculation capacity has loosen the employment and development of algorithms which require intense parallel calculation. Among other applications, deep learning has been astounding developed, with overwhelming success in many fields. Unlike conventional methods, deep learning-based ones commonly untangle themselves in hierarchical architectures, called deep neural networks. Whose name comes from a slight resemblance to the human brain neurons to recognize patterns from input data.

Deep learning algorithms have a main principle of encoding input data into effective feature presentations and then finding patterns in this latent representation to output predictions from specific tasks. In remote sensing field, deep learning-based methods have been adopted for a variety of tasks, including terrain surface classification, object detection, parameter inversion, despeckling, specific applications in Interferometric SAR (InSAR), and SAR-optical data fusion (ZHU et al., 2020).

Because of the diverse number of applications with many purposes and different data with different intrinsic meaning, they need to be differently handled by deep learning algorithms. Some major models and architectures have been developed to better correspond to determined applications and improve their capabilities. Following, some relevant deep learning concepts and models for image data processing are comprehensively reviewed.

3.1 Artificial neural network

Artificial neural network (ANN), or neural network (NN) for short in our context, may be seen as a system of many nodes, or neurons, in a certain topology, possessing pondered connections among them. The first layer of nodes is the NN input. Then, to be characterize as a deep learning model, there are hidden layers, which are more nodes with their own pondered connections. They may constitute from one to thousands of layers. Finally, there is the last layer, called output layer, which is again more nodes connected from the hidden layer and carries the expected result from the NN model.

Summarizing the 3 types of layers:

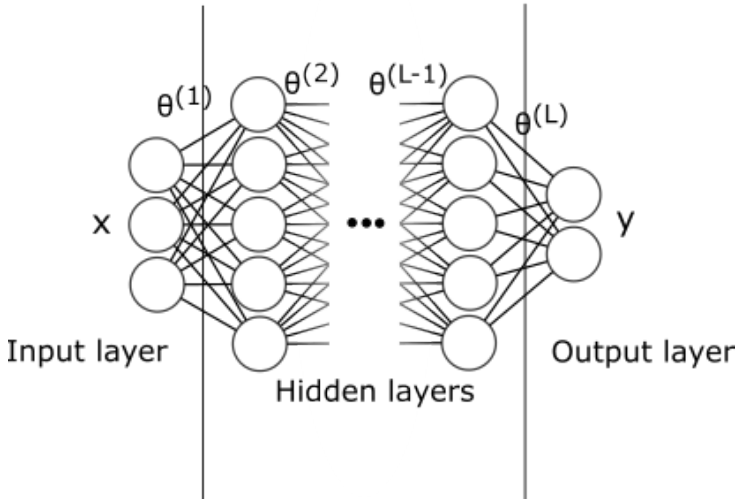
- input layer: input variables;
- hidden layers: layers of nodes between the input and output layers, varying from one to

multiple layers;

- output layer: output variables.

Fig. 3.1 illustrates them.

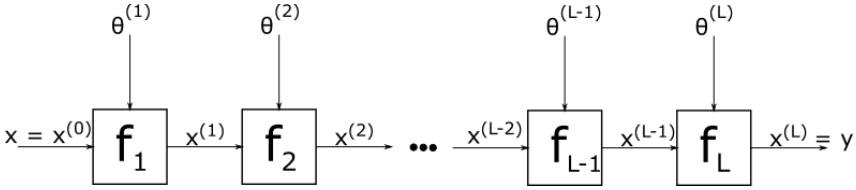
Fig. 3.1 – Input, hidden and output layers



Source: Author.

Which can be further represent as the block diagram in Fig. 3.2.

Fig. 3.2 – Block diagram of a NN



Source: Author.

Which can be mathematically written as Eq. 3.1.

$$\mathbf{y} = f(\mathbf{x};\boldsymbol{\theta}) = f_L\left(f_{L-1}\left(\ldots f_2\left(f_1\left(\mathbf{x};\boldsymbol{\theta}_1\right);\boldsymbol{\theta}_2\right)\ldots;\boldsymbol{\theta}_{L-1}\right);\boldsymbol{\theta}_L\right) \tag{3.1}$$

Where $\mathbf{x}$ is the input; $\mathbf{y} = f(\mathbf{x};\boldsymbol{\theta})$ is the output; $\boldsymbol{\theta}^{(l)}$ is the parameter set for the function f_l which represents the operations between all the nodes of layer l; and L is a hyperparameter that defines the number of layers in the NN (BOURMAUD, 2020).

Looking back on Eq. 3.1, a NN might be seen as a parametric sub-functions composition. By choosing the NN hyperparameters, some of its properties can be defined:

– **size** is the total number of nodes in the model;

– **width** is the number of nodes in a specific layer;

– **depth** is the total number of layers in a NN;

– **representational capacity** defines the type of functions that can be learned by the network configuration;

– **architecture** is the specific arrangement of the layers and nodes in the network.

The deeper the neural network, more and more intricate and sophisticated operations can be performed over the input data. However, although learning as much as possible from the data is the goal, supervised deep learning models, that is our case, can suffer from overfitting. More about this in the subsection 3.2, where the types of learning are presented.

3.2 Types of learning: supervised, unsupervised and reinforcement

Now the question that remains is, how these models learn? From the NN literature, for instance the great (GOODFELLOW; BENGIO; COURVILLE, 2016) book, it can be found 3 formal ways: supervised, unsupervised and reinforcement.

– **Supervised** is the most popular learning. However, input data in this method has the biggest constraint throughout them. It needs labeled datasets to be able to learn the objective function, i.e., for each input, there is also its right response, also called ground truth. Then, it can readjust itself to process the input and generate the output with the lowest loss. For this type of learning, one big challenge is to create a generalizable model for the data that hasn't participated in the training phase. It is vital that training data truly represents the true distribution of the entire dataset so it can handle all the unseen data (ZHU et al., 2020). The overfitting occurs when a model learns too much from the training data, including random noise that does not characterize the rest of the dataset. In practice this happens when the training phase is too long for the amount of available training data, passing too many times the same sample through the model and negatively affects the model's performance on new data (JOSEPH, 2020). Several techniques for minimizing this harm have been developed, e.g., batch normalization (IOFFE; SZEGEDY, 2015), which reduces internal covariate shift, and dropout (SRIVASTAVA et al., 2014), randomly dropping units along with their connections.

– **Unsupervised** is a type of learning that does not use labeled training data. Instead, it typically tries to find and extract patterns in the input data. In deep learning, generative models like autoencoders, variational auto-encoders (VAEs) (Kingma; Welling, 2013) and Generative Adversarial Networks (GANs) (GOODFELLOW et al., 2014) are some of popular techniques which work on unsupervised learning. Their primary goal is to generate output data from the same distribution as input data. The trick to learn this distribution is to also learn variance along with mean of latent representation at the encoder-decoder meeting point and add a Kullback-Leibler (KL)-divergence based loss term to the standard reconstruction loss function of the autoencoders (ZHU et al., 2020).

– **Reinforcement** reinforcement learning (RL) works similar to the human learning behavior. An agent, or multiples, to parallelize the learning, is/are created to interact with the environment and pondered rewards are given always that it goes towards the right path or accomplishes tasks as desired. Consequently, stimulating the agent to find the best actions for given states. In a classical RL, approximators are employed to calculate the probability of different actions in different states, generated by different types of neural networks (MNIH et al., 2015).

3.3 Supervised learning

Subsection 3.2 has presented the different types of learning. Now let us stick with the type of learning this paper's models use, the supervised. The term "learn" in supervised learning is the act of comparing its output results to the true answers (or labels) and then modifying itself to improve its results for the next inference.

As soon as a model is compiled, all the weights spread out within it are either randomly initialized or are precharged from a same model which has already been trained with some different dataset. The second option is the basis for transfer learning technique and will be discussed further. Either way, the model still needs to optimize itself for the new dataset seen in its input to the desired task in the output.

The idea is that the model needs to understand when it is wrong. To quantify how wrong the output prediction is, it is first necessary to define a loss function. The loss function depends of the problem at hand, but it commonly involves minimizing the discrepancy between the predicted output and the actual output (JOSEPH, 2020).

For object detection, there are typically 2 problems, object positioning and its classification. Giving thus 2 losses which are commonly combined to form an optimum joint loss for training the NN. For the classification, a typical loss calculation is the binary cross-entropy loss. The idea behind this function is that it compares the predicted distribution of whether an object is from a class to the actual distribution, and attempts to minimize the differences between these distributions (JOSEPH, 2020). Its expression is presented in Eq. 3.2.

$$J(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} y_i \log\left[f\left(x_i;\boldsymbol{\theta}\right)\right] + \left(1-y_i\right)\log\left[1-f\left(x_i;\boldsymbol{\theta}\right)\right] \tag{3.2}$$

Where n is the number of classes, $\boldsymbol{\theta}$ is the weights to be adjusted, y_i is the actual value, x_i is the input and $f\left(x_i;\boldsymbol{\theta}\right)$ is the predicted value.

For the object positioning, cross-entropy loss is no longer suitable, because now it becomes a regression problem. Rather, the mean squared error (MSE) loss is usually employed. Its idea is that it will try to minimize the squared difference between the actual value and a predicted value (JOSEPH, 2020). Its expression is presented in Eq. 3.3.

$$J(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i_i}^{n} \left[y_i - f\left(x_i;\boldsymbol{\theta}\right)\right]^2 \tag{3.3}$$

After defining the adequate loss function for the concerned task, the NN can start its process of optimizing its weights. The NN optimization should find the set of weights $\boldsymbol{\theta}$ which minimizes the calculated loss. If there was only one or a few weights, every combination could be tested to find a global minimum. Due to this method infeasibility in view of many weights, the gradient descent technique is employed, which is a first-order iterative optimization algorithm for finding a local minimum of a differentiable function. Its equations are shown in Eq. 3.4 and 3.5.

$$grad = \frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \tag{3.4}$$

$$\boldsymbol{\theta}_{new} = \boldsymbol{\theta}_{current} - \eta\frac{\partial J(\boldsymbol{\theta}_{current})}{\partial \boldsymbol{\theta}_{current}} = \boldsymbol{\theta}_{current} - \eta \times grad_{current} \tag{3.5}$$

Where η is the learning rate. In other words, it is the update step the algorithm takes at each interaction. This hyperparameter is very important because it can determine if the model will converge and how fast it will run, in risk of falling into a local minima. To mitigate this

problem, many modern models have been employing adaptive learning rate, starting with bigger steps and getting smaller as interactions pass (JOSEPH, 2020).

In a nutshell, the derivate of the loss function with respect to all the weights is calculated to determine the direction of maximum ascent. Then the model can use this value to change its weights to make the loss function drive downwards until it reaches a point of convergence at a local minimum.

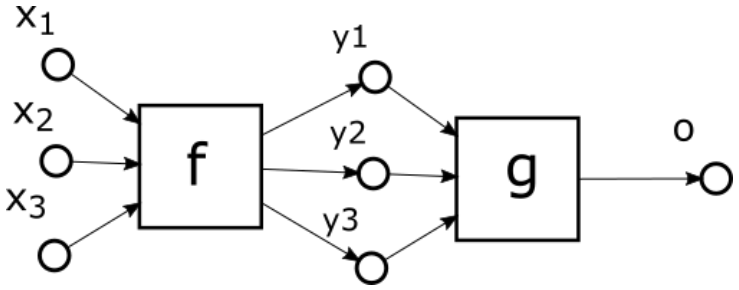
To develop Eq. 3.4, it is necessary the chain rule from calculus. A weight in the first layer may figure out how much it affects the loss in the output by knowing how much it affects the following layer's weights and how much the following layer's weights affect the output. And this process follows for all the weights in all the layers from output to input. That is why this process in the NN is known as back propagation.

Mathematically, if there is $\mathbf{o} = g(f(\mathbf{x}))$ with $\mathbf{o} = g(\mathbf{y})$ and $\mathbf{y} = f(\mathbf{x})$, the element to element derivative can be written as in Eq. 3.6.

$$\frac{\partial o}{\partial \mathbf{x}_i} = \sum_{j=1}^{M} \frac{\partial o}{\partial \mathbf{y}_j}\bigg|_{\mathbf{y}=f(\mathbf{x})} \frac{\partial \mathbf{y}_j}{\partial \mathbf{x}_i} \tag{3.6}$$

For instance, considering the scheme in Fig. 3.3, the derivative $\frac{\partial o}{\partial \mathbf{x}_2}$ can be calculated with Eq. 3.7.

Fig. 3.3 – Schematic for understanding back propagation between layers



Source: Author.

$$\frac{\partial o}{\partial \mathbf{x}_2} = \frac{\partial o}{\partial \mathbf{y}_1}\bigg|_{\mathbf{y}=f(\mathbf{x})} \frac{\partial \mathbf{y}_1}{\partial \mathbf{x}_2} + \frac{\partial o}{\partial \mathbf{y}_2}\bigg|_{\mathbf{y}=f(\mathbf{x})} \frac{\partial \mathbf{y}_2}{\partial \mathbf{x}_2} + \frac{\partial o}{\partial \mathbf{y}_3}\bigg|_{\mathbf{y}=f(\mathbf{x})} \frac{\partial \mathbf{y}_3}{\partial \mathbf{x}_2} \tag{3.7}$$

Once this decent is completed, the last obtained weights configures one of many 'optimal' possible sets for the model and dataset in the input. Actually, the random initialization of weights and the random sequence which data from the dataset is input during the training may direct the optimization towards different local minimums. This concept of variability of the

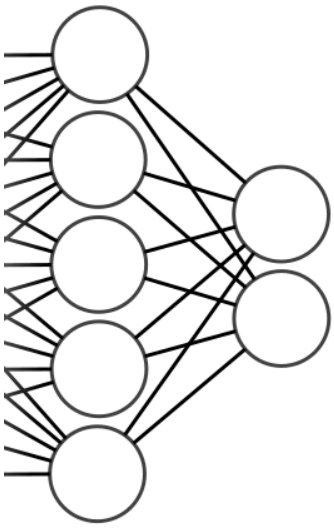
final optimized model is called deep ensemble, which is currently target of exploitation (Abe et al., 2022).

3.4 Types of layers

As detailed beforehand, a NN is a composition of layers that are equivalent to sub-functions. Some common layer topologies, i.e., functions, have been developed due to their capacities of feature extraction and data manipulation within a NN. Following some of them and their hyperparameters:

– fully connected layer (FC layer) connects every neuron in one layer to every neuron in the next layer. It is found in all different types of NNs ranging from standard neural networks to convolutional neural networks (CNN). Its main issue is the computational complexity, as their input grows, the combinatorial operation vector explodes (ISAKSSON, 2020). Fig. 3.4 illustrates this topology.

Fig. 3.4 – Fully connected layer



Source: Author.

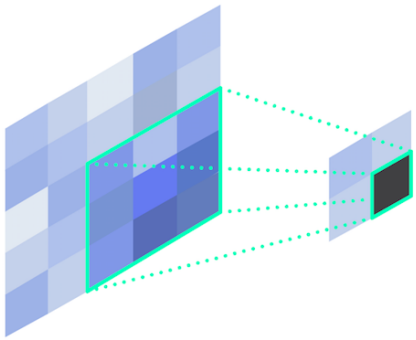
There are 3 main hyperparameters when applying this topology: activation function, number of neurons and dropout.

– convolution layer is largely employed for detecting features in images. It passes systematically multiplying different filters through patches of the entire input. These filters, also called kernels, are sets of weights with the same number of dimensions that of the input,

e.g., in a RGB image, the filter also has 3 dimensions. The result of this process in the output is the response of the patches to these filters, similar to a correlation operation. The filter describes the probabilities that a given pattern of pixels represents a feature (ISAKSSON, 2020).

Each filter travels from left to right and top to bottom over the entire image to detect features. The number of pixels by which the filter moves for the next iteration is called the stride. Padding may be added around the input image to ensure that the filter always fits within the total bounds of the image for a given stride (ISAKSSON, 2020). Fig. 3.5 illustrates this topology.

Fig. 3.5 – Convolution layer

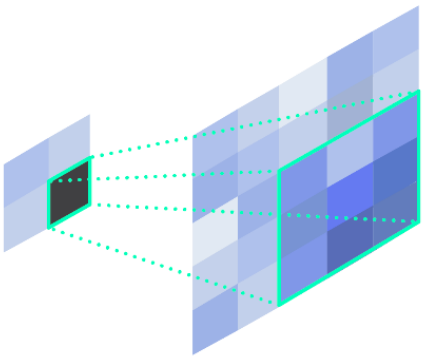


Source: Adapted from (ISAKSSON, 2020).

There are some main hyperparameters when applying this topology: dimensionality, patch size, stride, number of filters, padding strategy, activation function, dropout and pooling.

– **deconvolution layer** is a transposed convolution operation that upsamples its input to higher resolutions. The input typically is image pixels, feature maps generated from previous convolution layer, etc. (ISAKSSON, 2020). Fig. 3.6 illustrates this topology.

Fig. 3.6 – Deconvolution layer

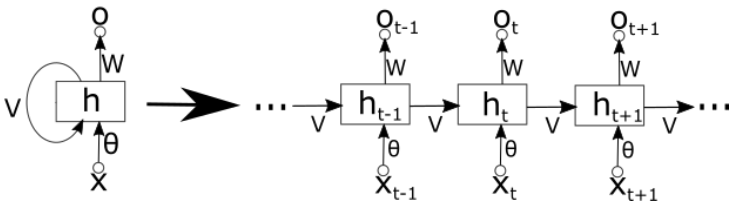


Source: Adapted from (ISAKSSON, 2020).

There are some main hyperparameters when applying this topology: dimensionality, stride, number of filters, padding strategy, activation function and dropout.

– **recurrent layer** possesses a logical connection from its previous state besides its normal input, conferring it memory capability. Recurrent layers form the basis of recurrent neural networks (RNNs) by maintaining a state across iterations, thus allowing sequential data processing, like natural language and time series (ISAKSSON, 2020). Fig. 3.7 illustrates this topology.

Fig. 3.7 – Recurrent layer



Source: Author.

Where V is the activation function, $\mathbf{h}$ is the weights for the connection of the hidden layer to the hidden layer, $\mathbf{X}$ is the input layer, θ is weights for the connection of the input layer to the hidden layer, $\mathbf{W}$ is the weights for the connection of the hidden layer to output layer and $\mathbf{O}$ is the output layer. t indicates the time step. Notice that a prediction at t needs information from later time steps.

There are some main hyperparameters when applying this topology: dimensionality, type of recurrent neural network (LSTM, GRU...), return sequence and dropout.

3.5 Deep learning models

Models are combinations of the aforementioned layers in subsection 3.4. Due to different properties that a data may present, e.g., a time series, where the order in which the data is input to the NN is meaningful, data processing may vary to become more appropriated. That is why different NN models have been proposed, each theoretically implementing different approaches to handle the input. Following some consecrated models in the literature:

– Convolutional Neural Network (CNN) is capable of learning from low semantic (also called low-level) and high resolution to high semantic (also called high level) and low resolution feature maps from image pixels, as we move further up the stacks of convolutional and pooling layers.

The first part of the NN is formally called *backbone* and is responsible for extracting very discriminatory characterization in form of feature maps which then feeds subsequent *task heads* to perform a final processing over these features to output a prediction over various computer vision tasks. These parts can be seen at Fig. 4.17. One of the first recognized CNNs is AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012), which consists of five convolutional layers, three max-pooling layers, and three fully-connected layers. By only employing 3x3-sized convolutional kernels, VGGNet (Simonyan; Zisserman, 2014) got a larger number of channels and more diverse features were obtained, which made it obtain better performance than AlexNet.

ResNet (HE et al., 2015), U-Net (RONNEBERGER; FISCHER; BROX, 2015), and DenseNet (HUANG; LIU; WEINBERGER, 2016) were later developed with even higher performance. The key characteristic of all these architectures is not only connecting neighboring layers, but inserting skip connections that allow sharing features from different levels in the NN. These new connections reduce the loss of information going deeper in the NN and mitigates the vanishing gradients issue of deep networks (ZHU et al., 2020).

– U-Net has skip connections concatenating features from the first layer to last, second to second last, and so on. In this way, it can transfer high-resolution features from

initial layers to the end layers with high semantic features (RONNEBERGER; FISCHER; BROX, 2015).

  - **ResNet** uses skip connections across individual blocks and not across each layer (HE et al., 2015).

  - **DenseNet** has all its layers attached to all preceding layers, reducing the size of the NN, but increasing memory usage (HUANG; LIU; WEINBERGER, 2016).

– **Recurrent Neural Network (RNN)** is founded on the recurrent units, which take the input and also their previous state into account, creating a memory capacity. Therefore, its applications are mainly text and sequence-dependent data, e.g., time series, which require retaining past information to interact with present information. One very popular architecture is the long short term memory (LSTM), which tackles the gradient diminishing problem typically present (ZHU et al., 2020).

– **Generative Adversarial Network (GAN)** consists of 2 networks called generator and discriminator. The generator must learn a latent space, through which generated samples are based to have the same distribution as the training data and the discriminator tries to distinguish the origin of the sample. This philosophy is the basis for the creation of artificial photo-realistic images, super-resolution, etc (ZHU et al., 2020).

– **Transformer** is similar to a LSTM, the transformer architecture converts one sequence into another with the help of two parts, the encoder and decoder. Nonetheless, it has a great technical improvement by applying an attention-mechanism.

3.6 Datasets

A dataset is a structured collection of data which should be easily accessible and manipulable. Data are observations or measurements represented as text, numbers, or multimedia (USGS, 2018).

Knowing that supervised models learn only what it is subjected to by the input data, the datasets must contain high quality descriptive data in large quantity. It is important to stimulate the model to observe all possible variations to which the input data can be conditioned under the same label. A richer training can allow a model to obtain deeper intricate patterns from the data and achieve better inference accuracy. When the available dataset does not provide it

sufficiently, data augmentation, in section 3.7, and transfer learning (TL), in section 3.8, are methods to be considered to improve training.

3.6.1 Training, validation and test sets

Typically, a dataset is divided into training, validation and sometimes also test sets. The training set feeds the model throughout the training phase to fit the model, while the validation and test sets are retained.

After fitting the model to the training set, it is necessary to evaluate the performance of the model to the task. Using the same training set to evaluate would result in a biased score. Therefore the model is evaluated on the held-out set to give an unbiased estimation of its performance. Thus, the validation set is employed to evaluate the model during training, but is not used for training, allowing to check for overfitting. The test set is another side set to provide an unbiased evaluation of the final fitted model (BROWNLEE, 2017).

The simplest way to split the entire data set is to define percentages for each set, e.g. 70, 20 and 10% for training, validation and test sets respectively. Once the data set has been split, the data between them no longer mixes. A more sophisticated splitting is the k-fold cross-validation instead of a separate validation dataset.

k-fold cross-validation splits the available dataset into k sets. The model is trained with $k-1$ sets and evaluated with the last remaining set. Then the model is reset from scratch and trained with other $k-1$ sets and evaluated with the other remaining set. It is repeated k times until each set has been selected as the evaluation set (BROWNLEE, 2017). At the end of this process, the model is usually summarized into the mean value and variance between the k outcomes.

3.7 Data augmentation

Large datasets intrinsically cover basically all the variations that the input data may suffer and data augmentation is not an indispensable needing. However, having a large dataset to train a model for particular tasks is often unusual, given the large amount of time it takes to label the data and possible difficulties in obtaining more data, depending of the application.

For small and medium datasets, the model is not capable to perform as well as it does when trained with larger datasets, since it lacks the aforementioned variability in the input data

to allow the extraction of deeper intricate patterns. In these cases, a way to improve it is trough transformations performed on the input data, i.e., data augmentation.

The possible transformations to be applied depend on the nature of the input data and model's tasks. Most computer vision tasks employ transformations in the images such as flipping, rotation, resizing, changing brightness and contrast, etc.

Remote sensing is a example of domain that contain many tasks that lack labeled data. Data augmentation therefore plays a major role in it. However, due to the remote sensing imagery eccentricity, the simple color and geometric transformations mentioned above do not correspond to all the natural variations perceived.

(ILLARIONOVA et al., 2021) proposes a data augmentation for the semantic segmentation task that works with high-resolution georeferenced satellite images. Summarizing, it crops objects from original images using their masks and pasts them to a new background. Thus, every object and background can be augmented independently to increase the variability of training images.

3.8 Transfer learning (TL)

Transfer learning (TL) is the process of transferring the "knowledge", i.e., the weights from many connections among neurons, from a model to another. It has been gaining great attention in current ML researches due to the results it envisions.

Consider a model which has been trained in a large dataset and perform very well in some task, e.g., detecting motorcycles. One could try to implement a model partially like the later to detect bicycles and use those weights as a starting point for training the new model.

This process can drastically reduce training time and allow training of larger models under small data sets, reducing overfitting as well as data augmentation.

For a reinforcement learning perspective, reusing information from previously learned tasks for the learning of new tasks has the potential to significantly improve the sample efficiency of a learning agent (WIKIPEDIA, 2021b).

3.9 Object detection metrics

To evaluate the performance of an object detector and thus be able to compare different detectors, it is necessary to establish metrics. In essence, the detection made can be classified

as:

– True positive (TP) is a correct detection;

– False positive (FP) is an incorrect detection;

– True negative (TN) is when the background is correctly not detected as an object;

– False negative (FN) is when a ground-truth is missed (not detected).

3.9.1 Intersection over union (IoU)

To determine whether the detection bounding box is correct, the degree of overlap between ground truth and detection areas is checked. This metric is called intersection over union (IoU) and can be defined as Eq. 3.8.

$$IoU = \frac{area_{\text{overlap}}}{area_{\text{union}}} = \frac{area_{\text{groundTruth}} \cap area_{\text{detection}}}{area_{\text{groundTruth}} \cup area_{\text{detection}}} \tag{3.8}$$

By defining a threshold value for the IoU, the detection bounding box can be classified between TP, FP, TN or FN.

3.9.2 Precision and recall

Two relevant metrics, precision and recall, are defined by evaluating the ratio between the number of TPs, FPs and FNs. Precision is the degree of exactness of the model in identifying only relevant objects, seen in Eq. 3.9. Recall measures the ability of the model to detect all ground truths, seen in Eq. 3.10 (KOECH, 2020).

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{allDetections} \tag{3.9}$$

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{allGroundTruths} \tag{3.10}$$

A perfect model would have precision and recall equal to 1.

The precision-recall (PR) curve is a plot of precision and recall at varying values of confidence score threshold in detection. Increasing the confidence score threshold increases the number of FN and reduces the number of FP, which increases the precision and reduces the recall. Decreasing the confidence score threshold produces the opposite, reduces the precision and increases the recall.

3.9.3 Average Precision (AP)

Average Precision (AP) is typically presented as $AP@\alpha$. It means the area under the precision-recall curve (AUC-PR) evaluated with α, the IoU threshold, and it can be defined as in Eq. 3.11 (KOECH, 2020).

$$\text{AP}@\alpha = \int_0^1 p(r)\, dr \tag{3.11}$$

Where α is the IoU threshold, r is the recall value and p is the precision-recall curve (AUC-PR). A high AP means high precision and high recall.

A very useful variation is the 11-point interpolation $AP@\alpha_{11}$, a plot of interpolated precision scores at 11 equally spaced standard recall levels, 0.0, 0.1, 0.2, . . . 1.0. It is defined as in Eq. 3.12 (KOECH, 2020).

$$\text{AP}@\alpha_{11} = \frac{1}{11} \sum_{r \in R} p_{\text{interp}}\left(r\right) \tag{3.12}$$

Where $R = 0.0, 0.1, 0.2, ...1.0$ and

$$p_{\text{interp}}\left(r\right) = \max_{r':r' \geq r} p\left(r'\right) \tag{3.13}$$

The justification is that the PR curve is not monotonically decreasing, and for a determined recall, the model would not operate with lower precision given the presence of higher precision for even higher recalls, therefore the metric takes the highest precision value for the specific recall or higher.

3.9.4 Mean Average Precision (mAP)

The aforementioned AP, and its variations, are calculated individually for each class. To synthesize the model's performance, usually the AP values of all classes are averaged to obtain the mean Average Precision (mAP). It can be defined as in Eq. 3.14.

$$\text{mAP}@\alpha = \frac{1}{n} \sum_{i=1}^{n} \text{AP}_\text{i}\text{, for n classes.} \tag{3.14}$$

Another usual practice done when reporting models is to calculate the AP for a series of IoU thresholds, α, and averaging them. E.g., AP@[.50:.5:.95] means the AP value averaged over 10 different IoUs, ranging from 50% to 95% at 5% step-size (KOECH, 2020).

3.9.5 F1 score

F1 score combines precision and recall into one metric by calculating the harmonic mean between those two (CZAKON, 2022). It is a special case of the more general function F beta, in Eq. 3.15.

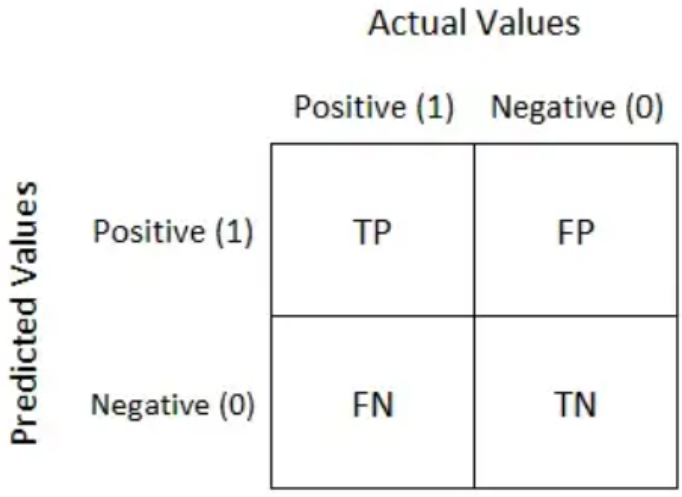
$$F_{\text{beta}} = \left(1 + \beta^2\right) \frac{\text{precision} * \text{recall}}{\beta^2 * \text{precision} + \text{recall}} \tag{3.15}$$

The beta in the F-beta score defines the prevalence of recall over precision. The F1-score weights recall and precision equally.

3.9.6 Confusion matrix

Confusion Matrix is a performance measurement for machine learning classification (NARKHEDE, 2018). It is a matrix that contains the ground truth classes by the inferred classes. It allows to check if the model is classifying correctly and which classes the model has more difficulty to process and identify. Fig. 3.8 shows a simple confusion matrix with 2 classes, giving 4 combinations.

Fig. 3.8 – Example of a confusion matrix with 2 classes



Source: Adapted from (NARKHEDE, 2018).

3.10 Benchmarks for models

Due to the vast number of new models currently emerging, for all kinds of research fields, e.g. computer vision, natural language processing, medical processing; and their many tasks that can be tackled, it is necessary to track their progress in an organized way. Benchmarks are available on the Internet, e.g., (PAPERSWITHCODE, 2022), where the leading-boards are separated into their domain of performance, the task to be addressed, and the dataset on which they were tested.

Important to emphasize the distinction between the performance of the same model evaluated under different datasets. As presented in the section 3.6, the importance of datasets for supervised learning, a model never has an absolute score, its performance is reported along with the training and test datasets, since datasets can have different degrees of quality and present different tasks with distinct difficulties.

4 LITERATURE REVIEW

Before to present the DL-based object detection specifically for the remote sensing imagery application, section 4.1 reviews general image deep learning-based object detection. The reason is to first understand DL-based object detectors, the main types, how they work and how they evolve to achieve better performance.

Section 4.2 then presents the SOTA DL-based object detectors in remote sensing imagery.

4.1 Deep learning-based object detectors

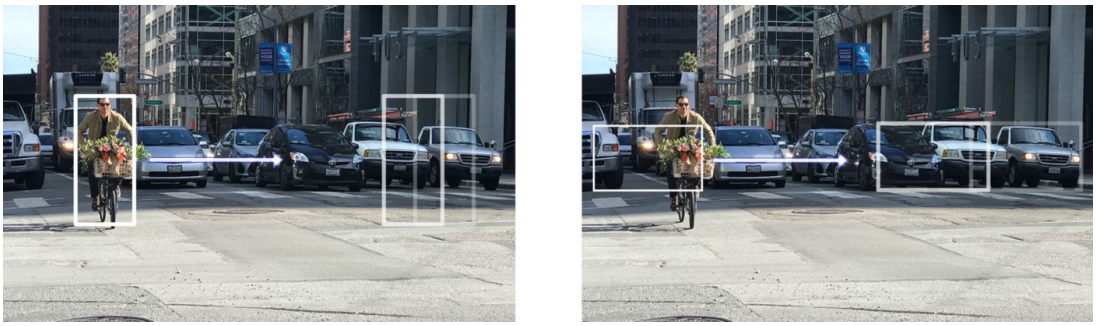
Two main concepts of object detectors are present in the literature: region-based (two-stage) and single shot (one-stage) object detector architectures.

4.1.1 Region-based object detector

Beginning the journey on object detection, there is the region-based object detector. This kind of detector has a strong project condition, where 2 parts are necessary: one is going to identify regions of interest (ROIs), and the other the class and the boundary box in these regions.

A brute force approach is sliding different size windows throughout the image, from left to right and up to down, as demonstrated in Fig. 4.1.

Fig. 4.1 – Windows sliding throughout the image



Source: Adapted from (HUI, 2018a).

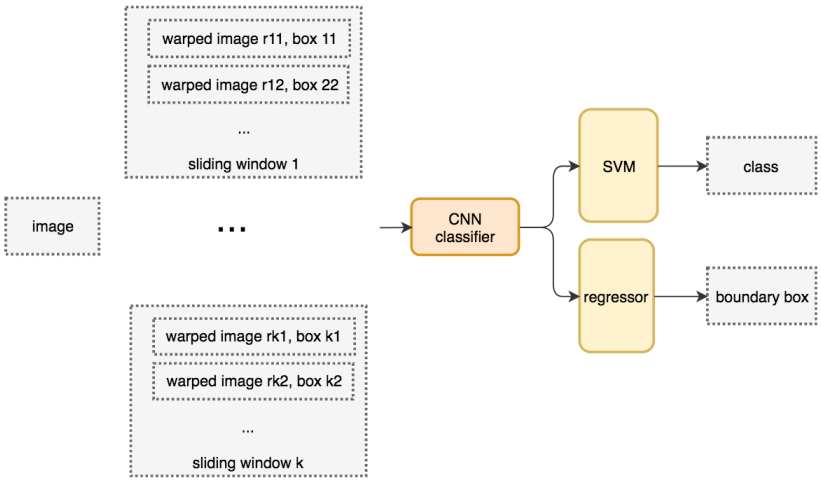
At the end of the process, for each window there will be a collection of patches that are later warped to a fixed size, as shown in Fig. 4.2.

Fig. 4.2 – Patch being warped to a fixed rectangular size

Source: Adapted from (HUI, 2018a).

These final patches are fed into a CNN classifier for extracting features. Finally, these features are inputted to a SVM classifier to identify the class and to a linear regressor for estimating the boundary box, as shown in the block diagram in Fig. 4.3.

Fig. 4.3 – Block diagram for the sliding-window detector



Source: Adapted from (HUI, 2018a).

4.1.1.1 R-CNN

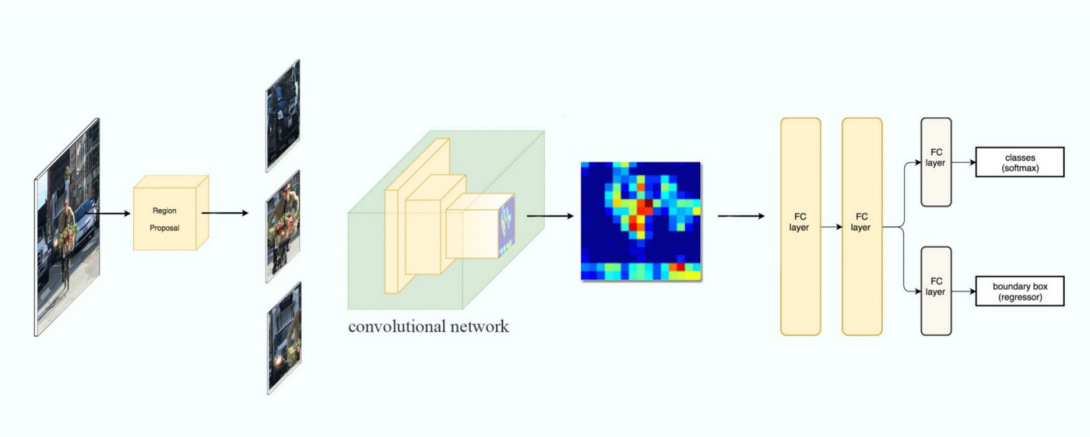
To reduce the unpractical number of patches of interest, a region proposal method is established in lieu of the sliding window. And this is how R-CNN (GIRSHICK et al., 2013) works. R-CNN is one of the first successful NN-based object detector. It first employs a region proposal method to propose about 2000 ROIs, like shown in Fig. 4.4, that are subsequently warped into a fixed size and individually fed into a CNN.

Fig. 4.4 – The progression of the selective search algorithm. Several regions are discriminated and then merged by their similitude

Source: Adapted from (UIJLINGS et al., 2013).

Finally, fully connected (FC) layers receive the CNN output features to classify the object and refine the boundary box. The R-CNN workflow is shown in Fig. 4.5.

Fig. 4.5 – R-CNN workflow from an input image to its prediction



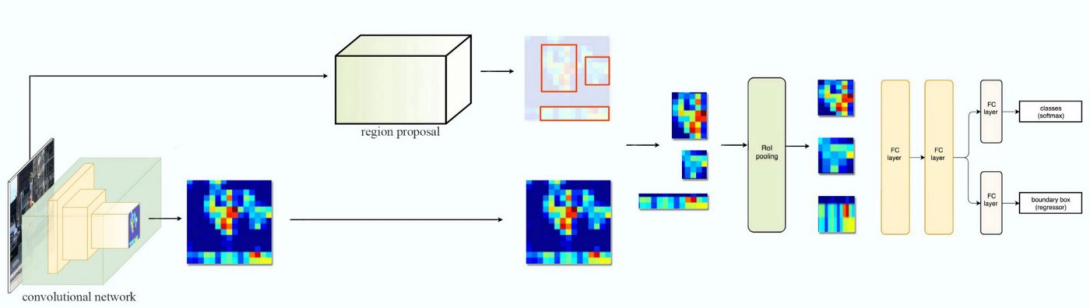
Source: Adapted from (HUI, 2018a).

4.1.1.2 Fast R-CNN

Since each one of the 2000 proposed patches are fed and processed by the CNN separately, it ends performing 2000 features extractions. Thus, its training and inference times are very slow. To avoid this problem, fast R-CNN (GIRSHICK, 2015) executes the feature extraction in the whole image first. The external region proposal method continues creating ROIs, but

now it uses them to patch directly in the feature map, on these locations. Then, these patches are warped to a fixed size using ROI pooling and feed FC layers for classification and localization. The new workflow for the fast R-CNN is presented in the Fig. 4.6.

Fig. 4.6 – Fast R-CNN workflow from an input image to its prediction



Source: Adapted from (HUI, 2018a).

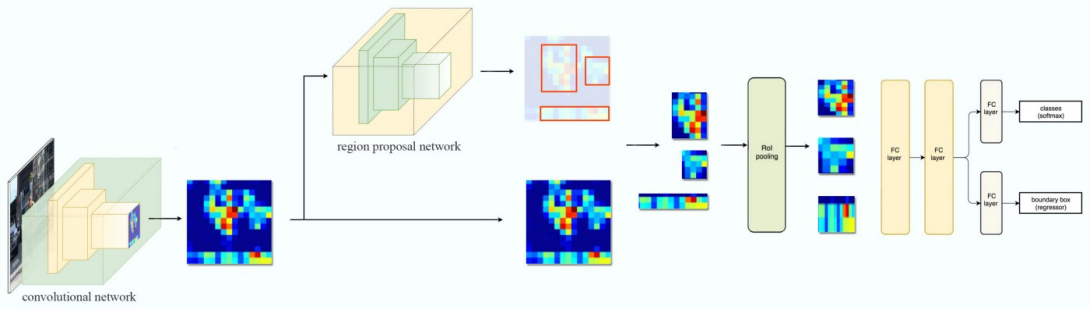
Because of this change in the feature extraction, fast R-CNN is 10 and 150 times faster, in training and inference times, respectively. Also, a great improvement in this model to be pointed out, is the straight path through NN-based components (feature extractor, classifier, and the boundary box regressor). Allowing an end-to-end training with multi-task losses, combining classification and localization losses and improving its accuracy (HUI, 2018a).

4.1.1.3 Faster R-CNN

Even though fast R-CNN had reduced training and inference times, the external region proposal method does not allow processing parallelization, given the dependency across pixels in the process of grouping regions. This drastically penalizes the whole model and losses the important parallelization quality from the other DL-based components.

Faster R-CNN (REN et al., 2015) gets around exactly this design failure. Replacing the external region proposal method by an internal DL-based network, called region proposal network (RPN), which generates ROIs very faster than before and speeds up the whole model several times. The new workflow for the faster R-CNN is presented in the Fig. 4.7.

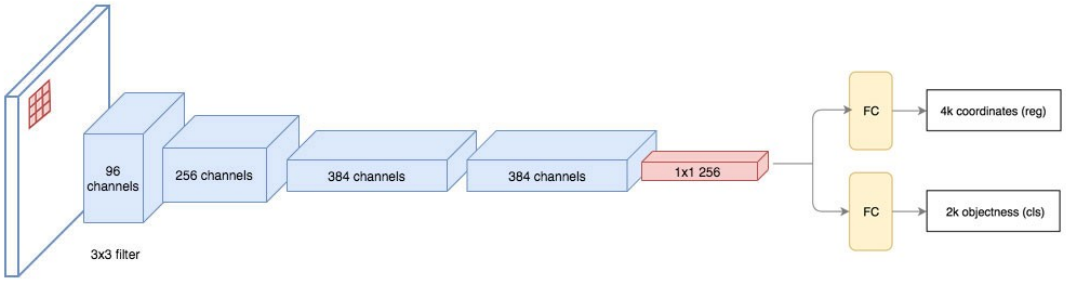
Fig. 4.7 – Faster R-CNN workflow from an input image to its prediction



Source: Adapted from (HUI, 2018a).

The RPN takes the feature map from the CNN and makes class-agnostic region proposals using a CNN. The RPN's output is then fed into 2 separate fully connected layers to predict 2 objectness scores (have or not an object) and a boundary box. The RPN architecture is shown in Fig. 4.8.

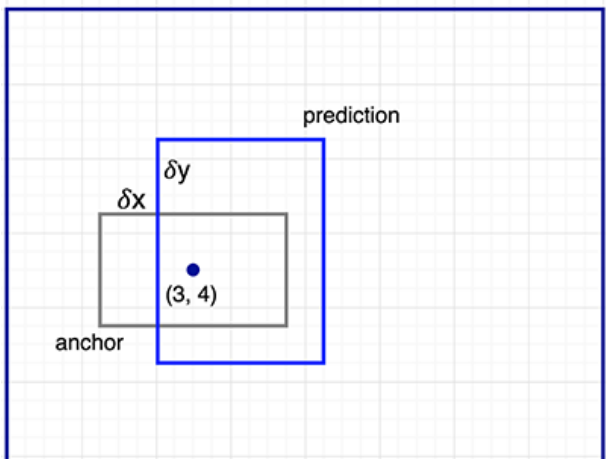
Fig. 4.8 – RPN architecture



Source: Adapted from (HUI, 2018a).

For each location in the feature map, there are k guesses, stemming from k reference boxes, also called priors or anchors. The idea is to cover possible object scales and aspect ratios in the image, from which it predicts constrained offsets δ_X and δ_y that are relative to the top left corner. In the Fig. 4.9, one anchor and its final prediction with offsets are shown.

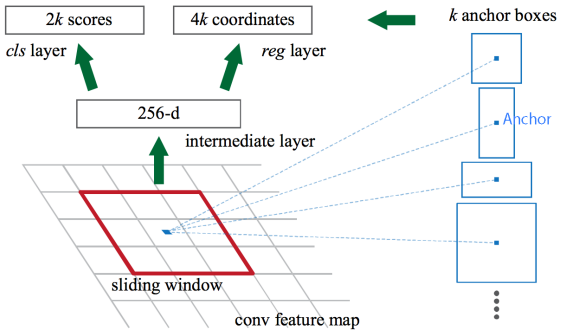
Fig. 4.9 – Example of an anchor and its offsets δ_X and δ_y predictions



Source: Adapted from (HUI, 2018a).

Fig. 4.10 clarifies all the predictions made for one sliding window.

Fig. 4.10 – Predictions made for one sliding window



Source: Adapted from (REN et al., 2015).

For each sliding window, an intermediate layer with 256 channels is generated in the feature map, which is then used by the *cls* layer, to estimate 2 objectness scores, and *reg* layer, to estimate the 4 boundary box's coordinates. Because for each window there are k anchor boxes, it is generated $2k$ scores in the *cls* layer and $4k$ coordinates in the *reg* layer. In the faster R-CNN implementation, there are 9 anchor boxes, combination of 3 different scales and 3 different aspect ratios.

4.1.2 Single shot object detector

Another powerful kind of object detector is the single shot. As presented in subsection 4.1.1, faster R-CNN has a dedicated region proposal network followed by its classifier and

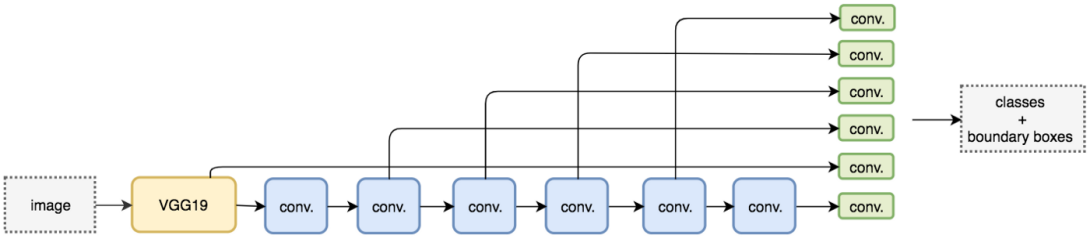
boundary box regressor. Which makes it accurate but unfortunately also slow. An alternative to reduce training and inference times could be forgetting the region proposal procedure and estimating classes and boundary boxes directly from the feature map.

Single shot object detector slides a window over the whole feature map and predicts the class and a refined boundary box overlapping the window region, i.e., there are not many windows with different shapes sliding, but one window that works as an anchor to a predicted boundary box. Because of this step reduction, single shot detectors trades accuracy with real-time processing speed, tending to have more issues detecting too small or too big objects present in the image (HUI, 2018b).

4.1.2.1 Single Shot MultiBox Detector (SSD)

Single Shot MultiBox Detector (SSD) (LIU et al., 2016) object detector employs a network called VGG16, from the family of VGGNet (Simonyan; Zisserman, 2014), with 16 layers, as feature extractor which feeds some convolution layers that finally make class and boundary prediction. Because convolution layers reduce resolution, and then spatial dimension of their input image, some implementations make multiple class and boundary boxes inferences by using feature maps from different convolution layers depth. Fig. 4.11 shows this multi-scale feature map inference.

Fig. 4.11 – Multi-scale feature map inference



Source: Adapted from (HUI, 2018b).

4.1.2.2 You only look once (YOLO)

Another single shot detector is You only look once (YOLO) (REDMON et al., 2015). It uses the DarkNet (REDMON, 2013–2016) instead of VGG16 as feature extractor and concatenates multi-scale feature maps rather than performing independent detections, allowing feature-richer inferences. DarkNet is small and efficient at the same time. It uses convolutional layers instead of fully connected layers.

4.1.2.3 YOLO v2

YOLOv2 (REDMON; FARHADI, 2016) sets off from its predecessor and introduces a bunch of new implementation improvements to go along correcting specific problems as it develops. Each feature introduced and its final contribution to the model's mean Average Precision (mAP) can be seen in the Fig. 4.12.

Fig. 4.12 – Features introduced to the YOLO v2 model and their contributions to the mAP

	YOLO							YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓	✓
anchor boxes?				✓	✓				
new network?					✓	✓	✓	✓	✓
dimension priors?						✓	✓	✓	✓
location prediction?						✓	✓	✓	✓
passthrough?							✓	✓	✓
multi-scale?								✓	✓
hi-res detector?									✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

Source: Adapted from (REDMON; FARHADI, 2016).

Batch normalization leads to significant improvements in convergence while eliminating the need for other forms of regularization (IOFFE; SZEGEDY, 2015). The original YOLO trains the classifier network at 224 × 224 and increases the resolution to 448 for detection. This means the network has to simultaneously switch to learning object detection and adjust to the new input resolution. For YOLO v2, the classification network is first fine tuned at the full 448 × 448 resolution for 10 epochs on ImageNet. This gives the network time to adjust its filters to work better on higher resolution input. Then the resulting network is fine tuned on detection. This configures the high resolution classifier.

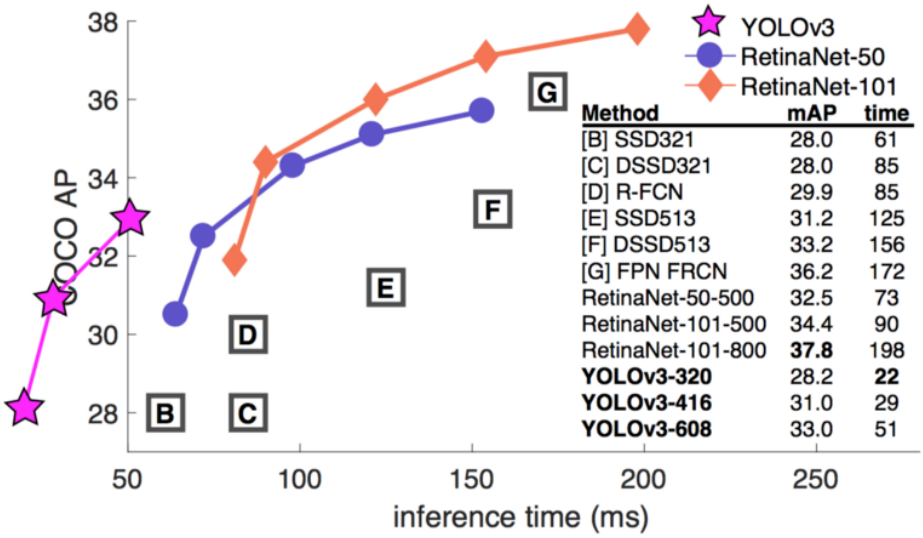
The network can learn to adjust the boxes appropriately from hand picked anchor boxes, but picking better priors for the network to start with, it can become easier for the network to learn to predict good detections. So, instead of choosing priors by hand, it runs k-means clustering on the training set bounding boxes to automatically find good priors. This modified YOLO predicts detections on a 13 × 13 feature map. While this is sufficient for large objects, it may benefit from finer grained features for localizing smaller objects. Faster R-CNN and SSD both run their proposal networks at various feature maps in the network to get a range of resolutions. YOLO v2 takes a different approach, simply adding a passthrough layer that brings features from an earlier layer at 26 × 26 resolution. The passthrough layer concatenates

the higher resolution features with the low resolution features by stacking adjacent features into different channels instead of spatial locations, similar to the identity mappings in ResNet (REDMON; FARHADI, 2016).

4.1.2.4 YOLO v3

YOLO v3 (REDMON; FARHADI, 2018) changes the former Darknet backbone network for feature extraction to Darknet-53. This new backbone uses skip connections like the residual network ResNet (HE et al., 2015), acquiring the same classification accuracy with less billion floating-point operations (BFLOP) than ResNet-152, i.e., processing much faster the input. Fig. 4.13 shows a comparison of the average precision on the MS COCO dataset (LIN et al., 2014) (COCO AP) by the inference time for YOLO v3 and RetinaNet-50 and 101 (LIN et al., 2017). RetinaNet is the first dense detector to employ Focal loss, proposed to address class imbalance by reshaping the standard cross entropy loss such that it down-weights the loss assigned to well-classified examples (LIN et al., 2017).

Fig. 4.13 – Comparison of the COCO AP by the inference time between YOLO v3 and RetinaNet-50 and 101

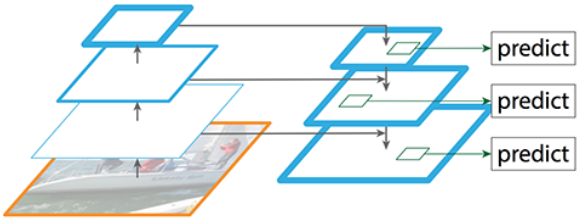


Source: Adapted from (REDMON; FARHADI, 2018).

To help correcting the small objects detection low accuracy, YOLO v3 employs the feature pyramid networks (FPN) (LIN et al., 2017). FPN is a feature extractor designed with feature pyramid concept. This network is basically an attachment to the usual feature extractor

network and provides better quality feature map. As shown in the Fig. 4.14, FPN is composed of 2 pathways, a bottom-up, which is the usual feature extractor CNN, and a top-down.

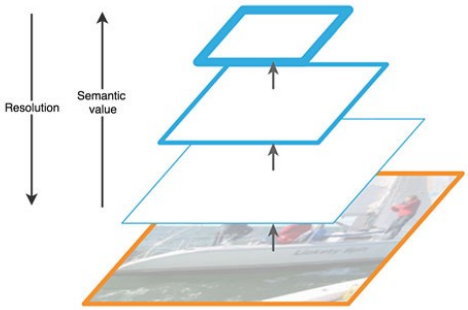
Fig. 4.14 – FPN composition



Source: Adapted from (LIN et al., 2017).

To understand the importance of the top-down pathway, it is important to observe the features' progression inside the feature extractor, as shown in Fig. 4.15. As it goes deeper, the features' spatial resolution decreases as well as more high-level structures become present. In other words, the resolution is reduced but the semantic value is increased.

Fig. 4.15 – Features' resolution and semantic value progression inside the feature extractor



Source: Adapted from (HUI, 2018b).

The top-down pathway allows reconstructing higher resolution layers from semantic rich layers. But these up-sampled reconstructed feature maps still lack precision, and then lateral connections are established between reconstructed layers and the feature maps at same depth in the bottom-up pathway.

4.1.3 Anchor-free methods

All the aforementioned detectors, one-stage and two-stages, except YOLO v1, are based in biased information, given the prior anchor boxes or anchor boxes from which they depart their detections and estimate objects' bounding boxes.

An alternative to get rid of these biased anchor boxes is through anchor-free approaches. They perform detection in a per-pixel prediction fashion, eliminating hyper-parameters related to anchor boxes and thus presenting less sensitive final detection performance.

YOLO v1 can be considered as one of the first anchor-free method because its bounding box predictions are made only through the pixels close to the center of objects, since they are considered to better characterize the object. However, this approach penalizes the model by achieving low recall (REDMON; FARHADI, 2016).

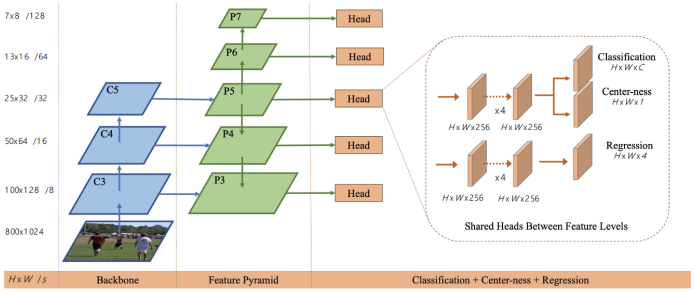
Fully Convolutional One-Stage Object Detection (FCOS) (TIAN et al., 2019), differently, uses all points in the ground truth bounding boxes to predict bounding boxes. The several low-quality detected bounding boxes are later suppressed by a new proposed center-ness branch. The center-ness function decays from 1 to 0 as the location deviates from the center of the object, as shown in Fig. 4.16.

Fig. 4.16 – The center-ness evaluated all over an image. Red, blue, and other colors denote 1, 0 and the values between them, respectively

Source: Adapted from (TIAN et al., 2019).

FCOS's architecture can be seen in Fig. 4.17.

Fig. 4.17 – FCOS's NN architecture



Source: Adapted from (TIAN et al., 2019).

4.1.4 Model development considerations

By looking at these object detectors, their problems and some of their past evolution, it is possible to better understand how the object detection task naturally orients and defines machine learning models capable of performing it.

One fact is that there is a trade-off between inference time and its accuracy. Single shot detectors demonstrate themselves faster than R-CNN ones, but with lower accuracy as a price to pay. Anchor-free models keep the simplicity of one-stage methods and are less biased without the anchor boxes requirement. The most suitable detector depends on the application, balancing between speed and accuracy required.

Besides meta-architectures and feature extractors, there are other smaller considerations to the final model implementation. There are different loss functions and boundary box encondings that can be employed. For instance, in object detection, the localization loss may be less pondered than its classification loss, stimulating the model to better locate objects. And the boundary box may be represented as absolute values or square root of width and height to normalize errors.

4.2 SOTA DL-based object detection in remote sensing imagery

A more complex detection has recently been adopted and developed for being able to better fit the bounding box to arbitrarily oriented objects, with dense distribution and large aspect ratio. Oriented object detection is a generalization of the earlier detection of horizontal bounding boxes (HBB). Fig. 4.18 exemplifies the difference between them, where oriented bounding boxes (OBB) are clearly tighter and more accurate to the true object area.

Fig. 4.18 – Comparison between HBB and OBB representations for objects. (a) OBB representation. (b) HBB representation.

Source: Adapted from (DING et al., 2021).

For this newer representation, the SOTA models can still be discriminated in two-stage and one-stage, anchor-based or anchor-free, detectors.

4.2.1 Two-stage detectors

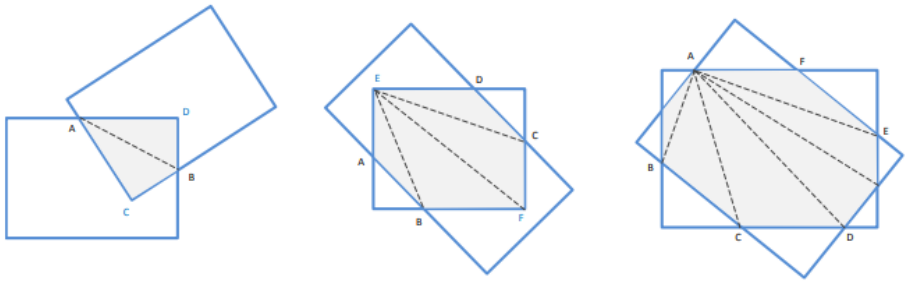
Two-stage object oriented detectors are largely limited by the regression loss calculation (YANG et al., 2022). The typical regression loss in horizontal detection is l_n-norms. l_n-norms are defined as in Eq. 4.1.

$$\|x\|_n = \sqrt[n]{\sum_i |x_i|^n} \tag{4.1}$$

Where x can be a vector or matrix with i elements and $n \in \mathbb{R}$ is the norm order.

However, the detection metric is mostly dependent of the Skew Intersection over Union (SkewIoU) score, exemplified in Fig 4.19, which is sensitive to the deviations of the object positions between large aspect ratio objects (YANG et al., 2022).

Fig. 4.19 – Examples of compute SkewIoU. Intersection points are marked in black, and vertices inside the other rectangle are marked in dark blue



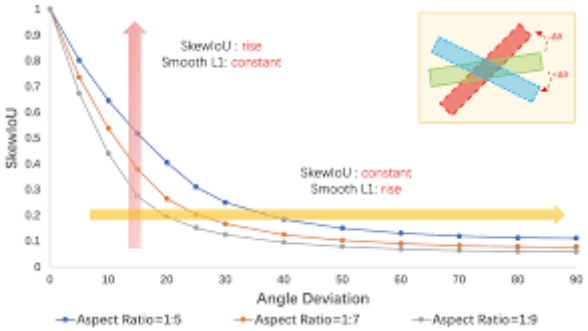
Source: Adapted from (YANG et al., 2022).

The mismatch between the detection metric and regression loss, which was already present in horizontal detection, is even more accentuated in orientated detection. Fig. 4.20 demonstrates the inconsistency between SkewIoU and Smooth l_1-loss, which is a combination of l_1-loss and l_2-loss, seen in Eq. 4.2.

$$l_{1;\,\text{smooth}} = \begin{cases} |x| & \text{if } |x| > \alpha \\ \frac{1}{|\alpha|}x^2 & \text{if } |x| \leq \alpha \end{cases} \tag{4.2}$$

Where α is a hyper-parameter, usually taken as 1. Smooth l_1-loss combines the advantages of l_1-loss, steady gradients for large values of x, and l_2-loss, less oscillations during updates when x is small (SREEKUMAR, 2019).

Fig. 4.20 – Inconsistency between the SkewIoU and regression-based loss Smooth L1



Source: Adapted from (YANG et al., 2022).

For a fixed angle deviation (indicated by the red arrow), SkewIoU decreases as the aspect ratio increases, while the Smooth l_1 loss does not change. The other way, when SkewIoU does not change (indicated by the orange arrow), Smooth L1 loss increases as the angle deviation increases (YANG et al., 2022).

Many solutions have been proposed in horizontal detection by using IoU loss and related variants, e.g., GIoU (REZATOFIGHI et al., 2019). However, these solutions are difficult to implement for oriented detection due to the complexity of computing the intersection between rotated boxes (ZHOU et al., 2019).

Because of that, many studies have been made to find better performing approximations for SkewIoU loss. When comparing the error variance with the final performance, it is evidenced that the consistency between metric and regression loss lies in the approximate and exact SkewIoU loss trend-level consistency rather than value-level consistency. In other words, it is more important to approximate the functions by their trend-level consistency, i.e., increasing and decreasing accordingly, than functions that produce similar values but do not

have trend-level consistency. This property simplifies the difficult of designing new alternatives (YANG et al., 2022).

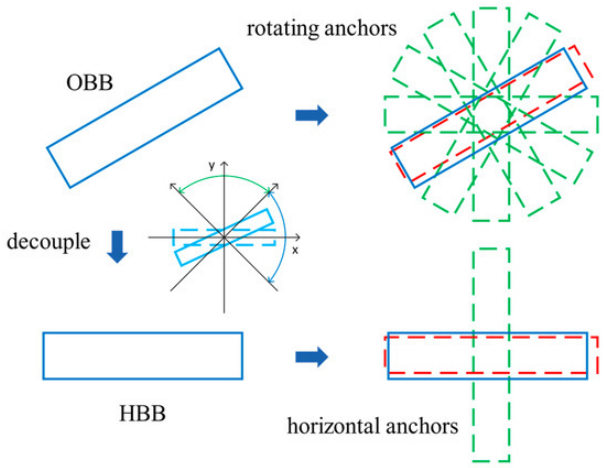
Many works have been made in which the OBB is converted into a Gaussian distribution, avoiding square-like problems and boundary discontinuity (YANG; YAN, 2020; QIAN et al., 2019). (YANG et al., 2021) converts the rotated bounding box into a 2-D Gaussian distribution and approximates the indifferentiable rotational IoU induced loss by the Gaussian Wasserstein distance (GWD) which can be learned efficiently by gradient back-propagation. Similarly, (YANG et al., 2021) changes the design of rotated regression loss based on the relation between rotated and horizontal detection. It converts the rotated bounding box into a 2-D Gaussian distribution and then calculates the Kullback-Leibler Divergence (KLD) between the Gaussian distributions as the regression loss. (YANG et al., 2022) goes even further by modeling the objects as Gaussian distribution and adopting a Kalman filter to inherently mimic the mechanism of SkewIoU, in contrast to GWD and KLD that involves a human-specified distribution distance metric.

4.2.2 One-stage anchor-based detectors

One-stage anchor-based detectors also have attention when treating with oriented detection, given their capacity to obtain high performance at lower complexity than two-stages detectors, i.e., being faster computed, at horizontal detection. The new degree of freedom brings an increase in the number of anchor boxes and the sensitivity of the IoU to changes of angle.

(ZHONG; AO, 2020) overcomes these problems by redesigning the matching strategy between oriented anchors and ground truth boxes. It decouples the OBB into a HBB during matching, thus reducing the instability of the angle to the matching process. Fig. 4.21 shows the difference between matching strategies.

Fig. 4.21 – Difference between rotating anchors matching and OBB rotation-decoupled matching strategy. The red bounding box indicates the matched anchor



Source: Adapted from (ZHONG; AO, 2020).

The strategy is to first transform the OBB into the closest HBB and perform the matching with an anchor. Only then, the matched anchor is rotated to further reduce the matching loss.

From the renowned YOLO family of models, introduced in 4.1.2, the last version adapted to work with OBB is the YOLOv5 (AL., 2021), transformed into YOLOv5-OBB (KAIXUAN, 2022). To make YOLOv5 capable to deal with the new OBB detection, it was necessary to change its data loader to fit the new annotation patterns, modify the loss function to consider the rotation loss given by the new ability to spin the bounding-box to better adjust itself, adapt its proposal filtering technique called non-maximum suppression (NMS) and other minor additions.

YOLOv7 (WANG; BOCHKOVSKIY; LIAO, 2022) is the newest YOLO model. In addition to architecture optimization, it proposes methods for the optimization of the training process. Focusing on some optimized modules and optimization methods which may strengthen the training cost for improving the accuracy of object detection, but without increasing the inference cost. It addresses model re-parameterization and dynamic label assignment topics.

Unfortunately YOLOv7 has not been adapted to work with OBB yet.

5 DEVELOPMENT

The previous chapters have built the necessary knowledge to allow the development of this paper's DL-based object detector in remote sensing imagery. Based on the many SOTA DL-based object detectors and their evolution from early models presented in chapter 4, it is established a guide of essential definitions to come up with a functional DL-based object detector model with SOTA performance.

First of all, the type of detection, horizontal or oriented object detection, might be decided. As presented in 4.2, oriented object detection is a bit more complex given its freedom to rotate the bounding-box on the object, but produces tighter and more accurate to the true object's area. Therefore this detection is chosen.

Once the type of detection has been chosen, the selection of an adequate dataset which will be employed to train and evaluate the model is fundamental. Then, a concept between the most present detectors in the literature, i.e., two-stage, one-stage anchor-based or one-stage anchor-free, needs to be selected.

With this set of decisions, a variety of SOTA models are available to serve as the basis of our new model. By evidencing the best qualities of different models, modifications may be conducted on them to build an improved model for the chosen application.

5.1 Dataset

As seen in section 3.6, a dataset has a major impact on 2 important points. The knowledge a supervised NN model can expect to learn relies entirely in the quality of the data on which it is trained. Even a good model may perform poorly if its training data is not good enough. In fact, a small model would perform better than a deeper model with little training data.

A second point that cannot be neglected is the model benchmarks tied to each dataset. For comparison purposes, it is reasonable to go after datasets available open-source, which have already been tested by many models. It gives the perspective of a better comparison of the obtained model and other models already made. Given the computational capacity available, it is not possible to perform evaluations of many models on unpopular datasets to create our own meaningful benchmark.

Therefore, it is chosen the Dataset for Object deTection in Aerial Images (DOTA) (DING

et al., 2021) as the source for training and evaluation data. DOTA is a large-scale dataset for object detection in aerial images. It can be used to develop and evaluate object detectors in aerial images. The images are collected from different sensors and platforms. Each image is of the size in the range from 800 × 800 to 20,000 × 20,000 pixels and contains objects exhibiting a wide variety of scales, orientations, and shapes. The instances in DOTA images are annotated by experts in aerial image interpretation by arbitrary (8 d.o.f.) quadrilateral (DING et al., 2021), describing oriented bounding-box, which is required for the chosen oriented detection.

There are 3 available versions:

– **DOTA-v1.0** contains 15 common 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. It has 2,806 images and 188, 282 instances. The proportions of the training set, validation set, and testing set in DOTA-v1.0 are 1/2, 1/6, and 1/3, respectively.

– **DOTA-v1.5** uses the same images as DOTA-v1.0, but the extremely small instances (less than 10 pixels) are also annotated. Moreover, a new category, ”container crane” is added. It contains 403,318 instances in total. The number of images and dataset splits are the same as DOTA-v1.0.

– **DOTA-v2.0** collects more Google Earth, GF-2 Satellite, and aerial images. There are 18 common categories, 11,268 images and 1,793,658 instances in DOTA-v2.0. Compared to DOTA-v1.5, it further adds the new categories of ”airport” and ”helipad”. The 11,268 images of DOTA are split into training, validation, test-dev, and test-challenge sets.

Because DOTA-v2.0 is a very recent dataset, there are fewer models to be compared in its benchmark and DOTA-v1.5 is judged as sufficient, with enough and high quality data. Therefore, it is selected DOTA-v1.5 as the best option for the project and from now on, when mentioned DOTA, it refers to DOTA-v1.5.

5.2 DL-based object detector concept and model choices

As seen in 4.1, each concept has its pros and cons. It is of interest to have good performance but also to be fast, preferably inferring objects in real time. Thus, the concept selected

is the one-stage detection, which is lighter, faster and presents similar performance to the two-stages.

The choice between anchor-based and anchor-free detector is a question that goes further into the models and their implementations available in the literature. A deep research in the models implementations reveals some troubles. The vast majority of the models presented with their results, performing oriented object detection, either do not make available their implementation or then, present their implementation but not trained.

The effort of finding a pre-trained model is reasonable because of the concept of transfer learning, present in 3.8. Better than initializing our model from random weights, modifying a pre-trained model is much more efficient to obtain great performance. Although the model had been trained for a different task, the absorbed knowledge of some intricate patterns are the same.

Another difficulty found when trying to use and modify some implementations is the lack of organization and good programming practices. Debugging has been demonstrated hard and for the implementations whose pre-trained weights were not made available, it is difficult to confirm the good performance claimed by their authors.

Therefore, the model chosen as the basis for this paper's DL-based oriented object detection in RS imagery is the YOLOv7 (WANG; BOCHKOVSKIY; LIAO, 2022). This is a very recent SOTA model (from July, 2022), with high performance and fast computation, as initially desired. Because YOLOs versions are always precursors of new methods and changes to the literature, they are very well documented and coded, which allows an easy understanding about their implementations and their results are trustworthy.

Furthermore, YOLOv7 is available with its pre-trained weights in the COCO (LIN et al., 2014) dataset, which is a very large dataset and thus, it comes with a good knowledge that can be taken advantage to our application trough transfer learning. These are many reasons for which it has been decided to work over this model.

5.3 Adapting YOLOv7 for OBB in RS imagery

YOLOv7 has been trained on COCO (LIN et al., 2014) dataset. This dataset has annotations for a few different tasks: HBB object detection, keypoint detection, stuff segmentation, panoptic segmentation, densepose, and image captioning. Given the interest to obtain an OBB detector, the most similar option is the model trained for HBB detection.

To allow the model to be trained on DOTA dataset, with OBB detection, the first notice-

able change is in its data loader. COCO and DOTA annotations are as follows:

– **COCO labels:** each object is annotated by an horizontal bounding box (HBB), which can be denoted as $(x, y,$ *width, height)* , where (x, y) denotes the top left image corner. Apart from HBB, each instance is also labeled with a category id. Each line represents an instance: $(x, y,$ *width, height), category_id*.

– **DOTA labels:** each object is annotated by an oriented bounding box (OBB), which can be denoted as $(x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4)$, where (x_i, y_i) denotes the i-th vertice of OBB. The vertices are arranged in a clockwise order. Apart from OBB, each instance is also labeled with a category and a difficulty which indicates whether the instance is difficult to be detected (1 for difficult, 0 for not difficult). Each line represents an instance: $x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4$, *category, difficulty*.

Another necessary change is setting the number of neurons in the last layer of the YOLOv7 head, called *IDetect*. Since COCO has 80 classes of objects, it has 80 final neurons that output the probability and location for objects of each class. DOTA-v1.5 has 16 classes and therefore there are 16 final neurons in the YOLOv7 head.

More complex modifications are necessary to the loss function. YOLOv7 loss function is composed by a combination of losses: box loss, class loss and object loss. Since OBB has a new degree of freedom, being able to spin the bounding-box to better adjust itself, it is imperative to establish a rotation loss and add it to the combined loss.

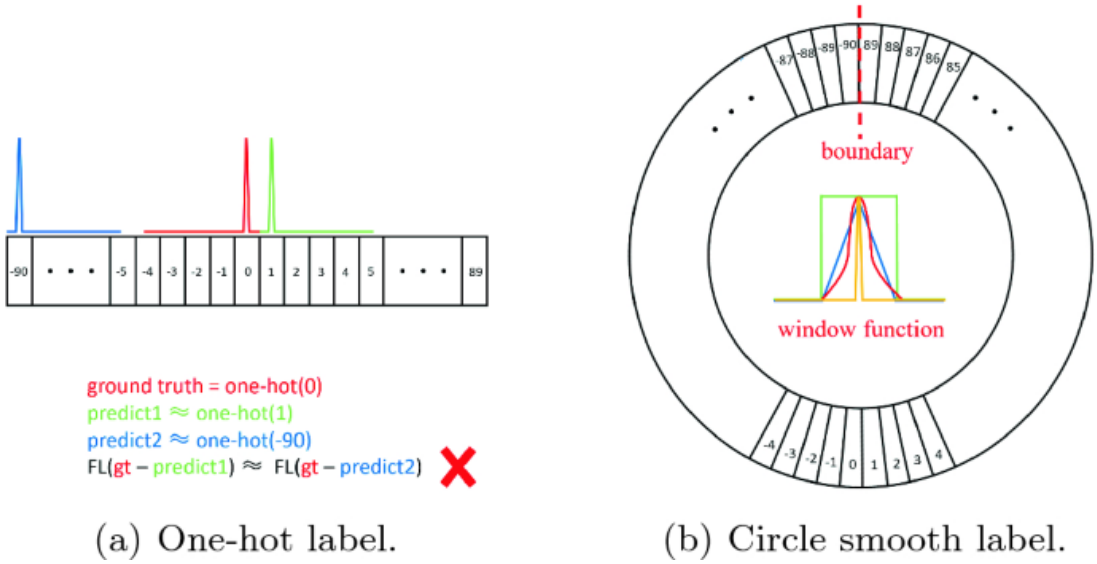
Paying attention to some losses present in the literature, discussed in 4.2, regular regression suffers the discontinuous boundaries problem, caused by angular periodicity or corner ordering (YANG; YAN, 2020). Also in (YANG; YAN, 2020), it finds out that the problem stems from the fact that the ideal predictions are beyond the defined range. It designs a new rotation detection baseline, transforming angular prediction from a regression problem to a classification task with little accuracy loss. Although the conversion from continuous regression to discrete classification, the impact of the lost accuracy on the rotation detection task is negligible.

However, performing the classification thorough one-hot label still has two drawbacks for rotation detection: the exchangeability of edges (EoE) problem from regression still remains and regular classification loss is agnostic to the angle distance between the predicted label and ground-truth, i.e., it does not consider that some classes are almost correct for angles relatively close to the right rotation. To mitigate these problems, (YANG; YAN, 2020) designs a circular

smooth label (CSL), which addresses the periodicity of the angle and increase the error tolerance between adjacent angles.

Fig. 5.1 clarifies that CSL involves a circular label encoding with periodicity, and the assigned label value is smooth with a certain tolerance. Now, instead of only considering a single class as correct and any other as completely incorrect, it accounts some value for predicting classes to rotation angles close to the right class. And it still overcomes the problem of EoE thanks to its periodicity.

Fig. 5.1 – One-hot and circle smooth labels for angular classification. FL means focal loss



(a) One-hot label. (b) Circle smooth label.

Source: Adapted from (YANG; YAN, 2020).

The expression of CSL can be seen in Eq. 5.1.

$$CSL(x) = \begin{cases} g(x), & \theta - r < x < \theta + r \\ 0, & \text{otherwise} \end{cases} \tag{5.1}$$

Where $g(x)$ is a window function, r is the radius of the window function and θ represents the angle of the current bounding box.

To transform this theory to implementation code and allow the integration into the YOLOv7 model, YOLOv5 (AL., 2021) and YOLOv5-OBB (KAIXUAN, 2022) are analyzed. YOLOv5-OBB also uses CSL and thus its loss implementation is very useful, requiring an adaptation to work with the new YOLOv7 model.

Throughout the process of generating proposals, many regions around a same object have similar scores to some extent and are considered as candidate regions, which leads to hundreds of proposals. To solve this problem, a technique called non-maximum suppression

(NMS) filters the proposals based on their confidence and IoU.

Thanks to (WHU, 2018), the NMS is implemented to OBB and its computations can be performed directly on the GPU, which is many times faster and has already been employed in YOLOv5-OBB.

Other minor additional functions are still necessary. An auxiliary function to transform the rectangular bounding-box to the new polygonal coordinate and a function able to scale the polygons as well as it was implemented to the rectangular bounding-box. The image annotator for drawing the ground-truths and predictions in the images must also gain new capabilities in order to be able to draw correctly the new bounding-boxes.

5.4 Training YOLOv7-OBB

As better described in 5.1, DOTA has very high resolution images and therefore the images must be split into smaller images before training/testing to achieve better performance. Thanks to (WHU, 2018), functions have already been made to perform the splitting. Trying to keep an equal basis for the sake of further comparison, it is chosen to divide the images into 1024x1024 pixels with a gap of 240 pixels, as well as made in YOLOv5-OBB.

The initial model weights come from the pre-trained models available at (WANG; BOCHKOVSKIY; LIAO, 2022). Table 5.1 shows the available pre-trained YOLOv7 models.

Table 5.1 – Available pre-trained YOLOv7 models on COCO dataset

Model	Test Size	APtest	$AP_{50}{}^{test}$	$AP_{75}{}^{test}$	batch 1fps	batch 32 average time
YOLOv7	640	51.4%	69.7%	55.9%	161fps	2.8 ms
YOLOv7-X	640	53.1%	71.2%	57.8%	114fps	4.3 ms
YOLOv7-W6	1280	54.9%	72.6%	60.1%	84fps	7.6 ms
YOLOv7-E6	1280	56.0%	73.5%	61.2%	56fps	12.3 ms
YOLOv7-D6	1280	56.6%	74.0%	61.8%	44fps	15.0 ms
YOLOv7-E6E	1280	56.8%	74.4%	62.1%	36fps	18.7 ms

Source: Adapted from (WANG; BOCHKOVSKIY; LIAO, 2022).

Where $APtest$ is the AP value averaged over 10 different IoU thresholds, ranging from 50% to 95% at 5% step-size, AP_{50}^{test} and AP_{75}^{test} are the AP values for IoU thresholds of 50% and 75%, respectively. Batch 1 and 32 means that it has been loaded 1 and 32 images, respectively, in each processed batch.

All these models are variations of YOLOv7, trained for different image resolutions and

number of model parameters. Naturally, the larger the model, the higher the model performance, but the slower its computation becomes. We decide to start with the smallest model, called YOLOv7 in the table 5.1, because it already performs well and is light enough to work in many circumstances.

Fig. 5.2 shows an example of the input image for training the model. It is a random mosaic of the images available in DOTA, generated by the YOLOv7 data loader. Mosaics are adopted as a form of data augmentation, since the images can be combined in a variety of ways, further increasing the ability of the model to find objects in mixed scenarios.

Fig. 5.2 – Example of random input mosaic for training the model

Source: Author.

For training, there is a collection of hyperparameters that can be tuned in order to adjust the progress of model weight updates. The table 5.2 has a list of the hyperparameters that control the training, along with their description and values. The values adopted are a middle ground between the values used in YOLOv7 and YOLOv5-OBB, which are very similar.

Throughout training, there are 5 losses that are kept under inspection: box loss, objectness loss, class loss, theta loss, and total loss (their combination). At the end of each epoch, the evaluation data is tested on the model to generate 4 metrics: precision, recall, $mAP_{0.5}$ and

Table 5.2 – Training hyperparameters

Hyperparameter	Value	Description
lr0	0.01	initial learning rate (SGD=1E-2, Adam=1E-3)
lrf	0.2	final OneCycleLR learning rate (lr0 * lrf)
momentum	0.937	SGD momentum/Adam beta1
weight_decay	0.0005	optimizer weight decay 5e-4
warmup_epochs	3.0	warmup epochs
warmup_momentum	0.8	warmup initial momentum
warmup_bias_lr	0.1	warmup initial bias lr
box	0.05	box loss gain
cls	0.5	cls loss gain
cls_pw	1.0	cls BCELoss positive_weight
theta	0.5	theta loss gain
theta_pw	1.0	theta BCELoss positive_weight
obj	1.0	obj loss gain (scale with pixels)
obj_pw	1.0	obj BCELoss positive_weight
iou_t	0.20	IoU training threshold
anchor_t	4.0	anchor-multiple threshold
# anchors	3	anchors per output layer
fl_gamma	0.0	focal loss gamma (efficientDet default gamma=1.5)
hsv_h	0.015	image HSV-Hue augmentation (fraction)
hsv_s	0.7	image HSV-Saturation augmentation (fraction)
hsv_v	0.4	image HSV-Value augmentation (fraction)
degrees	180.0	image rotation (+/- deg)
translate	0.1	image translation (+/- fraction)
scale	0.5	image scale (+/- gain)
shear	0.0	image shear (+/- deg)
perspective	0.0	image perspective (+/- fraction), range 0-0.001
flipud	0.5	image flip up-down (probability)
fliplr	0.5	image flip left-right (probability)
mosaic	0.85	image mosaic (probability)
mixup	0.1	image mixup (probability)
copy_paste	0.0	image copy paste (probability)
paste_in	0.0	image copy paste (probability), use 0 for faster training
loss_ota	0	use ComputeLossOTA, use 0 for faster training
cls_theta	180	number of theta classes
csl_radius	2.0	number of radius classes

Source: Author.

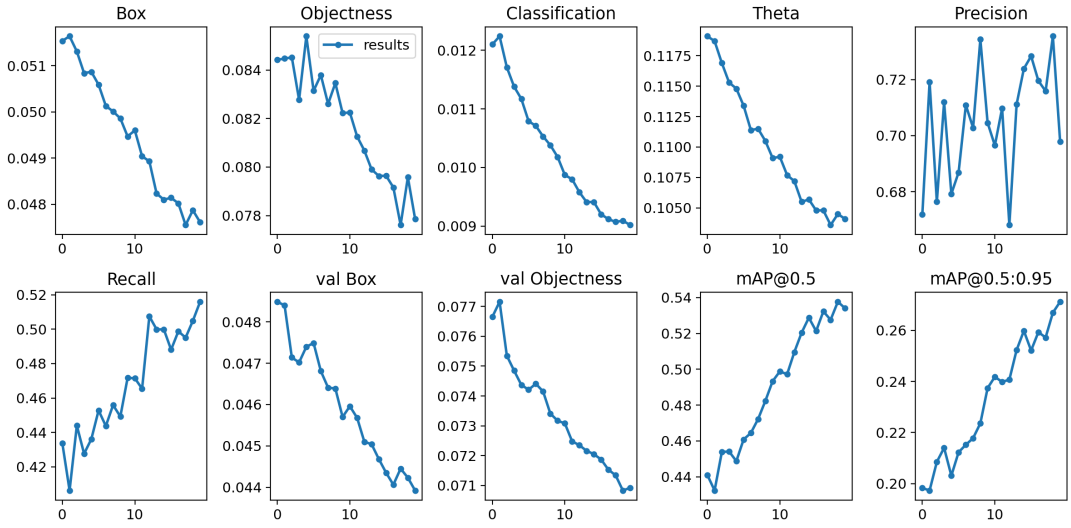
$\text{mAP}_{0.5:0.95}$.

Initially, the idea was to train the model using our local computer. However, the available graphic card (GPU) NVIDIA RTX 2070 (NVIDIA, 2018) only has 8 GB of memory. For the selected input size image of 1024x1024 pixels, the maximum number of images per batch, supported by the GPU, is 2, whereas previously it was 75 in the YOLOv5-OBB fine-tuning

process, due to the employment of many GPUs with more significant memory. Therefore, the training was too slow. In fact, when training the model with this configuration, it was not converging. As explained in (LARSSON, 2019), the size of the learning rate is limited, among other factors, by the batch size. Updating weights from errors calculated in small batches is noisier and can prevent the descent from completely converging to an optima at all. On the other hand, it can also be good, helping it to dodge local minima.

By reducing the learning rate to 1/100 from the initial learning rate, i.e., 1e-4, the model was capable of training and learning. Fig. 5.3 shows the evolution of the most important losses and metrics throughout the training for 20 epochs.

Fig. 5.3 – Evolution of losses and metrics throughout the training with small initial learning rate



Source: Author.

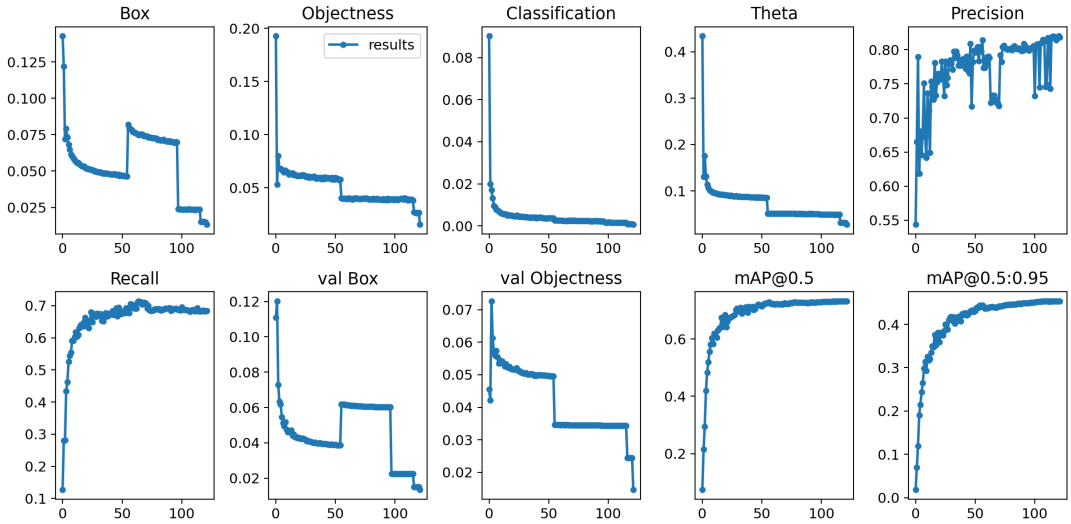
The x-axis is the epoch number and the y-axis is the metrics/losses values. These results suggest that the implemented model works. As it trains, lower its losses decrease and metrics increase. It is still far from good though. Starting training with a very low learning rate greatly reduces how much it improves with each interaction, so a large number of interactions would be required. With an average of a few hours per epoch, training could take months.

Given the circumstances, it was necessary to search for different online solutions. Google Collaboratory (GOOGLE, 2022a), or Colab, allows running python scripts on virtual machines (VMs) with fine hardware, GPUs in particular, which are needed for model training. Resources are limited, and to train our model, a Colab Pro+ subscription was required to gain enough computational time to train the model. The codes were downloaded from the GitHub repository (SANTOS, 2022) and the prepared dataset was first uploaded to Google Drive to be downloaded

by the VM to train the model.

Using the A100 GPU (NVIDIA, 2022) with 40GB of memory, provided by the VM, it was possible to increase the batch size to 18 images. Fig. 5.4 shows the evolution of the same losses and metrics shown in Fig. 5.3 for 125 epochs with the same initial learning rate used in the YOLOv7 and YOLOv5-OBB training.

Fig. 5.4 – Evolution of losses and metrics throughout the training with greater initial learning rate



Source: Author.

The x-axis is the epoch number and the y-axis is the metrics/losses values. Looking at Fig. 5.4, one can see the great improvement of the model as epochs pass and box, objectness, class and theta losses decrease, and precision, recall, $mAP_{0.5}$ and $mAP_{0.5:0.95}$ metrics increase, as well as the validation box and objectness losses also decrease.

Around the 50th and 100th epoch, there are 2 abrupt changes in the losses. These were caused by manual changes in the loss weights in an attempt to speed up the training process. The more traditional way is to let the weight decay exponentially reduce the learning rate as it trains, but given the limited computational time given in the VM, the losses were changed manually. In chapter 6, more on the implications and possible damage to model training are addressed.

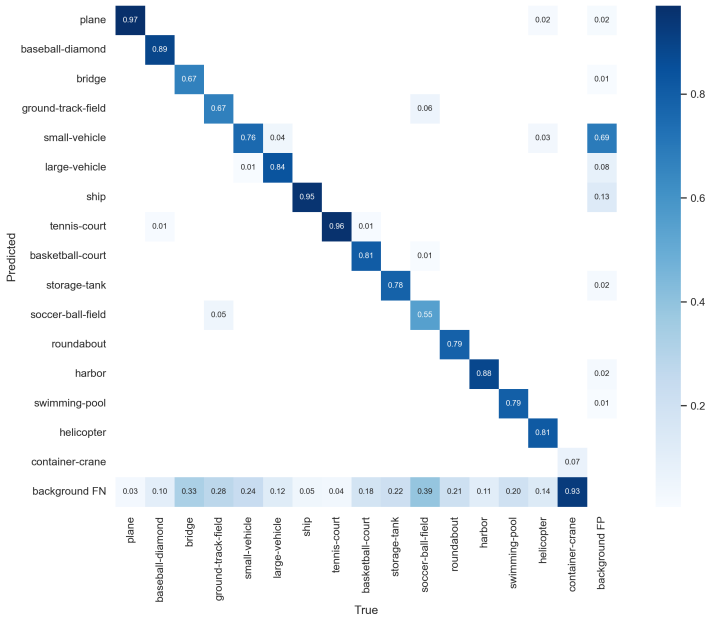
6 RESULTS

In this chapter, the usually evaluated metrics in object detection, described at section 3.9, are presented, analyzed and compared with YOLOv5-OBB results. The second part brings some detections using the model, performed in some interesting areas.

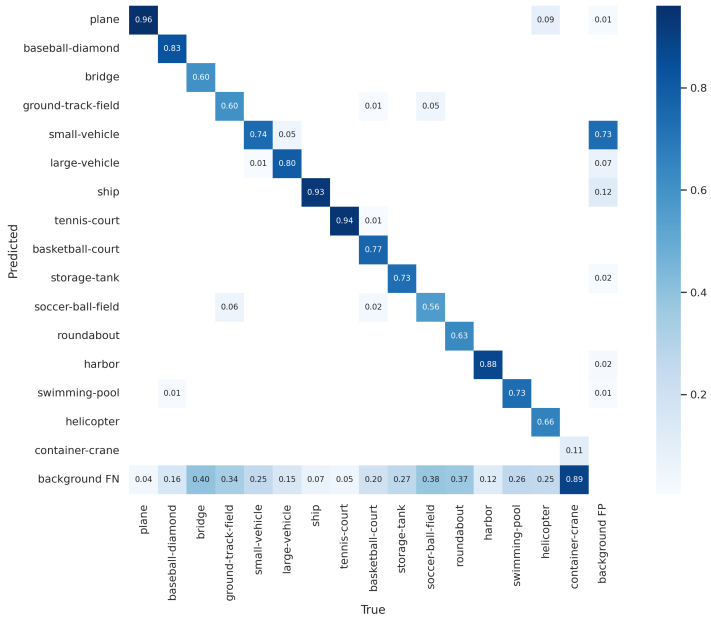
6.1 Metrics

The first metric to be addressed is the confusion matrix. Fig. 6.1 presents our model's confusion matrix in contrast with YOLOv5-OBB.

Fig. 6.1 – YOLOv7-OBB vs YOLOv5-OBB's confusion matrix



(a) YOLOv7-OBB (our model)



(b) YOLOv5-OBB

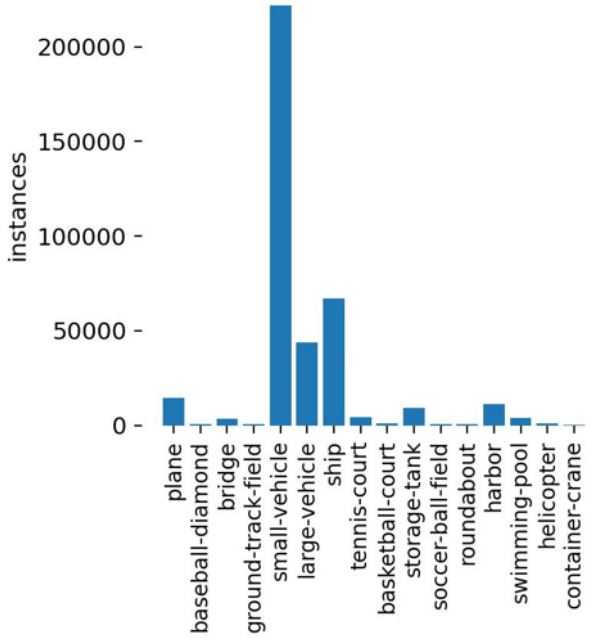
Source: (a) Author. (b) Adapted from (KAIXUAN, 2022).

Comparing the confusion matrices, it is evident the higher performance of our model, YOLOv7-OBB. For instance, it correctly identifies with a probability of 0.97, 0.89, 0.67 and 0.67% the following classes: plane, baseball diamond, bridge and ground track field. Meanwhile the YOLOv5-OBB obtains 0.96, 0.83, 0.6 and 0.6% for the same respective classes. Another interesting point is the reduction from 0.73 to 0.69% in the probability of mistaking

the background for a small-vehicle.

The only exception occurs for the container crane. While our model only correctly identifies container cranes 0.07% of the time, YOLOv5-OBB manages 0.11%. This exception might be explained by checking the class distribution from the training dataset, DOTA, fig. 6.2. The container crane is the least present object in the entire dataset. One should bear in mind that our model was trained for 125 epochs, while YOLOv5-OBB was trained on more than double that, for 300 epochs. So there is a noticeable problem in reducing the number of epochs in our training. It would be recommended to continue training further in more epochs, to allow the model to see container cranes more often and learn them better.

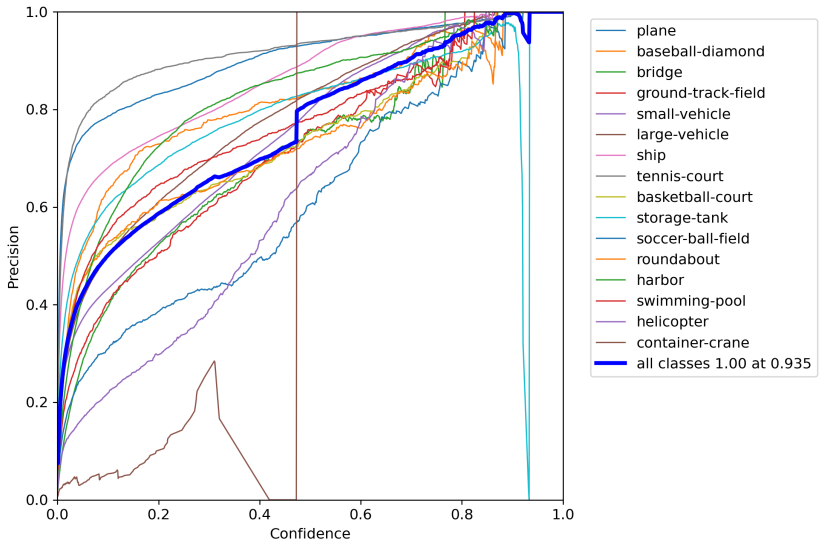
Fig. 6.2 – Class distribution of DOTA



Source: Author.

The precision curve, shown at Fig. 6.3, allows to examine how changing the confidence threshold changes the detection precision for each class of object. It would be desirable to have the curves as high as possible, even at low confidence. Again, it can be seen that the model performs differently for different object classes. Caused by the different complexities in discriminating each object and by training on an unbalanced dataset. The container crane has a distinctive deficient precision.

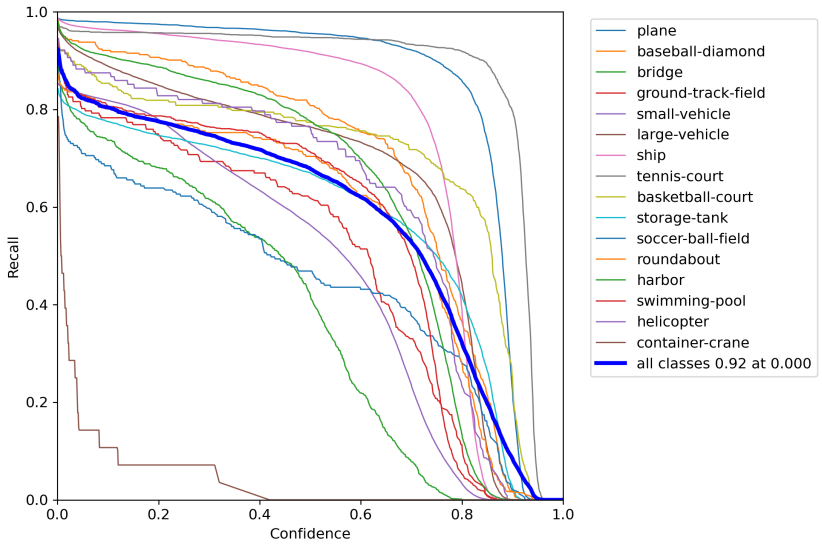
Fig. 6.3 – Precision curve



Source: Author.

But increasing the confidence to achieve better precision is not good because of recall. As can be seen in Fig. 6.4, recall tends to reduce by increasing the confidence threshold, simply because if the confidence is higher, more objects are overlooked because they lack the necessary confidence and then fewer objects are identified. Again, the container crane has a distinct deficient recall.

Fig. 6.4 – Recall curve

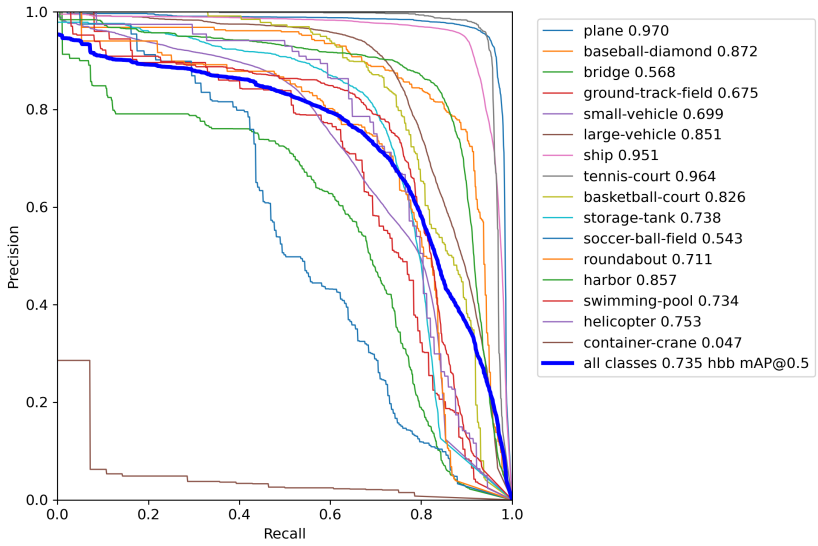


Source: Author.

The precision-recall curve, Fig. 6.5, brings these two metrics together. Ultimately, the

quest is to have high precision and high recall, but as seen in the previous curves, increasing recall makes it difficult to keep the precision high as well. All classes perform relatively similarly, with the exception of the container crane.

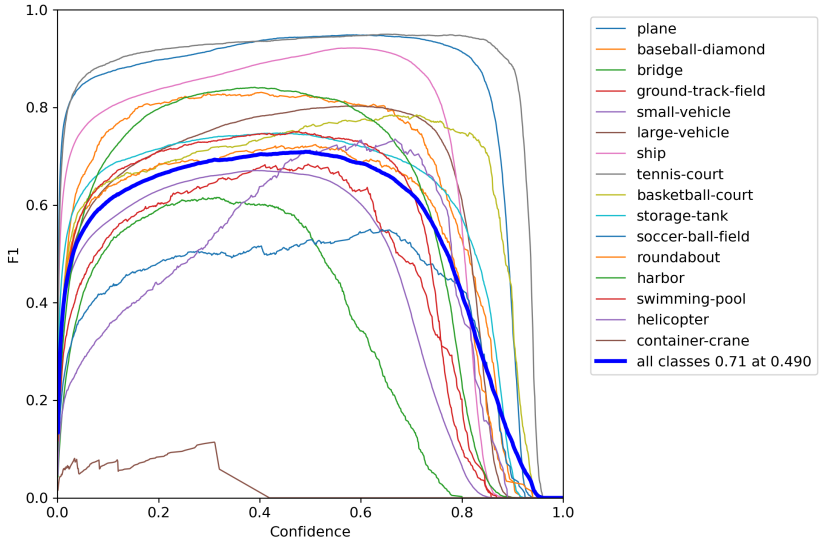
Fig. 6.5 – Precision-recall curve



Source: Author.

The last curve that synthesizes the model performance is the F1 curve, in Fig 6.6. Similar to the analysis of the other curves, the model tends to perform more or less the same for all classes except for the container crane.

Fig. 6.6 – F1 curve



Source: Author.

6.2 Detections

Statistically evaluating the model is of great importance. But now, this section brings some results in practice. First, some areas in Brazil are chosen to be scanned and the objects present in them, detected. All the images are taken directly from Google Maps (GOOGLE, 2022b).

The first interesting area is the Santa Maria Air Force Base, Rio Grande do Sul. The model was able to detect the following objects: 1 airplane, 1 basketball court, 96 cars, 1 tennis court and 1 helicopter.

Fig. 6.7 – Detection: Santa Maria Air Force Base, Rio Grande do Sul

Source: Author.

The second place is the campus in the Federal University of Santa Maria. In the Fig.

6.8, 748 small-vehicles, 2 large-vehicles and 5 storage-tanks have been detected.

Fig. 6.8 – Detection: Federal University of Santa Maria Campus, Rio Grande do Sul

Source: Author.

The third area is the Planalto Bus Parking in Santa Maria. In the Fig. 6.9, 1 ground-track-field, 56 small-vehicles and 35 large-vehicles have been detected.

Fig. 6.9 – Detection: Planalto Bus Parking, SM, Rio Grande do Sul

Source: Author.

Another place is the airport of Congonhas, São Paulo, because of its high airplane traffic. In the Fig. 6.10, 30 planes, 2 baseball-diamonds, 406 small-vehicles, 11 large-vehicles and 1 soccer-ball-field have been detected.

Fig. 6.10 – Detection: Congonhas airport, São Paulo

Source: Author.

One last place to be evaluated is the port of Santos, São Paulo, because of its high ship traffic. In the Fig. 6.11, 125 small-vehicles, 40 large-vehicles, 151 ships, 37 storage-tanks, 7 harbors and 1 swimming-pool have been detected.

Fig. 6.11 – Detection: Port of Santos, São Paulo

Source: Author.

7 CONCLUSION

This project met everything that was expected for the final work of the undergraduate course. It covered all the aspects from a initial research about RS, passed to machine learning architectures, capabilities, metrics, datasets, SOTA methods, etc. Finally, the union of this various knowledge culminated in the development of a DL-based model with SOTA performance in RS imagery.

Throughout the research process, many possibilities emerged to solve the problem of object detection in RS images, which could guide this work to different projects and performance. The first choice made was the detection type, oriented detection, because it allows a better fit of the bounding boxes to the objects. Next, the important trade-off between speed and inference quality filtered the available models, choosing a fast one with still great performance.

YOLO-v7 has served as the basis model for our development, a fast and with great performance model. However, since it only works with horizontal detection, it has been adapted with a new loss function, CSL. The results demonstrate that the model was able to effectively learn and reduce its losses as well as increase its metrics. Through a direct comparison with the already implemented predecessor, YOLOv5-OBB, it became evident that the developed model is superior detecting almost every object class in the DOTA-v1.5 dataset, except by the container crane.

The insufficiency of achieving better detection with the container crane object class was attributed to the smaller number of epochs for which the model was trained compared to YOLOv5-OBB. Training DL-based models can be computationally very expensive and it is a suggestion for future work to have better machines available for training to the limit of the models, extracting the best of each model. Also as a suggestion for future work, it would be interesting to try different losses, such as converting the rotating bounding box to a 2-D Gaussian distribution and other approximations described in the section 4.2.

The final model can have many real implications. Using the DOTA-v1.5 dataset, the model is capable of detecting 16 object classes in RS imagery: plane, ship, storage tank, baseball diamond, tennis court, basketball court, ground track field, harbor, bridge, large vehicle, small vehicle, helicopter, roundabout, soccer ball field, swimming pool and container crane. The model could be run in real time on incoming images of drones and other cameras, detecting these objects that can then be used to count or perform logistics management in parking lots,

streets, warehouses, etc.

By creating custom datasets with objects of interest is possible to retrain the model and expand the number of detected objects. For example, the model could learn to detect and distinguish plants, to count and estimate yield. It could even detect spots and marks that could determine problems in the crop. For livestock, the model could count the animals and detect animals lost in the field. In military applications, the model can detect different types of exposed structures and armaments, as well as cars, traffic, and forest fires, in the Amazon forest for example. The developed model even won as the highlight "Social Impact - Engineering", in the II Symposium of Academic Exchange (SAE), UFSM, 2022.

REFERENCES

Abe, T. et al. Deep Ensembles Work, But Are They Necessary? **arXiv e-prints**, [S.l.], p.arXiv:2202.06985, Feb. 2022.

AL., G. J. et. **ultralytics/yolov5: v6.0 - yolov5n 'nano' models, roboflow integration, tensorflow export, opencv dnn support**. [S.l.]: Zenodo, 2021.

BOURMAUD, G. **Introduction aux réseaux de neurones pour l'apprentissage supervisé**. 2020.

BROWNLEE, J. **What is the Difference Between Test and Validation Datasets?** Access on December 2021, https://machinelearningmastery.com/difference-test-validation-datasets/.

CZAKON, J. **F1 Score vs ROC AUC vs Accuracy vs PR AUC**: which evaluation metric should you choose? Access on November 2022, https://neptune.ai/blog/f1-score-accuracy-roc-auc-pr-auc.

DIETRICH, J. **New Off Nadir Resolution Calculator**. Access on July 2022, http://adv-geo-research.blogspot.com/2016/08/off-nadir-res-calc.html.

DING, J. et al. Object Detection in Aerial Images: A large-scale benchmark and challenges. **CoRR**, [S.l.], v.abs/2102.12219, 2021.

DIRECT, S. **Remote Sensing Image**. Access on November 2021, https://www.sciencedirect.com/topics/computer-science/remote-sensing-image.

GIRSHICK, R. Fast R-CNN. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION (ICCV), 2015. **Anais...** [S.l.: s.n.], 2015. p.1440–1448.

GIRSHICK, R. B. et al. Rich feature hierarchies for accurate object detection and semantic segmentation. **CoRR**, [S.l.], v.abs/1311.2524, 2013.

GISGEOGRAPHY. **What is Remote Sensing? The Definitive Guide**. Access on May 2022, https://gisgeography.com/remote-sensing-earth-observation-guide/.

GOODFELLOW, I. et al. Generative Adversarial Nets. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS. **Anais...** Curran Associates: Inc., 2014. v.27.

GOODFELLOW, I. J.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. Cambridge, MA, USA: MIT Press, 2016. 25; 140; 146p. http://www.deeplearningbook.org.

GOOGLE. **Google Colaboratory**. Access on August 2022, https://colab.research.google.com/.

GOOGLE. **Google Maps**. Access on November 2022, https://www.google.com.br/maps.

HE, K. et al. Deep Residual Learning for Image Recognition. **CoRR**, [S.l.], v.abs/1512.03385, 2015.

HUANG, G.; LIU, Z.; WEINBERGER, K. Q. Densely Connected Convolutional Networks. **CoRR**, [S.l.], v.abs/1608.06993, 2016.

HUI, J. **What do we learn from region based object detectors (Faster R-CNN, R-FCN, FPN)?** Access on October 2021, https://jonathan-hui.medium.com/what-do-we-learn-from-region-based-object-detectors-faster-r-cnn-r-fcn-fpn-7e354377a7c9.

HUI, J. **What do we learn from single shot object detectors (SSD, YOLOv3), FPN Focal loss (RetinaNet)?** Access on October 2021, https://jonathan-hui.medium.com/what-do-we-learn-from-single-shot-object-detectors-ssd-yolo-fpn-focal-loss-3888677c5f4d.

ILLARIONOVA, S. et al. Object-Based Augmentation Improves Quality of Remote SensingSemantic Segmentation. **CoRR**, [S.l.], v.abs/2105.05516, 2021.

IOFFE, S.; SZEGEDY, C. Batch Normalization: accelerating deep network training by reducing internal covariate shift. **CoRR**, [S.l.], v.abs/1502.03167, 2015.

ISAKSSON, M. **Four Common Types of Neural Network Layers**. Access on October 2021, https://towardsdatascience.com/four-common-types-of-neural-network-layers-c0d3bb2a966c: :text=The

JOSEPH, T. **The Mathematics Behind Deep Learning**. Access on October 2021, https://towardsdatascience.com/the-mathematics-behind-deep-learning-f6c35a0fe077.

KAIXUAN, H. **YOLOv5-OBB**. Access on August 2022, https://github.com/hukaixuan19970627/yolov5$_o bb$.

Kingma, D. P.; Welling, M. Auto-Encoding Variational Bayes. **arXiv e-prints**, [S.l.], p.arXiv:1312.6114, Dec. 2013.

KOECH, K. E. **Object Detection Metrics With Worked Example**. Access on July 2022, https://towardsdatascience.com/on-object-detection-metrics-with-worked-example-216f173ed31e: :text=Average%20Precision%20(AP)%20and%20mean,COCO%20and%20PASCAL%20V

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. ImageNet Classification with Deep Convolutional Neural Networks. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS. **Anais...** Curran Associates: Inc., 2012. v.25.

LARSSON, S. **Possible for batch size of neural network to be too small?** Access on August 2022, https://datascience.stackexchange.com/questions/52884/possible-for-batch-size-of-neural-network-to-be-too-small.

LIN, T. et al. Focal Loss for Dense Object Detection. **CoRR**, [S.l.], v.abs/1708.02002, 2017.

LIN, T.-Y. et al. **Microsoft COCO**: common objects in context. cite arxiv:1405.0312Comment: 1) updated annotation pipeline description and figures; 2) added new section describing datasets splits; 3) updated author list.

LIN, T.-Y. et al. Feature Pyramid Networks for Object Detection. In: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR), 2017. **Anais...** [S.l.: s.n.], 2017. p.936–944.

LIU, W. et al. SSD: single shot multibox detector. In: COMPUTER VISION – ECCV 2016, Cham. **Anais...** Springer International Publishing, 2016. p.21–37.

MAHONY, N. O. et al. Deep Learning vs. Traditional Computer Vision. **CoRR**, [S.l.], v.abs/1910.13796, 2019.

MNIH, V. et al. Human-level control through deep reinforcement learning. **Nature**, [S.l.], v.518, n.7540, p.529–533, Feb. 2015.

NARKHEDE, S. **Understanding Confusion Matrix**. Access on August 2022, https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62.

NVIDIA. **GeForce RTX 20 Series**. Access on September 2022, https://www.nvidia.com/en-us/geforce/20-series/.

NVIDIA. **NVIDIA A100 Tensor Core GPU**. Access on September 2022, https://www.nvidia.com/en-us/data-center/a100/.

PAPERSWITHCODE. **Browse State-of-the-Art**. Access on July 2022, https://paperswithcode.com/sota.

QIAN, W. et al. Learning Modulated Loss for Rotated Object Detection. **CoRR**, [S.l.], v.abs/1911.08299, 2019.

REDMON, J. **Darknet**. Access on 2021, http://pjreddie.com/darknet/.

REDMON, J. et al. You Only Look Once: unified, real-time object detection. **CoRR**, [S.l.], v.abs/1506.02640, 2015.

REDMON, J.; FARHADI, A. YOLO9000: better, faster, stronger. **CoRR**, [S.l.], v.abs/1612.08242, 2016.

REDMON, J.; FARHADI, A. YOLOv3: an incremental improvement. **CoRR**, [S.l.], v.abs/1804.02767, 2018.

REN, S. et al. Faster R-CNN: towards real-time object detection with region proposal networks. In: NIPS. **Anais...** [S.l.: s.n.], 2015. p.91–99.

REZATOFIGHI, H. et al. Generalized Intersection Over Union: a metric and a loss for bounding box regression. In: IEEE/CVF CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR), 2019. **Anais...** [S.l.: s.n.], 2019. p.658–666.

RONNEBERGER, O.; FISCHER, P.; BROX, T. U-Net: convolutional networks for biomedical image segmentation. **CoRR**, [S.l.], v.abs/1505.04597, 2015.

RUSSAKOVSKY, O. et al. ImageNet Large Scale Visual Recognition Challenge. **International Journal of Computer Vision (IJCV)**, [S.l.], v.115, n.3, p.211–252, 2015.

SANTOS, P. T. P. dos. **YOLOv7-OBB**. Access on November 2022, https://github.com/3-14o/yolov7_obb.

SHIN, Y.; BALASINGHAM, I. Comparison of hand-craft feature based SVM and CNN based deep learning framework for automatic polyp classification. In: 2019 . **Anais...** [S.l.: s.n.], 2017. v.2017, p.3277–3280.

Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. **arXiv e-prints**, [S.l.], p.arXiv:1409.1556, Sept. 2014.

SREEKUMAR, G. **How to interpret smooth l1 loss?** Access on May 2022, https://stats.stackexchange.com/users/168982/gautam-sreekumar.

SRIVASTAVA, N. et al. Dropout: a simple way to prevent neural networks from overfitting. **Journal of Machine Learning Research**, [S.l.], v.15, n.56, p.1929–1958, 2014.

TIAN, Z. et al. FCOS: fully convolutional one-stage object detection. **CoRR**, [S.l.], v.abs/1904.01355, 2019.

UIJLINGS, J. et al. Selective Search for Object Recognition. **International Journal of Computer Vision**, [S.l.], v.104, p.154–171, 09 2013.

USGS. **What are the differences between data, a dataset, and a database?** Access on November 2021, https://www.usgs.gov/faqs/what-are-differences-between-data-dataset-and-database.

WANG, C.-Y.; BOCHKOVSKIY, A.; LIAO, H.-Y. M. YOLOv7: trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. **arXiv preprint arXiv:2207.02696**, [S.l.], 2022.

WHU, C. **DOTA devkit**. Access on November 2022, https://github.com/CAPTAIN-WHU/DOTA$_{d}evkit$.

WIKIPEDIA. **Remote sensing**. Access on November 2021, https://en.wikipedia.org/wiki/Remote$_{s}ensing$.

WIKIPEDIA. **Transfer learning**. Access on May 2022, https://en.wikipedia.org/wiki/Transfer$_{l}earning$: : $text$ = $Transfer\%20learning\%20(TL)\%20is\%20a, when\%20trying\%20to\%20recognize\%20trucks$.

YANG, X. et al. Rethinking Rotated Object Detection with Gaussian Wasserstein Distance Loss. **CoRR**, [S.l.], v.abs/2101.11952, 2021.

YANG, X. et al. Learning High-Precision Bounding Box for Rotated Object Detection via Kullback-Leibler Divergence. **CoRR**, [S.l.], v.abs/2106.01883, 2021.

YANG, X. et al. The KFIoU Loss for Rotated Object Detection. **CoRR**, [S.l.], v.abs/2201.12558, 2022.

YANG, X.; YAN, J. Arbitrary-Oriented Object Detection with Circular Smooth Label. In: COMPUTER VISION – ECCV 2020: 16TH EUROPEAN CONFERENCE, GLASGOW, UK, AUGUST 23–28, 2020, PROCEEDINGS, PART VIII, Berlin, Heidelberg. **Anais...** Springer-Verlag, 2020. p.677–694.

ZHONG, B.; AO, K. Single-Stage Rotation-Decoupled Detector for Oriented Object. **Remote Sensing**, [S.l.], v.12, n.19, 2020.

ZHOU, D. et al. IoU Loss for 2D/3D Object Detection. **CoRR**, [S.l.], v.abs/1908.03851, 2019.

ZHU, X. et al. Deep Learning Meets SAR. **ArXiv**, [S.l.], v.abs/2006.10027, 2020.

ZHUANG, S. et al. A Single Shot Framework with Multi-Scale Feature Fusion for Geospatial Object Detection. **Remote Sensing**, [S.l.], v.11, 03 2019.

NUP: 23081.009983/2023-97 **Prioridade:** Normal

Homologação de ata de defesa de TCC e estágio de graduação

125.322 - Bancas examinadoras de TCC: indicação e atuação

COMPONENTE

Ordem	Descrição	Nome do arquivo
5	Trabalho de Conclusão de Curso	TCC_final Pietro.pdf

Assinaturas

25/01/2023 11:38:22

NATANAEL RODRIGUES GOMES (PROFESSOR DO MAGISTÉRIO SUPERIOR)
07.09.14.00.0.0 - CURSO DE ENGENHARIA EM TELECOMUNICAÇÕES - CETEL

26/01/2023 12:10:49

DANIEL FERNANDO TELLO GAMARRA (PROFESSOR DO MAGISTÉRIO SUPERIOR)
07.54.00.00.0.0 - DEPARTAMENTO DE PROCESSAMENTO DE ENERGIA ELÉTRICA - DPEE