UCLouvain

École polytechnique de Louvain

# Joint optimization of mega-image compression and detection by deep learning

Author: **Cyril Leconte**
Supervisor: **Benoît Macq**
Readers: **Karim El Khoury**, **Tiffanie Godelaine**, **Jean-Didier Legat**
Academic year 2022–2023
Master [120] in Mathematical Engineering

## Abstract

**Context** Object detection is a critical task in computer vision that involves identifying and locating objects of interest within an image or video. It has become increasingly important in recent years due to its wide range of applications in various fields, such as autonomous driving, surveillance, robotics, and healthcare. There are multiple machine learning methods for object detection, based on neural networks or not, but most recent and efficient methods use Deep Learning.

These learning methods require vast amounts of data to achieve the most effective models. The exchange and storage of such large amounts of data also poses significant challenges as it is time-consuming, costly and the data flow increases exponentially. Hence, it is often more advisable to compress these data, usually with loss. The challenge then becomes a trade-off between the loss of information due to compression and the model's detection performance. To find the best balance, it is necessary to use efficient compression methods that minimize image size while retaining maximum information. It is equally essential to develop the most robust models possible to deal with this compression. This latter point is the subject of this work.

**Material and methods** The experiments are conducted using the DOTA dataset and YOLOv5 as deep learning detection algorithm. The images are compressed using JPEG2000 coding system. The first part of the experiments concerns the optimisation of the training of the network by pre-processing the dataset, training on compressed images and fine-tuning. The second part focuses on applying and analyzing ensemble methods, namely Test-time augmentation (TTA) and models ensembling.

**Results** Optimizing the training in the scope of detecting objects on compressed images led to an improvement of the performances by 50%. The application of TTA resulted in a notable performance boost of up to 4%, specifically on small objects and low-compressed images. By selecting models carefully, ensembling demonstrated a slight improvement of around 1%. The combination of the two methods produced the best performance measured on highly compressed images with an improvement of around 2%.

**Conclusion** Training optimizations and ensemble methods can be joint to enhance the detection of objects by deep CNNs on compressed images. Moreover, recent publications and ongoing research hold promise for further advancements in this field.

# Acronyms

**CNN** Convolutional Neural Network. 1, 3, 9, 10, 12, 18, 19, 44

**DWT** Discrete Wavelet Transform. 6, 7, 46

**FPN** Feature Pyramid Network. 12, 13

**HBB** Horizontal Bounding Box. 20, 21

**IOU** Intersection Of Union. 14, 15, 31, 33

**mAP** mean Average Precision. 10, 14, 15, 19, 21–29, 32–43, 46

**OBB** Oriented Bounding Box. 20

**TTA** Test-time augmentation. 1–3, 17, 18, 31–33, 38–40, 43, 46

**YOLO** You Only Look Once. 3, 8–14, 18, 19, 21–23, 31, 44, 45, 49, 50

# Contents

# Chapter 1

# Introduction

## 1.1 Context

Object detection is a critical task in computer vision that involves identifying and locating objects of interest within an image or video. It has become increasingly important in recent years due to its wide range of applications in various fields, such as autonomous driving, surveillance, robotics, and healthcare. There are multiple machine learning methods for object detection, based on neural networks or not, but most recent and efficient methods use Deep Learning.

These learning methods require vast amounts of data to achieve the most effective models. However, these data, especially images in the context of object detection, can vary in resolution and quality. Therefore, it is important and necessary to have models that are robust in the face of this diversity of information forms. The exchange of such large amounts of data also poses a significant challenge as it is time-consuming and the data flow increases exponentially. Their storage is also costly and not convenient. Hence, it is often more advisable to compress these data, usually with loss. The challenge then becomes a trade-off between the loss of information due to compression and the model's detection performance. To find the best balance, it is necessary to use efficient compression methods that minimize image size while retaining maximum information. It is equally essential to develop the most robust models possible to deal with this compression. This latter point is the subject of this work.

## 1.2 Objectives and plan of this thesis

The primary goal of this work is to optimize the detection of objects, such planes and cars in this context, on compressed (aerial) images by applying combination of state-of-the-art techniques. The first chapter will be a state-of-the-art review of the

involved materials. The next part will cover the optimization of the training of the neural network used for the detection of objects with goal to increase robustness against compression of the images. The third part will then review the application of ensemble techniques on the models obtained from previous chapter. Finally, a conclusion will review results, limitations and future possible works.

## 1.3 Writing methodology

The writing of this work has been assisted by the use of ChatGPT, an AI language model developed by OpenAI ([1],[2]). It was utilized to enhance the writing process by exploring alternative phrasings, translating ideas into coherent sentences, restructuring complex sentences with goal to ensure clarity. To structure the document and maintain consistent formatting, cross-referencing, and citation management, LaTeXproved invaluable. Overleaf, an online LaTeXeditor, was utilized, enabling seamless transition across multiple devices and easy sharing of this work for feedback, without the need of any user installation.

# Chapter 2

# State-of-the-art review

This chapter will in first place provide a comprehensive overview of the state-of-the-art and scientific background of all the materials involved in this work. Subsequently, a concise picture of related works will be presented.

## 2.1 JPEG2000 compression

JPEG2000 is a compression standard and a file format that was developed by the Joint Photographic Experts Group (JPEG) committee, published in 2000 as an improvement over the original JPEG standard [3]. The goal of JPEG2000 is to achieve better compression ratios and image quality, particularly at lower bitrates. The wavelet-based approach also enables JPEG2000 to support a range of different compression modes, from lossless to highly lossy depending on the adjustment of the many available parameters. JPEG2000 also includes a number of advanced features compared to the original JPEG standard, such as the ability to store multiple resolutions of an image in a single file, and the ability to selectively decode regions of an image without decoding the entire file (ROI). Despite its technical advantages, JPEG2000 is not widely used and is still not supported by many popular web browsers. This is partly due to its complexity and the high computational requirements for encoding JPEG2000 files. It is, however, the norm in digital cinema.

### 2.1.1 Technical Overview

JPEG2000 uses wavelet-based compression instead of the discrete cosine transform (DCT) used by the original JPEG standard. Wavelets are small oscillations (like a wave) starting and finishing with zero (see Fig. 2.1). The wavelet transform (DWT) is used to represent a signal (an image is a 2D discrete signal, with multiple

channels for RGB) using wavelet as base. This transformation follows a Laplace pyramid structure : at each decomposition layer, the image is convoluted using kernels with wavelets in horizontal, vertical and diagonal direction of the image , namely wavelets filters (Fig. 2.2). Those three new representations with half of the resolution of the original image are called details. The image is also sub-sampled by a factor two using an average convolution, retaining low frequencies (example shown on Fig. 2.3). Details can be interpreted as high frequencies, such edges, in each direction. This operation is reversible, and the original image can be reconstructed without any loss. To achieve efficient lossy and lossless compression, the transformed representations of the image are then quantized and encoded (divided in code-blocks). Adjustable parameters include compression ratio of layer, quality of layer, number of DWT decompositions, code-block size, ...

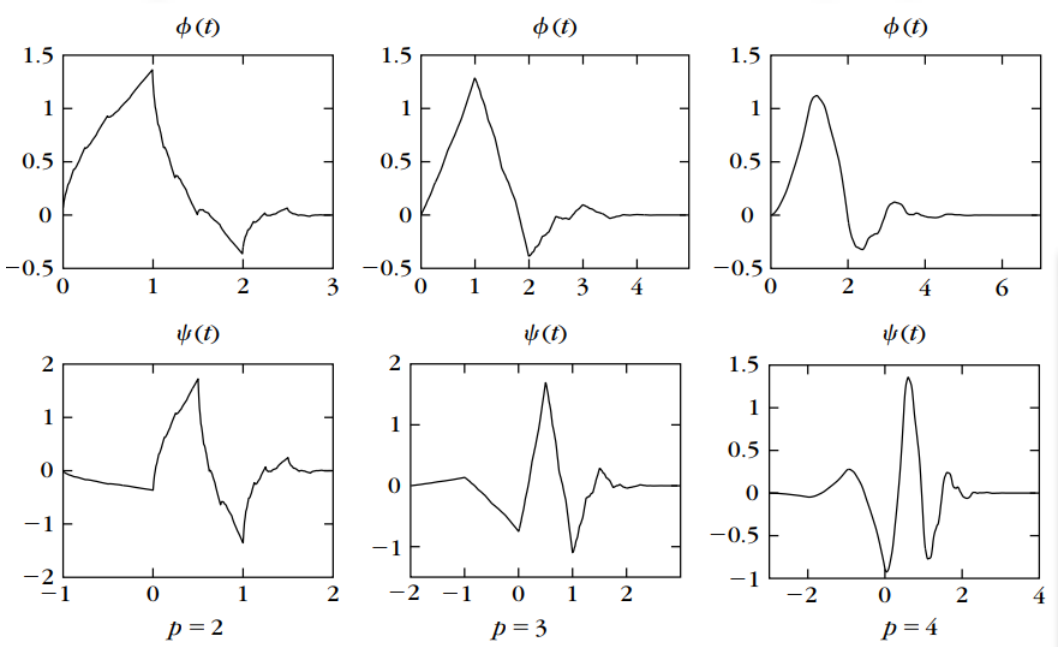



Figure 2.1: Daubechies scaling function $\phi$ and wavelet $\psi$ with $p$ vanishing moments. [4].

### 2.1.2 Why JPEG2000 ?

The technical advantages mentioned previously makes JPEG2000 an ideal compression method for transferring and storing images for object detection as it seems the best compromise between the loss of information on compressed images, crucial for object detection, and the weights of the compressed image files.

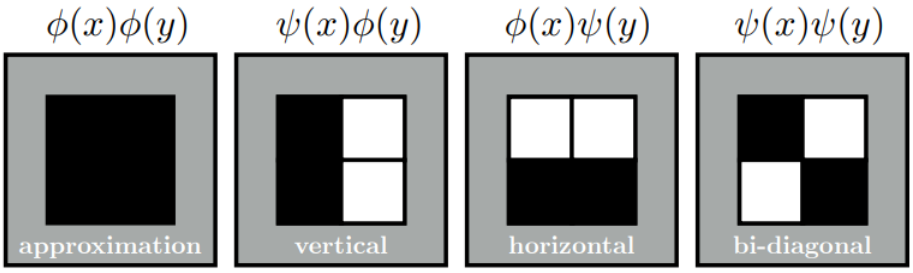



Figure 2.2: Illustration of the 4 wavelets kernels used for the decomposition of an image in approximation and details (from [5]).

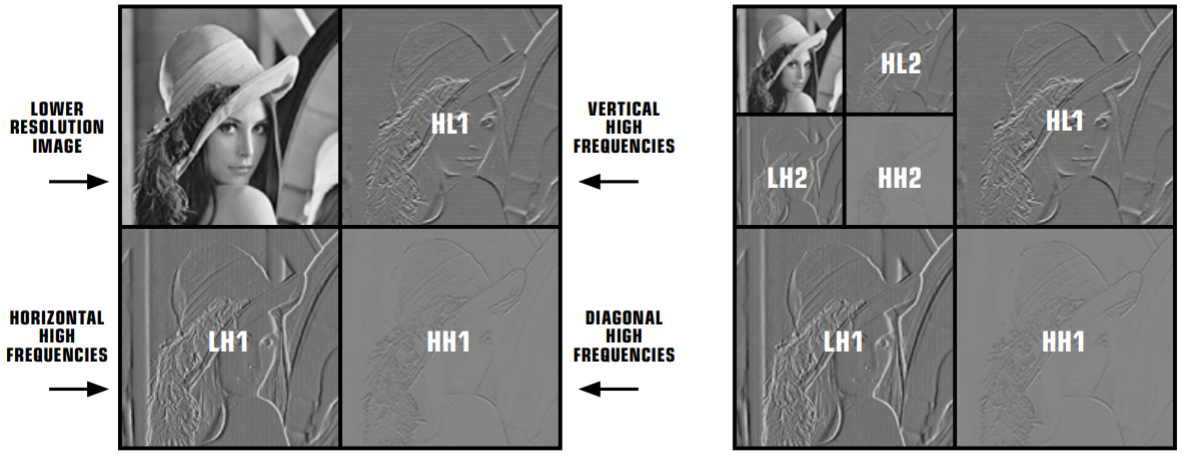



Figure 2.3: 2 successive wavelet decomposition of an image, showing the Laplace pyramid structure [6].

In the following work, the encoded images are not used as is because the implementation of the detection model YOLOv5 used does not support JPEG encoded format, but the advantages of such methods are covered in papers such [7],[8] and [9]. Instead, the images are compressed at the desired ratio and then decompressed (to png files format), using the OpenJPEG implementation[1]. This still leads to an irreversible loss of information contained in the image as desired for further analysis (Example of loss due to compression on Fig. 2.4).

[1] http://www.openjpeg.org/

(a) Original image (b) Image compressed at ratio 96

Figure 2.4: Representation of information loss when on highly compressed image

## 2.2 Object detection & Machine learning

### 2.2.1 Object detection history

Object detection is a computer vision task that involves detecting instances of objects in digital images. Its goal is to provide the localization and the class of objects on the image. The main challenges of such task are the accuracy and the speed of the predictions. It serves as a foundation for various other computer vision tasks, including instance segmentation, image captioning, and object tracking. The advancement of deep learning techniques has significantly contributed to the progress of object detection, making it a prominent and highly researched area. It has become widely used in domain such autonomous driving, robot vision, and video surveillance, as evident from the increasing number of publications in this field [10].

One of the first "traditional" famous object detection method was the *Viola Jones Detectors* published in 2001. It relied on sliding windows and handcrafted feature representations for the detection of human faces. After some new methods and improvements to existing methods, the performance of traditional methods reached their limitations with *Deformable Part-Based Models (DPMs)*, leading to a slowdown in the research of object detection after 2010. However, after the rebirth of Convolutional Neural Network (CNN) in 2012, object detection methods reached a new turning point in 2014 : deep learning-based detection (see the detailed timeline on Fig. 2.7). On one side the single-stage detectors with YOLO

and on the other side two-stage detectors as R-CNN. The latter works first by generating a set of bounding boxes that are likely to contain objects of interest and then, in the second stage, classify the objects based on the extracted features of these bounding boxes. In contrast, one-stage detectors integrate both task into a single step. They offer faster inference and a simpler structure, making them ideal for real-time applications. However, they may experience limitations in their discrimination ability in comparison of two-stage detectors [10]. New state-of-the art object detection techniques such DETR (DEtection TRansformer) use Transformers instead of using mainly CNNs and anchor boxes and can thus achieve a global-scale receptive field.

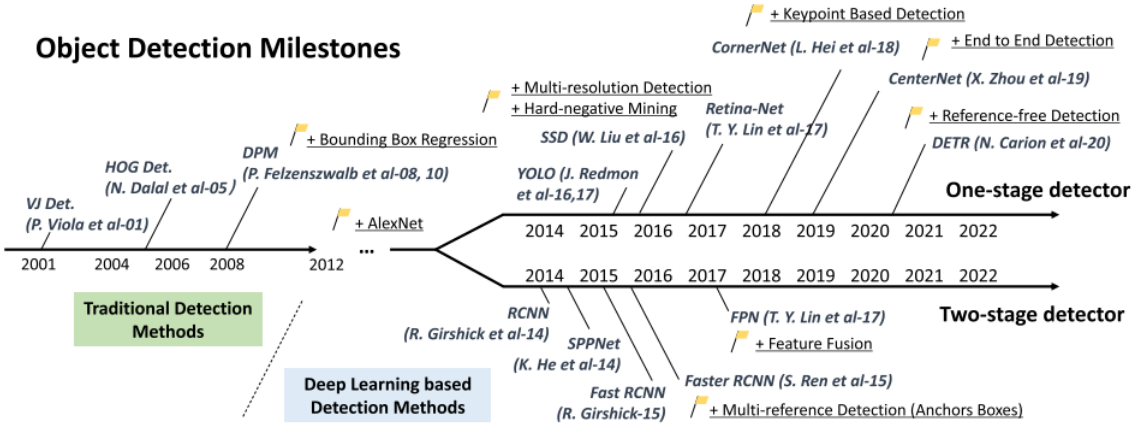



Figure 2.5: Time line of object detection methods [10].

### 2.2.2 YOLO

You Only Look Once (YOLO) was the first one-stage detector to be proposed in 2015 during the deep learning era. Compared to state-of-the-art detectors at this time, YOLO had a tendency to make more localization errors but was less likely to predict false positives on background [11]. It also outperformed all real-time detector in term of mAP by a factor two. The YOLO algorithm demonstrated superior generalization capabilities for detecting objects across image representations with diverse styles compared to other concurrent detectors in 2015 such as DPM and R-CNN. Moreover, it was also faster and all the network is trained jointly.

You Only Look Once works by dividing the images into (S x S) grid cells and predicting a fixed number (B) of bounding boxes based on the center of a supposed object with a confidence score, in each of them simultaneously. A bounding box is defined by a prediction of its center, $\mathbf{x}$ and $\mathbf{y}$, its height and width, $\mathbf{h}$ and $\mathbf{w}$ and a

confidence score **p**. The center is relative to the current grid cell and **w** and **h** are relative to the whole image. For each grid cell is also predicted a set of (C) class probabilities $Pr(Class_i|Object)$, regardless of the number of boxes (Fig. 2.6). The bonding box confidence is then multiplied with the class probability vector of its cell. The final prediction tensor is of size $(S, S, (5B + C))$.

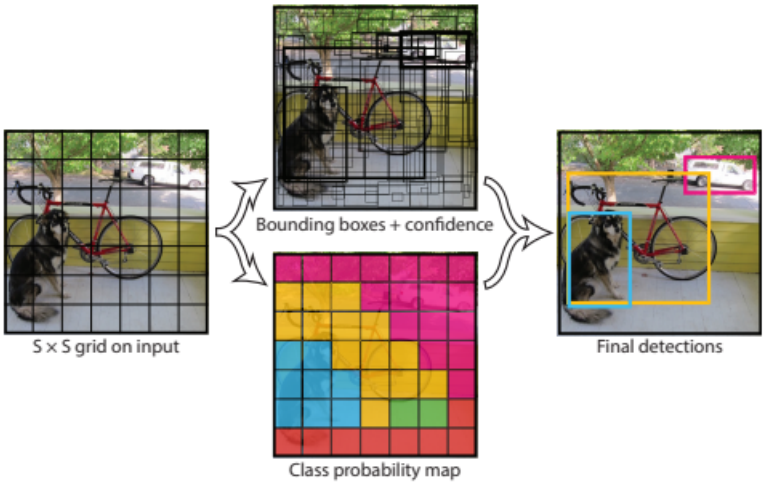



Figure 2.6: YOLO Model workflow : The image is divided in S x S grid cells and B bounding boxes predicts the presence of an object with an associated confidence. Each grid cell also has its set of class probabilities [11].

You Only Look Once (YOLO) is inspired by the image classification model GoogLeNet [11]. The network is composed of twenty-four convolutional layers followed by two fully connected layers. The first convolutional neural layers extract the features of the image. The fully connected layers then output the predicted coordinates and respective confidence of bounding boxes. The network uses the leaky activation function

$$\phi(x) = \begin{cases} x & x > 0 \\ 0.1x & \text{otherwise} \end{cases}$$

except for the final layer where classical linear activation function is used. The model is trained by minimizing the sum-squared error of the output, which consider the size of bounding boxes and reducing the error for boxes containing no objects. Non-maximal suppression is finally applied to remove redundant overlapping boxes.

The main limitations are that the model uses relatively coarse features for predicting bounding boxes since its architecture has multiple downsampling layers from the input image. This impact performances on small objects. Most errors are also due to incorrect localizations.

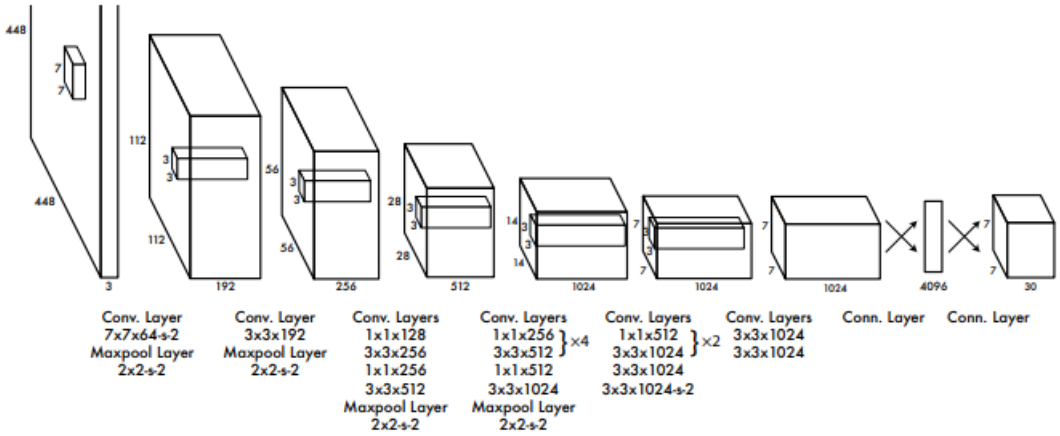



Figure 2.7: Architecture of YOLO. The network is composed of 24 convolutional layers followed by 2 fully connected layers. For this example, S=7, B=2 and C=20 [11].

### 2.2.3 YOLOv5

The main ideas of YOLO has been used to build new versions with multiple improvements and combinations of state-of-the-art techniques [12]. Here a brief overview of the improvements made by each version :

YOLOv2 introduces batch normalization. In addition, YOLOv2 reintroduces anchor boxes utilizing K-means picking instead of fully connected layers, resulting in a fully convolutional architecture that enables the detection of over a thousand bounding boxes in a single image. Fine-Grained Features extraction is made possible by refining the grid by a factor two and enhance the detection of smaller objects. As the network is now fully convolutional, the image size can be adjusted on the fly allowing multi-scale training thus a better generalization. The core of the architecture of YOLO is replaced by the Darknet-19 architecture, a CNN with nineteen convolutional layers [13].

The major improvement of YOLOv3 is the use of Feature Pyramid Network (FPN). FPNs combines features from (three) distinct levels of the CNN to create a pyramid of features. The rest of the network utilizes those features at different scales and resolutions to output a more robust prediction, allowing a multi-scale detection. The backbone of the network, Darknet-19, is replaced by the Darknet-53 architecture, a deeper network achieving better accuracy and performance. Residual-connections are also introduced to address the problem of vanishing gradients by adding parallel connections that skip some convolutional layers [14].

YOLOv4 paper [15] defines a general structure of the network by dividing its

architecture in several parts (referring to vertebrate animals) :

- Backbone : Part responsible for features extraction
- Neck : Pyramidal structure to link backbone to head and usually used to collect feature maps from various stages
- Head : YOLOv3 head

Cross-Stage-Partial-connections is added for improving information flow and features representation between layers and stages. SPP (spatial pyramid pooling) is added over CSPDarknet-53. This makes it possible to generate a fixed-length representation regardless of the size of the image. As a consequence, this leads to a significantly increase in the receptive field, isolating the most significant contextual features while minimally impacting the network's operational speed. This improved version also uses PANet as the method of parameter aggregation from different backbone levels for different detector levels, instead of the previous FPN. The model uses now Mish activation function. The new function to minimize is the CIoU (Complete Intersection over Union) loss as it simultaneously considers the overlapping area, the distance between center points, and the aspect ratio and is thus suitable for the bounding box regression problem of object detection.

YOLOv5 is the PyTorch implementation of the previous version with some minor changes and added modules such for data augmentation. The use of python and PyTorch[2] instead of Darknet makes the code simpler to read and use. In summary, the YOLOv5 network architecture is :

- Backbone: CSP-Darknet53
- Neck: SPPF and CSP-PAN
- Head: YOLOv3 Head

Its implementation comes in various size from tiny to large depending on the complexity of the architecture and thus number of weights. The complete architecture of the large one is represented on Fig. A.1. The version used in this work is the small model with 7.2 million parameters for ease of computation time. It remains, however, very efficient in comparison of other State-of-the-art models (Fig. 2.8).

[2]https://pytorch.org/

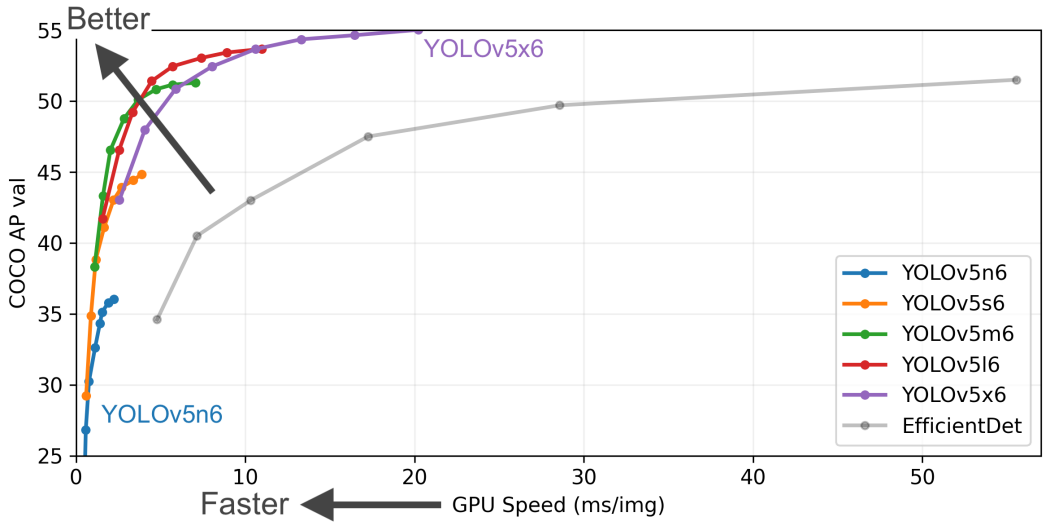



Figure 2.8: COCO mAP on validation set of different size of YOLOv5 (over various inference sizes from 256 to 1536) and EfficientDet [16] (from [17]). Except its nano version, YOLOv5 outperforms EfficientDet in speed while reaching a better accuracy on biggest versions.

### 2.2.4 Performance evaluation

In the following work, the performance of the detection models is evaluated using the mean Average Precision (mAP). This metric is commonly used in the field of computer vision to evaluate the performance of object detection algorithms. It is a single number that summarizes the precision and recall of the algorithm across multiple thresholds, such the Intersection Of Union (IOU, details on Fig. 2.9). The Average Precision (AP) is calculated as the area under the precision(recall) curve.

$$AP = \int_0^1 p(r)dr$$

Precision and recall are two important metrics used in the evaluation of more generic machine learning models, such binary classification problems. Precision (Eq. 2.1) refers to the fraction of true positives among all the predicted positives by the model. It measures how accurately the model identifies the positive cases. Recall, on the other hand, refers to the fraction of true positives among all the actual positives in the dataset. It measures how well the model is able to identify all the relevant cases. In other words, precision is concerned with how often the model is correct when it predicts positive, while recall is concerned with how many

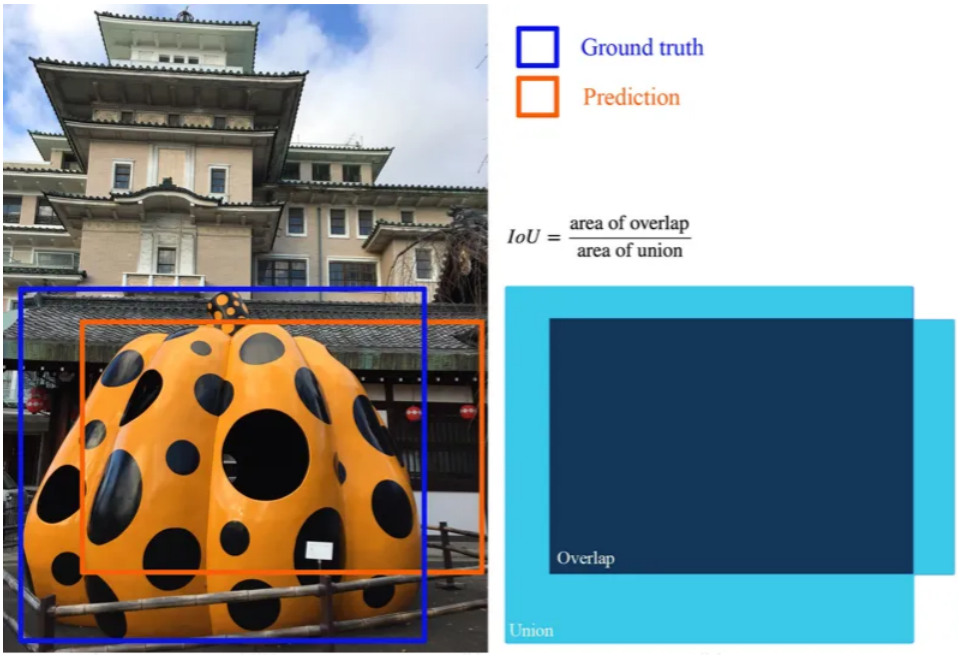



Figure 2.9: Illustration and definition of the Intersection Of Union (IOU) [18].

of the actual positives the model can identify.

$$Precision = P(relevant|retrieved) = \frac{TP}{TP + FP} \tag{2.1}$$
$$Recall = P(retrieved|relevant) = \frac{TP}{TP + FN}$$

The mAP is finally computed by averaging the AP of each class. However, there are times when no distinction is made between AP and mAP when the context is already understood or unambiguous. $mAP_{0.5}$ means that only detection with an IOU$>0.5$ regarding the ground truth box are selected as true positive in the computation of Precision and Recall. $mAP_{0.5:0.95}$ is the average of mAP with multiple IOU threshold from 0.5 to 0.95, usually ten different values.

In general, a good model should have both high precision and high recall. However, there is often a trade-off between the two, where increasing precision may result in lower recall and vice versa. Therefore, it is important to consider both precision and recall together, using in this case mAP to compare the performance of different object detection models.

## 2.3 Ensemble methods

The goal of ensemble methods is to improve the predictive performance of a single model by training various models and combining their predictions. The main idea is that by combining multiples models, the error caused by one model would be compensated by other models and the overall prediction performance will increase [19]. Ensemble methods also try to mitigate some common machine learning challenges such class imbalance (by using models trained on balanced sub-sample of a dataset), concept drift (some objects change over time) or curse of dimensionality. The main drawbacks of this type of method are a longer time of prediction due to the aggregation of multiple predictions into one and an increase in computational cost due to the training of multiples models in first place.

The key principles to generate a good ensemble are the diversity and the predictive performance of the models. These must be sufficiently diverse to obtain predictions for an object from some models that are not detected by others to gain a benefit from assembling. Models also need to predict individually as good as possible, at least better than random guess detection. The output prediction is then merged by a defined method. The most common are weighted methods : a weight is assigned to each model and the final output is chosen according to a strategy, like computing a weighted average of the predicted probabilities or scores. Another type of methods are meta-learning methods. They involve the addition of a dedicated phase that learns to merge the outputs of individual models (commonly known as Stacking). Some well-known ensemble methods are ([19]) :

- Bagging (Bootstrap Aggregating): Bagging involves creating multiple bootstrap samples of the training data and training a separate model on each sample. The final prediction is made by averaging the predictions of all the models.
- AdaBoost (Adaptive Boosting): AdaBoost is a boosting algorithm that assigns weights to the training samples based on their classification accuracy. In each iteration, it places more weight on the misclassified samples to improve their classification in the next iteration.
- Random Forest: Random forest is an extension of bagging that involves creating multiple decision trees on random subsets of the features and combining their predictions using majority voting.

The obtained ensemble can often be simplified by ensemble pruning in order to reduce the number of models while maintaining or improving its performance. The goal of ensemble pruning is to identify and remove the redundant or less useful

models from the ensemble. Two popular techniques are ranking methods and search-based methods. The first select the top ranked base models according to a predefined threshold while the other evaluate the different subsets of the ensemble and keep the best one.

### 2.3.1 Application to Object Detection

Ensemble methods can be applied to the particular case of object detection algorithms : the predicted features are then boxes and their corresponding classes. They can be based on the architecture of the model (as for the two-stage detector) or based on the output of these detectors. This second option will be explored in the following experiments. A simple approach to combine the output of detection models is the application of techniques to eliminate redundant bounding boxes like Non-Maximum Suppression, Soft-NMS, weighted box fusion [20], ... However, these techniques do not consider the classes of the detected objects, or the number of models that detected a particular object. This can lead to the detection of many false positives. The research paper [21] suggest an ensemble method based on voting strategy in order to select which boxes to select or not. The three types of strategy are :

- Affirmative : The affirmative strategy consists of validating all the detection done by the base models.
- Consensus : The consensus strategy validates the boxes detected by at least a (user defined) part of the base models. If this part is the half of the set of models, this is equivalent to a majority voting strategy.
- Unanimous : The unanimous strategy only validates regions detected by all the base models.

They have different effects and results depending on the performances and diversity of the base models. The affirmative strategy decreases the number of false negatives but increases the likelihood of incorrect detections (false positives). Conversely, the unanimous strategy results in the fewest false positives but increases the incidence of false negatives. The consensus strategy should offer a reasonable compromise between false positives and false negatives, based on the specific performance requirements of the application.

### 2.3.2 Test-time augmentation

Data augmentation is commonly used during training time to increase the size and the diversity of a training dataset by applying various transformations to data.

Test-time augmentation (TTA) is a variant of data augmentation applied during test time [21]. It consists of applying various transformations to the test image, resulting in a diverse set of variations. Each element in this set is then inferred, and finally, an ensemble method is applied to combine the individual prediction outputs, yielding the final prediction for the image (see Fig. 2.10). This operation is then repeated for each image in the test set.

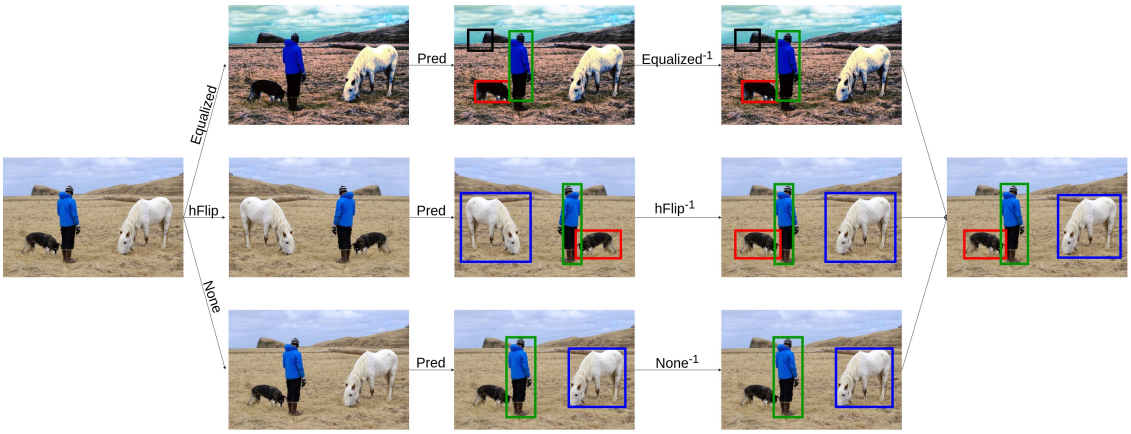



Figure 2.10: Test-time augmentation workflow. Multiple transformations are applied and then inferred. BBoxes are then "reversed" and finally ensembled [21].

## 2.4 Related works

*First gradually, then suddenly: understanding the impact of image compression on object detection using deep learning* [22] studies the influence of image compression on object detection using deep learning. Some methods suggested in the conclusion of this research paper to overcome compression influence includes

- Pre-detection quality improvement using super-resolution.
- Training models on compression-degraded training sets.
- Building dedicated models for specific ranges of compression quality to be used as ensemble or dedicated models for small and non-small objects.

All these methods are (partially) covered in the following chapters.

*Swin-Transformer-Enabled YOLOv5 with Attention Mechanism for Small Object Detection on Satellite Images* [23] focus on the detection of small objects from satellite image. The architecture of the network is adapted and the regular CNN prediction heads of YOLO is replaced by a Swin transformer Prediction Heads

(SPHs). This new version of YOLO achieves a mAP 0.071 higher than with original YOLOv5.

*Object Detection performance variation on compressed satellite image datasets with iquaflow* [24] (January 2023) implements a python package named "iquaflow" designed to study image quality and model performance variation given an alteration of the image dataset. They use DOTA dataset, YOLOv5 and JPEG compression.

*Evaluation and optimization of image compression for Convolutional Neural Network in segmentation and classification* [25] is a master thesis from Ecole Polytechnique de Louvain (2020) that aims to analyze and evaluate different techniques in order to reach higher compression ratios for equivalent segmentation and classification results. Those techniques include the training of CNNs (2/3D U-Net) at several compression ratios similar to the methodology used in Chapter 3 (and later published in [26]). It also includes optimization of the JPEG quantization tables for some rates that improved the performance of classification by 3%.

# Chapter 3

# Training optimization of the detection network

## 3.1 Dataset selection and pre-processing

This work focuses on object detection from compressed images, with a particular interest in aerial image detection. These images are part of the DOTA dataset [27], consisting of both RGB and grey-scale images. The formers are collected from Google Earth, while the latter are obtained from the GF-2 and JL-1 satellites provided by the China Centre for Resources Satellite Data and Application. Specifically, version 1.5 of the dataset is used, which includes annotations for sixteen different object classes : 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. Each object is annotated by an Oriented Bounding Box (OBB), consisting of the coordinates of its four vertices. For simplicity, these boxes are later converted in Horizontal Bounding Box (HBB).

The dataset was originally divided into a training set and a test set, which were merged and then randomly split into new training, validation, and test sets. This represents proportions of approximately 0.7, 0.2, and 0.1, respectively. The distribution of the number of instances of each object represented in the training set is not equal as shown in Fig. 3.1. The representation of certain objects, specifically small vehicles, is significantly more pronounced relative to other objects like roundabouts. This kind of data imbalance can lead to poor or biased learning of underrepresented classes. Since the mitigation of this phenomenon is not the purpose of this work, the focus will be placed on the most represented classes which include planes, small vehicles and ships.

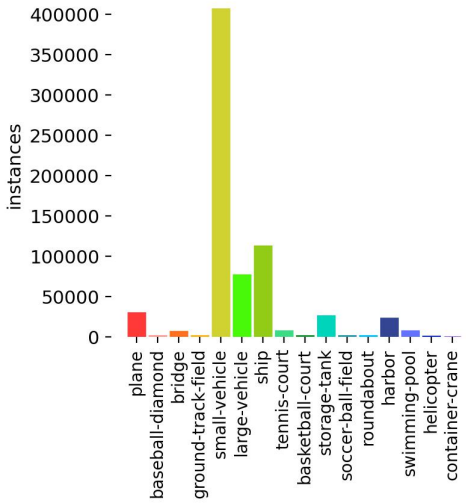



Figure 3.1: Label distribution of the training set (and the dataset)

## 3.2 Baseline experiments

The first step is to establish a baseline model : the simplest case giving a base for good comparison with further enhanced models.

### 3.2.1 First experiment

As an initial attempt to create a baseline model, a pre-trained model on the COCO dataset, later called "blank" model for simplicity, is trained for 300 epochs using the new training set. YOLOv5 uses combination of Mixup and Mosaic as data augmentation techniques to expand the training set during learning. This took about 35 hours on personal computer (i5, RTX 3060 and SSD NVME). The performance of this model is then evaluated using the mAP as described in 2.2.4, by inferring the test set compressed at different compression ratios : 1:1, 8:1, 16:1, 24:1, 32:1, 48:1, 64:1 and 96:1. Those results are compiled in one plot represented in Fig. 3.2. The average of the mAP of all objects (including low-represented ones, which are indicated by the black curve) is low in comparison to the other state-of-the-art detection models on this dataset. Indeed, the top ten average $mAP_{50}$ from the official detection task of HBB for the DOTA dataset range between 0.7419 and 0.7953 while the one for this model is lower than 0.6. This metric also degrades very quickly as the test set is increasingly compressed. One could remark that the planes are less impacted by the increasing compression ratio. On the other hand, the detection of cars is more affected. This can be interpreted as being related to the difference in size and shape of the objects [22]. Cars are only represented

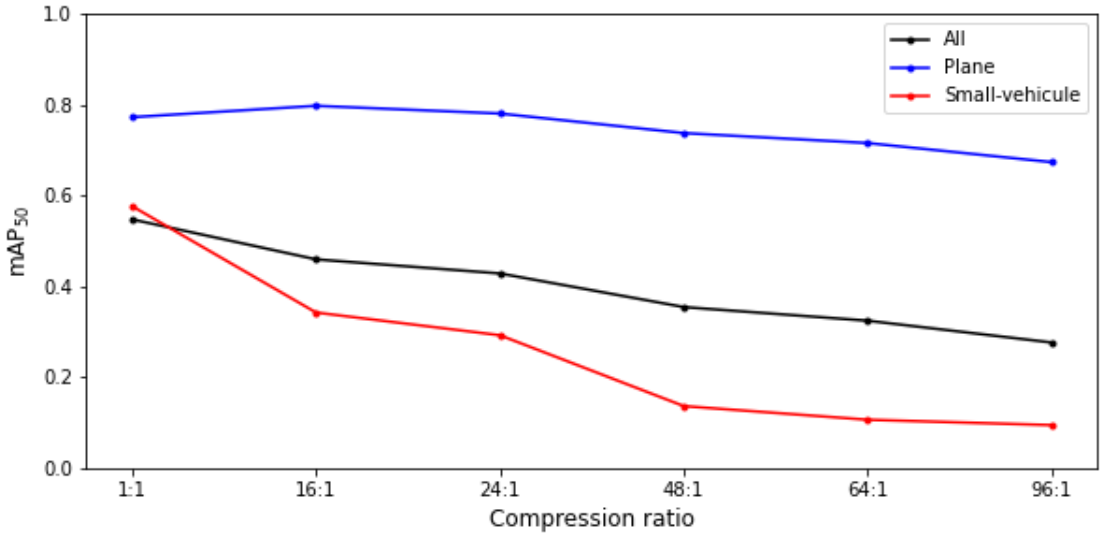
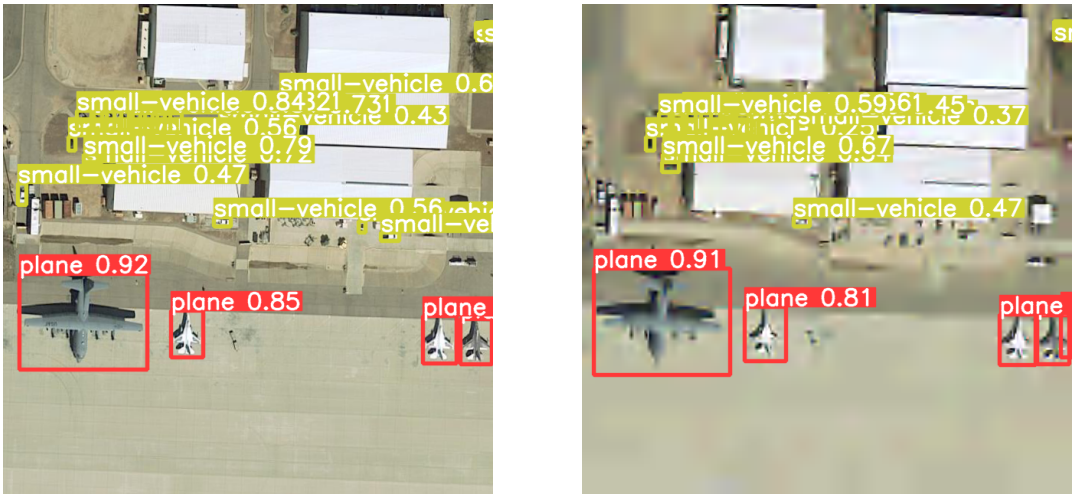



Figure 3.2: $mAP_{50}$ of the first try of base model.

by rectangular shapes with few pixels and tend to disband with the background (see Fig. 2.4) when the image is compressed while planes have sharper forms and are multiple times bigger than a car. The planes are mostly white and therefore have a better contrast with the background which could explain better detection performances. The difference of degradation between these two objects on the same image/scale is visible on Fig. 3.3.

### 3.2.2 Second dataset pre-processing

One reason why those results are not as good as expected is that the handling of images of different resolutions that is done by YOLOv5 stretches images to match a fixed resolution specific to its architecture. To remove any uncontrolled bias due to these deformations, each image is split in tiles with equal defined resolution corresponding to the one specified to the architecture. More specifically, they are split in rectangular tiles of 640 pixels by 640 pixels and padded with black pixels. This seems a vialbe choice as most of the objects can fit in a rectangular box of a dozen pixels, especially the selected ones (most represented). For the training set, a half-size overlap of the tiles is added in both vertical and horizontal direction to ensure learning objects that would be cut on one tile. For instance, a 1147 by 1023 pixels image in the training set is divided into nine tiles (each measuring 640 by 640 pixels) with black pixels added as padding (see Fig. 3.4). This resulted in a significant increase in the number of images in the training set, from 1318 to 75859. The validation set and test set are also divided but without overlap to avoid double




(a) Original image (b) Image compressed at ratio 64

Figure 3.3: On the second image, the planes are still recognizable while the cars present in the middle and upper-left of the image are mixed with the background and sometimes not detected.

detection and skew the results.

### 3.2.3 Baseline model

A blank model is trained using those new sets, this time during eighty epochs for computational and time reasons as the total number of pixels has more than doubled and there are over fifty times more images. This train took about 16 hours. The results, plotted in Fig. 3.5, are more consistent with the literature with an average $mAP_{50}$ over all classes of 0.77 (on uncompressed test set and with the small version of YOLOv5). This new model outperforms the previous one regardless of the compression ratio, as shown by the difference between the blue curve and the dashed black one. One of the causes must be that the shapes of the objects are no more altered by the stretching as it is the case in the previous experiment. Roundabouts are however poorly detected because of its under-representation as well as some labelling errors in the dataset. This is again a reason for which only most represented classes will be compared in the next experiments. This model can therefore be deemed satisfactory as a baseline.

(a) Original image (b) Split image with overlap

Figure 3.4: Illustration of the division in tiles of an image from the training set.

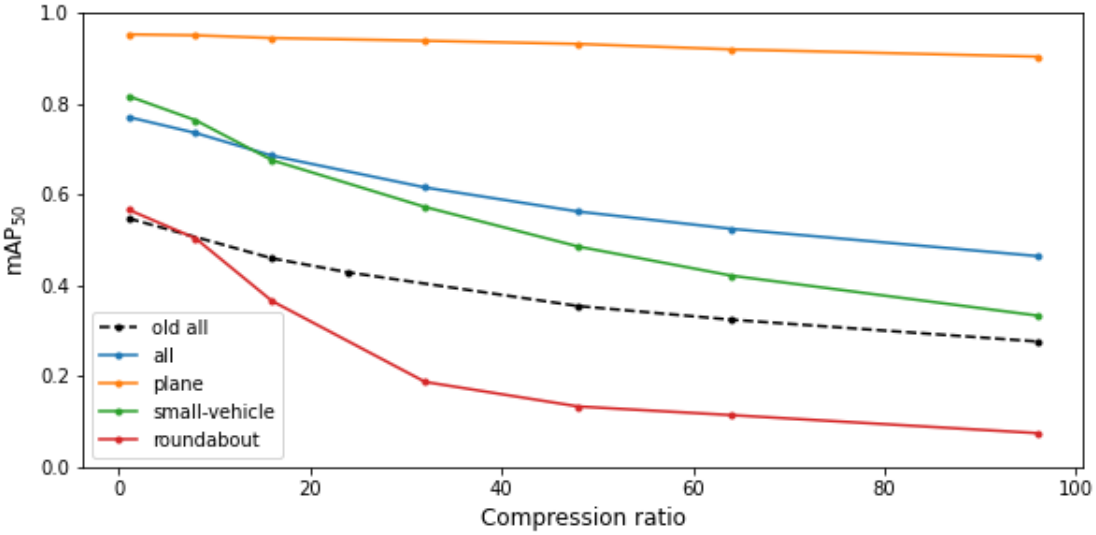



Figure 3.5: $mAP_{50}$ of the baseline model showing a gap in performances in regards of the previous one in dashed black.

## 3.3 Fine-tuning

Following the method from [26], the first step is to train the baseline model on compressed versions of the training set. Thanks to transfer learning, a better robustness of the detection can be expected when the images are increasingly compressed. The compression/decompression of the training set at a specific compression ratio takes about 20 hours. Then the previous (baseline) model is trained during twenty epochs on one of this compressed set, in this case with

compression ratios 16, 32 or 64 (each model to a specific ratio). The three different models are then evaluated. The results are summarized on Fig. 3.6 and Fig. 3.7. Table 3.1 compares the size of test set with corresponding performances of fine-tuned model. Table 3.2 gathers the $mAP_{50}$ of each model and for each object of interest from compressed test sets.

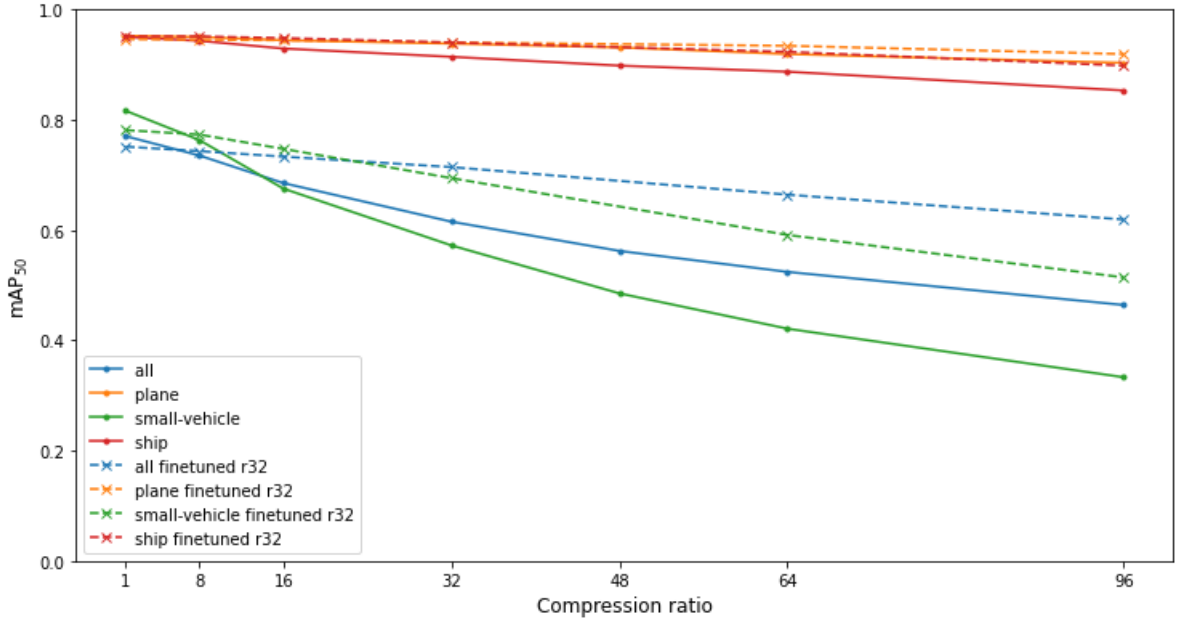



Figure 3.6: Comparison of the $mAP_{50}$ of the baseline model and fine-tuned one at ratio 32. Except on uncompressed images, the fine-tuned model outperforms the base model.

The fine-tuned model shows, as expected, way better detection performances on compressed images than the baseline model. The gap is even greater for objects whose detection performance collapsed with increasing compression of the test images, as small vehicles. Fig.3.7 shows the difference of performance between models fine-tuned using various compression ratio. A slight increase in performance at high compression ratios is observed when utilizing a model that has been fine-tuned with more highly compressed images. The advantage of this method is that there is no need to retrain a model completely from scratch. Consequently, a model can be adapted to a specific compressed ratio within a short computation time.

| comp. r. | comp. size | uncomp. size | $mAP_{50}$ all | $mAP_{50}$ car | $mAP_{50}$ plane |
|---|---|---|---|---|---|
| 1 | 862 | 961 | 0.751 | 0.781 | 0.946 |
| 8 | 214 | 922 | 0.743 | 0.773 | 0.945 |
| 16 | 112 | 842 | 0.733 | 0.747 | 0.944 |
| 32 | 57.8 | 735 | 0.714 | 0.694 | 0.94 |
| 48 | 38.8 | 666 | 0.689 | 0.64 | 0.938 |
| 64 | 29.1 | 616 | 0.664 | 0.591 | 0.934 |
| 96 | 19.5 | 544 | 0.619 | 0.514 | 0.919 |

Table 3.1: Compressed (.j2k) and uncompressed (.png) size in MB of test set at several compression ratios with their corresponding detection performance by the fine-tuned model at comp. ratio 32. At ratio 96, the set size is 2% of the original one while the detection of planes performance decreased by only 0.027. However, the detection performance of the cars decreased by 0.267. Note that the detection model requires decompression before inference.

| object | model | r1 | r8 | r16 | r32 | r48 | r64 | r96 |
|---|---|---|---|---|---|---|---|---|
| all | *baseline* | **0.77** | 0.73 | 0.69 | 0.61 | 0.56 | 0.52 | 0.46 |
| | *f.-t. r16* | 0.76 | **0.75** | **0.73** | **0.71** | 0.68 | 0.65 | 0.59 |
| | *f.-t. r32* | 0.75 | 0.74 | **0.73** | **0.71** | **0.69** | 0.66 | 0.62 |
| | *f.-t. r64* | 0.72 | 0.72 | 0.72 | 0.70 | **0.69** | **0.67** | **0.64** |
| plane | *baseline* | **0.95** | 0.95 | 0.94 | 0.94 | 0.93 | 0.92 | 0.90 |
| | *f.-t. r16* | **0.95** | **0.95** | **0.95** | **0.94** | **0.94** | 0.93 | 0.91 |
| | *f.-t. r32* | **0.95** | 0.94 | 0.94 | **0.94** | **0.94** | 0.93 | 0.92 |
| | *f.-t. r64* | **0.95** | **0.95** | 0.94 | **0.94** | **0.94** | **0.94** | **0.93** |
| cars | *baseline* | **0.82** | 0.76 | 0.68 | 0.57 | 0.48 | 0.42 | 0.33 |
| | *f.-t. r16* | 0.80 | **0.79** | **0.75** | 0.68 | 0.62 | 0.56 | 0.48 |
| | *f.-t. r32* | 0.78 | 0.77 | **0.75** | **0.69** | **0.64** | 0.59 | 0.51 |
| | *f.-t. r64* | 0.75 | 0.74 | 0.72 | 0.68 | 0.63 | **0.60** | **0.54** |
| ship | *baseline* | **0.95** | 0.94 | 0.93 | 0.91 | 0.90 | 0.89 | 0.85 |
| | *f.-t. r16* | **0.95** | **0.95** | **0.95** | **0.94** | 0.92 | 0.91 | 0.88 |
| | *f.-t. r32* | **0.95** | **0.95** | **0.95** | **0.94** | **0.93** | 0.92 | **0.90** |
| | *f.-t. r64* | **0.95** | 0.94 | 0.94 | **0.94** | **0.93** | **0.93** | **0.90** |

Table 3.2: Table of $mAP_{50}$ of different models for different object/compression. In bold type are the best score at a specific ratio/object and among the models. Models performs best at their corresponding ratio and those close.

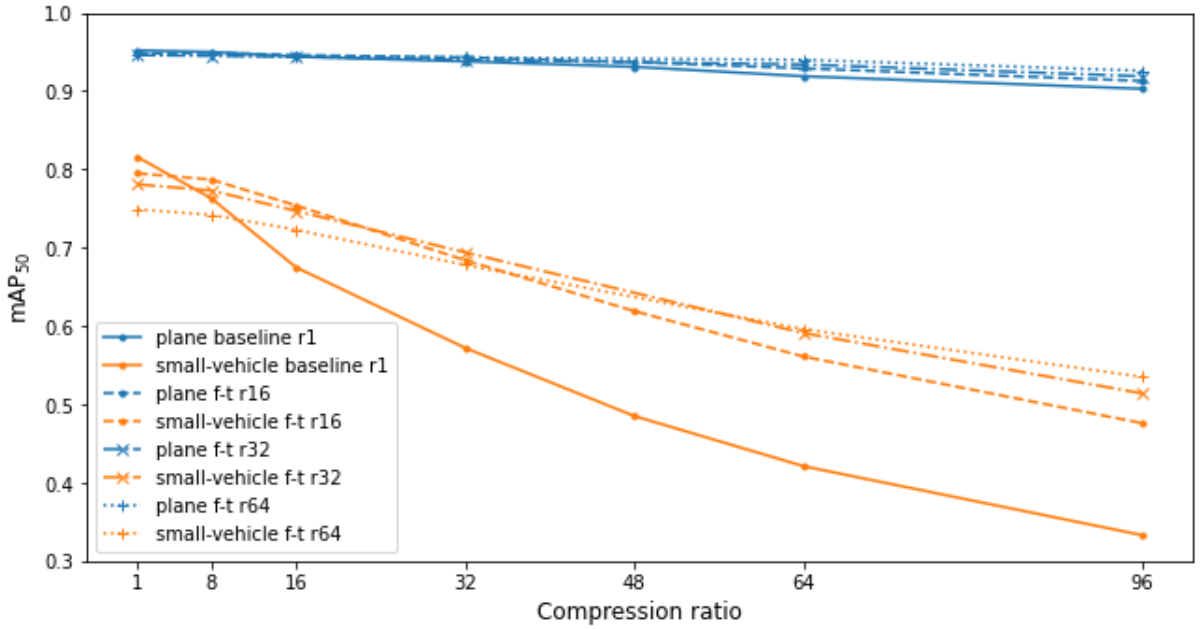



Figure 3.7: Comparison of the $mAP_{50}$ of fine-tuned models at different ratio. The differences are slight and the models differ the most on low or highly compressed images.

## 3.4 Fully compressed training

Another solution is to directly train a model on a compressed version of the training set starting from "blank" weights (pre-trained COCO weights). As for the baseline model, this new model is trained for eighty epochs, this time using a compressed version of the train set at specific compression ratios. Different models were thus train at ratio 16, 32 and 64. This took about 16 hours per model. The results and comparison of these models are shown in Fig. 3.8 and Fig. 3.9.

The model's performance detection is comparable to those of fine-tuned models over the more compressed test set (Fig. 3.10). However, their performance is comparatively poorer when dealing with non-compressed or sparsely compressed test sets.

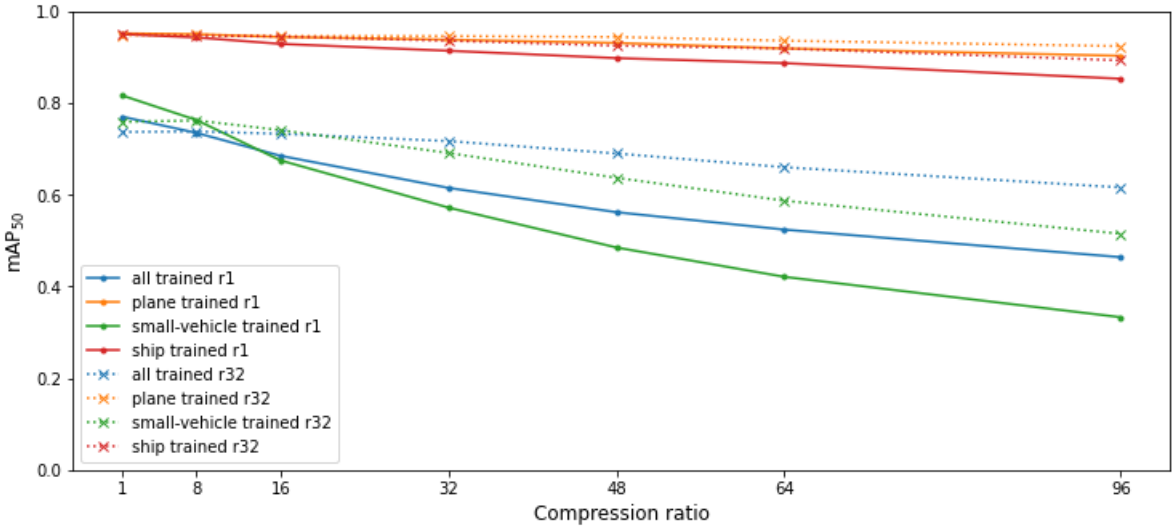



Figure 3.8: Comparison of the $mAP_{50}$ of the baseline model with trained one at ratio 32.

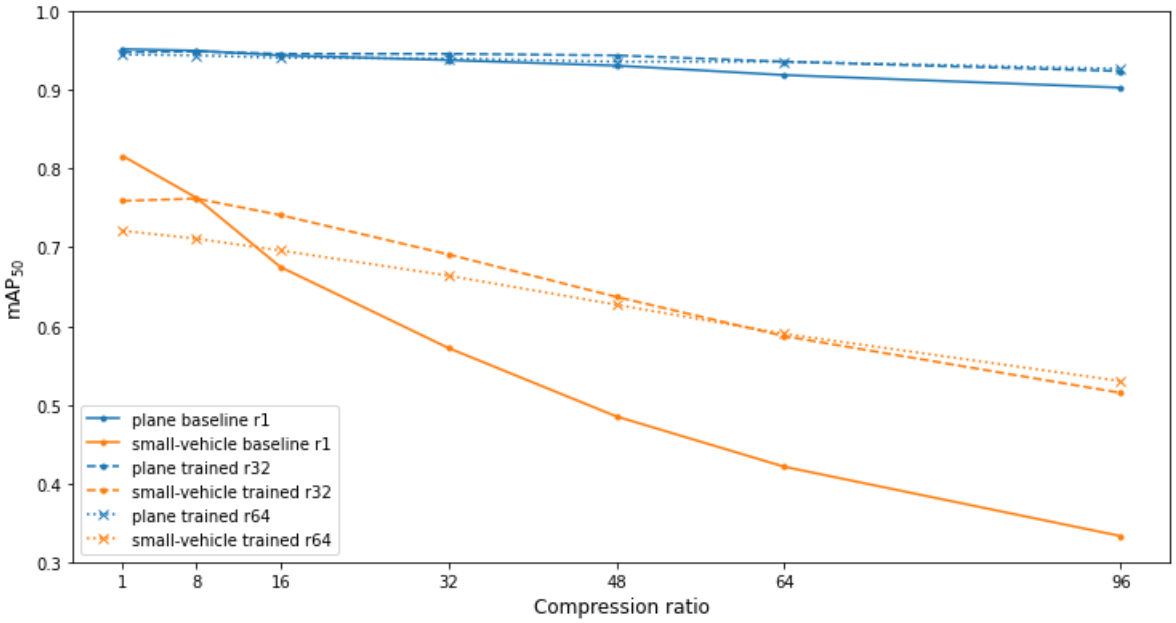



Figure 3.9: Comparison of the $mAP_{50}$ of the baseline model and the fully trained one at ratio 32 and 64.

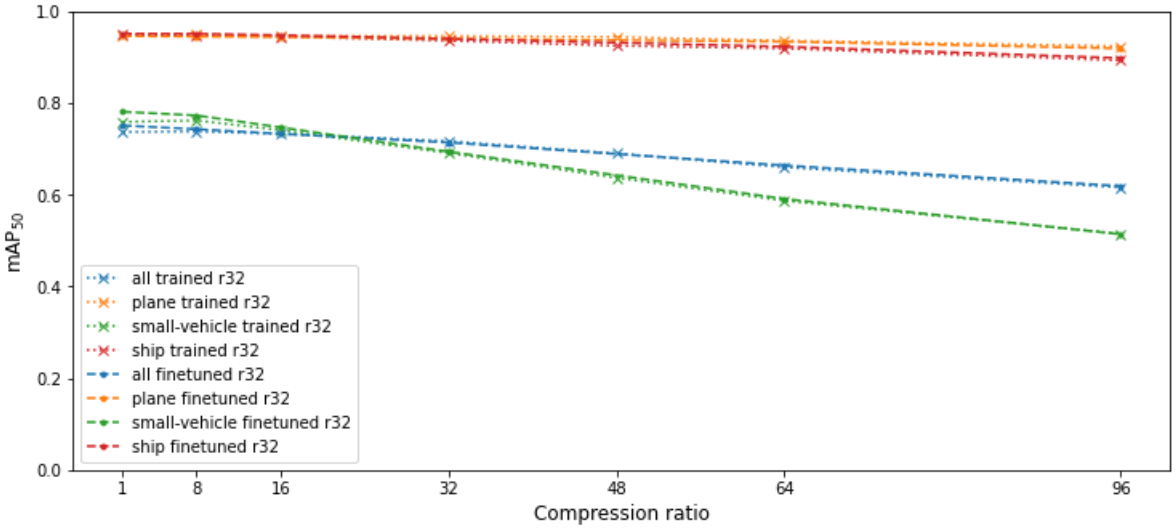



Figure 3.10: Comparison of the $mAP_{50}$ of the fine-tuned model and the fully trained one at compression ratio 32. Main difference is that fine-tuned models performs slightly better on uncompressed images.

## 3.5 Conclusions

The goal of this chapter was first to define a baseline model and then maximize its robustness of detection to the compression of images by playing with the compression of the training set.

In the first experiment trying to establish a baseline model, the utility of splitting (mega-)image in tiles, insuring similar spacial resolution and adapted shape to the detection model training was proven, leading to improved detections performances.

The next experiments showed that fine-tuned models performed as well as fully trained models on highly compressed images and better on slightly compressed one. Moreover, the advantage of fine-tuning is the gain in calculation time.

In the following chapters, the selected models will not be further trained as next methods only apply during test time and with the output predictions.

# Chapter 4

# Ensemble methods

The following chapter will cover experiments on application and optimization of ensemble methods, in particular Test-time augmentation (TTA) and models ensembling methods.

## 4.1 Implementation

The implementation of YOLOv5 [17] comes with modules for Test-time augmentation and ensemble. However, these techniques are not the one at the state-of-the-art presented in Chapter 2.

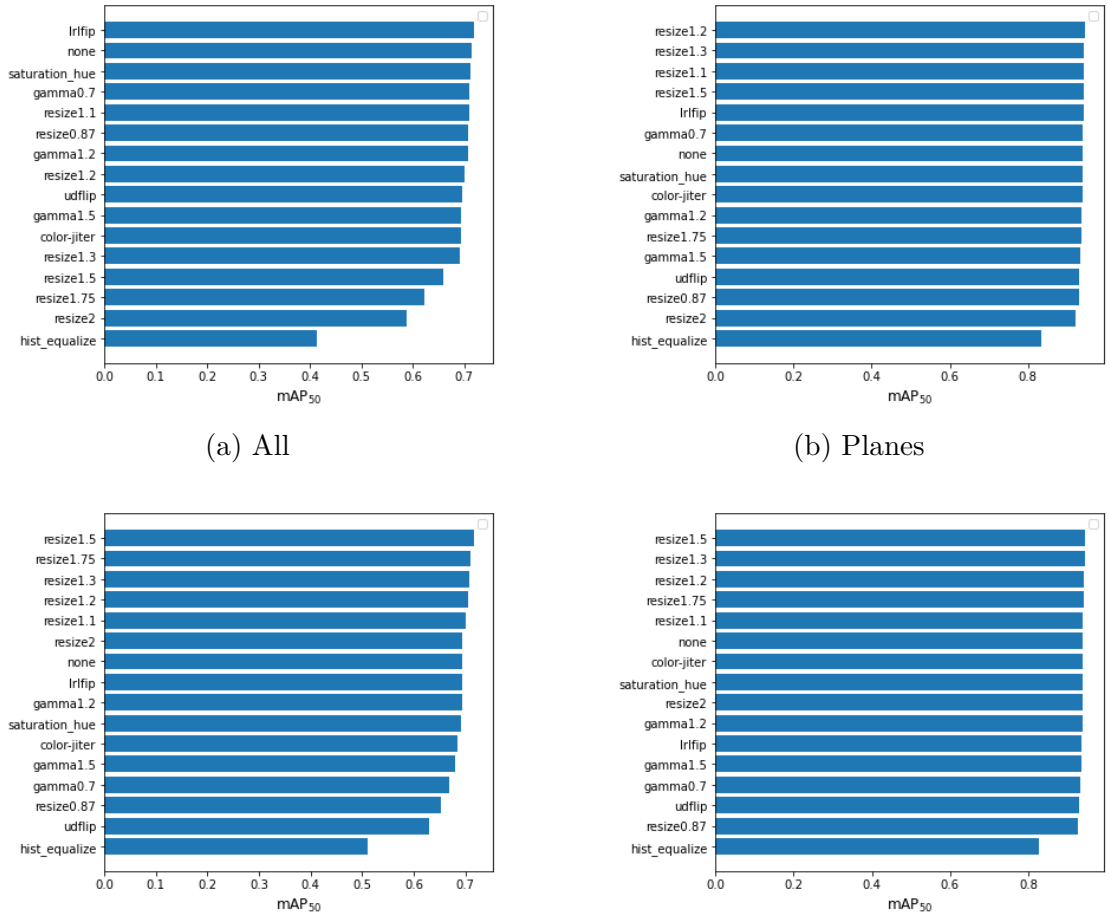
The TTA implementation only covers image resizing and flipping and then all the detections are merged using Non-Maximum Suppression, corresponding to the affirmative strategy. Modifications to the original code has been done to efficiently support other augmentations like histogram equalization, color shifting and gamma correction.

The model ensembling method proposed in the original implementation supported three options : merging all detected bounding boxes by Non-Maximum Suppression, max ensemble and mean ensemble. The first one corresponds to the affirmative strategy while the other does not match any strategy from state-of-the-art ensembling presented in Chapter 2. Therefore, it has been implemented and (slightly) optimized to achieve a respectable inference speed in the order of magnitude of single model inference time. The Intersection Of Union of the Non-Maximum Suppression was set to 0.6 to align with the parameters used in [21]. Additionally, the Intersection Of Union threshold, which determines if two bounding boxes from two different models should represent the same object, has also been set to 0.6.

## 4.2 Experiments

### 4.2.1 Optimal Test-time augmentation

Multiples evaluations are done with different augmentation methods, such as left-right flip, resize, gamma correction, in order to identify the most effective TTA for maximizing the model's accuracy and generalization ability. The selected model used for these inferences is the fine-tuned one at compression ratio 32. The evaluations are done on the test set compressed at the same ratio. Fig. 4.1 shows the performance ($mAP_{50}$) of each augmentation relatively to "none" which means no augmentation.




(a) All

(b) Planes

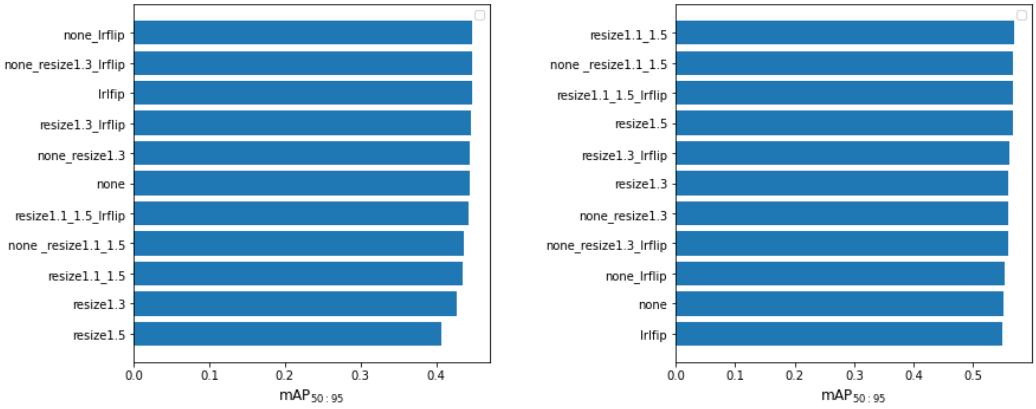
(c) Small vehicles

(d) Ships

Figure 4.1: Comparison of $mAP_{50}$ when applying various TTA individually for various objects with model fine-tuned at compression ratio 32 on compressed test set at ratio 32. Resize and L-R flip augmentations lead to the best results.

The resize augmentation, which means that the test image is resized by a given factor using bilinear interpolation, improves the accuracy of the model. In particular, resize at factor 1.5 gives the best results over the most represented object. Left-Right flip improves the $mAP_{50}$ of the average of all objects.

These augmentations can be ensembled to take advantage of each of their particularities. The effectiveness of a combination of the best-performing techniques, involving resizing at various factors, LR flipping, and no augmentation, is evaluated using an affirmative voting strategy to determine the most suitable approach for the compressed test set and compared using the $mAP_{50:95}$ (mean of the mAP at different IOU thresholds between 50 and 95) on Fig. 4.2).




(a) Average on all 16 classes (b) Mean of 3 most represented classes

Figure 4.2: Comparison of ensemble of TTA techniques using on left average of $mAP_{50:95}$ of all 16 classes, and on right the mean of the $mAP_{50:95}$ of the 3 most represented objects.

The ensemble of Test-time augmentations "resize1.1_1.5", composed of the up-sampling of the image at factor 1.1 and factor 1.5, performs the best in term of $mAP_{50:95}$ for the detection of the three objects of interest. However, this ensemble deceases the performances on average for all classes. This TTA ensemble is then evaluated on test set compressed at each ratio and compared with the same model without TTA on Fig. 4.3.

The mean Average Precision is sensibly increased, except for the average of all sixteen classes without surprise, up to image at compression ratio 48. On highly compressed images, the performance becomes equal to that of the non-augmented model. For example, the $mAP_{50}$ of small vehicles is increased by 0.03 on test sets compressed at ratio 8 and 16.

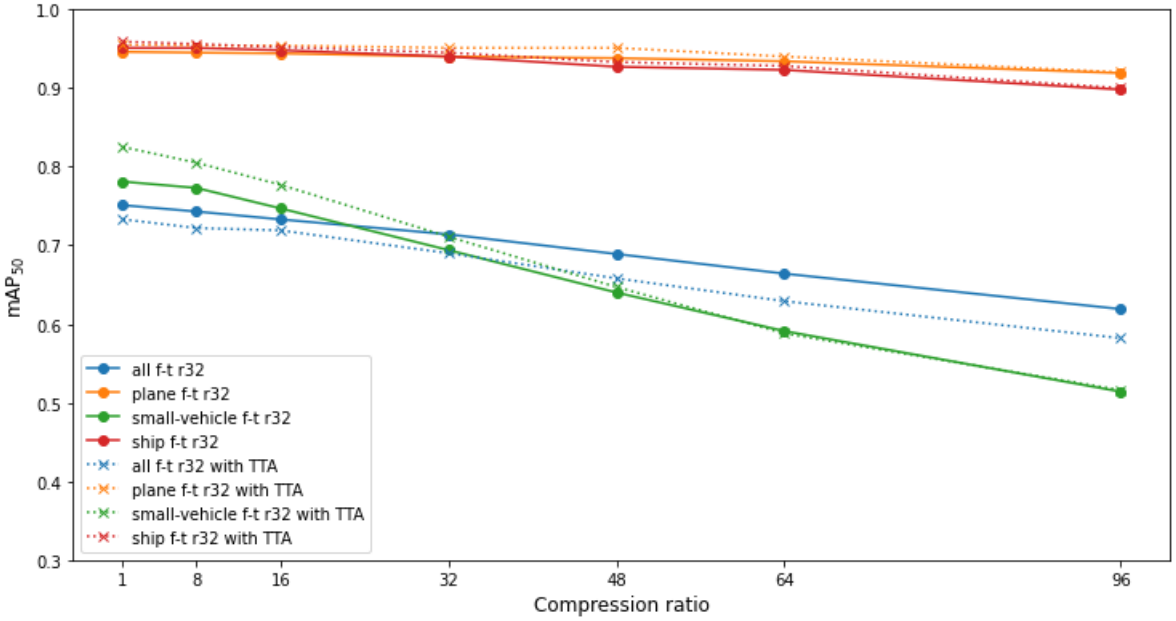



Figure 4.3: Comparison of the $mAP_{50}$ of the fine-tuned model at compression ratio 32 with and without TTA ensemble (resize 1.1 & 1.5). TTA is effective on selected objects, in particular on small vehicles, until compression ratio 48.

### 4.2.2 Model ensemble

The goal of this section is to find the best performing subset among the seven trained models : the baseline model, two models fully trained (80 epochs) on compressed dataset at ratio 32 and 64, three models respectively fine-tuned (20 epochs) at ratio 16, 32 and 64, and a model fine-tuned only for small-vehicles at ratio 32.

Firstly, the ensemble of the seven models is evaluated using the tree different ensembling strategies. Then two models are discarded to form a subset of five models. The selection is made to preserve good performances at every compression ratio while maximizing the diversity of models. Thus, the discarded models are the fully trained one at ratio 64 and the fine-tuned one at ratio 16. Finally, forming a subset of three models is a more complex task as there are many possibilities. Fully trained model at ratio 32 and fine-tuned model at ratio 16 are discarded as a first try. The results are shown on Fig. 4.1. These ensembles are evaluated on compressed sets, at compression ratio 32 and 64 (15 and 22 times lighter than lossless compressed originals).

The consensus strategy works the best for ensembles composed of more than three models. The full ensemble performs the best on all classes while keeping

| | Test set r32 | | | | Test set r64 | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | all | plane | car | ship | all | plane | car | ship | FPS |
| fine-tuned r32 | 0.714 | 0.94 | 0.694 | 0.94 | 0.664 | 0.934 | 0.591 | 0.923 | 115 |
| fine-tuned r64 | 0.702 | 0.943 | 0.678 | 0.939 | 0.669 | **0.94** | 0.596 | 0.925 | 115 |
| 7 models aff. | 0.723 | 0.946 | 0.691 | 0.941 | 0.679 | 0.939 | 0.596 | **0.93** | 3.75 |
| 5 models aff. | 0.72 | 0.946 | 0.692 | 0.941 | 0.673 | 0.938 | 0.596 | 0.925 | 6.66 |
| 3 models aff. | 0.712 | 0.943 | 0.694 | 0.94 | 0.66 | 0.934 | 0.592 | 0.919 | 16.5 |
| 7 models cons. | **0.726** | 0.947 | 0.696 | **0.944** | **0.681** | 0.939 | 0.602 | **0.93** | 4.08 |
| 5 models cons. | 0.721 | **0.948** | **0.698** | **0.944** | 0.676 | **0.94** | **0.603** | 0.926 | 7.09 |
| 3 models cons. | 0.679 | 0.945 | **0.698** | 0.932 | 0.617 | 0.932 | 0.597 | 0.915 | 17 |
| 7 models una. | - | - | - | - | 0.215 | 0.27 | 0.356 | 0.32 | 8.74 |
| 5 models una. | 0.434 | 0.742 | 0.493 | 0.804 | - | - | - | - | 9.1 |
| 3 models una. | 0.505 | 0.805 | 0.568 | 0.853 | 0.464 | 0.773 | 0.479 | 0.835 | 23 |

Table 4.1: Results ($mAP_{50}$) of various ensemble using various voting strategy. The best value by row are in bold and the value close by 0.002 are underlined. Large consensus ensembles accumulates best values.

good score on the detection of most represented objects. Five models ensemble performs the best on most represented objects at the expense of the detection of the average of all classes. However, this ensemble is nearly two times faster. For smaller ensembles, the $mAP_{50}$ is globally higher using the affirmative voting strategy. The detection of small vehicles is still better performed by the consensus strategy, no matter the size of the ensemble. This can be explained by a better discarding of false positives (recall from 0.611 to 0.624 for 3 models ensemble on r32) that would have been detected by only one (or few) model(s).

The unanimous voting strategy shows poor results on largest ensemble as it includes one model that is trained mainly on small-vehicles and therefore detects other objects poorly. Even by discarding this particular model in subsets, unanimous ensembles performances are still way under fine-tuned models ones and are thus not interesting.

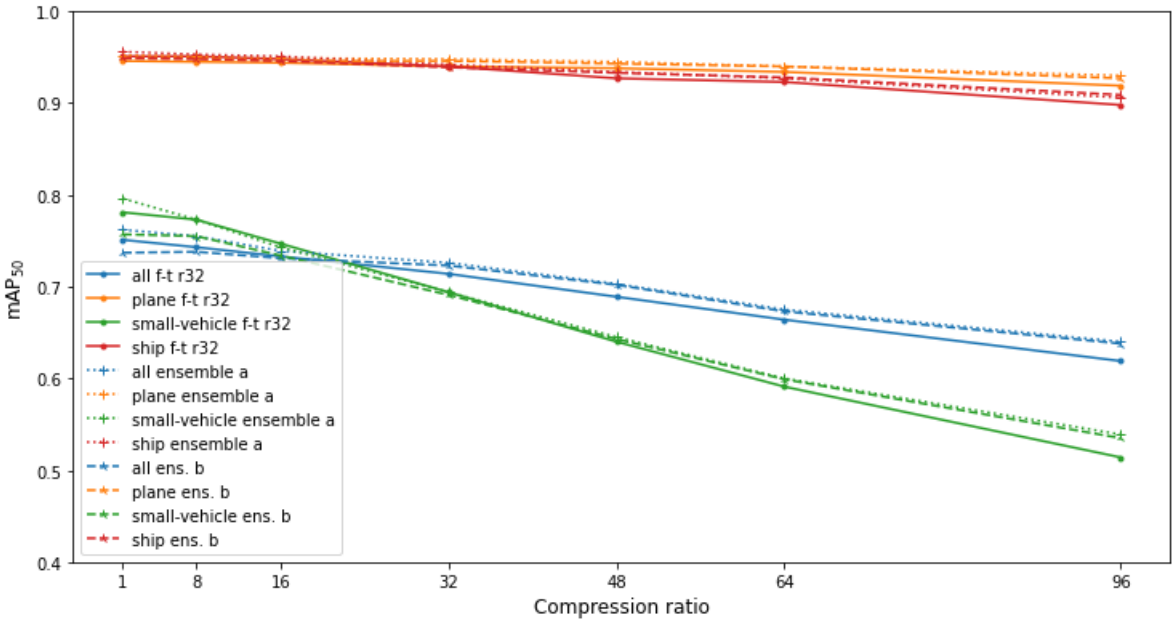
Depending on the goal (objects and their size) of detection of the global model, different ensembles can be chosen. With a view to achieve greater versatility on the most represented objects, consensus voting strategy for large ensemble and affirmative for the smallest will be further analyzed. Table 4.2 compiles the results in the same way as Table 4.1 for various subsets. Ensembles are encoded by tr = fully trained, ft = fine-tuned and sv = small-vehicles, with corresponding ratio.

No subset clearly stands out as the best for both compressed sets, as most of ensembles obtains comparable results. Bigger ones tend to perform as well on all object for both compressed sets but need longer time of inference. The smallest

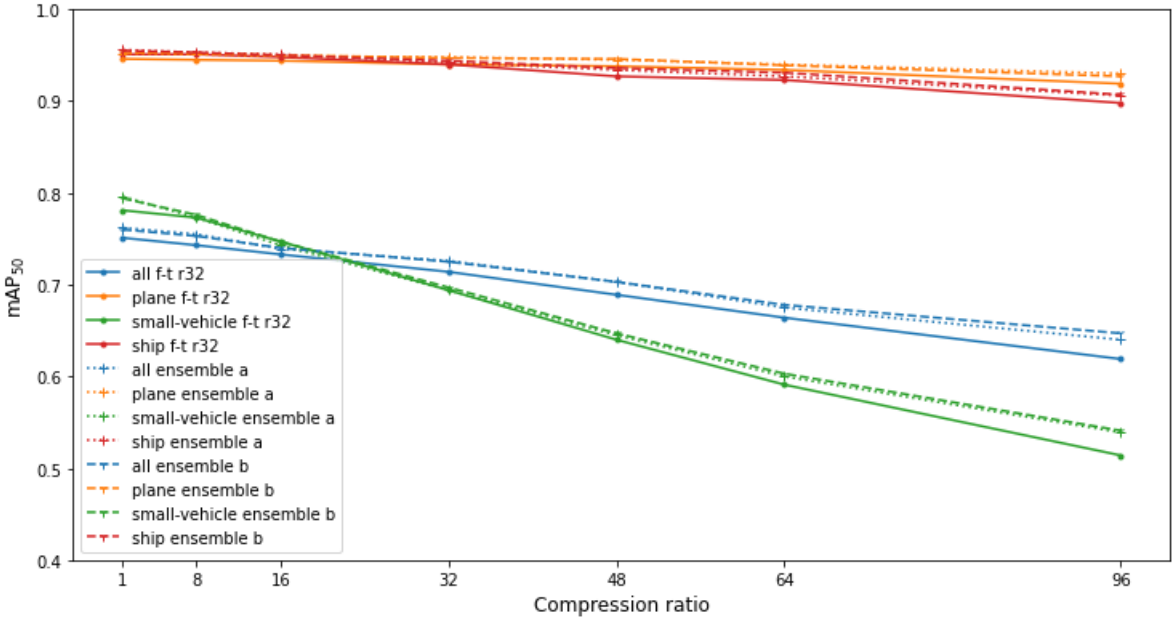
| | Test set r32 | | | | Test set r64 | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | all | plane | car | ship | all | plane | car | ship | FPS |
| 7 models cons. | 0.726 | 0.947 | 0.696 | 0.944 | 0.681 | 0.939 | 0.602 | 0.93 | 4.08 |
| 6 no ftsv | 0.725 | 0.947 | 0.697 | **0.945** | 0.681 | 0.939 | 0.601 | 0.931 | 4.9 |
| 6 no tr1 | **0.727** | 0.945 | 0.696 | 0.944 | **0.682** | 0.939 | 0.603 | 0.932 | 4.9 |
| 6 no ft16 | 0.725 | 0.947 | 0.696 | 0.944 | 0.678 | **0.94** | 0.603 | 0.931 | 4.9 |
| 6 no tr64 | 0.726 | 0.948 | 0.697 | 0.944 | 0.679 | 0.938 | 0.6 | 0.927 | 4.9 |
| 5 no tr64 ft16 | 0.721 | **0.948** | **0.698** | 0.944 | 0.676 | **0.94** | 0.603 | 0.926 | 7 |
| 5 no tr1 ft16 | 0.725 | 0.946 | 0.697 | **0.945** | 0.681 | **0.94** | 0.605 | **0.932** | 7 |
| 5 no tr32 tr64 | 0.721 | 0.946 | 0.695 | 0.943 | 0.677 | 0.937 | 0.598 | 0.925 | 7 |
| 5 no ft16 ftsv | 0.725 | 0.947 | 0.697 | 0.944 | 0.678 | 0.939 | 0.603 | 0.931 | 7 |
| tr1 ft32 ft64 ftsv | 0.714 | 0.945 | 0.696 | 0.943 | 0.674 | 0.938 | 0.601 | 0.927 | 9.25 |
| tr1 ft16 ft32 ft64 | 0.725 | 0.945 | 0.696 | 0.944 | 0.677 | 0.936 | 0.594 | 0.926 | 9.25 |
| tr32 tr64 ft32 ft64 | 0.721 | 0.945 | 0.694 | 0.943 | 0.679 | **0.94** | 0.603 | **0.932** | 9.25 |
| tr1 tr32 tr64 | 0.726 | **0.948** | 0.694 | 0.942 | 0.675 | **0.94** | 0.6 | 0.927 | 17 |
| tr1 ft32 ft64 | 0.714 | 0.945 | 0.695 | 0.943 | 0.673 | 0.938 | 0.599 | 0.926 | 17 |
| tr1 tr32 ft32 | 0.723 | **0.948** | **0.698** | 0.943 | 0.669 | 0.937 | 0.597 | 0.925 | 17 |
| tr1 ft32 ftsv | 0.679 | 0.945 | **0.698** | 0.932 | 0.617 | 0.932 | 0.597 | 0.915 | 17 |
| tr32 ft32 ft64 | 0.722 | 0.946 | **0.698** | 0.944 | 0.677 | **0.94** | 0.603 | 0.929 | 17 |
| tr64 ft32 ft64 | 0.722 | 0.945 | 0.693 | 0.944 | 0.681 | **0.94** | **0.604** | **0.932** | 17 |
| ft32 ft64 ftsv | 0.712 | 0.942 | **0.698** | 0.942 | 0.675 | **0.94** | 0.603 | 0.928 | 17 |
| tr1 tr32 tr64 aff. | 0.724 | 0.945 | 0.691 | 0.939 | 0.672 | 0.938 | 0.596 | 0.926 | 16.5 |
| tr32 ft32 ft64 aff. | 0.72 | 0.946 | 0.694 | 0.942 | 0.676 | 0.939 | 0.598 | 0.927 | 16.5 |
| tr32 tr64 | 0.723 | 0.946 | 0.691 | 0.939 | 0.673 | **0.94** | 0.599 | 0.928 | 44 |
| ft32 ft64 | 0.712 | 0.942 | 0.692 | 0.941 | 0.673 | 0.939 | 0.597 | 0.927 | 44 |
| tr32 ft32 | 0.724 | 0.946 | 0.696 | 0.942 | 0.669 | 0.936 | 0.594 | 0.926 | 44 |
| tr64 ft64 | 0.715 | 0.943 | 0.677 | 0.939 | 0.678 | **0.941** | 0.596 | 0.929 | 44 |
| ft64 ftsv | 0.701 | 0.943 | 0.69 | 0.939 | 0.669 | 0.94 | 0.601 | 0.925 | 44 |
| fine-tuned r32 | 0.714 | 0.94 | 0.694 | 0.94 | 0.664 | 0.934 | 0.591 | 0.923 | 115 |
| fine-tuned r64 | 0.702 | 0.943 | 0.678 | 0.939 | 0.669 | **0.94** | 0.596 | 0.925 | 115 |

Table 4.2: Some results ($mAP_{50}$) of ensemble with consensus strategy evaluated on compressed test set at ratio 32 and 64. The best value by row are in bold and the value close by 0.002 are underlined. Ensembles are encoded by tr = fully trained and ft = fine-tuned and sv = small-vehicles, with corresponding ratio. The best added value to the $mAP_{50}$ due to ensembling is on average of 0.01.

subsets seem better at detecting most represented objects at a given compression ratio and are faster but therefore do not generalize well for compressed image at unknown compression ratio (Fig. 4.4a).




(a) Comparison of the $mAP_{50}$ of the fine-tuned model at compression ratio 32 with ensemble a : consensus tr1 tr32 tr64 and ensemble b : tr32 tr64. Both ensemble perform better than single model for highly compressed images but subset a is still better than fine-tuned model for less compressed images.




(b) Comparison of the $mAP_{50}$ of the fine-tuned model at compression ratio 32 with ensemble a : consensus tr1 tr32 tr64 (the same as Fig. 4.4a) and ensemble b : consensus tr1 tr32 tr64 ft32 ft64. Both ensemble perform better than single model, the second one slightly better for r>16.

Figure 4.4

When evaluating different compression ratios for the test set (Fig. 4.4a and 4.4b), once again, no particular ensemble stands out. While there may be slight performance variations among ensembles of the same size, the primary trade-off lies in deciding between a steady increase in the $mAP_{50}$ across a wide range of ratios or achieving faster inference. This might be decided depending on the intended application.

### 4.2.3 Model ensemble combined with TTA

The combination of the best performing TTA and model ensemble does not guarantee that results will further enhance. The chosen TTA is effective until compressed images at ratio 48 whereas model ensembling improves detection on uncompressed and highly compressed images. Moreover, the total inference time increases due to each model in the ensemble inferring multiple augmented versions of an image.

The main augmentations used for next computations is the affirmative ensemble of resizing with factor 1.1 and with factor 1.5, which is the most effective for detecting the most represented objects. On the other hand, this has a negative impact on the average $mAP_{50}$ of all sixteen classes. Unless specified, all the ensembles of this section use a consensus voting strategy. Part of the results are compiled in Table 4.3 in the same way as previous sections.

| | Test set r32 | | | | Test set r64 | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | all | plane | car | ship | all | plane | car | ship | FPS |
| 7 m. cons. & TTA | 0.706 | 0.95 | 0.713 | 0.946 | 0.652 | **0.943** | 0.607 | 0.931 | 2.5 |
| 5 m. cons. & TTA | 0.707 | 0.95 | **0.714** | **0.948** | 0.658 | **0.943** | **0.61** | **0.933** | 4 |
| 3 m. cons. & TTA | 0.71 | **0.952** | 0.707 | 0.944 | 0.657 | 0.942 | 0.609 | 0.926 | 9 |
| 7 m. cons. | **0.726** | 0.947 | 0.696 | 0.944 | **0.681** | 0.939 | 0.602 | 0.93 | 4.08 |
| 5 m. cons. | 0.725 | 0.947 | 0.697 | 0.944 | 0.678 | 0.939 | 0.603 | 0.931 | 7 |
| 3 m. cons. | **0.726** | 0.948 | 0.694 | 0.942 | 0.675 | 0.94 | 0.6 | 0.927 | 17 |
| ft32 & TTA | 0.69 | 0.951 | 0.711 | 0.945 | 0.629 | 0.94 | 0.588 | 0.928 | 70 |
| ft32 | 0.714 | 0.94 | 0.694 | 0.94 | 0.664 | 0.934 | 0.591 | 0.923 | 115 |

Table 4.3: Results ($mAP_{50}$) of ensembles with consensus strategy and using TTA. They are tested on compressed test set at ratio 32 and 64. The ensembles are composed of (((tr1 tr32 tr64 ) ft32 ft64 ) ft16 ftsv). They are compared with results plotted in previous section. Ensemble using TTA obtains the best precision scores for the three most represented objects.

When combining Test-time augmentation (TTA) with models ensembling, it is observed (Table 4.3) that the detection performance on objects of interest at these compression ratios is slightly superior compared to each technique applied

independently. The synergy between TTA and ensembling indeed leads to improved results in terms of precision and recall, and thus $mAP_{50}$.

Performances over each compressed set is then compared between single model with TTA, ensemble of three models and combination of ensembling and TTA on Fig. 4.5.

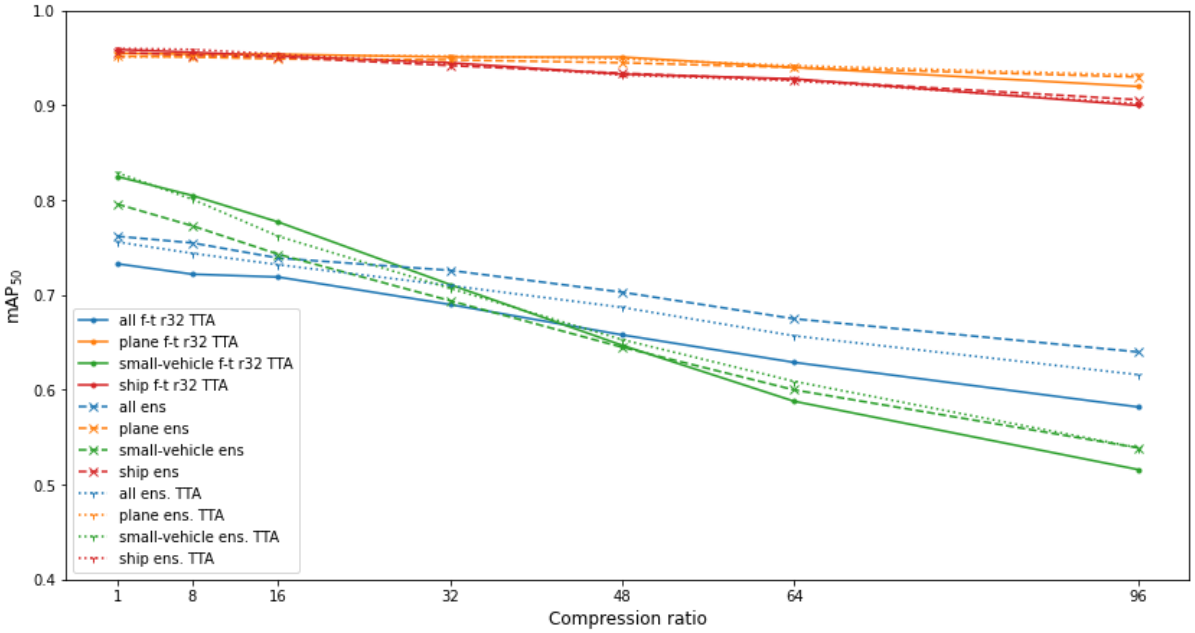



Figure 4.5: Comparison of the $mAP_{50}$ of the fine-tuned model at compression ratio 32 using TTA (resize 1.1 & resize 1.5) with ensemble (consensus tr1 tr32 tr64) using or not TTA. Ensemble with augmentations performs better than ensemble without at every compression ratio but distinguish only from f.-t. with TTA after ratio 32.

Due to the selected augmentations, the $mAP_{50}$ for the average of all classes is negatively affected by the combination of both techniques. For detecting the three most represented objects, both the single model with TTA and the ensemble with TTA yield equivalent results up to a compression ratio of 48, where this type of augmentations starts to lose effectiveness. However, beyond that point, ensembling can provide a boost of up to 0.02 in $mAP_{50}$, as is the case with small vehicles.

The next plot (Fig. 4.6) compares detectors using TTA with ensemble composed of one, three and five models.

There is no noticeable difference in performance between the 3 ensembles up to compression ratio 48, beyond which the larger ensembles perform relatively better, leading to the same conclusion as previous plot.

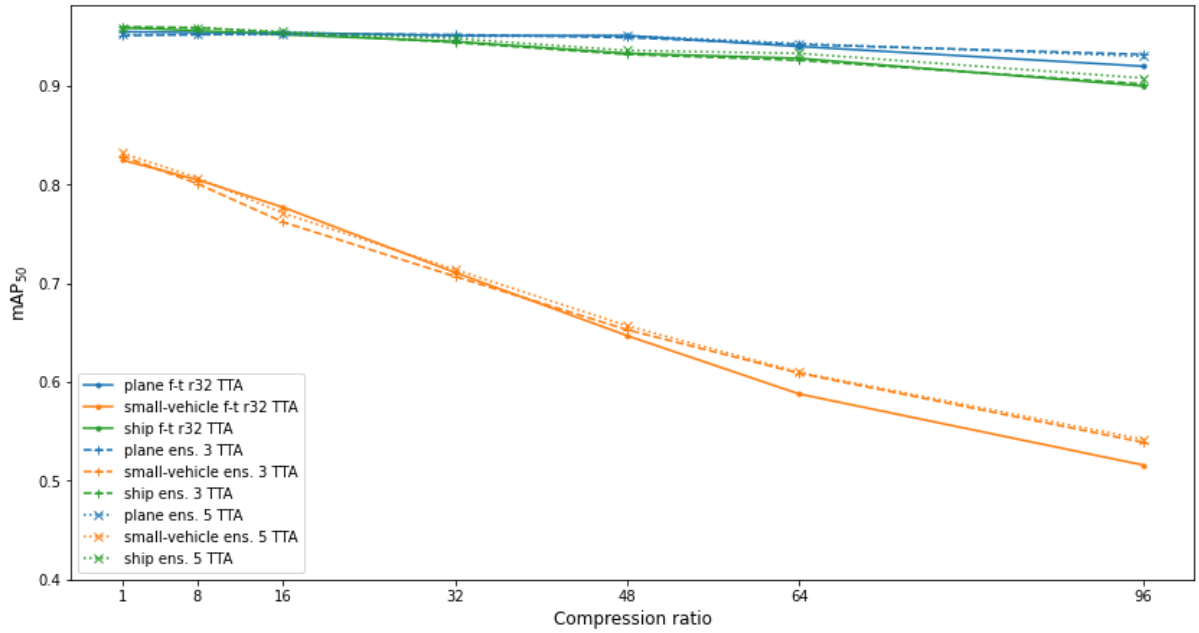



Figure 4.6: Comparison of the $mAP_{50}$ for different ensemble using TTA. Bigger ensembles perform better, especially at higher compression ratio.

### 4.2.4 Simulation of a practical application

The cases described above are suited for detecting objects well represented in the dataset and on compressed images with an undefined compression ratio. However, real-life applications, like satellites in this context, transmit data in a well precise way [28] and should set a compression ratio. In that scope, the simulated application will use a compression rate of 32, resulting in a reduction of file size by a factor of approximately fifteen. The goal will be the detection of the three most represented objects, with a particular attention to small vehicles as planes and ships are detected with a $mAP_{50}$ of above 0.9 using single fine-tuned model. Results for some ensembles are displayed in Table 4.4.

The results demonstrate that the application of Test-time augmentation significantly enhances the performance (defined as $mAP_{50}$) in detecting small vehicles, with an improvement of 0.023 of the base case. When ensembling is applied independently, there is a marginal increase of 0.004 for small vehicles and approximately 0.01 for plane detection. By combining both techniques, a boost of up to 0.031 (equivalent to a 4.5% increase) in $mAP_{50}$ for small vehicles is achieved compared to the fine-tuned model at a compression ratio of 32.

The top three ensemble, in order, are : (tr32 tr64 ft32 ft64 ftsv), followed by (tr32 tr64 ft32 ft64), and finally (tr32 ft32 ft64). All of them utilize the TTA

| | all | plane | car | ship | FPS |
|---|---|---|---|---|---|
| 7 models TTA | 0.706 | 0.95 | 0.713 | 0.946 | 2.5 |
| 7 models TTA2 | 0.682 | 0.949 | 0.718 | 0.948 | 2.5 |
| tr1 tr32 tr64 ft32 ft64 TTA | 0.707 | 0.95 | 0.714 | 0.948 | 4 |
| tr32 tr64 ft32 ft64 ftsv TTA2 | 0.687 | 0.948 | **0.725** | **0.951** | 4 |
| tr32 ft32 ft64 ftsv TTA2 | 0.678 | 0.947 | 0.723 | 0.948 | 7 |
| tr32 tr64 ft32 ft64 TTA2 | 0.686 | 0.948 | 0.724 | **0.951** | 7 |
| tr1 tr32 ft32 TTA | 0.704 | **0.953** | 0.711 | 0.942 | 9 |
| tr1 tr32 tr64 TTA | 0.71 | 0.952 | 0.707 | 0.944 | 9 |
| tr1 ft32 ftsv TTA | 0.66 | 0.95 | 0.711 | 0.935 | 9 |
| tr32 ft32 ft64 TTA | 0.703 | 0.951 | 0.717 | 0.946 | 9 |
| tr32 ft32 ft64 TTA2 | 0.678 | 0.947 | 0.723 | 0.948 | 9 |
| tr32 ft32 ftsv TTA | 0.707 | 0.952 | 0.716 | 0.943 | 9 |
| tr32 ft32 ftsv TTA2 | 0.679 | 0.946 | 0.723 | 0.944 | 9 |
| tr32 ft32 TTA | 0.705 | 0.952 | 0.715 | 0.944 | 16 |
| tr32 ft32 TTA2 | 0.675 | 0.947 | 0.72 | 0.945 | 16 |
| ft32 ftsv TTA | 0.691 | 0.949 | 0.713 | 0.943 | 16 |
| tr1 TTA2 | 0.549 | 0.933 | 0.578 | 0.917 | 64 |
| ft32 TTA | 0.69 | 0.951 | 0.711 | 0.945 | 64 |
| tr32 TTA2 | 0.674 | 0.943 | 0.712 | 0.936 | 64 |
| ft32 TTA2 | 0.659 | 0.943 | 0.717 | 0.945 | 64 |
| ft32 w/ none_resize1.5 | 0.701 | 0.944 | 0.714 | 0.945 | 64 |
| ft32 | 0.714 | 0.94 | 0.694 | 0.94 | 115 |

Table 4.4: $mAP_{50}$ for different objects and consensus ensemble with TTA (resize 1.1 & 1.5) or TTA2 (resize 1.5). TTA2 leads to the better detection, in combination with larger ensembles up to 5 models.

resize1.5. Compared to other ensembles, they excel in detecting small vehicles and ships, while also delivering particularly good performance for planes. The final selection among these sets will depend on the available computational power and the amount of data to process, as the best ensembles are ranked inversely based on speed. The $mAP_{50}$ of the best one is plotted along with other ensembles/models performances on Figure 4.7.

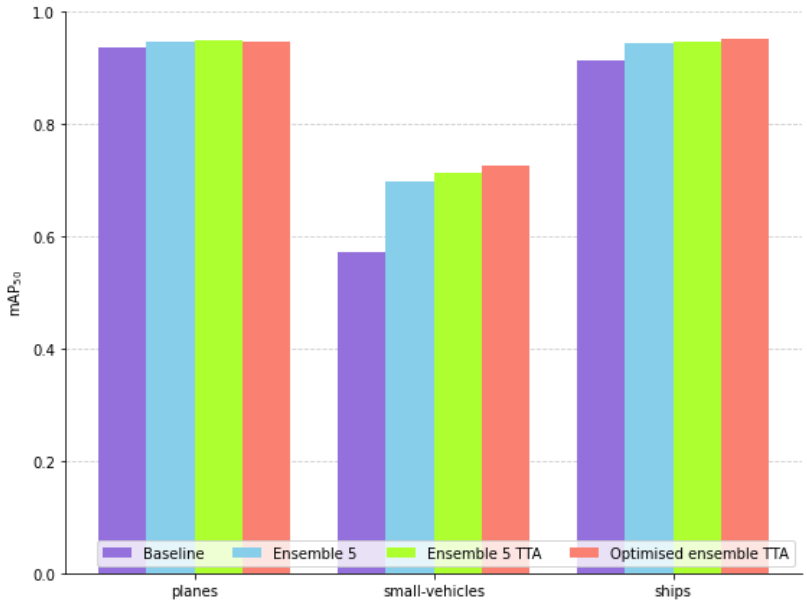



Figure 4.7: $mAP_{50}$ of optimized ensemble in comparison with previous ensembles and models, evaluated on test set compressed at ratio 32. The performance of small-vehicle detection, as well as ship detection, has shown improvement.

## 4.3 Conclusions

Experiments showed that TTA and ensembling models improves detection performances on compressed images, even more when they are combined. However, they should be adjusted in accordance with the desired application.

Among all the augmentations compared, left-right flipping of the image was the most efficient among all classes. However, the resizing to bigger resolution (i.e. upscaling using interpolation) was almost the only augmentation leading to an improve of the mean Average Precision for the three most represented objects. This improvement is even bigger for small objects like cars as they are originally represented by only few pixels.

Experiments showed that ensembling models using a consensus voting strategy leads most of the time to the best overall $mAP_{50}$ (in the case of the same architecture). The selection of the size of theses ensembles is a trade-off between computation time and accuracy. Although smaller ensembles perform equally well as larger ones at a specific compression rate, they tend to lose generalization in favor of speed and specialization (on objects/compression ratio). Therefore, smaller ensembles are a suitable choice for applications that involve detecting objects in images that are often compressed at the same rate. The selection of the models that must be part of the ensemble is also a crucial choice. Detection models should be accurate while being pertinent in regard of the desired application.

For the case of highly compressed images, combining TTA and ensemble of models leads to higher accuracy compared to applying either technique separately. However, experiments (using the available diversity of models) demonstrate that applying TTA alone remains more effective (and faster) for compressed images at a ratio of 16 or smaller.

# Chapter 5

# Conclusions

## 5.1 Limitations

### 5.1.1 Dataset

The DOTA dataset [27] used for experiments during this work is not perfect. It happens that some objects are not labelled. This impacts both the training by reducing the number of seen instances and the evaluation by counting some detections as false positive when they are correct. The distribution of instances per object is not uniform, but the most represented ones were individually analyzed to mitigate this impact. Furthermore, the combination of RGB and grayscale images in the dataset may have a non-negligible negative impact on the accuracy of the models. However, this last factor was not considered during the analysis.

### 5.1.2 Convolutional Neural Network complexity and YOLO version

During this work, only the small version of YOLOv5 was trained and optimized for the sake of computational efficiency. Even if this version has good performances relatively to its complexity (Fig. 2.8), the possibility remains that deeper networks could yield superior results, possibly approaching or even surpassing human-level detection capabilities. During the course of this work, new and more efficient Convolutional Neural Network-based detection models have been published, in particular a new version of YOLO, YOLOv8. Consequently, the effectiveness of the methods proposed in this study may vary, particularly if the base performance of these models converge to a theoretical detection limit.

### 5.1.3 Extensive ensemble research

Finding a subset ensemble that achieves a performance equivalent to that the entire ensemble has been proven to be an NP-hard problem [19]. Therefore, the more potential models that are to include, the more difficult it becomes to find the optimal subset in a decent computation time. However, in the context of this study, since there are few models, they share the same architecture and the best voting strategy is consensus, it was possible to make assumptions about which model can be useful or not to the ensemble in order to quickly find a high-performing ensemble, but never with certainty.

### 5.1.4 Ensemble heterogeneity

All the models used in this work are based on the YOLOv5 network architecture. However, following the literature ([19],[21]), the ensembles benefit from diversity of models, both in terms of their specialization and their algorithm. Indeed, some architectures, for example, enable better accuracy while others have better object discrimination compared to the background. It should be noted that the lack of this type of diversity has not been analyzed.

## 5.2 Future works

### 5.2.1 Super-resolution

Super-resolution consists of enhancing the resolution and level of detail in an image (or video) beyond its original resolution. It can be done by various techniques but most recent and promising ones use deep-learning networks like Real-ESRGAN [29]. It was applied on compressed test set to artificially recovers details but didn't leads to successful results (using generic weights) and need a whole dedicated works. Super-resolution has already been successfully applied to objects detection in paper such [30] and [31], but they focus more on the detection of small objects instead of mitigating the effect of compression (although the subjects are linked). One idea to explore the efficiency of SR regarding the detection on compressed images would be to apply it to one compressed tile, divide it in 4 tiles of the original resolution then performs detection on each of them and finally merge/transform the obtained detection boxes in the original resolution format.

### 5.2.2 Architecture optimisation

As mentioned in Section 2.1.2, some research has been done on training and detecting directly on images in compressed domain. This approach eliminates the

time-consuming requirement of going back to the reconstructed domain before evaluation with a deep learning model while also speeding up the detection process. [7] and [9] works in JPEG domain, with the DCT coefficients, while [32] use compressed features from its own trained compression-reconstruction network. Training a model using DWT coefficients extracted from JPEG2000 compression could also lead to better performance for the same compression rate than a model based on JPEG domain. Ensembling those kind of networks with detectors studied in this work could bring diversity and further improve final predictions.

## 5.3 Summary

The goal of this master thesis was to optimize the detection of objects using CNN on compressed images, maximizing performance at high compression rates. For this purpose, various state-of-the-art methods in object detection and deep learning were analyzed, combined, and applied.

In first place, the state-of-the-art methods review has provided valuable insights into the current cutting-edge techniques and approaches of all related materials.

Then, the first batch of experiments showed that dividing (mega-)image in regular tiles and training the networks using compressed dataset increased accuracy of detection by up to 50% compared to the baseline. In addition, fine-tuning the baseline model allowed to rapidly obtain outperforming models, especially at a desired compression ratio.

The second batch of experiments proved the efficiency of two techniques that boost prediction in both compressed and uncompressed images. Test-time augmentation showed good improvement, in particular resizing of a factor 1.5 by interpolation and left-right flipping, for compressed images until ratio 64. This represents a boost of up to 4% in mAP for small objects and on low compressed image. By selecting models carefully, ensembling demonstrated a slight improvement of around 1% depending on the object and when using the consensus voting strategy. Combining both techniques resulted in an average accuracy boost of 2% for compressed images, with an even more pronounced effect as the level of compression increased.

Finally, the relationship and optimization between object detection and image compression have not been thoroughly explored yet and recent and potentially upcoming works that have been mentioned should be promising in this way.

# Personal ethical reflections

While object detection has numerous beneficial applications such medical diagnostics, autonomous vehicles, and detection of emergency situations, it raises subsequent ethical questions. One big actual concern is about individual's privacy. Biggest companies that should be responsible of protecting personal data of their users are harvesting and analyzing them without their informed consent, thanks to objects detection algorithms and more generally deep learning. While this technology has the potential to enhance user experiences and improve services, advanced profiling such for targeted advertisement can lead, in some case, to the manipulation of individual behaviors [33]. That is why I think that this area of research must remain as transparent as possible and ethical guidelines should be developed and enforced to ensure responsible applications.

Another concern relates to military applications of object detection, such as surveillance and tracking systems, which could invade civilian privacy. Weapons systems automated or assisted by object detection algorithms also raises ethical questions.

# Acknowledgements

In first place, I would like to thank Pr. Benoît Macq for giving me the opportunity to work on this subject, and for his guidance throughout the academic year. His sound advice and supervision helped me to successfully complete my thesis.

Secondly, I would like to thank Karim El Khoury, Tiffanie Godelaine and Maxime Zanella for their availability. Their regular support and willingness to address my questions and concerns have been immensely valuable throughout the course of my research. Their constructive feedback has significantly contributed to the improvement of my work. I am also grateful to Pr. Jean-Didier Legat for being one of my readers.

Finally, I would like to thank my family, my friends and anyone who might have participated directly or indirectly in the accomplishment of my work.

# Appendix A

# Additional figures

## A.1 Architecture of YOLOv5l

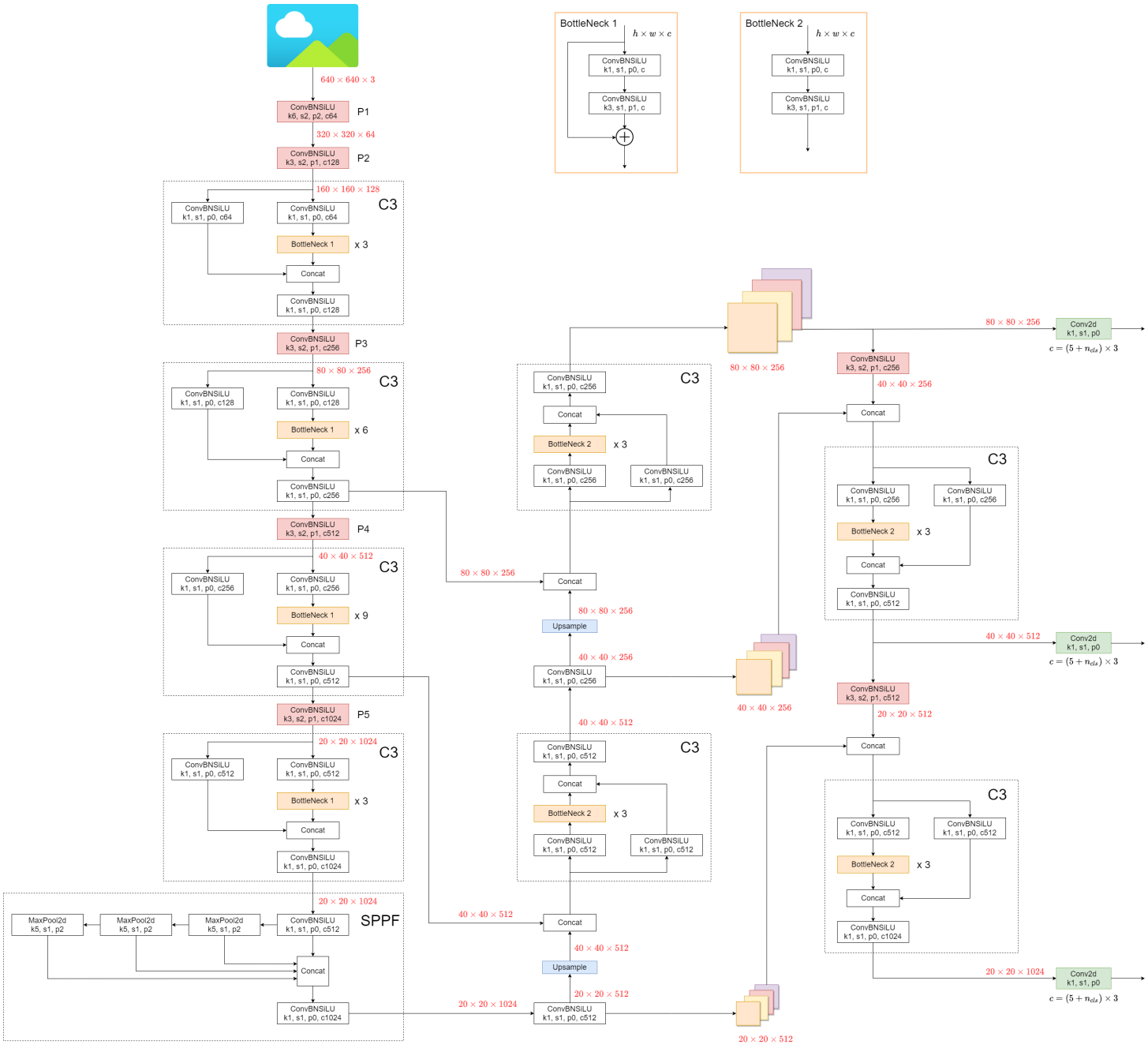



Figure A.1: Architecture of YOLOv5l [17].

# Bibliography


[1] OpenAI. ChatGPT, https://openai.com/research/chatgpt, 2021.

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[3] David S Taubman and Michael W Marcellin. Jpeg2000: Standard for interactive imaging. *Proceedings of the IEEE*, 90(8):1336–1357, 2002.

[4] S Mallat. *A Wavelet Tour of Signal Processing*. Ac. Press, 3rd edition, 2008.

[5] L. Jacques. Lelec2885 : Image processing and computer vision [The Wavelet Transform, slide 51]. Course slides, UCLouvain, https://uclouvain.be/en-cours-2021-lelec2885, 2021.

[6] IntoPIX. Everything you always wanted to know about jpeg 2000. https://www.intopix.com/Ressources/WPs$_a nd_S c_P ub/intoPIX$%20 − %20$Pocket$%20$book$%20$about$%20$JPEG$%202000$.pdf$, 2012.

[7] Benjamin Deguerre, Clément Chatelain, and Gilles Gasso. Fast object detection in compressed jpeg images. In *2019 IEEE intelligent transportation systems conference (ITSC)*, pages 333–338. IEEE, 2019.

[8] B. Ugur Töreyin, A. Enis Çetin, Anil Aksay, and M. Bilgay Akhan. Moving object detection in wavelet compressed video. *Signal Processing: Image Communication*, 20(3):255–264, 2005.

[9] Samuel Felipe dos Santos, Nicu Sebe, and Jurandy Almeida. The good, the bad, and the ugly: Neural networks straight from jpeg. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 1896–1900. IEEE, 2020.

[10] Zhengxia Zou, Keyan Chen, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey. *Proceedings of the IEEE*, 2023.

[11] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[12] Peiyuan Jiang, Daji Ergu, Fangyao Liu, Ying Cai, and Bo Ma. A review of yolo algorithm developments. *Procedia Computer Science*, 199:1066–1073, 2022.

[13] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[14] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[15] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[16] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020.

[17] Glenn Jocher. YOLOv5 by Ultralytics. https://github.com/ultralytics/yolov5, May 2020.

[18] Jonathan Hui. map (mean average precision) for object detection. https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173, 2018.

[19] Omer Sagi and Lior Rokach. Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1249, 2018.

[20] Roman Solovyev, Weimin Wang, and Tatiana Gabruseva. Weighted boxes fusion: Ensembling boxes from different object detection models. *Image and Vision Computing*, 107:104117, 2021.

[21] Ángela Casado-García and Jónathan Heras. Ensemble methods for object detection. In *ECAI 2020*, pages 2688–2695. IOS Press, 2020.

[22] Tomasz Gandor and Jakub Nalepa. First gradually, then suddenly: understanding the impact of image compression on object detection using deep learning. *Sensors*, 22(3):1104, 2022.

[23] Hang Gong, Tingkui Mu, Qiuxia Li, Haishan Dai, Chunlai Li, Zhiping He, Wenjing Wang, Feng Han, Abudusalamu Tuniyazi, Haoyang Li, Xuechan Lang, Zhiyuan Li, and Bin Wang. Swin-transformer-enabled yolov5 with attention mechanism for small object detection on satellite images. *Remote Sensing*, 14(12), 2022.

[24] Pau Gallés, Katalin Takats, and Javier Marin. Object detection performance variation on compressed satellite image datasets with iquaflow, 2023.

[25] Martin Fockedey. Evaluation and optimization of image compression for convolutional neural network in segmentation and classification. *Prom. : Macq, Benoît, Ecole polytechnique de Louvain, Université catholique de Louvain, http://hdl.handle.net/2078.1/thesis:25142*, 2020.

[26] Karim El Khoury, Martin Fockedey, Eliott Brion, and Benoit Macq. Improved 3d u-net robustness against jpeg 2000 compression for male pelvic organ segmentation in radiotherapy. *Journal of Medical Imaging*, 8(4):041207–041207, 2021.

[27] Jian Ding, Nan Xue, Gui-Song Xia, Xiang Bai, Wen Yang, Michael Yang, Serge Belongie, Jiebo Luo, Mihai Datcu, Marcello Pelillo, and Liangpei Zhang. Object detection in aerial images: A large-scale benchmark and challenges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021.

[28] Alcatel. Metop HRPT/LRPT User Station Design Specification. Design Specification EPS-ASPI-DS-0674, EUMETSAT, March 2003.

[29] Xintao Wang, Liangbin Xie, Chao Dong, and Ying Shan. Real-esrgan: Training real-world blind super-resolution with pure synthetic data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1905–1914, 2021.

[30] Jiaqing Zhang, Jie Lei, Weiying Xie, Zhenman Fang, Yunsong Li, and Qian Du. Superyolo: Super resolution assisted object detection in multimodal remote sensing imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 61:1–15, 2023.

[31] Yi Wang, Syed Muhammad Arsalan Bashir, Mahrukh Khan, Qudrat Ullah, Rui Wang, Yilin Song, Zhe Guo, and Yilong Niu. Remote sensing image super-resolution and object detection: Benchmark and state of the art. *Expert Systems with Applications*, page 116793, 2022.

[32] Zhenzhen Wang, Minghai Qin, and Yen-Kuang Chen. Learning from the cnn-based compressed domain. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3582–3590, 2022.

[33] Jim Isaak and Mina J. Hanna. User data privacy: Facebook, cambridge analytica, and privacy protection. *Computer*, 51(8):56–59, 2018.

**UNIVERSITÉ CATHOLIQUE DE LOUVAIN**
**École polytechnique de Louvain**
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**