UCLouvain

École polytechnique de Louvain

JPEG2000 image compression for deep learning of convolutional neural networks

Author: **Maxime Masy**
Supervisor: **Benoît Macq**
Readers: **Christophe De Vleeschouwer, Karim El Khoury, Sébastien Lugan, Jonathan Samelson**
Academic year 2020–2021
Master [120] in Computer Science and Engineering

Abstract

Introduction The Deep Learning of Convolutional Neural Networks has become quite a popular and really effective approach to tackle complex problems especially with images. Also, with the recent improvements to the performance of GPUs the ease of training DL models has increased and their adoption is now broader than ever. However, a big issue still remains; those models require a lot of data to train on and the sizes of the datasets are increasing at the same rate as the complexity of those models. This is why compression has become a necessity in order to reduce the storage and bandwidth requirements. Another issue is that standard codecs were thought and optimised for the human visual system and could be sub-optimal when used for a computer-vision task. In order to address this issue, this master thesis will explore some of the parameters of a specific compression codec and will aim to bring light to their effect on the performance of a computer-vision task.
Material and methods In order to see if some behaviours are generalisable, two bundles each containing a dataset and a model were used. The first bundle used YOLOv4 on a face-mask detection dataset to have a first glance at the influence of the compression as well as to try some ideas. The second bundle used Tiny-YOLOv4 with the UA-DETRAC vehicle dataset in order to get fast training times and this way be able to explore as many parameters of JPEG2000 as possible.
Results The first set of experiments highlights that it could be possible to compress a dataset heterogeneously in order to maximise the overall compression ratio while retaining good performance. The main exploration of the JPEG2000 parameters done in the second set of experiments yields mixed results. Optimising the parameters is difficult because of the random nature of the results but is possible, even though overall this set of experiment did not bring major improvements to the performance.
Conclusion This field is still rather unexplored and future works could have a tangible impact on how the datasets of tomorrow are stored and used.

Acknowledgements

Firstly, I would like to thank professor Benoît Macq for his guidance throughout the year. Our regular meetings helped me stay on track and these discussions were a great source of inspiration.
Secondly, I would like to thank Karim El Khoury, Sébastien Lugan and Jonathan Samelson for their support and availability. More specifically, I want to thank Karim for his great supervision and advice, and Jonathan and Sébastien for their precious help regarding Deep Learning and compression respectively.
Thirdly, I would like to thank the people that proofread my thesis with a special note to my friends and the small game they organised, where to win one had to find an error before the others. I especially thank Alice De Corte for her help and her patience listening to me ramble on about some of my ideas and concerns I had during this master thesis.

Acronyms

CNN Convolutional Neural Networks. 1, 5

DL Deep Learning. 1, 3–5, 11, 14, 15, 33, 53, 54, 56, 58

DWT Discrete Wavelet Transform. 3, 4, 9, 34–37, 39, 42, 43

GAN Generative Adversarial Network. 57

GPU Graphical Processing Unit. 1, 5

ICT Irreversible Colour Transform. 34, 45, 46, 49

MAP Mean Average Precision. 20, 23, 26, 31–34, 36, 47, 48, 51, 54

MRCT Modified Reversible Colour Transform. 46, 49

MSE Mean Squared Error. 4, 38

PSNR Peak Signal-to-Noise Ratio. 8

RCT Reversible Colour Transform. 34, 46, 48, 50

ROI Region Of Interest. 4, 7, 10, 56, 58

SR Super Resolution. 4, 57

Contents

Chapter 1

Introduction

1.1 Context

With the recent improvements to the computational power provided by GPUs and the relative ease of obtaining large datasets of images, Deep Learning and Convolutional Neural Networks have become quite a popular and effective approach to tackle complex tasks of computer vision.
But in addition to the high computational cost of Deep Learning, the datasets containing huge quantities of images require a lot of space to store them. This is why most datasets come with compressed images to lower this storage requirement. It then becomes important to find the right balance between the amount of compression and the reduction it induces on the Deep Learning performance.
Also, would it be possible to reduce this performance drop by tweaking some of the parameters of the compression ?
This master thesis will aim to answer this question and will have as main focus the exploration of the impact of the compression parameters on the computer-vision task.

1.2 Master thesis plan

Main objective The main goal of this master thesis will be to explore the parameters of the JPEG2000 compression and see how they affect the performance of a Deep Learning task.
Background material First, all necessary material to motivate the choice of the compression scheme and DL models as well as to understand them will be explored in the next chapter.
Previous work in the field In Chapter 2, previous works that used compression with a Deep Learning task will be explored in order to see how what is done in

this master thesis compares to them as well as learn from and apply a few of their ideas.

Experimental study As JPEG2000 is quite a complex compression scheme with a lot of possible parameters to tune; this master thesis will focus on some of the most promising ones. Chapters 4 and 5 will take the form of a collection of experiments that are run across two different pairs of datasets and model complexities. In Chapter 4, YOLOv4 will be used on a face-mask dataset to setup a baseline performance curve and see how the compression affects the model performance as well as try some promising ideas.

The main exploration of the JPEG2000 parameters is located in Chapter 5 and will try to unearth any notable improvements to the computer-vision task. The explored parameters were chosen to cover several of the main stages in the compression process to see which one could bring the best possible improvements. Tests will be run on a vehicle dataset using the really fast Tiny-YOLOv4 model so that more parameters can be explored.

Multiplying tests across different datasets and model complexities will help discover any disparity in behaviour if present.

Limitations and future works Chapter 6 will address the encountered limitations as well as new possible ideas in this field that could be explored.

Conclusion Lastly, Chapter 7 will summarise the work that has been done, the limitations encountered, the available perspectives.

Chapter 2

Background Material

2.1 Compression

Choice of the model The compression model that was chosen for this master thesis is JPEG2000. This choice was made with several reasons in mind. Firstly it is a continuation of preceding works that aimed to optimise JPEG instead of JPEG2000 as it is one of the most popular codec (see Section 3.1). Secondly, JPEG2000 brings a lot of improvements over JPEG (see a comparison of the compression performance in Figure 2.3 below) and does have a lot of nice properties that could eventually be exploited to harness better results or apply innovative ideas. These properties are as follow :

1. **Resolution scalability:** JPEG2000 is able to generate lower resolution versions of the original images at half the resolution, a quarter, etc.

2. **Distortion scalability:** It is possible to progressively improve the quality of the image using several quality layers up to a lossless representation.

3. **Component scalability:** An image can be first represented using only grey levels and then be represented in colours by adding them later on.

4. **Region Of Interest coding:** JPEG2000 lets the user define a main region of focus where the effect of the compression will be lessened at the cost of increasing it everywhere else.

The rest of this section will quickly go over the main building blocks of JPEG2000.

Figure 2.1: JPEG (PSNR: 26.64 dB). Figure 2.2: JPEG2000 (PSNR: 29.22 dB).

Figure 2.3: Comparison between JPEG and JPEG2000 at 0.2 bpp [5].

2.1.1 Colour transform or decorrelation transform

In this first block, the idea will be to change how the image is encoded. Usually an image is encoded on three distinct colour channels corresponding to the three basics colour : Red, Green and Blue (RGB). Although this system works well to store an image somewhat efficiently, it becomes better to use a different representation when your main focus is to compress the image. The main representations used in this case are the YCbCr for the irreversible colour transform and the YCoCg for the reversible one. A good illustration comparing the RGB and YCbCr can be found in the Figure 2.4 below. Those representations are used in compression because they can concentrate the majority of the information in the Y luminance/luma channel and decorrelate the information between the different channels. This is also because of this change of channels that JPEG2000 offers component scalability as the Y channel is taken into account first and chrominance information can be added if enough bitrate is allocated.

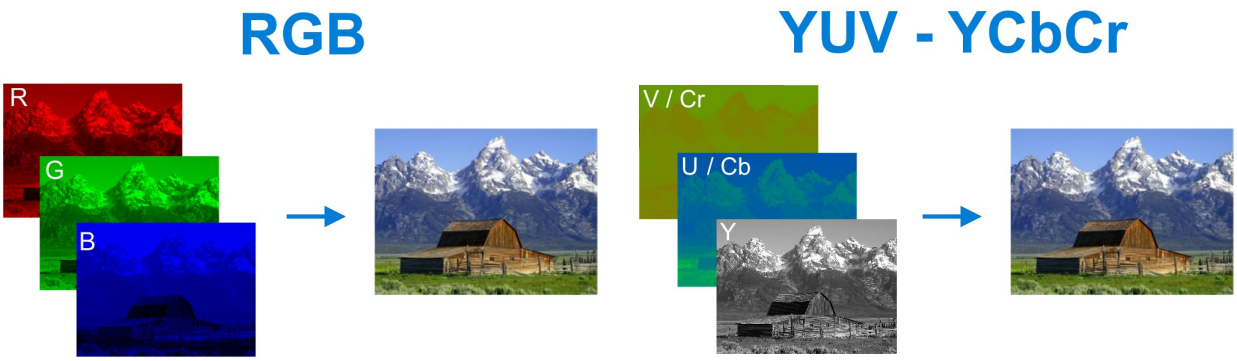

Figure 2.4: RGB versus YCbCr image encoding comparison [11].

2.1.2 Discrete Wavelet Transform

The main part of the JPEG2000 compression scheme revolves around the use of the Discrete Wavelet Transform. This special transformation helps transform each image channel into a multi resolution decomposition. It is this transform that gives the resolution scalability to JPEG2000 as it is possible to generate smaller resolutions (1/2, 1/4,...) variations of the original one. In the Figure 2.5 you can see a representation of how the image is decomposed and stored.

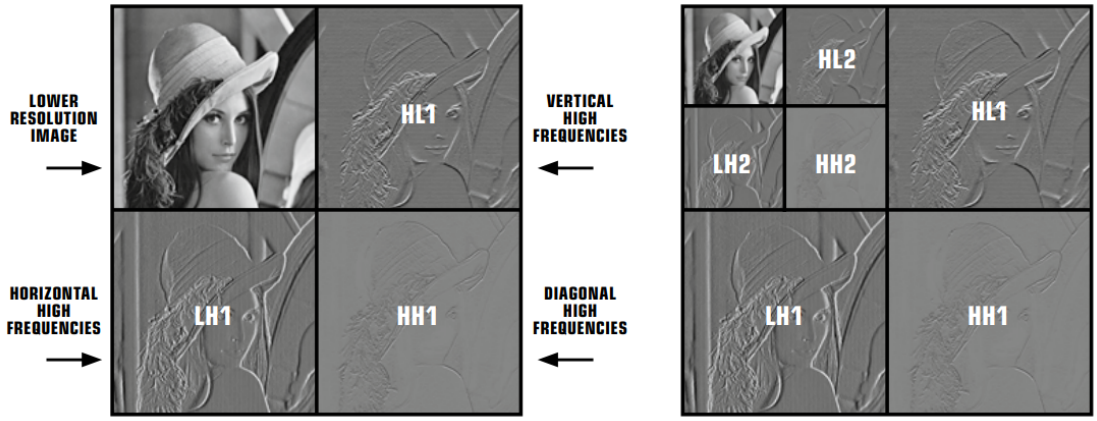

Figure 2.5: Illustration of the Discrete Wavelet Transform [2].

2.1.3 Quantisation

After having generated the coefficients of the DWT representation, they are quantised, so they are approximated to the nearest discrete value as illustrated in the Figure 2.6 below.

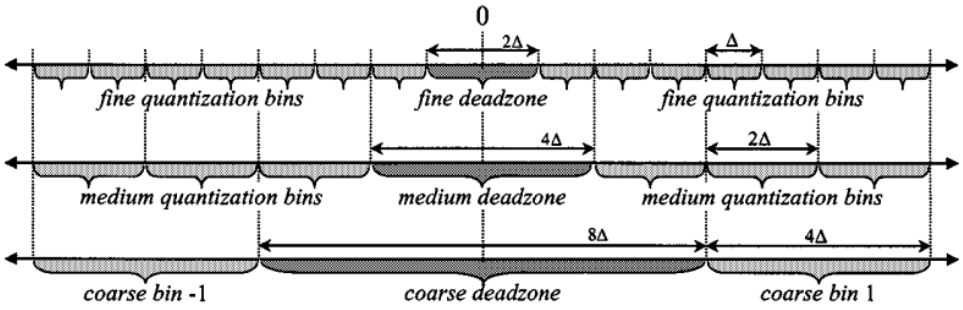

Figure 2.6: Illustration of the deadzone quantifiers each having a region twice a large around zero [16].

2.1.4 Region of interest

JPEG2000 also lets you define regions of interest to separate between foreground and background. This works by giving a finer quantisation to the foreground at a cost of a coarser one for the background. An example of compression using this idea to highlight a face is available in the Figure 2.7 below.

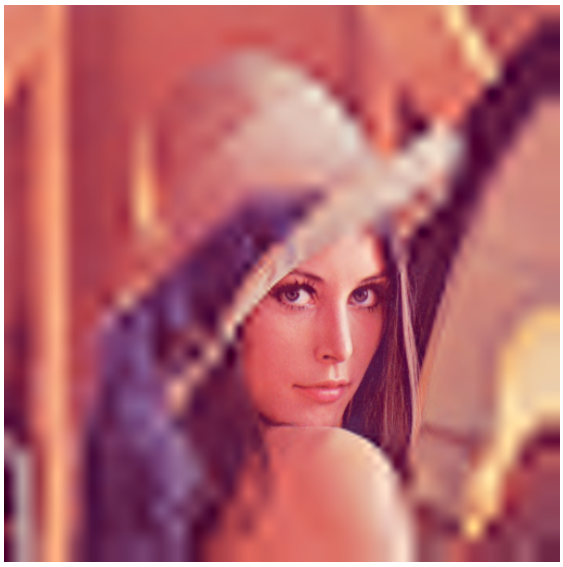

Figure 2.7: Example use of the Region Of Interest in JPEG2000 [5].

2.1.5 Entropy coding

This last important step will aim to represent the most frequent quantised coefficients using a shorter representation in term of number of bits and a longer

one for the less frequent ones. Doing so, it manages to losslessly compress the already compressed representation of the image even more.

2.2 Computer vision

Choice of the Deep Learning task : The detection task was chosen mainly to be a good middle ground between a simpler image classification task and a more complex segmentation mask task with labels. It will also be interesting to see if the behaviour for this task vastly differs from the one using a segmentation model like U-net (see Section 3.1 of the state-of-the-art review).

Choice of the model : Nowadays there is a lot of renowned Deep Learning models out there that are good candidates to choose from as they help for reproducibility and are representative of the state-of-the-art. As this master thesis is taking the form of an exploration of the compression parameter, the chosen model needs to be fast to train and evaluate. This is the main reason why YOLOv4 and Tiny YOLOv4 were chosen (from this revision creator's Github [10]). But this was not the only motivation in using them, in this case YOLOv4 represent a somewhat complex model with state-of-the-art performance. Tiny YOLOv4, on the other hand, represents a more lightweight model with lower accuracy but still has very acceptable detection performance.

Main ideas of YOLO The algorithm works by applying a neural network on the complete image. This neural network will in turn separate the image in a grid. Then for every cell in the grid, B bounding boxes are predicted, containing a class and a confidence (see Figure 2.8 below). The confidence denotes how certain the model is that an object is present in the bounding box and how well the bounding box encloses the object.

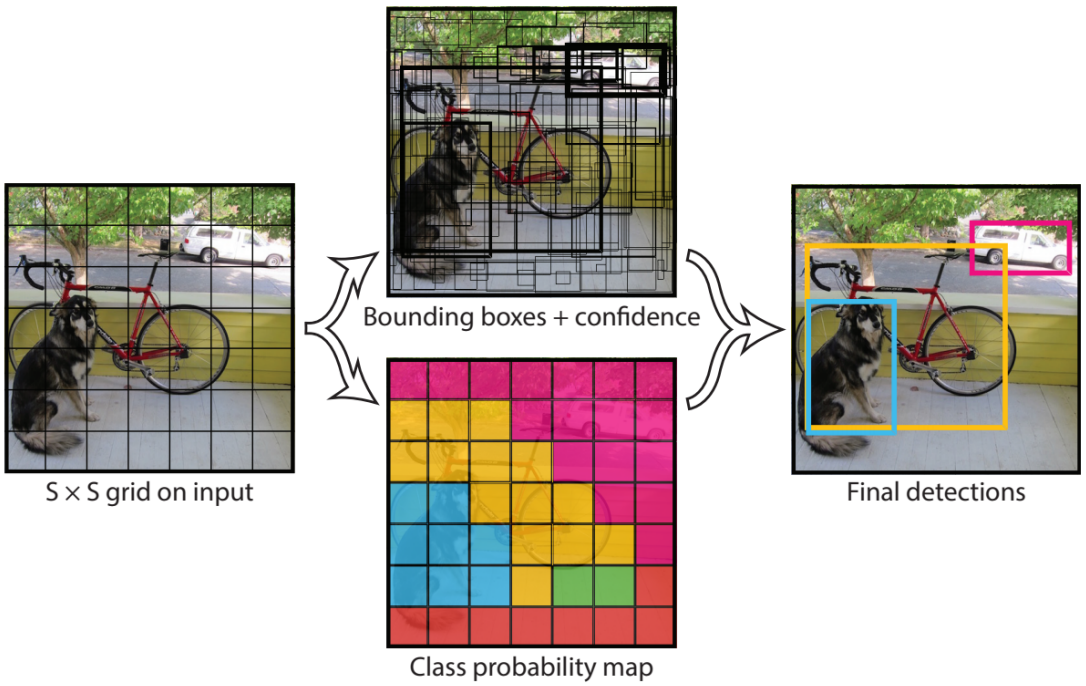

Figure 2.8: Illustration of the inner working of YOLO from the YOLO paper [15].

2.2.1 YOLOv4

YOLOv4 is the fourth revision of YOLO and brings improvements to the architecture. For example, it implements state-of-the-art neural connections like Weighted-Residual-Connections (WRC) and Cross-Stage-Partial-connections (CSP). It incorporates new features such as Cross mini-Batch Normalization (CmBN), Self-adversarial-training (SAT) and Mish-activation. It also uses a built-in mosaic data augmentation technique to improve the training efficiency. Using all of this, YOLOv4 manages to have state-of-the-art performance while being extremely fast (see Figure 2.9). Indeed, it can reach 53 frames per second (fps) for inference on a 2080ti using the Darknet implementation (the one used in this master thesis) and 150 fps using the tkDNN one with FP16 precision and a batch size of 4.

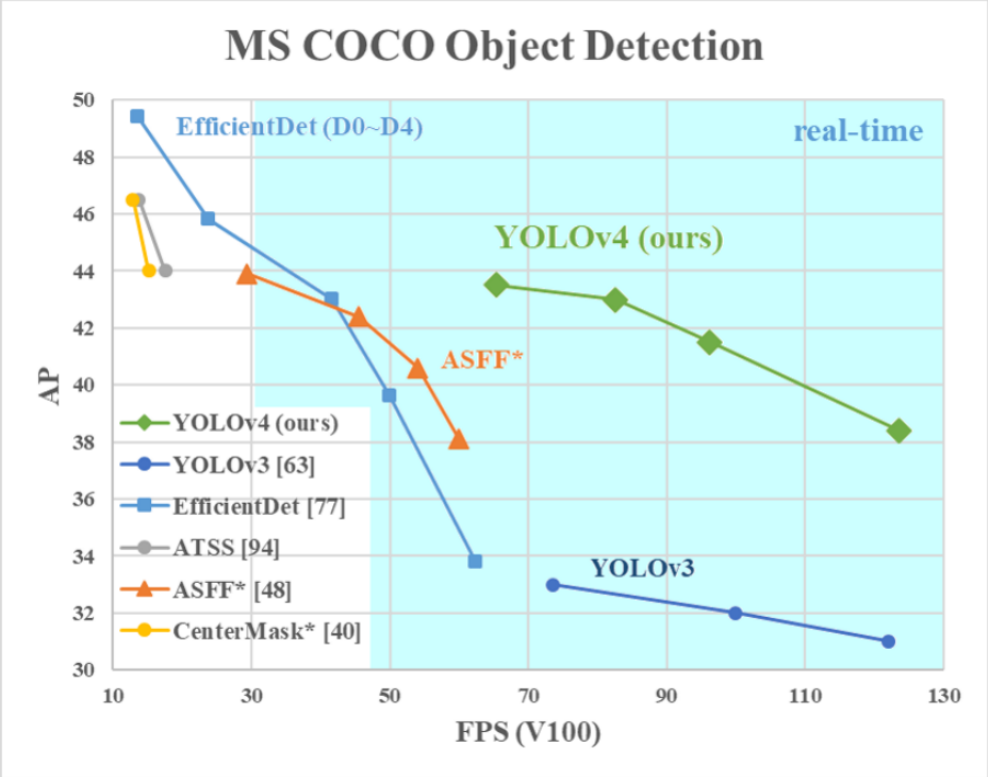

Figure 2.9: Comparison of the average precision (AP) of well-know detectors from the YOLOv4 paper [12].

2.2.2 Tiny-YOLOv4

The tame down version of YOLOv4, tiny-YOLOv4 uses less convolutional and a simpler architecture but manages speeds beyond real-time. It can reach an inference speed of 443 fps on the Darknet implementation with a RTX 2080ti (the one used in this master thesis) and up to 1774 using the tkDNN one with FP16 precision and a batch size of 4. This increase in speed comes at a cost; it has roughly 2/3 of the accuracy of YOLOv4 on the MS COCO dataset (a complex dataset containing a great variety of objects: [7]). This decrease in performance is not an issue as this master thesis aims to evaluate the impact of compression parameters, and it will enable the possibility to perform more tests as the training time decreases accordingly.

Chapter 3

Compression for deep learning state-of-the-art review

This chapter will cover some of the existing works that did inspire this master thesis. Firstly the impact of the compression on a Deep Learning segmentation task will be explored. Also, the idea of optimising the compression for the Deep Learning task is not new and previous works on the JPEG compression will be reviewed.
Secondly, we will take a closer look at possible optimisation of the deep learning making use of an uncompressed dataset in order to boost the performance on the compressed one.
And lastly a more in depth review of the colour transforms and how it was thought for the human visual system will be done.

3.1 Impact of compression on Deep Learning and JPEG optimisation

Impact of compression

This first work is from a previous master thesis of a fellow engineer (see [14]). He did several experiments that this master thesis aims to pursue on other tasks to see if their behaviour differ. For example, he explored and compared the impact of the JPEG and JPEG2000 compression ratio on the performance of a 3D U-net segmentation model that was segmenting organs in CT and CBCT medical scans. Some results of this experiment are available in the Figure 3.1 below and we can see that JPEG2000 largely outperforms JPEG. The metrics used are the Dice Similarity Coefficient (DSC) and the Symmetric Mean Boundary Distance (SMDB) and the results are better for higher and lower values respectively. We can see that

the performances of both curves at lower compression ratio are on par, but then JPEG is not able to go past a ratio of 48:1 and has worse results around that mark. This is the main reason why only the parameters of JPEG2000 will be explored in this master thesis.

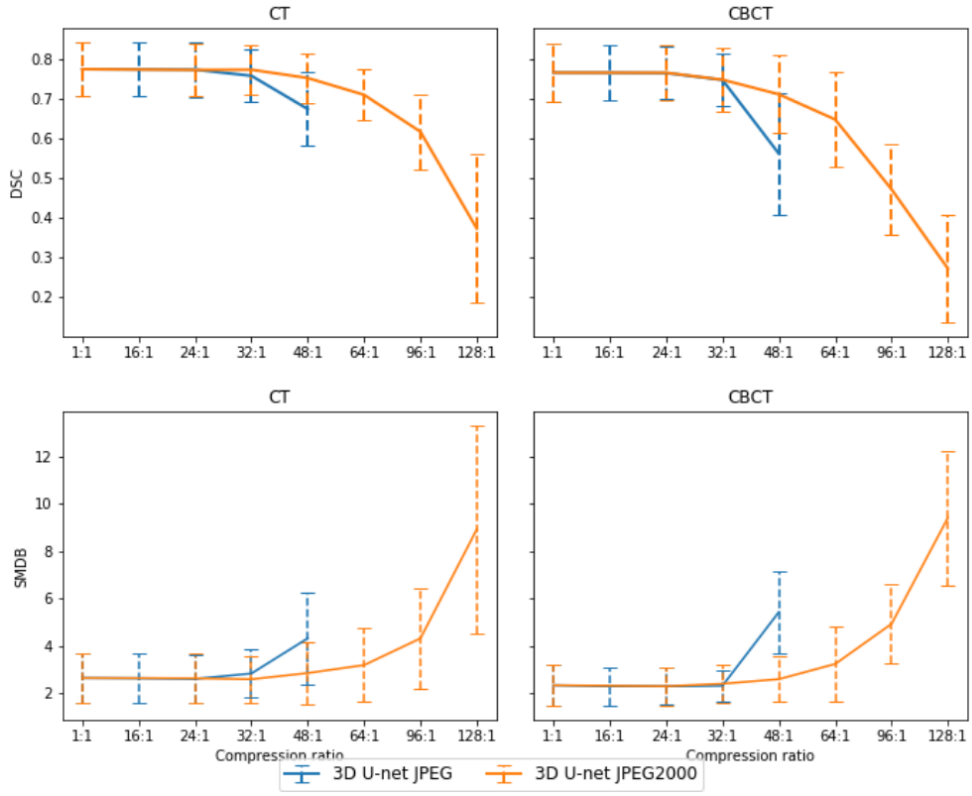

Figure 3.1: 3D U-Net trained on uncompressed images and tested on JPEG2000 and JPEG compressed images.

Optimising JPEG

Another experiment he did was to try to optimise the JPEG quantisation tables using different optimisation methods in order to improve the DL performance. The optimisation methods used were the following : a Lagrangian optimisation and a genetic one. The dataset that was used for this optimisation was the CIFAR-10 (see [1]) that contains 60,000 images of 32x32 pixels and is used for classification in 10 different classes. The classification model he used is small with only 4 convolutional layers and a fully connected one. He did manage to slightly improve the precision of the model but not by a huge amount (it did go up to 3%). It will be interesting to see if it is possible to harvest more improvements by optimising some of the JPEG2000 parameters.

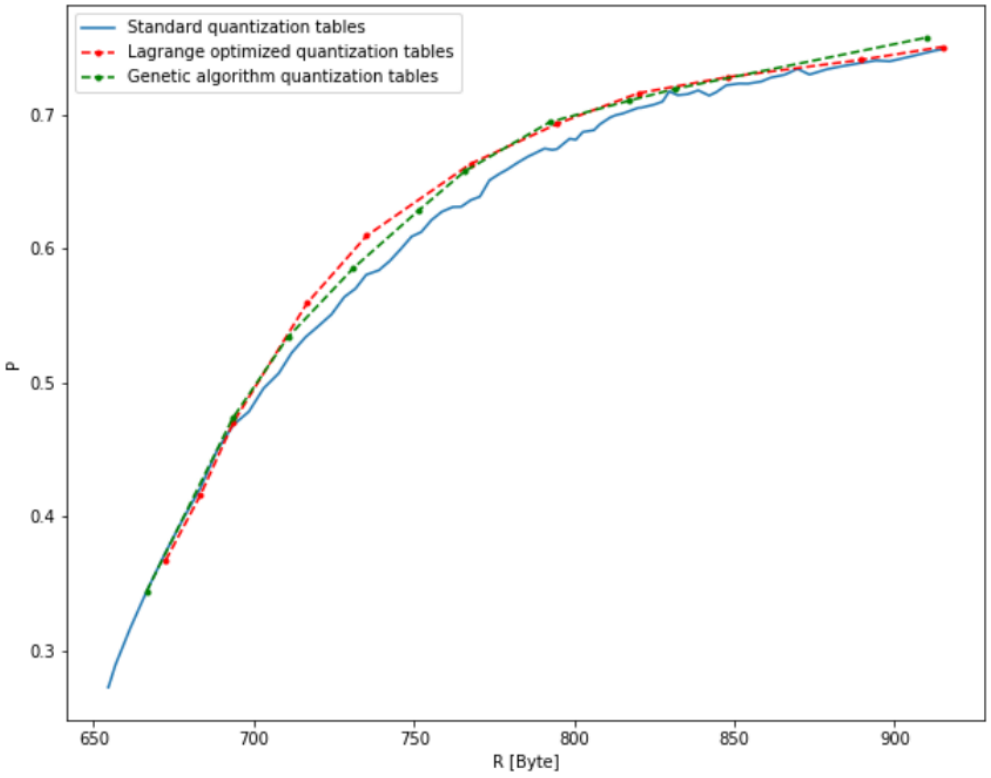

Figure 3.2: Performance of the Lagrangian optimised quantisation tables (red), the Lagrangian optimised quantisation tables (green) and the standard quantisation tables (blue) (from [14]).

3.2 Optimising deep learning

The previous work also made some nice discoveries when it comes to training a neural network on compressed data and a paper was written on this subject [13]. The main observation was that it is possible to increase the performance on the compressed dataset if the computer-vision model is first trained on uncompressed data and then fine-tuned on the compressed one at the right ratio instead of being solely trained on compressed data. The main results of this fine tuning and the corresponding illustrations of the produced segmentation mask are available in the two figures below (Figure 3.3 and 3.4). This idea of two stages training will be explored and applied somewhat differently in this master thesis in order to not only gain performance but also time.

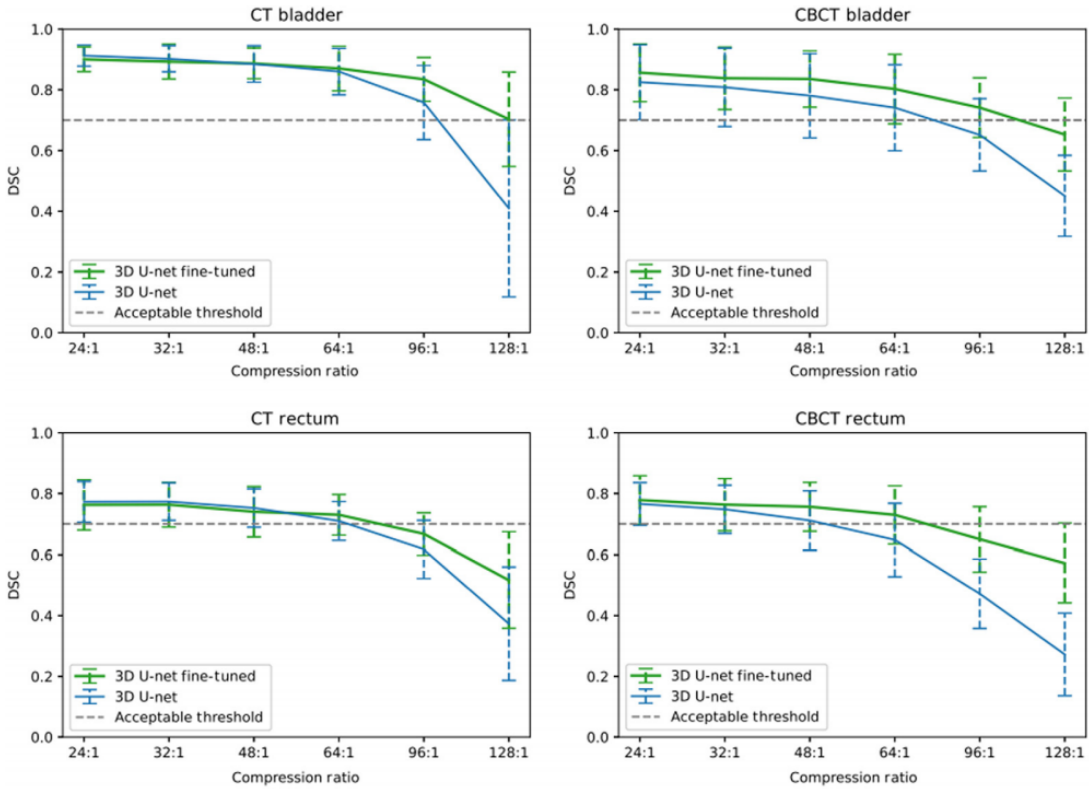

Figure 3.3: DSC performance after training and testing at several compression ratios for 3D and 3D fine-tuned U-Net segmentation of the bladder and the rectum on CT and CBCT images (from [13]).

Figure 3.4: Visual results for bladder segmentation using 3D and 3D fine-tuned U-Net with a compression ratio of 96:1 on a CBCT scan. (from [13])

Chapter 4

Experiments using YOLOv4 on the face mask dataset

4.1 Experimental setup

The first set of tests were done using YOLOv4 on a small but relatively high quality dataset found on Kaggle: the face-mask dataset [3]. This dataset contains images of people wearing a face mask or not and the goal of the detection task is to detect both cases (see Figure 4.1). It has only 700 images in the training set and 220 images in the test set at various resolutions. When using YOLOv4, a good rule of thumb is to have roughly 1000 examples of each class to train on. Therefore some data augmentation techniques were used in order to enrich the training set. Also, while YOLOv4 is a rather fast detection model, training it still required $\sim$ 14-17 hours on my personal machine[1]. For all following experiments, if not specified otherwise, the datasets were compressed using OpenJPEG into the j2k file formats and then decompressed back into png to feed them to YOLOv4. Note that neither validation set nor re-sampling method like cross validation were used as this set of experiments aims to give a rough idea of how the performance evolves at different compression ratios.

[1] an AMD Ryzen 5 3600, Nvidia RTX 2060 and solid state drive equipped PC

Figure 4.1: Example image of the face mask dataset with the prediction boxes and labels.

4.2 Data augmentation

Using the *imgaug* data augmentation library and a mix of some of the available transformations, the amount of samples in the training set went up from 700 to 2100 which is a more reasonable amount of images to train YOLOv4 on. The used library is implemented in python and is available on Github [4]. A sample image and the transformed version are available for comparison in Figure 4.3. The exact transformations used and their random intensity range are reported in the table below :

transformations	% of data to apply on	random intensity range
Canny, additive gaussian noise	40%	[0, 0.2], 0.05*255
Gaussian blur	40%	[0, 2]
Multiply hue and saturation	40%	[0.5, 1.5]
Multiply brightness	80%	[0.5, 1.5]

Figure 4.2: Data augmentation transformations and parameters.

Figure 4.3: Example of data augmentation randomly multiplying the hue and saturation.

4.3 Baseline performance

This first experiment will aim to establish how the detection task using YOLOv4 is affected by the compression ratio parameter of JPEG2000. To do so, YOLOv4 was trained on the uncompressed dataset and the performance is evaluated on datasets compressed at different compression ratios. The Mean Average Precision for each class and the overall mean are reported in the Figure 4.4 below.

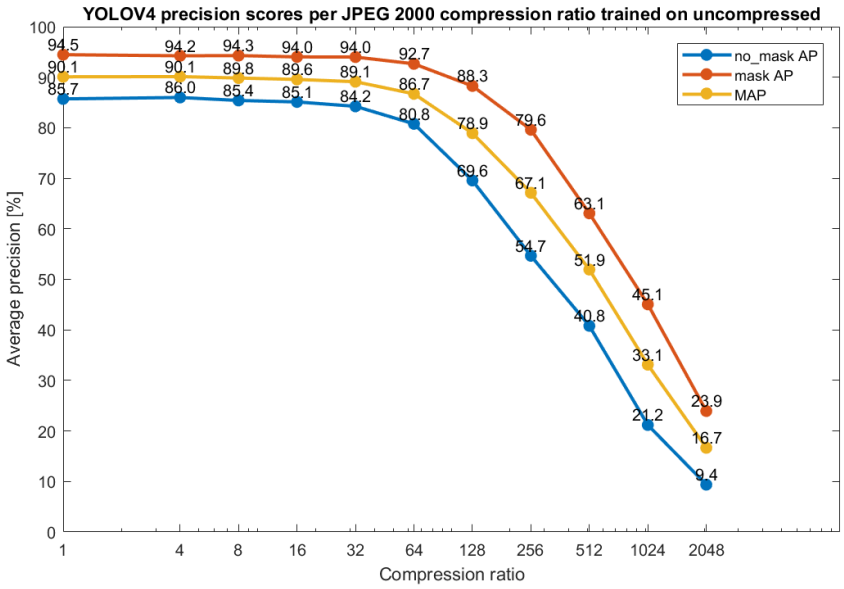

Figure 4.4: Baseline performance of YOLOv4 on the face mask dataset at different compression ratios.

Interpretation There are several things to notice in this graph. Firstly, the detection of a masked person seems easier for the detection model. This could either come from a better coverage of the training set for this class or how recognisable a mask is for the model (as it is does have a simpler profile than a face in term of geometry, details and colour). Secondly, the performance only really begins to drop at a compression ratio of 128 and starts to reach an unacceptable level (<70%) at 256.

4.4 Compression for data augmentation

By using all available data (all images at all the compression ratios) it is possible to create a new data-augmented dataset containing 11 times more data. Training on it and evaluating it for different compression ratios gives us the following results:

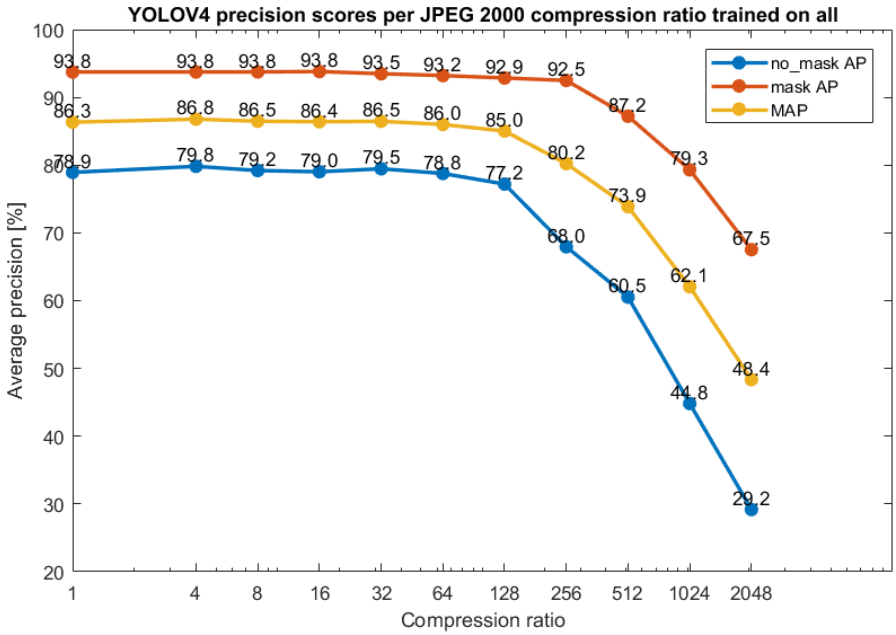

Figure 4.5: Performance when training on all images at all compression ratios and evaluating at the specific ratio.

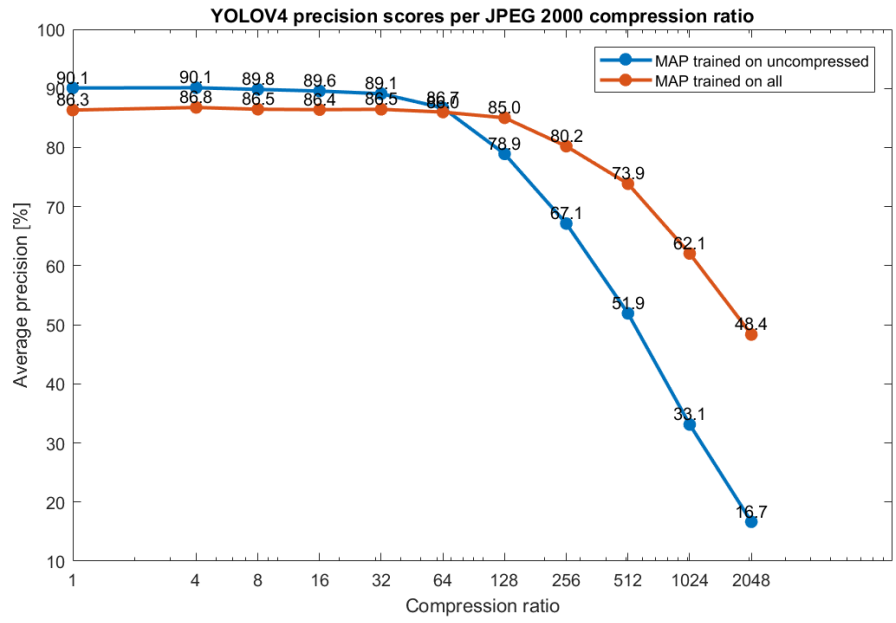

Figure 4.6: Comparison to the baseline curve.

Interpretation When compared to the baseline curve, the performance somewhat decreases for small values of the compression ratio but tremendously increases for higher ones. These results can be explained by the fact that the quality of the images varies a lot with the compression ratio. Therefore YOLOv4 is not able to discover and learn a universal truth for detecting the classes but is still able to have a quite good representation that can hold itself at higher compression ratios. For a performance threshold of 70%, we can retain an acceptable level of performance up to a ratio of 512 (73.9% MAP) which is a nice improvement over the previous maximum ratio of 128 (78.9% MAP). It is also possible, for example, to reach a higher value of the MAP at a higher compression ratio than in the previous curve. Indeed the model achieves 80.2 MAP at a ratio of 256 in the new curve compared to the previous 78.9 at 128.

4.5 Joint Progressive decompression

4.5.1 Different impact of the compression

On the compression side of things: The amount of information lost in an image to reach a given compression ratio depends from one to the other. For example, take an image containing a few pure solid colours, compressing it heavily will not make it lose much information. But an image of a people-filled stadium will on the contrary become blurry quite quickly as the higher resolutions bring a lot of information in this context.
On the computer-vision side of things : An image could be harder or simpler for the detection task. For example when detecting faces, it is easier to detect in a portrait than in a crowded stadium where faces are minuscule. Thus any compression on harder images will probably remove information necessary to make out difficult objects in a scene. therefore images like that should probably be less compressed and portrait-like ones could be compressed a bit more and still retain good detection performance.
This is why it is important to know if the information that will be lost by compressing the image is useful for the computer-vision task. This experiment will try to find the best compression ratio to use for each image in the test set by evaluating the confidence of all the found predictions with a decreasing ratios until they are satisfactory (confidence $> 90\%$). The compression ratio found for each image brings enlightenment on how much sensitive they are to compression : a high compression ratio denotes an image that does not loose much useful information for the detection task. On the contrary, a low one denotes that the image will quickly loose useful information or that was already quite hard to do detection on. An exact description in pseudo code of the algorithm (Algorithm 1) can be found

in the appendix A. Also, an illustration of the found ratios (Figure 4.7) can be found below :

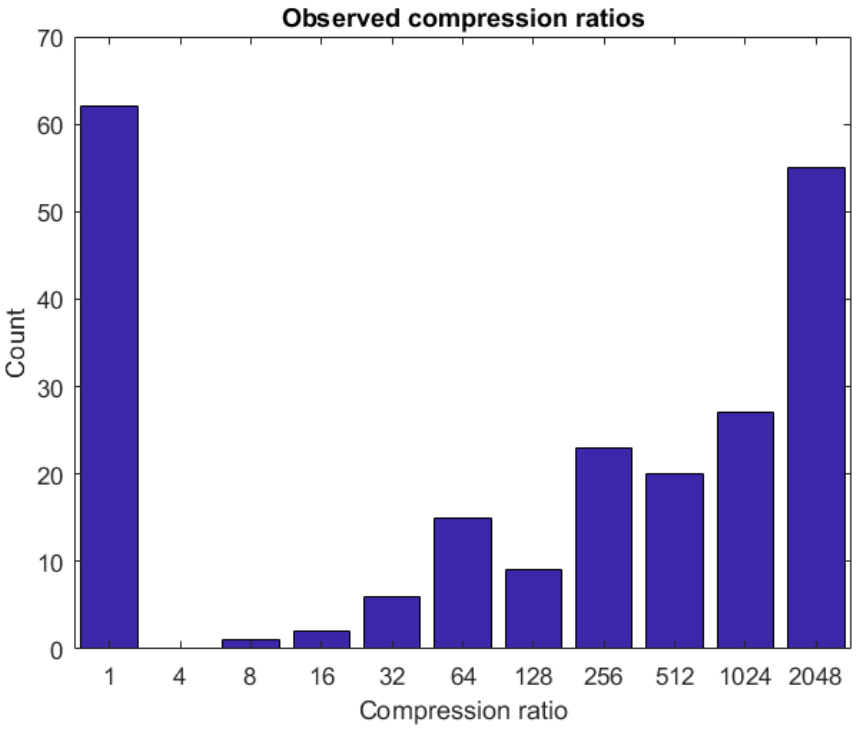

Figure 4.7: Observed compression ratio to reach 90+ % confidence.

Interpretation The first thing to observe is that the model did reach a satisfactory level of confidence for a good amount of the images with a compression ratio of 2048. All images that were assigned a 1 are in fact images for which the model was still uncertain even for the uncompressed version of the image. In practice, those images could be assigned a threshold compression ratio defined as the lowest one we agree to compress the images at. In this case, it could be optimal to choose 8 as there is no improvement by lowering the compression ratio any further. It could also be set to a compression ratio where the performance is acceptable, in this case at 128 (see Figure 4.5) as the performance only really starts to drop at a ratio of 256.

4.5.2 Effective compression ratio

With all the individual compression ratios, it is now possible to find an effective compression ratio that summarise how compressed the images are and to compute the performance associated to it. By averaging the found ratios (in Figure 4.7) with their respective appearance frequency, an excellent effective ratio of 722 is found. This effective ratio is in fact even higher, as if we take the resolution of the images into account, it seems that bigger images are able to be compressed more and smaller ones less. Doing so gives an effective ratio of 1016. Note that both effective ratios would be a bit bigger if a threshold for the lowest compression ratio was used as discussed in the paragraph before (724 and 766 instead of 722 for a threshold of 8 and 128 respectively).

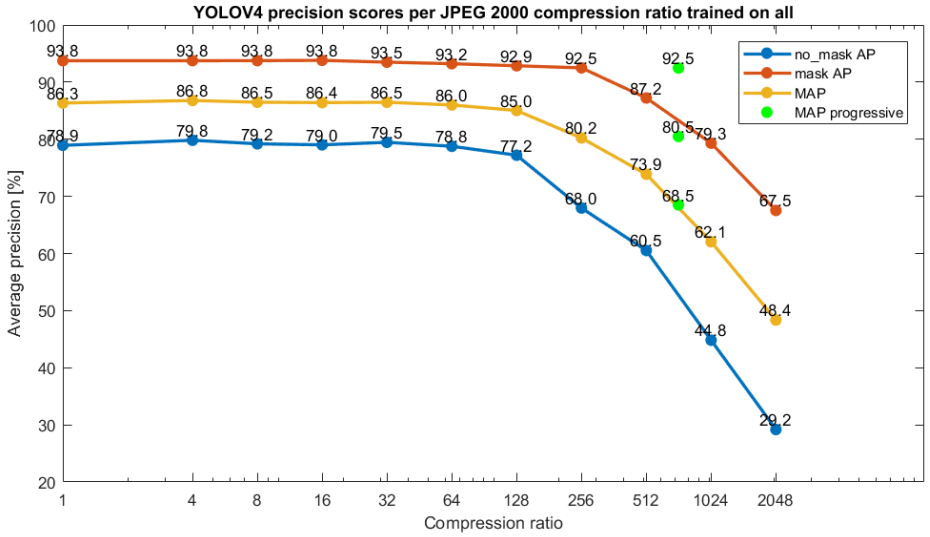

Figure 4.8: Performance for an effective compression ratio of 722.

Interpretation There is a few things to notice in these graphs. Firstly, the three green points have roughly the same performance as for a compression ratio of 256 but at a way higher effective one. Applying such kind of strategy on the training set could improve the compression and computer-vision performances. As this master thesis mainly aims to optimise the compression, this will not be further explored but could be part of future works.

Chapter 5

Experiments using Tiny-YOLOv4 on the UA-Detrac vehicle dataset

5.1 Experimental setup description

The main exploration work of the JPEG2000 parameters will take place in this chapter. To do so, a lighter detection model was used with a new dataset: the Tiny-YOLOv4 model [10] with the UA-DETRAC dataset [9]. This change was made because the previous dataset had some limitations and flaws. As aluded to before, it was really small and did require data augmentation in order to be fully usable. Also the quality of the images was quite heterogeneous; most of the images were of a great quality but not all of them. This new dataset will try to cover those weaknesses and will serve as a comparison to notice any major difference in the behaviour of the performance that could be linked to either the dataset or the task. Tiny-YOLOv4 does perform worse overall but is still quite acceptable: in this case, it has 59% MAP instead of 72 but it will improve once some issues of the dataset are addressed. Using this model did reduce the training time from the previous ∼14-17 hours to a more modest ∼2 hours. To train it, the UA-DETRAC vehicle dataset was used. This dataset contains images shot with a Cannon EOS 550D camera at 25 frames per second of the traffic in Beijing and Tianjin in China (see example in Figure 5.1). The images have a fixed resolution of 960×540 pixels and contain four classes of vehicles: *car*, *bus*, *van* and *other*. The dataset also provides regions to ignore in the form of boxes to cut out stationary vehicles and vehicles that are too far, as both of these cases do not have any detection box nor label associated to them. These "cut-out" boxes are quite useful to help the training process of Tiny-YOLOv4: the area they describe will be removed and replaced with black for the training but not for the compression. This way, this does not give an unfair advantage to the compression, as it would remove some of the information that

needs to be compressed. It is also important to note that no cross validation was used in the following experiments as the dataset contains frames from videos that were sometimes shot by the same camera in different weather or lighting conditions. Therefore, using these images with such a re-sampling technique without making sure that the training folds and validation fold do not have similar images will likely result in an unnatural increase in the performance. Just using the provided training set and test set as-is should not be an issue as they are quite big and complete. Also, it is rather beneficial in terms of reproducibility. Although this dataset has plenty of images to train on, it does have some drawbacks that will be covered and tackled in the next section.

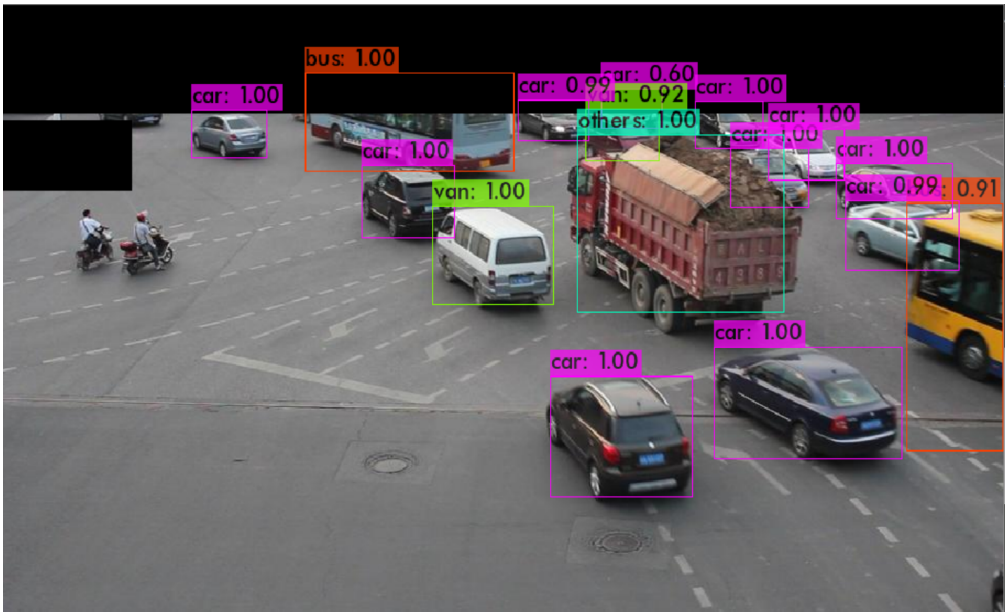

Figure 5.1: Example image of the UA-DETRAC (vehicle) dataset with the detection boxes and labels as well as the removed regions to ignore.

5.2 Dataset issues

Too many images One of the first issues was the excessive number of available images (∼140.000). This is an issue as the dataset needs to be compressed and stored multiple times. Also as the images originate from a video, neighbouring frames are very similar and taking them all will not help much in the training process. To reduce this number, one frame was sampled per ten images, reducing the number of images to 8394 for the training set and 5640 for the test set.
Classes imbalance The main issue of the dataset then comes from a big imbalance in terms of the representation of every class in the data. The *car* class is well

represented (51926 times) but the three other classes not so much (3408, 5925 and 384 see Figure 5.2), which leads to performance imbalance between classes. To tackle this issue, a special parameter in the Tiny-YOLOv4 model was used: *counters_per_class*. This parameter will weigh each class when computing the overall loss. It will optimise each class equally instead of focusing on improving the overall score, sometimes at the cost of poorer performance on the less represented classes. This did help in performance a bit but mainly in stabilising the performance of the less represented classes.

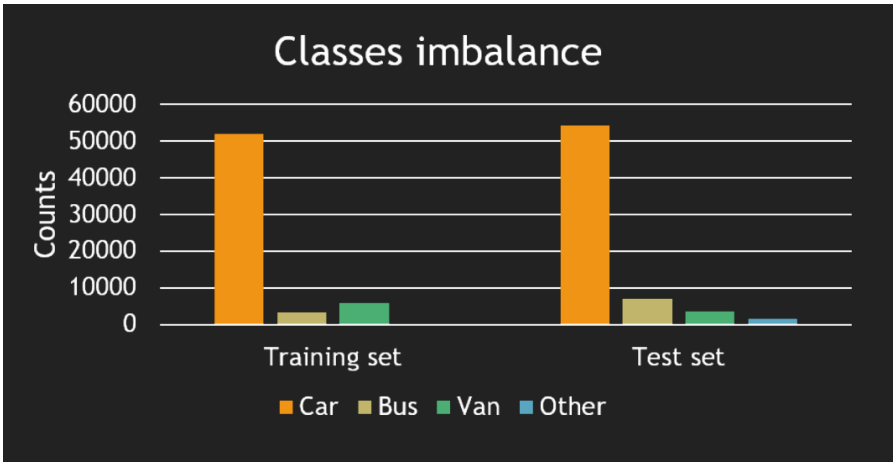

Figure 5.2: Class representation imbalance.

Poorly represented class Another big issue comes from the lack of coverage of the last class; the *other* class. This class is only represented 384 times in the training set (see Figure 5.2 above) and contains a great variety of vehicles, therefore the detection model is unable to grasp the essence of what makes this class and thus its performance is very poor (see Figure 5.3). To solve this issue, this last class was removed from the dataset and the training and performance evaluation were only conducted on the 3 other classes.

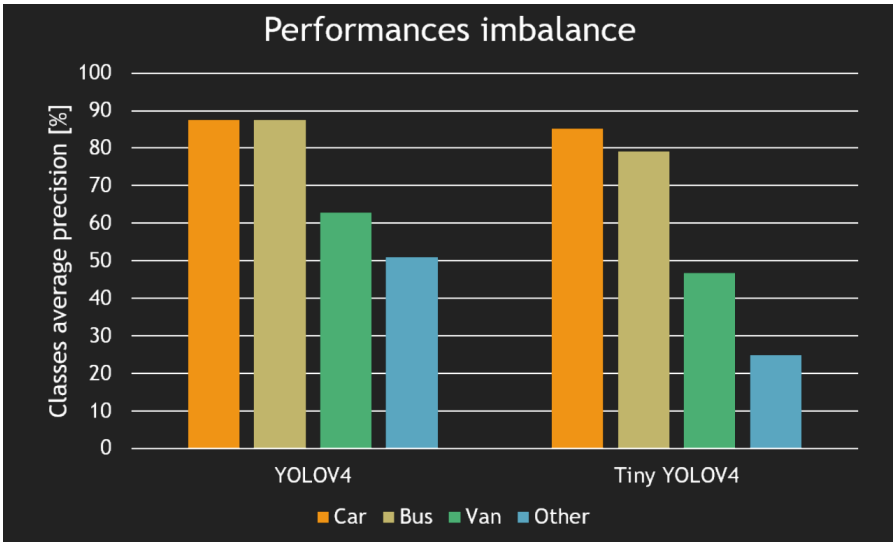

Figure 5.3: Class performance imbalance.

Imperfect regions to ignore The last issue comes from the sometimes poor quality of the provided regions to ignore. Those boxes could sometimes contain detection boxes but would completely hide the object to detect. In those instances, the detection box and associated label were removed from the ground truth using a score and a threshold. The used score is computed as the ratio between the area of the intersection of the black boxes and the detection box and the area of the detection box itself and denotes how covered up the detection is by the black regions.

$$overlap = \frac{ignore \cap detection}{detection}$$

with *ignore* being the box of the region to cut out and *detection* being the box associated to a detection. An overly high ratio (>90%) means that the object is mostly covered and therefore unrecognisable, and should be dropped from the ground truth. This strategy was applied on both training and test set to make sure the results were accurate.

5.3 Baseline performance

As in previous chapter, the first thing to do is to establish the baseline performance of Tiny-YOLOv4 when training it on uncompressed images and to evaluate it on compressed images at different compression ratios (see Figure 5.4 below).

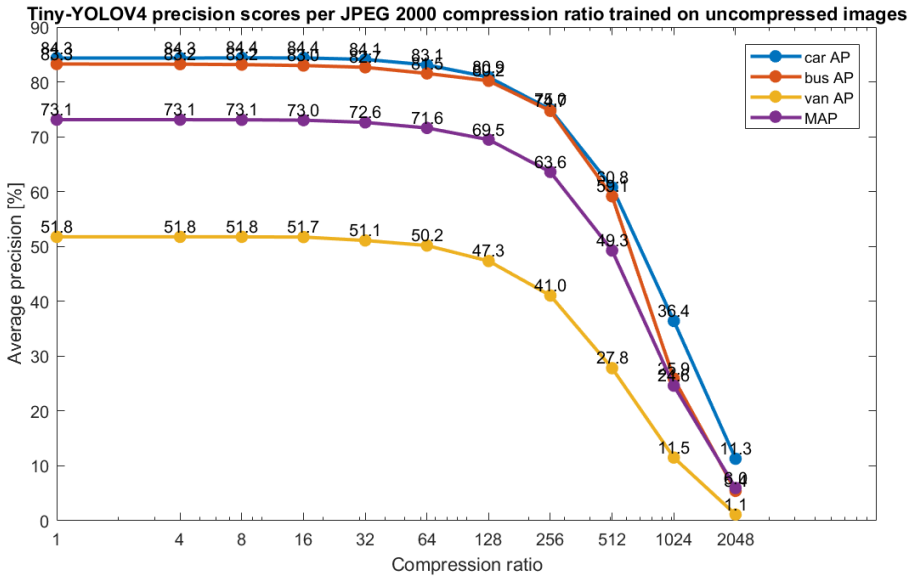

Figure 5.4: Performance when training on uncompressed data and evaluating on data at different compression ratios.

Interpretation These baseline curves do follow quite a similar logic to the ones of previous chapter (Figure 4.4). Before, the performance began to slowly drop at a ratio of 32 and the first big drop appeared at 128. This time, the performance also starts to slowly drop at a ratio of 32 but the first big drop appears at 256. The overall performance is also quite lower than before but this is due to the model and dataset used. The performance of the *van* class is quite low compared to the rest, this might come from the model having difficulties differentiating a van from a car.

5.4 Training/evaluating at the same ratio

This time, as Tiny-YOLOv4 is a much faster model to train, it is now practical to train it and evaluate it for the same compression ratio. Also, to help comparing the performance, a score is computed in all following comparison graphs. This score is the average of the point-wise division between the performance of both curves. It is equal to one if both curves are overall equal, is less than one if the fist curve (1, blue) is worse and greater than one otherwise. Mathematically speaking:

$$score = \sum_{i=0}^{nbr} \frac{MAP_{1,i}}{MAP_{2,i}}$$

with nbr being the number of points that have the same abscissa in both graphs and $MAP_{c,i}$ being the MAP of point i from curve c.
Doing so gives us the following graphs:

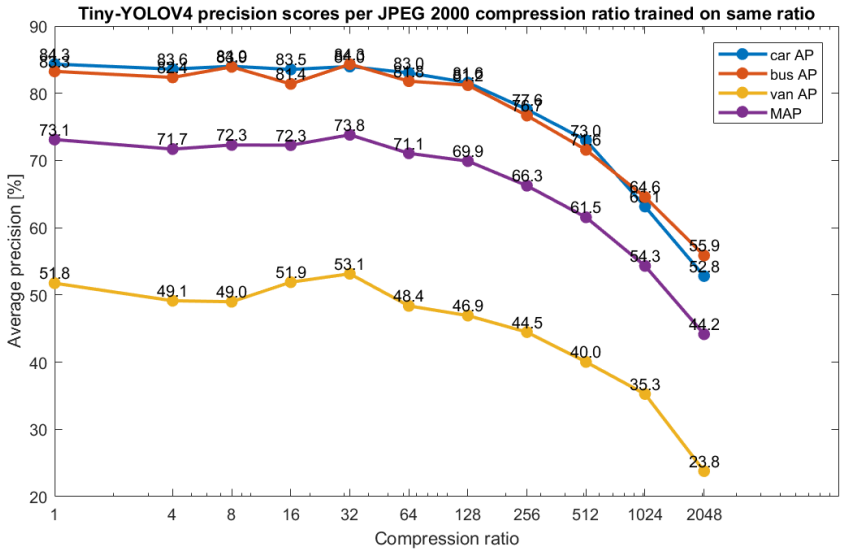

Figure 5.5: Performance when training and evaluating at the same compression ratio.

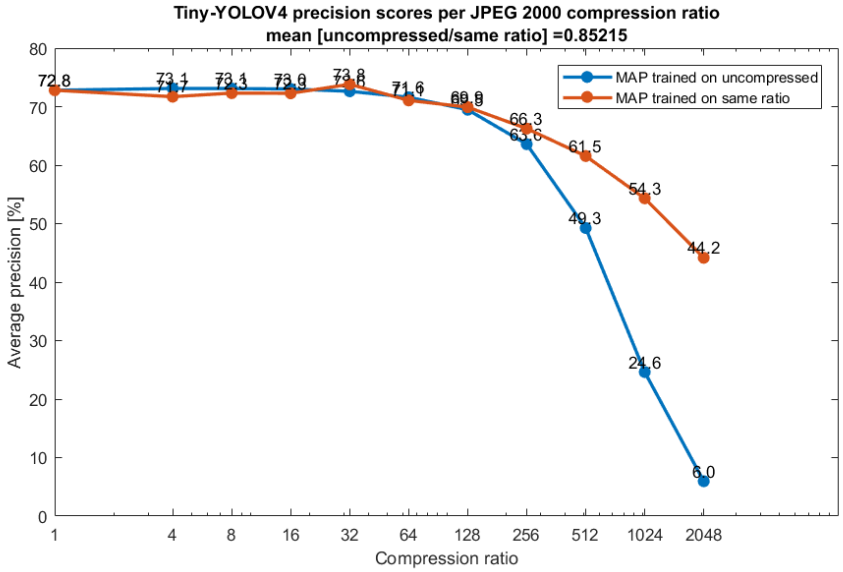

Figure 5.6: Comparison between the MAP obtained from training on purely uncompressed data and data at the same compression ratio as the test one.

Interpretation The first important thing to notice is the increase in run to run randomness. Before, the MAP was not varying much for ratios up to 64; now it is the opposite. Although the *car* MAP does not vary much, the MAP of less represented classes does. This probably comes from the lower coverage from the dataset for these classes with regards to the complexity of the objects to detect. To tackle this randomness issue, the mean of several runs will be used in the future (see details in next subsection).

5.5 Optimising Deep Learning

Using preceding works as inspiration, a similar approach as the two stage training described in [13] (see Section 2.2 of the state-of-the-art review) was developed. This time, the network was first trained on uncompressed data for a smaller amount of epochs and then trained for the remaining amount on compressed data at a specified compression ratio.
The amount of epochs in the pre-training will have an influence on several factors. The first one to consider is the amount of speedup we can achieve, as the only part of the training that is repeated is the fine-tuning on compressed data. Therefore having a bigger amount of pre-training will speed up the experiments even more. Secondly, if too little fine-tuning is done, the network will not correctly adapt to the compression and the performance will be decreasing instead of increasing. Lastly, as the fine-tuning starts from fixed pre-trained weights, the amount of randomness from runs will decrease and will tend to follow a smoother behaviour as seen in the first baseline graph (Figure 5.4).
To choose the right amount of epochs for the pre-training, several quick tests were done with 5000, 6000 and 7000 out of the total 8000 epochs and overall the one giving the best balance between the performance increase, randomness reduction and total training speedup was with 5000 epochs. The details of some metrics can be found in table 5.5 below for a compression ratio of 2048. Training this way noticeably improved the overall performance and helped reduce the total amount of epochs to train on the compressed data, leading to a roughly four-fold speedup. The $\sim$ four times increase in speed comes from the reduced number of epochs in the fine-tuning and the activation of tensor cores (specialised AI cores in the Nvidia RTX 2060 used for training) after the first 2000 epochs that further speeds up the training process. Note that applying this method for increasing the performance of a single run will work but will not bring any speedup and is somewhat impractical to the end user as he would need two copies of the dataset (one uncompressed and one compressed).

pre-training epochs	MAP	fine-tuning time [min]
5000	46.8	∼30
6000	45.2	∼20
7000	39.8	∼10
baseline/0	44.2	∼120

Figure 5.7: Evolution of some metrics for different amounts of pre-training at a compression ratio of 2048.

5.5.1 New baseline curve from optimised DL

To see how this improved training influences the performance at any compression ratio, a new baseline curve was produced by using the average of the MAP on four runs to further lower the run to run randomness.

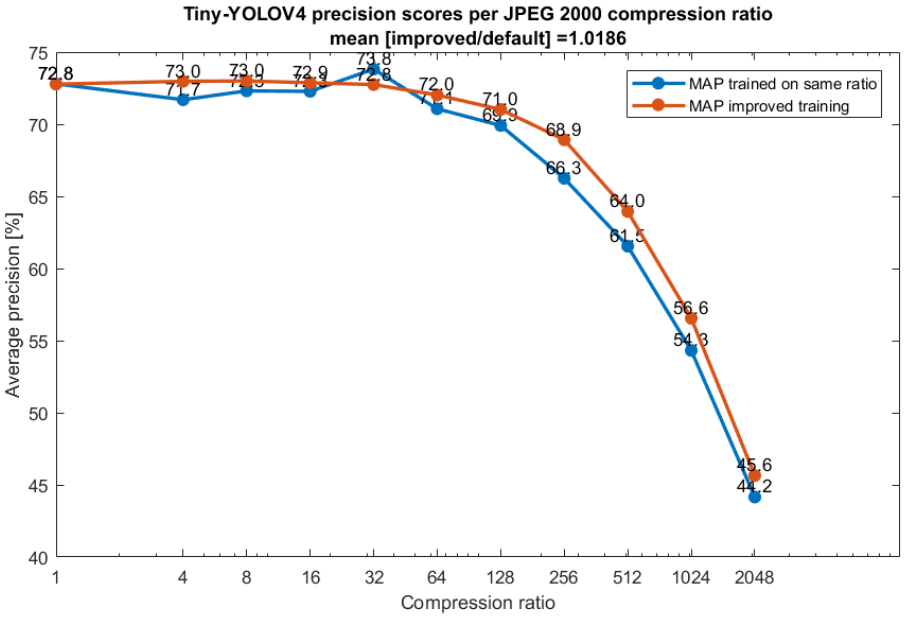

Figure 5.8: Comparison between the baseline performance and the average obtained from four runs of improved training.

Interpretation The new curve brings improvements across the board to the performance, especially at higher compression ratios. The previously obtained low scores at low compression ratios could have stemmed from bad luck from the randomness between runs. The new curve also helps in smoothing out the results: now if a point were to stand out to its neighbours, it would probably be for a reason instead of pure randomness.

5.6 OpenJPEG software

5.6.1 Number of DWT decompositions and reversibility

In OpenJPEG [8], the reversibility is a bundle parameter that influences the choice of the kernel and the colour transform simultaneously. The reversible parameter uses the reversible 5/3 DWT kernel with the RCT. The irreversible one uses the 9/7 DWT kernel and the ICT. This experiment will aim to find an optimal number of DWT decompositions to use with each reversibility parameter. The default number used by OpenJPEG is 6; it will be interesting to see how good it is by default. For the optimisation, the results of four improved runs were averaged for each number of DWT decompositions (spanning from 4 to 9) and for each reversibility parameter at a fixed compression ratio of 512. This fixed ratio was chosen with several criterion in mind. Firstly it is a good compromise between too much and too little effect of the compression. Secondly, it must retain a reasonable amount of performance, as it would otherwise not make any sense to optimise for this value.

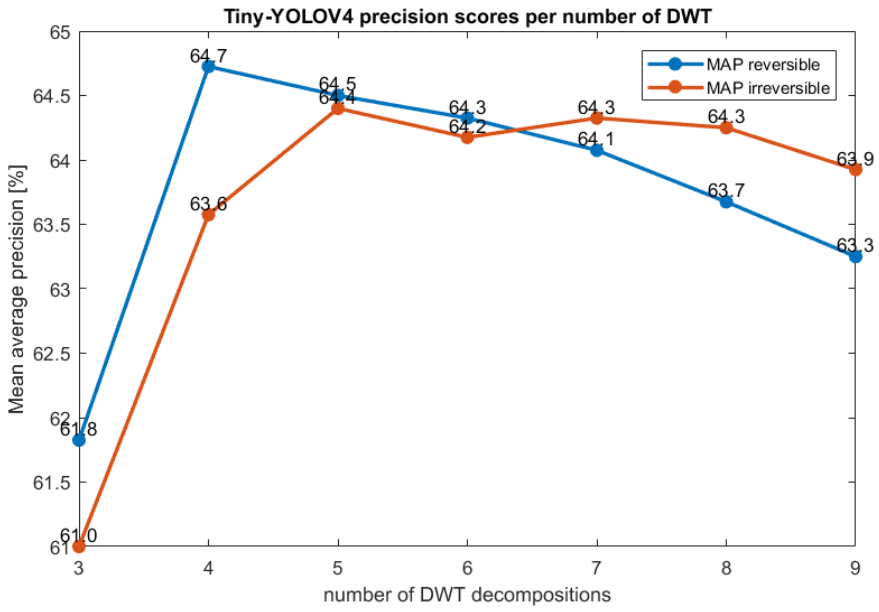

Figure 5.9: Average MAP on 4 runs of improved training for different numbers of DWT decompositions at a compression ratio of 512.

Interpretation Interestingly enough, the number of DWT decompositions seems to influence the reversible or irreversible curve differently. Decreasing it seems to somewhat improve the performance of the reversible parameter up to a threshold

of 4. When going past this threshold, the performance drops tremendously; the reasons for it will be explored later. Modifying this parameter with the irreversible one does not seem to bring any notable improvement nor highlight a behaviour in the curve, except for the minimal threshold that is also present for the reversible parameter. The following figure will aim to compare the appearance of images compressed using the reversible bundle and four or nine decompositions in the DWT.

Figure 5.10: Comparison between images compressed 512 times using 4 (left) or 9 (right) DWT decompositions and the reversible parameter.

Interpretation The main difference between the images is the amount of colour. The right one (9 DWT) has a bigger variety of colours but everything is slightly more blurry. The left image does have very little colours and is mainly in grey level but some details are more readable (e.g. the bus stop sign).

Optimised curve

By taking the best performing set of parameters (reversible bundle and 4 DWT decompositions), the following graph was produced to see if the performance improvement is general or not.

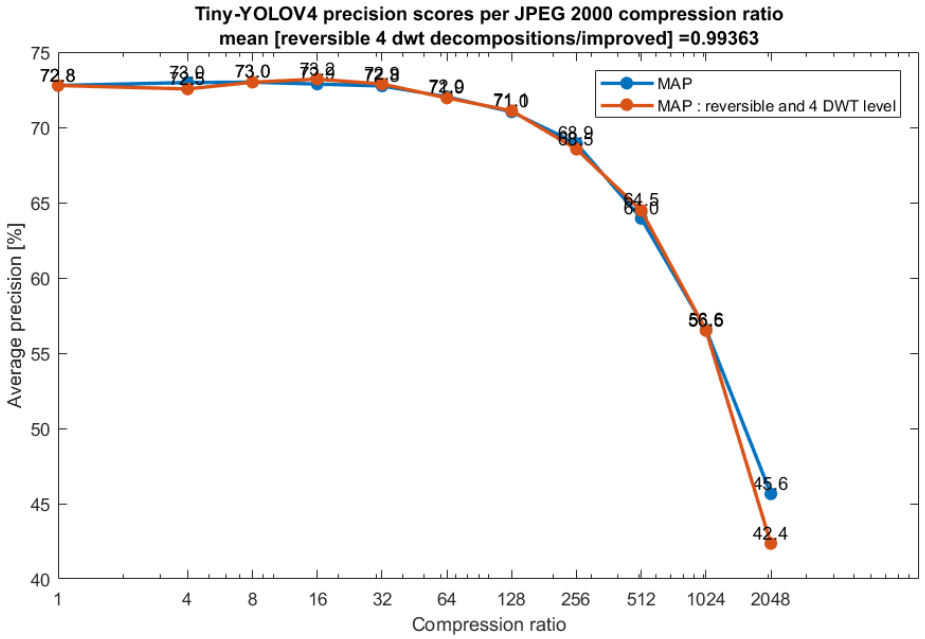

Figure 5.11: Average MAP on 4 runs of improved training with the previously found best number of resolution and reversibility.

Interpretation The first thing to spot is that the performance is mostly on par with the one obtained before (blue curve) at the exception of a ratio of 2048. For this compression ratio, the four DWT decompositions are not enough for the compression to work correctly. There are too few colours to represent the images and details start to disappear,e.g., cars on the road, which in our case is a very big issue (see Figure 5.12). Note that tests were rerun and therefore the performance for a ratio of 512 is actually different than the one in the previous graph (5.9). This minor variation further highlights the need for reducing the randomness between runs. This is why in the next experiments, the mean will be averaged over five runs.

Figure 5.12: Comparison between images compressed 2048 times using 4 (left) or 9 (right) DWT decompositions and the reversible parameter.

5.6.2 Conclusion

The main takeaway from these experiments is that the number of DWT decompositions seems to bring a balance between spatial resolution and colour resolution. When this number is low, the images are mostly black and white with a slight increase in the amount of details. On the other hand, when it is high, the images are mostly coloured and blurry. The optimisation of this parameter seems to bring a minor but noticeable improvement to the performance at a specific compression ratio but the optimal parameter value changes with it.

OpenJPEG parameters limitation OpenJPEG is a great compression software for most users as it does work very well and does not bother you with an over-complex list of parameters to play with. On the other hand, Kakadu software [6] lets the user adjust a lot of things and in some cases even too much as some parameter choices might have a disastrous impact on the compression efficiency and image appearance. As this master thesis aims to explore more advanced parameters and ideas, we will now switch to using Kakadu software for the rest of the experiments.

5.7 Kakadu software

With the swap to Kakadu software comes a lot of changes. Firstly, there is no compression ratio parameter but instead a bitrate that has an equivalent effect, as long as it is chosen in the following manner :

$$bit\_rate = \frac{24}{compression\_ratio}$$

Indeed, 24 bits per sample/pixel (8 bits per colour channel) is the bitrate for an uncompressed image. Therefore, the bitrates were chosen to have a similar effect to the compression ratios from OpenJPEG and are rounded for simplicity to four digits of precision. Please note that from a software to another, there might still be some degree of disparity in the compression performance but it is not the goal of this master thesis to compare them extensively.
Secondly, the amount of runs was increased from 4 to 5 in order to further reduce the effect of the randomness.
Thirdly, exception made for the new baseline curve, all tests on the two highest bitrates (lower compression ratio) were dropped because the effect of the compression in this region was not noticeable and dropping them allows for faster and more numerous testing.

5.7.1 Baseline and pure MSE optimisation

By default Kakadu internally uses what they call a visual weighting instead of a pure MSE optimisation. So a first test is to establish and compare the baseline for Kakadu software with an MSE-optimised one.

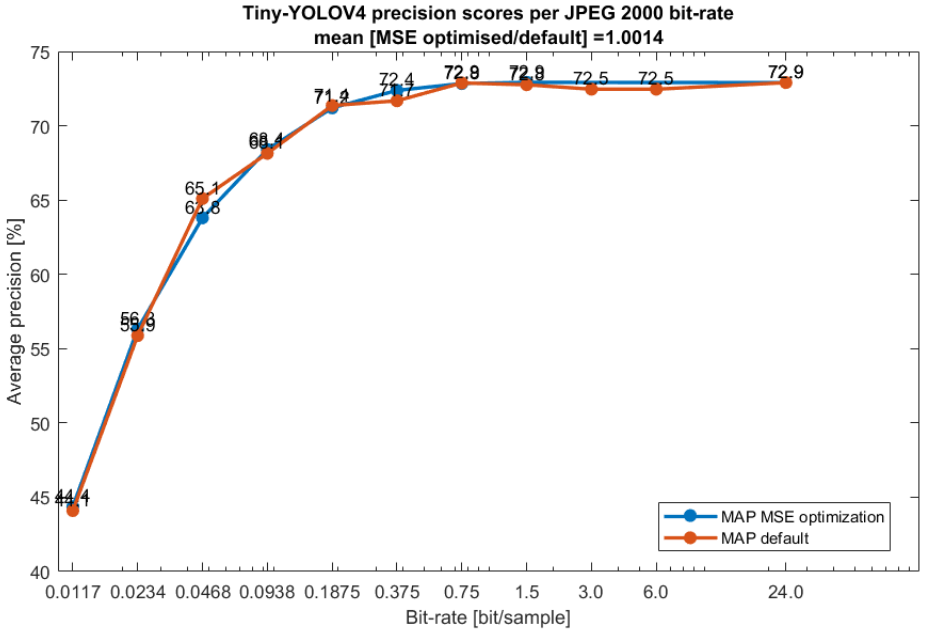

Figure 5.13: Kakadu baseline versus MSE-optimised.

Interpretation A first thing to notice is that the performance is quite close to the one obtained using OpenJPEG and seems to be following exactly the same tendency (compared to Figure 5.8). The only difference comes from two points at a bitrate of 0.0468 (ratio of 512) and 0.375 (ratio of 64). For unknown reasons, the performance of the default parameters is slightly worse than they should be at a bitrate of 0.375, as this is a local and sudden drop compare to the neighbouring points. The opposite phenomenon happened at a bitrate of 0.0468 and in this case, the performance is better than expected (as the point is higher than one that would strictly follow the behaviour of the curve).

5.7.2 DWT resolution levels weighting

With the uses of Kakadu software comes new possibilities, it for example lets the user assign weights to the different resolution levels. Doing so, the user can explicitly give more importance to some resolutions and lessen the effect of some. This experiment will explore the effect of assigning weights in a low-pass, band-pass and high-pass manner. There is by default five resolution levels in Kakadu, organised from the coarsest (1) to the finest (5). We will stick to the default five for this experiment but if the results are promising, it is possible to change this value. The low-pass filter will give more importance to the coarsest resolutions and scale down linearly to the finest (see Figure 5.14 below). The high-pass does the exact opposite and the band-pass gives the same importance to the finest and coarsest resolutions scaling up linearly to the middle one.

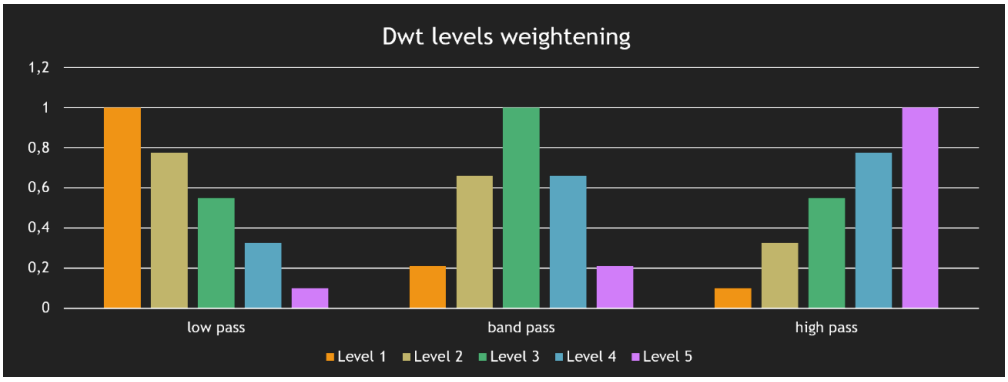

Figure 5.14: Weights used on the resolution levels.

Low-pass

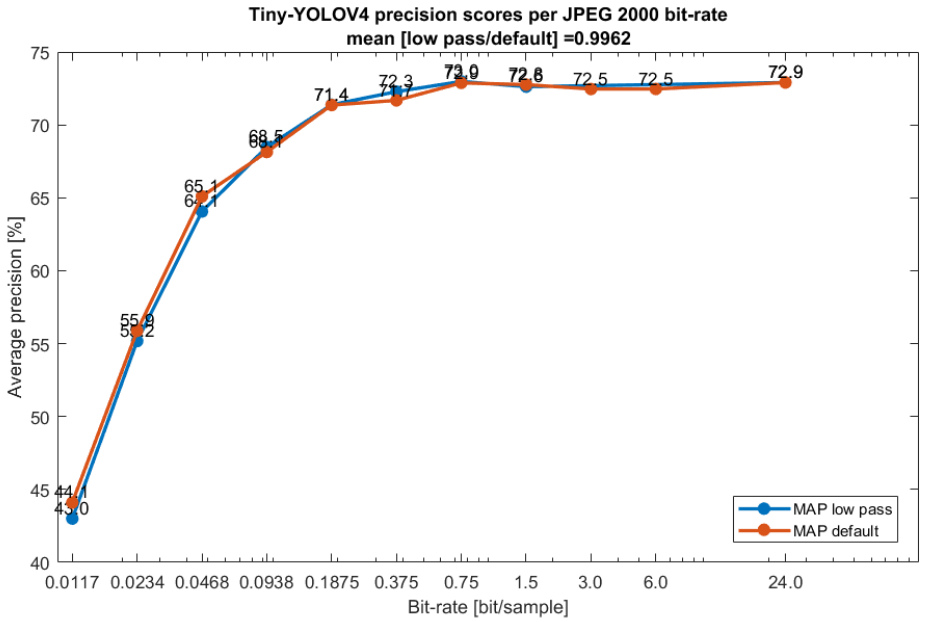

Figure 5.15: Performance of the low-pass weights.

Interpretation Overall, the low-pass weights performance is quite close to the default one but it starts to be a bit worse at lower bitrates. This similarity is probably due to the fact that, by default, the coarsest resolutions are more efficient at describing the image and therefore prioritised. Then the decrease at lower bitrates could come from the loss of useful details, in the form of higher resolution salient points that are not being represented and replaced by an increase in precision (more colours) in the coarsest resolutions.

Band-pass

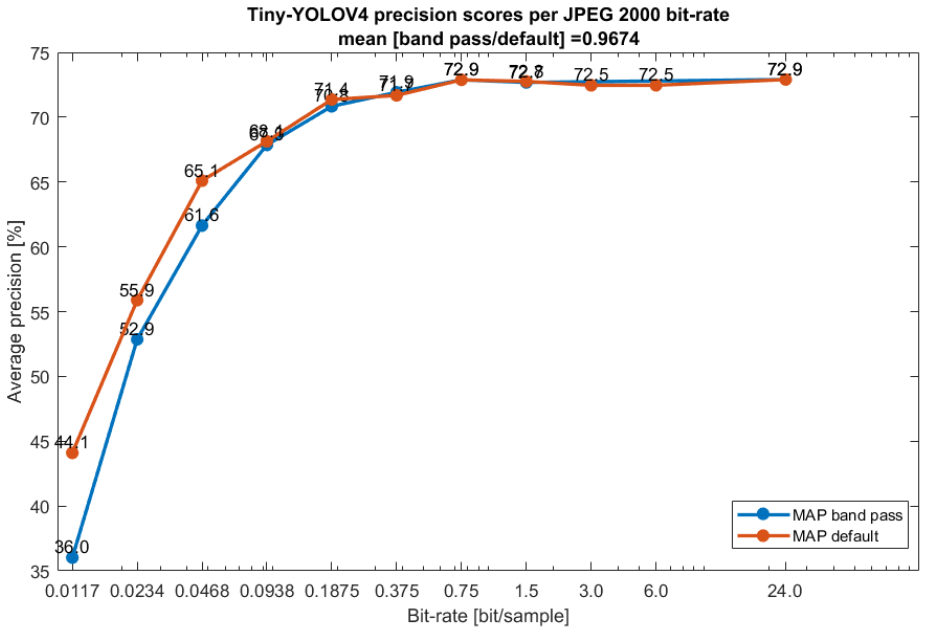

Figure 5.16: Performance of the band-pass weights.

Interpretation For the band-pass graph, the performance does not really differ until a small enough value of bitrate is reached. When reached, the compression is probably much less efficient as it tries to represent images using higher levels of resolutions than it should. Indeed, the coarsest resolutions that are useful to summarise the image at very low bitrates do not have a high enough importance to be able to represent the outline of the objects.

High-pass

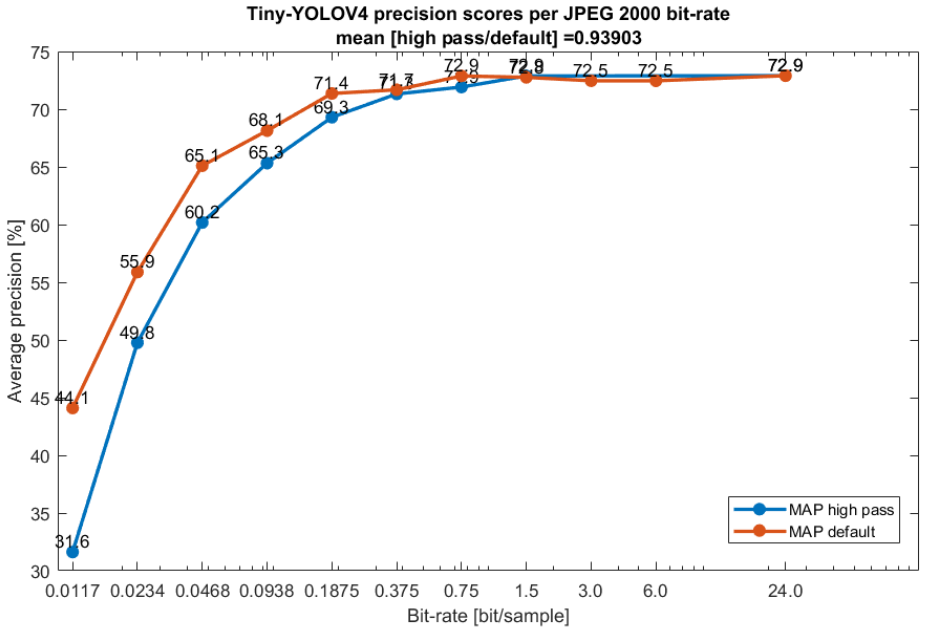

Figure 5.17: Performance of the high-pass weights.

Interpretation The high-pass filter is the worst performing attempt as the compression with this weighting is quickly unable to efficiently represent images when the bitrate heavily restrains the amount of information to keep. In this case, high resolution details that are spread over images and that could be part of the background are prioritised at the cost of an overall blurrier image.

Conclusion

Assigning different weights to the resolution levels did not help in the same manner as lowering the number of DWT decompositions. Here, these filter-like weightings will just assign less importance to some resolutions in exchange for more in some others. Furthermore, this approach does not have a positive impact on the detection performance. In the best case, the performances were alike and in worse cases they dropped tremendously at lower bitrates. This highlights the fact that the detection task probably likes to have a summarising representation of the object in a coarse form. It also does seem to prefer additional information like finer resolution salient points instead of having more colours at the same coarse resolution.

5.7.3 Wavelet kernels

This part will briefly compare the performance of some of the available DWT kernels. Unfortunately, as this option is not well documented, only the most known ones will be used, beginning with the reversible and irreversible kernel bundles as the reversibility and therefore use of colour transform is derived from the kernel. The Haar wavelet (in its irreversible form) was also used to see if the detection would like the unique compression noise that comes with it: the image will form small rectangular regions of the same colour instead of having a slightly blurry representation (see Figure 5.19 below).

Reversible/irreversible wavelet

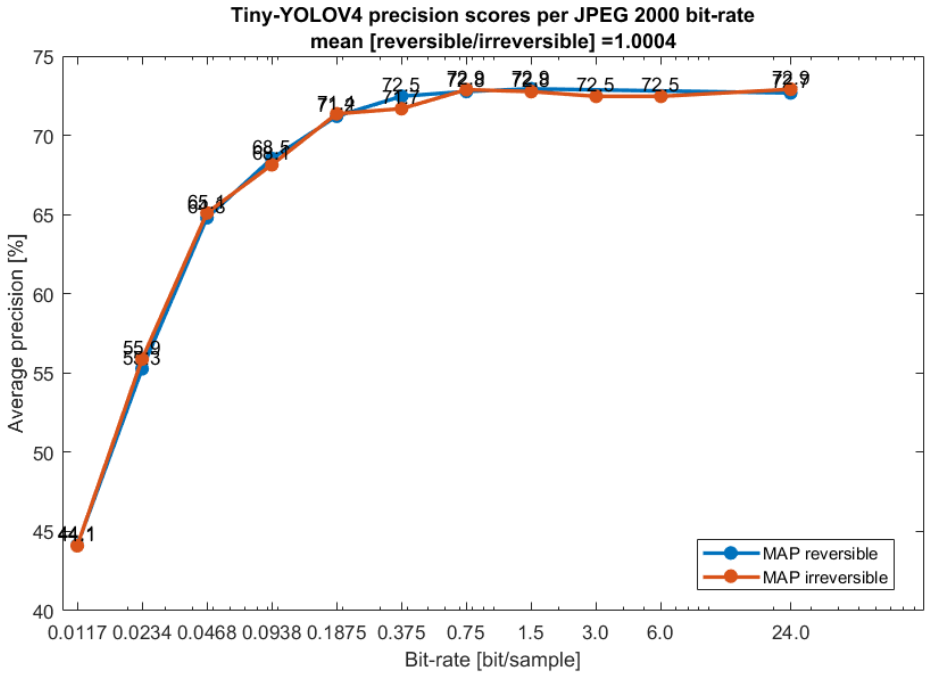

Figure 5.18: Performance comparison between the irreversible (default) and reversible parameters.

Interpretation Overall, the performance is on par, exception made of the experiment at a bitrate of 0.375 were the reversible bundle was better and followed the expected behaviour of the curve.

Haar wavelet

Figure 5.19: Example of an image compressed with the Haar wavelet at a bitrate of 0.1875.

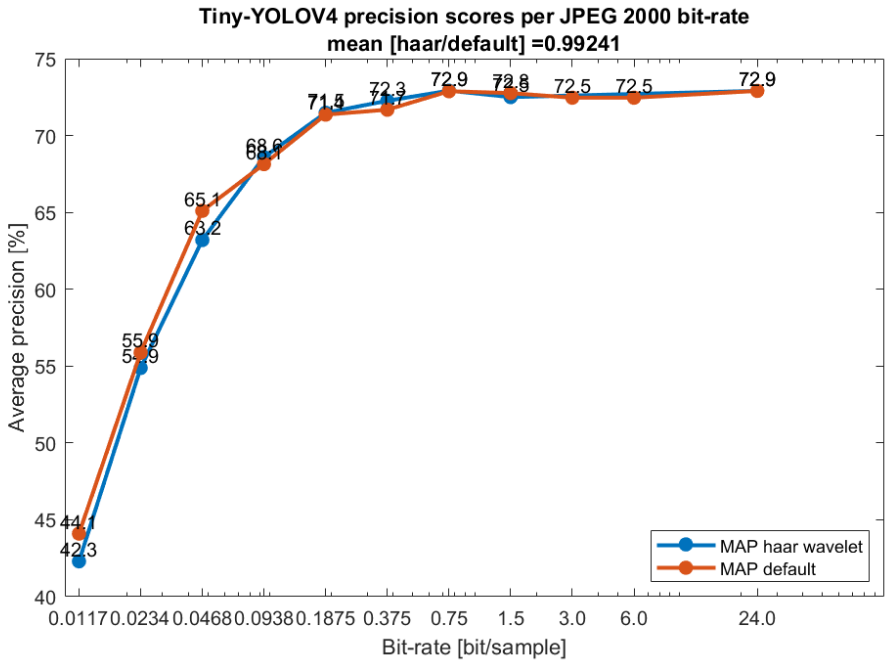

Figure 5.20: Performance of the Haar wavelet

Interpretation The Haar wavelet does worse at low bitrates where images lose a lot of details by being described as big coloured rectangles but was overall equivalent elsewhere. It seems that the detection does not prefer the form of the noise induced by this kernel and prefers the one induced by the conventional kernels of JPEG2000.

5.7.4 Colour transform

RGB versus YCbCr

The first experiment will aim to establish the individual performance of each colour channel present in the RGB (usual) and YCbCr (ICT) representations.

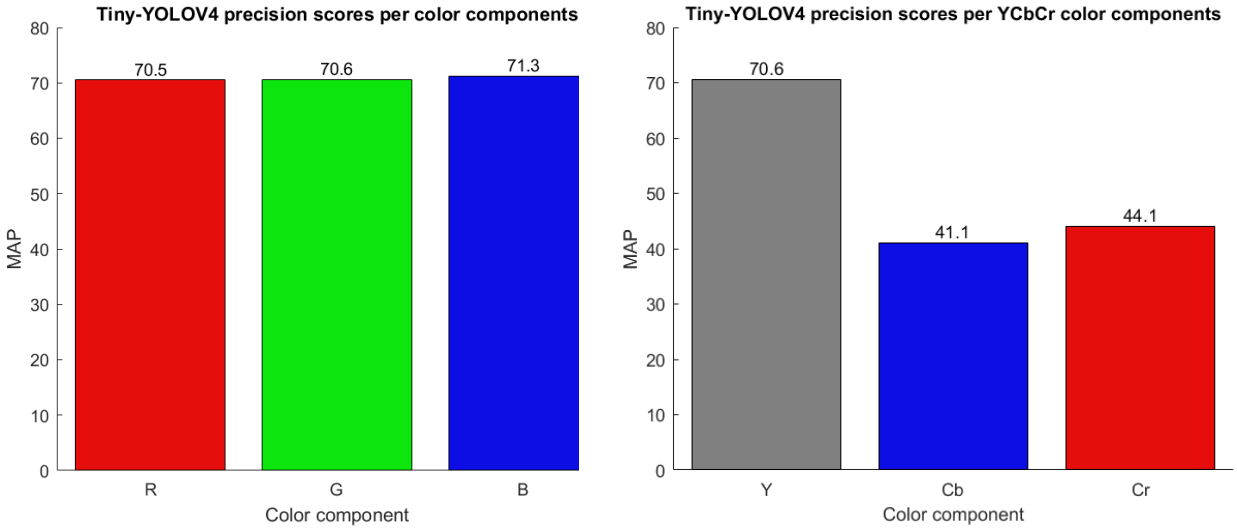

Figure 5.21: Comparison between the performance of Tiny-YOLOv4 on uncompressed single component images.

Interpretation It seems that the blue channel (B) might be slightly better at representing and discriminating the objects to detect but overall the performance is similar for every colour channel. To verify if this holds true at every bitrate, another graph was produced (see Figure 5.22).
The reasons for the poor performance of the chrominance channels are twofold. First, the majority of the details that are shared between the R, G and B channels are this time concentrated into the Y channel and therefore shapes are less recognisable. Second, those chrominance channels might also suffer from chroma subsampling, further diminishing the amount of information available in them.

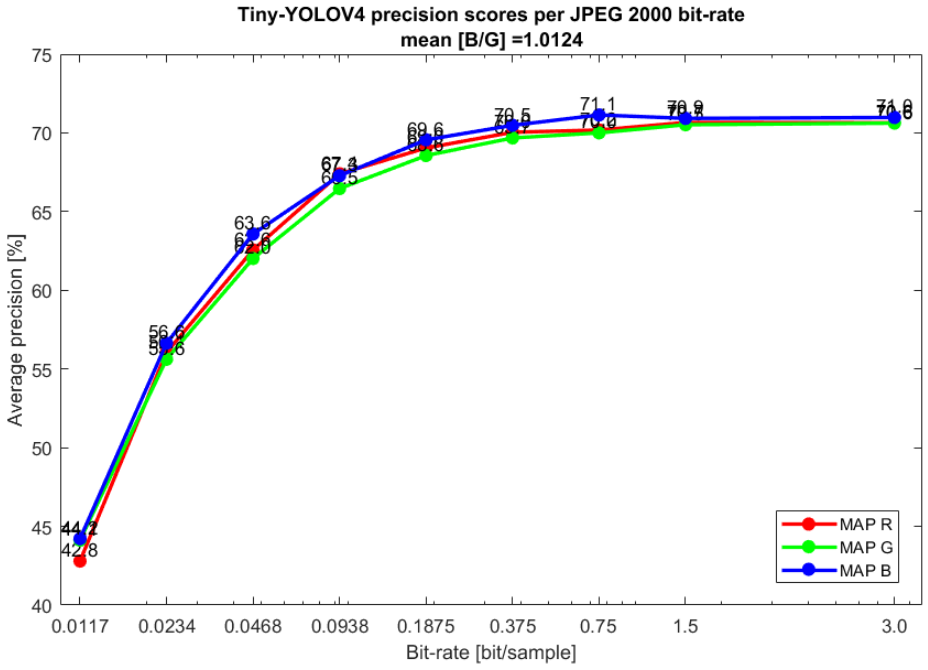

Figure 5.22: Performance of each individual colour component.

Interpretation The first thing to notice is that the blue channel seems overall superior at all bitrates with the green channel being overall the worse. The red channel performance is fluctuating a lot but remains overall worse than the blue and slightly better than the green.

Single Y channel

The following experiments will aim to compare the different colour transforms using their resulting Y grey level channel. In the table below are written the coefficients for each colour transform to produce the Y representation from the RGB one.

colour transform	R	G	B
mean	0.3333	0.3333	0.3334
RCT	0.25	0.5	0.25
ICT	0.299	0.587	0.114
MRCT	0.25	0.25	0.5

Figure 5.23: Coefficients used in each colour channel for every colour transform.

Single channel Grey versus 3 channels RGB The first thing to do is to compare how the performance is affected by the swap from the three RGB channels images to the single Y channel using the mean coefficients.

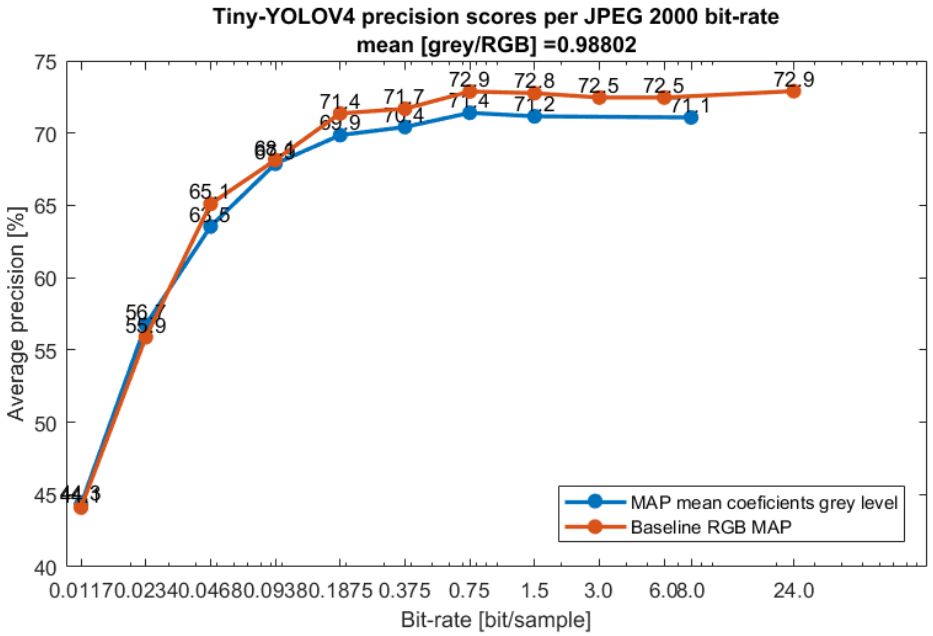

Figure 5.24: Performance of the mean coefficients Y channel.

Interpretation We can see that the MAP is worse across the board but becomes on par if not a bit better at very low bitrates. This can be explained as at very low bitrates the images are almost completely grey and therefore do not differ much anymore. This experiment also highlights the importance of colours for the detection task as the three channel images yield the best results overall.

RCT

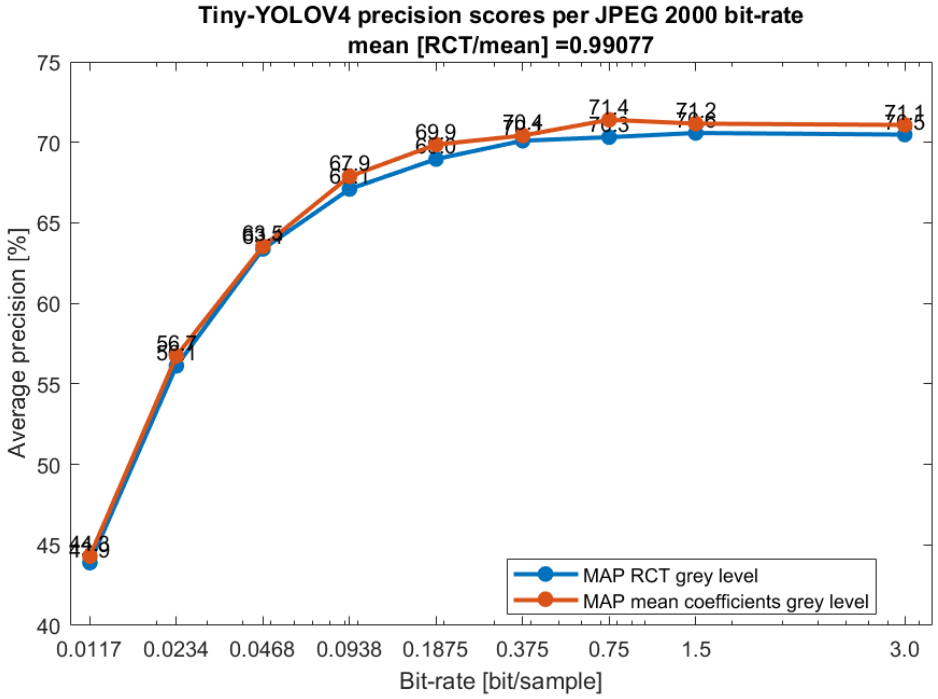

Figure 5.25: Performance of the RCT coefficients Y channel.

Interpretation Interestingly enough, the Y channel of the Reversible Colour Transform is performing rather poorly compared to the one from the mean coefficients. Unfortunately no satisfying explanation of this behaviour could be found. As the decrease in MAP is general, either the compression is performing worse or the detection task is having more difficulties with the generated compressed images. However, to the eye, the images were not noticeably worse.

ICT

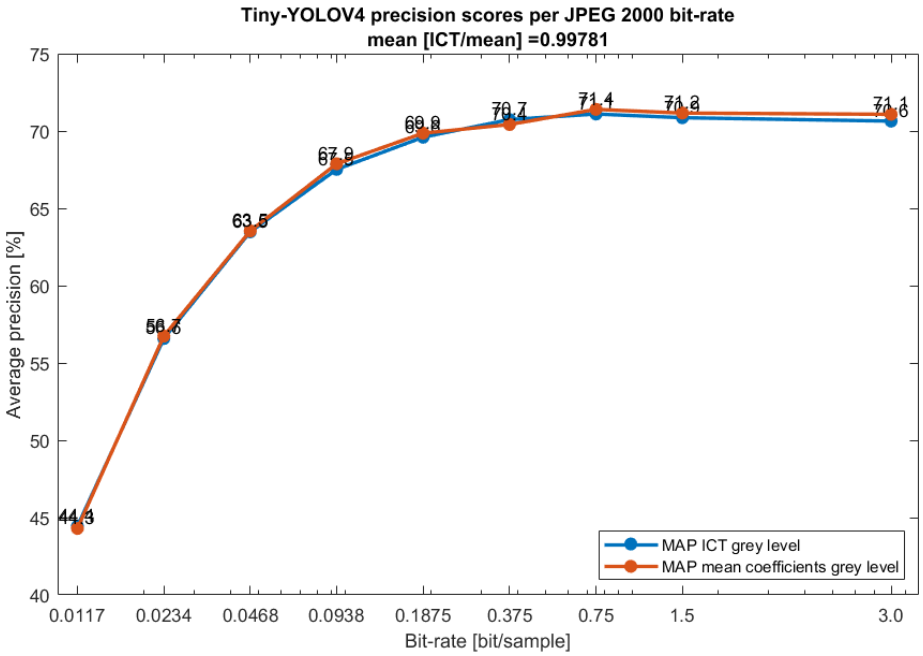

Figure 5.26: Performance of the ICT coefficients Y channel.

Interpretation The Irreversible Colour Transform Y produced channel was similar to the mean generated one if not maybe for a small decrease in the performance. This decrease however could very well come from run to run variations.

Modified Reversible Colour Transform As the blue channel was the best performing one and because the green and red ones were mostly on par, new coefficients were tested in order to produce the Y channel. Those coefficients assign a bigger importance to the blue channel and an equal importance to the red and green ones (see table 5.23 above).

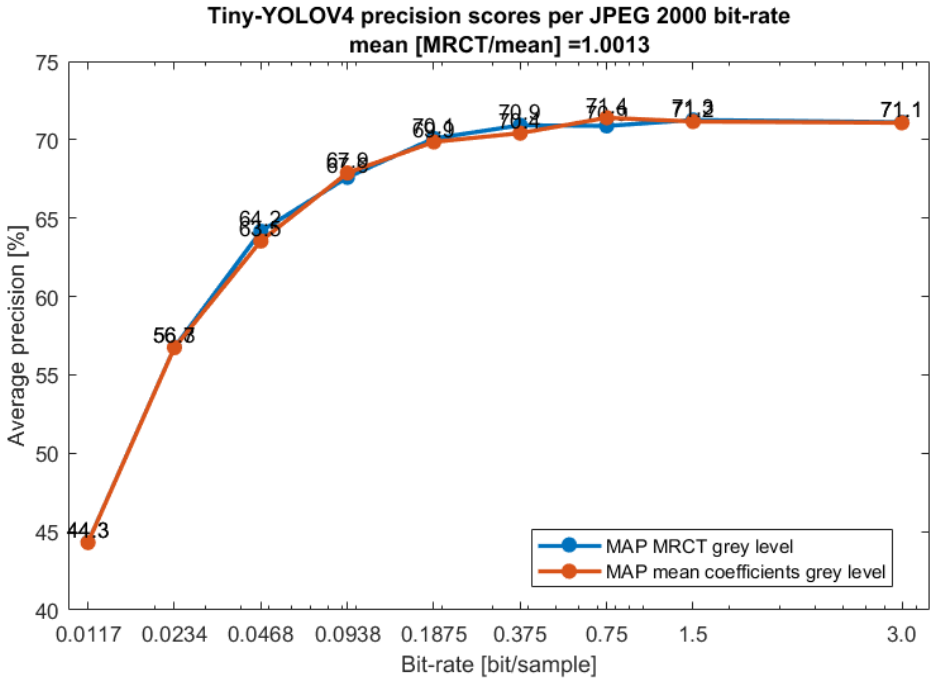

Figure 5.27: Performance of the modified RCT coefficients Y channel.

Interpretation Overall, this set of coefficients is the best performing one of the bunch but not by a great enough margin to call this a definitive win over the mean coefficients.

Best Y transform versus B channel This experiment will aim to compare the results of the best Y colour transform to the ones obtained by using only the blue channel. This will help highlight whether the detection task overall prefers a well-built Y transform, bringing information from all colour channels, or just the best performing one.

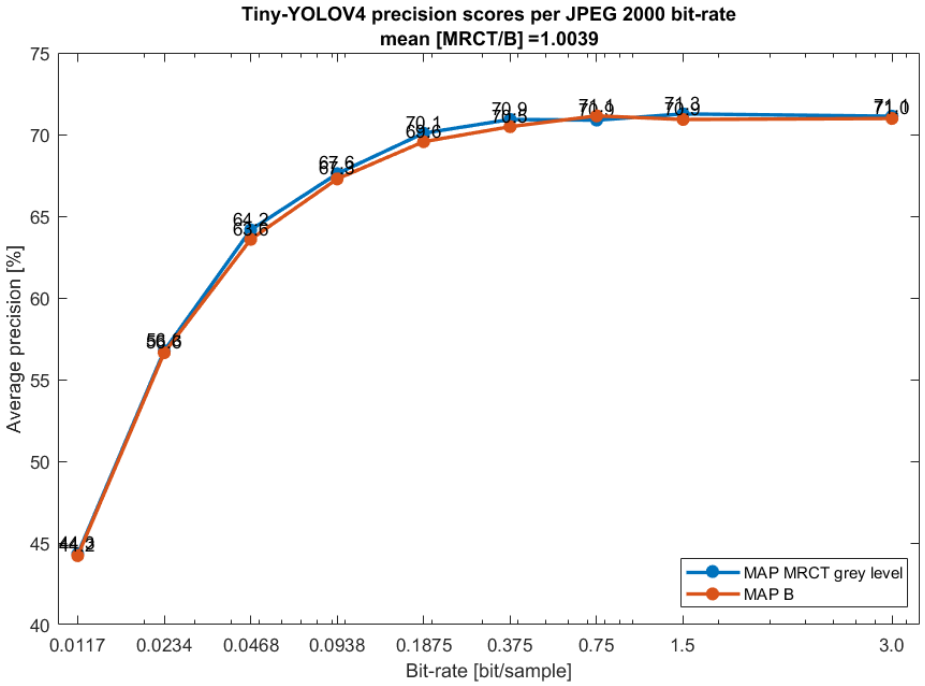

Figure 5.28: Performance of each individual colour component.

Interpretation It seems that the better performing Y channel brings small improvements to the MAP across all bitrates but really small ones. At very small bitrates, the performance merges. On the other hand, at high levels of bitrates, the performance is indiscernible due to the randomness between runs.

5.7.5 Cropped images

Removing the black boxes before compression This dataset is a bit special as it does give regions to ignore for the training. In previous experiments, those regions were cut out and replaced by black pixels after the compression so that it did not ease its task. But what would have been the performance if this operation was done before the compression instead ? This experiment will aim to answer this question and explore the possibility of removing useless or potentially harmful parts of the image for the training of the detection model.

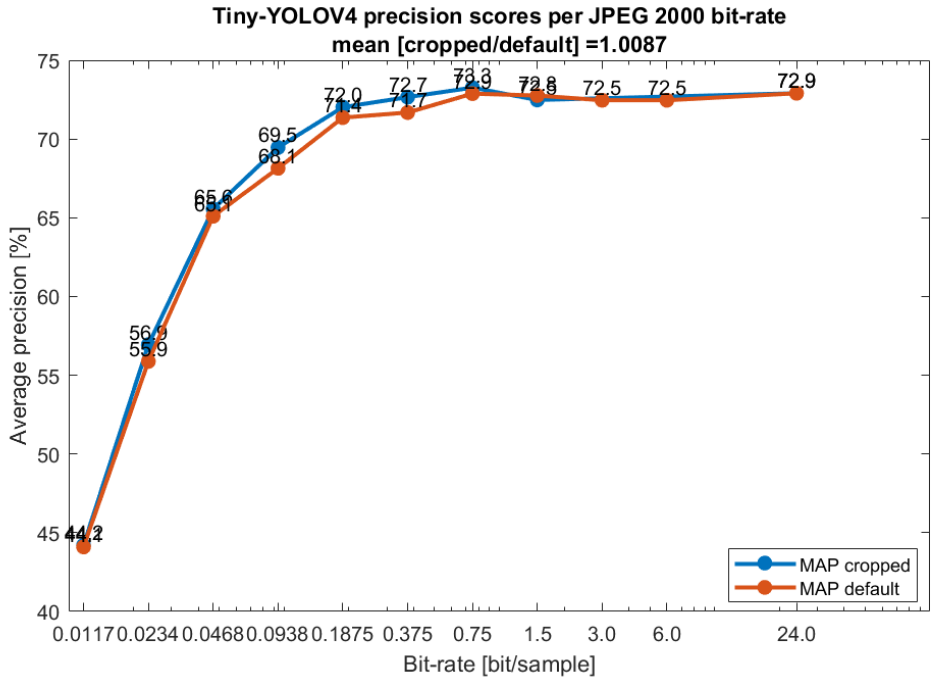

Figure 5.29: Performance on the cropped dataset.

Interpretation As expected, the overall performance did increase, especially for intermediate values of the bitrate. This behaviour can be explained by the fact that before reaching a low enough bitrate, the compression was not constrained enough by it and therefore was not degrading the images much. When this threshold is reached, the images with the black cutouts are advantaged as they will be able to retain more details in other regions of the image at the cost of degrading the already useless black ones. At really low values of the bitrate, however, the compression will still need to remove a lot of details in useful regions, and the small added amount that is not removed does not seem to be enough to help the detection model to better discern the objects.

Chapter 6

Limitations and future works

6.1 Limitations

6.1.1 Dataset

Finding a quality dataset It is generally quite difficult to find datasets that satisfy a lot of the following criterion at the same time. Although the datasets used in this master thesis were good to some extent, they all had issues that needed to be addressed. In this case, the ideal dataset would have been one containing a generous enough amount of dissimilar data, so that the Deep Learning model could have the best possible inner representation of every class and thus would have been able to generalise well. Also for the experiments that were conducted here, an uncompressed dataset with no chroma subsampling or any other degrading operations would have helped to better highlight the effect of each compression parameter on the DL performance.

6.1.2 Compression optimisation

Randomness between runs A pure pairwise optimisation (as illustrated in Figure 6.1) below) is not doable because of the amount of run to run variation in the performance of the DL model. Indeed, it is quite hard to evaluate the impact of a compression parameter, as tweaking it even in the right direction does not guarantee an increase in the performance. A way to tackle this randomness issue is by averaging several tests but even if it does indeed lower the randomness of the output, averaging cannot completely erase this randomness and it does slow down the optimisation process tremendously. Also, as noticed in some experiments before, some compression parameters do not have the same effect for all compression ratios, so this process would need to be reproduced several times. The performance of the computer-vision task could be enhanced by optimising the compression differently,

without actively relying on the optimisation of the compression parameters themselves, which could therefore be used as an additional measure. Examples of such optimisations will be explored in the following section.

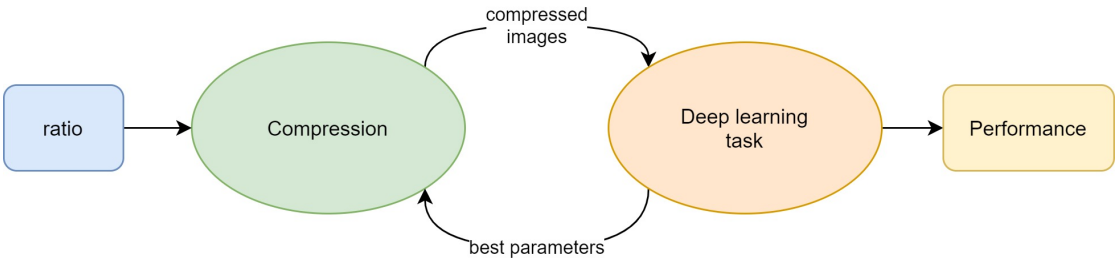

Figure 6.1: Pairwise optimisation of the compression and a DL task at a fixed compression ratio.

6.2 Future works

6.2.1 Heterogeneous compression for dataset of images

As seen in the progressive decompression experiment (see Section 4.5), some images can be compressed more than others while remaining entirely useful to the DL model. This is especially true for datasets containing heterogeneous resolutions, as the loss in information from the compression will be different for low-resolution and high-resolution images. This is why I hereby propose several possible algorithms based on the progressive decompression experiment that will this time focus on compressing the training set but could be adapted to the test set as well.

Achieving uncompressed-like levels of performance One way to optimise the compression of the dataset could be by using a similar approach to that in the k-fold cross-validation. The idea is to cut the training set into n equal subdivisions and then apply a similar approach as the one explored in Section 4.5. This is done by training a model on n-1 subdivisions, then evaluate how compressed each remaining image should be by computing either the confidence or the MAP on them and using a minimum threshold for those values (see Figure 6.2 below). If the threshold is well-chosen, the performance should be quite close to the one in the uncompressed case, because it ensures that the loss of information in the images is not too harmful to the performance of the computer-vision task.

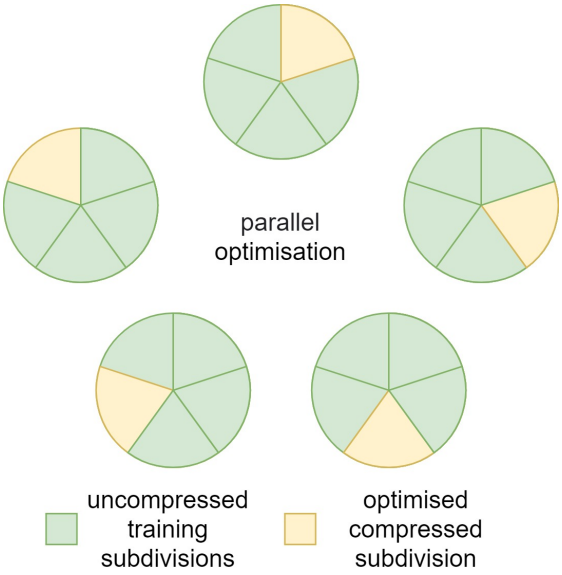

Figure 6.2: Heterogeneous compression in a cross-validation-like manner.

Achieving the highest effective compression ratio Another way to proceed is by iteratively optimising each subdivision (see results of each iteration in Figure 6.3 below). In the first iteration, only uncompressed subdivisions are used to optimise the compression of the first one. In the next iterations, the newly compressed subdivision is used with the n-2 others in the optimisation, and so on. As the effective compression ratio of a subdivision is dependent on how well the model trained using all other subdivisions is accustomed to the compression, it is possible to re-optimise every subdivision several times by trying to lower the compression ratio of every image in it. This is why continuing the iterations should slowly increase the effective compression ratios of subdivisions (see iteration 5 and 10). Some stopping conditions could be either a fixed threshold of iterations or when no or too little ratios changes after a full re-optimisation tour of the training set.

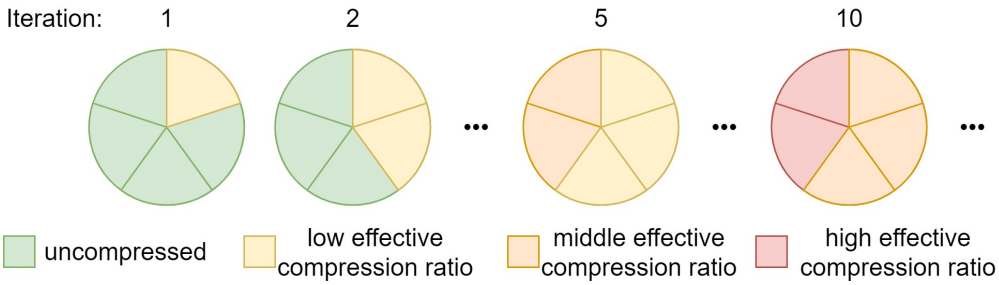

Figure 6.3: Iterative heterogeneous compression of the training set.

Potential issues This method could potentially work quite well but one must pay attention to some requirements when planning on using such an approach. Firstly, for the same reasons that no cross validation was used to explore the parameters in this master thesis, the dataset must not contain images too similar to each other in different subdivisions, e.g., images from neighbouring frames, images and their data-augmented variant, etc. For the same reasons that we do not evaluate the performance of a model on its training data, having images in the test fold overly similar to the ones in the training folds will result in an unnatural improvement to the results and to the effective compression ratio. Therefore the performance obtained when using the resulting compressed dataset could be much worse.

6.2.2 Segmentation mask for Region Of Interest

One of the strong points of Kakadu software is that it allows for the use of a segmentation mask to separate the foreground from the background in the ROI-based compression. It is thus possible to achieve overall higher compression rates but the computer-vision performance is then dependent on the quality of the segmentation. Indeed, using as hypothesis that the selected region is part of the foreground, a type one error will be the worst outcome as the compression will heavily compress important regions in the image. A type two error, on the other hand, is not as severe because the compression will give more importance to some potentially useless regions and thus lower the overall quality. Strategies to compute such segmentation masks are multiple: it is possible for example to extract moving parts in a video and tag them as foreground. Nowadays, there also exists DL driven segmentation models with good performance that could be a good fit. An example of block diagram incorporating this idea is available in the Figure (6.4) below.

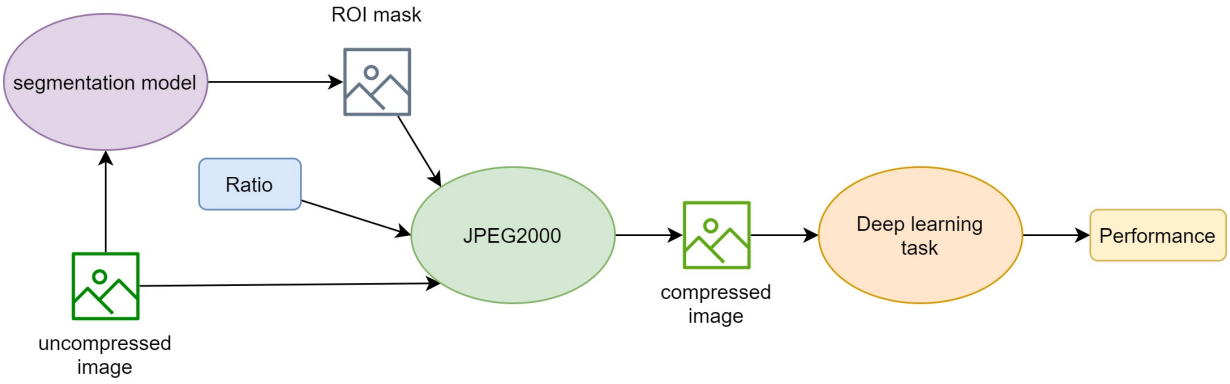

Figure 6.4: Example block diagram using a segmentation model to produce the ROI mask.

6.2.3 SR-like JPEG2000 decoder

By training a neural network and making it learn the JPEG2000 compression noise from the wavelet representation, it could be possible to recreate details lost in the quantisation process and therefore perceptually improve the quality of the decoded image. Such details could be colour gradients that are now represented as banded colours, sharp edges that went soft and blurry, missing higher resolution textures, etc. Nowadays, there exists GAN-based super resolution techniques that are able to recreate artificial details; it could be possible to modify them in order to work with the wavelet representation. Even though this idea focuses on the perceptual side of things it is not impossible that such improvements might have a positive influence on computer-vision tasks. The main advantage of this approach is that it could work with any standard JPEG2000 encoder and would not require any specific changes. The development of a decoder using this strategy could be the subject for future research.

Chapter 7

Conclusion

The main focus of this master thesis was to explore the plethora of possibilities that comes with the JPEG2000 compression scheme, evaluate their impact on computer-vision tasks and tweak some of them in order to keep the best possible performance while being able to compress as much as possible.
We carried out experiments on some of the most promising compression parameters. Those experiments were run on two bundles containing a dataset and a DL model each, in order to spot any difference that could stand out. Even though tweaking most parameters did not yield noticeable improvements, it paved the way to future works in this field and it helped bring some behaviours to light. Indeed, only very few parameters were explored and it would be interesting to see how they would affect each other when varying several at a time.
There are also other possible measures that could be used in addition to the parameters optimisation. One could exploit prior knowledge on the images in order to optimise the compression; for example, a segmentation mask could serve as a Region Of Interest in JPEG2000. Another measure could aim to find how to apply the compression heterogeneously on each individual image, as the information loss will differ and could be more or less harmful to the performance of the computer-vision task.
The joint optimisation of the compression and computer-vision tasks is still rather unexplored and is very promising as it could have a tangible impact on how datasets are stored and used.

Appendices

Appendix A

Progressive decompression pseudo code

Algorithm 1: Joint Progressive decompression.

```
input  : Dataset
output : Found_ratios
    /* Iteratively improve images until the predictions are sufficiently
       certain and return the associated ratios                          */
1   Found_ratios = List()
2   ratio = 2048
3   remaining_images = Dataset[ratio, :]
4   while !remaining_images.empty() do
5       results = YOLOv4.infer(remaining_images)
6       for image in remaining_images do
7           satisfactory = !results(image).detections.empty()
                                                  // at least one detection
8           for detection in results(image).detections do
9               if detection.confidence < 90% then
                                      // all detections are sufficiently certain
10                  satisfactory = False
11              end
12          end
13          if satisfactory then
14              remaining_images.remove(image)
15              Found_ratios.append(Tuple(image, ratio))      // save the
                 ratio found for the image
16          end
17      end
18      ratio = ratio/2 if ratio!= 4 else 1
19      remaining_images = Dataset[ratio, remaining_images]  // update
         the remaining images to less compressed version of themselves
20  end
21  return Found_ratios
```

Bibliography

[1] Cifar-10 dataset. https://www.cs.toronto.edu/~kriz/cifar.html. Accessed: 2021-05-21.

[2] Everything you always wanted to know about jpeg 2000. https://fr.intopix.com/Ressources/WPs_and_Sc_Pub/intoPIX%20-%20Pocket%20book%20about%20JPEG%202000.pdf. Accessed: 2021-05-21.

[3] Face mask dataset. https://www.kaggle.com/aditya276/face-mask-dataset-yolo-format. Accessed: 2021-05-21.

[4] imgaug data augmentation library. https://github.com/aleju/imgaug. Accessed: 2021-05-21.

[5] The jpeg 2000 standard (iso/iec 15444-1). https://w3.ual.es/~vruiz/Docencia/Apuntes/Coding/Image/04-JPEG2000/index.html. Accessed: 2021-05-21.

[6] Kakadu software. https://kakadusoftware.com/. Accessed: 2021-05-21.

[7] Ms-coco dataset. https://cocodataset.org/#home. Accessed: 2021-06-5.

[8] Openjpeg library. https://github.com/uclouvain/openjpeg. Accessed: 2021-05-21.

[9] Ua-detrac benchmark suite. http://detrac-db.rit.albany.edu/. Accessed: 2021-05-21.

[10] Yolo v4, v3 and v2 for windows and linux. https://github.com/AlexeyAB/darknet. Accessed: 2021-05-21.

[11] Yuv, ycbcr and rgb. http://www.latelierducable.com/tv-televiseur/yuv-420-ycbcr-422-rgb-444-cest-quoi-le-chroma-subsampling/. Accessed: 2021-05-21.

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

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

[14] Martin Fockedey and Benoît Macq. Evaluation and optimization of image compression for convolutional neural network in segmentation and classification. Master's thesis, UCL, 2020.

[15] 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.

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

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