UCLouvain

epl

École polytechnique de Louvain

Evaluation and optimization of image compression for Convolutional Neural Network in segmentation and classification

Author: **Martin Fockedey**
Supervisor: **Benoît Macq**
Readers: **Eliott Brion, Antonin Descampe, Christophe De Vleeschouwer, Karim El Khoury**
Academic year 2019–2020
Master [120] in Electrical Engineering

Abstract

Introduction The cloud computing for computer vision is in expansion but the use of videos and images for these applications will need an important bandwidth of the network. In order to reduce the bandwidth and the memory used by this videos and images, the use of standard codecs is privileged but these codecs were optimized for the Human Visual System and therefore can be re-optimized for computer vision applications. Furthermore, the neural networks are usually trained and tested with full quality images but the use of lossy compressed images would diminish the bandwidth require. It is therefore interesting to evaluate the impact of the use of these images and to optimize the networks for these images.

Material and methods To evaluate these different approaches we tackle two computer vision problems resolved by deep convolutional neural networks: the multi-organ segmentation of the male pelvic region on CT and CBCT images and the classification of the CIFAR-10 dataset. For the multi-organ segmentation, we first assess the 3D and 2D U-net robustness against JPEG2000 compression before optimizing the networks for various compression ratios by retraining them with compressed images. For the CIFAR-10 classification, we train a standard network with uncompressed images before optimizing the JPEG quantization tables to maintain for each rate the best classification accuracy available. To do so, we use two different algorithms: a greedy one related to the Lagrangian technique and a handcrafted genetic algorithm.

Results For the first task, we find that the U-net multi-organ segmentation robustness to JPEG2000 can be improved by selecting the right network and by fine-tuning the network weights with a new training with compressed images. We show that for a compression ratio of 128:1 we can nearly double the segmentation quality (Dice coefficient) between the worst configuration (2D-network) and the best one (3D-netwtork with fine-tuned weights). For the second task, we find that by using the optimized quantization in JPEG the classification accuracy can be improved by up to 3%.

Conclusion The compression and the DL architectures can be optimized in order to compress the image to a higher level while maintaining equivalent DL performances.

Acronyms

Bpp Bits per pixel

CBCT Cone Beam Computed Tomography

CNN Convolutional Neural Network

CT Computed Tomography

DCT Discrete Cosine Transform

DL Deep Learning

DWT Discrete Wavelet Transform

EBRT External Beam RadioTherapy

HVS Human Visual System

ML Machine Learning

MSE Mean Square Error

pdf probability density function

PSNR Peak Signal-to-Noise Ratio

Saas Software as a service

SMBD Symmetric Mean Boundary Distance

SNR Signal-to-Noise Ratio

SSIM Structural SIMilarity

Contents

Chapter 1

Introduction

According to recent reports from Cisco [1, 2], a leading company in the network market, IoT devices will represent 50 percent of all global networked devices and connections by 2023. For each human on the planet there will be 3.6 devices connected to the internet, exchanging data with the rest of the world. Such devices have many applications that rely on images like video surveillance, automation and self-driving cars and they will all be contributing in a major way to the growth of the internet traffic. On the other hand, image has become central in the way we communicate and consume information, in social media like Instagram or Snapchat, in communication media like Skype or Zoom, and in advertisement services like Youtube and Netflix. Adding to that the multiplication of smartphones and a bigger part of the world population having access to them, videos and images will represent a major part of the global IP traffic that is supposed to increase 3-fold from 2018 to 2023 according to Cisco. All this "Big Data" is both a challenge for data management (storage and transmission) but also a goldmine for machine learning (ML) and computer vision which can extract strategic information and value from it. This master thesis aims at addressing this challenge of data management through the evaluation and optimization of the compression process for computer vision purposes.

1.1 Computer Vision

A surveillance camera which detects the presence of burglars, a car able to recognize people and obstacles or a computer finding cancer cells in medical images, this is just a preview of what computer vision has to offer. This field has known a growing interest these last few years due to the leap in performance induced by the Big Data, the Deep Learning (DL) and the reduction of computation cost. This is why a lot of actors in the industry have taken interest in it and have developed new

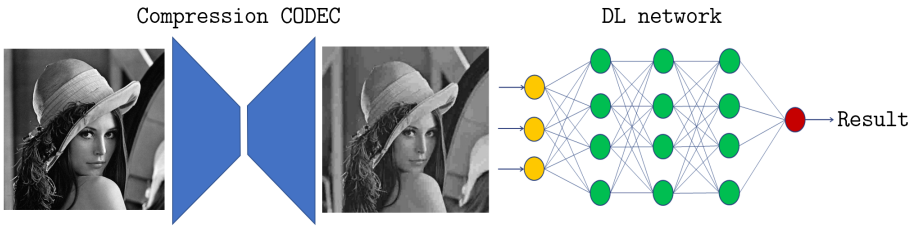


Figure 1.1: Data flow with compression for DL computer vision.

products in the form of Software as a service (Saas) for computer vision. We can name for instance Amazon Rekognition[3], Face++[4], Baidu[5], SenseTime[6] or Google Vision[7] for applications going from content moderation to face recognition. Those important actors and their Saas rely on remote access to offer more flexibility to their clients and because of the large computational and memory cost of their algorithms which cannot be supported by front-end devices. But the apparition of these new services comes at a bandwidth cost for the internet traffic, firstly for the training where large datasets are constructed and exchanged and secondly when the Saas is used under a client/server configuration and that the request and reply need transportation through the internet.

1.2 Compression

To handle the image management issue partially induced by those new computer vision applications, compression is a privileged solution as its size reduction allows to stock and transport images and videos more efficiently. Through standards like JPEG, JPEG2000 and MPEG, compression compacts the images and videos usually without distorting them in a noticeable way for the HVS. However, for machine-based purposes we can wonder if the full quality is really necessary and if the human-oriented information preserved by compression is the one needed for the computer vision task. Figure 1.1 contains a usual data flow observed in computer vision, after acquisition the images are compressed and stored using a standard codec and then they are uncompressed to enter a DL network which gives a result for a specific task the network was trained for. If we want to optimize this flow to reduce the size of the compressed file while improving or maintaining the result of the network, multiple interrogations arise concerning either the compression or the network. Indeed, we can wonder what the best codec is and what parameters it should contain or for the DL what type of network should be used and what training it should go through.

1.3 Experimental study

In order to answer these questions, two experimental cases are selected. In the first one, we build on Eliott Brion and Jean Léger's work [8, 9] concerning the multi-organ segmentation of the pelvic region using U-Net and in the second one, we tackle the CIFAR-10 classification problem that is well known in the computer vision community.

Segmentation

In fractioned external beam radiotherapy (EBRT), the goal is to send radiation to tumor cells in order to destroy their DNA. First a specialist defines the localization of the tumor and the volume to target. This planning phase is based on computed tomography (CT) x-rays images. Then, radiation doses are delivered to the tumor in multiple sessions. Before each session, cone beam CT (CBCT) images are taken to reposition the patient in order to match the planning made by the specialist. This phase is important because if the targeted zone does not correspond to the actual tumor, it might damage healthy tissues, and if the whole tumor is not targeted, there is a possibility of cancer cells resurgence. The re-positioning has the problem that it consists in a simple translation of the patient while in some areas (like the pelvic region), the organs undergo additional deformations for which this translation is insufficient. To consider these deformations, new techniques are developed with a daily evaluation of the organ deformation and a possible re-contouring of the dose distribution based on new images. But these techniques of online planning can require large computational abilities which might not be available in the scanner room or even in the hospital. Hence, it could be necessary to send the CBCT images to a remote server for processing. However, the CBCT are heavy 3D images that require large bandwidth. A simple way to reduce their size would be to compress them via a lossy compression codec. Yet, before using these compressed medical images, we need to assess the impact of the compression to insure the treatment validity. Furthermore, we can try to take into account this compression to make the algorithms more resilient to it. In the first experimental part of this thesis we will go through this matter for the multi-organ segmentation that can be a sub-part of the online planning process.

Classification

The CIFAR-10 dataset is one of the most widely used dataset in machine learning research. It consists of 60,000 images divided in ten categories of animals and vehicles, and it has mainly been used to test various neural network architectures and techniques for a classification task. This dataset and task will be used in this

thesis to optimize the JPEG compression. First, we will train a network with the uncompressed dataset and then we will use two algorithms (a Lagrangian and a genetic algorithm) to optimize the JPEG compression parameters in order to maintain the best possible performance for different compression rates.

1.4 Master thesis plan

The main goal of this master thesis is to analyze and evaluate different techniques optimizing the data flow presented in figure 1.1, for the purpose of reaching higher compression ratios for equivalent DL results. The content can hence be summarised as (i) a state of the art overview of the research concerning this subject, (ii) the selection and training of a DL segmentation network considering a pre-existent compression of the input, (iii) the parameterization of a codec for a DL classification network. To the best of our knowledge, this is the first time that one optimizes a segmentation network considering compression or optimizes the JPEG standard with a Lagrangian or a genetic algorithm for a DL purpose.

This work will be organized as follows, we will start by reviewing the scientific knowledge on this matter in section 2, looking into different studies linking compression and deep learning for optimization or evaluation. Then we will realize various experiments comparing different codecs, models and training for multi-organ segmentation in section 3. Afterwards, we will use a Lagrangian and a genetic optimization of the JPEG quantization tables to improve the classification of CIFAR-10 in section 4. Finally, we will give some leads on further possible works in this field in chapter 5 before concluding in chapter 6.

Chapter 2

State-of-the-art review

In this chapter, we will investigate the previously existing work concerning the link between compression and deep learning. We will start by looking at the impact of compression on deep learning. Afterwards, we will consider different techniques to optimize the compression process knowing that the compressed images are aimed at a specific machine-based task. Finally, we will consider different papers that evaluate the possibility of working with the compressed features of images for computer vision. They use, among others, the Discrete Cosinus Transform (DCT) and the Discrete Wavelet Transform (DWT) coefficients in images or the motion vectors and residues in videos.

2.1 Impact of compression

When doing compression, two types of algorithm are available: lossless and lossy. Lossless compression offers no loss of information but is limited in its minimum bit rate, while lossy compression can offer lower compression rate but at the price of some information loss. Before using lossy compression on images contained in the training set or on images used in the final application of a DL algorithm, we need to validate the fact that this compression does not have a significantly negative impact on the performances. It seems logical that lossy compression would reduce the performances because it creates information loss and distortions. Moreover, a compression configuration which creates a lower compression rate should create more artifacts and diminish the DL performance. Nevertheless, we still need to validate this hypothesis and to establish the relation between the compression ratio and this presumed performance diminution. Once this relation is found, we can select the compression/performance trade-off we want for this application.

This methodology was partially applied by Zanjani et al. who published a research on the impact of JPEG2000 on deep convolutional neural networks for

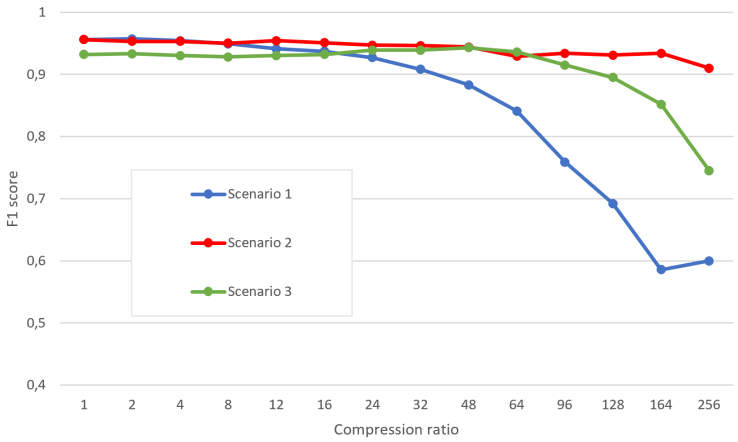


Figure 2.1: Results of [10], "Impact of JPEG 2000 compression on deep convolutional neural networks for metastatic cancer detection in histopathological images"

cancer detection in histopathological images in the Journal of Medical Imaging [10]. In histopathological imaging, image compression is of major importance because of the Whole-Slide Images (WSIs) size (about 1000 histopathological WSIs use more than 3 terabyte of memory). In order to address this challenge, the research team proposes 3 different scenarios:

1. The network is trained on original images and tested on compressed images at different ratios.

2. The network is trained on compressed images at a specific ratio and is tested on images compressed at the same ratio.

3. The network is trained on compressed images at the ratio 48:1 and is tested on images compressed at different ratios.

Each scenario explores via its configuration a specific optimization in resources or in performances. The first one gives a lot of resources to the training because of the full quality of the image used for the training. On the other hand, it aims at limiting the resources used for the final application (e.g. for a remote use of the CNN) by testing the network with compressed images. The second and third scenario both use less resources for the training and testing, the second one optimizes the network for a specific rate and the third one examines the flexibility of a network trained on a specific ratio in front of other ratios. A part of the results they obtain is presented on figure 2.1. Their conclusions are the following:

- Scenario 1: The performances are stable up to a compression ratio of 24:1 but drops significantly at higher compression.

- Scenario 2: Training and predicting at the same compression level improves the results.
- Scenario 3: Even if it is trained for a specific ratio a network gives good results for lower compression level.
- Scenario 2 & 3: It is possible to teach a network the artifacts of compression.

This study is closely related to chapter 3 of this master thesis and has been a major source of inspiration.

The proposition of master thesis from Professor Benoit Macq linking compression and machine learning came from a first study written in 2002 in collaboration with the MIT. Named "Classification of Compressed DICOM Liver Tissue Images" [11]. This study used a low depth multi-layer perceptron to classify healthy and sick liver images and showed that their model was robust against different levels of JPEG2000 compression. In parallel, they observed better performances for lowly compressed images than for original images and gave the hypothesis that a low rate compression can have a denoising effect and can be beneficent for machine learning.

If we get out of the neural network field, some other papers considered the impact of compression on hand-crafted and machine learning algorithms. In [12], Rathgeb et al. examine the impact of JPEG, JPEG2000 an JPEG-XR on the performance of the CAHT and WAHET algorithms for iris segmentation. In [13] Delac et al. investigate the influence of JPEG and JPEG2000 on ICA, PCA and LDA based face recognition. Both of these studies try to find the best standard for their application and show a superiority of JPEG2000 for low bit rates (lower than 0.2 bpp) and while in [12] JPEG gives better results for higher bite rate, the results of the different compressions are comparable in [13]. Like professor Macq's study[11], [13] observes lightly better results for lowly compressed images than for original images and explains it by the filtering effect of the compression.

2.2 Compression optimization

All the studies of section 2.1 used the compression standards as black boxes and only considered their standard parameter, but they forget that the creation and optimization of these standards only aimed at one goal: the Human Vision System (HVS). With the development of computer vision, images and videos might for some tasks never be seen by humans and could directly be used by algorithms, therefore a compression optimized for a human criterion is not optimal. Some papers already raised this paradox and offered solutions to the compression optimization for machine-based tasks.

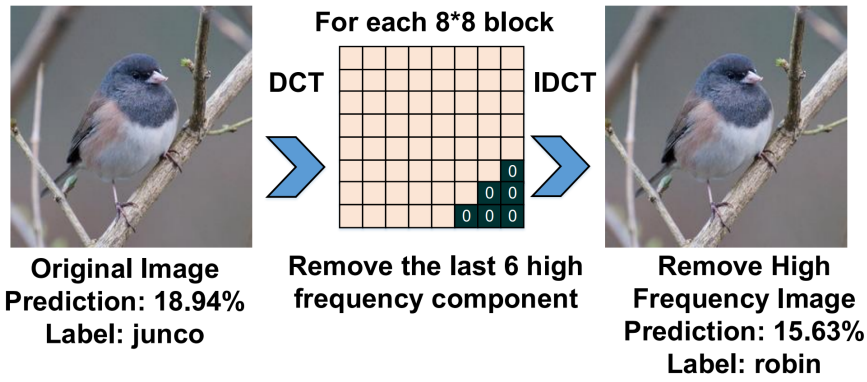


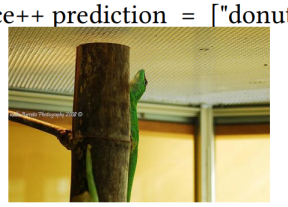
Figure 2.2: Example used in [14] to illustrate the impact of high frequencies on classification CNN, source [14].

In [14], Liu et al. remark that machine-based tasks can be affected by the high-frequency components of an image. Figure 2.2 illustrates this phenomenon, the junco image does not visually change but by putting to zero the high frequency DCT coefficients, the CNN is mistaken and interprets it as a robin. Despite this, the JPEG quantization tables tend to cancel those components because the HVS is not much affected by those frequencies. To solve this issue and to create quantization tables that match machine-based tasks, they create DeepN-JPEG, a JPEG codec with optimized quantization tables. To create those new quantization tables, they sample images from the different categories of the ImageNet dataset, they compute the DCT of these images and based on that, they find the standard deviation of the frequency components. The reasoning is as follows: if a frequency component has an influence on the classification of an image, then it should be statistically different in the different classes and hence, have an important standard deviation. Thus, it means that if a DCT component has a big standard deviation, it is surely discriminant for the classification and that the quantification coefficient should be lower in order to conserve this component accuracy. Applying this to the ImageNet dataset and testing it on different DL classifier they showed that their DeepN-JPEG can achieve more than 3 times higher compression rate than the classical JPEG standard while maintaining the same accuracy.

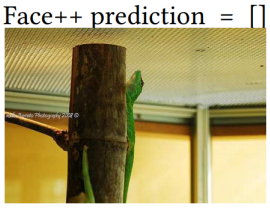
In [15], Li et al. notice that for online computer vision services, the JPEG standard quality level (Q) chosen for compression often creates a size overhead in comparison with the lowest quality level necessary to reach correct performances. To solve this issue, they create an agent which, based on an image and a target computer vision service, gives the proper JPEG quality level considering the compression/performance trade-off. The agent is trained using a reinforcement learning framework with a reward feedback depending on the performance achieved

(1a) Q=75
Face++ prediction = ["donut"]

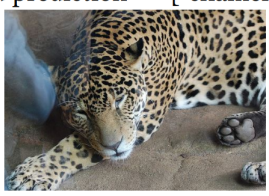
(1b) Q=55
Face++ prediction = []



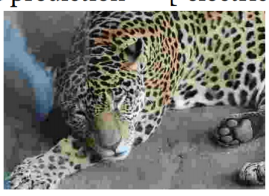
(2a) Q=75
Baidu prediction = ["chameleon"]



(2b) Q=55
Baidu prediction = ["electric fan"]



(3a) Q=75
Baidu prediction = ["leopard"]



(3b) Q=5
Baidu prediction = ["leopard"]

Figure 2.3: For images 1 and 2, minor or invisible changes cause different predictions for image 3 the cloud model still outputs the correct label from a severely compressed image even though it is very different. The prediction of a deep learning model does not only rely on the input image's quality but also on its content, making it inefficient to use the same quality level for all images, source [15].

(a) Conventional solution: fixed user-defined compression level

(b) AdaCompress solution: input image and model aware compression

Figure 2.4: Classical model (a) and optimized model (b), source [15].

on the computer vision service, see figure 2.4. This article makes an important statement as it stands that the proper compression quality level does not only depend on the application for which it is used, but also on the image itself (see figure 2.3). Indeed, each image has its own complexity and will suffer differently from compression. Hence, each image should have its own quality factor. This method is beneficial for the bandwidth consumption but it adds a computational cost on the client side.

In [16], Baluja et al. offer two new ideas concerning specialized compression for machine-based tasks. The first one is to pre-process the images before the compression in order to determine the regions of interest and to crop them in order to only transmit these regions. The second one is to use an autoencoder as codec which does not use the correspondence between original and compressed images (SSIM, MSE, ...) as loss function but rather uses the performance that the compressed image will have on the machine-based task. Both ideas were not experimentally tested but remain theoretically pertinent. A drawback of such techniques is the increased complexity of the resulting codecs in comparison with simple codecs like JPEG.

In the last few years, a new paradigm of compression appeared, called Analyze then Compress. It is in opposition with the Compress then Analyze paradigm previously used. In this new paradigm we first analyze the images by extracting its features and then compress these features to transmit them. This new computer vision oriented paradigm has been especially developed by MPEG with the CDVS [17] (Compact Descriptor for Visual Search) and CDVA [18] (Compact Descriptor for Video Analysis) standards. CDVS is applied to images, it computes the SIFT descriptors and it compresses them in a optimal way, while CDVA is applied to videos and samples images of this video to extract the SIFT descriptors and some deep learning features which are then compressed to be transmitted. Due to their limited features, both of those standards are mainly designed for image and video retrieval or matching tasks. More recently MPEG has gone further in this direction with Video Coding for Machine (VCM) developed by Duan et al. in [19]. The intent of VCM is to join the feature coding for computer vision with the video

coding for human vision using collaborative compression. In opposition to CDVA, VCM seeks to compress collaboratively both the video feed and the feature feed to match multiple goals like human perception and machine analysis. VCM is still in its exploration phase but we can expect more details on this standard from the next MPEG meetings and future publications. CDVS, CVDA and VCM offer a new perspective because by targeting image features, they are aimed for computer vision.

2.3 Compression features

The process of image and video compression is divided in multiple steps, each step creating a representation of the image or the video in a different domain. Depending on the domain, it is possible to take advantage of its properties to improve a deep learning network. Indeed, the transformation in another domain can be seen as a feature extraction, a pre-processing step that can improve the speed or the convergence of a network. Furthermore, the use of this intermediary features on the decoder side would allow us to skip the last steps of decompression and hence save time and computational resources. On the other hand, this representation in another domain does not contain more information. Hence, if the features of these domains led to better performances, we could expect that a complex enough DL network would create them itself. It means that this new representation should not significantly improve the accuracy of the results.

Concerning the use of compressed representations, a large number of research has already been led on the subject for the JPEG DCT representation [20, 21, 22, 23, 24]. In [20], Dan Fu & Gabriel Guimaraes show how the use of the low frequency DCT coefficients can accelerate the convergence of the training of a fully connected neural network while maintaining performances for a classification task. In [22], Gueguen et al. try different configurations of modified ResNet-50 networks using DCT coefficients as input. Depending on the configurations, in particular if we concatenate the YCbCr DCT features early or late in the convolutional process, they observe big variations in the accuracy. For the ImageNet classification problem, they find networks both faster and slightly more accurate compared to the classical ResNet-50 with RGB inputs. [23] and [21] also use DCT coefficients as input for their networks and show a small impact on classification accuracy with the CIFAR and MNINST datasets. For classification in [22] and object detection in [24] the researchers find that the DCT networks and the RGB networks reach comparable accuracy, while the DCT network is smaller than the RGB and thus is around 2 times faster. This can be explained by the fact that the DCT coefficients are high-level input features, requiring less processing, and hence allowing the use of smaller and faster networks for comparable results.

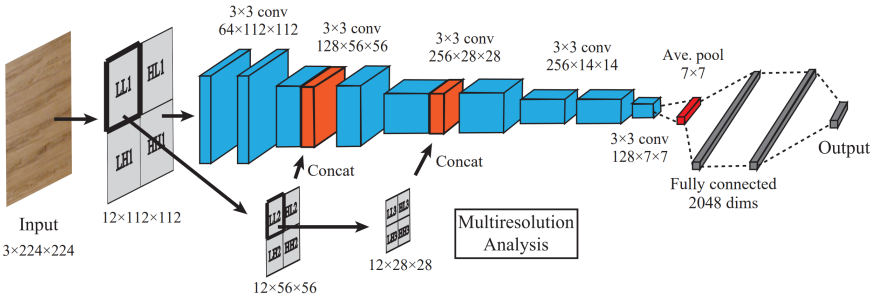


Figure 2.5: Network used in "Wavelet Convolutional Neural Networks for Texture Classification", source [27].

Another image compression feature which can be used as CNN input is the wavelet representation. This representation is used in [25, 26, 27] to improve deep-learning performances thanks to the multi-resolution and the spatial/frequency information it offers. These studies did not aim at using these features straight out of a compressed dataset but we can infer this possibility. All three studies show the benefits of the DWT in terms of accuracy and network speed. Like the DCT coefficients, the DWT coefficients can be considered as high-level features but with the advantage of multi-resolution. Indeed, the multi-resolution DWT representation divides the image in matrices with different size and frequency content, this can be an asset for CNN because it allows to add different frequency content at different places in the network. A good example is the network used in [27] for texture classification. In textures, high frequencies are of first importance and thanks to the DWT multi-resolution it is possible to put the high frequencies as first input of the network and then to progressively add the lowest frequencies. It allows a deeper processing of this high frequencies and hence better results (see figure 2.5).

For a few years now, a lot of research has been conducted on autoencoders. The goal of such networks is to find a new representation with a reduced size and dimensionality. Due to the image shrinkage allowed by the autoencoders, some attempts have been made to use them for compression and one in particular interests us. In [28], Torfason et al. analyze the use of autoencoders bottleneck deep features as direct input for classification and segmentation networks, meaning that the image is compressed into deep features and that those features are directly used for another deep learning task without reconstructing the image. They show that by skipping the decoding step of the autoencoder and directly putting the bottleneck features into the network, we need between 1,5 and 2 times less operations for a comparable accuracy. They also find that for low compression rates the direct use of deep features is more efficient than the image reconstitution

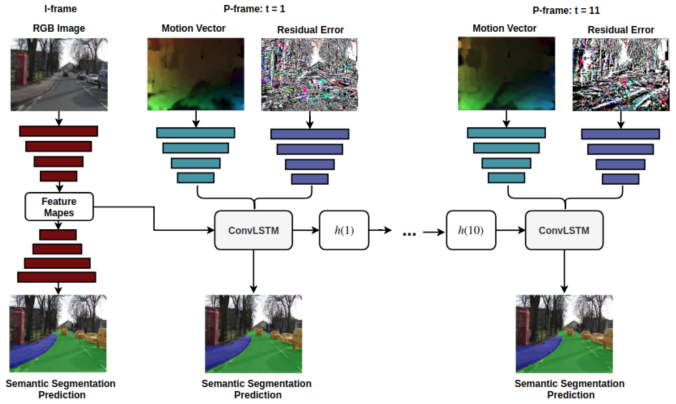


Figure 2.6: Configuration used in "Semantic Segmentation in Compressed Videos, source [29]."

and its classification/segmentation. In a second time they train both the codec and the classification/segmentation task with a loss function that depends on both the error metric of the reconstructed images and the accuracy of the classification/segmentation task. By doing so, they observe that this parallel optimization significantly improves the classification/segmentation accuracy.

This way of using compression features has been extended from image to video in [29, 30, 31, 32] where the features of block motion compensation video compression are used for tasks like action recognition, object detection and semantic segmentation. Using I-frames (images) and P-frames (motion vectors and residues), they find that it is possible to improve the speed in fps in a significant way (from 3 to 15 times) for equivalent accuracy. Different configurations are possible, in [30], Wu et al. use three different CNNs (for the I-frames, the accumulation of motion vectors and the accumulation of residues) each contributing to the final score of a motion recognition task while in [29, 31] authors prefer a recurrent neural network where a hidden state is propagated through time like shown on figure 2.6.

Chapter 3

DL optimization

After reviewing the literature in chapter 2, we now have multiple ways of optimizing the compression and the DL. In this chapter we will explore some of these possibilities for the DL side as we will establish an evaluation and optimization process for the multi-organ segmentation of the male pelvic region based on U-Net CNN. To do so, we will conduct four different experiments:

- In section 3.2 we compare the robustness of the 3D and 2D U-Net architecture against JPEG2000 compression in order to select the right architecture and find that the 3D architecture is more resilient to compression.
- In section 3.3 we show that we can improve the U-Net robustness by retraining the network with a compressed dataset. The resulting network has fine-tuned weights specific to a compression ratio.
- In section 3.4 we test some of the fine-tuned networks developed in section 3.3 and show their flexibility by evaluating their performance at ratios they were not fine-tuned for.
- In section 3.5 we redo the experiment of sections 3.2 and 3.3 with JPEG instead of JPEG2000 in order to compare their performances and select the right codec for compression and show the superiority of JPEG2000 for high compression rates.

3.1 Experimental Setup

In this experimental phase we will perform a supervised training of the rectum and bladder segmentation based on a dataset composed of CT and CBCT images but before showing our results we will describe in detail the dataset, networks and metrics used.

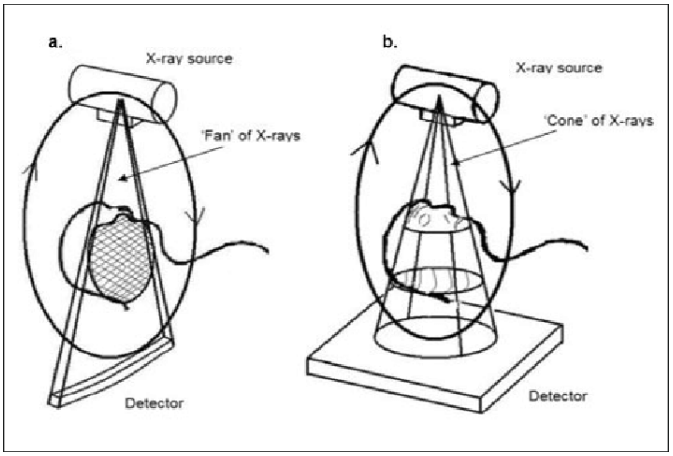


Figure 3.1: Caption process for CT (a) and CBCT (b) images, source [33].

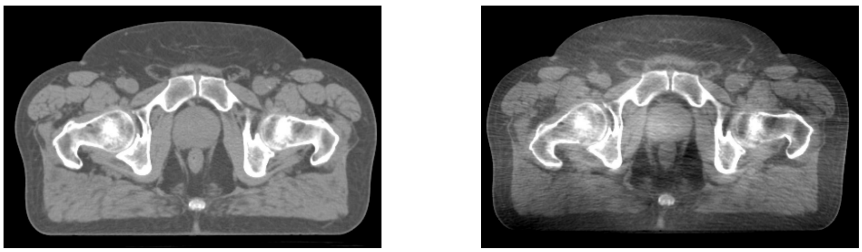


Figure 3.2: Comparison between CT (left) and CBCT (right) images, source [8].

3.1.1 CT and CBCT images

The dataset is composed of two types of images: CT and CBCT, both rely on x-rays for their acquisition but use different equipment and reconstitution techniques. Figure 3.1 represents the caption process of CT and CBCT images. In CT the images are reconstructed slice by slice, which gives 2D images with pixels, while in CBCT images are directly reconstructed in a volume and in voxels. CT images are often of better quality than CBCTs and this is why they are used for the planning in radiotherapy, but their acquisition is slower and this why CBCTs are used for a fast daily repositioning before each treatment session. CBCTs also contain more artifacts (noise, beam hardening or scattering) and they offer poor soft tissue contrast (as shown on figure 3.2). This lower quality of CBCTs make them harder to segment.

3.1.2 Dataset and pre-processing

The dataset was provided by the CHU-Charleroi Hôpital André Vésale and the CHU-UCL-Namur. The CT images come from 74 male patients (22 from André Vésale and 52 from the CHU-UCL-Namur) and the CBCT images come from 56 male patients (all from André Vésale). The images were first contained in the DICOM format and underwent a first compression with unknown parameters. We suppose this first compression to create nearly no distortion as it is the standard used in the medical field and it was made not to impact physician decisions. To create volumes, CT slices were concatenated and to assure similar dimensions and scale between the CT and CBCT volumes, they were re-sampled to obtain a 1.2 x 1.2 x 1.5 mm regular grid in 192 x 192 x 160 matrices. In these greyscale matrices, voxels have an integer value between 0 and 255. Each volume comes with 3 binary masks of identical dimensions representing the bladder, the rectum and the background. The dataset was divided in a training set consisting of 105 volumes (45 CT and 60 CBCT) and in a test set consisting of 25 volumes (11 CT and 14 CBCT) to reach a proportionality of 80%/20% of the dataset. Before entering the networks, each matrix was normalized using the mean and standard deviation of the training set.

3.1.3 Networks

Two different architectures are considered: 3D U-Net and the 2D U-Net. The networks are presented in figures 3.3 and 3.4, the code implementing these architectures was largely inspired by [34] for the 2D U-Net and [35] for the 3D U-Net. Both networks are very similar in their structure, except for the following differences: (i) the 3D uses an additional dimension at all levels, (ii) at each level the 2D has four times more feature-maps than the 3D, (iii) the 3D network uses an additional pooling/up-convolution, (iv) the volumes were re-scaled for the 3D from 192 x 192 x 160 to 160 x 160 x 128 to insure convergence under GPU memory limitations. The U-Net configuration was first developed in [36] by Ronneberger et al. and is now considered as a reference architecture especially efficient for segmentation tasks. This fully convolutional network offers the advantage of using a contracting path in order to extract contextual information and create high level features while propagating large scale features in an up-scaling path that allows a fine-grained feature positioning and a detailed large scale output (like a segmentation map). The 2D network (resp. 3D network) uses 3x3 convolution (resp. 3x3x3) with zero padding in order to preserve the feature dimensions. Both networks double the number of feature-maps at each scale in the down-sampling path where 2x2 (resp. 2x2x2) max poolings with stride 2 are used. The up-sampling path uses 2x2 (resp. 2x2x2) up-convolutions which halves the number of feature maps at each scale.

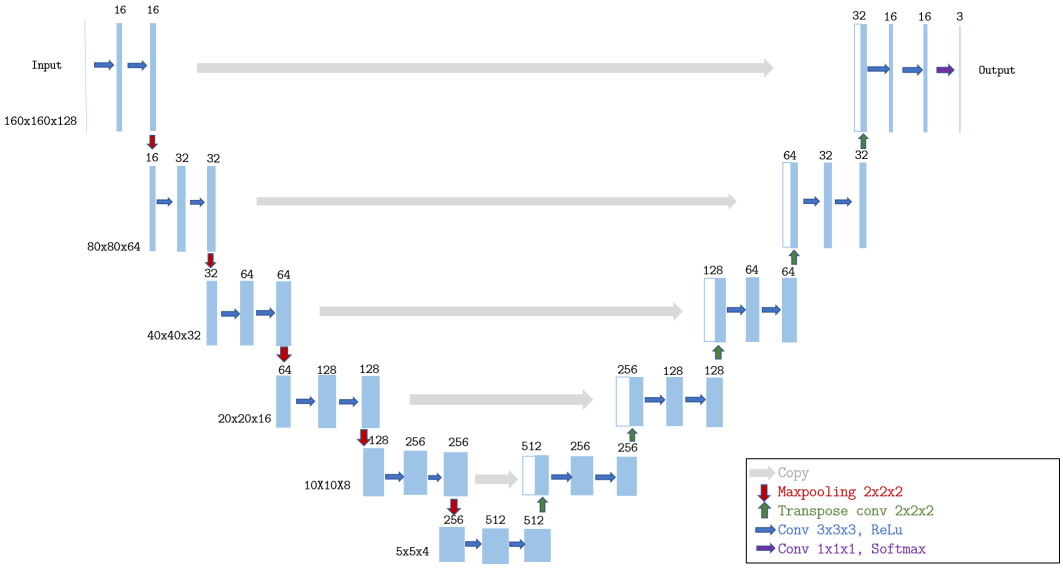


Figure 3.3: 3D U-Net architecture scheme.

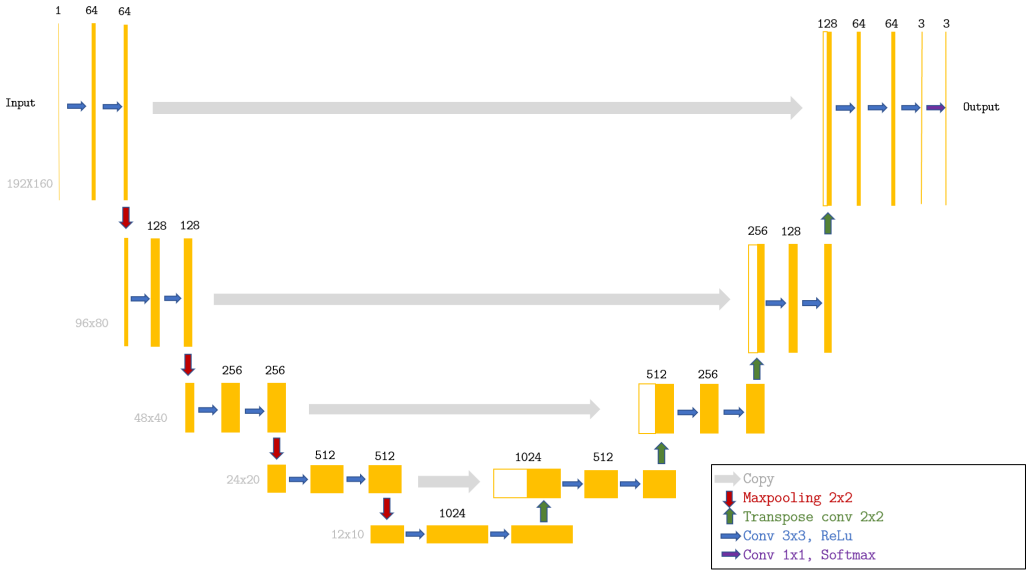


Figure 3.4: 2D U-Net architecture scheme.

3.1.4 Metrics and loss function

In this subsection we define the metrics used to analyze the performance of the segmentation and the loss functions used for the networks training. We use P_i and $p_{i,(x,y,z)}$ (resp. T_i and $t_{i,(x,y,z)}$) to describe the predicted (resp. targeted) set i and the predicted (resp. targeted) voxel values at position (x,y,z) in the volume. The target is the mask made by the medical specialist and the prediction is the output of the network. The i value is 0,1 or 2 and stands for the segmentation class (0=bladder, 1=rectum and 2=background).

The first metric used is the Dice Similarity Coefficient (DSC) as defined in equations 3.1 and 3.2, it measures the validity of the segmentation as it compares the prediction and target overlap volumes to their own volumes. This metric is bounded between 0 and 1 where 1 is the best segmentation achievable.

$$DSC_i = \frac{2 \cdot |P_i \cap T_i|}{|P_i| + |T_i|} \tag{3.1}$$

$$DSC_i = \frac{2 \cdot \sum\limits_{x,y,z} p_{i,(x,y,z)} \cdot t_{i,(x,y,z)}}{\sum\limits_{x,y,z} p_{i,(x,y,z)} + t_{i,(x,y,z)}} \tag{3.2}$$

The second metric used is the symmetric mean boundary distance (SMBD). If we consider a point $p_{i,b}$ belonging to the predicted border $P_{i,b}$ and the target border $T_{i,b}$ we can describe the distance $d(p_{i,b}, T_{i,b})$ as

$$d(p_{i,b}, T_{i,b}) = \min_{t \in T_b} ||s \odot (p_{i,b} - t_{i,b})||_2 \tag{3.3}$$

where $||.||_2$ denotes the usual Euclidean norm and $s^t = (1.2, 1.2, 1.5)$ is the pixel spacing in mm that allows a dimensional validity. Then we measure the mean distance

$$D(P_{i,b}, T_{i,b}) = \frac{1}{n} \sum_{p_{i,b}} d(p_{i,b}, T_{i,b}). \tag{3.4}$$

This definition is asymmetric as $D(P_{i,b}, T_{i,b}) \neq D(T_{i,b}, P_{i,b})$, it is then convenient to introduce the Symmetric Mean Boundary Distance,

$$SMDB = \frac{D(P_{i,b}, T_{i,b}) + D(T_{i,b}, P_{i,b}))}{2}. \tag{3.5}$$

While the DSC measures the overlap of the segmentation masks, the SMDB looks at the distance between the mask contours. This two metrics give two different approaches of the segmentation accuracy.

For the training of the 3D network, we use the loss function L_{3D} as defined in equation 3.6 which is based on the opposite of the organs DSC. The uneven volumes

of the classes created convergence issues which were solved by the weighting of three for the rectum DSC and the discard of the background DSC.

$$L_{3D} = -DSC_{bladder} - 3 \cdot DSC_{rectum} \tag{3.6}$$

For the 2D network it could happen for some batches that none of the organs was present in the slices making the use of a loss function like L_{3D} impossible. Thus, we decided to use the cross-entropy function for the L_{2D} loss function,

$$L_{2D} = -\sum_{i=0,1,2} \sum_{x,y} t_{i,(x,y)} log(p_{i,(x,y)}). \tag{3.7}$$

To compare the 2D and 3D network performances in a rigorous frame, the DSC and SMDB results shown will always be computed based on 3D matrices. Thus, for the 2D network each slice will be predicted individually and then, they will be concatenated to reconstruct the volume before calculating the DSC and SMDB.

The first goal of our networks is not to reach a state-of-the-art level of accuracy. Therefore, the hyper-parameter selection of our networks was a greedy process which only aimed at an acceptable convergence of the training set performances. To execute a rigorous hyper-parameter selection, we should have tested multiple loss-functions, learning rates and batch sizes which is a time consuming process. For the same reason, no data-augmentation was performed to gain time during the training.

3.1.5 Compression

To perform the compression of our volumes, we divided our volume into 192 slices of size 192x160 and we compressed each slice individually. Two different standards were used for this master thesis in order to compress the data set: JPEG and JPEG2000. For JPEG2000 the python glymur library (which contains an interface to the OpenJPEG library) was used, this library allowed us to fix a target compression ratio for the image compression. For JPEG the python imageio library was used, in this library the compression ratio cannot be directly settled as the compression depends on a quality factor bounded between 1 and 100 that determine the quantization tables used. In order to compress at a target ratio, a corresponding quality factor has been manually determined for each ratio.Images with a quality factor of 1 gave a compression ratio of 48:1 which was hence the maximal compression achievable in JPEG. In appendix B is a table containing the target ratios, the expected compression size and the mean size achieved by JPEG2000 and JPEG (plus the quality factor chosen for JPEG).

	2D U-Net	3D U-Net experiment
Optimizer	ADAM	ADAM
Starting learning rate	10^{-4}	10^{-4}
Loss Function	L_{2D}	L_{3D}
Evaluation metrics	DSC and $SMDB$	DSC and $SMDB$
Batch size	8	1
Number of epochs	100	150

Table 3.1: Training parameters.

3.1.6 Result tables

All the graphs shown in the following sections have a corresponding table in appendix C that contain the precise DSC and SMDB values and their standard deviations.

3.2 Experiment 1: 3D vs 2D

In medical imaging a lot of recent studies prefer the use of 3D segmentation over the 2D segmentation because the additional dimension is an additional source of information that leads to better accuracy. Despite this improved accuracy the 3D segmentation faces some challenges like the lack of 3D annotated dataset [8] and its large memory cost. In order to further evaluate and compare the 3D and 2D segmentation, we decided to test their robustness to JPEG2000 compressed images. In this first experiment we will **train our networks on uncompressed images and test them on compressed images at different ratios** in order to establish their robustness to compression. The 2D and 3D networks were adapted in their architecture so that they both reach comparable performances on uncompressed images. The parameters used for the training are presented in table 3.1.

3.2.1 Results

Figures 3.5 and 3.6 contain the results of this experiment. From these figures we can observe that the segmentation is relatively unaffected up to a compression ratio of 32:1 and that the accuracy decays for superior ratios. We find that in general the 3D offers a better robustness to JPEG2000 for the rectum and the bladder and for both the DSC and SMBD but that there are some exceptions for which the large standard deviation of the results does not provide a statistical evidence (like the SMDB of the rectum and the DSC of the CT bladder). Figures 3.7 and 3.8 contain representative examples of segmented organs for images compressed at ratio 96:1.

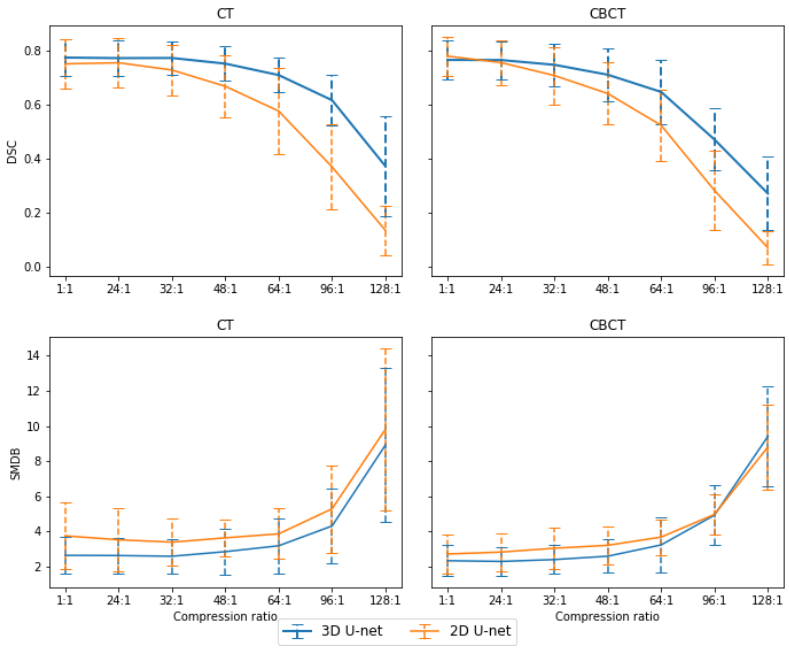


Figure 3.5: Mean and standard deviation of the segmentation performance for the rectum.

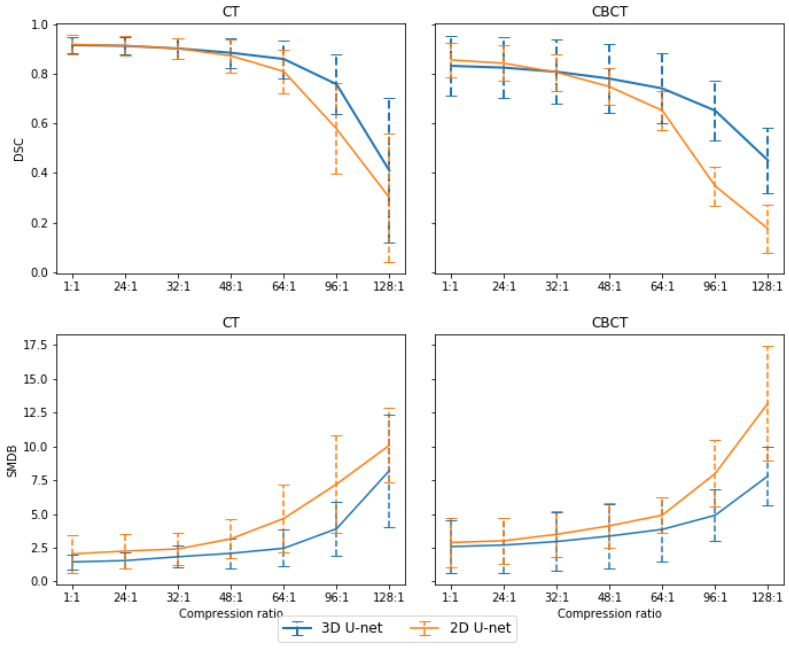


Figure 3.6: Mean and standard deviation of the segmentation performance for the bladder.

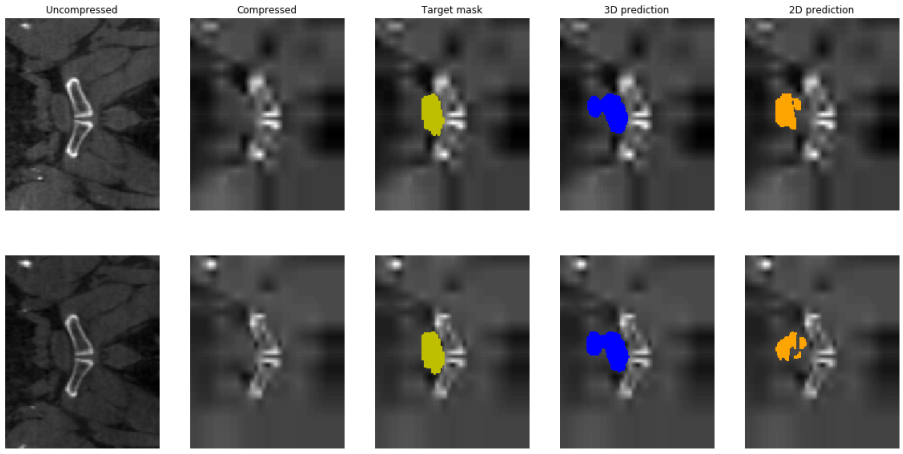


Figure 3.7: Example of the segmentation of the bladder for two successive slices compressed at ratio 96:1.

On them we observe a segmentation closer to the target mask for the 3D prediction than for the 2D prediction. Additionally, we observe large discontinuities between slices in 2D prediction while the 3D predictions tend to be more continuous. This discontinuities in the 2D segmentation can significantly increase the SMBD value.

3.2.2 Analysis

We can explain the robustness of the 3D network to compression by its additional dimension. indeed, the JPEG2000 artifacts will mainly consist in a blurring of the artifacts and it is easier to recognize a blurred 3D object than a blurred 2D object. Furthermore, the 3D volumes were compressed slice by slice meaning that the distortion was relatively independent for each slice. This has an important impact on the 2D segmentation as 2 successive slices which are very close to each other undergo a very different distortion and hence give very different results (see figure 3.7). On the other hand, using the third dimension in the 3D U-Net allows the network to average the noise of the distortion. This idea is represented in figure 3.9, where a rectangular parallelepiped is cut into slices and where a noise is added at the border of each slice, we see that it is easier to find the original shape when we use series of noisy slices than when we use individual slices because the comparison of successive noisy slices allows us to denoise the edges.

In appendix A we consider and develop a parallel and more mathematical approach where we consider the images and volumes as a signal and the artifacts

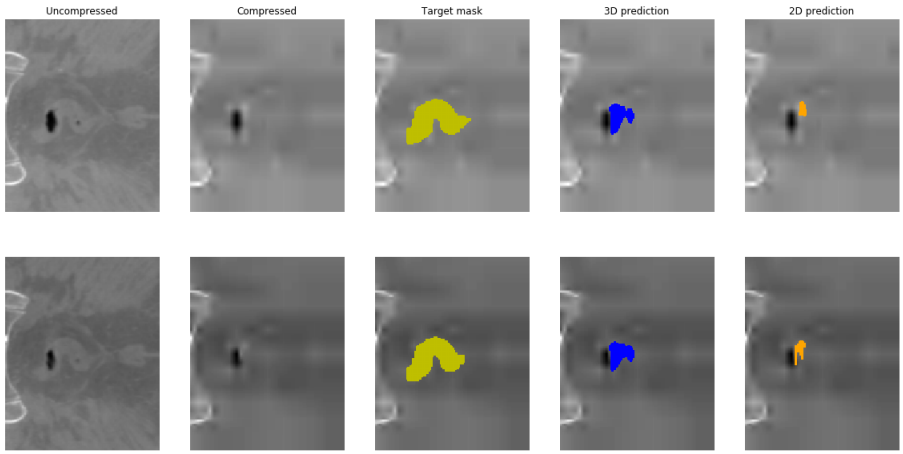
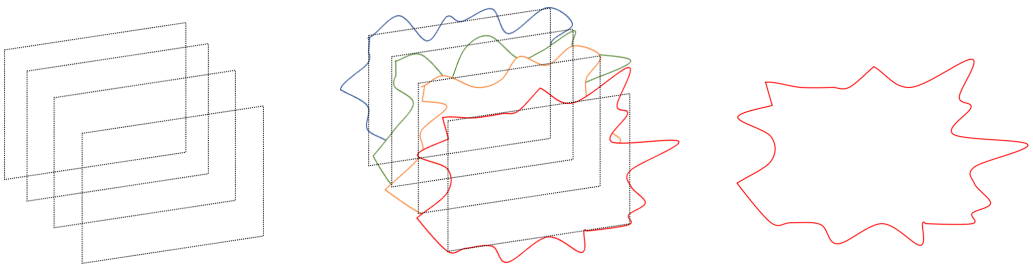


Figure 3.8: Example of the segmentation of the rectum for two successive slices compressed at ratio 96:1.



(a) (b) (c)

Figure 3.9: 3D denoising representation, in (a) is a rectangular parallelepiped divided in slices, in (b) we add an independent noise to the borders of each slice and in (c) is an individual slice. We observe that it is easier to find the original shape in (b) than in (c).

	3D U-Net	Fine-tuned 3D U-Net
Initial weights	random	3D U-Net
Optimizer	ADAM	ADAM
Starting learning rate	10^{-4}	10^{-4}
Loss Function	L_{3D}	L_{3D}
Evaluation Function	DSC_*	DSC_*
Batch size	1	1
Number of epochs	150	50

Table 3.2: Training parameters Experiment 2.

of compression as a noise. We find that it is possible to compare the signal-to-noise ratio (SNR) after a convolution in 3D and 2D and under some hypothesis we show that the 3D SNR is always higher than the 2D SNR.

3.3 Experiment 2: Weights fine-tuning

Now that we have shown the superiority of the 3D, we can try to optimize it in order to make it more resilient to compression. In [10], Zanjani et al. find that it is possible to increase the performance of a CNN for higher compression ratio by using compressed images in the training process. This idea seems pertinent as the use of compressed images during the training allows the network to learn the compression artifacts and to adapt itself to them. **For this experiment we will start from the network trained during the first experiment on uncompressed images and we will fine-tune the networks weights by re-training them on the same training dataset but compressed at different compression ratios with JPEG2000**. For each compression ratio we will restart the training from the basic 3D-unet and create a specific fine-tuned network for this ratio.

3.3.1 Results

Figures 3.10 and 3.11 present the performance of the basic and fine-tuned 3D U-Nets for a test set compressed at different compression ratios (for the fine-tuned curves at each ratio a different fine-tuned network was used to match the ratio). For the compression ratios below 48:1, we observe a slight improvement in the CBCT images and no improvement in the CT images. For higher ratios, the improvement of the fine-tuning is obvious and is even more noticeable for the CBCT than for the CT (e.g. at ratio 128:1 the mean CBCT rectum DSC is nearly doubled). Figures 3.12 and 3.13 show the difference between the basic and fine-tuned 3D U-Net at a

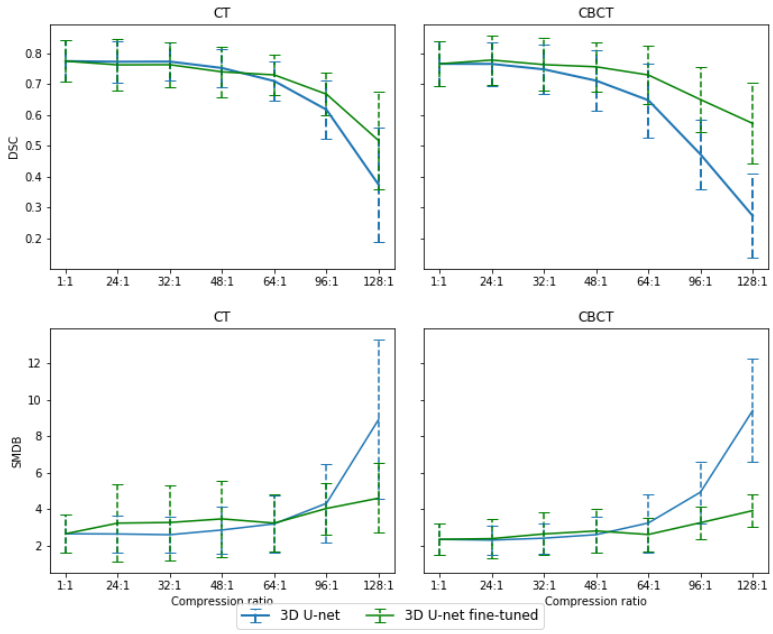


Figure 3.10: 3D U-Net and fine-tuned 3D U-Net segmentation performance at different compression ratios for the rectum.

ratio of 96:1 for successive representative slices. We can observe a slightly better segmentation of the fine-tuned network on them.

3.3.2 Analysis

In different papers like [37] and [38], it was shown that it is possible to train CNNs in order to remove the artefacts created by the compression, meaning that is is possible for a CNN to learn and recognize compression artefacts and to remove them. In a sense, it is comparable to what happens when we fine-tune our network, while keeping the original segmentation task, we teach the network to adapt to the compression artefacts and to remove the distortion in the deep features. The comparable results at low compression ratios for the two networks can be explained by the fact that there is not much room for improvement at these ratios as the performance are closed to the optimal values obtained on uncompressed images. The results at higher ratios show that indeed the fine tuning helps the network to specialize itself.

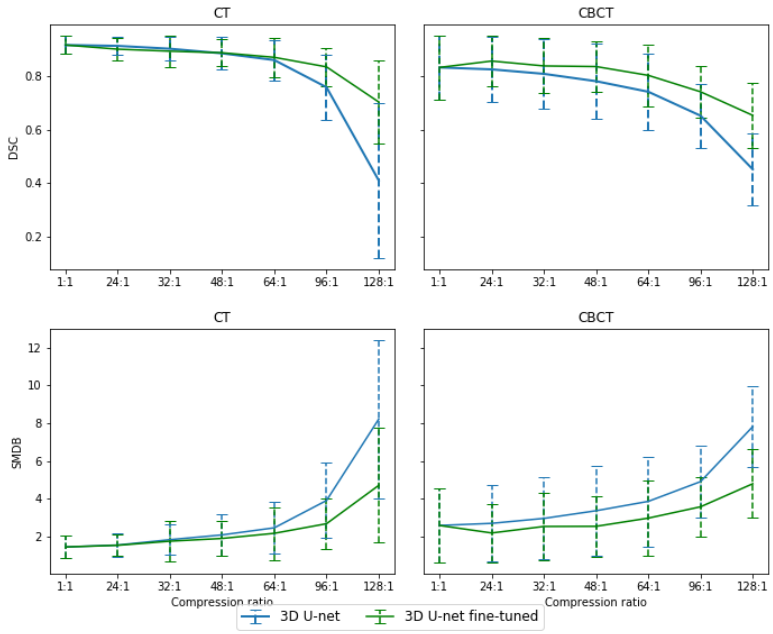


Figure 3.11: 3D U-Net and fine-tuned 3D U-Net segmentation performance at different compression ratios for the bladder.

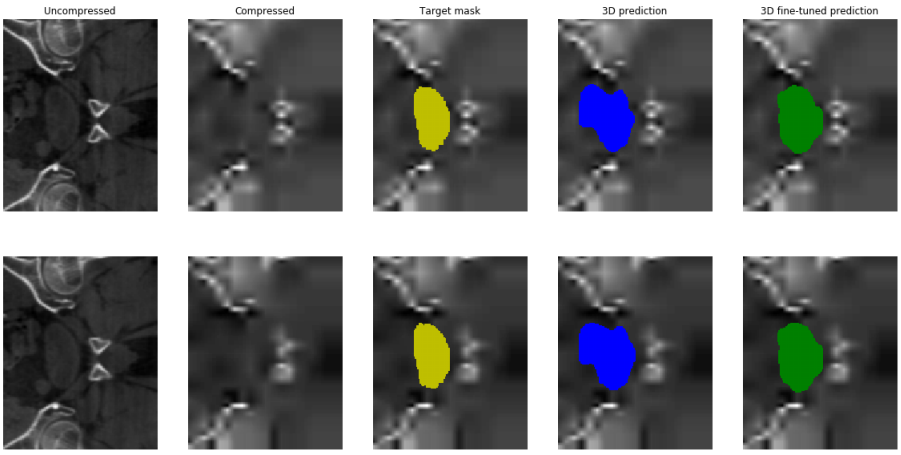


Figure 3.12: Example of the segmentation of the bladder for two successive slices compressed at ratio 96:1.

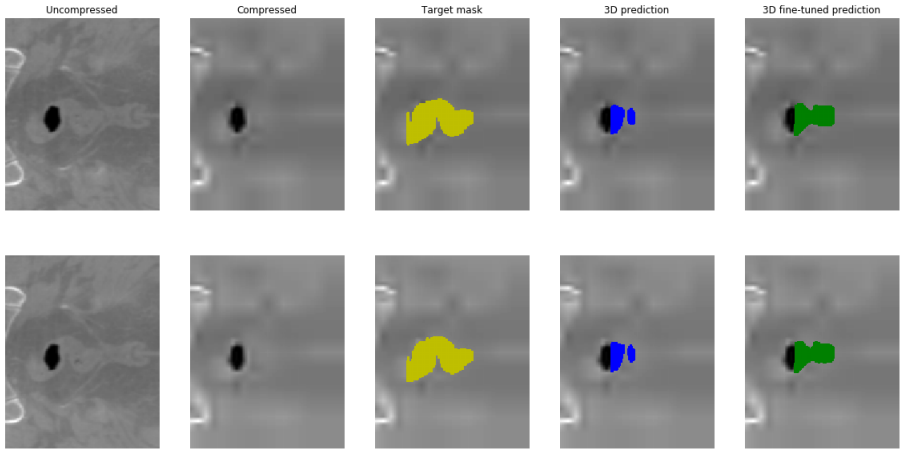


Figure 3.13: Example of the segmentation of the rectum for two successive slices compressed at ratio 96:1.

3.4 Experiment 3: Fine-tuned network flexibility

In this experiment we consider some of the fine-tuned networks trained in section 3.3 and we test them at ratios they were not fine-tuned for. This experiment is important because at the end, only one network will be used and it could happen that the image it treats did not undergo the specific compression ratio it was trained for (e.g. images coming from different hospitals could have different compression ratios). Hence, before using them we need to prove that the performance at these ratios are still acceptable for our network.

3.4.1 Results

On figure 3.14 are represented the rectum performances of the fine-tuned networks trained on ratios 48:1, 64:1 and 96:1 for the test set compressed at different compression ratios. We observe that for each fine-tuned network the performances are relatively stable for compression ratios between 1:1 and the compression ratio it was fine-tuned for (see figure 3.15) , for higher compression ratios the performance decays significantly.

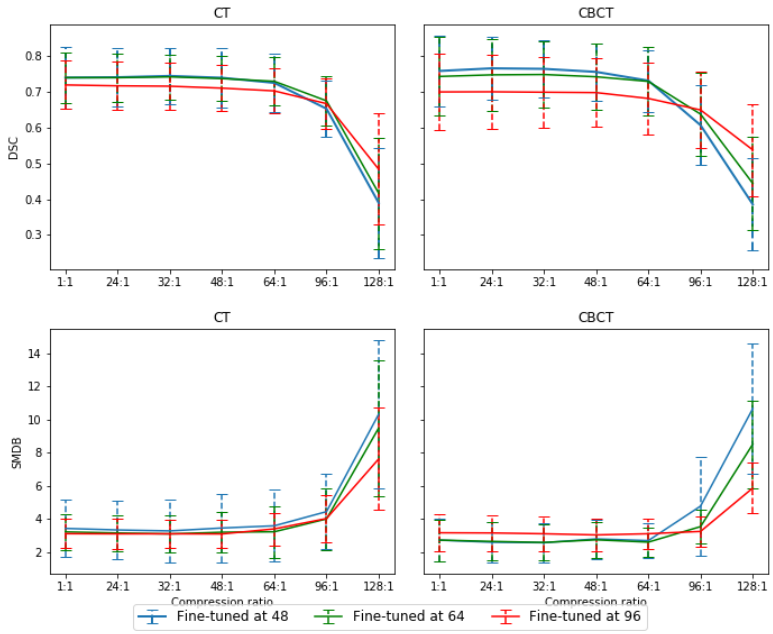


Figure 3.14: 3D U-Net and fine-tuned 3D U-Net segmentation performance for different compression ratios for the bladder.

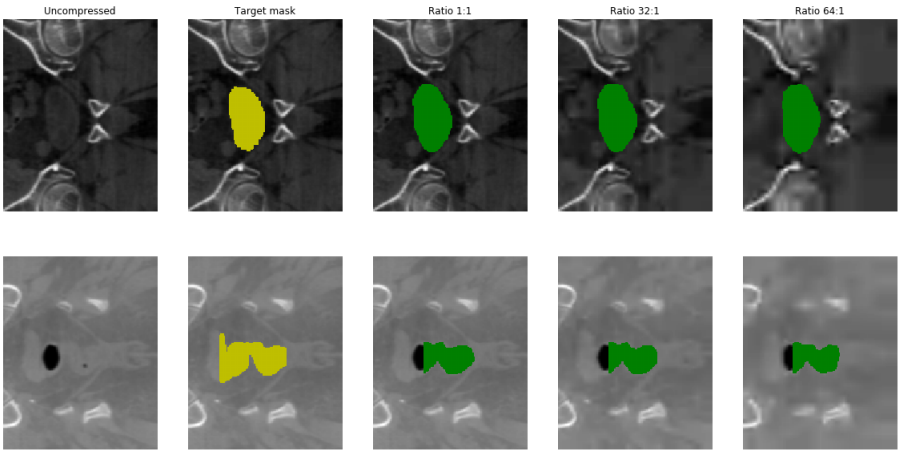


Figure 3.15: Bladder and rectum representative segmentation by the fined-tuned network at 64:1 at different ratios.

3.4.2 Analysis

These results are interesting because they show that while fine-tuning improves the performance at a specific ratio, it tends to limit the performance at all the lower compression ratios. It is as if the fine-tuning training had made the network forget about its first training where it learned to use the details of uncompressed images. A way around this generalization issue could be to not separate the network in two different learning phases and to perform only one training with a training set composed of the basic uncompressed training set and the same training set compressed at various compression ratios. We also observe that for high compression ratios, the network fine-tuned for higher compression ratios tend to perform better (e.g. at ratio 128:1 the fine-tuned at 96:1 is the best and the fine-tuned at 64:1 performs better than the one at 48:1). This is due to the continuity of the compression distortion. Indeed, the compression artefacts at 128:1 can be seen as a worsened version of the artefacts at 96:1 and as an even worsened version of artefacts at 64:1. Hence the fine-tuned at 96:1 will perform better at this ratio as it was trained for compression artefacts close to the ones at 128:1.

3.5 Experiment 4: JPEG vs JPEG2000

The compression standard used so far in the experiments is JPEG2000. It is a well-known and efficient standard, but it is not used in a lot of applications. On the other hand, the JPEG standard, which is older, has proven lower PSNR and HVS performances but is still used in many fields and specially on the web. In this experiment we will compare the performances of the 3D U-Net and the fine-tuning for JPEG and JPEG2000 compressed images. The training parameters will be the same as the ones used in experiment 1 and 2. The goal here is to establish what compression standard should be recommended and under which conditions the JPEG format can be used.

3.5.1 Results

The results are presented in figure 3.16 for the "classical" 3D U-Net and in figure 3.17 for the fine-tuning experiment. The maximal compression ratio of 48:1 limits the comparison, but we observe that JPEG gives lower performances than JPEG2000 at this ratio.

3.5.2 Analysis

This lower performance of JPEG could have been expected just by taking a look at the images after compression. Indeed, in figures 3.18 and 3.19 we can clearly see

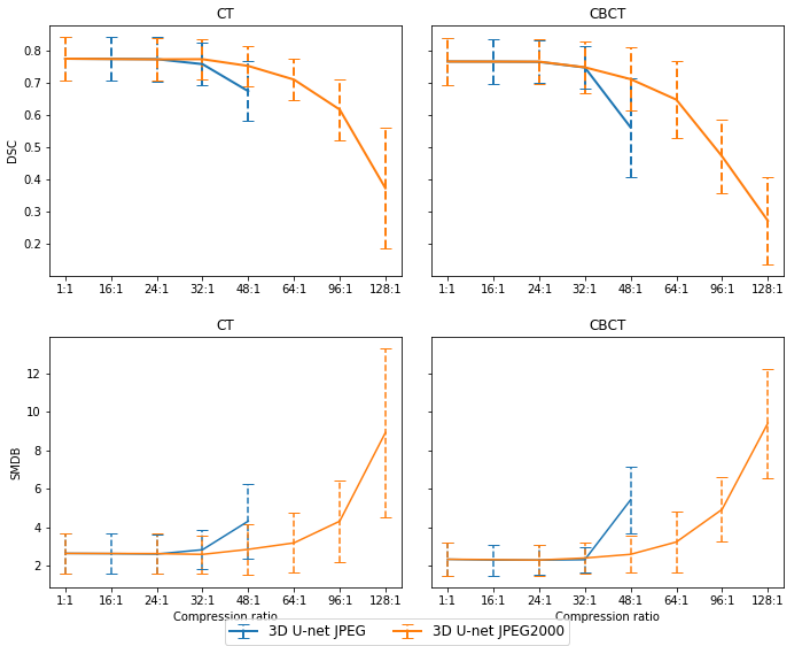


Figure 3.16: 3D U-Net trained on uncompressed images and tested on JPEG2000 and JPEG compressed images.

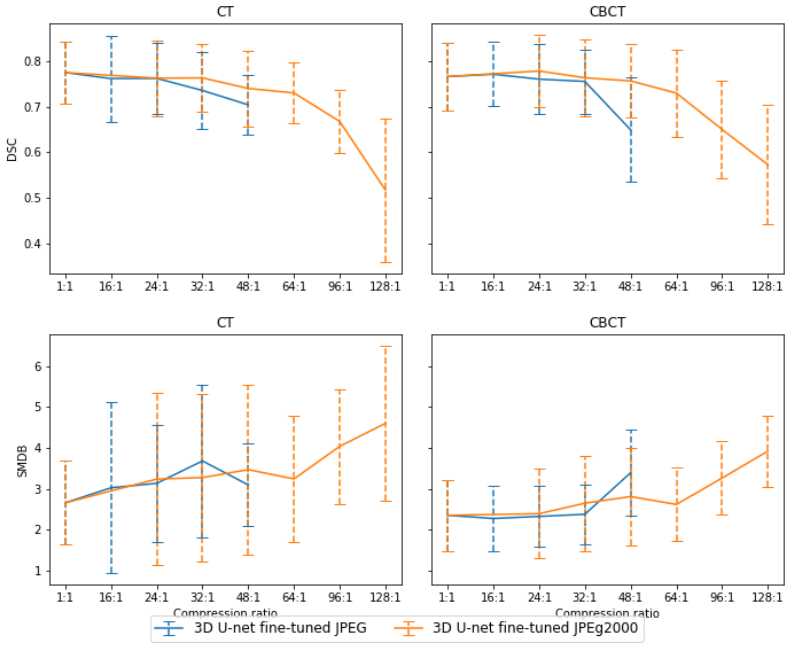


Figure 3.17: Fine-tuned 3D U-Net trained on uncompressed images fine-tuned on compressed images and tested on JPEG2000 and JPEG compressed images.

(a) (b)

Figure 3.18: Slice compressed at ratio 48:1 for JPEG2000 (a) and JPEG (b).

(a) (b)

Figure 3.19: Slice compressed at ratio 32:1 for JPEG2000 (a) and JPEG (b).

that at ratio 32:1 all the elements can be distinguished for both standards while at ratio 48:1 the grid artifacts becomes predominant in JPEG as nearly each 8x8 block is represented by its DC component. From there the superiority of JPEG2000 can be easily explained by its ability to conserve more details. We could wonder how it is possible to obtain a DSC above 0.5 from the JPEG compression at ratio 48:1 when looking at figure 3.18. Yet, it is important to remind ourselves that the network is a **3D** U-Net and that it can then find structures in the third dimensions even if they have coarse edges. For ratio 48:1, if we had used the JPEG images in the 2D network developed in experiment 1 the results would have dropped heavily.

Chapter 4

JPEG quantization table optimization

So far, we have considered the compression standards JPEG and JPEG2000 as black boxes that allowed us to reach a target compression ratio. But it was shown in [14] that JPEG quantization tables could be optimized for DL classification and in [39], by Benoit Brummer, that JPEG-XS parameters could be optimized using an evolutionary algorithm to improve segmentation at low bitrates. In this experiment we will discuss two optimization methods for the JPEG quantization tables: the Lagrangian and the genetic algorithm.

4.1 CIFAR-10

In section 3.4 we saw that JPEG compression had a maximum compression ratio of 48:1 for our medical imaging dataset and that the only variation in performance was observable between 32:1 and 48:1. Then, if we wanted to optimize JPEG for the multi-organ segmentation, we would of had a very limited range of ratios to work with. Furthermore, the small performance decay between 32:1 and 48:1, the large standard deviation and the small size of the test set (25 patients) would make it difficult to obtain a statistically valid proof of amelioration. In order to tackle this problem, we decided to swap towards another dataset and computer vision problem, CIFAR-10 [40]. The CIFAR-10 dataset contains 60,000 32x32 color images divided in 10 different classes (see figure 4.1) and it is well known in the machine learning research. Unlike the medical imaging dataset, the large size of the CIFAR-10 dataset allows us to have a test set with a significant statistical value. Moreover, the optimization process of the JPEG quantization tables require a lot of compressions and network predictions and these operations were slower for the medical imaging task that handles large volumes and heavy architectures

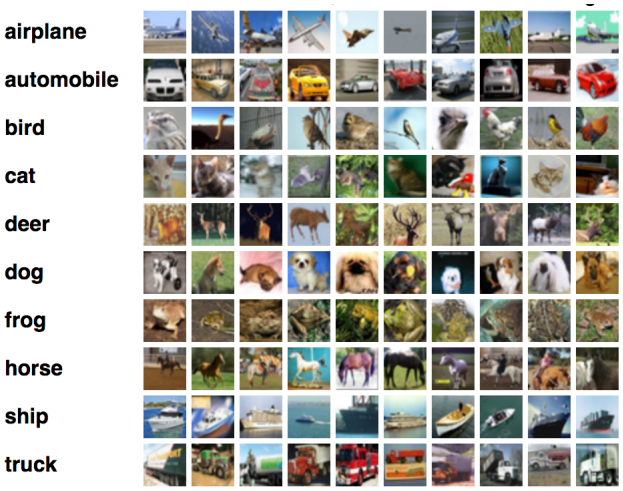


Figure 4.1: CIFAR-10 dataset, source [40].

than for CIFAR-10 task that uses smaller images and requires a shorter network, it was hence faster to optimize the compression for CIFAR-10. For the architecture we decided to use a quite small network consisting of 4 convolution layers and a fully connected layer (more details on this network and its training are available in appendix D). The only metric that will be used for this task in order to assess the performance is the mean accuracy of the classification (the number of correct classifications on the total number of element to classify).

4.2 JPEG compression

The JPEG compression process can be visualized in figure 4.2. First, it creates the YCrCb features from the RGB image (the YCrCb features have the advantage of being more decorrelated) and it sub-samples the Cr and Cb canals as the human eye is less sensible to those components (this sub-sampling is not mandatory). Then, after dividing the image into 8x8 blocks and shifting the values, each block undergoes a mathematical transformation to another 8x8 domain where each coefficient corresponds to a specific frequency and direction, the DCT transform. Afterwards, in the quantization step the 8x8 DCT matrices are divided by a quantization table and each coefficient is rounded to an integer value. It is during the quantization step that the artifacts are created as the division and the rounding consist in an information loss. Despite the information loss, quantization is also an opportunity to increase the compression ratio as it reduces the total entropy. The choice of the quantization table is free and can be decided by the encoder, but in a lot of libraries this choice relies on an encoding parameter called the quality factor.

This factor has an integer value between 1 and 100 and to each of them, there is a quantization table that corresponds.

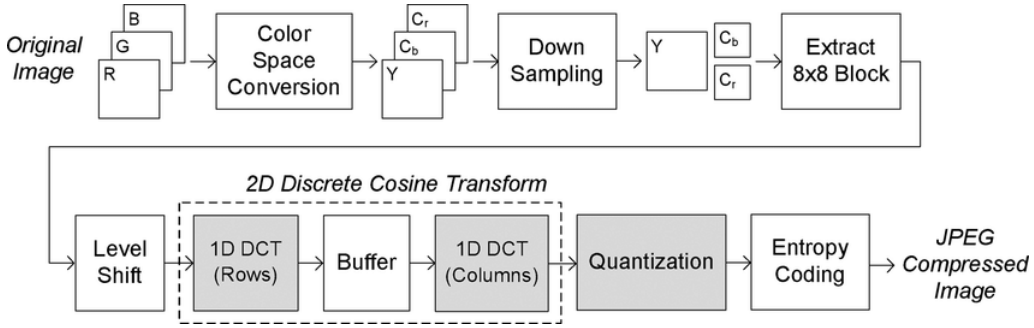


Figure 4.2: JPEG compression process, source [41].

To create these tables, the HVS was the main goal considered and therefore higher values were put in the quantization table at the frequencies that the HVS was proven less sensitive to (generally the highest ones). These quantization tables are hence optimized for HVS perspective but if the compression aims at a DL application it should be the DL performance that is used as optimization metric. Indeed, the sensitivity of DL CNN is different because it is able to comprehend every frequency. As a matter of fact, the different convolution kernels can be considered as filters and hence each frequency can be assessed and used for a decision, it is up to the network to determine what frequencies it prefers to use. Usually, in JPEG there are two quantization tables, one for the luminance (Y channel) and one for the chromance (Cr and Cb channels). In our work we will only consider the optimization of the first one to limit the research space and because the Y channel is known to contain more information.

4.3 Quantization table optimization

As a quantization table contains 64 elements taking value between 1 and 255 for a total of about $3,74 \cdot 10^{460}$ possibilities, the optimization of the quantization table is a large discrete combinatorial problem. The goal is to optimize the accuracy performance P for each possible rate R or, in other words, to find the Pareto frontier in figure 4.3. In this figure each point in the R/P plan corresponds to a quantization table and the Pareto frontier describes the set of optimum solutions. The red dots are the standard quantization tables which might be part of or close to the Pareto frontier and hence the distance between the red points and the borders is all that can be improved compared to the standard compression. The main challenge of this optimization problem is the large search space and the computational cost of

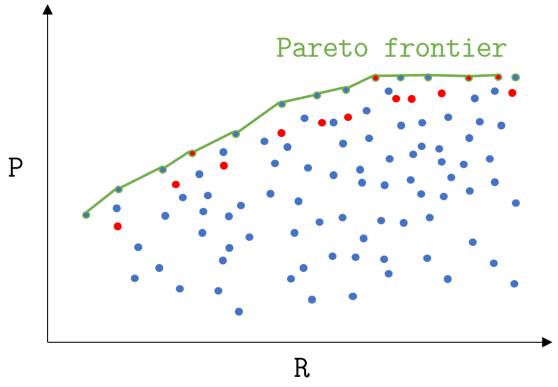


Figure 4.3: Pareto frontier representation in the R/P plan, each dot corresponds to a quantization table, the optimal quantization tables are along the Pareto Frontier in green.

the evaluation function which consists in the compression/decompression of the dataset and the prediction through the network. Two different approaches were evaluated, both with their advantages and drawbacks, the Lagrangian assimilated to a greedy gradient descent and the genetic algorithm based on a stochastic and evolutionary technique.

4.4 Lagrangian optimization

The Lagrangian optimization is one of the basic optimization techniques of optimization theory and it is usually the first tool used to teach constraint optimization. Even if this technique is well suited for continuous mathematical functions, it has to be adapted for discrete problems like the one we face here. In this section we will start by linking the Lagrangian equations to our algorithm before giving some details on how our algorithm works.

4.4.1 Continuous to discrete Lagrangian optimization

If we consider the quantization table $Q = (q_1, q_2, ..., q_{64})$, the mean rate achieved on the dataset for this quantization table $R(Q)$, the mean accuracy of our CNN for the compressed/uncompressed dataset with this quantization table $P(Q)$ and a target rate R_t, then we can set the following constraint optimization problem:

$$\max_{Q} \quad P(Q) \quad s.t. \quad R(Q) = R_t \tag{4.1}$$

Based on the Lagrangian optimization technique, equation 4.1 can be reformulated as a unconstrained maximization problem and if we set the Lagrange function $\mathcal{L}$ as

$$\mathcal{L}(Q) = P(Q) - \lambda(R(Q) - R_t) \tag{4.2}$$

where λ is the Lagrange multiplier, then we can find an optimum at the point where

$$\nabla_{Q,\lambda}\mathcal{L}(Q) = 0 \iff \begin{cases} \nabla_Q P(Q) = \lambda \nabla_Q R(Q) \\ R(Q) = R_t \end{cases} \tag{4.3}$$

Thus, in order to reach the optimum, we need to find a value of Q and λ that satisfies the two conditions of equation 4.3. If we reformulate the first gradient condition by making it correspond to a specific component q_i of Q, we obtain:

$$\frac{\partial P(Q)}{\partial q_i} = \lambda \frac{\partial R(Q)}{\partial q_i} \tag{4.4}$$

Here is the real challenge of the lagrangian optimization and the difficulty our algorithm will have to confront: how do we find Q and λ so that equation 4.4 is satisfied for each component q_i.

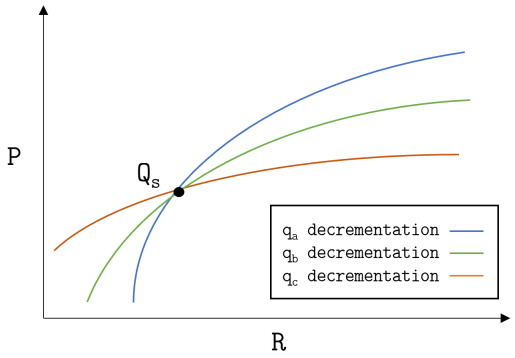


Figure 4.4: Representation in the P/R plan of the impact of the decrementation of one component from a Q_s quantization table, for three different components q_i.

To reach this goal we consider a starting value for Q named Q_s so that $R(Q_s) < R_t$ and we plot the evolution of $P(Q_s)$ and $R(Q_s)$ when modifying only one parameter q_i and conserving the others (see figure 4.4). If we look now at the slope of each curve in this figure, we can mathematically link it to the Lagrangian multiplier of equation 4.4,

$$\lambda_i = \frac{\frac{\partial P(Q)}{\partial q_i}}{\frac{\partial R(Q)}{\partial q_i}} = \frac{\partial P(Q)}{\partial R(Q)}. \tag{4.5}$$

We argue that in order to find the Lagrange optimum, we have to find Q so that $\forall i\ \lambda_i = \lambda$, meaning that we should find a point in the P/R space where each curve has the same slope. To do so we consider two assumptions:

- **Each $\mathbf{q_i}$ component is totally independent in its impact on P and R** meaning that a displacement along one curve does not affect the other curves which hence are just translated along the curve, see figure 4.5. This assumption is reasonable because each q_i will impact a different frequency and if we make the hypothesis that each frequency affects individually the rate and the performance, then their impact is independent from each other.

- **The slope of each curve is strictly decreasing and positive when we decrement $\mathbf{q_i}$**. The decreasing slope (or second derivative negativity) assumption can be made if we consider the progressive creation of artefacts in compression. For high rates the compression gets rid of details that have little impact on the performance but when we try to reduce the rate we start to delete more important information and create bigger artefacts. The positive slope assumption can be made because the more we decrement q_i, the less there is distortion due to the quantization and the better the performances are.

Based on these two assumptions and in order to displace ourselves in the R/P plan towards a position where the slopes converge, we find that we should always displace ourselves along the curve with the highest slope. This idea is illustrated on figure 4.5 where a displacement from Q_s to Q'_s along the highest slope and a translation of the other slopes induced by the independence between q_i components bring the curves and mainly their slope at the intersection closer to each other. This is a sign that we get closer to a Lagrangian optimum where every slope is identical.

Starting from these results we implement an algorithm described in the pseudo-code of algorithm 1 that starts from a quantization table Q_s and decrements its coefficients depending on their λ_i. Because we are working in a discrete environment we can only approximate the value λ_i with a discrete formula as in equation 4.6 where $Q_s = (q_{s,1}, q_{s,2}, ..., q_{s,i}, ..., q_{s,64})$ and $Q_s = (q_{s,1}, q_{s,2}, ..., q_{s,i} - dec, ..., q_{s,64})$.

$$\lambda_i = \frac{P(Q_s) - P(Q'_s)}{R(Q_s) - R(Q'_s)} \tag{4.6}$$

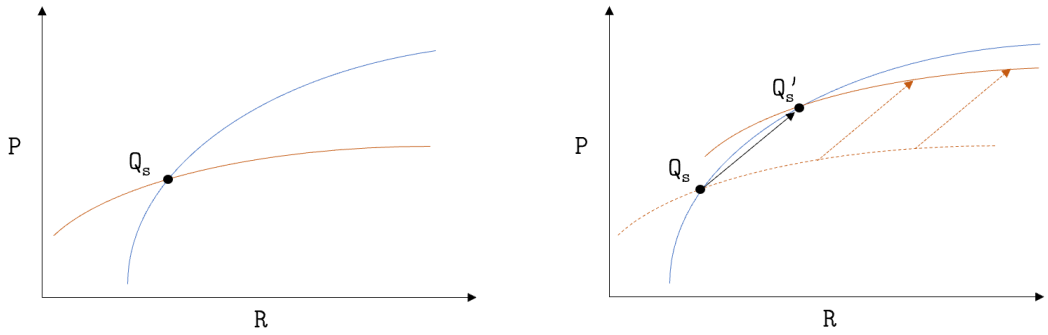


Figure 4.5: We see from figure (a) to figure (b) a displacement along the blue curve due to an decrementation of the corresponding q_i value and we observe a translation of the orange curve allowed by the independence between the q_i components. We observe at the point Q'_s that the slopes of the curves are closer to each other and hence closer to a local optimum according to the Lagrangian.

Another way to see this optimization technique without thinking of a Lagrangian is to imagine a greedy gradient descent algorithm. This greedy algorithm has 64 parameters q_i that it can decrement. A decrementation of q_i will create an increase in P and R named ΔP_i and ΔR_i but it wants to maximize P and minimize R, hence it wants to maximize $\Delta P_i / \Delta R_i$. In other word, it will choose to decrement the coefficient which will offer the best performance improvement by compression rate loss. In figure 4.6 this algorithm is represented (limited to 4 q_i component), at each step we move through the component with the highest slope and after each iteration we recompute the slope of the component we just used. The algorithm stops when it exceeds R_t the target rate.

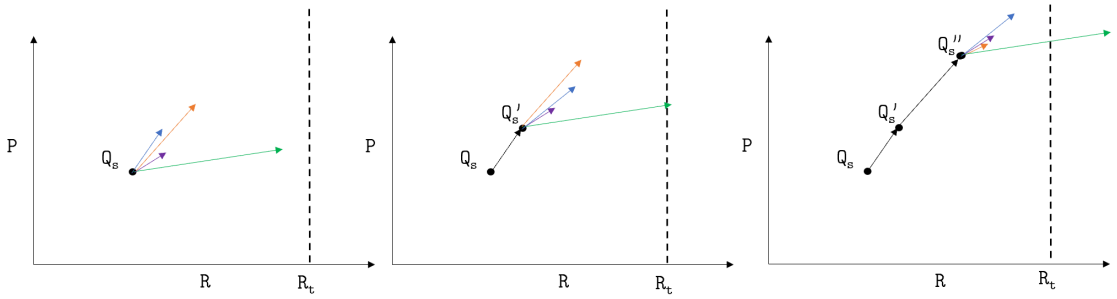


Figure 4.6: Lagrangian algorithm representation, the different arrows represent the displacement the quantization table modification will do in the P/R plan if a specific q_i is decremented, at each iteration the algorithm greedily selects the arrow with the higher slope (the higher λ_i).

Algorithm 1 Lagrangian optimization implementation

Input: The image dataset $\mathcal{I}$, the target rate R_t and a starting quantization matrix Q_s.
Initial rate and performance calculation: Compute the rate R and performance P for the initial Q_s.
Rate variation calculation: Calculate the rate matrix $R_{m,i}$ produced by a decrementation of the q_i component of Q_s for $1 \leq i \leq 64$.
Performance variation calculation: Calculate the performance matrix $P_{m,i}$ produced by a decrementation of the q_i component of Q_s for $1 \leq i \leq 64$.
Lagrangian multiplier estimation: Computes the Lagrangian multipliers base on equation 4.6, $\lambda_i = \frac{P_{m,i}-P}{R_{m,i}-R}$ for $1 \leq i \leq 64$.
while $R \leq R_t$ **do**
  Argmax: Finds i so that $\lambda_i > \lambda_j \forall (j \neq i)$
  Decrementation: Decrements the q_i component of Q_s
  Update: Recomputes R,P,$R_{m,i}$ and $P_{m,i}$ based on the new Q_s
  Recomputes λ_i
end while
return Q_s

4.4.2 Algorithm details

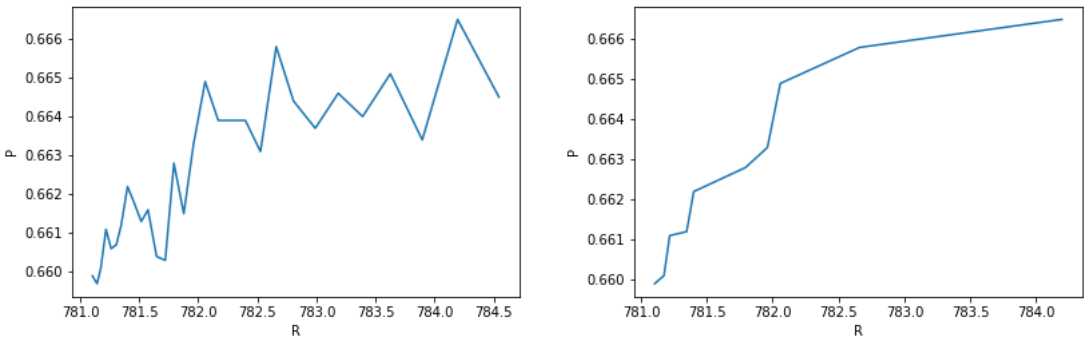
Due to the high computational complexity of the problem, some decisions had to be taken in order to limit the running time of our algorithms. Furthermore, some assumptions announced previously are partially incorrect and the algorithm had to be adjusted in consequence.

Start quantization table

Theoretically, the best starting quantization table available is the one with all its coefficients set to 255 because it lets the possibility to discover the optimum among every quantization table. But such quantization table is distant from the optimum and then it requires a very long time in order to give a solution. Thus, we decided for a limitation rate $R_t(Q(qf = i))$ (where $Q(qf = i)$ is the quantization table corresponding to the quality factor i) to use a starting quantization table $Q_s = Q(qf = i - 10)$. This will allow a starting point close to the pareto frontier but with some distance from the target rate to let the algorithm reach an optimum.

Limited computation of λ_i

In algorithm 1 we compute a first time the λ_i for each q_i component and then we recompute only the λ_i of the component which was decremented. This way of



(a) (b)

Figure 4.7: The curve for a constant step of decrementation of 1 for a component q_i in (a) and the curve for a smoothing step for the same component in (b).

doing is based on the hypothesis of independence of impact between components but it was observed experimentally that this hypothesis was not totally true as the decrementation of a component modifies the λ_i of the other components. Hence, in order to have a correct evaluation of the λ_i at every time, we should have a re-evaluation of each λ_i after each decrementation but such a repeated computation is too timeconsuming and computationally expensive. It was hence decided to only recompute the decremented component.

Step selection

A hypothesis that was made previously is that the slope created by the decrementations is always positive. Figure 4.7a shows how this assumption is false because the curve of decrementation is noisy. This is a serious problem as the noisy curve creates negative λ_i which will hence never be selected to be decremented. In order to solve this issue, we allow the possibility of a decrementation step different than one. To choose this step the algorithm will look at the sign of the λ_i and it will increment the step until it improves the performance. This has the impact of virtually smoothing the curve as shown in figure 4.7b.

Training and test set

The different performances, rates and λ used in the algorithms were computed on a subset of 10.000 elements of the 50.000 elements training set used to train the network. We did not use the whole training set to limit the computational cost. The testing was made on the 10.000 elements of the test set.

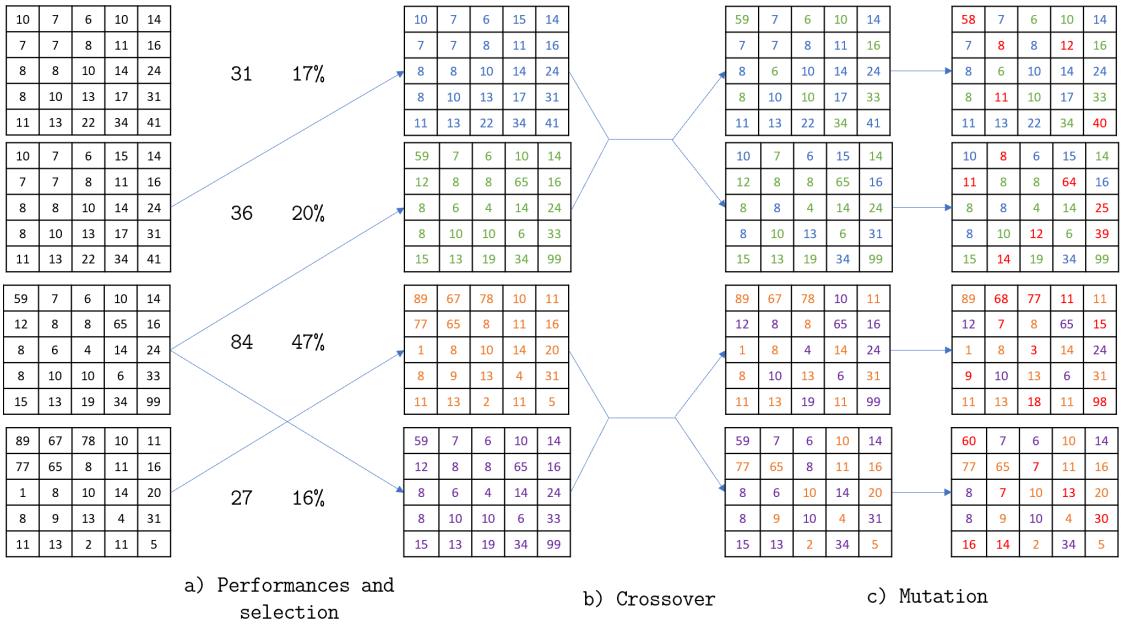


Figure 4.8: Scheme of the genetic algorithm implementation for the quantization table optimization. For visual purpose the quantization tables are represented by 5x5 matrix instead of 8x8 matrix.

4.5 Genetic Algorithm

The Lagrangian approach explained previously is based on a greedy kind of gradient descent. Yet figure 4.7a (a) shows that the optimization landscape is far from smooth and in this kind of landscape a gradient descent based algorithm could easily fall into a local minima. It is why we also attempt the use of a genetic algorithm which has large exploratory abilities and is well suited for a high dimensional and combinatorial space or for complex fitness landscapes. The genetic algorithms is a stochastic algorithm based on Darwin's theory of evolution describing "the survival of the fittest". In this algorithm we create a population of solutions and evaluate the fitness of each solution. Based on their performance the algorithm will tend to conserve the fittest solutions. Afterwards, it will create new solutions by mixing the old ones, hoping that the new generation will conserve the best sub-parts of the previous solutions. At the end the solutions undergo some small mutations which give them the opportunity to develop new abilities. If these abilities impact the solutions positively, they will improve the score of the solutions and then, they will tend to be conserved. This algorithm offers large exploration abilities which prevent local minima solutions thanks to the mutations and crossovers.

Figure 4.8 contains a representation of our implementation of the genetic

algorithm and it is divided into 3 main steps:

1. Selection: During this phase, for each table of the population, we compress and decompress a subset of the training set for each quantization table Q_i. Then we let these elements go through the classification network and we measure the mean accuracy of the classification p_i. Without any external intervention the algorithm will progressively decrement each coefficient and this will surely improve the performances but it will also reduce the compression ratio r_i. In order to avoid that the rate gets above the target compression rate r_t, we penalize a quantification table that compresses less than the target rate by dividing its performance by a coefficient a.

$$if \ \ r_i < r_t \qquad p_i = \frac{p_i}{a} \tag{4.7}$$

   Then, to obtain the probability P_i of a quantization table being selected, we apply a softmax function with coefficient b to all the population performances. The softmax function allows a normalization of the performances and the b emphasizes the difference of performances. Indeed, the variation of performance is very small between two neighboring quantization tables and in order to have a bigger impact on the selection, the difference has to be enhanced.

$$P_i = \frac{e^{b \cdot p_i}}{\sum_{j=0}^{n} e^{b \cdot p_j}} \tag{4.8}$$

   Finally, based on the P_i probabilities the population is renewed.

2. Crossover : During the crossover step we mix the quantization tables two by two. Each coefficient conserves its position (DCT component) but the result contains a proportion c of the first table coefficients and a proportion $(1 - c)$ of the second table coefficients.

3. Mutation : At the end each coefficient has a probability d of being either incremented or decremented of one unit.

For our handcrafted implementation of the algorithm, we used the parameters presented in table 4.1. The test set was a subset of the training used to train the network and the test set is the same as the one used to evaluate the network. The initial population consists of the standard quantization giving the target compression rate R_t.

a	4
b	120
c	0.7
d	0.25
Population size	12
Number of iteration	700
Training set size	5000
Testing set size	10000

Table 4.1: Genetic algorithm parameters.

4.6 Results and discussion

4.6.1 Performance comparison

In figure 4.9 we present the performances of the optimized quantization tables we found with the Lagrangian and the genetic algorithm and the standard ones obtained by the pillow python library for the different quality factors. We see that for an interval between 700 and 850 bytes the Lagrangian and genetic quantization tables gives better results and that the improvement goes up to 3% in classification accuracy for some rates. We observe that Genetic and Lagrangian optimization give comparable results except for 1) a range of rate around 730 where the Lagrangian is superior and 2) for the highest rates where the genetic still offers an improvement. The first difference could be explained by the need for more iterations at these rates for the genetic algorithm and the second one could be explained by the ability the genetic algorithm has to decrement and increment the quantization table coefficient and thus to discover more various possibilities (while the Lagrangian can only decrement).

4.6.2 Image comparison

On figure 4.10 is an example where 3 images were compressed at the same rate using the 3 different quantization tables; the rate was in the middle of the improvement interval of figure 4.9. For these three images the classification was false for the standard table but correct for the two optimized ones, even if there are no or very few visual differences. This illustrates the sensibility of CNNs for distortions and frequencies imperceptible by men.

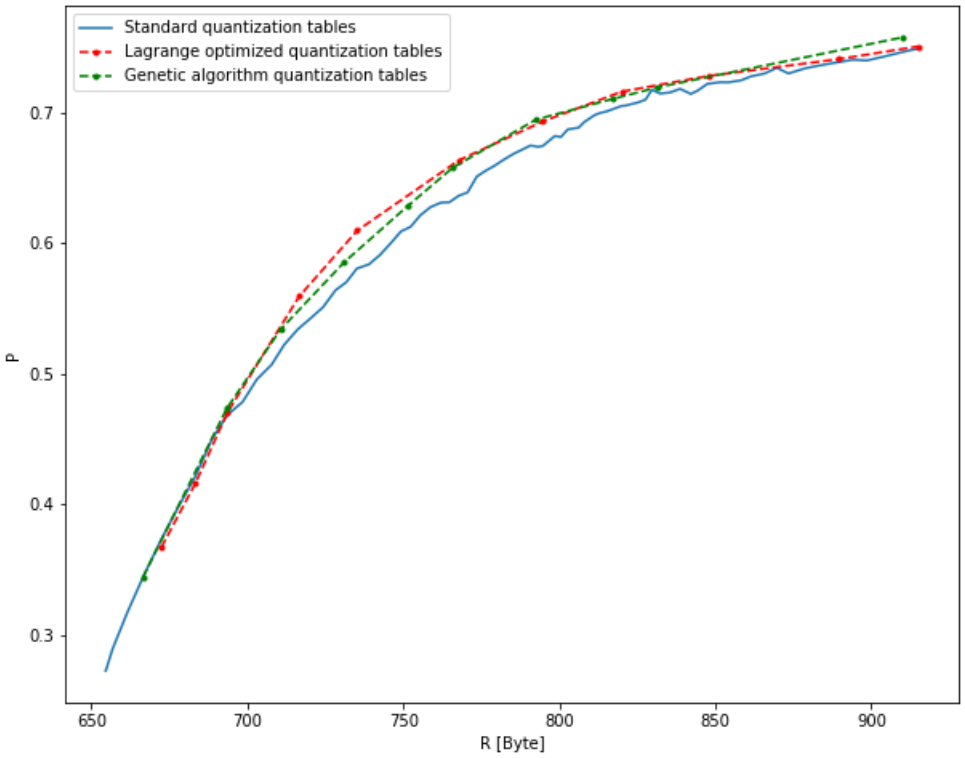


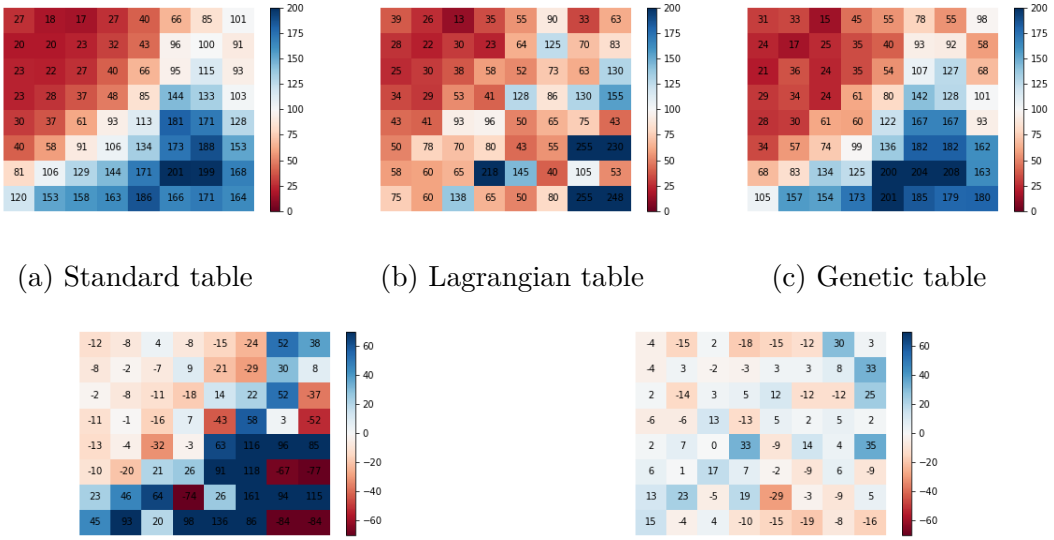
Figure 4.9: Performances of the Lagrangian optimized quantization tables (red), the Lagrangian optimized quantization tables (green) and the standard quantization tables (blue).

(a) Uncompressed (b) Standard (c) Lagrangian (d) Genetic

Figure 4.10: Examples of compression corresponding to a quality factor of 40 where the classification failed for the standard compression but succeeded with the Lagrangian and genetic tables.

4.6.3 Quantization table comparison

On figure 4.11 (a) , (b) and (c) we can observe the different quantification tables for a rate corresponding to a standard compression with quality factor 40; this rate corresponds to the middle of the improvement interval of figure 4.9. The study of these tables allows us to find the frequencies and directions that the optimization processes decided to privilege or discard. To better analyze and compare the standard and optimized tables, we look at figure 4.11 (d) and (e) that contain the differences between the standard and optimized tables. In these figures the coefficients in red (resp. blue) mean that the corresponding frequencies have a lower (resp. higher) impact on the neural network performance according to the optimization. We observe that for the Lagrangian the evolution is independent of the direction and is kind of continuous for the frequencies, the lower and some very high frequencies are diminished, and the medium and high frequencies are preserved. For the genetic table the differences are more chaotic but there are some similarities with the Lagrangian concerning the discarding of low and high frequencies and the promotion of medium frequencies, the chaotic evolution is due to the random mutation of the genetic algorithm and might be a sign that the algorithm did not fully converge and needed more iterations. The very few visual differences observed in figure 4 are probably due to the fact that most of



(a) Standard table (b) Lagrangian table (c) Genetic table

(d) Standard table minus Lagrangian table (e) Standard table minus Genetic table

Figure 4.11: Quantization tables corresponding to rate at a quality factor of 40 for the standard compression (a),(b) and (c) and subtraction between the standard and optimized tables ((d) and (e)).

the differences happen at medium and high frequencies and these differences are not of high amplitude. Equivalent observations were observed at other rates in the improvement interval of figure 4.9 but they differ or become less obvious for higher and lower rates. We notice that the Lagrangian and genetic quantization tables contain a lot of differences and that lead us to believe that there might be multiple solutions with nearly optimal and identical performances.

4.6.4 Other metric comparison

It is interesting to analyze and compare the performance of the newly found quantization tables for other classical metrics like the PSNR and the SSIM. Indeed, it could be that the improvement offered by the optimized tables is not specific to the CNN and that it generalizes to the other metrics meaning that the optimization process improved JPEG even for its global purpose (at least for this specific dataset). On the other hand, it could be that the optimized tables performs less efficiently for the other metrics, meaning that they would be specialized for the CNN task. On figures 4.12 and 4.13 the evolution of the PSNR and SSIM metrics are represented at different rates for the different tables, we observe that except for a lower performance at low rates for the Lagrangian; the tables offer

very comparable results on these metrics. It means that the optimization solely improves the CNN-performance without impacting the classical quality metrics.

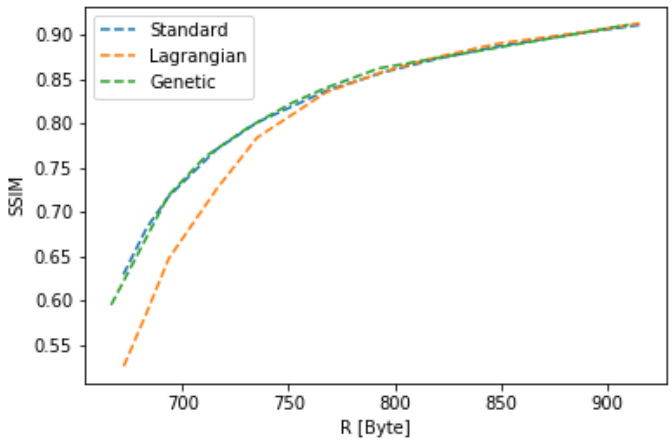


Figure 4.12: SSIM for the different quantization tables.

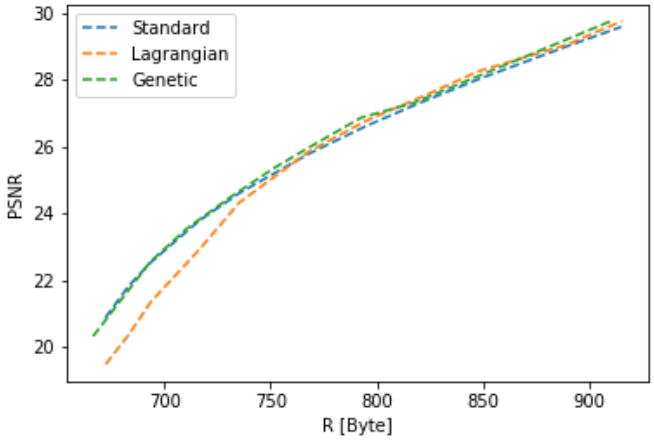


Figure 4.13: PSNR for the different quantization tables.

Chapter 5

Limitations and future works

In this chapter we will build on the performed experiments and review their limitations in order to give leads on how to improve them. We also explore new ideas on how compression and deep learning can be used together and we explore some possible future works in this field.

5.1 Limitations

5.1.1 DL optimization

During the first experimental phase we limited ourselves to the JPEG and JPEG2000 standards, but the field of image compression is composed of a large set of standards (BPG, WEBP, ...). It could therefore be interesting to enter these formats into the comparison. Furthermore, the experimental problem used 3D volumes and compression only considered slices of these volumes, neglecting the important redundancy of the third dimension and the ensuing possible compression gain. Indeed, there exists standards specific for 3D like JPEG2000 3D or we can consider 3D volumes as videos where the third dimension is time and then use video codecs on them. Moreover, this approach can be legitimated by previous studies that showed encouraging results for the CT volume compression based on these codecs ([42] for JPEG2000 3D and [43] for MPEG). On the other hand, we limited our study to the impact on 3D and 2D U-Net segmentation but nowadays the number of different architectures in convolutional networks is growing and so do deep learning innovations. Hence, in order to observe the impact of image compression in a more global manner, it would be necessary to study more architectures and to analyze how compression reacts to DL models with different parameters (depth, number of feature maps, number and type of pooling,...). One of the main default of our study is the lack of statistical evidence, indeed the medical dataset was composed

of 130 volumes and we only did one training/testing split. For such a small dataset it can be judged insufficient as the performance could be specific to this split, in order to obtain an acceptable statistical evidence we would have to redo multiple training/testing splits.

5.1.2 Compression optimization

For the compression optimization part, we only considered the luminance quantization table of JPEG, but the optimization can be expanded to the chromance table in order to furthermore improve the results. Another point we can integrate is the use of higher resolution datasets, indeed we used CIFAR-10 which has a very low resolution (32x32). A dataset with higher resolution images would have a different frequency content and the optimized quantization tables would vary in consequence. Concerning the optimization techniques we limited ourselves to only two optimization techniques but many other approach could of have been used like CMA-ES [44] used by Brummer et al. in [39] or simulated annealing [45]. Moreover, we can imagine similar parameter optimization for other lossy codecs like JPEG2000 and BPG.

5.1.3 Joint optimization

In the DL optimization and in the compression optimization, we have tackled two different tasks of classification and segmentation but if we had used the same task and network, a new approach would have been possible: the joint optimization of the compression and the network. We could imagine for example a cyclic optimization where we would optimize the compression parameters considering a CNN and then optimize the CNN considering this new codec and repeat this process multiple times.

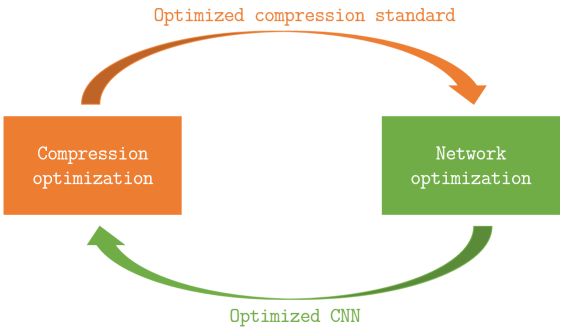


Figure 5.1: Cyclic joint optimization

5.2 Further works

5.2.1 Compression for data augmentation

In machine learning, the data augmentation is a technique used to artificially increase the dataset size and so improve the generalization ability of networks. In computer vision it can be used by applying affine transformations to the images (rotation, symmetries, ...), cropping them or applying kernel filters to them. The idea is to start from dataset images and to modify them so that their representation changes but that the information useful for the network remains. The network is hence trained to handle more different representations and scenarios (e.g. an image with the sky at the bottom and the ground at the top) and it generalizes better. A new way of doing data augmentation, which has to our knowledge never been tested, could be to compress and decompress some images of the dataset in a lossy manner. For a lot of codecs the lossy compression is associated with a blurring, the loss of details or the addition of artifacts. All these effects will modify the image appearance but generally (and depending on the compression rate) they will conserve the main image components necessary to recognize and separate objects.

5.2.2 Compressed dataset

Another axe of research that could be explored is the impact of compression for dataset size reduction. In chapter 3 we first trained our network on uncompressed data and then fine-tuned it with the same training set compressed at a certain ratio. This technique requires the storage of the full quality dataset and it suffers from the memory cost that comes with it. In order to reduce the memory used by the dataset, we can investigate how the performance would be affected by a training set composed only of compressed images. Furthermore, we could explore the idea of a dataset consisting of compressed and uncompressed images, so that there would be no repetition between compressed and uncompressed images. In such a configuration, the images would have different utilities during the training. The uncompressed one would enable the use of details in the feature maps while the compressed ones would extend the number of scenarios for features depending on coarse elements.

5.2.3 A new metric for image quality

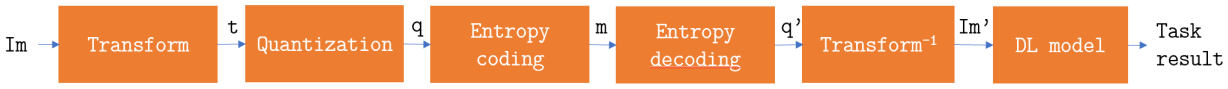
One important problem in image processing is the evaluation of image quality. Usually, we do not really measure the quality of an image but we measure the distortion between two images. Indeed, it is easier to find a difference of quality with a referential than to compute an absolute quality value which would be hard

to define. For this reason, metrics like the MSE and PSNR were developed. They rely on the difference between pixels but their mathematical definition does not take into account image-related variables like contrasts, edges and pixel structures. In order to solve this problem, the structural similarity (SSIM) was proposed. It relies on image mean, variance and covariance and it is able to perceive change in structural information but it remains a purely statistical tool. All these objective metrics were created in order to be related to the HVS which is subjective and psychophysical,and therefore is problematic to model. Indeed, the HVS depends on complex biological processes and it is biased in the way that it treats frequencies and colors, but it is still the main target of image processing techniques as most images are intended for humans. Let us now think about what the HVS is used for in daily life: object recognition and detection, semantic segmentation, action recognition, ... We realize that a lot of these tasks are now at least partially achievable by computer vision techniques and DL. Thus, in order to obtain a more general quality metric that would be closer to the information contained in the image and correlate to the aims of the HVS, we could use the performance (or difference of performance) on these DL tasks. This new metric would have the advantage of being fully objective and measurable unlike the HVS perception quality, and it would be fully related to the information contained in the image. On the other hand, this idea has some drawbacks,

- ML algorithms are often based on a dataset with a limited number of classes. This creates a limitation concerning the images that does not contain any of these classes.
- ML algorithms have generally one specific application (classification, segmentation, ...) and each will be impacted differently by distortion. There is hence a difficulty concerning the choice of selecting a set of these applications and concerning their fusion in one metric.
- ML algorithms are imperfect and unpredictable, it could hence happen that a distortion leads to an improvement of the performance. Thus, in order to have viable metrics, the computation should not be performed on individual images but on image batches that would offer statistical stability.
- This metric would not be bound to the HVS but to algorithm aims, meaning that an image with a large visual distortion, but also which does not affect the algorithms, could still be judged as having a good quality.

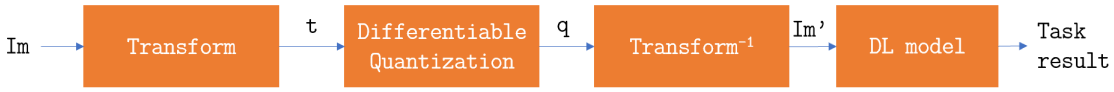
5.2.4 End-to-end learning

In section 5.1.3 we develop the idea of a cyclic optimization of the compression standard and the CNN, this cyclic process is necessary as the simultaneous training



$$Loss = size(m) + \beta\, error(\texttt{Task result})$$

Figure 5.2: End-to-end model with non-differentiable operations and its corresponding loss, the transform and transform^{-1} represent the operations which aims at reducing the entropy (DCT in JPEG, DWT in JPEG2000 or CNN for autoencoders).



$$Loss = entropy(q) + \beta\, error(\texttt{Task result})$$

Figure 5.3: End-to-end model with differentiable only operations and its corresponding loss.

and optimization of these two blocks is hard to conceive. A way to solve this problem could be to use an end-to-end architecture that would consist of one block (containing an encoder, a decoder and a CNN). Then, we could do an end-to-end training of the parameters using differentiation and back-propagation.

A problem arises from such models: most codecs contain an entropy coding and a quantization step which are non-differentiable operations, and that prevents the use of the back-propagation. Figure 5.2 contains a rough representation of a compression standard followed by a DL task. In this configuration, it is impossible to derive the loss function and propagate the back-propagation for the parameters of the first transform and of the quantization step, this prevents the end-to-end training of all the parameters at once. In order to solve this issue, we propose the model on figure 5.3 where no entropy coding is performed and where the compression measurement of the loss function is replaced by an entropy estimation which, if the entropic coder is efficient, corresponds to the compression measurement. Furthermore, in this model the hard quantization is replaced by a differentiable version; different techniques were developed in the autoencoder literature based on continuous approximations or stochastic methods [46, 47]. Such an approach can be fully optimized with one end-to-end training and might hence produce better results than the one presented in this master thesis. This method was partially used in [28] and has shown encouraging results.

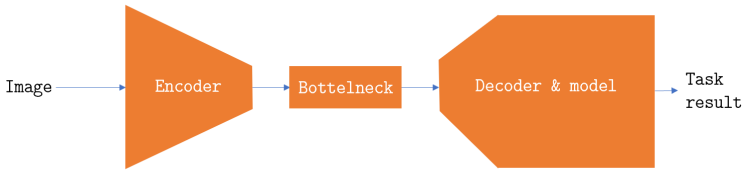


Figure 5.4: Autoencoder model where the decoder is merged with another architecture to directly perform a task.

5.2.5 Autoencoder, compression and DL tasks

If we build on the end-to-end model described in figure 5.3 and we think of the specific case of the autoencoder (where the transform block is a contraction path with convolutions and poolings and the transform^{-1} block is the inverse path consisting of up-convolutions). We find that the transform^{-1} and the DL model can be of the same nature if the DL model is a CNN. Hence, it is possible to consider a simpler representation where these two blocks merge into one DL block responsible for both the decoding and the task at hand. Such a configuration is represented on figure 5.4, where the bottleneck contains a low dimensional or "compressed" representation of the image. If this type of network is trained for a task, it is sure that the image will not be reconstructed anywhere in the network because the network will directly use the deep features to create the expected task result. This model has the advantage of fully combining the dimensional reduction of machine learning and the entropy reduction of the compression at the encoder side, and at the decoder it processes this advantageous low dimension and semantically rich representation to reach its goal. This configuration can reduce the computational cost and accelerate the processing because the image does not have to be reconstructed and the model does not have to create deep features, thus the whole complexity of the Decoder and DL model can be reduced.

5.2.6 A new compression limit

In classical image compression the limit of size is known to be the entropy but in configuration such as that shown in figure 5.4 this limit of compression has to be redefined. For this model, it is no longer the image that we compress but the deep features of this image which are used as discriminant information for a specific task. This nuance has its importance as generally the information necessary to make a decision in an image is smaller than the size of the image itself. Thus, it is possible to compress the deep features at a higher level than the image itself without affecting the DL analysis. Nevertheless, a new question remains because this specific case must also have a compression limit, and we found this limit to be

the entropy of the task result itself. In other words, **the limit of compression for the information necessary to answer a question is the answer itself**. For example, in an image classification problem where the model has to make a binary decision on whether there is a cat or not in the image, then the maximal compression of the image information necessary to decide if there is a cat of not in the image is 1 bit, the maximal entropy value of a binary pdf corresponding to uniformly distributed dataset. Of course, in this case, if the encoder reaches this limit, it means that it has also found the answer and it is the same for any other task or question, the encoder can always find the response and then the decoder just has to represent it in the requested format. Of course, a major constraint which has not been discussed so far and that can prevent the encoder from reaching its entropy limit is the limited complexity of this encoder. Indeed, usually the compression takes place right after the image acquisition on end devices with a limited computational and memory capacity. Due to this complexity limitation for most cases, the computation load of finding the solution will be shared between the encoder block and the decoder model block, and the compression limit will not be reached.

Chapter 6

Conclusion

For the moment, compression is mainly done with the HVS in mind and most DL networks are not made to be used with compressed images. In this master thesis, we wanted to optimize the image compression process for deep learning and the deep learning techniques for compressed images. The main purpose being to compress images as much as possible while maintaining their performance on DL tasks.

To achieve this, we started by doing a state of the art review of the research on the subject where multiple approaches were investigated. Then, in a first experimental phase, we showed that by selecting the right network architecture, compression standard and network training the performance of a medical segmentation task could be doubled for some high-compression ratios. In a second experimental phase, we showed that we could improve the performance of classification by up to 3% for some rates by optimizing the JPEG quantization tables with a Lagrangian or genetic algorithm.

In the previous chapter, we gave some leads on how to pursue this work by using compression and deep learning specificities and similarities. Indeed, the dataset size reduction, the use of compression for data augmentation, or the end-to-end training of model doing both compression and deep learning, open multiple ways for further promising works. Finally, we found that when doing compression for a specific DL task, the limit of compression is not the image entropy anymore but the task result entropy. The boundary between compression and feature extraction disappears, leaving a new paradigm where compression and image processing become one.

Acknowledgements

First, I would like to thank professor Benoit Macq for his support during the whole academic year, our regular meetings were a motivating source of advancement and he was always available for any concern I might have had. Furthermore, his research knowledge and creativity was a motor for the conception and development of this master thesis.

Secondly, I would like to thank Karim El Khoury and Eliott Brion, which assisted and helped me when I faced difficulties. More specifically, I thank Karim for his constant supervision, presence and proactivity and Eliott for his precious help concerning the deep-learning and his encouragements to be rigorous in the research process.

Thirdly, I would like to thank the learning-jpeg research group, an informal research group composed of UCLouvain and intoPIX research engineers which provided me with great feedbacks during our meetings.

Finally, I thank the CT and CBCT dataset providers, the CHU-UCL-Namur (Dr. Vincent Remouchamps) and CHU-Charleroi (Dr. Nicolas Meert), Sara Teruel Rivas for the data acquisition and annotation and Dr. Geneviève Van Ooteghem for the validation of the contours.

This master thesis gave me the chance of exploring a large question that inspired me. I liked the opportunity of reading the discoveries and thoughts of many researchers around the glob. Then, I liked confronting these ideas with my promoter, other researchers and my own opinion before experimentally try them by myself. Finally I liked that this subject could have a real-life implication with environmental benefits.

Appendix A

3D and 2D network SNR after convolution

In this appendix, we attempt to explain the compression robustness of 3D-networks compared to 2D-networks. To do so, we will use a stochastic representation of the images as if it was a signal, we will consider the artifacts created by the compression as an noise and we will compute the signal-to-noise ratio for the 2 networks after the first convolution.

- **2D-network SNR** If we consider a slice $s[n]$ which is the result of the compression of a slice $r[n]$ where an artifact or noise $n[n]$ was added. We can write the expression of the slice as

$$s_{i,j}[n] = r_{i,j}[n] + n_{i,j}[n]$$

where i and j are de index in the slice. If we let this slice go through the first layer of a CNN it will undergo a convolution with a kernel containing coefficients $a_{i,j}$ and give a new signal $X[n]$,

$$X[n] = \sum_j \sum_i a_{i,j} s_{i,j}[n] = \sum_j \sum_i a_{i,j} r_{i,j}[n] + \sum_j \sum_i a_{i,j} n_{i,j}[n].$$

We set for easier representation:

$$p[n] = \sum_j \sum_i a_{i,j} r_{i,j}[n]$$

and

$$b[n] = \sum_j \sum_i a_{i,j} n_{i,j}[n]$$

.

If we now consider this signal as stochastic, we can compute the power of the signal as being its variance,

$$\sigma^2_{X[n]} = E[(X[n] - E[X[n]])^2]$$

and by making the hypothesis that it is zero-mean it becomes,

$$\sigma^2_{X[n]} = E[X[n]^2]$$

and if the noise is independent from the image then

$$\sigma^2_{X[n]} = E[p[n]^2 + 2p[n]b[n] + b[n^2]] = E[p[n]^2 + b[n]^2]] = E[p[n]^2] + E[b[n]^2].$$

We then have a separation between the noise power $\sigma^2_n = E[b[n]^2]$ and the signal power $\sigma^2_{sig} = E[p[n]^2]$. We get

$$\begin{aligned}
\sigma^2_{sig} = E[p[n]^2] &= E[\sum_j \sum_i a_{i,j} r_{i,j}[n] \cdot \sum_{j'} \sum_{i'} a_{i',j'} r_{i',j'}[n]] \\
&= \sum_j \sum_i \sum_{j'} \sum_{i'} a_{i,j} a_{i',j'} E[r_{i,j}[n] r_{i',j'}[n]] \\
&= \sum_{(i,j)=(i',j')} a^2_{i,j} \sigma^2_r + \sum_{(i,j)\neq(i',j')} a_{i,j} a_{i',j'} Cov(r_{i,j}[n], r_{i',j'}[n])
\end{aligned}$$

where σ^2_r is the power of the input signal r and $Cov(r_{i,j}[n], r_{i',j'}[n])$ is the spatial covariance of this input signal. We get

$$\begin{aligned}
\sigma^2_n = E[b[n]^2] &= E[\sum_j \sum_i a_{i,j} n_{i,j}[n] \cdot \sum_{j'} \sum_{i'} a_{i',j'} n_{i',j'}[n]] \\
&= \sum_j \sum_i \sum_{j'} \sum_{i'} a_{i,j} a_{i',j'} E[n_{i,j}[n] n_{i',j'}[n]] \\
&= \sum_{(i,j)=(i',j')} a^2_{i,j} \sigma^2_n + \sum_{(i,j)\neq(i',j')} a_{i,j} a_{i',j'} Cov(n_{i,j}[n], n_{i',j'}[n])
\end{aligned}$$

where σ^2_n is the noise power and $Cov(n_{i,j}[n], n_{i',j'}[n])$ is the spatial covariance of the noise.

Based on this the SNR becomes

$$SNR_{2D} = \frac{\sigma^2_{X[n]}}{\sigma^2_n} = 1 + \frac{\sigma^2_{sig}}{\sigma^2_n} = 1 + \frac{\sum_{(i,j)=(i',j')} a^2_{i,j} \sigma^2_r + \sum_{(i,j)\neq(i',j')} a_{i,j} a_{i',j'} Cov(r_{i,j}[n], r_{i',j'}[n])}{\sum_{(i,j)=(i',j')} a^2_{i,j} \sigma^2_n + \sum_{(i,j)\neq(i',j')} a_{i,j} a_{i',j'} Cov(n_{i,j}[n], n_{i',j'}[n])} \tag{A.1}$$

and if the noise is spatially decorrelate

$$Cov(n_{i,j}[n], n_{i',j'}[n]) = 0$$

then

$$SNR_{2D} = \frac{\sigma^2_{X[n]}}{\sigma^2_n} = 1 + \frac{\sigma^2_{sig}}{\sigma^2_n} = 1 + \frac{\sum\limits_{(i,j)=(i',j')} a^2_{i,j}\sigma^2_r + \sum\limits_{(i,j)\neq(i',j')} a_{i,j}a_{i',j'}Cov(r_{i,j}[n], r_{i',j'}[n])}{\sum\limits_{(i,j)=(i',j')} a^2_{i,j}\sigma^2_n}. \tag{A.2}$$

- **3D-network SNR** We can make an analog development considering this time the signal as being 3 dimensional and using three dimensional convolution. In this case the fix slice n becomes the slice n+t where t can vary.

$$SNR_{3D} = 1 + \frac{\sum\limits_{(i,j,t)=(i',j',t')} a^2_{i,j}\sigma^2_r + \sum\limits_{(i,j,t)\neq(i',j',t')} a_{i,j}a_{i',j'}Cov(r_{i,j}[n+t], r_{i',j'}[n+t'])}{\sum\limits_{(i,j,t)=(i',j',t')} a^2_{i,j}\sigma^2_n + \sum\limits_{(i,j,t)\neq(i',j',t')} a_{i,j}a_{i',j'}Cov(n_{i,j}[n+t], n_{i',j'}[n+t'])} \tag{A.3}$$

and if the noise is spatially decorrelate

$$Cov(n_{i,j}[n+t], n_{i',j'}[n+t']) = 0$$

then

$$SNR_{3D} = 1 + \frac{\sum\limits_{(i,j,t)=(i',j',t')} a^2_{i,j,t}\sigma^2_r + \sum\limits_{(i,j,t)\neq(i',j',t')} a_{i,j}a_{i',j',y'}Cov(r_{i,j}[n+t], r_{i',j'}[n+t'])}{\sum\limits_{(i,j,t)=(i',j',t')} a^2_{i,j,t}\sigma^2_n}. \tag{A.4}$$

If we consider equations A.1 and A.3 and choose kernels of size 3x3 and 3x3x3 for the 2D and 3D network, if we neglect the kernel coefficients by putting $a_*a_* = 1$ and if we set the noise and signals powers and covariance to be constant, these equations become:

$$SNR_{2D} = 1 + \frac{9\sigma^2_r + 27Cov(r)}{9\sigma^2_n + 27Cov(n)} \quad \text{and} \quad SNR_{3D} = 1 + \frac{27\sigma^2_r + 702Cov(r)}{27\sigma^2_n + 702Cov(n)}.$$

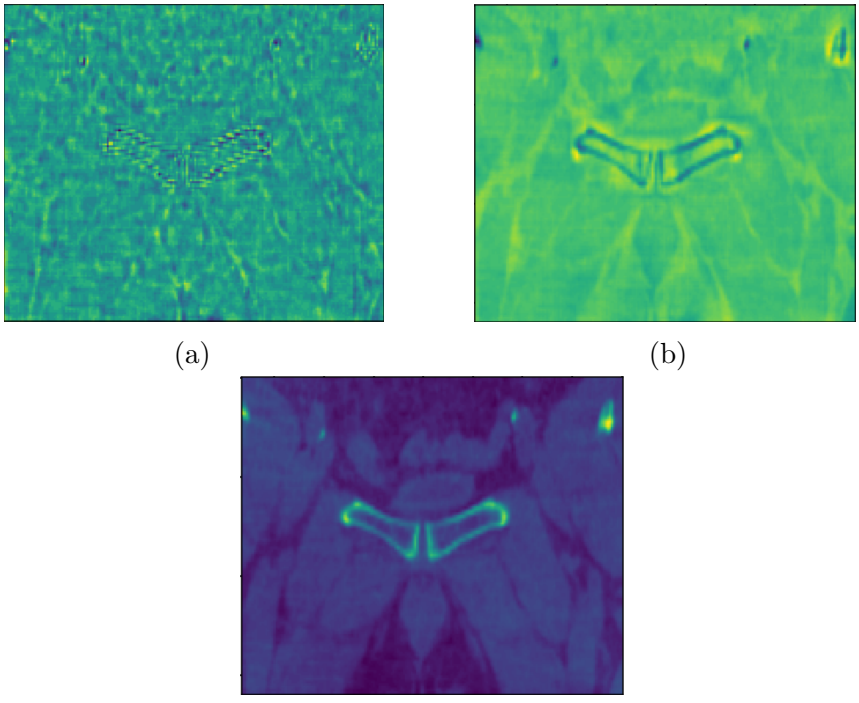
Furthermore if the variances and covariances are equal in 2D and 3D we can wonder under which conditions the SNR of the 3D is better than the one in 2D :

$$SNR_{2D} < SNR_{3D},$$

after substitution and a few arithmetic operations and knowing that each element is positive we ind the following condition for the superiority of the 3D-network:

$$\frac{Cov(n)}{\sigma_n^2} < \frac{Cov(r)}{\sigma_r^2}$$

This is an interesting conclusion, because the condition is that the signal spatial correlation normalized by the signal power has to be higher than the noise spatial correlation normalized by the noise power. Remembering that the signal is here a natural image, we can suppose that adjacent pixels should be highly correlated while noise tends by nature to be decorrelate. This explanation has to be taken with caution as it is the result of many hypothesis and an oversimplification of the problem. Indeed, considering the pixels/voxels of the images as being zero-mean is a strong assumption and it can be shown that the image-noise correlation is not negligible for high compression ratios (in figure A.1 we can clearly see the same elements in the noise at ratio 96:1 and in the original image). Furthermore, we consider here the SNR after one convolution, the U-Net architecture is way more complex due to its high depth and then the relation SNR-performance is not trivial.



(a) (b)

(c)

Figure A.1: Original image (c) and noise for a compression ratio of 32:1 (a) and 96:1 (b).

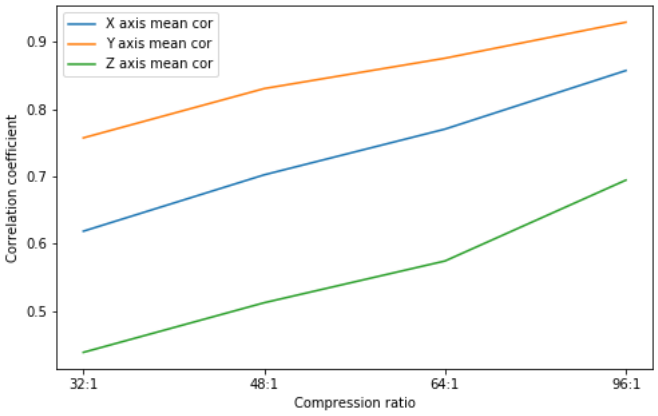


Figure A.2: Mean inter-slice noise correlation at different compression ratio and in the three directions.

Another observation that could explain the superiority of 3D over 2D U-Net is the difference in noise correlation for the different spatial dimensions. Each volume is compressed slice by slice and hence, each noise is produced slice by slice. Then, we could expect from the noise to be more decorrelated in the third dimension. To experimentally show this, we use the inter-slice correlation as defined in equation A.5.

$$cor = \frac{\sum_m \sum_n (f(m,n) - \bar{f}).(g(m,n) - \bar{g})}{\sqrt{\sum_m \sum_n (f(m,n) - \bar{f})^2 . \sum_m \sum_n (f(m,n) - \bar{f})^2}} \tag{A.5}$$

Figure A.2 contains the evolution of this coefficient at different compression ratios and in the three directions. We observe a higher correlation in the y and x direction which describe the plan in which the images were compressed while there is a lower correlation in the z direction. If we get back to equation A.3, we observe that the SNR_{3D} depends on the noise and signal spatial covariance in the 3 dimensions. While the signal will keep a relative equivalent covariance in the different directions figure A.2 shows that the noise correlation (and hence to some extent covariance) is lower in the z direction. Thus this lower noise correlation improves the SNR_{3D} .

Appendix B

JPEG and JPEG2000 mean compression size

Ratio	Expected size	JPEG2000 size	JPEG	
			Quality factor	Size
8:1	3840	3730	91	4026
16:1	1920	1883	67	1944
24:1	1280	1269	35	1307
32:1	960	958	17	979
48:1	640	646	1	719
64:1	480	489	/	/
96:1	320	332	/	/
128:1	240	253	/	/

Table B.1: Compression sizes for a compressed slice in bytes.

Appendix C

Experiment result tables

Bladder		Compression Ration						
		1	24	32	48	64	96	128
CT	2D DSC	0.9187	0.9141	0.9024	0.8728	0.8098	0.5783	0.301
	Std	0.03932	0.03817	0.0428	0.0671	0.088	0.1826	0.2602
	3D DSC	0.9161	0.9129	0.9022	0.885	0.8599	0.7581	0.4097
	Std	0.0334	0.0348	0.0432	0.0601	0.0765	0.122	0.2912
	2D SMBD	2.0405	2.2515	2.407	3.1574	4.6631	7.2049	10.0828
	Std	1.4042	1.2801	1.1705	1.455	2.5465	3.5983	2.7415
	3D SMBD	1.4453	1.5467	1.831	2.083	2.4582	3.9016	8.2059
	Std	0.5792	0.6155	0.8068	1.1235	1.3563	1.9957	4.1868
CBCT	2D DSC	0.8559	0.8426	0.8059	0.7483	0.6518	0.3474	0.1753
	Std	0.0681	0.0718	0.0741	0.0733	0.0782	0.0775	0.0962
	3D DSC	0.8322	0.825	0.8082	0.7807	0.741	0.651	0.4509
	Std	0.1187	0.123	0.129	0.1389	0.1418	0.118	0.1334
	2D SMBD	2.8838	3.0158	3.4905	4.1213	4.9126	7.9894	13.2038
	Std	1.8273	1.7022	1.6994	1.6625	1.3364	2.4664	4.2513
	3D SMBD	2.5822	2.6953	2.9558	3.3616	3.8574	4.9021	7.8118
	Std	1.9630	2.0333	2.1679	2.3644	2.3719	1.9205	2.1518

Table C.1: Table exp1 bladder

Rectum		Compression Ration						
		1	24	32	48	64	96	128
CT	2D DSC	0.7515	0.7556	0.7287	0.6685	0.5769	0.3705	0.1359
	Std	0.0932	0.0909	0.0919	0.1141	0.1597	0.1558	0.0919
	3D DSC	0.7747	0.7727	0.7732	0.7523	0.7099	0.6170	0.3731
	Std	0.0680	0.0669	0.0616	0.0628	0.0637	0.0948	0.1866
	2D SMBD	3.7605	3.5372	3.4026	3.6466	3.8751	5.2961	9.8117
	Std	1.0292	1.0278	0.9774	1.2947	1.5554	2.1288	4.3740
	3D SMBD	2.6539	2.6422	2.6023	2.8625	3.1970	4.3224	8.9179
	Std	1.0292	1.0278	0.9774	1.2947	1.5554	2.1288	4.3740
CBCT	2D DSC	0.7801	0.7562	0.7082	0.6419	0.5244	0.2836	0.0718
	Std	0.07165	0.0838	0.1065	0.1149	0.1326	0.1474	0.0620
	3D DSC	0.7660	0.7656	0.7480	0.7113	0.6478	0.4718	0.2723
	Std	0.0735	0.0701	0.0789	0.0974	0.1201	0.1141	0.1357
	2D SMBD	2.7309	2.8378	3.0572	3.2231	3.6900	4.9779	8.8025
	Std	1.0856	1.0885	1.1606	1.0731	1.0223	1.1600	2.4254
	3D SMBD	2.3487	2.3083	2.4133	2.6049	3.2451	4.9352	9.4040
	Std	0.8678	0.8029	0.8235	0.9521	1.5938	1.6895	2.8379

Table C.2: Table exp1 rectum

Bladder		Compression Ration						
		1	24	32	48	64	96	128
CT	3D FT DSC	0.9161	0.9006	0.8930	0.8871	0.8701	0.8343	0.7029
	Std	0.0334	0.0410	0.0576	0.0508	0.0736	0.0724	0.1550
	3D DSC	0.9161	0.9129	0.9022	0.885	0.8599	0.7581	0.4097
	Std	0.0334	0.0348	0.0432	0.0601	0.0765	0.122	0.2912
	3D FT SMBD	1.4452	1.5335	1.7508	1.8956	2.1674	2.6751	4.7017
	Std	0.5792	0.5548	1.0867	0.9418	1.3953	1.3303	3.0330
	3D SMBD	1.4453	1.5467	1.831	2.083	2.4582	3.9016	8.2059
	Std	0.5792	0.6155	0.8068	1.1235	1.3563	1.9957	4.1868
CBCT	3D FT DSC	0.8322	0.8562	0.8378	0.8355	0.8025	0.7411	0.6528
	Std	0.1187	0.0952	0.1029	0.0928	0.1145	0.0979	0.1201
	3D DSC	0.8322	0.825	0.8082	0.7807	0.741	0.651	0.4509
	Std	0.1187	0.123	0.129	0.1389	0.1418	0.118	0.1334
	3D FT SMBD	2.5822	2.1853	2.5299	2.5375	2.9713	3.5701	4.7891
	Std	1.9631	1.5482	1.7828	1.5983	1.9765	1.5779	1.8158
	3D SMBD	2.5822	2.6953	2.9558	3.3616	3.8574	4.9021	7.8118
	Std	1.9630	2.0333	2.1679	2.3644	2.3719	1.9205	2.1518

Table C.3: Table exp2 bladder

Rectum		Compression Ration						
		1	24	32	48	64	96	128
CT	3D FT DSC	0.7747	0.7625	0.7630	0.7398	0.7299	0.6674	0.5166
	Std	0.0681	0.0827	0.0732	0.0826	0.0667	0.0693	0.1581
	3D DSC	0.7747	0.7727	0.7732	0.7523	0.7099	0.6170	0.3731
	Std	0.0680	0.0669	0.0616	0.0628	0.0637	0.0948	0.1866
	3D FT SMBD	2.6539	3.2365	3.2723	3.4678	3.2393	4.0389	4.6026
	Std	1.0292	2.1109	2.0581	2.0748	1.5520	1.4070	1.9032
	3D SMBD	2.6539	2.6422	2.6023	2.8625	3.1970	4.3224	8.9179
	Std	1.0292	1.0278	0.9774	1.2947	1.5554	2.1288	4.3740
CBCT	3D FT DSC	0.7660	0.7781	0.7633	0.7563	0.7297	0.6498	0.5724
	Std	0.07356	0.07975	0.0854	0.0802	0.0953	0.1069	0.1305
	3D DSC	0.7660	0.7656	0.7480	0.7113	0.6478	0.4718	0.2723
	Std	0.0735	0.0701	0.0789	0.0974	0.1201	0.1141	0.1357
	3D FT SMBD	2.3487	2.3881	2.6448	2.8077	2.6143	3.2613	3.9209
	Std	0.8678	1.0941	1.1705	1.1844	0.9028	0.9038	0.8679
	3D SMBD	2.3487	2.3083	2.4133	2.6049	3.2451	4.9352	9.4040
	Std	0.8678	0.8029	0.8235	0.9521	1.5938	1.6895	2.8379

Table C.4: Table exp2 rectum

Rectum		Compression Ration						
		1	24	32	48	64	96	128
CT	3D FT 48 DSC	0.74026	0.7413	0.7449	0.7398	0.7255	0.6533	0.3897
	Std	0.0862	0.0825	0.07914	0.0826	0.0821	0.0777	0.1545
	3D FT 64 DSC	0.7402	0.7401	0.7424	0.7375	0.7299	0.6753	0.4168
	Std	0.07	0.0674	0.0627	0.0633	0.0667	0.0676	0.1554
	3D FT 96 DSC	0.7197	0.7173	0.7162	0.7108	0.7031	0.6674	0.4839
	Std	0.0677	0.0683	0.0663	0.0636	0.0621	0.0693	0.1553
	3D FT 48 SMBD	3.4415	3.3493	3.2913	3.4678	3.5999	4.4547	10.3377
	Std	1.7392	1.7575	1.9110	2.0748	2.1699	2.2698	4.4771
	3D FT 64 SMBD	3.231	3.1671	3.1115	3.2128	3.2393	4.0133	9.5047
	Std	1.08063	1.0996	1.1216	1.2056	1.5520	1.8549	4.1064
	3D FT 96 SMBD	3.1251	3.1147	3.1176	3.1093	3.4139	4.0389	7.6373
	Std	0.8726	0.9192	0.8770	0.8563	0.9858	1.4070	3.0802
CBCT	3D FT 48 DSC	0.7589	0.7662	0.7648	0.75631	0.7318	0.6071	0.3858
	Std	0.0990	0.0871	0.0797	0.0802	0.0863	0.1123	0.1285
	3D FT 64 DSC	0.7436	0.7481	0.7489	0.7426	0.7297	0.6381	0.4444
	Std	0.1094	0.1008	0.0915	0.0927	0.0953	0.1168	0.1309
	3D FT 96 DSC	0.7003	0.7006	0.6994	0.6983	0.6820	0.64981	0.5381
	Std	0.1073	0.1043	0.0996	0.0966	0.0996	0.1069	0.1286
	3D FT 48 SMBD	2.7348	2.6089	2.5821	2.8077	2.7080	4.7891	10.6715
	Std	1.2725	1.2221	1.1767	1.1844	1.0406	2.9934	3.9406
	3D FT 64 SMBD	2.7382	2.6585	2.5956	2.7541	2.6143	3.5524	8.5252
	Std	1.2507	1.1697	1.1024	1.0641	0.9028	1.0372	2.6398
	3D FT 96 SMBD	3.17966	3.1682	3.1200	3.0502	3.1215	3.2613	5.8803
	Std	1.0956	1.0787	1.05	0.9549	0.9276	0.9038	1.5358

Table C.5: Table exp3 rectum

Appendix D

CIFAR-10 Network

The network used in the JPEG compression optimization originates from `https://keras.io/examples/cifar10_cnn/`, it consits of 4 convolution layers with one maxpooling and with relu activation functions, one fully connected layer with relu activation functions and it ends with a softmax to obtain a vector of length 10 with values between 0 and 1. The training used 50.000 of 60.000 images of the dataset with an ADAM optimizer for 100 epochs and a learning rate of 10^{-4}.

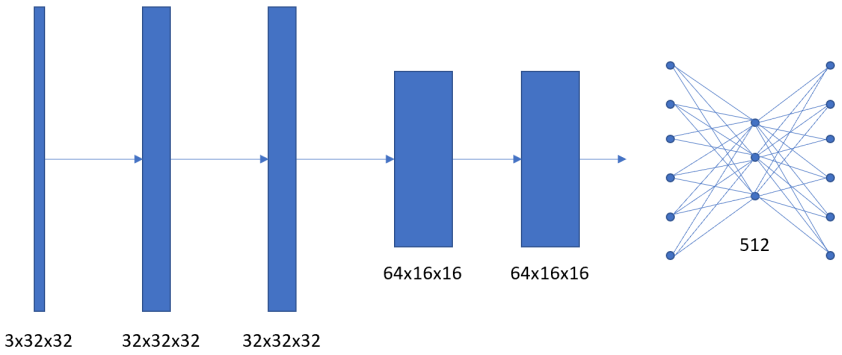


Figure D.1: CIFAR10 network simplified scheme

Bibliography

[1] *Cisco Annual Internet Report (2018–2023)*. Cisco. 2020.

[2] Thomas Barnett Jr. et al. *Cisco Visual Networking Index (VNI) Complete Forecast Update, 2017–2022*. Cisco. 2017.

[3] https://aws.amazon.com/rekognition/. *Amazon Rekognition*. Amazon. 2020.

[4] https://www.faceplusplus.com/. *Face++ Cognitive Services*. Face++. 2020.

[5] https://ai.baidu.com/. *Baidu AI Open Platform*. Baidu. 2020.

[6] https://www.sensetime.com/. *Sense Time*. 2020.

[7] https://cloud.google.com/vision/. *IA Vision*. Google. 2020.

[8] Jean Léger et al. "Cross-domain data augmentation for deep-learning-based male pelvic organ segmentation in cone beam CT". In: *Applied Sciences* 10.3 (2020), p. 1154.

[9] Eliott Brion et al. "Using planning CTs to enhance CNN-based bladder segmentation on cone beam CT". In: *Medical Imaging 2019: Image-Guided Procedures, Robotic Interventions, and Modeling*. Vol. 10951. International Society for Optics and Photonics. 2019, p. 109511M.

[10] Farhad Ghazvinian Zanjani et al. "Impact of JPEG 2000 compression on deep convolutional neural networks for metastatic cancer detection in histopathological images". In: *Journal of Medical Imaging* 6 (Apr. 2019), p. 1. DOI: 10.1117/1.JMI.6.2.027501.

[11] Kathleen Steinhöfel et al. "Classification of compressed DICOM liver tissue images". In: (Feb. 2002), pp. 186–187. DOI: 10.1109/MCTE.2002.1175067.

[12] Robert Ives et al. "Effects of image compression on iris recognition performance and image quality". In: May 2009, pp. 16–21. DOI: 10.1109/CIB.2009.4925681.

[13] Kresimir Delac, Mislav Grgic, and Sonja Grgic. "Effects of JPEG and JPEG2000 compression on face recognition". In: vol. 3687. Aug. 2005, pp. 136–145. DOI: 10.1007/11552499_16.

[14] Zihao Liu et al. “DeepN-JPEG: a deep neural network favorable JPEG-based image compression framework”. In: June 2018, pp. 1–6. DOI: 10.1145/3195970.3196022.

[15] Hongshan Li et al. “AdaCompress: Adaptive Compression for Online Computer Vision Services”. In: Oct. 2019, pp. 2440–2448. ISBN: 978-1-4503-6889-6. DOI: 10.1145/3343031.3350874.

[16] Shumeet Baluja, David Marwood, and Nicholas Johnston. “Task-specific color spaces and compression for machine-based object recognition”. In: 2019.

[17] L. Duan et al. “Overview of the MPEG-CDVS Standard”. In: *IEEE Transactions on Image Processing* 25.1 (2016), pp. 179–194.

[18] L. Duan et al. “Compact Descriptors for Video Analysis: The Emerging MPEG Standard”. In: *IEEE MultiMedia* 26.2 (2019), pp. 44–54.

[19] Ling-Yu Duan et al. “Video Coding for Machines: A Paradigm of Collaborative Compression and Intelligent Analytics”. In: (Jan. 2020).

[20] Dan Fu and Gabriel Guimaraes. *Using compression to speed up image classification in artificial neural networks*. 2016.

[21] Matej Ulicny and Rozenn Dahyot. “On using cnn with dct based image data”. In: *Irish Machine Vision and Image Processing Conference*. Vol. 2. 2017.

[22] Lionel Gueguen et al. “Faster Neural Networks Straight from JPEG”. In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio et al. Curran Associates, Inc., 2018, pp. 3933–3944. URL: http://papers.nips.cc/paper/7649-faster-neural-networks-straight-from-jpeg.pdf.

[23] Max Ehrlich and Larry S. Davis. “Deep Residual Learning in the JPEG Transform Domain”. In: *CoRR* abs/1812.11690 (2018). arXiv: 1812.11690. URL: http://arxiv.org/abs/1812.11690.

[24] Benjamin Deguerre, Clément Chatelain, and Gilles Gasso. “Fast object detection in compressed JPEG Images”. In: *CoRR* abs/1904.08408 (2019). arXiv: 1904.08408. URL: http://arxiv.org/abs/1904.08408.

[25] Eunhee Kang, Junhong Min, and Jong Chul Ye. “A deep convolutional neural network using directional wavelets for low-dose X-ray CT reconstruction”. In: *Medical physics* 44.10 (2017), e360–e375.

[26] Travis Williams and Robert Li. “Advanced image classification using wavelets and convolutional neural networks”. In: *2016 15th IEEE international conference on machine learning and applications (ICMLA)*. IEEE. 2016, pp. 233–239.

[27] Shin Fujieda, Kohei Takayama, and Toshiya Hachisuka. "Wavelet convolutional neural networks for texture classification". In: *arXiv preprint arXiv:1707.07394* (2017).

[28] Robert Torfason et al. "Towards image understanding from deep compression without decoding". In: *arXiv preprint arXiv:1803.06131* (2018).

[29] Ang Li, Yiwei Lu, and Yang Wang. "Semantic Segmentation in Compressed Videos". In: Sept. 2019, pp. 1–5. DOI: 10.1109/MMSP.2019.8901826.

[30] Chao-Yuan Wu et al. "Compressed Video Action Recognition". In: June 2018, pp. 6026–6035. DOI: 10.1109/CVPR.2018.00631.

[31] Wang Shiyao et al. "Fast Object Detection in Compressed Video". In: (Nov. 2018).

[32] Vadim Kantorov and Ivan Laptev. "Efficient feature extraction, encoding and classification for action recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 2593–2600.

[33] William Scarfe, Allan Farman, and Predag Sukovic. "Clinical Applications of Cone-Beam Computed Tomography in Dental Practice". In: *Journal (Canadian Dental Association)* 72 (Mar. 2006), pp. 75–80.

[34] zhixuhao. *Unet for images*. https://github.com/zhixuhao/unet. 2017.

[35] Eliott Brion. *Pelvis segmentation*. https://github.com/eliottbrion/pelvis_segmentation. 2020.

[36] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[37] T. Kim et al. "SF-CNN: A Fast Compression Artifacts Removal via Spatial-To-Frequency Convolutional Neural Networks". In: *2019 IEEE International Conference on Image Processing (ICIP)*. 2019, pp. 3606–3610.

[38] Pavel Svoboda et al. "Compression Artifacts Removal Using Convolutional Neural Networks". In: *Journal of WSCG* 24 (May 2016), pp. 63–72.

[39] Benoit Brummer and Christophe De Vleeschouwer. "Adapting JPEG XS gains and priorities to tasks and contents". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2020.

[40] Alex Krizhevsky, Geoffrey Hinton, et al. "Learning multiple layers of features from tiny images". In: (2009).

[41] Dong-U Lee et al. "Energy-Efficient Image Compression for Resource-Constrained Platforms". In: *Image Processing, IEEE Transactions on* 18 (Oct. 2009), pp. 2100–2113. DOI: 10.1109/TIP.2009.2022438.

[42] Bohyoung Kim et al. “JPEG2000 3D compression vs 2D compression: An assessment of artifact amount and computing time in compressing thin-section abdomen CT images”. In: *Medical physics* 36.3 (2009), pp. 835–844.

[43] P Gabriel Peterson et al. “Extreme compression for extreme conditions: Pilot study to identify optimal compression of CT images using MPEG-4 video compression”. In: *Journal of digital imaging* 25.6 (2012), pp. 764–770.

[44] Nikolaus Hansen and Andreas Ostermeier. “Completely derandomized self-adaptation in evolution strategies”. In: *Evolutionary computation* 9.2 (2001), pp. 159–195.

[45] Max Hopkins, Michael Mitzenmacher, and Sebastian Wagner-Carena. “Simulated Annealing for JPEG Quantization”. In: *2018 Data Compression Conference*. IEEE. 2018, pp. 412–412.

[46] Eirikur Agustsson et al. *Soft-to-Hard Vector Quantization for End-to-End Learning Compressible Representations*. 2017. arXiv: 1704.00648 [cs.LG].

[47] George Toderici et al. “Variable rate image compression with recurrent neural networks”. In: *arXiv preprint arXiv:1511.06085* (2015).

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