

Master Thesis



Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Radioelectronics

Performance Evaluation of Learning-based Image Compression Techniques

Jan Ošťádal

Supervisor: Ing. Karel Fliegel, Ph.D.
January 2023

I. Personal and study details

Student's name: **Oš ádal Jan** Personal ID number: **474444**
Faculty / Institute: **Faculty of Electrical Engineering**
Department / Institute: **Department of Radioelectronics**
Study program: **Electronics and Communications**
Specialisation: **Audiovisual and Signal Processing**

II. Master's thesis details

Master's thesis title in English:

Performance Evaluation of Learning-based Image Compression Techniques

Master's thesis title in Czech:

Hodnocení ú innosti kompresních metod obrazu založených na u ení

Guidelines:

Provide an overview of recent machine learning-based image compression techniques. Focus also on approaches to efficient performance evaluation of these methods. Based on this knowledge, compare the performance of the studied methods to conventional approaches. Perform an analysis of compression artifacts, especially with regard to experiments with a group of observers.

Bibliography / sources:

- [1] Upenik, E., Testolina, M., Ebrahimi, T., Towards super resolution in the compressed domain of learning-based image codecs, Proc. SPIE 11842, 2021.
- [2] Upenik, E., Testolina, M., Ascenso, J., Pereira, F., Ebrahimi, T., Large-scale crowdsourcing subjective quality evaluation of learning-based image coding, IEEE VCIP 2021.

Name and workplace of master's thesis supervisor:

Ing. Karel Fliegel, Ph.D. Department of Radioelectronics FEE

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **26.01.2022** Deadline for master's thesis submission: **10.01.2023**

Assignment valid until: **30.09.2023**

Ing. Karel Fliegel, Ph.D.
Supervisor's signature

doc. Ing. Stanislav Vítek, Ph.D.
Head of department's signature

prof. Mgr. Petr Páta, Ph.D.
Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

Date of assignment receipt

Student's signature

Acknowledgements

I would like to thank my supervisor Ing. Karel Fliegel, Ph.D. for his patience with me in the times this work was created.

Declaration

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Place:

Date:

Signature:

Abstract

This master thesis focuses on image compression and how compression artifacts affect face detection task with respect to differences between traditional algorithms and algorithms based on machine learning. Firstly an overview of a standardization process of JPEG AI initiative is provided. Then the algorithms for compression and for face detection are described, with a brief introduction to image quality assessment. For the experiment 5 traditional compression algorithms and 2 machine learning models were selected for compression task. MTCNN and RetinaFace detectors were identified as suitable algorithms for face detection task. As an input data served images that were picked from WiderFace dataset. The performance of the compression algorithms is evaluated with objective quality metrics. The thesis concludes with the description of which metrics are a reliable indicator of detection quality and which algorithms are suitable for different tasks.

Keywords: Face detection, Image Compression, JPEG AI, Machine learning, Quality assessment

Supervisor: Ing. Karel Fliegel, Ph.D.
Katedra radioelektroniky FEL,
Technická 2,
166 27 Praha 6

Abstrakt

Tato diplomová práce se zabývá obrazovou kompresí a jak kompresní algoritmy ovlivňují detekci obličejů s ohledem na rozdíly mezi tradičními algoritmy a algoritmy založenými na strojovém učení. V první řadě je podán přehled standardizačních procesů iniciativy JPEG AI. Poté jsou popsány algoritmy používané v kompresi a při detekci obličejů se stručným úvodem do hodnocení kvality obrázků. Pro experiment bylo zvoleno 5 tradičních kompresních algoritmů a 2 strojově učené. MTCNN a RetinaFace byly identifikovány jako vhodné algoritmy pro detekci obličejů. Jako vstupní data posloužily obrázky vybrané z databáze WiderFace. Účinnost kompresních algoritmů byla zhodnocena pomocí objektivních metrik kvality. Práce je uzavřena popisem toho, které metriky jsou spolehlivým indikátorem kvality detekce a které algoritmy jsou vhodné pro různé úlohy.

Klíčová slova: Detekce obličejů, Kompresie obrázků, JPEG AI, Strojové učení, Hodnocení kvality

Překlad názvu: Hodnocení účinnosti kompresních metod obrazu založených na učení

Contents

Introduction	1	3.5 BPG	22
1 Image compression	3	3.6 Deep-learning based algorithms .	22
1.1 Traditional approach	3	3.7 RetinaFace	23
1.1.1 JPEG XL	3	3.8 MTCNN	23
1.1.2 Better Portable Graphics	4	3.9 Images	24
1.1.3 WebP	4	4 Analysis of the results	27
1.2 Machine learning based		4.1 Methodology of the experiment .	27
compression	4	4.2 Compression artifacts	29
1.2.1 Context-adaptive Entropy		4.3 Normality test of metrics data ..	34
Model for End-to-end Optimized		4.4 Image quality assessment	36
Image Compression	5	4.5 Detection of faces	38
1.2.2 Full Resolution Image		4.6 Figures of metrics	41
Compression with Recurrent Neural		4.7 Spearman’s rank correlation	46
Networks	5	4.8 Suitability of compression	
1.2.3 Channel-wise Autoregressive		algorithm for different tasks	50
Entropy Models for Learned Image		5 Conclusion	51
Compression	5	5.1 Possible extension	52
1.2.4 Joint Autoregressive and		Bibliography	53
Hierarchical Priors for Learned		A Digital attachment	59
Image Compression	6		
1.3 Image quality assessment	6		
1.3.1 Subjective metrics	6		
1.3.2 Objective metrics	7		
1.3.3 No reference	8		
1.3.4 Full reference	8		
1.4 JPEG AI	10		
1.4.1 Framework and requirements	10		
1.4.2 Proposal conditions	11		
1.4.3 Common Training and Test			
Conditions	12		
1.4.4 Objective testing	12		
1.4.5 Subjective tests	12		
1.4.6 Complexity	13		
1.4.7 Computer vision	13		
1.4.8 Image processing	13		
2 Computer vision	15		
2.1 Multi-task Cascaded Convolutional			
Networks	15		
2.2 RetinaFace	16		
3 Software requirements and			
available implementations	19		
3.1 Windows Subsystem for Linux . .	19		
3.2 JPEG and JPEG 2000	20		
3.3 JXL	21		
3.4 WebP	21		

Figures

1.1 Compression pipeline with quality metrics	7	4.18 Scatter plots for Bitrate and <i>accuracy</i> , RetinaFace detector	48
1.2 Structural similarity index	9	4.19 Scatter plots for PSNR and <i>accuracy</i> , MTCNN detector	48
1.3 JPEG AI Framework[7].....	11	4.20 Scatter plots for PSNR and <i>accuracy</i> , RetinaFace detector	49
1.4 Objective testing pipeline[8]	12		
2.1 MTCNN architecture[23]	16		
2.2 RetinaFace Structure[16].....	17		
3.1 Testing images examples	24		
3.2 Histogram of No-reference metrics of the input	25		
4.1 Image quality metrics for the image kodim14.png in the Kodak database.	28		
4.2 Framework of the experiment ..	29		
4.3 JPEG artifacts	30		
4.4 JPEG XL artifacts, bitrate = 0.168 <i>bpp</i> for the left image, bitrate = 0.196 <i>bpp</i> for the right image.....	30		
4.5 MBT artifacts	31		
4.6 MS artifacts	31		
4.7 J2K artifacts	32		
4.8 BPG artifacts	32		
4.9 WebP artifacts	33		
4.10 Image quality assessment of all images	36		
4.11 Face detection on an uncompressed image with MTCNN detector on the left and RetinaFace on the right	38		
4.12 Face detection for JPG, JXL, MBT and MS compressions in the listed order with MTCNN on the left and RetinaFace on the right	39		
4.13 Face detection for J2K, BPG and WebP compressions with MTCNN on the left and RetinaFace on the right	40		
4.14 Dependence of detection <i>accuracy</i> on objective metrics for MTCNN .	42		
4.15 Dependence of detection <i>accuracy</i> on objective metrics for RetinaFace	43		
4.16 Medians without Error bars with MTCNN on the left, RetinaFace on the right	44		
4.17 Scatter plots for Bitrate and <i>accuracy</i> , MTCNN detector	47		

Tables

4.1 Lilliefors test's p-value for Bit-rate	34
4.2 Lilliefors test's p-value for PSNR	34
4.3 Lilliefors test's p-value for MS-SIM	34
4.4 Lilliefors test's p-value for VIF	35
4.5 Lilliefors test's p-value for FSIMc	35
4.6 Lilliefors test's p-value for <i>accuracy</i> of MTCNN	35
4.7 Lilliefors test's p-value for <i>accuracy</i> of RetinaFace	35
4.8 Spearman's rank correlation between <i>accuracy</i> and metrics for MTCNN.....	46
4.9 Spearman's rank correlation between <i>accuracy</i> and metrics for RetinaFace.....	46




Introduction

The number of multimedia content generated daily increases every year. This content includes images and videos shared on the internet, video conferences, live streams, medical screenings of patients, astronomical data or other scientific data. More effective compression is needed to store increasing number of data generated every day. The effectiveness does not include only possibility of lowering bitrate of the data, but also keeping the quality of the images as high as possible. Today two approaches are recognized in terms of compression, traditional approach and machine learning approach. Nowadays the latter is being more and more spread[47]-[49]. The advantage of machine learning algorithms is that they are able to achieve higher quality at the lower bitrates[37]. However, there is no standard available for machine learning compression in the time of creation of this work, which leads to a variety of technical knowledge on the user side and inconsistency in what is available for the user in terms of additional features apart from compression.

To test the quality of a compression algorithm, one can utilize various subjective and objective image quality tests. Both the subjective and objective approaches have their advantages and disadvantages, with the subjective experiments being demanding to conduct and objective tests not always being able to correlate with perceived quality. L. Zhang (2011)[31] studies the correlation between MOS and selected objective metrics. Similar research was conducted also by JPEG in 2019[5] with more metrics. Both studies show that popular pixel metric peak signal-to-noise ratio (PSNR) metrics are outperformed in terms of correlation with subjective tests by those based on structural similarity and human visual system. However, the PSNR metric is still being used for quality assessment due to its low computational complexity. Study [36] from 2019 shows that MS-SSIM optimized ML algorithms achieve better correlation to subjective tests compared to PSNR optimisation. Hu et. al. (March 2021)[37] gives an overview of 10 ML compression algorithms and provides objective quality assessment of those ML algorithms. As representatives of classic approach JPEG and BPG were selected.

Apart from image compression, the machine learning approach is also being preferred in another field, object detection[1], image segmentation[2], face detection, verification[16] and the list could go on. Face detection can be critical in resolving security incidents, might be a part of securing either



devices or premises and can bring valuable information to the statistics of various events, this area was chosen to be an integral part of this thesis.

Face detection studies focus mainly on objective evaluation and processing time. In [38] authors focus on detectors available in 2020 and examine how the processing time changes when the original image is resized. Another review was done in 2021 on both available literature and popular face detectors[16]. It examines effectiveness of more than 10 different detectors on 3 different datasets via objective metrics. Study [39] examines an ease of face detection in a distorted image asking 26 participants.

Since subjective quality assessment of compression performance is an area thoroughly described for both traditional and machine learning based compression[4] and the topic of compression artifacts and their influence on face detection is not very well explored, the main goal of this thesis shifted to assess artifacts of traditional and learning based compression algorithms. To not completely avoid the influence of compression on human observers, objective quality metrics with high correlation with the subjective tests were utilized.

Chapter 1 covers the differences between traditional and machine learning compression algorithms. It also summarises the standardization attempt of IEC, ISO and ITU organizations to standardize machine learning image coding and offers a brief overview to image quality assessment. Chapter 2 describes various Computer vision task with main focus on Face detection. Conducted experiments and used implementations are covered in chapter 3 and Chapter 4 describes the results of the experiments.

Chapter 1

Image compression

Compression can be divided into lossy and lossless. This chapter describes lossy compression techniques, both traditional and based on machine learning. Unlike other areas of human research, traditional compression is being replaced by machine learning methods slowly, as the traditional approaches offer a great competition in quality to bitrate ratio and there is no standard that would unify objectives of various research laboratories.

1.1 Traditional approach

Traditional compression utilizes an invertible transformation of the image into different space and then operating in such space. Such transformation can be a Discrete cosine transformation that is used in JPEG¹ and WebP² or a wavelet transformation used by JPEG 2000³. The compression is then done by discarding some values in the transformation space. The parameters of the coding are set during the inventing and engineering of a compression codec. Codecs of traditional algorithms typically have various settings that can be adjusted before compression, like compression quality, compression ratio, target bitrate, or colour space.

1.1.1 JPEG XL

JPEG XL described under ISO/IEC 18181⁴, it is an image coding system that targets the specific needs for responsive web, wide colour gamut, and high dynamic range applications. As such the standard is not only describing the codec, but also a set of features that this codec has. Not only it supports natural and synthetic images, it is capable of compressing animations and photo bursts. The codec has 2 coding modes, one for lossy compression, one for mathematically lossless compression. For the lossy compression, the image is transferred into XYZ colour space, which is a colour model inspired by the human visual system, facilitating perceptually uniform quantization. Then

¹<https://jpeg.org/jpeg/index.html>

²<https://developers.google.com/speed/webp>

³<https://jpeg.org/jpeg2000/index.html>

⁴<https://www.iso.org/standard/77977.html>

the image features are extracted on top of the decoded image, for precise and dense representations of patches, curvilinear image features and adaptive, intensity dependent synthetic noise modeling. Colour is then decorrelated using signaled multipliers. Blocks are then filtered to reduce artifacts and keeping the detail. The blocks are then transformed using Variable-sized DCT (square or rectangular from 2×2 to 256×256) and quantized via adaptive quantization. Then prediction is run using a pixel-by-pixel decorrelator without side information, including a parameterized self-correcting weighted ensemble of predictors. After prediction LZ77 entropy coding is utilized, supported by Asymmetric Numeral Systems or Huffman coding[13].

■ 1.1.2 Better Portable Graphics

BPG is a lossy and lossless compression format based on HEVC. It supports grayscale, YCbCr, RGB, YCgCo and CMYK colour spaces with an optional alpha channel. The bit depth of each component is from 8 to 14 bits. The colour values are stored either in full range (JPEG case) or limited range (video case). The YCbCr colour space is either BT 601 (JPEG case), BT 709 or BT 2020. The chroma can be subsampled by a factor of two in horizontal or both in horizontal and vertical directions (4:4:4, 4:2:2 or 4:2:0 chroma formats are supported). In order to be able to transcode JPEG images or video frames without modification to the chroma, both JPEG and MPEG2 chroma sample positions are supported. Arbitrary metadata (such as EXIF, ICC profile, XMP) are supported[46]. This project is no longer in development

■ 1.1.3 WebP

WebP is an image format developed by Google and supported in Chrome, Opera and Android that is optimized to enable faster and smaller images on the Web. The lossy compression is based on VP8 key frame encoding. VP8 is a video compression format created by On2 Technologies as a successor to the VP6 and VP7 formats. The format supports the Alpha channel, which can be used along with lossy RGB. It also preserves EXIF and XMP metadata.[29]

■ 1.2 Machine learning based compression

When one talks about machine learning in image compression, by that is usually meant deep learning approaches, for deep learning is a subprocess of machine learning techniques. Such process takes an input and weights for this random, which are randomly initialized during the first run, process those in a model, calculates a loss function and updates the weights (Forward pass)⁵. The weight are then updated by following an optimization algorithm that is chosen by the developers of such network (Backward pass).

⁵<https://towardsdatascience.com/learning-process-of-a-deep-neural-network-5a9768d7a651>

The typical training procedure takes a set of images created for a specific task, such as image classification, and separates it into 3 subsets: Training, validation and test. The validation set is used to measure the accuracy of the model. This separation is used to prevent overfitting of the model. Typically neural networks are trained for one specific task, like detecting and recognizing certain objects. In image compression, this task is to compress image into certain bitrate or by certain compression ratio, therefore such algorithms require to train a new model in case we want a different bitrate of the output.

■ 1.2.1 Context-adaptive Entropy Model for End-to-end Optimized Image Compression

The proposed model is a context-adaptive entropy model for use in end-to-end optimized image compression. It exploits two types of contexts, bit-consuming contexts and bit-free contexts, distinguished based upon whether additional bit allocation is required. Based on these contexts, the model is allowed to more accurately estimate the distribution of each latent representation with a more generalized form of the approximation models, which accordingly leads to an enhanced compression performance. Based on their experiments, the proposed method outperforms the traditional image codecs, such as BPG and JPEG2000, as well as other previous artificial-neural-network (ANN) based approaches, in terms of the peak signal-to-noise ratio (PSNR) and multi-scale structural similarity (MS-SSIM) index.[9]

■ 1.2.2 Full Resolution Image Compression with Recurrent Neural Networks

This model was proposed by George Toderici, Damien Vincent, Nick Johnston, Sung Jin Hwang, David Minnen, Joel Shor and Michele Covell. In their paper they present a set of full-resolution lossy image compression methods based on neural networks. Each of the architectures described can provide variable compression rates during deployment without requiring retraining of the network: each network need only be trained once. All architectures consist of a recurrent neural network (RNN)-based encoder and decoder, a binarizer, and a neural network for entropy coding. The paper provides comparison of RNN types (LSTM, associative LSTM) and introduce a new hybrid of GRU and ResNet. They also compare their model to previous work, showing improvements of 4.3%-8.8% AUC (area under the rate-distortion curve), depending on the perceptual metric used.[10]

■ 1.2.3 Channel-wise Autoregressive Entropy Models for Learned Image Compression

Authors developed an image compression architecture capable of matching the rate-distortion (RD) performance of a context-adaptive model while minimizing serial processing that can lead to slow decoding times. They

proposed two architectural enhancements: channel-conditioning (CC) and latent residual prediction (LRP). In their study, authors showed how training synthesis transforms with rounded latent values interacts positively with CC and LRP to further boost RD performance. The combined effect of these improvements is a highly parallelizable architecture that according to authors should outperform specific context-adaptive models by 6.7% on Kodak and 11.4% on the Tecnick image set. Their method also show larger gains compared to standard codecs and learning-based models that do not use context in Figures 2 and 3 of their work. The coding improvements provided by CC and LRP are most effective at low bit rates where the model saves more than 16% compared to the context-adaptive baseline and as much as 25% relative to BPG[11]. In this work this model is referred to with the abbreviation *MS*.

■ 1.2.4 Joint Autoregressive and Hierarchical Priors for Learned Image Compression

This model was proposed by David Minnen, Johannes Ballé, George Toderici. The models were build on the work of Ballé et al, which uses a noise-based relaxation and introduces a hierarchical prior to improve the entropy model. While most previous research uses a fixed, though potentially complex, entropy model, Ballé et al. use a Gaussian scale mixture (GSM) where the scale parameters are conditioned on a hyperprior. The authors extend this GSM-based entropy model in two ways: first, by generalizing the hierarchical GSM model to a Gaussian mixture model, and by adding an autoregressive component. They assess the compression performance of both approaches, including variations in the network architectures, and discuss benefits and potential drawbacks of both extensions[12]. For this model the abbreviation *MBT* is used.

■ 1.3 Image quality assessment

The image quality assessment test are divided into 2 categories, subjective and objective. The objective methods can then be divided into subcategories, no-reference, reduced-reference and full-reference metrics.

■ 1.3.1 Subjective metrics

A group of observers is invited to take part in the subjective testing either in a laboratory or it is possible to take part in such an experiment via crowdsourcing in their homes. Different methods can be chosen for the experiment, like single-stimulus or double-stimulus ratings, ordering by force-choice pairwise comparison and pairwise similarity judgements[44]. In the single and double stimulus ratings, the assessed image is shown for a set amount of time and then the observer is asked to rate the quality of the image on a 5 points scale, from bad to excellent. In the single stimulus

method only testing image is shown, in double stimulus the reference is also shown to the observer. The force-choice method requires from the observer to choose a higher quality image from a pair of images of 1 particular scene. In this method, there is no time limit in which the images are shown. The observer has to choose 1 image even though they do not see any differences. The similarity judgement method is similar to the force-choice method with the difference that the observer also indicates a difference in quality on a continuous scale. From the answers is then calculated mean opinion score (MOS), defined as

$$MOS = \sum_{n=1}^N \frac{R_n}{N}, \quad (1.1)$$

where N is the number of observers and R is a rating of an observer n .

1.3.2 Objective metrics

The objective metrics do not require any observers and can be integrated into the compression pipeline, resulting in automated process of compression and evaluation as shown in Figure 1.1. The most commonly used objective metrics can be divided into 2 categories based on availability of the data. Those categories are called No Reference (NR) and Full reference (FR) metrics.

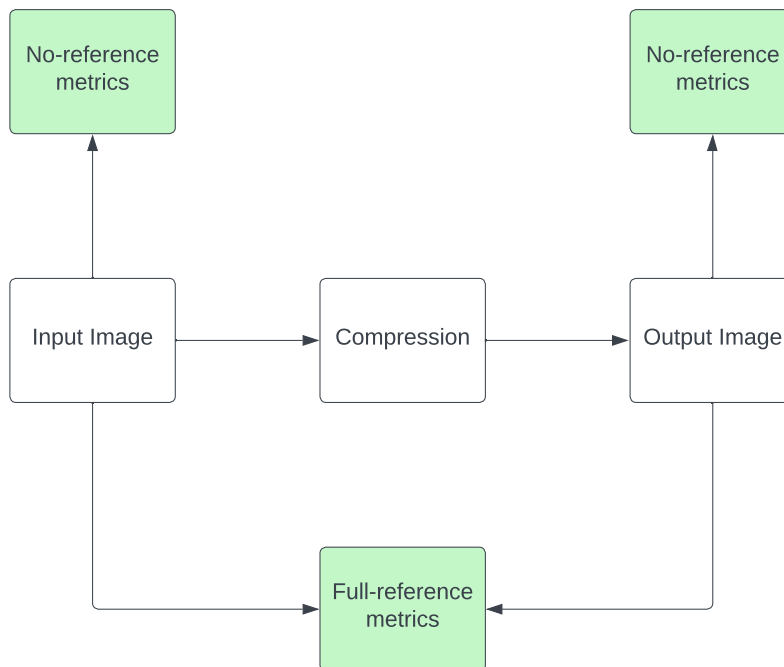


Figure 1.1: Compression pipeline with quality metrics

1.3.3 No reference

NR metrics are used in a case where no input data is available for evaluation. Their aim is to objectively describe an image quality with one number. Several metrics are available for MATLAB, for this work Blind/referenceless image spatial quality evaluator (BRISQUE) and Perception-based Image Quality Evaluator (PIQE) were used. Results in [50] show that the BRISQUE index correlates more with MOS of human observers than some of the FR metrics. The article also explores computational efficiency of this metric, concluding that the efficiency of the BRISQUE metric is higher compared to other NR metrics.

BRISQUE [51] first normalizes the image and extracts Natural Scene Statistics using Mean Subtracted Contrast Normalization (MSCN):

$$I(\hat{i}, j) = \frac{I(j, i) - \mu(i, j)}{\sigma(i, j) + C}, \quad (1.2)$$

where $i \in 1, 2, \dots, M, j \in 1, 2, \dots, N$ M is image height, N is image width, and $\mu(i, j), \sigma(i, j)$ are local mean field and local variance field.

From the normalized image the relationship between a pixel and its neighbors is calculated using pair-wise products of MSCN image with a shifted version of the MSCN image. The shift is made in horizontal, vertical, left-diagonal and right-diagonal directions. As a next step a 36x1 feature vector is calculated by fitting MSCN image to a Generalized Gaussian Distribution and results from the previous step to Asymmetric Generalized Gaussian Distribution. This process is then repeated once on an image with half of the resolution of the original. Last step a regression model is trained, specifically a support vector machine regression model with a radial basis function kernel. For training and testing images from LIVE IQA database were randomly selected. The higher values indicates worse quality.

The first step for PIQE[52] is the same as for BRISQUE. The image is then separated into blocks of size 16-by-16. Using the MSCN the blocks are marked as either uniform or spatially active. The MSCN is also used in the spatially active blocks to identify and measure distortion and using authors' specified criteria the blocks are marked as distorted with blocking artifacts or with Gaussian noise. The score is then calculated as a mean of scores across all distorted blocks. As with BRISQUE, the lower score indicates better quality of the image.

1.3.4 Full reference

Full reference metrics require both the original image and a distorted image, where an output from compression algorithm can be used as the distorted image. Those metrics can be divided into 2 subcategories, pixel based and human visual system (HVS) based. The main advantage of the pixel metrics is their low computational speed, however, their correlation with the MOS is fairly low compared to HVS based metrics[32]. Those metrics with high

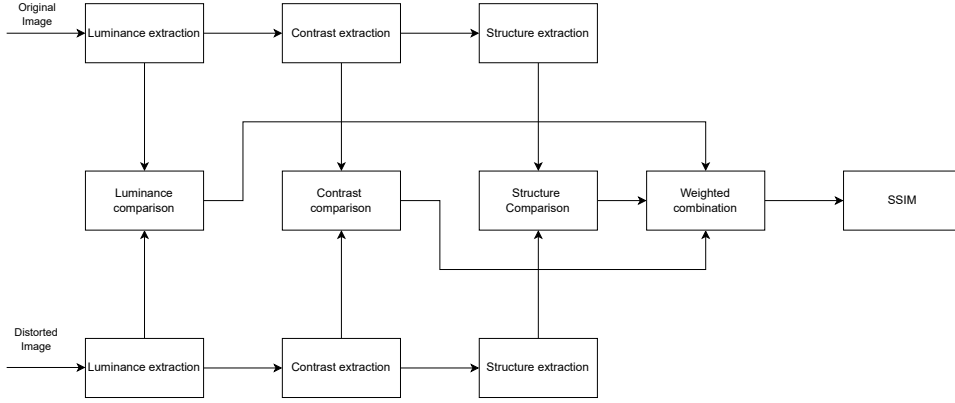


Figure 1.2: Structural similarity index

correlation coefficients are suitable substitutions for more time and cost expensive subjective tests.

The most common pixel based metric is Peak signal to noise ratio defined as:

$$PSNR = 10 \log_{10} \frac{D^2}{MSE}, \quad (1.3)$$

where D is the maximum pixel value and

$$MSE = \sum_{i=1}^N (x_i - y_i)^2 \quad (1.4)$$

is mean square error.

Another popular metric is a structural similarity index (SSIM) [33], or its extension multi-scale structural similarity index (MS-SIM). The SSIM metric extracts image structure, luminance, and contrast, compares those values between reference image and testing images, which can be expressed as:

$$SSIM(x, y) = [l(x, y)]^\alpha \cdot [c(x, y)]^\beta \cdot [s(x, y)]^\gamma \quad (1.5)$$

Diagram of SSIM can be seen Figure 1.2. The MS-SIM extracts the contrast and the structure of the images and then scales down the image by a factor of 2 and repeats the process up until scale M . Luminance is only extracted at the scale M [34]. The equation 1.5 than changes into:

$$MSSIM = [l_M(x, y)]^{\alpha_M} \cdot \prod_{j=1}^M [c_j(x, y)]^{\beta_j} \cdot [s_j(x, y)]^{\gamma_j} \quad (1.6)$$

Another metric correlating well with subjective IQA across various testing databases is Feature similarity index (FSIM/FSIMc)[31]. This metric combines extraction and comparing similarity of Phase Congruency (PC) and Gradient Magnitude (GM). PC is responsible for extracting highly informative features from image and GM to encode local contrast information. It is also shown that this metrics correlate well with MOS, which means they can serve

well as a substitute for human observers, reducing financial and time costs of experiments.

Visual information fidelity (VIF) is another metric that scored high correlation across various databases[31]. VIF quantifies the loss information to the distortion. It utilizes combination of natural scene statistics, HVS, and an image distortion modeling.

Those are not the only objective IQA metrics that exist. The listed metrics are those that are commonly used in modern IQA and apart from PSNR achieved very high correlation with subjective tests.

1.4 JPEG AI

Since there is no standard on machine learning compression, as can be seen in previous section, Section 1.2, scientist around the globe working on the topic are using methods that befits their own needs and therefore there is no unification between machine learning compression algorithms. As a part of a JPEG AI project, standardization organizations IEC, ISO and ITU decided to adress this problem and issued a call for proposals in order to find a technical solution for a new standard of learning-based image coding. Teams from all over the world were encouraged to propose their technological solutions. The proposed algorithms have to meet various conditions set by the JPEG AI in order to be considered as an adequate candidate for the new standard. The reason behind this activity is that learned-based algorithms offer higher quality levels at lower bitrates.[11],[12].

1.4.1 Framework and requirements

To achieve those goals JPEG AI defines a framework of how such image coding system should work. As can be seen in Figure 1.3, the system takes an image as an input and performs transformation into the latent space. Depending on the desired task, the latent representation is then either used for image reconstruction or the image processing and computer vision tasks are carried out directly with the representation without the need of reconstructed image as can be seen in Figure 1.3. Such method often offers faster running time with similar quality compared to performing denoising and compression in cascade[3]. Document [8] also defines intended usage of this standard with key requirements for for individual application, namely:

- Cloud storage
- Visual surveillance
- Autonomous vehicles and devices
- Image collection storage and management
- Live monitoring of visual data

■ Media distribution

The final system as a whole is meant to support at least 8-bit and 10-bit images of a various content as an input with resolution up to 8K. In the same document additional requirements for the compressed bitstream are also defined as core, or must-have, features and desired, extra, features. Some of the core and desired features are presented in the following list:

- Effective compressed domain image processing and computer vision tasks (core)
- Hardware platform independent (core)
- Support for higher bit depth (desired)
- Support for animated image sequences (desired)

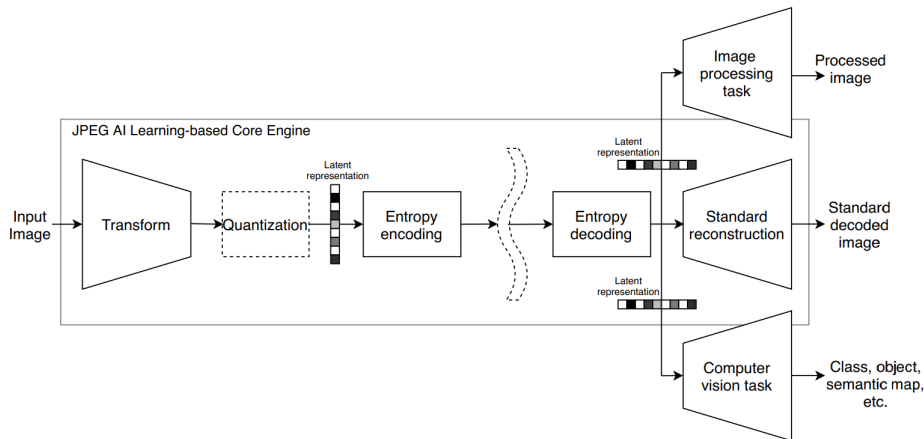


Figure 1.3: JPEG AI Framework[7]

■ 1.4.2 Proposal conditions

Up until 25th February 2022 teams have time to send their proposal to JPEG[7]. The teams are to provide a full technical description of their proposed solution. For the training part JPEG AI dataset has to be used, either full or a subset of it, which is supposed to be clearly stated. The training dataset can also be filled in with additional images, but again this fact must be stated in the description. Other mandatory information are also meant to be provided within the description document, e.g:

- Performance validation (R-D curves)
- Description of how the core and desired requirements are met
- Complexity evaluation

- Encoder and Decoder implementations
- Compressed bitstreams and decoded images

Full description of the registration requisites and specific deadlines are provided by the relevant document[7].

■ 1.4.3 Common Training and Test Conditions

Various tests will be conducted on proposed technologies by at least 2 independent labs[6]. Those tests include evaluation of objective and subjective metrics, complexity of the codecs and performance on computer vision and image processing tasks.

■ 1.4.4 Objective testing

Figure 1.4 shows how the objective evaluation will be done. The target bitrates for the compression are 0.03, 0.06, 0.12, 0.25, 0.50, 0.75, 1.00, 1.50, and 2.00 bpp, with 10% difference acceptance. The used objective metrics are MS-SSIM, IWSSIM, VMAF, VIF, PSNR-HVS-M, NLPD and FSIM. The learned-based codecs will be tested against each other and against traditional codecs, namely JPEG, JPEG 2000, HEVC Intra and VVC Intra.

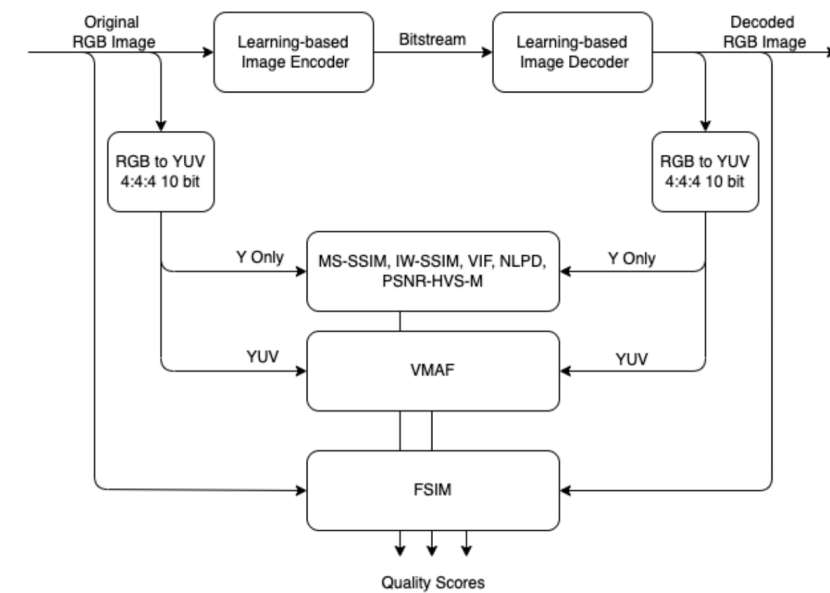


Figure 1.4: Objective testing pipeline[8]

■ 1.4.5 Subjective tests

For subjective testing The Double Stimulus Continuous Quality Scale was chosen. It utilizes a continuous scale divided into 5 levels, excellent, good, fair,

poor, bad, and rating both the original and compressed images without the test subject knowing which of the showed images is the reference. Because of the current pandemic, crowdsourcing approach was selected for subjective testing using QualityCrowd2[14] framework.

■ 1.4.6 Complexity

Additional set of tests aims to measure a complexity of the provided solutions, specifically number of parameters, model precision, CPU and GPU running time for encoding and decoding and minimum GPU memory size for encoding/decoding images in 8K resolution.

■ 1.4.7 Computer vision

One of the objective tasks is an image classification. The image classifiers receive as an input a quantized latent representation (and not a decoded image) and should achieve competitive image classification accuracy with respect to full decoding followed by image classification as well as lower complexity. The quantized latent code of an image is the input to the compressed domain image classification network. Suitable training is also needed to derive the weights of the compressed domain image classification network. The image classification task is based on Resnet-50 image-domain classification network. One of the categories can be an image of a face, and therefore this thesis is also partially inspired by this effort.

■ 1.4.8 Image processing

The image processing tasks are divided into super-resolution and denoising tasks. Compressed-domain SR aims at reducing the computational cost of potentially required up-scaling of a compressed image. Moreover, super resolution may contribute to reducing bandwidth costs, as well as lowering the required storage capacity for local and cloud-based systems. The method is divided into 2 separate branches:

1. The super resolution methods are applied to the images from the test dataset that have been down-sampled from original high-resolution images. All up-sampling is applied before any compression, thus avoiding any compression artifacts.
2. Super resolution is applied in pixel domain to fully decoded images. These images have been obtained by down-sampling the original highresolution images using the methods specified above, which are then encoded and decoded using the learning-based image codec that was submitted for the standard reconstruction track.

Compressed-domain Image Denoising is an image processing task that aims at removing the noise directly from the latent representation of learning-based coding solution while decoding. For this purpose, a compressed-domain

decoder that integrates denoising operations at the decoder side of learning-based image compression methods should be proposed. The pipeline should be able to compress and denoise images simultaneously, being the information of the noise distribution and standard deviation known. Integrating the denoising operations in the decoder has the advantage of reducing the computational complexity and, potentially, improving the performance of the pipeline when compared to the decoding and denoising in cascade. Same as super-resolution, the testing method is also dedived into 2 branches:

- The image denoising methods are applied to the images of the given noisy test dataset. The denoising is applied before any compression, thus avoiding any compression artifact.
- Denoising is applied in the pixel domain to fully decoded images. These images should be created by encoding and decoding the given noisy test dataset using the learning-based image codec that was submitted for the standard reconstruction track.

Chapter 2

Computer vision

Compression is not the only image processing field where the ML techniques take over traditional approaches[16],[45]. With increasing power of computer devices and accessibility via cloud services the ML methods are being deployed in more and more areas of human life, like medicine, security, Earth observation, traffic control and many more[18].

Image classification is the basic computer vision task that is being solved by machine learning. The goal is to assign one label the whole image. Such image can be a picture of a pet or a picture of a diseased crop[17]. In this area, the most popular method from supervised learning are convolutional neural network (CNN), with ResNet50 being the most common. It is a CNN that is 50 layers deep with a pretrained weights trained on more than a million images from the ImageNet database¹.

Another task commonly solved by ML algorithms is image segmentation. In this task labels are assigned to every pixel in the picture, which partitions the image of regions with common characteristic. This way the image is simplified from a multi channel representation into single channel image[20]. This simplified representation is then used for autonomous driving, change detection in satellite imagery or in medicine[19].

Third field in which ML algorithms dominate is object detection. Those algorithms can be separated into two main categories, single-stage and multi-stage algorithms. The single stage is dominated by the YOLO architecture² and in the multi-stage category excels RCNN and it's derivatives[21]. Those algorithms help with identifying trespassers, counting containers or supervising traffic. One of the subfields of object detection is face detection, which plays an integral part in this work. Therefore two state-of-the-art algorithms are described in the following sections.

2.1 Multi-task Cascaded Convolutional Networks

MTCNN[23] is a three-stage face detection and face alignment framework utilizing convolutional networks. In the first stage candidate windows are

¹<https://www.image-net.org/>

²<https://github.com/AlexeyAB/darknet>

obtained, calibrated and overlapping windows are merged together using a non-maximum suppression. This stage is also called Proposal Network (P-net). During second stage false candidates are rejected via another convolutional network labeled as Refine Network (R-net). The last stage is meant to describe facial landmarks, namely position of eyes, nose and mouth. This stage uses a network called Output Network (O-net). The structures of those networks can be seen in Figure 2.1.

P-net was trained with randomly selected cropped patches from WIDER FACE database, which contains over 32000 images in 61 categories with more than 390000 faces, and as landmark faces cropped faces from CelebA[25] were used. CelebA is a face attributes dataset with over 200000 celebrity images with 5 face landmark locations and 40 attribute annotations per image. R-Net was trained utilizing the first stage and detecting faces from WIDER FACE to collect positives, negatives and part face while landmark faces are detected from CelebA. The last stage was trained in a same manner as stage 2, but with using both stage 1 and stage 2 to detect faces.

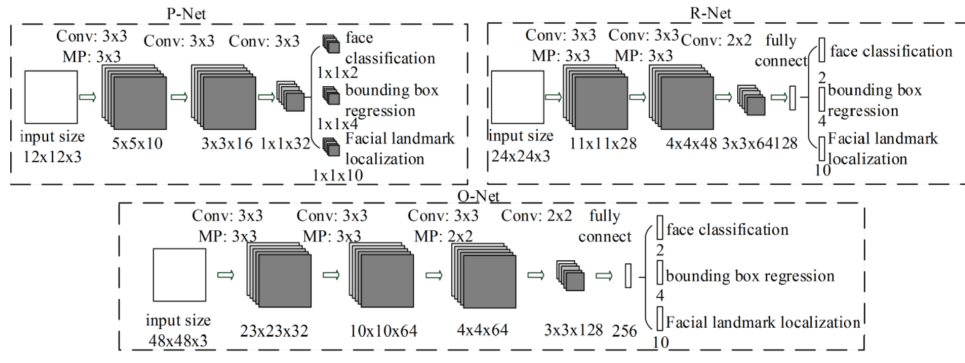


Figure 2.1: MTCNN architecture[23]

2.2 RetinaFace

RetinaFace is a single-stage face detector based on the feature pyramids with ResNet backbone network. The feature pyramid is composed of levels P_2 up to P_6 , where P_2 to P_5 are computed from the output of the corresponding ResNet residual stages as described in [15] and P_6 is calculated through a 3×3 convolution with stride=2 on stage 5 (C_5) of ResNet residual stages[22].

The whole network was trained with WIDER FACE training dataset. For the purpose of detection improvement, facial landmarks were annotated in training and validation subsets. Based on this study[16], RetinaFace outperforms other state-of-the-art detection methods in terms of accuracy on WIDER FACE database and The Receiver operating characteristic (ROC) curves on Fddb benchmark.

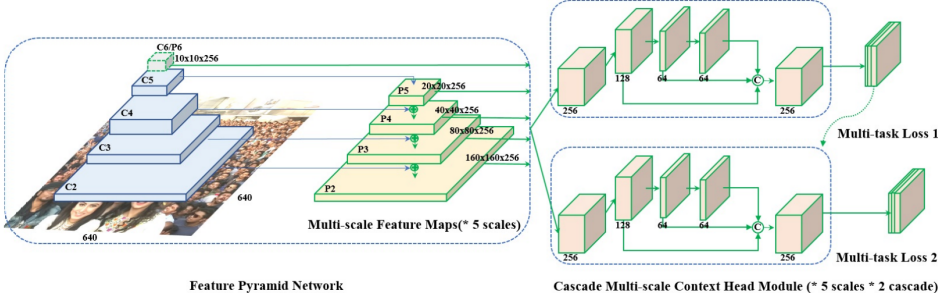


Figure 2.2: RetinaFace Structure[16]

Chapter 3

Software requirements and available implementations

Inspired by the Call for Proposals the conducted experiments focus on how the image compression affects computer vision tasks, namely face detection. As the machine learning compression algorithms don't utilize discrete transformations, we can expect them to express a different type of artifacts than traditional methods. In order to analyze the effect of those artifacts on face detection task, a series of tests was devised. The implementations for testing were chosen based on their free accessibility and simple usability. The database of images should contain several numbers of faces per image and various conditions so the results can be used to generalize the effects of compression artifacts.

Five traditional compression codecs, JPEG JPEG XL, JPEG 2000, WebP and BPG, and two machine learning codecs, discussed in 1.2.4 and 1.2.3, were chosen for the compression part. Algorithms discussed in Sections 2.1 and 2.2 were selected for face detection. The compression and detection scripts were run in MATLAB R2020a supported by Python version 3.8.12 on Windows 10 platform. Windows subsystem for Linux[26] (WSL) was utilized for implementations that required Linux distribution, namely JXL and compression models discussed in Chapter 2. Python scripts were called via MATLAB *cmd* function, but as of MATLAB R2021b

3.1 Windows Subsystem for Linux

WSL is available for Windows 11 or Build 19041 and higher of Windows 10. In order to install WSL, start Windows command line or PowerShell and run `wsl -install` command, which will install all components, Linux kernel and Ubuntu distribution. After installation is done, WSL will require to create a new username and password. If another distribution is desired, the installing command then changes to `wsl -install -d <Distribution Name>`. Available distributions are the following:

- Ubuntu - command name: `ubuntu`
- Debian - command name: `debian`

- Kali Linux Rolling - command name: `kali-linux`
- OpenSuse Leap 42 - - command name `openSUSE-42`
- SUSE Linux Enterprise Server v12 - command name: `SLES-12`
- Ubuntu 16.04 LTS - command name: `ubuntu-16.04`
- Ubuntu 18.04 LTS - command name: `ubuntu-18.04`
- Ubuntu 20.04 LTS - command name: `ubuntu-20.04`

In order to start using Linux subsystem, one can either run Ubuntu on Windows application or run the command `wsl` in Windows command line. The application opens Linux Terminal in Linux subsystem root folder. On the other hand running `wsl` command opens the Terminal inside Windows command line in the current working directory. It is possible to open Windows File Explorer using `explorer.exe` and manage files and folders without the Terminal.

3.2 JPEG and JPEG 2000

For JPEG and JPEG 2000 compression MATLAB built-in `imwrite()` function can be used. This function is a part of a basic MATLAB licence and does not require additional toolboxes. Apart from input image matrix and file name, user can specify additional parameters depending on a desired file format specified in file name, e.g. `.jpg`, `.png`, `.j2k`. Whole function documentation is available in the MathWorks' documentation[27].

The following commands in MATLAB can be used in order to compress image to JPG format.

- `input=imread(input_path);`
- `output_name=['output.jpg'];`
- `imwrite(input,output_name,'Quality',k);`

For JPEG 2000 the output suffix changes to `.j2k` and instead of quality we specify compression ratio.

- `input=imread(input_path);`
- `output_name=['output.j2k'];`
- `imwrite(input,output_name,'CompressionRatio',1);`

3.3 JXL

JPEG XL has a reference implementation available on GitHub[28]. The library is written in C++. For this experiment it was installed on WSL Ubuntu. Installing on Windows platform is possible, however, this approach was not tested by the developers and therefore the built might be unstable or not working at all.

Installation requires C++ compiler, the developers recommend using Clang v.7 or newer:

- `sudo apt install clang`
- `export CC=clang CXX=clang++`

After installing Clang and setting *CC* and *CXX* the installation is done by the following commands:

- `git clone https://github.com/libjxl/libjxl.git -recursive -shallow-submodules`
- `sudo apt install cmake pkg-config libbrotli-dev libgflags-dev`
- *Optional dependencies:* `sudo apt install libgif-dev libjpeg-dev libopenexr-dev libpng-dev libwebp-dev`
- `cd libjxl`
- `mkdir build`
- `cd build`
- `cmake -DCMAKE_BUILD_TYPE=Release -DBUILD_TESTING=OFF ..`
- `cmake -build . - -j$(nproc)`

After following previous steps, the encoder and decoder will be available in `build/tools` directory. The compression is done using `cjxl` and `djxl` files:

- `cjxl input.png output.jxl`
- `djxl input.jxl output.png`

The encoder supports various settings for encoding. Run `cjxl -h` or `cjxl -h -v` to see more.

3.4 WebP

The WebP codec is available for download from Google developers website [29]. The archives contain precompiled coder and decoder for Windows, macOS and Linux. If the binaries fail to run on macOS and Linux, compiling instructions on how to build one's own versions are also available. The following commands were used for WebP compression:

1. Compression: `cwebp -q 80 image.png -o image.webp`
2. Decompression: `dwebp image.webp -o image.png`

Both `cwebp` and `dwebp` offer additional options, like defining target bitrate instead of quality for `cwebp` or specify colour space for `dwebp`. All the available options can be listed with `cwebp -h` and `dwebp -h`.

3.5 BPG

The website[46] offers an archive with the source code of the `bpgenc`, `bpgdec` and `bpgview` command line utilities (for Linux) and the associated `libbpg` library. It also includes the source code of the Javascript decoder. Binary distribution for Windows is only available for 64 bit systems. For Mac users, the BPG utilities are available in the `libbpg` Homebrew formula.

Compressing with BPG is similar to WebP, however the quality level, or quantizer parameter as described in the documentation, scales from 0 to 51 with 0 giving the highest quality and 51 giving the lowest quality. The commands for compression and decompression are:

1. Compression: `bpgenc -q 0 -o output.bpg input.png`
2. Decompression: `bpgdec -o output.png input.bpg`

Additional options can be obtained by running `bpgenc -h` and `bpgdec -h`.

3.6 Deep-learning based algorithms

As they are the state of the art techniques and showed stable results across various images during previous testing, only algorithms 1.2.3 and 1.2.4 were selected for further evaluation. Both MBT and MS models are available as a part of TensorFlow Compression (TFC) library[30] for Python. This library is only compatible with Linux and Mac OS, for usage on Windows utilizing Docker is recommended by the developers. For this experiment the library version 2.7 was installed on the WSL. This version requires Tensorflow v2.7 module and compatible Python version, 3.7 or newer, installed on the machine. For the purpose of this work Python 3.8 was used, which was the default Python version installed on the WSL. In order to install TFC, run the following commands in WSL Terminal:

- `pip install tensorflow==2.7.0`
- `pip install tensorflow-compression==2.7.0`

If any additional module is missing, install the module using:

- `pip install module-name`

The MS and MBT models can be accessed via `tfci.py` script. The command `python3 tfci.py models` lists all available models and provides basic description of them. The MS and MBT models offer optimisation for either MS-SSIM or MSE.

The individual images can be then compressed and decompressed via following commands:

- `python3 tfci.py compress mbt2018-mean-msssim-[1-8] input.png output.tfci`

- `python3 tfci.py decompress output.tfci output.png`

for MBT model and:

- `python3 tfci.py compress ms2020-cc8-msssim-[1-9] input.png output.tfci`

- `python3 tfci.py decompress output.tfci output.png`

for MS model.

■ 3.7 RetinaFace

Python RetinaFace library is available on GitHub[41]. This implementation is made by Sefik Ilkin Serengil. The library contains various Python functions that are able to detect faces, align them and verify. Detection function returns facial area coordinates, facial landmarks and confidence score of the detection. On the site there are usage examples of those functions. However, the library does not include full scripts for those tasks. In order to install RetinaFace library for Python, run the following command:

- `pip install retina-face`

■ 3.8 MTCNN

For detection using MTCNN TensorFlow implementation from Iván de Paz Centeno was used. The source is also available at GitHub[42] and can be installed using PIP. The library is compatible with Python 3.4 or newer and TensorFlow 2.0+ is required in order to use it. In the library there is an example script `example.py` available on how to detect faces using the library. Installation of this library can be done using

- `pip install mtcnn`

3.9 Images

As a source of images for compression, 180 selected images from Wider Face database were used. The advantage of this database is a large number of images where there is more than one face in the image, which is optimal for our test scenario. The disadvantage is that the authors of the database keep in secret the real number of faces and their locations in the images. At least 3 images were supposed to be chosen from every category of the database. However, if there were images with only 1 face per image in a category, images from different categories were selected. The total number of images selected is 180. Those images vary in resolution, colour, and number of people/faces contained in it. The images also show a variable quality. Figure 3.1 shows 9 samples of the chosen images and Figure 3.2 shows the no-reference metrics histogram, namely PIQE and BRISQUE, which shows the variable quality of the input.



Figure 3.1: Testing images examples

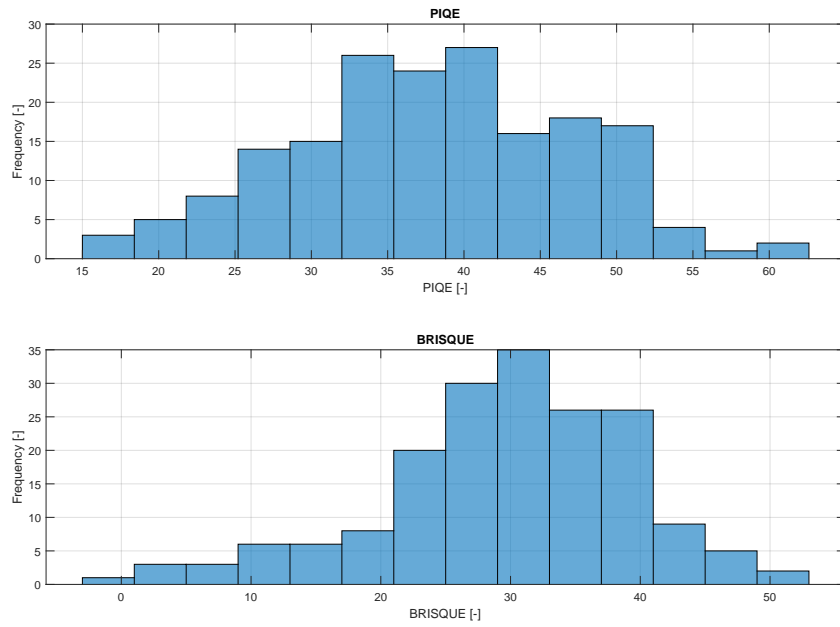


Figure 3.2: Histogram of No-reference metrics of the input

Chapter 4

Analysis of the results

In the following sections the compression algorithms are evaluated based on several criteria. First of the compression artifacts are visually compared to each other. Some examples are provided. Normality tests were conducted on the results in order to get an information about the data distribution and to choose the best representation form for the figures. Then the algorithms are put through objective quality assessment that evaluates their ability of creating visually appealing images with limited bitrate. Lastly the correlation between detection *accuracy* and objective metrics is examined both visually and via Spearman's rank correlation coefficient.

4.1 Methodology of the experiment

The implementations allow usage of MATLAB *system* function, which executes a command in the command line and returns an output of the command. For consistency, the input images were firstly converted into PNG image format using Image Converter application available for Windows. Then a number of faces within each of the original images are detected using *Retina.py* and *MTCNN.py* Python scripts written for this work. Those scripts take an image as the input and return number of faces detected on the image using either RetinaFace or MTCNN detector.

After that the image is compressed using the implementations discussed in sections 3.2 to 3.6. Using trial and error method, the quality levels for the traditional codecs were selected to correspond with lower 5 compression levels of MBT and MS models. Other options of the MATLAB fuction *imwrite* remained default. Based on observations in a previous project, it was concluded that for the higher quality the metrics differences between compression algorithms were insignificant[43].

4. Analysis of the results

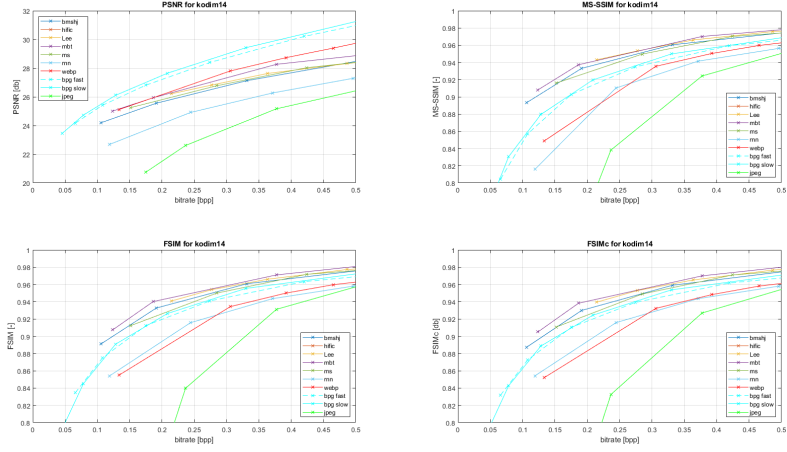


Figure 4.1: Image quality metrics for the image kodim14.png in the Kodak database.

For JPEG XL compression the same quality level as for JPEG is used, since according to the description of the JPEG XL encoder *cjxl*, the positive quality values approximately correspond with those of JPEG algorithm. Apart from quality level, no other options were changes.

TFC allows user to choose from several models. For this work models optimised for MS-SSIM were chosen, since this metric correlates better with subjective mean opinion score (MOS) than PSNR[31].

After compression is done, the face detectors are used to detect faces on the compressed images. This number is then used for computing the *accuracy* (*Acc*) of the detection:

$$Acc = \frac{N_{comp}}{N_{orig}}, \quad (4.1)$$

where N_{comp} is number of faces in a compressed image and N_{orig} is number of faces in the original image.

Objective quality metrics PSNR, MS-SIM, FSIM, FSIMc and VIF, described in Section 1.3.4 are calculated between corresponding original and output of every compression algorithm and compression level. PSNR was chosen for legacy reasons and fast computational time, even though the metric has low correlation with subjective tests. The MS-SIM is an evolutionary step of SSIM metric. FSIM and VIF metrics have a very high correlation with MOS, therefore there are a suitable substitution for subjective testing, which would be slow, complicated and inflexible due to lack of relevant literature on the topic of subjective face recognition testing. The PSNR and MS-SIM metrics are available in Image Processing Toolbox for MATLAB and FSIM and FSIMc are available for MATLAB from the FSIM authors[32]. Figure 4.2 shows individual parts of the framework with green coloured boxes referring to file *Framework.m* and blue coloured boxes referring to file *Results.m*. The latter file also contains various plots that are not shown on the diagram.

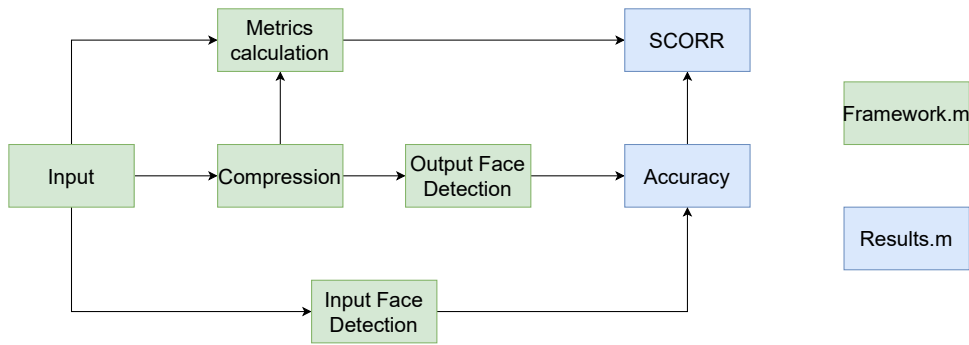


Figure 4.2: Framework of the experiment

4.2 Compression artifacts

We can see the artifacts of JPEG format in Figure 4.3 with details of figures. There is a typical block structure visible in both grey-scale and colour images. In the colour image we can also see a loss of colour information especially in the background and on the left side. The details and face features are unreadable. JPEG XL images in Figure 4.4 still contain a visible block structure, however, it is not as perceivable as in JPEG compression. The colour is not disturbed and the background is also clearer. The details like grass, bushes and face features are more visible than in JPEG images.

Figures 4.5 and 4.6 show the compression artifacts of the machine learning algorithms. The block structure typical for JPEG is not present. The background shows almost no disruption, the details are also very well preserved, but the edges and higher details seem to be smoothed. The MS algorithm has a slightly darker colour scheme and keeps more details, which is visible especially in the background of the coloured image and on the grass in the grey-scaled image. The faces are more distinguishable, however, the facial marks are reconstructed very poorly or not at all and instead of them the face is covered by a flat texture of the face's skin colour. With higher compression level, the smoothness is less noticeable and the face reconstruction is more accurate. In the bottom of the images there is a visible artefact line spreading across the whole width of the image in the MBT model.

4. Analysis of the results



Figure 4.3: JPEG artifacts



Figure 4.4: JPEG XL artifacts, bitrate = 0.168bpp for the left image, bitrate = 0.196bpp for the right image



Figure 4.5: MBT artifacts



Figure 4.6: MS artifacts

4. Analysis of the results

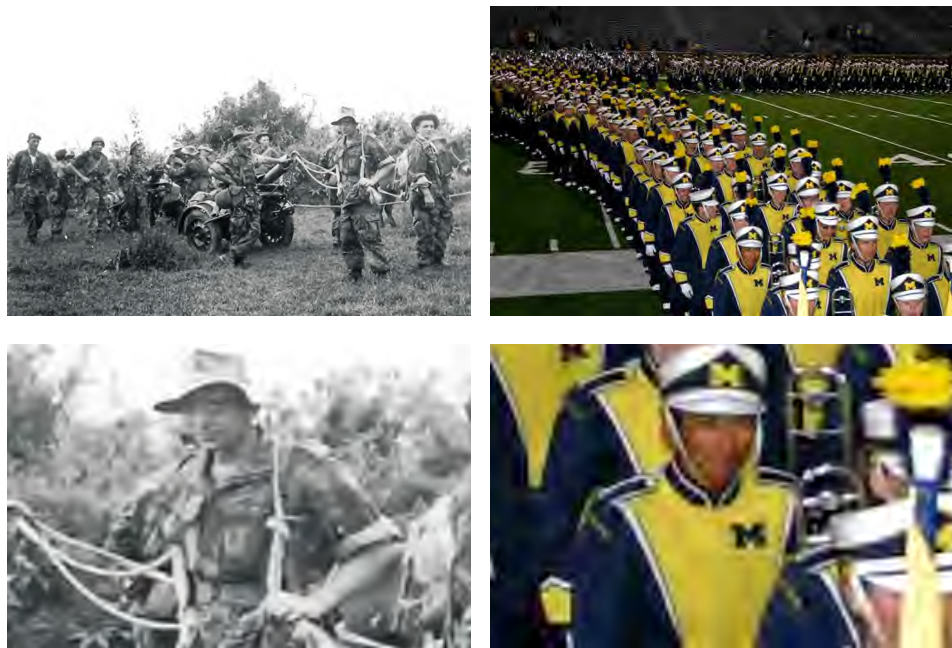


Figure 4.7: J2K artifacts

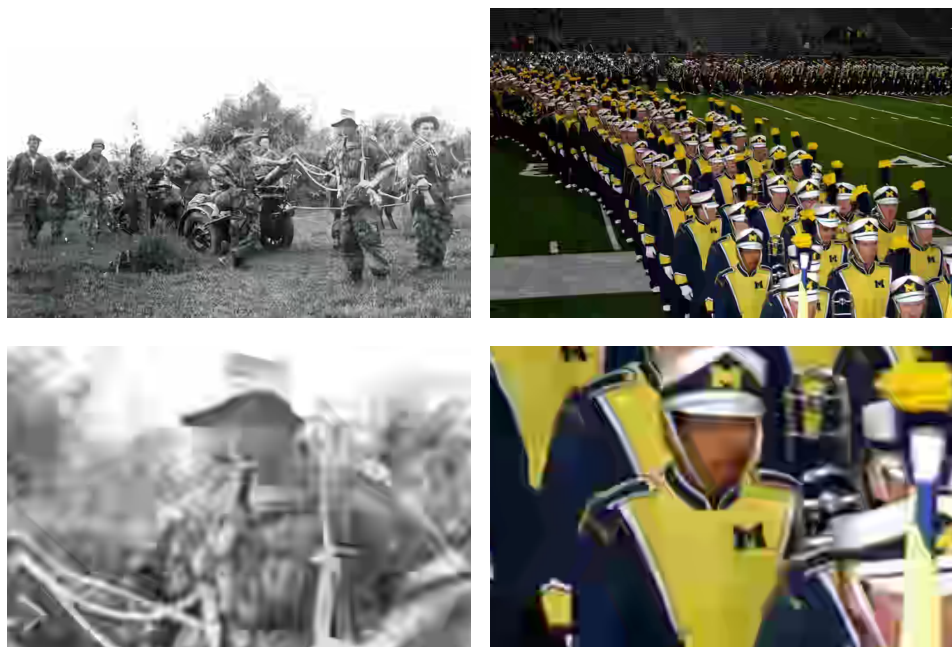


Figure 4.8: BPG artifacts



Figure 4.9: WebP artifacts

JPEG 2000 artifacts can be seen in Figure 4.7. There is no detailed block structure visible, however, there are visible transitions between parts of the picture itself. We can also notice blurring on the edges, as can be seen on arms of the soldier or shoulders of the cap and shoulders of the parade attendee. colour-wise there seems to be a little change compared to the original. Figure 4.8 shows a visible transition between the macro block and edge blurring, leading to general detail suppression. Similarly to the ML algorithms the face details are in some cases smoothed out. The WebP artifacts in Figure 4.9 are similar to the artifacts of JPEG XL, although the pixel structure contains smaller blocks. There is also a visible colour degradation, in general for the grayscale picture and especially on the edges with transitions between several colours for the other picture.

4.3 Normality test of metrics data

In order to decide if we can use mean and standard deviation for plotting results, Lilliefors normality test¹ was utilized, which tests the hypothesis that the data comes from a normal distribution family, but does not specify which, e.g Gaussian, Log-normal, Student's t distribution. The alternative hypothesis is that the data does not come from such a distribution. For that the MATLAB function *lillietest* was used. This function returns 1 for rejecting the null hypothesis and 0 for accepting the null hypothesis at the 0.05 significance by default. It can also return p-value and the test statistic.

The tables 4.1 to 4.7 show the p-value of the statistic for every compression level, algorithm and metric. If the p-value is less than 0.05, the null hypothesis is rejected.

Apart from bitrate and PSNR, we can see that almost always the null hypothesis is rejected, meaning the data does not come from normal distribution.

Compression level	JPG	JXL	MBT	MS	J2K	BPG	WebP
1	0.001	0.16	0.001	0.32	0.001	0.001	0.001
2	0.004	0.21	0.50	0.05	0.001	0.001	0.001
3	0.02	0.05	0.25	0.20	0.006	0.001	0.001
4	0.02	0.09	0.50	0.50	0.001	0.007	0.003
5	0.05	0.1	0.50	0.50	0.001	0.01	0.001

Table 4.1: Lilliefors test's p-value for Bit-rate

Compression level	JPG	JXL	MBT	MS	J2K	BPG	WebP
1	0.42	0.25	0.26	0.14	0.05	0.22	0.41
2	0.18	0.33	0.06	0.13	0.05	0.21	0.38
3	0.11	0.45	0.05	0.04	0.08	0.47	0.50
4	0.14	0.25	0.32	0.02	0.08	0.17	0.32
5	0.09	0.15	0.13	0.19	0.12	0.03	0.25

Table 4.2: Lilliefors test's p-value for PSNR

Compression level	JPG	JXL	MBT	MS	J2K	BPG	WebP
1	0.07	0.005	0.02	0.08	0.15	0.001	0.001
2	0.03	0.001	0.11	0.24	0.07	0.001	0.001
3	0.001	0.001	0.001	0.03	0.02	0.001	0.001
4	0.004	0.001	0.001	0.001	0.003	0.001	0.001
5	0.002	0.001	0.001	0.001	0.002	0.001	0.001

Table 4.3: Lilliefors test's p-value for MS-SIM

¹<https://uk.mathworks.com/help/stats/lillietest.html>

Compression level	JPG	JXL	MBT	MS	J2K	BPG	WebP
1	0.03	0.1	0.08	0.02	0.004	0.50	0.27
2	0.27	0.20	0.24	0.11	0.001	0.50	0.19
3	0.07	0.04	0.02	0.50	0.004	0.50	0.35
4	0.13	0.02	0.50	0.50	0.06	0.50	0.23
5	0.24	0.03	0.50	0.50	0.09	0.04	0.08

Table 4.4: Lilliefors test's p-value for VIF

Compression level	JPG	JXL	MBT	MS	J2K	BPG	WebP
1	0.001	0.001	0.001	0.001	0.001	0.001	0.001
2	0.001	0.001	0.001	0.001	0.003	0.001	0.001
3	0.001	0.001	0.001	0.001	0.001	0.001	0.001
4	0.001	0.001	0.001	0.001	0.001	0.001	0.001
5	0.001	0.001	0.001	0.001	0.001	0.001	0.001

Table 4.5: Lilliefors test's p-value for FSIMc

Compression level	JPG	JXL	MBT	MS	J2K	BPG	WebP
1	0.001	0.001	0.03	0.03	0.003	0.001	0.01
2	0.001	0.02	0.02	0.001	0.001	0.006	0.003
3	0.05	0.11	0.001	0.001	0.001	0.001	0.001
4	0.001	0.001	0.001	0.001	0.001	0.001	0.001
5	0.001	0.001	0.001	0.001	0.001	0.001	0.001

Table 4.6: Lilliefors test's p-value for *accuracy* of MTCNN

Compression level	JPG	JXL	MBT	MS	J2K	BPG	WebP
1	0.001	0.002	0.001	0.001	0.001	0.02	0.001
2	0.001	0.001	0.001	0.001	0.001	0.001	0.001
3	0.001	0.001	0.001	0.001	0.001	0.001	0.001
4	0.001	0.001	0.001	0.001	0.001	0.001	0.001
5	0.001	0.001	0.001	0.001	0.001	0.001	0.001

Table 4.7: Lilliefors test's p-value for *accuracy* of RetinaFace

4.4 Image quality assessment

In this section objective image quality is assessed. As was mentioned before, since the subjective image quality assessment is thoroughly described only for image compression and not for following tasks, objective quality metrics that highly correlate with MOS were chosen as a substitution for subjective testing. Figure 4.10 shows dependence of a quality metrics on bitrate with error bars in both axes. Where error bar in X axis represents the interquartile range (IQR) of bitrate for a specific quality. The higher the Y value is, the higher the perceptual quality would the images have for human subjects. And the lower the X value is, the less space the image requires for storage on a disk.

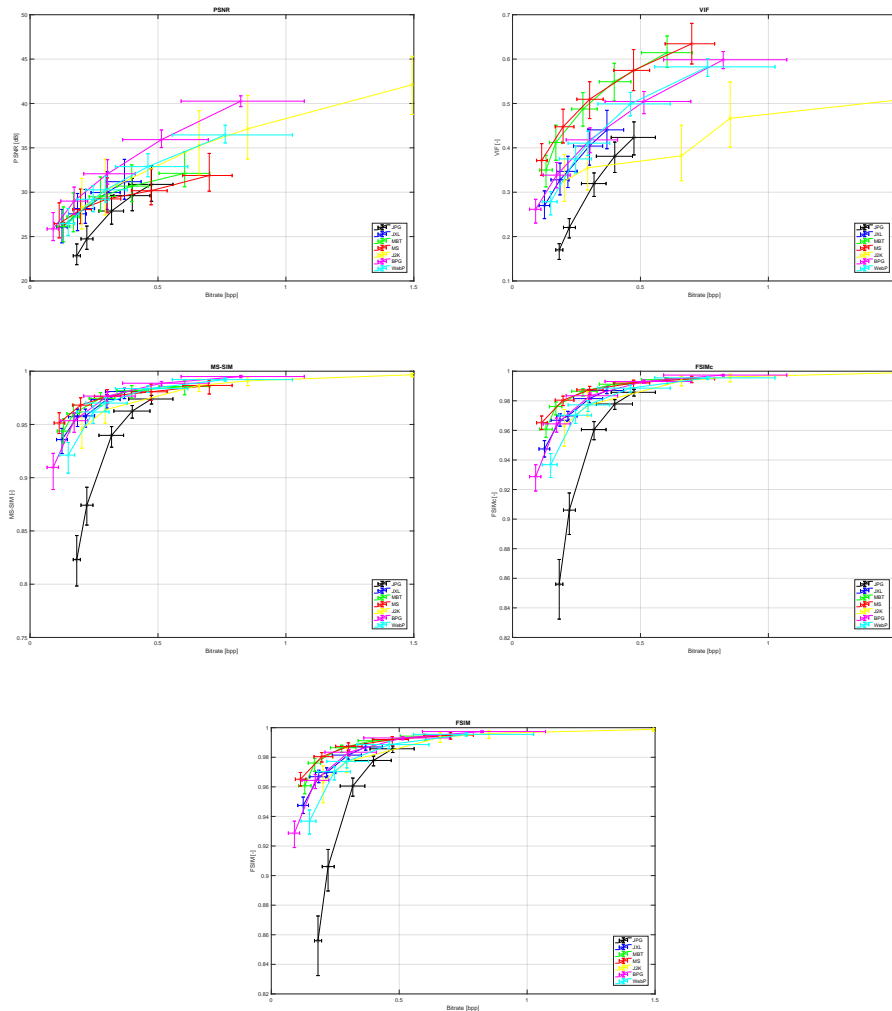


Figure 4.10: Image quality assessment of all images

We can notice that with increasing quality level, the IQR for bitrate increases. On the other hand the range decreases for metrics as the compression quality level gets higher. The only exception is JPEG 2000, where IQR in bitrate is unnoticeable and does not seem to be changing on Y axis for any metric. With higher bitrate the differences in FSIM and MS-SIM between individual algorithms are minimal, which cannot be said for PSNR and VIF. PSNR plot shows that apart from JPEG every other algorithm was able to achieve more or less similar values for the lowest quality level. With increasing quality the differences are more distinguishable, with BPG algorithm achieving the highest values, followed by JPEG 2000 and WebP. Interestingly enough, ML algorithms achieved the lowest PSNR for the highest quality level than any other algorithm, including JPEG.

In the VIF plot we can notice an interesting trend. There is a noticeable step in value between the following groups: MS+MBT, JPEG XL+BPG+WebP, JPEG 2000, JPEG. This indicates that the loss of information caused by compression follows similar rules for ML algorithms, algorithms with discrete sine and cosine transforms and wavelet based algorithms.

With PSNR excluded, we can see that the machine learning algorithms achieved higher metrics for lower bitrate compared to traditional approaches. Followed by BPG and JPEG XL and then by JPEG 2000 and WebP. JPEG algorithm was not able to achieve comparable values for the lowest three quality levels in any of the metrics. The trends in all metrics suggest that JPEG algorithm would be able to achieve similar quality for bitrates over 0.5bpp.

4.5 Detection of faces

Figure 4.11 shows the detected faces in one of the input images on the lowest compression level. We can see that MTCNN detector (left) detected other objects, in this case hats, as a face. It can be safely assumed that this image is not a unique case. Unlike MTCNN the Retina Face detector was able to detect a face in the foreground that is partially covered by a pole. In this case the RetinaFace detector (right) seemingly did not detect any other objects than faces.

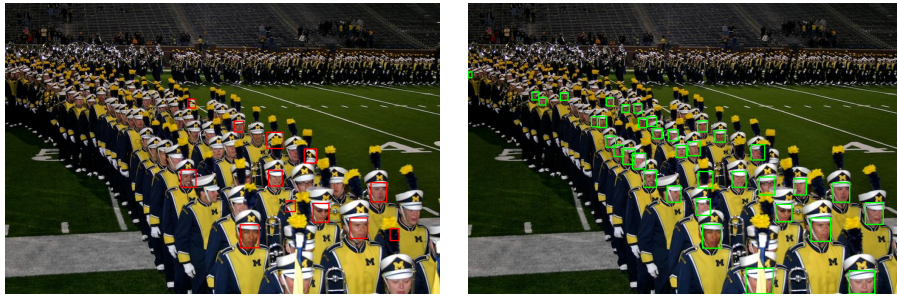


Figure 4.11: Face detection on an uncompressed image with MTCNN detector on the left and RetinaFace on the right

Detected faces in compressed images for JPEG, JPEG XL, MS and MBT can be found in Figure 4.12. The Figure represents only the lowest compression level. Each row represents different compression algorithm. RetinaFace was able to detect more faces, excluding JPEG compression, where MTCNN was able to detect something, but the detection looks like a false positive. In JPEG XL compression, we can see one face that was detected here, but not in the MBT image. We can notice that the most detected faces are in the image compressed via MS algorithm. Comparing MS to MBT compression, RetinaFace was able to detect faces that are further in the background and also the face that was partially covered by a pole in the foreground.

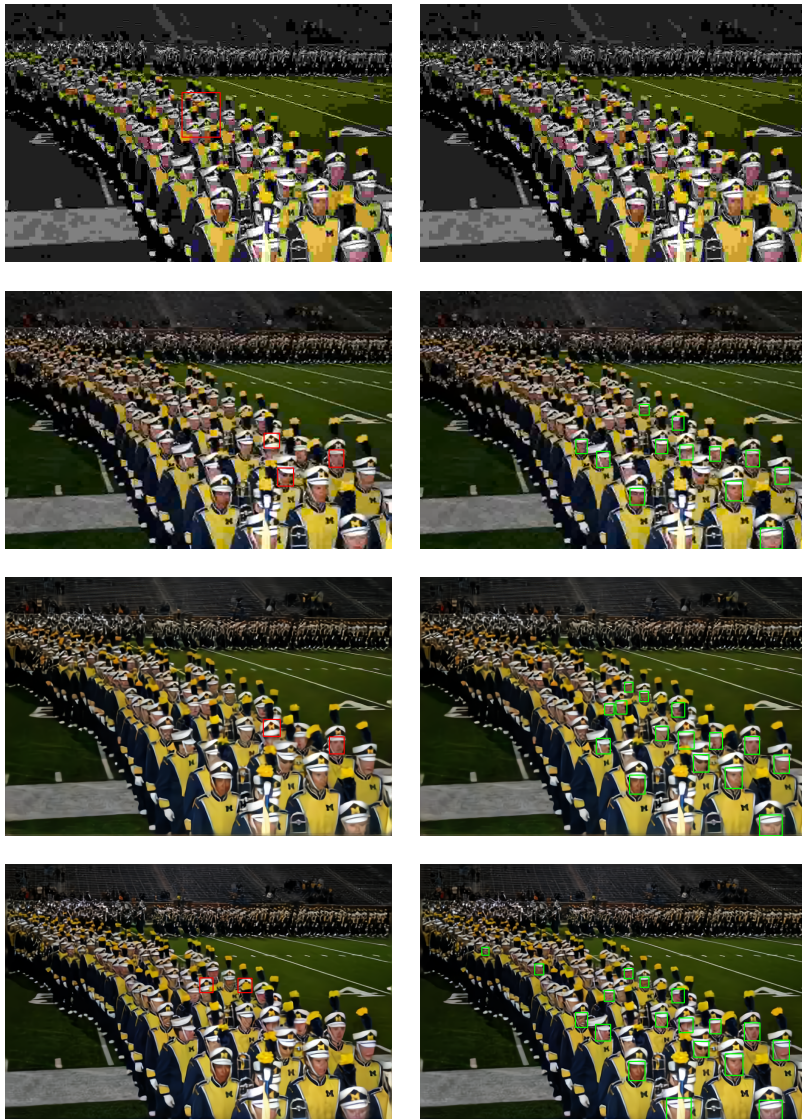


Figure 4.12: Face detection for JPG, JXL, MBT and MS compressions in the listed order with MTCNN on the left and RetinaFace on the right

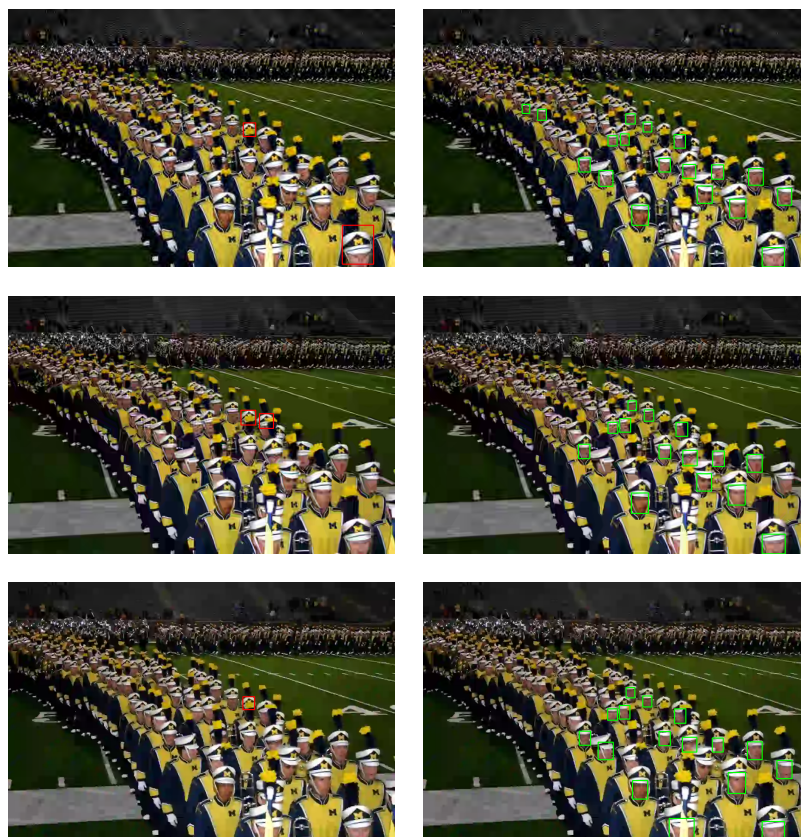


Figure 4.13: Face detection for J2K, BPG and WebP compressions with MTCNN on the left and RetinaFace on the right

From Figure 4.13 we can see that the artifacts of JPEG 2000 have less impact on face detection for RetinaFace. In this case the number of detection is the same as for MS, although the detector was able to detect different faces, and some faces detected in MS are not detected here, notably the one covered behind a pole. This face was again detected in the WebP image, however overall number of detection decreased compared to MS. The MTCNN detector was able to detect only 1 real face and only for JPEG 2000, however the bounding box suggests that the main object of interest for the detector is the cap.

4.6 Figures of metrics

Figures 4.14 and 4.15 show median *accuracy* dependence on median bit-rate and objective metrics for MTCNN detector and RetinaFace detector respectively. Since the normality tests show that the results are mostly not normally distributed, the x coordinates of the center points correspond with medians of the specific quality metric or bitrate for 5 compression levels and the y coordinates are medians of *accuracy* for those compression levels. The error bars represent first and third quartiles for both X and Y values. All plots show an ascending trend in median values, apart from JPEG XL curves in plots for MTCNN, where the third compression level *accuracy* median values are lower than the previous ones.

In the bit-rate - *accuracy* dependence plot for MTCNN detector, there is more than 40% difference in *accuracy* medians between JPEG and other compression algorithms for the lowest compression level. This difference is lower with raising bitrate. However, the *accuracy* IQR is also significantly wide, covering more than 50% *accuracy*. This coverage is getting lower with higher bitrate/compression level. On the other hand, the higher the bitrate is, the higher the intervals between first and third percentiles for bit-rate are.

The PSNR plot also shows difference between JPEG and other compression algorithms, this time however we can see that the other algorithms were able to achieve higher PSNR on the lowest compression level. As with the bitrate, with increasing PSNR/compression level, the IQR of PSNR also increases.

The MS-SIM,FSIM and FSIMc plots behave similarly to each other. The JPEG achieved lower metric values on the 2 lowest compression levels than the other algorithms and thus achieving lower *accuracy* on those levels. With higher compression levels the differences between individual algorithms are less and less distinct. Also the IQR of those metrics appears to be lower and more stable then in previous cases.

4. Analysis of the results

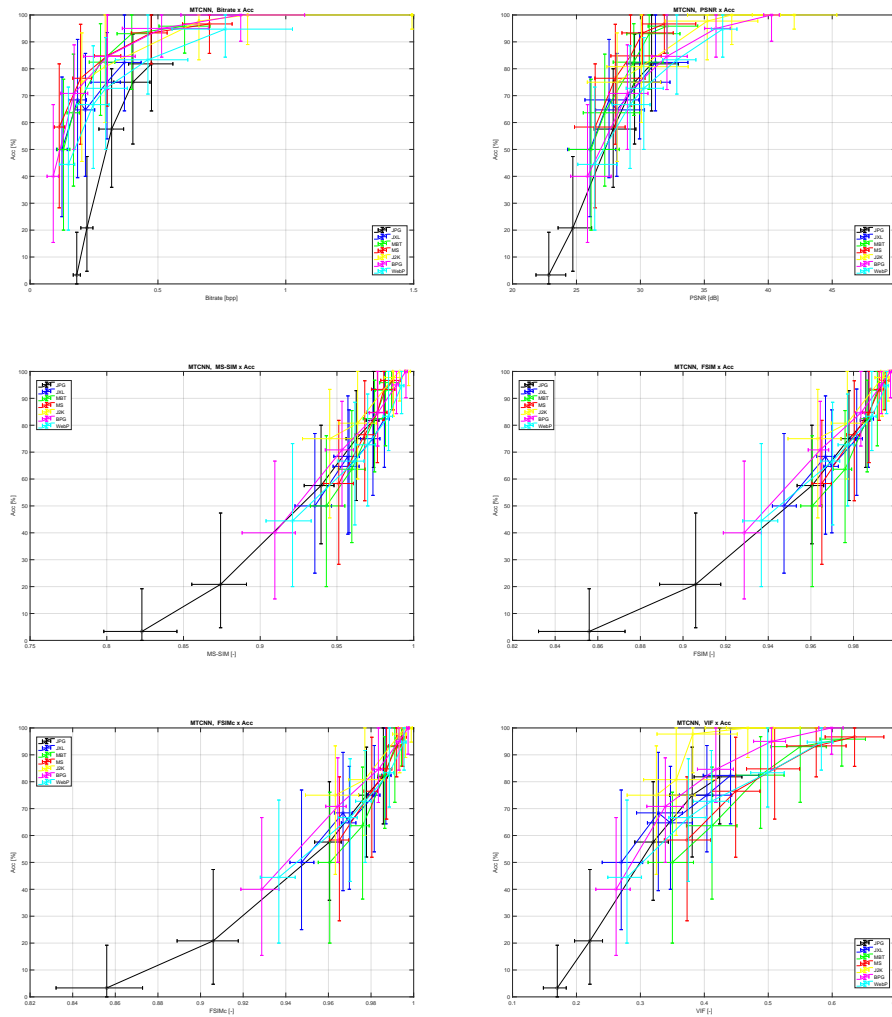


Figure 4.14: Dependence of detection *accuracy* on objective metrics for MTCNN

Compared to the plots for MTCNN detector, in the RetinaFace plots in Figure 4.15 there is a higher IQR for *accuracy* for algorithms other than JPEG. Same as in the MTCNN plots, we can see that the range is getting lower with higher compression level. Unlike in previous case, the *accuracy* for JPEG is close to 0 on the second lowest compression levels. There is no visible *accuracy* drop for JPEG XL codec between second and third compression level. As with the previous figure, we can see that the higher the metric value is, the difference in *accuracy* between individual compression algorithms is lower.

4. Analysis of the results

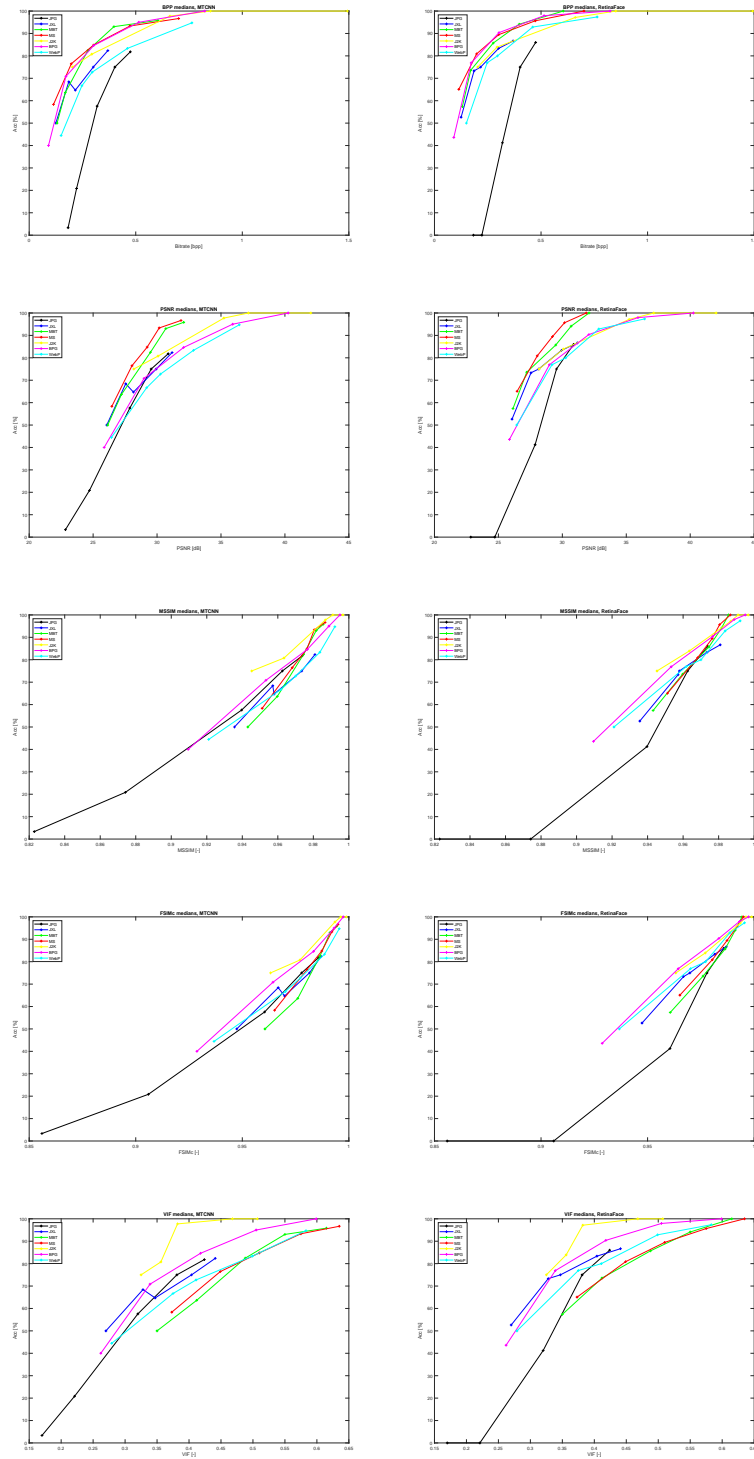


Figure 4.16: Medians without Error bars with MTCNN on the left, RetinaFace on the right

For better clarity, Figure 4.16 shows dependence of *accuracy* on a metric without the first and third quartile. We can see that the *accuracy* of the learning-based algorithms is more or less the same and that they achieve higher *accuracy* on lower compression levels compared to JPEG and JPEG XL. It is also more visible that the MS-SIM, FSIM and FSIMc metrics show lower dependence on the actual algorithm used, since for the highest compression levels the curves overlap with each other, which cannot be stated for bitrate and VIF metric.

If we compare the plots to the calculated correlation in Section 4.7, we can truly see that there is a non-zero correlation between *accuracy* and metrics in regards to medians and interquartile range. The interquartile range of *accuracy* is high for both detectors. This could be caused by the differences in the quality of input images. If we could isolate false positives and negatives from the *accuracy*, it could also possibly decrease this range. The fact that the curves for bitrate and PSNR are highly dependent on used algorithm means that we cannot reliably use those metrics to predict the *accuracy* based on them. With MS-SIM, FSIM and FSIMc metrics we could say that the lower the value of one of those metric is, the lower the *accuracy* will be. As a side effect of those plots we can observe that the JPEG codec achieved much lower quality metrics on the lowest quality settings, but with slightly higher bitrate compared to the the other codecs.

4.7 Spearman's rank correlation

Spearman's rank correlation was used to test a null hypothesis that there is no correlation between a quality metric and the *accuracy* against an alternative hypothesis that there is a non zero correlation between the two. Correlation analysis revealed that there is a moderate (above 0.40) to weak (above 0.2 and below 0.40) correlation between the metrics and the *accuracy*, however the correlation coefficient differs for different compression and face detection algorithms. For every correlation value the statistical significance p was lower than or equal to 0.001.

From the table 4.8 we can see that the correlation coefficient differs between algorithms and between metrics. The strongest correlation between the *accuracy* and the other metrics is for the JPEG algorithm with JPEG XL achieving very weak correlation (0.1) for Bit-rate.

	JPG	JXL	MBT	MS	J2K	BPG	WebP
BPP	0.51	0.10	0.39	0.33	0.29	0.39	0.30
PSNR	0.62	0.35	0.47	0.42	0.47	0.51	0.43
MS-SIM	0.59	0.24	0.43	0.35	0.40	0.47	0.37
FSIM	0.59	0.24	0.48	0.42	0.40	0.48	0.36
FSIMc	0.59	0.24	0.48	0.42	0.41	0.48	0.37
VIF	0.62	0.32	0.56	0.53	0.45	0.55	0.43

Table 4.8: Spearman's rank correlation between *accuracy* and metrics for MTCNN

The table 4.9 shows a strong correlation for the JPEG algorithm and every metric. We can also notice a decrease in the correlation coefficient across most of the values, dropping close to 0.10 and close to zero. Especially low is the coefficient in case of JPEG 2000, . This difference between the tables 4.8 and 4.9 is noticeable especially for PSNR.

	JPG	JXL	MBT	MS	J2K	BPG	WebP
BPP	0.80	0.34	0.31	0.26	0.12	0.45	0.41
PSNR	0.59	-0.06	-0.01	-0.09	-0.10	0.16	0.10
MS-SIM	0.79	0.21	0.26	0.16	0.02	0.35	0.34
FSIM	0.77	0.12	0.18	0.11	-0.05	0.32	0.28
FSIMc	0.77	0.12	0.18	0.11	-0.05	0.31	0.27
VIF	0.73	0.10	0.11	0.03	-0.11	0.30	0.26

Table 4.9: Spearman's rank correlation between *accuracy* and metrics for RetinaFace

From the scatter plots 4.17 to 4.20 we can see an ascending trend in *accuracy* for higher metric, however, these plots do not offer any other insight into the correlation itself. Only plots for Bitrate and PSNR are shown, the other metrics carry the same trend. The colour scheme is as follows: Black for JPEG, Blue for JPEG XL, Green for MBT, Red for MS, Yellow for JPEG 2000, Magenta for BPG and Cyan for WebP.



Figure 4.17: Scatter plots for Bitrate and *accuracy*, MTCNN detector

4. Analysis of the results

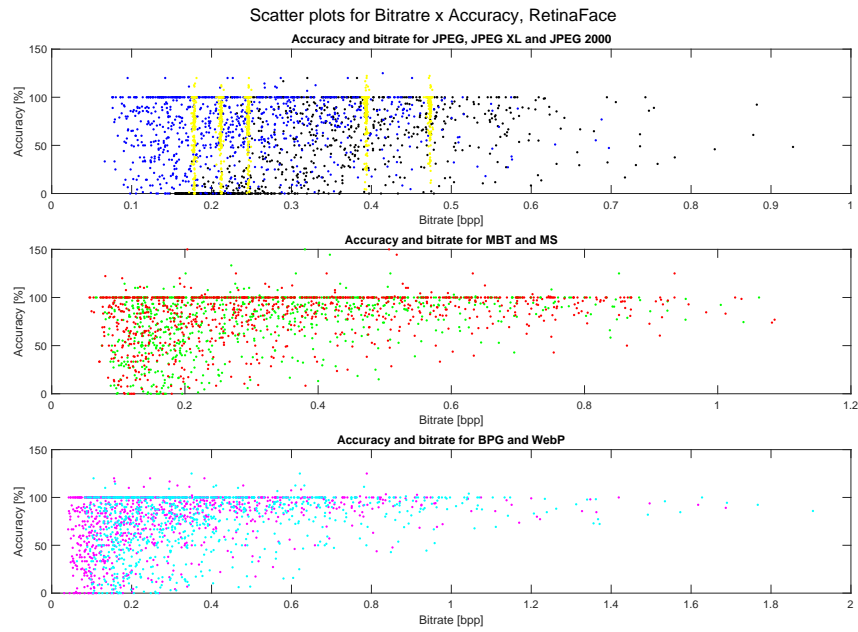


Figure 4.18: Scatter plots for Bitrate and *accuracy*, RetinaFace detector

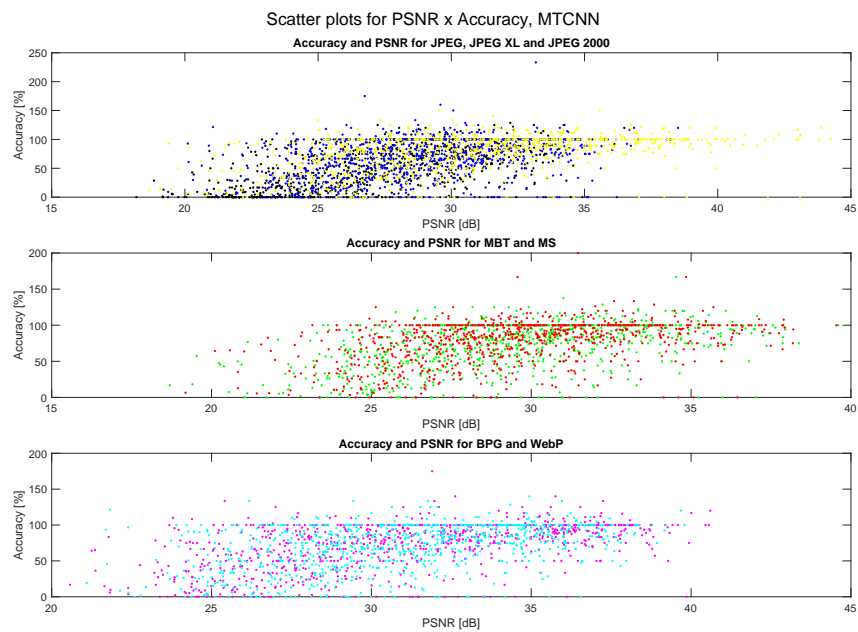


Figure 4.19: Scatter plots for PSNR and *accuracy*, MTCNN detector

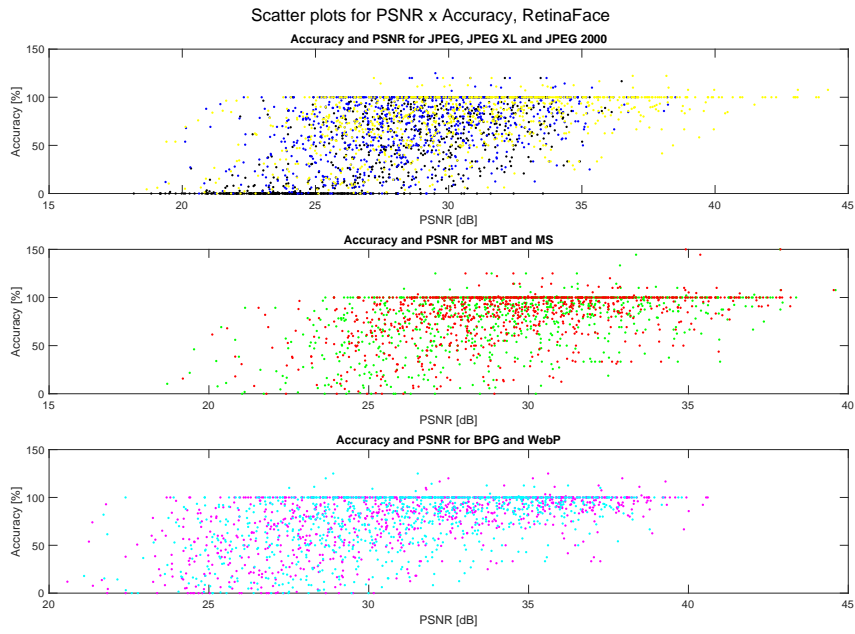


Figure 4.20: Scatter plots for PSNR and *accuracy*, RetinaFace detector

4.8 Suitability of compression algorithm for different tasks

For all of the shown metrics in Figure 4.10 we can say that the closer to the upper left corner of the plot the curves are, the more suitable the compression algorithm is for storing on disk and for viewing by a human observer. By comparing the results for all of the metrics, we can conclude that out of the chosen compression algorithms the ML based compression is better performing in terms of compromise between visual quality and storage, followed by the triplet JPEG XL, BPG, WebP. It was again proved that traditional JPEG is no longer able to compete against other algorithms in terms of compression efficiency, since there are more powerful algorithms available. The differences in quality between the algorithms are noticeable the most under 0.5bpp, over this number the algorithms were able to achieve similar values, therefore, if we are not concerned about storage capacity, JPEG is still recommended to use as it is one of the most common image formats on the internet.

The dependence of the *accuracy* metric on metric revealed that bitrate, PSNR and VIF cannot be used for predicting performance of a face detector based on an objective metric, as the curves for different algorithms followed parallel trends without a convergence point. For MS-SIM, FSIM and FSIMc clearly converge to a point in the north east corner of the figures and the curves follow a similar path no matter which compression they represent. If we wanted to use face detection on data in a storage with limited capacity, we would have to utilize an algorithm that is able to achieve higher perceptual quality for lower bitrate. Therefore based on the results showed in Section 4.4 and Section 4.6, we can conclude that for our input data the selected machine learning algorithms are more suitable for this task selected than traditional approaches.

Chapter 5

Conclusion

Chapter 1 described requirements for a standardized machine learning codecs as well as a testing methodology of those codecs that desire to be selected as a new standard for JPEG AI. It was also stated what else the candidates should offer apart from image compression to be considered a suitable solution. In the second chapter there were described the main differences between traditional and ML compression algorithms. Chapter 3 offers a brief information about computer vision tasks wherein the machine learning algorithms are utilized.

Chapter 4 contains information about conducted experiment. The first part of this chapter describes how to implement and utilize various compression algorithms using Python and Matlab. Information on how to install two machine learning face detection algorithms is provided as well. Image selection from WiderFace library and experiment methodology were covered in the last 2 parts of Chapter 4. It was discovered that although the WiderFace library offers a variety of image categories to choose from, different angles and number of faces per image, the images also come with a variety of visual quality, which might have brought an uncertainty into the final results. *Accuracy* metric, which quantifies detection precision, was defined in the last part of Chapter 4. The results of the experiment are discussed in Chapter 5. It was showed on an example image how visually different the compression algorithms are. Unlike the compression algorithms based on DCT and DST, which had a visible block structure of different size, the machine learning algorithms produced artifacts that smoothed out the details of faces. It was shown that the different compression algorithm indeed affect the face detection, with the artifacts of the machine learning algorithms appearing to be less challenging for the face detection, even though the facial landmarks were often completely missing.

Via the image quality assessment, it was revealed that the machine learning algorithms were able to achieve higher perceptual quality for lower bitrate, however, for MS-SIM, FSIM and FSIMc the differences became less significant as the bitrate increased. For PSNR the machine learning algorithms achieved lower ratio than other algorithms and in VIF plots the algorithms followed their own trends in certain groups. It was revealed that with increasing bitrate, the interquartile range of the bitrate (X-axis) increased whereas the range in metrics (Y-axis) decreased.

By comparing the dependence of the detection accuracy on various metrics it was revealed that the detection is dependant on MS-SIM, FSIM and FSIMc no matter which compression algorithm was studied, which cannot be said for PSNR, VIF and bitrate, which makes the latter less suitable for predicting detection quality. Correlation between the *accuracy* metric and those metrics was studied utilizing Spearman's rank correlation coefficient. The achieved values were dependant on which of the detection algorithm was used. For MTCNN the correlation was between 0.3 and 0.6, but for RetinaFace the values drastically dropped for most of the metrics and algorithms, in some cases very close to 0. The lowest correlation was achieved in PSNR metric for all algorithms but JPEG and for all metrics for JPEG 2000.

The results indicate that machine learning compression algorithms are more suitable for storage as even for lower bitrate the images express higher quality than traditional algorithms and that for face detection higher image quality suggest a better performance of the face detector.

5.1 Possible extension

As an extension to this thesis, a subjective tests could be devised to evaluate which of the compression artefacts are more intervening with the ability of an observer to certainly detect a face. However, this could be difficult due to limited relevant literature on the topic and because, in my opinion, human observer requires much less information to identify a face, like where the hair ends or if there is only one eye visible. Also there is also an option to increase the number of both detection and compression algorithms, which could include more compression quality levels in order to make the correlation more or less significant. Interesting would be training the detectors on compressed data. This might improve detectors' ability to detect faces in both poorly captured and compressed images.



Bibliography

- [1] DIWAN, Tausif, G. ANIRUDH and Jitendra V. TEMBHURNE. Object detection using YOLO: challenges, architectural successors, datasets and applications. *Multimedia Tools and Applications* [online]. [cit. 2023-01-09]. ISSN 1380-7501. Available at: [doi:10.1007/s11042-022-13644-y](https://doi.org/10.1007/s11042-022-13644-y)
- [2] KHERADMANDI, Narges and Vida MEHRANFAR, 2022. A critical review and comparative study on image segmentation-based techniques for pavement crack detection. *Construction and Building Materials* [online]. 321 [cit. 2023-01-09]. ISSN 09500618. Available at: [doi:10.1016/j.conbuildmat.2021.126162](https://doi.org/10.1016/j.conbuildmat.2021.126162)
- [3] TESTOLINA, Michela, Evgeniy UPENIK, Touradj EBRAHIMI, Andrew G. TESCHER and Touradj EBRAHIMI, 2021. Towards image denoising in the latent space of learning-based compression. *Applications of Digital Image Processing XLIV* [online]. SPIE, 2021-8-1, [cit. 2022-02-20]. ISBN 9781510645226. Available at: [doi:10.1117/12.2597828](https://doi.org/10.1117/12.2597828)
- [4] ASCENSO, Joao M., Pinar AKYAZI, Fernando PEREIRA, Touradj EBRAHIMI, Peter SCHELKENS and Tomasz KOZACKI, 2020. Learning-based image coding: early solutions reviewing and subjective quality evaluation. In: *Optics, Photonics and Digital Technologies for Imaging Applications VI* [online]. SPIE, 2020-4-1, 27- [cit. 2023-01-09]. ISBN 9781510634787. Available at: [doi:10.1117/12.2555368](https://doi.org/10.1117/12.2555368)
- [5] ISO/IEC JTC 1/SC 29/WG 1 (ITU-T SG16), *Coding of Still Pictures*. 85th Meeting. Online. November 2019, [cit. 2022-04-26]. Available at: https://jpeg.org/items/20191203_jpeg_ai_performance_evaluation.html
- [6] ISO/IEC JTC 1/SC29/WG1 N100106. *textitJPEG AI Common Training and Test Conditions*. 94th Meeting. Online. February 2022, [cit. 2022-02-20]. Available at: https://jpeg.org/items/20220209_press.html
- [7] ISO/IEC JTC 1/SC29/WG1 N100095. *Final Call for Proposals for JPEG AI*. 94th Meeting, Online. February 2022, [cit. 2022-02-20]. Available at: https://jpeg.org/items/20220209_press.html

- [17] MUKTI, Ishrat Zahan and Dipayan BISWAS, 2019. Transfer Learning Based Plant Diseases Detection Using ResNet50. In: 2019 4th International Conference on Electrical Information and Communication Technology (EICT) [online]. IEEE, 2019, s. 1-6 [cit. 2023-01-09]. ISBN 978-1-7281-6040-5. Available at: doi:10.1109/EICT48899.2019.9068805
- [18] LI, Yinglong, 2022. Research and Application of Deep Learning in Image Recognition. In: 2022 IEEE 2nd International Conference on Power, Electronics and Computer Applications (ICPECA) [online]. IEEE, 2022-1-21, s. 994-999 [cit. 2023-01-09]. ISBN 978-1-6654-4276-3. Available at: doi:10.1109/ICPECA53709.2022.9718847
- [19] CAI, Lei, Jingyang GAO and Di ZHAO, 2020. A review of the application of deep learning in medical image classification and segmentation. *Annals of Translational Medicine* [online]. 8(11), 713-713 [cit. 2023-01-09]. ISSN 23055839. Available at: doi:10.21037/atm.2020.02.44
- [20] ASGARI TAGHANAKI, Saeid, Kumar ABHISHEK, Joseph Paul COHEN, Julien COHEN-ADAD and Ghassan HAMARNEH, 2021. Deep semantic segmentation of natural and medical images: a review. *Artificial Intelligence Review* [online]. 54(1), 137-178 [cit. 2023-01-09]. ISSN 0269-2821. Available at: doi:10.1007/s10462-020-09854-1
- [21] ZHANG, Hongkai, Hong CHANG, Bingpeng MA, Naiyan WANG and Xilin CHEN, 2020. Dynamic R-CNN: Towards High Quality Object Detection via Dynamic Training. In: VEDALDI, Andrea, Horst BISCHOF, Thomas BROX a Jan-Michael FRAHM, ed. *Computer Vision – ECCV 2020* [online]. Cham: Springer International Publishing, 2020-11-16, s. 260-275 [cit. 2023-01-09]. Lecture Notes in Computer Science. ISBN 978-3-030-58554-9. Available at: doi:10.1007/978-3-030-58555-6_16
- [22] DENG, Jiankang, Jia GUO, Yuxiang THOU, Jinke YU, Irene KOTSIA and Stefanos ZAFEIRIOU. *RetinaFace: Single-stage Dense Face Localisation in the Wild* [online]. In: . 4 May 2019 [cit. 2022-03-25]. Available at: doi:https://doi.org/10.48550/arXiv.1905.00641
- [23] ZHANG, Kaipeng, Zhanpeng ZHANG, Zhifeng LI and Yu QIAO, 2016. Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks. *IEEE Signal Processing Letters* [online]. 23(10), 1499-1503 [cit. 2022-04-05]. ISSN 1070-9908. Available at: doi:10.1109/LSP.2016.2603342
- [24] SIMONYAN, Karen and Andrew ZISSERMAN. *Very Deep Convolutional Networks for Large-Scale Image Recognition* [online]. In: . 10 Apr 2015 [cit. 2022-03-24]. Available at: doi:https://doi.org/10.48550/arXiv.1409.1556
- [25] LIU, Ziwei, Ping LUO, Xiaogang WANG and Xiaoou TANG, 2015. *Deep Learning Face Attributes in the Wild*. In: 2015 IEEE International Conference on Computer Vision (ICCV) [online]. IEEE, 2015,

- Based Image Compression. In: 2019 IEEE International Conference on Image Processing (ICIP) [online]. IEEE, 2019, s. 719-723 [cit. 2022-04-26]. ISBN 978-1-5386-6249-6. Available at: doi:10.1109/ICIP.2019.8803824
- [37] HU, Yueyu, Wenhan YANG, Zhan MA a Jiaying LIU. Learning End-to-End Lossy Image Compression: A Benchmark. IEEE Transactions on Pattern Analysis and Machine Intelligence [online]. 1-1 [cit. 2022-04-26]. ISSN 0162-8828. Available at: doi:10.1109/TPAMI.2021.3065339
- [38] CHAVES, Deisy, Eduardo FIDALGO, Enrique ALEGRE, Rocío ALAIZ-RODRÍGUEZ, Francisco JÁÑEZ-MARTINO a George AZZOPARDI, 2020. Assessment and Estimation of Face Detection Performance Based on Deep Learning for Forensic Applications. Sensors [online]. 20(16) [cit. 2022-04-27]. ISSN 1424-8220. Available at: doi:10.3390/s20164491
- [39] KHODABAKHSH, Ali, Marius PEDERSEN a Christoph BUSCH, 2019. Subjective Versus Objective Face Image Quality Evaluation For Face Recognition. In: Proceedings of the 2019 3rd International Conference on Biometric Engineering and Applications - ICBEA 2019 [online]. New York, New York, USA: ACM Press, 2019, s. 36-42 [cit. 2022-04-27]. ISBN 9781450363051. Available at: doi:10.1145/3345336.3345338
- [40] SERENGIL, Sefik Ilkin a Alper OZPINAR, 2021. HyperExtended Light-Face: A Facial Attribute Analysis Framework. In: 2021 International Conference on Engineering and Emerging Technologies (ICEET) [online]. IEEE, 2021-10-27, s. 1-4 [cit. 2022-05-04]. ISBN 978-1-6654-2714-2. Available at: doi:10.1109/ICEET53442.2021.9659697
- [41] Serengil/retinaface: RetinaFace: Deep Face Detection Library for Python, c2022. GitHub [online]. San Francisco (California), 16 Feb 2022 [cit. 2022-05-04]. Available at: <https://github.com/serengil/retinaface>
- [42] Ipazc/mtcnn: MTCNN face detection implementation for TensorFlow, as a PIP package, c2022. GitHub [online]. San Francisco (California), 9 Jul 2021 [cit. 2022-05-04]. Available at: <https://github.com/ipazc/mtcnn>
- [43] OŠŤÁDAL, Jan, 2022. Performance evaluation of learning-based image compression techniques: Project evaluation for master's thesis. Prague.
- [44] MANTIUK, Rafał K., Anna TOMASZEWSKA and Radosław MANTIUK, 2012. Comparison of Four Subjective Methods for Image Quality Assessment. Computer Graphics Forum [online]. 31(8), 2478-2491 [cit. 2022-05-14]. ISSN 01677055. Available at: doi:10.1111/j.1467-8659.2012.03188.x
- [45] KUMAR, Ashu, Amandeep KAUR and Munish KUMAR, 2019. Face detection techniques: a review. Artificial Intelligence Review [online]. 52(2), 927-948 [cit. 2022-05-15]. ISSN 0269-2821. Available at: doi:10.1007/s10462-018-9650-2
- [46] BELLARD, Fabrice. BPG Image format [online]. Apr 21 2018 [cit. 2022-05-15]. Available at: <https://bellard.org/bpg/>



Appendix A

Digital attachment

On the disk and as a digital attachment to this work scripts used in this work are available. For MATLAB there is *Framework.m*, which serves as a main script for compression, face detection and metrics calculation, *Results.m*, which calculates the *accuracy* metric, correlation, tests the data distribution and calls *Figures.m*, which plots the results and saves them into the folder *resultsPDF_obj*.

For Python there are 4 scripts. First one is *Detector_boxes.py*, which iterates over images in selected folder, uses RetinaFace and MTCNN to detect faces in those images and draws bounding boxes around detections. This scripts should be modified before calling, as it was not written for direct calling from a command line. The scripts *Retina.py* and *MTCNN.py* are callable from command line and take image path as an input. Both of those scripts output number of faces detected in the input. Last python script is *vifp.py*, which takes two images as an input and calculates VIF as an output.

The root directory also contains *input_names.mat*, which is a list of images used from WiderFace database. Outputs generated for this work from *Framework.m* are saved in *Variables.mat* and the plots are in *Thesis results* folder. A set of instruction and a list of websites with implementations is available in *Instructions.txt*. Lastly there is also attached a copy of this thesis and licencing agreement with CTU Prague.