Democratic and People's Republic of Algeria Ministry of Higher Education and Scientific Research

جامعة العربي التبسي - تبسة
Universite Larbi Tebessi - Tebessa

Master thesis

Presented at the University of Tébessi
Faculty of Natural and Life Sciences

Department of: **Mathematics and Computer Science**
Specialty: **Computer Science**
Option: **Systems and Multimedia**

By:
CHABBIA Oussama Bachir

Image compression based on machine learning technics

JURY:

Framer	**MCA**	**MENASSEL Rafik**	University of Tebessa
Framer- assistant	**MCA**	**GATTAL Abdeljalil**	University of Tebessa
President	**MCA**	**DJEDDI Chawki**	University of Tebessa
Examiner	**MCA**	**BENDIB Issam**	University of Tebessa

Dedication

To the one who sacrificed herself for my happiness and my success, to my dear mother, who always encouraged me. May God bless her.

To My dear father (CHABBIA NOUREDDINE), I pray to Allah to keep him by my side.

To My Grandfathers, Grandmother

To My dear brothers Mohamed, Seifeddine, Abdellatif, Takieddine.

To all my paternal uncles and aunts: Dalila, Horia, Fatima, Khedidja, Kamal, Mounir

To all my maternal uncles and aunts: Nedjmeddine, Mohamed, Djalel, Soraya, Linda, Ahlem, Sihem

To Amine Zerouali, Anis Braktia, and Ibrahim Kadri

To Guerti Nabil, Mansouri Chaker, Guerti Moundji, and Belalia Faouzi

To my soulmate, Amira.

To all my family: Chabbia, Chabbi

CHABBIA Oussama Bachir

Acknowledgments

First of all, we would like to thank God,

To have given us the health, the will and the patience to complete our Master's training and to be able to carry out this research work.

I would like to express my deep thanks to my coach Mr.

MENASSEL Rafik who provided me with the subject of this brief and guided me with his valuable advice and suggestions, and the confidence he has shown me throughout this work.

I would also like to thank the jury members for their interest in my research by agreeing to review my work.

I would also like to thank all the teachers in the Mathematics and Computer Science Department and the individual computer teachers.

Finally, I address my sincere sentiments of gratuity and gratitude to all those who have participated in this work from near and far.

Abstract

Image compression is required in a variety of situations. Unfortunately, artifacts emerge whenever a lossy compression method is utilized. Image artifacts, which are created by compression, tend to remove high-frequency detail while also adding noise or minor image structures in some circumstances.

Several technologies have been employed to refine picture compression methods in order to lessen their impact on the human or software user's ability to appreciate images.

We attempted to investigate ways of combining machine learning methods with one of the deep learning approaches, specifically convolution networks for picture compression, in this master's thesis.

Indeed, for this assignment, we employed a technique based on Run Length Encoding and a quantification vector combined with a Convolution Autoencoder, which was efficient and based on principles.

Our method produces more detailed images than generic methods and other machine learning methods combined with deep learning.

Keywords:

Image Compression, Machine Learning, Deep Learning, Run Length Encoding, Vector Quantization, Convolution AutoEncoder

Résumé

La compression d'image est nécessaire dans diverses situations. Malheureusement, des artefacts apparaissent chaque fois qu'une méthode de compression avec perte est utilisée. Les artefacts d'image, qui sont créés par compression, ont tendance à supprimer les détails à haute fréquence tout en ajoutant du bruit ou des structures d'image mineures dans certaines circonstances.
Plusieurs technologies ont été utilisées pour affiner les méthodes de compression d'images afin de réduire leur impact sur la capacité de l'utilisateur humain ou logiciel à apprécier les images.
Dans notre mémoire de Master, nous avons tenté de trouver des façons de combiner les méthodes d'apprentissage automatique avec l'une des approches d'apprentissage profond, en particulier les réseaux de convolution pour la compression d'images.
En effet, pour cette mission, nous avons utilisé une technique basée sur l'encodage de longueur d'exécution et un vecteur de quantification combiné à un AutoEncoder de Convolution, qui était efficace et basé sur des principes.
Notre méthode produit des images plus détaillées que les méthodes génériques et d'autres méthodes d'apprentissage automatique combinées avec l'apprentissage profond.

Mots clés :

Compression d'image, apprentissage automatique, apprentissage profond, codage de longueur d'exécution, quantification vectorielle, convolution AutoEncoder

الملخص

ضغط الصورة هو حاجة محسوسة في كثير من الظروف. لسوء الحظ، في كل مرة يتم فيها استخدام خوارزمية ضغط خاسرة، تظهر القطع الأثرية. تميل القطع الأثرية للصور، الناتجة عن الضغط، إلى التخلص من التفاصيل عالية التردد وفي بعض الحالات يمكن أن تضيف ضوضاء أو هياكل صور صغيرة.

تم استخدام العديد من الأدوات لتحسين طرق ضغط الصور من أجل تقليل تأثير هذه الأخيرة على تقدير الصور من قبل مستخدم الإنسان أو البرمجيات.

في هذه الرسالة الرئيسية، حاولنا دراسة طرق استخدام طرق التعلم الآلي بإحدى طرق التعلم العميق، وهي الشبكات التلافيفية لضغط الصورة

في الواقع، استخدمنا نهجًا يعتمد على أداء ترميز الطول ومتجه القياس الكمي مدمجًا مع مشفر التلافي التلقائي، وفعال ويستند إلى مبادئ لهذه المهمة سمحت لنا بضغط هذه الصور

نهجنا قادر على إنتاج صور بتفاصيل أكثر من الأساليب العامة بالإضافة إلى طرق التعلم الآلي الأخرى التي تتحد مع التعلم العميق

الكلمات الرئيسية:

ضغط الصورة، التعلم الآلي، التعلم العميق، التشفير بطول التشغيل، تكميم المتجهات، التشفير التلقائي

Table of contents

List of Figures

List of Table

General Introduction

When image size is weighed against image quality, storage, and transmission, the digital universe's foundational stakes become evident. Compression is an unavoidable step in making enormous amounts of data more useful in computer networks. The primary goal of image compression is to reduce the amount of data required for a visually accurate copy of the original image. In general, compression algorithms are classified based on the amount of data lost. Reversible approaches are based solely on the reduction of redundancy and do not result in any loss. Irreversible approaches establish a rough representation of the data.

All of these methods have benefits and drawbacks, the goal and by cons is to find a compromise between the various criteria to be checked after a compression algorithm (calculation time, image size, total degradation...etc.), and thus to design a compression system that allows for a better ratio (quality/other criteria).

We will conduct a more extensive investigation of the development of a novel way of image compression based on the integration of a deep learning algorithm and two machine learning methods in this Master's Thesis for the first time.

This Master's thesis is divided into three chapters as follows:

A general introduction:

presenting the context of the study which contains two main axes: image compression and machine learning, then we move on to the objective of our research which is the development of a new compression system based on the use of a deep learning algorithm with two machine learning algorithms

A first chapter: Image compression generalities and machine learning:

This chapter is devoted to the presentation of the main concepts of compression and machine learning. Where we will start the different methods of image compression of the two families (with loss and without loss),

In the second part of this chapter we will discuss the field of machine and deep learning

The second Chapter: Proposed Approach:

In this chapter we focus, first of all, on the work in the context of the integration of deep learning algorithms in image compression. The rest of the chapter will be devoted to explaining our proposed approach to highlight a new compression method based on.

The third chapter: Results and discussion

This last chapter will explain the various results obtained after the experiment on standard images.

We will conclude this Master's thesis with a general conclusion and some perspectives opening the door for further research.

Chapter 1 Image compression generalities and machine learning

I. Introduction

In recent years, there has been a rapid increase in the development and demand of multimedia products, this contributes to insufficient network bandwidth and memory storage.
Image compression is very important for reducing disk space and bandwidth.
Used when loading images. In this chapter, we will focus on this technique as well as the deep learning that can be used to compress images to improve its Performance.

1. Part 1 : IMAGE COMPRESSION

1.1.Image definition

The image is a representation of a person or an object by painting, drawing, photography, films, etc. It is also an organized set of information which after being displayed on the screen has meaning for the human eye.
It can be described as an I(x, y) function with analog brightness continuous defined specific field, so that x and y are the coordinates space of a point in the image and I is a function of light intensity and the color. Under this aspect, the image is not usable by the devise which requires its scanning [1]

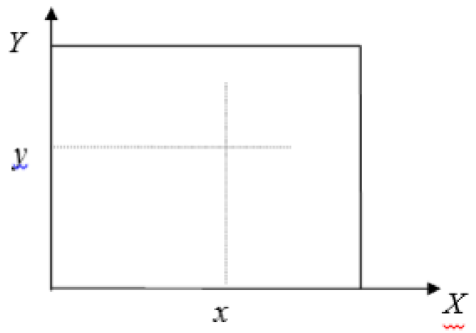


Figure 1 A point in the coordinate image (x, y)

1.2.Digital image

Pixels make up digital images, and each one represents the color of a particular point in the image. We can generate a numerical approximation of an image by measuring its color in a large number of locations. In tables, pixels are grouped in regular rows and columns. A digital image is a two-dimensional image that is represented digitally.

Figure 2 Digital Image

1.2.1. <u>The attributes of the image</u>

- Pixels
- Dimension
- Noise
- Resolution
- Histogram
- Contours an textures
- Luminance
- Contrast
- The weight of image

1.2.2. <u>The different type of image</u>

- Grayscale image
- Monochrome mode
- Color image

1.2.3. <u>Image formats</u>

- Bitmap images :
  - The GIF format (Graphic Interchange Format)
  - JPG or JPEG format (Joint Photographique Experts Group)
  - The PCX format (Picture Exchange Image Bitmap Zsoft)
- Vector Image :

- The EPS format (Postscript /Encapsulated Postscript)
- The CGM format (Computer Graphics Metafile)

1.2.4. <u>Digital image quality</u>

The resolution of a digital image is one of many factors that determine the quality of a digital photo. There are four main factors working together to create the quality of digital pictures:

- The quality of recording device (optical and camera sensor, sensor of the scanner)
- The size (in pixels) of the digital image
- The digital format which it is stored (lossless compression vs lossy compression)
- The photographer's technical competence

1.3. <u>Why compress</u>

The manipulation of images however poses much more complex problems than those of the text. The image is a tow-dimensional space, which poses two major problems [1]

- The volume of data to be processed is much larger (Problems of storage, processing and transmission that appear quickly with the size of the images). [1]
- The structure of this data is much more complex, with some limitations (through image processing, these constraints are resolved or circumvented). [1]

To learn more, we present some examples, which give an idea of the needs involved in the use of digitization technology, in compression as a solution to the problems of storage and transmission of the image [1]:

- One color image in the Red-Green-Blue space, 512x512 pixels in size, each of which is encoded in 8 bits per pixel represents 786 K bytes.
- One 800x600 pixel image in 16 million colors (24 bits per pixel), corresponding to a wallpaper, takes 1 million 400 000 bytes. [1]
- In CIF format (The QCIF format [Quarter of CIF] is defined by 167 pixels on 144 lines for luminance and 88x72 pixels for chrominance. [1] The CIF format [Common intermediate format] is defined by 352 pixels on 288 lines for luminance and 176x144 pixels for chrominance). [1]

Compression and coding consists of reducing the physical size of an information block (by reducing the number of bits per pixel to be stored or transmitted), by exploiting informational redundancy in the image. [1]

1.4.Image compression performance measures

- **Ratio and compression rate**

The compression ratio is one of the most important characteristics of all compression methods, it represents the ratio between the numbers of bits of the canonical form to the number of bits after encoding [1]:

$$\mathbf{CR} = \frac{\boldsymbol{IO\ bit\ number}}{\boldsymbol{Ic\ bit\ number}}$$

Therefore the compression rate is a percentage of the space obtained after compression relative to the total space required the data before compression. [1]

It is defined by:

$$\mathbf{Ratio} = \left(\mathbf{1}-\frac{\mathbf{1}}{\boldsymbol{CR}}\right)*\mathbf{100}$$

- **Entropy**

In an image, entropy is a quantity that characterizes the quality of the information contained in the image. For example an image with all pixels of the same value contains very little information because it is extremely redundant, its entropy is low. On the other hand, an image with all pixels having a random value contains a lot of information, its entropy is strong. [1]

In practice, the entropy of a digital image is inversely related to the probability of the grey levels appearing in the image. By definition zero order entropy H 0 is given by:

$$\boldsymbol{H_0} = \sum_{\boldsymbol{k=0}}^{\boldsymbol{2^R-1}} \boldsymbol{p(k)} * \boldsymbol{lnp(k)ppb}$$

With: P (K) is the probability of the gray levels appearing in the image, K is the gray value and R is the number of bits per pixel. [1]

The H0 entropy of an original image provides the minimum throughput that can be achieved by compression, pixel by pixel without degrading the image, is by the same, a maximum lossless compression ratio. [1]

However, by deterioration of the image exceeded the entropy as indicated on curve E (Quantity of information, Deterioration) below :

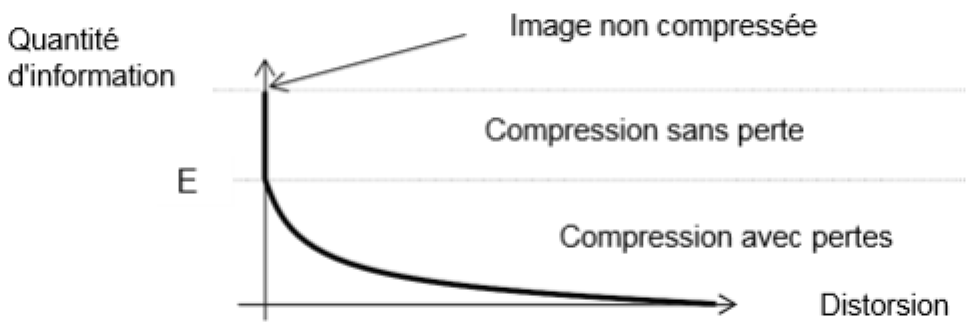


Figure 3 Image deterioration as a function of entropy

- **Distortion measures**

The most important examples for mathematical distortion measurements are the Mean Square Error, Peak Signal/Noise Ratio and Maximum Error:

$$\boldsymbol{MSE = \frac{1}{M*N}\sum_{m=o}^{M-1}\sum_{n=0}^{N-1}[I(m,n) - I'(m,n)]}$$

Signal to Noise Ratio is defined by:

$$\boldsymbol{SNR = 10\,log_{10}\frac{\sum_m\sum_n[x(m,n)^2]}{MSE}DB}$$

The Peak Signal to Noise Ratio for an image whose maximum is $2^R - 1$ denoted by PSNR is determined by the formula:

$$\boldsymbol{PSNR = 10log_{10}\frac{(2^R - 1)}{MSE}\ decibels\ DB}$$

In image compression the PSNR of an 8x $(512)^2$ bit ($(512)^2$ image size indicates a 512 pixel size image, each pixel is coded to 8 bits) is more often defined by:

$$\boldsymbol{PSNR = 10log_{10}\frac{(255)^2}{MSE}}$$

Distortion measurements are very useful in determining the performance of a method relative to other methods. [1]

1.5. The value of image compression

Compression and coding consists of reducing the physical size of an information block (reducing the number of bits per pixel to be stored or transmitted), by exploiting the informational redundancy in the image. Three kinds of redundancies are exploited in image compression:

- The spatial redundancy between adjacent pixels or blocks in the image.
- Temporal redundancy between successive images in a sequence.
- Spectral redundancy between color planes or spectral bands

The most important criteria for any compression method are:

- Quality measurement: The quality of a system is measured by the quality of the image reconstruction.
- The speed of the encoder and decoder.
- Flow reduction: compression ratio depends on application [1]

1.6. <u>General model for analysis of compression methods</u>

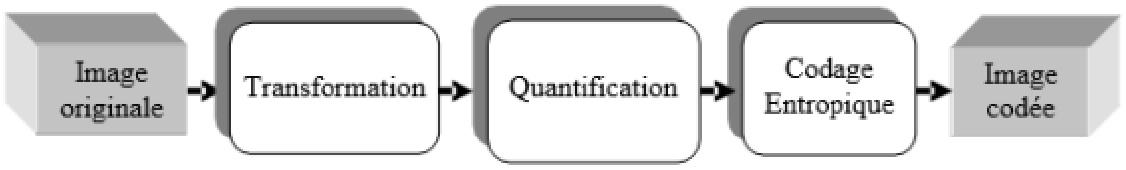


Figure 4 General diagram of compression

➢ **Transformation**

The dependence between each of the pixels and its neighbors (the brightness varies very little from one pixel to another) reflects a very strong correlation on the image. The decorrelation consists in transforming the initial pixels into a set of less correlated coefficients to reduce the volume of information, it is a reversible operation. [2]

➢ **Quantification**

The quantification of coefficients aims to reduce the number of bits needed for their representations. It represents a key step in compression. It approximates each value of a signal by an integer multiple of a quantity q, called elementary quantum or no quantification. It can be scalar or vector. [2]

➢ **Entropic coding**

Entropic encoding performs lossless encoding on quantified values. This last step is necessary in lossless methods, but it is often also present in irreversible algorithms, since transformed and quantified values contain more redundancies. [2]

1.7. <u>Compression methods</u>

➢ lossless methods

- **Run Length Encoding**
- The coding of HUFFMAN
- Arithmetic coding
- Coding LZW (Lempel-Ziv-Welch)

- **lossy methods**
  - **Vector Quantization**
  - Transform coding
  - Coding in sub-bands

1.8.<u>Run Length Encoding</u>

Run Length Encoding is probably the simplest compression method, very useful in the case of repetitive data. This technique replaces identical symbol sequences (pixels), called runs, with shorter symbols. The general idea of this method is to replace consecutive repetitive occurrences of a symbol with an occurrence of the symbol followed by the number of occurrences.

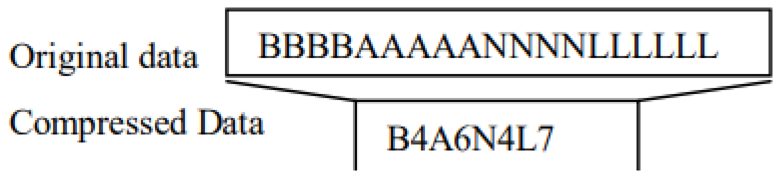


Figure 5 Run Length Encoding

1.9.<u>Vector Quantization</u>

It offers a high compression ratio and a simple decoding process. This technique consists of developing a dictionary of fixed-size vectors, called code vectors. A given image is then divided into non-overlapping blocks called image vectors. Then, for each image vector, the closest vector in the dictionary is determined and its index in the dictionary is used as encoding of the original image vector. Because of their quick search capabilities on the decoder side, VQ-based coding schemes are particularly interesting for multimedia applications.

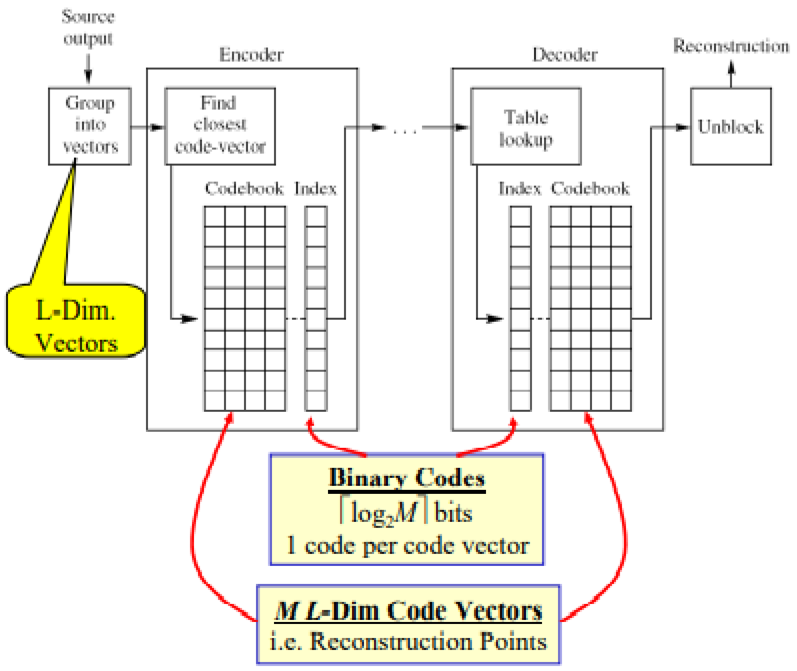


Figure 6 Run Length Encoding

2. Part 2 : Machine Learning

2.1. Machine learning definition

Machine learning is an artificial intelligence (AI) application that allows systems to learn and improve automatically from the experience itself without being explicitly programmed. Machine learning focuses on developing computer programs that can access data and use it to learn for themselves. [4]

➢ Supervised learning

The technique of annotating data for supervised learning is termed labeling. In supervised learning, input data is tagged with expert information detailing what the output or expected response will be. There are neural networks, Bayesian networks, decision-pulled algorithms, and support vector machines among its algorithms.

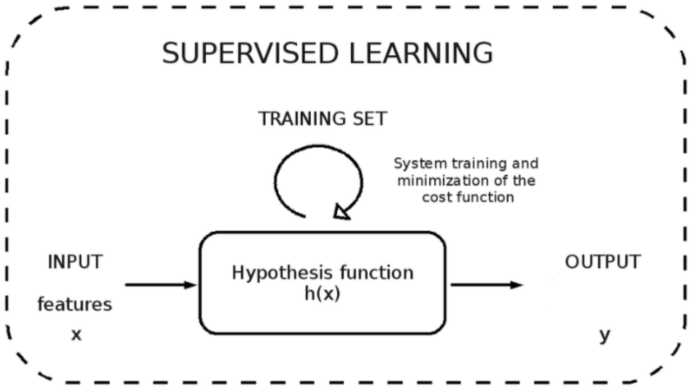


Figure 7 Diagram of a supervised model

- **Unsupervised Learning**

The algorithm is required to discover the structure data and the relationship of the variables by observing a raw data set in unsupervised learning because no additional data or meta-data is offered to it. K-means clustering, hierarchical clustering, and main component analysis are among its methodologies.

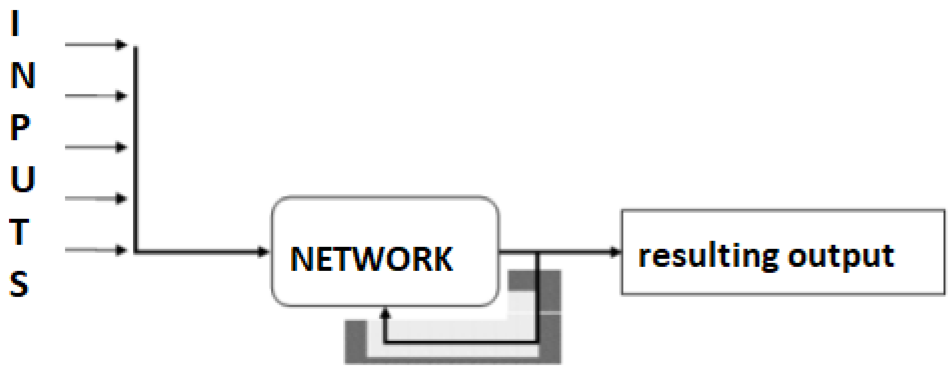


Figure 8 Diagram of an unsupervised model

- **semi-supervised learning**

A limited set of learning data is tagged in semi-supervised learning, however there are substantial gaps in labeling. When we know that a limited number of factors lead to a result but don't know the extent of the variables involved, we apply this strategy. The term "enhanced learning" refers to a type of semi-supervised learning. An expert informs the algorithms whether its output is correct or not, which is referred to as enhanced learning. Expectation–maximization, transductive support vector machine, and Markov decision processes are among its algorithms.

2.2. Neural networks (NRs)

A neural network is a series of algorithms that attempt to recognize the underlying relationships in a data set through a process that mimics the functioning of the human brain. In this sense, neural networks refer to systems of neurons, of an organic or artificial nature. Neural networks can adapt to changing inputs, so the network generates the best possible result without having to redefine the output criteria [5].

These are a series of algorithms that mimic the operations of the human brain to recognize the relationships between large amounts of data.

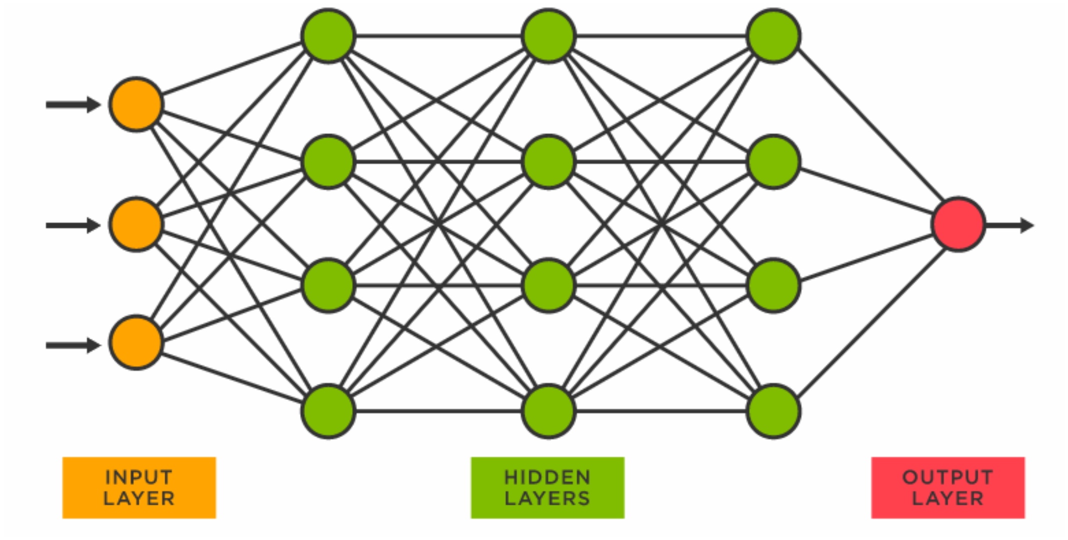


Figure 9 Representation of neural networks

2.3.<u>Deep Learning</u>

Deep Learning is a machine learning technique based on the model of neural networks: tens or even hundreds of layers of neurons are stacked to bring greater complexity to the establishment of rules. [6]

2.4.<u>The functioning of an artificial neuronal network</u>

An artificial neural network consists of processing units called neurons. A neuron receives multiple inputs from different sources, and has only one output.

A weight is the connection to the signal. The product of the weight and input gives the strength of the signal.

(Wi) weights are numerical values that represent either the power value of inputs or the power value of connections between neurons. There are operations that happen at the level of the artificial neuron. The artificial neuron will make a product between the weight (w) and the input value (x), and then add a bias (b), the result is transmitted to an activation function (f) which will add a certain non-linearity. [7]

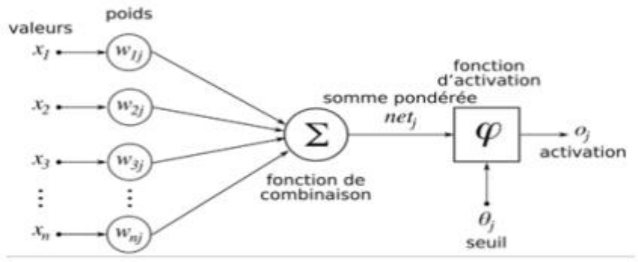


Figure 10 the functioning of an artificial neuronal network

2.4.1. <u>The activation functions</u>

An activation function is used when introducing a non-linearity in the functioning of the neuron. It usually has three intervals: [8]

- Below the threshold (seuil), the neuron is non-active (often in this case, its output is 0 or -1).
- A transition phase around the threshold.
- Above the threshold, the neuron is active (often in this case, its output is worth 1).

There are different functions used for activation. One of the most commonly used activation functions is the sigmoid function

2.4.2. <u>The sigmoid function</u>

In mathematics, the sigmoid function (known as aussicourbe) is defined by:

$$\boldsymbol{F(x) = \frac{1}{1 + e^{-x}}}$$

Knowing that Sum = $\boldsymbol{\sum_{i=1}^{n} XiWi}$

The sum is the weighted sum of inputs multiplied by the weights between one layer and the next. The interconnection of these individual neurons forms the neural network. [8]

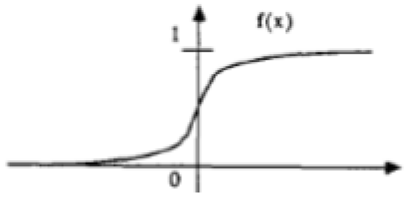


Figure 11 Graphical presentation of the sigmoid function

2.4.3. <u>The ReLu function</u>

ReLu is a non-linear activation function used in multilayer neural networks or deep neural networks. This function can be represented as follows:

F(x) =max (0.x)

Where x = an input value

According to equation, the ReLu output is the maximum value between zero and the input value. An output is equal to zero when the input value is negative and to the input value when the input is positive. [8]

We can rewrite equation 1 as follows:

$$\boldsymbol{F(x) = \begin{cases} 0, si\ x < 0 \\ x, si\ x \geq 0 \end{cases}}$$

Where x = an input value

2.5.<u>Convolution Neural Networks (CNN)</u>

Convolution neural networks are a particular type of multilayer neural networks. They are widely used in pattern recognition problems. Compared to other techniques, convolution neural networks have the advantage of recognizing visual models with minimal pretreatment and extreme variability of models. It is also worth noting the exceptional architecture that we will discuss in detail in the next subsection.

2.5.1. <u>Architecture of convolution neural networks</u>

CNN architecture consists of distinct layers that transform the input volume into an output volume. We present some types of these layers that are most commonly used:

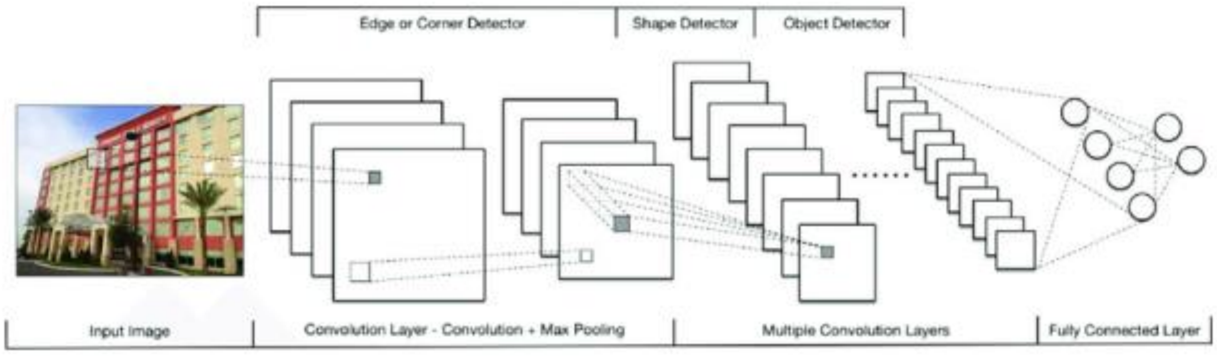


Figure 12 Architecture of convolution neural networks

➢ **The convolutional layer**

The Convolutional layer applies filters on these inputs (example a color image), Each filter is small in terms of dimensions of height, width and especially the depth of this filter which must be the same as the depth of the input (color images with depth of 3). The filter is initially positioned in the upper left corner on the image matrix. It performs a multiplication between these values and that of the selected section of the matrix, which generates a new value. This process is repeated on each new position (the move is one step to the right). The collection of these generated values constructs a result matrix called the feature map (feature map). Before the convolution step is performed, we need to define three parameters that directly control and affect the size of the feature map (feature map). These parameters are as follows:

- **First parameter: Depth of the layer**

The feature map depth (feature map) is the same as the number of filters we use for the convolution operation. Convolution performed on the image using three separate filters produces three different feature maps (feature map)

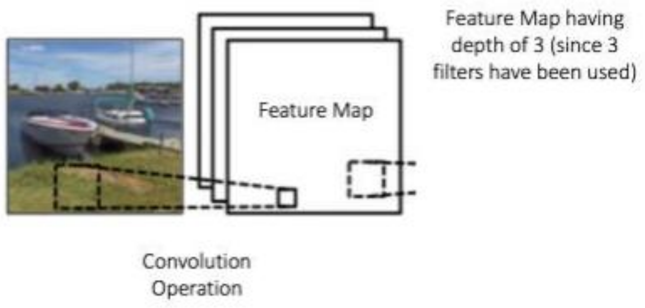

Feature Map having depth of 3 (since 3 filters have been used)
Feature Map
Convolution Operation

Figure 13 Feature map depth

- **Second parameter: Stride**

The stride is the number of pixels by which we drag our filter on the image matrix. The stride is 1 by default. Note that using a larger stride will produce smaller feature maps

- **Third parameter : The margin at zero**

Padding is a zero margin that is placed around the image. In many cases, the filter does not fit perfectly on the image matrix. This causes us to choose between two options:

  - Fill (padding) the image with zeros (zero-padding)
  - Keep only a valid part of the image by removing the part where the filter does not cover it.

Note that the use of filling helps to solve another relevant problem since if we continue to apply convolutional layers, the volume size will decrease faster.

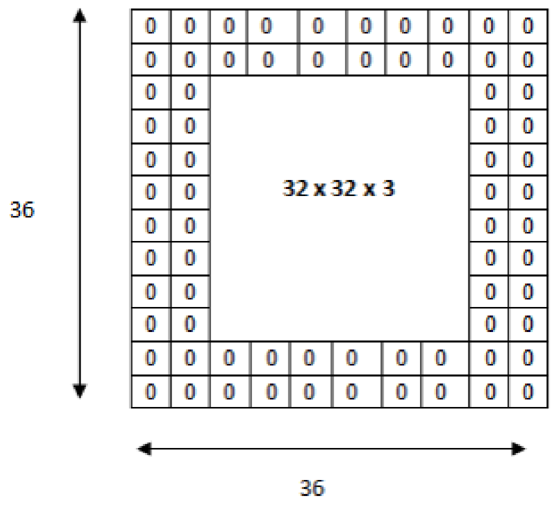


Figure 14 Fill the image with zeros

- **The pooling layer**

The pooling layer takes a filter (normally of size 2x2) and a stride of the same length. It then applies to the input volume and generates the maximum number in each region around which the filter is convoluted. The idea behind pooling in convolution neural networks is that the exact location of an entity is not as important as its relative location relative to other entities. This layer also considerably reduces the spatial dimension (length and width but not depth) of the input volume.

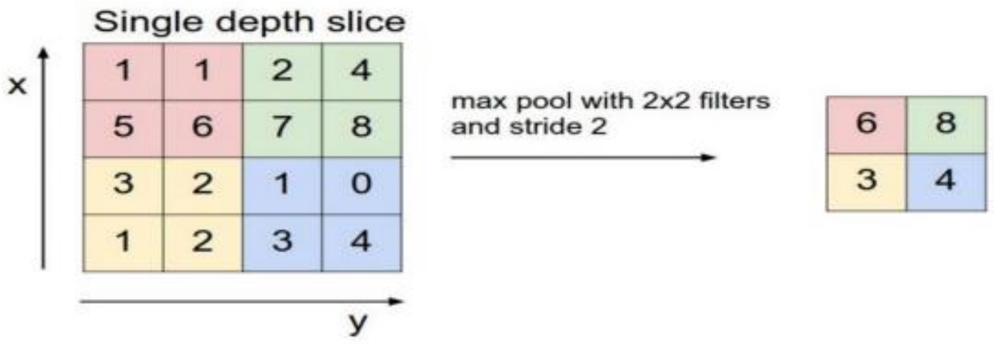


Figure 15 Max pooling function

- **The correction layers**

The ReLu (Rectified Linear Unit) is one of the most used activation functions in deep learning. It introduces non-linearity into the convolution network and is usually applied after the convolutional layer. The ReLu layer modifies all negative input values to zero by applying the function f(x) to all input volume values.

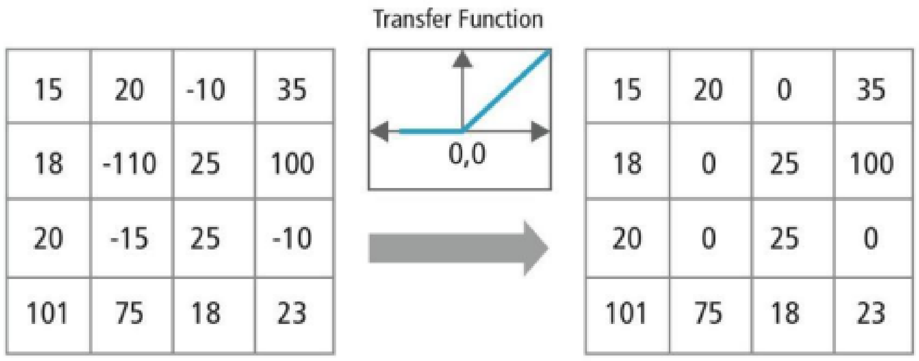


Figure 16 Transfer function

- **Fully connected layers**

After several layers of pooling and convolution, the learning of the characteristics on the network is completed. Our network now switches to classification via fully connected layers (fully connected layers). Each neuron in one layer connects to the other, which means that this layer is connected to all previous activations. This layer takes an input volume and generates a N-dimensional vector where N is the number of classes known by the network.

2.6.<u>Application areas of Deep Learning</u>

Deep learning applications are used in a variety of industries, from automated driving to medical devices.

- Automated driving
- Aerospace and Defense
- Medical research
- Industrial automation
- Treatment of the natural language
- Visual recognition
- Facial recognition
- Text generation
- Etc.…

II. <u>Conclusion</u>

The main objective of this chapter was, first, the presentation of general information on image compression and machine learning.

We found that there is a link, even a strong one, between the two lines of research that researchers have already begun to explore

In the next chapter we will see, the various contributions fairly in the field of image compression optimized by Machine Learning. (Deep learning)

Next, we detail our Image compression meth

Chapter 2 Proposed compression system

I. Introduction

Improving image compression has recently become the focus of researchers in this area, and as mentioned earlier, researchers were interested in improving compression.
In this chapter, we will illustrate some of the important work done in compression, and then we will propose our approach.

1. Related work

In 2018 Zhengxue Cheng, Heming Sun, Masaru Takeuchi and Jiro Katto [9] they propose an image compression architecture based on an AutoEncoder Convolutif (CAE), their main contributions are in two orders:

1. To replace transformation and reverse transformation in traditional codecs, they design a symmetrical CAE structure with multiple sub-sampling and over-sampling units to generate a low-dimensional characteristic map, they optimize this CAE by using a function of approximate loss of rate distortion
2. To generate a more compact representation on the energetic plane they propose a rotation based on component analysis to generate more zeros in the characteristic map, then quantification and entropy encoder are used to further compress the data

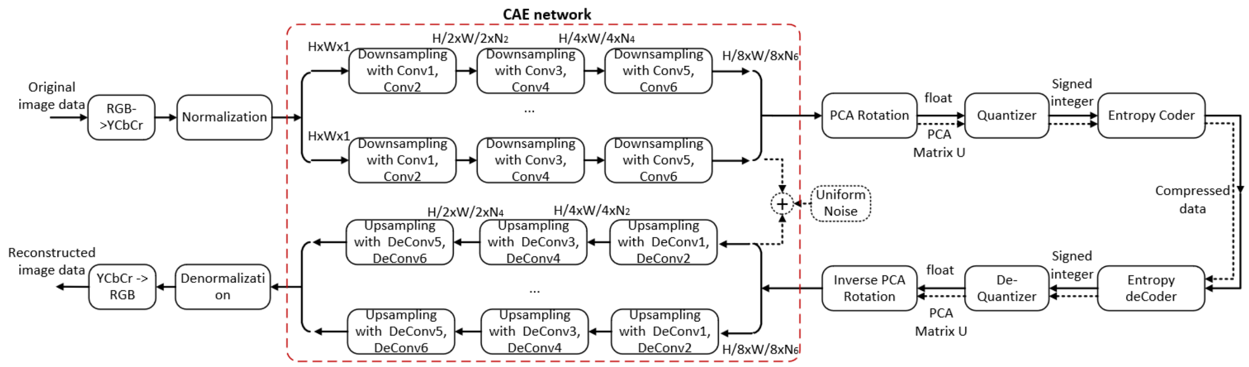


Figure 17 Block diagram of the proposed CAE based image compression

In the same year 2018 Haojie Liu, Tong Chen, Qiu Shen, Tao Yue and Zhan Ma [10] they proposed an image compression system based on a deep residual network (CNN), It optimized image compression end-to-end rate distortion performance. The overall structure consists of a front encoder, a quantifier, a rear decoder, an optimization of the distortion rate (that is, the estimation of the rate and the measurement of the distortion) and visual improvement subsystems.

They create the deep neural network gradually by transfer learning to achieve speedy convergence, that is, using the network built at a light compression (lower quantification) to learn the network at a larger compression ratio (higher quantification). They also added perception loss and contradiction loss in the end-to-end optimization pipeline to generate texture and sharp detail for noticeable increases in visual quality, prompted by the aforementioned perceptual gains utilizing GAN and VGGnet.

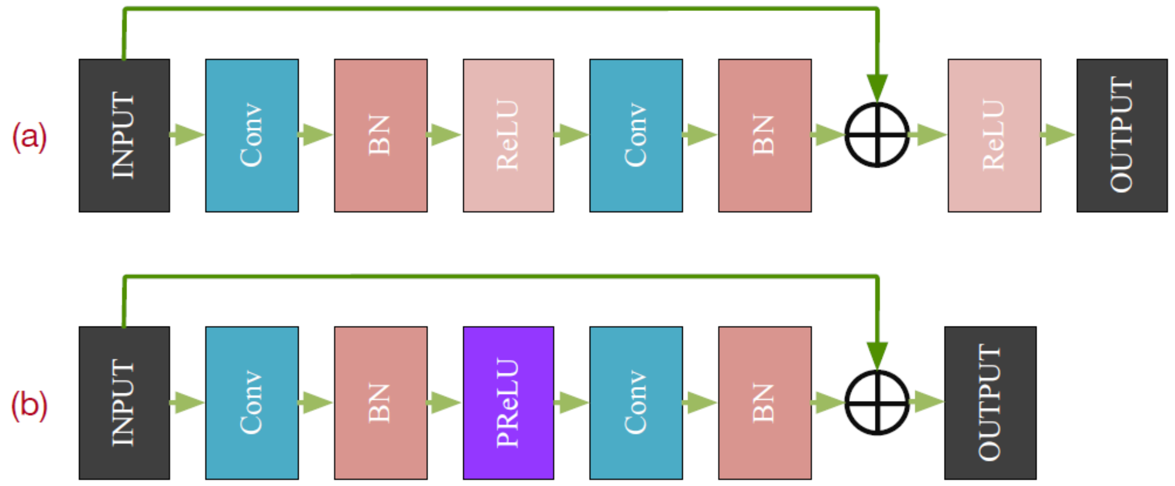


Figure 18 A residual units with (a) the default design and (b) their proposed architecture with ReLU changes.

They demonstrate how deep CNN may be utilized to remove compression artifacts by directly improving the SSIM on reconstructed images, and how this technique leads to superior performance on numerous benchmarks. SSIM is a better picture quality model than PSNR or MSE, but it is still too simplistic and unable to describe the complexity of the human perceptual system. As a result, they rely on a generative adversarial network to build improved reconstruction models, as it is not essential to define a loss directly modeling image quality. They conducted a variety of experiments in order to assess the varied advantages of the various types of networks mentioned in this article. Subjective and objective evaluations are used. First, they show that not only is the objective SSIM metric improved, but that object detector performance on highly compressed images increases as well, which is notably true for GAN artifact removal. Second, human observers claim that their GAN reconstruction is more accurate than uncompressed versions of images.

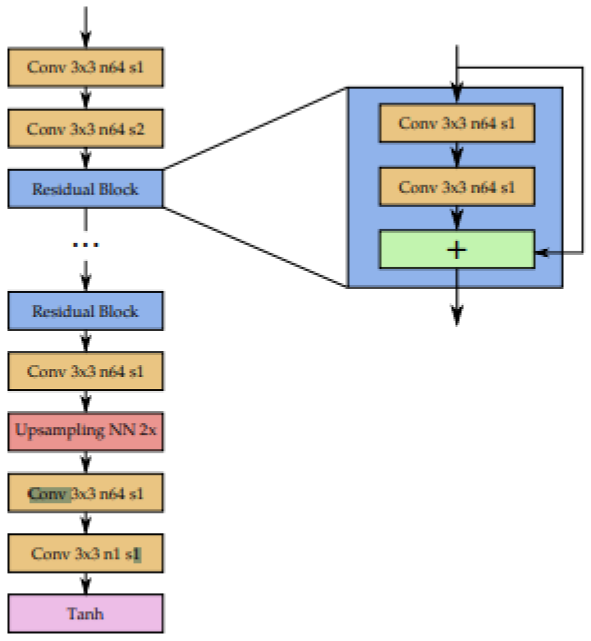


Figure 19 Generator Network Architecture, with n denoting the number of filters and s representing the stride value for each Convolutional Layer

In October 2018 Eirikur Agustsson, Michael Tschannen, Fabian Mentzer, Radu Timofte and Luc Van Gool [12], they offer an adversarial generative network (GAN)-based framework for compressing extreme scholarly images, which allows for the creation of visually attractive images at significantly lower rates than earlier techniques. This is made possible by their GAN learnt compression formulation, which combines a generator/decoder that operates in full resolution on the picture and is trained in conjunction with a multi-scale discriminator. Furthermore, if the original image has a semantic label map, their method can fully synthesize the unimportant parts in the decoded image, such as streets and trees, using the label map. The semantic label map and the preserved area are stored together.

In June 2017 Shibani Santurkar, David Buddeny and Nir Shavit [13] Traditional video and picture compression techniques rely on hand-crafted encoder/decoder pairs (codecs), are inflexible, and are unaffected by the data being compressed. They present the notion of generative compression, which is data compression based on generative models, and argue that this is a path to take in order to provide more accurate and visually beautiful reconstructions for image and video data at considerably greater compression levels. They also show that generative compression outperforms typical variable-length coding schemes in terms of binary error rates (e.g., noisy wireless channels).

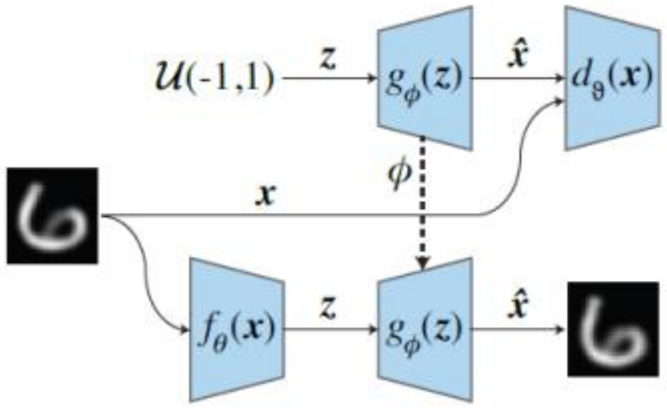


Figure 20 Generative compressions for image data

In the same year, Mu Li et al. [14], the bit rate of different sections of the image is adapted to the local content, and the bit rate sensitive to the content is assigned according to a weighted content of the vector of importance, driven by the fact that the content of local information is spatially variable in an image. To regulate the compression ratio, the sum of the vectors of importance can thus be used as a continuous alternative to discrete entropy estimate. The binariser is a device that converts binary data into binary (the function is introduced to approximate the binary operation in retro propagation to make it differentiable). From beginning to end, the encoder, decoder, binariser, and vector of significance can all be optimized together. And a convolutional entropy encoder for lossless compression of critical vector and binary codes is described. Experiments show that this approach greatly outperforms JPEG and JPEG 2000 in terms of the Structural Similarity Index (SSIM), and can generate a far better visual result with clean edges, rich textures, and less artifacts in low bit rate picture compression.

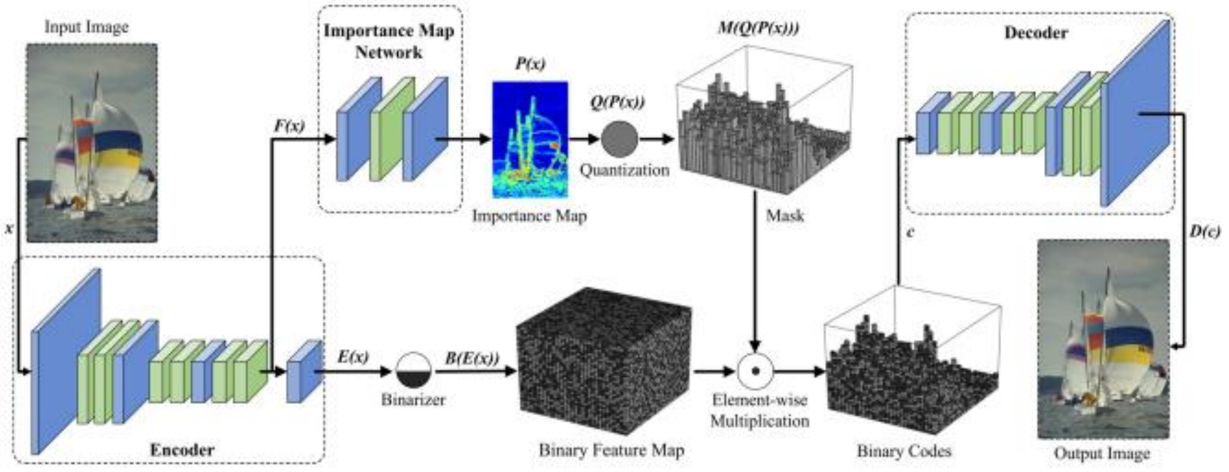


Figure 21 Proposed system of Mu Li and al.

2. <u>Proposed methodology</u>

Image compression is about finding an image of good quality and smaller size than the original image

So after these studies, we aim to establish an image compression system based on neural networks and quantification vectors combined with the Run Length Encoding (RLE) technique, and the proposed system is illustrated in the following figure

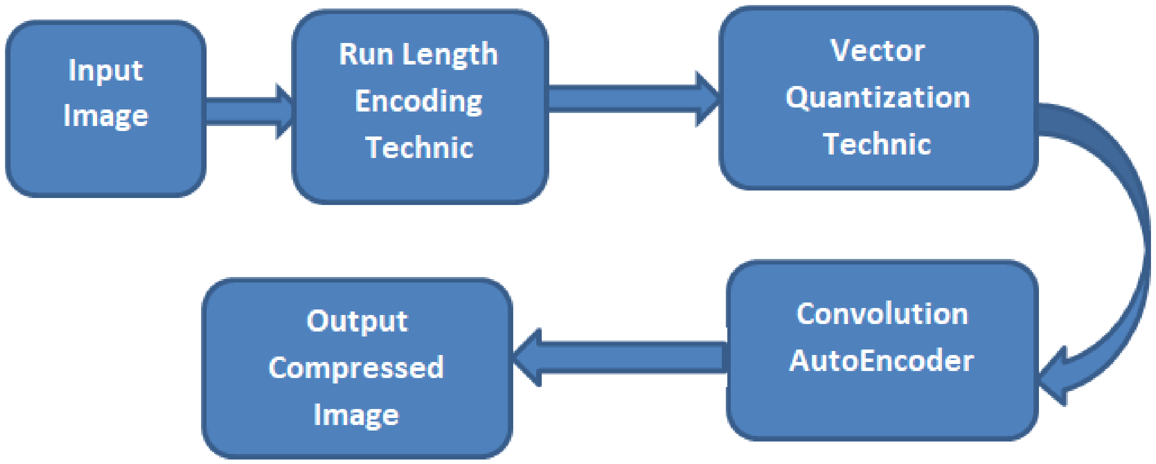


Figure 22 The global structure of our system

Our approach is to concatenate two machine learning compression methods with a deep learning method convolution autoencoder (CAE)
first, have to start with the first part of CAE, the import of dataset and the creation of convolution model follows it to train and test, and finally save the model
Second, follow it with the Run Length encoding (RLE) method,
all the above, have to import the image, then have a converter on array R,G,B, then, have them saved as raw.text file, and after compress and save them as file rle.text and finally decode the last one and display it
third, the vector quantization method have to import the compresser_RLE image, then extract the pixels from dataset and store them in arrays, and after that have to generate the codebook, in order to create it we still only compress and store it,
Finally, have to call the CAE model and compress the image compresser_VQ.
Here is our run length encoding and vector quantization algorithm:

Algorithm 1 RLE:

```
Algorithm 1 Run Length Encoding RLE
1.  RLE function
2.  Read the image
3.  Normalize the image
4.  Def loop raw {
5.  Convert image to table
6.  2-code the table with the RLE method}
7.  Def. loop all {store pixels in R, G, B table}
8.  Def. run {
9.  1-save the image as raw.text file
10. 2-save the tables as a file rle.txt}
11. Def. decoder {decoder file rle.txt}
12. Display the image and store it
13. End function
```

<u>Algorithm 2 VQ:</u>

```
Algorithm 2 Vector Quantization VQ
1.  Load image compressed_RLE
2.  Normalize image
3.  Store in a list
4.  Transform into table
5.  Transform the array into a feature vector
6.  Generate the codebook
7.  Creation of 3D matrix to use to generate pixels
8.  Compress image
9.  Display and store
10. End function
```

<u>Algorithm 3 CAE:</u>

```
Algorithm 3 Convolution Autoencoder CAE
14. Import dataset
15. Normalize dataset
16. Reshape input data for the model
17. Model creation
18. train the model
19. Prediction model(get compressed image )
20. Choose random image from dataset to compress it
21. Save model
```

II. Conclusion

In this chapter we have cited recent work and the various methods proposed in the field of image compression improvement.

Thus we have presented our compression method

We will see in the next chapter, our results obtained, thus the different logical and physical tools used in our work.

Chapter 3 Results and Discussion

I. Introduction

In Chapter 2, we have detailed our proposed approach of using machine learning and deep learning to compress images

In this chapter we will look at the experimental results obtained.

1. Equipment used

To realize our application we used a portable microphone that has the following specifications:

Brand	**Dell**
Model	**Latitude E7250 - Windows 10**
Processor	**Intel(R) Core(TM) i5-5300U CPU @2.30GHz**
RAM	**8, 00 Go**
System uses	**Windows 10 Professional**
Graphics card	**Intel(R) HD Graphics 5500**

2. Presentation of development tools

This section describes the development tools used in the proposed system design and implementation.

➢ Python

Python is an object-oriented, high-level interpreter programming language. Python’s simple and easy-to-learn syntax emphasizes readability and therefore reduces the cost of maintaining the program. Python supports modules and packages, which is why python is used by a large community of developers and programmers. [15].

➢ TensorFlow

Is an open-source platform for creating machine learning applications. It is a library of symbolic mathematics that uses data flow and differentiable programming to perform various tasks focused on the formation and inference of deep neural networks. It allows developers to create machine learning applications using various tools, libraries and community resources [16].

- **Keras**

Keras is an application of high-level neural networks, written in Python and capable of running on TensorFlow. Keras was developed to allow rapid experimentation. It is used by researchers to be able to do quality research and move from the idea to the result as quickly as possible. Keras are used to build rapid prototyping and support convolutif and recurring networks and runs on CPU and GPU. [17]

- **Google Colaboratory**

Colab est un environnement de notebook Jupyter gratuit qui fonctionne entièrement dans le nuage qui ne nécessite pas de configuration et les notebooks que vous créez peuvent être édités simultanément par les membres de votre équipe - exactement comme vous éditez des documents dans Google Docs [18].

Google Colab prend en charge de nombreuses bibliothèques d'apprentissage automatique populaires qui peuvent être facilement chargées dans votre notebook il vous offre l’exécution basée sur le GPU en évitant la plantation des machines à faibles ressources

3. Dataset

The CIFAR-10 dataset consists of 60000 32x32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.[19]

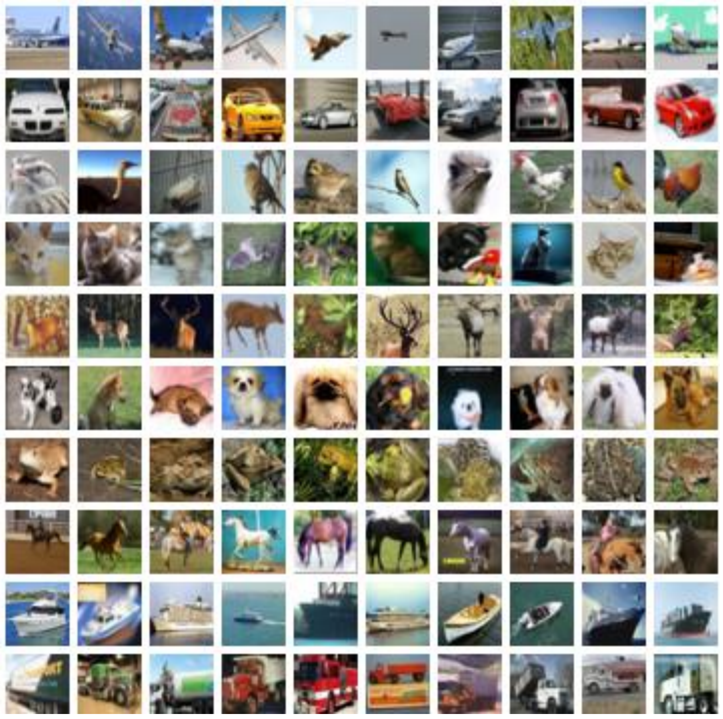
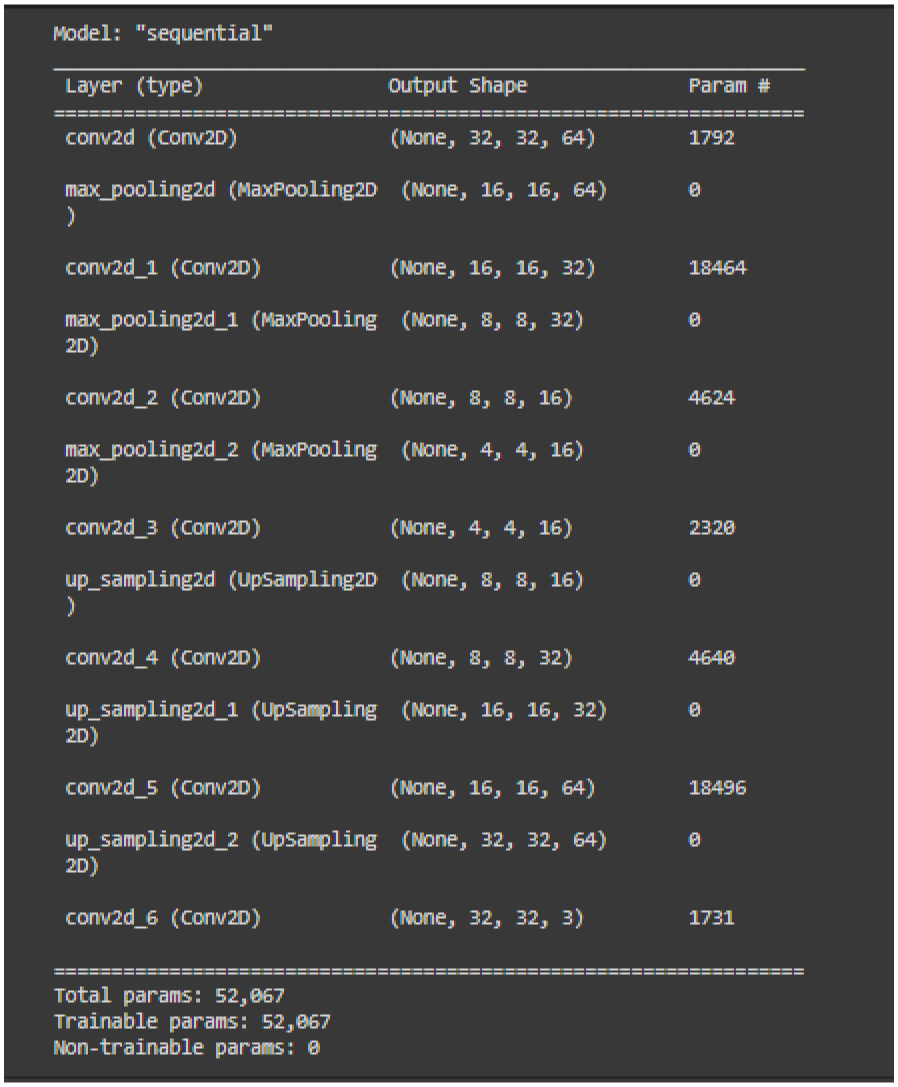


Figure 23 some image of the dataset cifar10

4. Algorithm and Implementation

➢ Auto-encode architecture

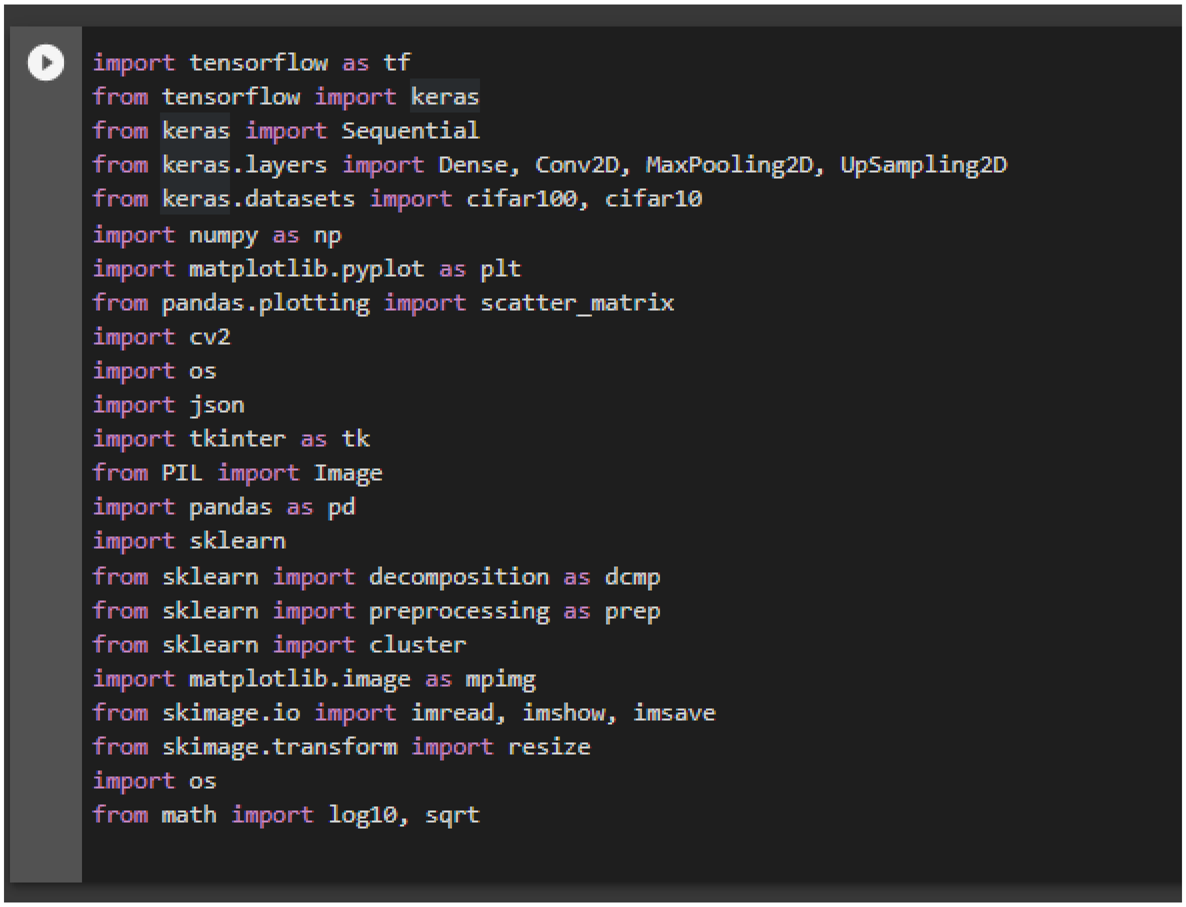
Here is the synthesis of our architecture obtained from the execution



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 32, 32, 64)        1792

 max_pooling2d (MaxPooling2D  (None, 16, 16, 64)       0
 )

 conv2d_1 (Conv2D)           (None, 16, 16, 32)        18464

 max_pooling2d_1 (MaxPooling  (None, 8, 8, 32)         0
 2D)

 conv2d_2 (Conv2D)           (None, 8, 8, 16)          4624

 max_pooling2d_2 (MaxPooling  (None, 4, 4, 16)         0
 2D)

 conv2d_3 (Conv2D)           (None, 4, 4, 16)          2320

 up_sampling2d (UpSampling2D  (None, 8, 8, 16)         0
 )

 conv2d_4 (Conv2D)           (None, 8, 8, 32)          4640

 up_sampling2d_1 (UpSampling  (None, 16, 16, 32)       0
 2D)

 conv2d_5 (Conv2D)           (None, 16, 16, 64)        18496

 up_sampling2d_2 (UpSampling  (None, 32, 32, 64)       0
 2D)

 conv2d_6 (Conv2D)           (None, 32, 32, 3)         1731

=================================================================
Total params: 52,067
Trainable params: 52,067
Non-trainable params: 0
```

Figure 24 Auto-encoder architecture

Here's a sample of my code:



```
import tensorflow as tf
from tensorflow import keras
from keras import Sequential
from keras.layers import Dense, Conv2D, MaxPooling2D, UpSampling2D
from keras.datasets import cifar100, cifar10
import numpy as np
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix
import cv2
import os
import json
import tkinter as tk
from PIL import Image
import pandas as pd
import sklearn
from sklearn import decomposition as dcmp
from sklearn import preprocessing as prep
from sklearn import cluster
import matplotlib.image as mpimg
from skimage.io import imread, imshow, imsave
from skimage.transform import resize
import os
from math import log10, sqrt
```

Figure 25 Part of the code in Google Colaboratory

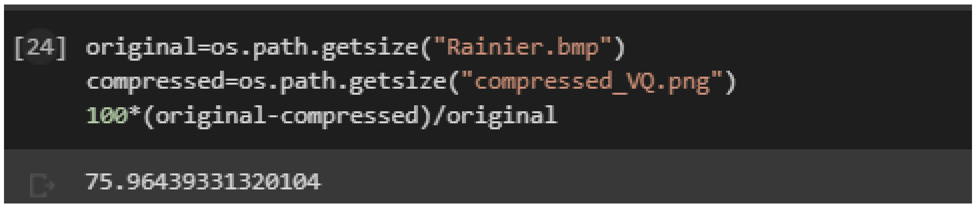
5. <u>Experimental results</u>

➢ <u>Method Vector Quantization</u>

In this part as we talked about in the previous chapter we have to load the image and normalize it after that in order to generate the codebook with the k-means method this method is a predefined method

We have to choose at the beginning the random parameters as follows:

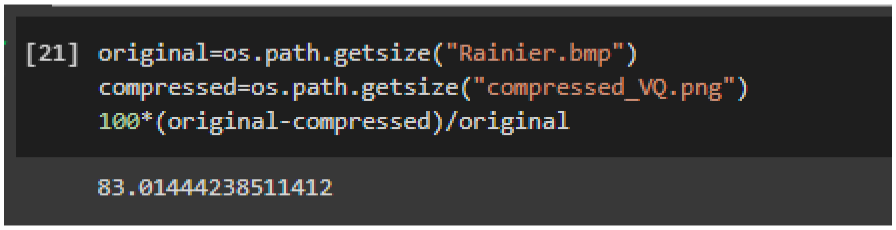
1- n_cluster=64 and n_init= 10 the compression rate is as follows:



```
[24] original=os.path.getsize("Rainier.bmp")
     compressed=os.path.getsize("compressed_VQ.png")
     100*(original-compressed)/original

     75.96439331320104
```

Figure 26 Compression rate n_cluster=64, n_init=10

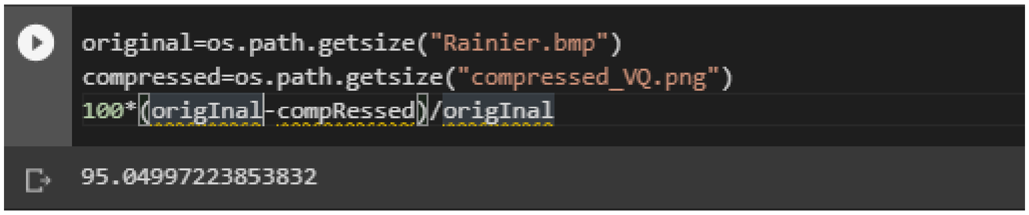
2- And with n_cluster= 128 and n_init =10 we have the following rate:



```
[21] original=os.path.getsize("Rainier.bmp")
     compressed=os.path.getsize("compressed_VQ.png")
     100*(original-compressed)/original

     83.01444238511412
```

Figure 27 Compression rate n_cluster=128, n_init=10

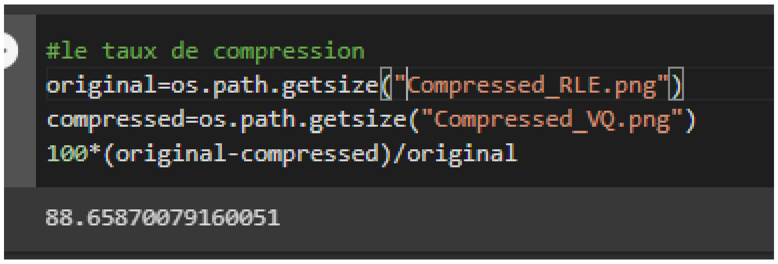
3- 3-We tried several parameters until the following parameters n_cluster=5, n_init=5 with the rate:



```
original=os.path.getsize("Rainier.bmp")
compressed=os.path.getsize("compressed_VQ.png")
100*(origInal-compRessed)/origInal

95.04997223853832
```

Figure 28 Compression rate n_cluster=5, n_init=5

4- We tried several parameters until the following parameters n_cluster=64, n_init=5 with the rate:



```
#le taux de compression
original=os.path.getsize("Compressed_RLE.png")
compressed=os.path.getsize("Compressed_VQ.png")
100*(original-compressed)/original

88.65870079160051
```

Figure 29 Compression rate n_cluster=64, n_init=5

This is the best parameter since the results I succeeded, the rate is a little less than the others but the quality and better than the previous results

In this part we have combined our two algorithms as follows:

➢ <u>Combination Run Length Encoding With Vector Quantization</u>

We started with RLE combined with VQ we got the following results:

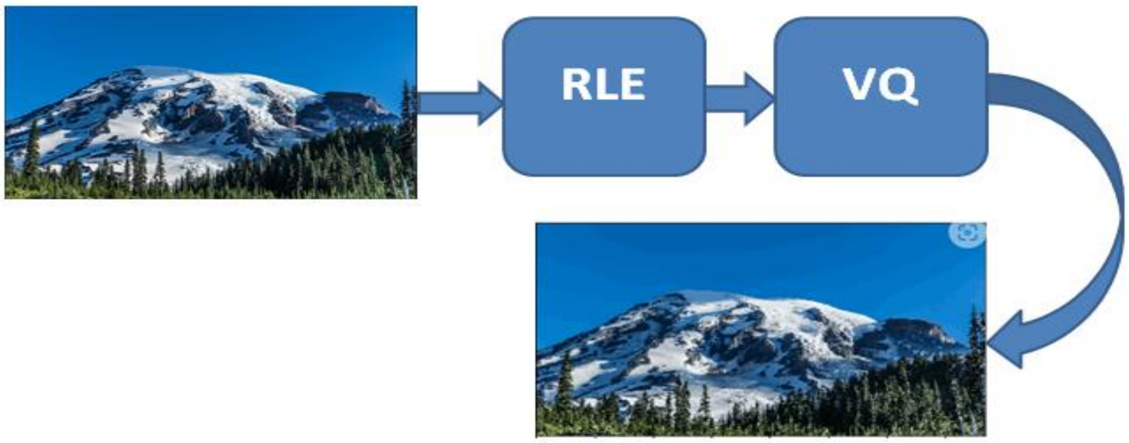
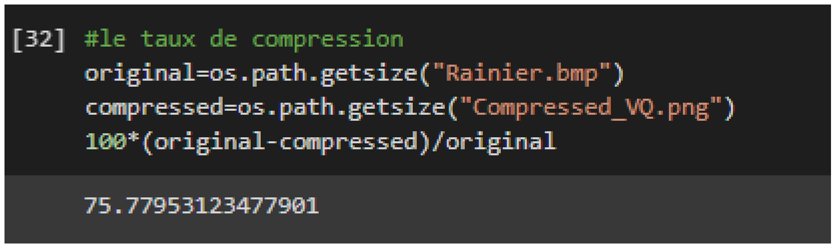


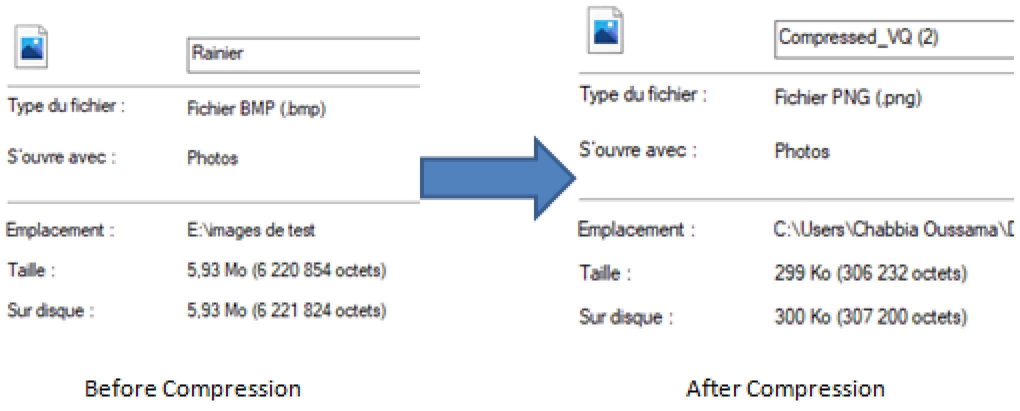
Figure 30 Image input and output from RLE_VQ

With a compression ratio:



```
[32] #le taux de compression
     original=os.path.getsize("Rainier.bmp")
     compressed=os.path.getsize("Compressed_VQ.png")
     100*(original-compressed)/original

     75.77953123477901
```

Figure 31 Compression rate RLE_VQ



	Rainier
Type du fichier :	Fichier BMP (.bmp)
S'ouvre avec :	Photos
Emplacement :	E:\images de test
Taille :	5,93 Mo (6 220 854 octets)
Sur disque :	5,93 Mo (6 221 824 octets)

Before Compression

	Compressed_VQ (2)
Type du fichier :	Fichier PNG (.png)
S'ouvre avec :	Photos
Emplacement :	C:\Users\Chabbia Oussama\D
Taille :	299 Ko (306 232 octets)
Sur disque :	300 Ko (307 200 octets)

After Compression

Figure 32 Size before and after compression RLE_VQ

We have concluded that to keep the image quality a bit lower the compression rate

➢ <u>Combination Vector Quantization With Run Length Encoding</u>

With VQ combined with RLE we got the following results:

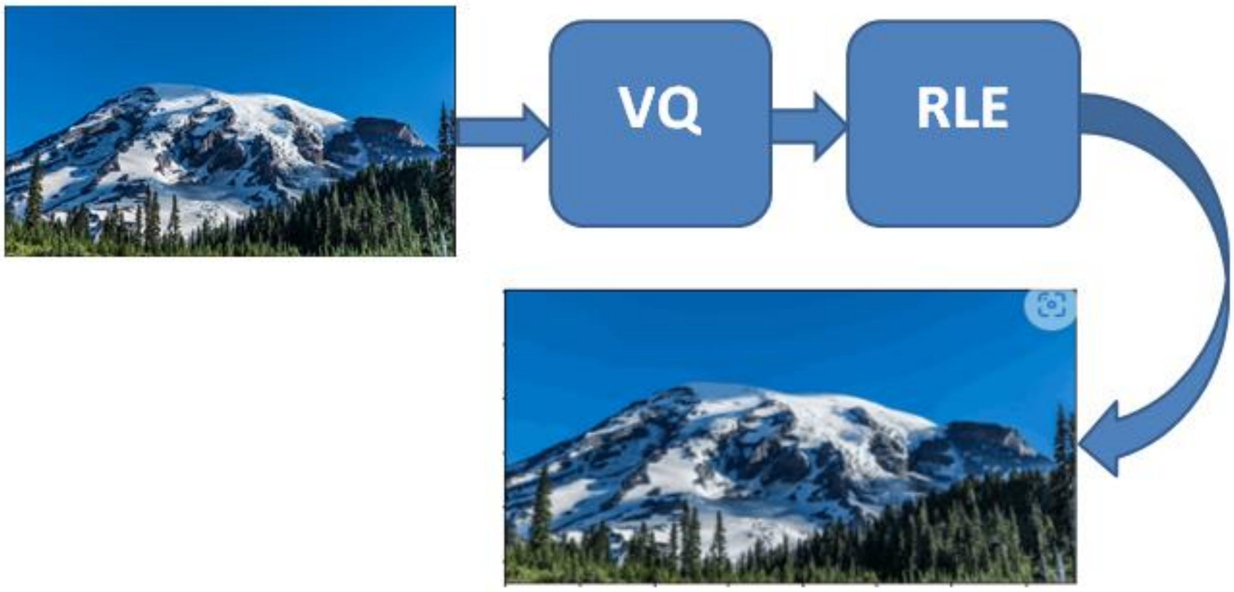
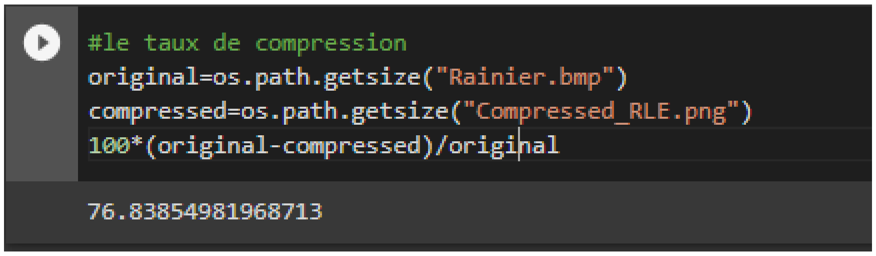


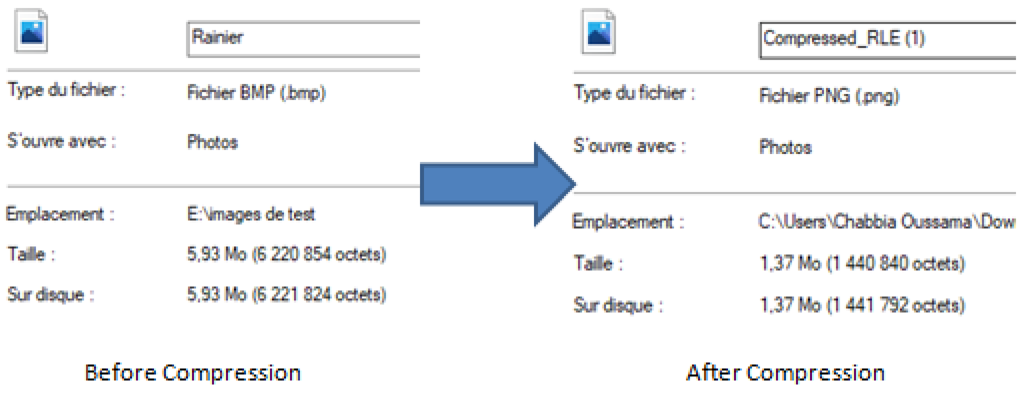
Figure 33 Image input and output from VQ_RLE

With a compression ratio:



```
#le taux de compression
original=os.path.getsize("Rainier.bmp")
compressed=os.path.getsize("Compressed_RLE.png")
100*(original-compressed)/original

76.83854981968713
```

Figure 34 Compression rate VQ_RLE



	Rainier
Type du fichier :	Fichier BMP (.bmp)
S'ouvre avec :	Photos
Emplacement :	E:\images de test
Taille :	5,93 Mo (6 220 854 octets)
Sur disque :	5,93 Mo (6 221 824 octets)

Before Compression

	Compressed_RLE (1)
Type du fichier :	Fichier PNG (.png)
S'ouvre avec :	Photos
Emplacement :	C:\Users\Chabbia Oussama\Dow
Taille :	1,37 Mo (1 440 840 octets)
Sur disque :	1,37 Mo (1 441 792 octets)

After Compression

Figure 35 Size before and after compression VQ_RLE

So we noticed that in the 1st algorithm that the vector quantization eliminates almost all pixel redundancy, so the RLE either causes the size to increase or change nothing.

➢ **Method CAE**

In this part we import cifar10 dataset and normalize it after that we create the CAE model this model is composed of three layers then we train it with the parameters set as follows:

- Epochs = 50
- Batch_size = 32
- Validation_split = 0.05

After the training we have the value of loss and accuracy as displayed in this graph

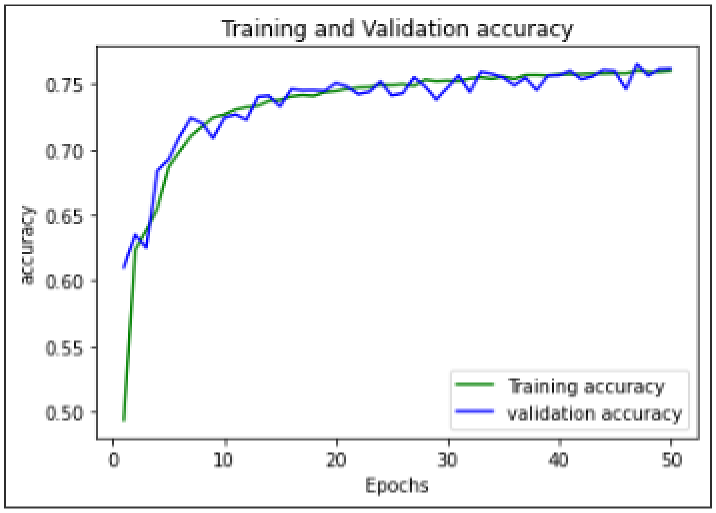


Figure 36 Training and validation accuracy

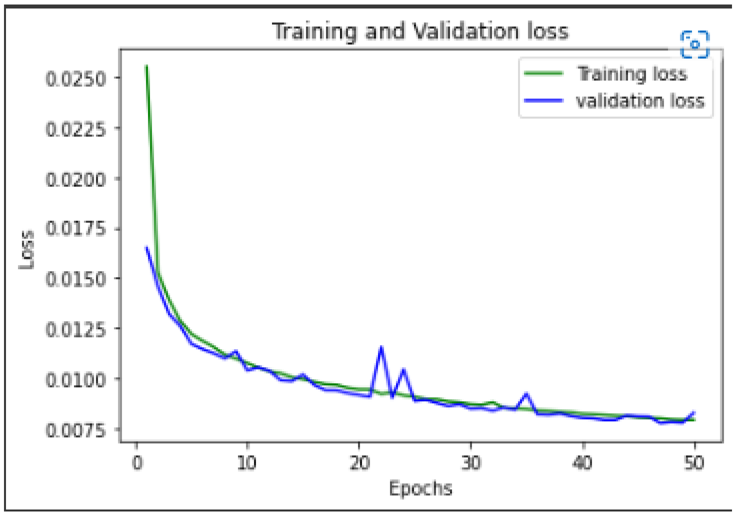
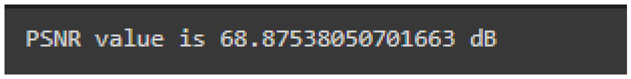


Figure 37 Training and validation loss

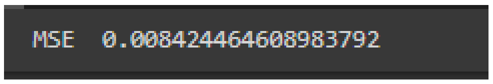
At the end of the training we choose a random image from image tests and compress and display it, and finally we calculate the MSE and PSNR and save the model

The PSNR and MSE value as fallow:



```
PSNR value is 68.87538050701663 dB
```

Figure 38 PSNR value



```
MSE  0.008424464608983792
```

Figure 39 MSE value

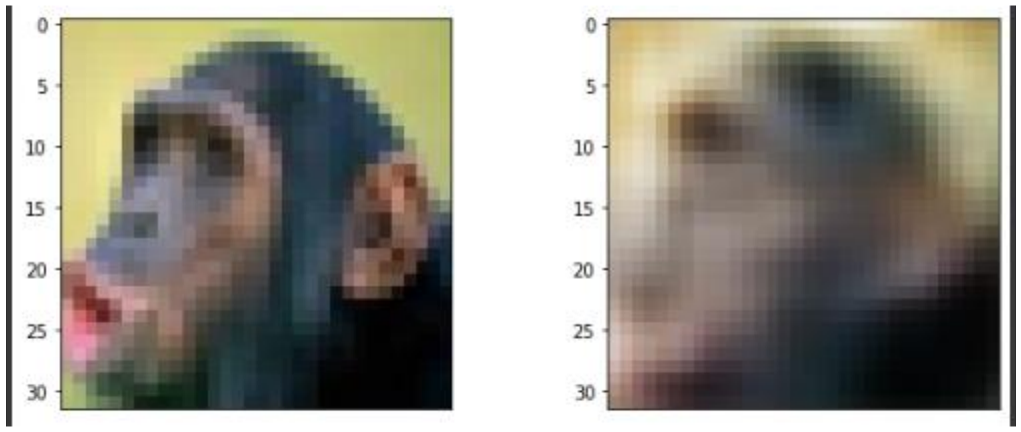


Figure 40 Original and Compressed image from tests dataset

➢ <u>Method Of Resize</u>

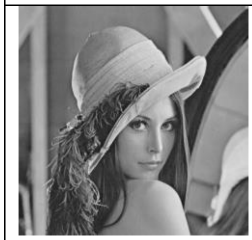
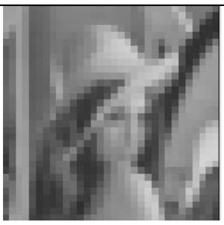
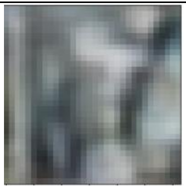
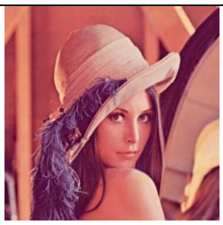
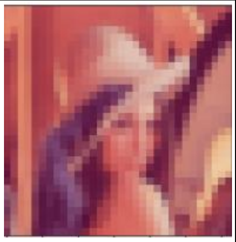
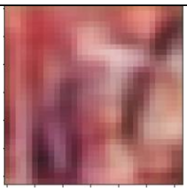
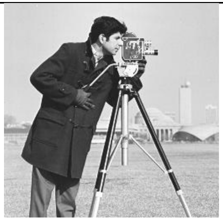
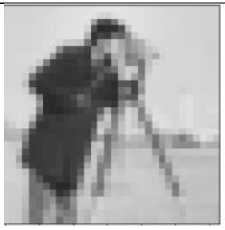
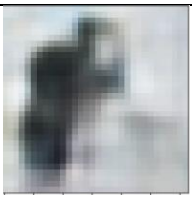
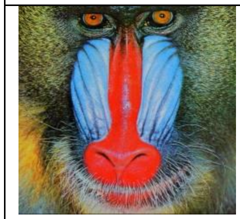
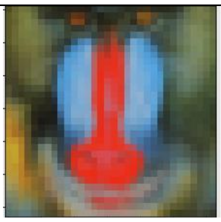
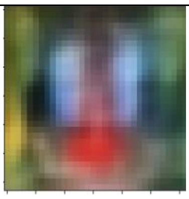
in this part we talk about the method we chose to resize the image for the model our model and fix with shape of input as following (32, 32, 3), so we have to resize, well we chose the resize function which is predefined in the scikit-image library and we have to apply it as follows: from skimage.transform import resize

➢ <u>Combination Of All Algorithms</u>

In this part we will talk about our compression system in general, as we said before we will combine the algorithms in the order that gives us the best result.
Our system has parts as:
Part of model creation and training after saving it then we will combine RLE with VQ then we import the model
So the structure is RLE +VQ +CAE

➢ <u>Results</u>:

Therefore, we tested our system with several different images, we received the following values:

Original Image	**Size**	**Resized image**	**Compressed image**	**Size**	**MSE**	**PSNR**
 Lena	257 Ko	 Resized lena	 Compressed Lena	1,24 Ko	0.0057072584	70.56652828046711 dB
 Lena color	768 Ko	 Resized Lena color	 Compressed Lena color	1,25 Ko	0.0062905387	70.14392522202031 dB
 Cameraman	65,0 Ko	 Resized cameraman	 Compressed cameraman	1,23 Ko	0.0068281987	69.78774211894799 dB
 Baboon	703 Ko	 Resized baboon	 Compressed baboon	1,2 Ko	0.011085306	67.68332663411067 dB

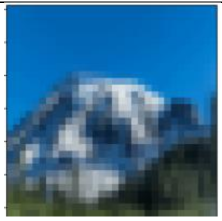
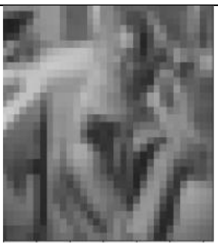
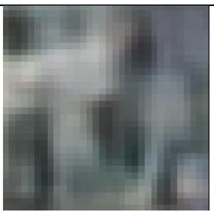
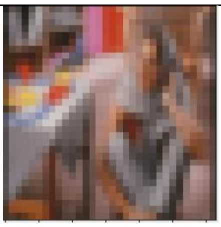
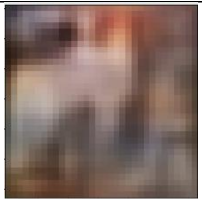
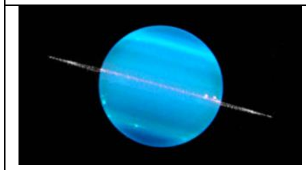
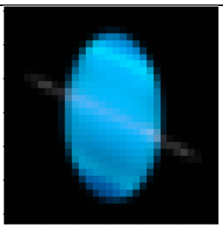
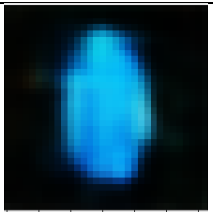
Rainier	5,93 Mo	 Resized rainier	Compressed rainier	1.2 Ko	0.005684556	70.58383813679625 dB
Barbara	257 Ko	 Resized Barbara	 Compressed Barbara	1.1 Ko	0.006261245	70.16419671173621 dB
Barbara color	1,18 Mo	 Resized Barbara color	 Compressed Barbara color	1.24 Ko	0.008950629	68.61226791522711 dB
 Uranus	5,93 Mo	 Resized Uranus	 Compressed Uranus	1.23 Ko	0.0036120184	72.55330404095177 dB

Table 1 Results of our system

➢ **<u>comparisons between the MSEs of the related works and the MSEs of the proposed technique:</u>**

Image	Related Works MSE	Proposed Technique MSE
	CAE Of (Zhengxue)	
Kodim 07 [20]	0.295	0.0039745937
Kodim 23 [20]	0.294	0.010007425
Kodim 15 [20]	0.293	0.0067679416

Table 2 Comparisons between our technique and other

- <u>**comparisons between the PSNRs of the related works and the PSNRs of the proposed technique:**</u>

Image	**Mu Li system PSNR**	**Our system PSNR**
Kodim06 [20]	26.48	75.9978539864428
Kodim01 [20]	25.34	73.29377497535388
Kodim08 [20]	25.01	69.93368207596095
Kodim02 [20]	30.27	74.68240690502307

Kodim18 [20]	25.35	73.23427825582152

Table 3 comparisons between our PSNRs and others

II. <u>Conclusion</u>

In this chapter we have seen the different stages of execution of our compression system based on the use of machine learning and deep neural networks.

The quality of the reconstructed images remains quite acceptable and in many cases exceeds the quality offered by the generic algorithm.

General conclusion

The volume of data produced is outpacing the storage capability. As a result, we must compress the data before using it (Acquisition of scientific, medical, industrial data, telephony, heritage conservation, etc.). As a result, the field has a long future ahead of it. Every year, new significant algorithms emerge.

Any compression attempts to eliminate duplication by using an alternative structure that is reversible, allowing the original to be restored (lossless compression), or by eliminating a portion of data that is considered irrelevant or unimportant (methods with losses, irreversible). Lossless methods have a substantially lower compression ratio than irreversible approaches! Of course, there are situations when losing information is unavoidable. However, there are many other criteria for evaluating compression algorithms.

RLE compression is a lossless compression method based on the simple idea that every image has redundancies and similarities. As a result, this compression method entails recognizing recurrences at various scales, with the goal of reducing image redundancy.

VQ compression is a lossy compression method based on the following simple principle: when the values of a multidimensional vector space are encoded or replaced by a key with values of a discrete subspace of smaller dimension, the smaller space vector requires less storage space, and thus the data is compressed. As a result, this compression method entails employing a dictionary to replace pixels that have already been compressed (Codebook).

These methods have been increasingly popular in recent years due to their close resemblance to the original image.

Deep learning algorithms, on the other hand, are a subset of machine learning algorithms that can find solutions to problems on their own by recognizing models in databases. To put it another way, machine learning enables computer systems to recognize models based on current algorithms and datasets and to build relevant solution concepts. These algorithms, among other things, can be applied to any sort of problem (or almost).

All of these considerations have given us the confidence to delve a little deeper into this sector and present a compression system based on machine learning approaches combined with an auto-encoder to reduce compression time while maintaining the highest possible image quality.

Our work has allowed us to integrate the concept of Deep Learning in order to optimize the compression of images, indeed according to the results obtained we motivated to improve our system after our problems, as the lack of materials for training has a larger Dataset and contains more characteristics

References:

1. [1]- SCE ZEROUAL DJAZIA, implémentation d'un environnement parallèle pour la compression d'image à l'aide des fractales, 2006
2. [2]- Digital Images | Encyclopedia.com
   Address link: https://www.encyclopedia.com/computing/news-wires-white-papers-and-books/digital-images
3. [3]- Microsoft PowerPoint - Ch_10_1 VQ Description (binghamton.edu)
   Address link:
   http://www.ws.binghamton.edu/fowler/fowler%20personal%20page/EE523_files/Ch_10_1%20VQ%20Description%20(PPT).pdf
4. [4]- Data Science [on line] Qu'est-ce que l'apprentissage automatique ? Une définition — Apprentissage Automatique — DATA SCIENCE
   Address link: https://datascience.eu/fr/apprentissage-automatique/quest-ce-que-lapprentissage-automatique-une-definition/
5. [5]- research gate [on line]
   AN_INTRODUCTION_TO_ARTIFICIAL_NEURAL_NETWORK
   Address link: https://www.researchgate.net/publication/319903816
6. [6]- Deep Learning ou Apprentissage Profond : qu'est-ce que c'est ? (datascientest.com)
   Address link : https://datascientest.com/deep-learning-definition
7. [7]- Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167, 2015.
8. [8]- Palash Goyal, Sumit Pandey, and Karan Jain. Introduction to natural language processing and deep learning. In Deep Learning for Natural Language Processing, pages 1–74. Springer, 2018. doi: 10.1007/978-1-4842-3685-7-
9. [9]- Cheng, Z., Sun, H., Takeuchi, M., & Katto, J. (2018, June). Deep convolutional autoencoder-based lossy image compression. In *2018 Picture Coding Symposium (PCS)* (pp. 253-257). IEEE.

10. [10]- Liu, H., Chen, T., Shen, Q., Yue, T., & Ma, Z. (2018). Deep image compression via end-to-end learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 2575-2578)
11. [11]- *Leonardo Galteri*, *Lorenzo Seidenari*, *Marco Bertini*, *Alberto Del Bimbo*; Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017, pp. 4826-4835
12. [12]- Eirikur Agustsson, Michael Tschannen, Fabian Mentzer, Radu Timofte, Luc Van Gool; Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2019, pp. 221-231
13. [13]- Agustsson, E., Tschannen, M., Mentzer, F., Timofte, R., & Gool, L. V. (2019). Generative adversarial networks for extreme learned image compression. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 221-231)
14. [14]- Mu Li, Wangmeng Zuo, Shuhang Gu, Debin Zhao, David Zhang; Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 3214-3223
15. [15]- What is Python? Executive Summary, https://www.python.org/doc/essays/blurb/
16. [16]- What is TensorFlow? How it Works? Introduction & Architecture (guru99.com)
    a. Page address: https://www.guru99.com/what-is-tensorflow.html
17. [17]- Keras: the Python deep learning API
    a. https://keras.io/
18. [18]- Welcome To Colaboratory - Colaboratory (google.com)
    a. https://colab.research.google.com/notebooks/welcome.ipynb
19. [19]- CIFAR-10 and CIFAR-100 datasets (toronto.edu)
    a. https://www.cs.toronto.edu/~kriz/cifar.html
20. [20]- True Color Kodak Images (r0k.us)
    a. http://r0k.us/graphics/kodak/index.html