Universidade de Aveiro
Ano 2022

Raúl Kevin do Espírito Santo Viana

Deep learning architecture for fast intra-mode CUs partitioning in VVC

Arquitetura de aprendizagem profunda para particionamento rápido de CUs no modo intra no VVC

Universidade de Aveiro
Ano 2022

Raúl Kevin Do Espírito Santo Viana

Deep learning architecture for fast intra-mode CUs partitioning in VVC

Arquitetura de aprendizagem profunda para particionamento rápido de CUs no modo intra no VVC

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Eletrónica e Telecomunicações, realizada sob a orientação científica do Professor Doutor António Navarro do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, com co-orientação do Professor Pedro Assunção do Departamento de Engenharia Eletrotécnica do Instituto Politécnico de Leiria.

Este trabalho teve o financiamento da FCT-Programa Operacional Regional do Centro, fazendo parte do projeto ARoundvision CENTRO-01-0145-FEDER-030652, Instituto de Telecomunicações (IT).

o júri / the jury

presidente / president	**Prof. Doutor Pétia Georgieva Georgieva** Professora Associada do Dep. de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.
vogais / examiners committee	**Prof. Doutor Luís Alberto da Silva Cruz** Professor Auxiliar do Dep. de Eng. Eletrotécnica da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.
	Prof. Doutor António José Nunes Navarro Rodrigues Professor Auxiliar do Dep. de Electrónica, Telecomunicações e Informática da Universidade de Aveiro (Orientador)

agradecimentos / acknowlegments

Graças ao contributo, que agradeço, do Instituto de Telecomunicações, Fundação para Ciência e Tecnologia, Universidade de Aveiro e do DETI, esta investigação foi realizada com sucesso. Sem a infraestrutura e o conhecimento disponibilizado por estas instituições, não seria possível a execução desta dissertação. Por isso, este documento não é a expressão de um trabalho individual, mas sim coletivo. Colegas, professores e os orientadores ajudaram na realização deste trabalho.
Um especial agradecimento ao José Filipe pelas intervenções e comentários que contribuíram para o enriquecimento deste trabalho.
Finalmente, mas de certeza não menos importante, agradeço a minha família. Obrigado, Maria Eduarda. Obrigado, Carla. Obrigado, Eduardo. Obrigado, Tatiana. Obrigado a todos meus amigos. Sem o vosso apoio este trajeto teria sido mais difícil.

Thanks to the contribution, which I am grateful for, from Instituto de Telecomunicações, Fundação para Ciência e Tecnologia, Universidade de Aveiro and DETI, this investigation was successfully carried out. Without the infrastructure and knowledge provided by these institutions, it would be impossible to execute this dissertation. Therefore, this text does not represent an individual's labor, but rather a community effort. Colleagues, professors and the supervisors helped me in the realization of this work.
Special thanks to the José Filipe for his contributions and insightful remarks that enriched this work.
Last, but certainly not least, I would want to thank my family. Thank you, Maria Eduarda. Thank you, Carla. Thank you, Eduardo. Thank you, Tatiana. Thanks to all my friends. This journey would have been more difficult without their help.

palavras-chave

Aprendizagem Automática, Redes Neurais Convolucionais, Aprendizagem Profunda, Codificação de vídeo, VVC, QTMT, Codificação Intra

resumo

O surgimento de novas tecnologias que proporcionam experiências audiovisuais criativas, como filmes em 360º, realidade virtual, realidade aumentada, 4K, 8K UHD e 16K, demonstram a demanda por vídeo no mundo moderno. Por causa desta tensão, Versatile Video Coding (VVC) foi desenvolvido devido à necessidade de introdução de novos padrões de codificação. Apesar dos avanços alcançados com a introdução deste padrão, sua complexidade aumentou em comparação ao seu antecessor, High Efficiency Video Coding (HEVC). Isso deve-se à inclusão de novas ideias, como duas novas transformadas, 32 novos modos de previsão intra-angular e uma nova metodologia de partição de blocos. A nova técnica de particionamento é responsável pela maior parte do aumento no tempo de codificação. Esta duração estendida está associada à ao processo de otimização em termos de bito-distorção. Embora o VVC ofereça taxas de compressão mais altas, a sua complexidade é alta.

Tendo em conta a complexidade desta norma, esta dissertação analisa o Multi-Stage Exit Convolutional Neural Nework (MSE-CNN). Este modelo é baseado em *Deep Learning* e está disposto numa estrutura sequencial composta por diversos estágios cujo objetivo é simplificar o método de partição para o modo intra no VVC. Cada estágio, que representa uma específica profundidade de partição, contém uma variedade de camadas para extrair características de uma *Coding Tree Unit* (CTU) e tomar uma decisão em como realizar a *partição* desta. O MSE-CNN reduz a complexidade através da simplificação do processo de partição. Logo, com este modelo, ao invés do VVC recorrer a estratégias recursivas para encontrar a melhor forma de dividir uma imagem, este consegue prever a maneira mais adequada de o fazer. Neste trabalho é apresentado um modelo do MSE-CNN que segue estratégias diferentes em relação à implementação original do treino desta rede. Com as modificações feitas foi possível obter, utilizando um limite de seleção conservativo, uma perda de Y-PSNR de 0.65% e uma redução de complexidade de 41.49%. Para além destes resultados, foi estabelecido um conjunto de passos para tratar o *dataset* utilizado, foi criado o *ground-truth* para treinar e validar o modelo, e foi feita uma interpretação do trabalho realizado pelos criadores originais do MSE-CNN.

keywords

Machine Learning, Convolutional Neural Networks, Deep Learning, Video Encoding, VVC, QTMT, Intra Coding

abstract

The emergence of new technologies that provide creative audiovisual experiences, such as 360-degree films, virtual reality, augmented reality, 4K, 8K UHD, and 16K, demonstrates the demand for video data in the modern world. Because of this tension, Versatile Video Coding (VVC) was developed because of the necessity for the introduction of new coding standards. Despite the advancements achieved with the introduction of this standard, its complexity has increased in comparison to its predecessor, High Efficiency Video Coding (HEVC). This is due to the inclusion of new ideas such as two new transforms, 32 new intra-angular prediction modes, and a new block partition methodology. The new partitioning technique is responsible for much of the increase in encoding time. This extended duration is linked with the optimization of the Rate-Distortion cost (RD cost). Although VVC offers higher compression rates, the complexity of its encoding is high.

In light of this, this dissertation examines the Multi-Stage Exit Convolutional Neural Network (MSE-CNN). This Deep Learning-based model is organised in stages in a sequential structure, with the objective of simplifying the partitioning scheme for intra mode VVC. Each stage, which represents a different partition depth, encompasses a set of layers for extracting features from a Coding Tree Unit (CTU) and deciding how to partition it. Instead of using recursive approaches to determine the optimal way to fragment an image, this model allows VVC to estimate the most appropriate way of doing it. This work presents a model of the MSE-CNN that employs training procedures distinct from the original implementation of this network. With the improvements made, it was possible to achieve an Y-PSNR loss of 0.65% and complexity reduction of 41.49%. In addition to these results, a pipeline to process the used dataset was established, the ground-thruth to train and validate the model was created, and an interpretation of the work done by the MSE-CNN's original creators was provided.

Contents

List of Figures

List of Tables

Glossary

Epoch An Epoch occurs when a whole dataset is processed by a neural network.

Iteration A step in which a single batch of training data from the actual dataset that goes through a neural network.

Mini-batch Number of training examples from the actual dataset that go through a neural network during an iteration.

One-hot Encoding Process in which a single integer is taken and turned into a vector with one element that is 1 and the rest are 0s.

Overfitting Condition in which a model memorises precise patterns and noise in the training data, but lacks the flexibility to make predictions on other sources of data.

Overlapping convolution Type of convolution in which two or more consecutive receptive fields share the same values.

Receptive field A segment of an input image used to calculate the activation of a neuron in a feature map.

Chapter 1

Introduction

1.1 Context and Motivation

20 years ago, the state of the technology associated with multimedia was completely different from what we have today. Humanity went from having mobile devices that could only be used to make calls or send SMS messages and having very slow and inconsistent video calls over the internet; to cell phones that can be used as video players, cameras, web browsers, email clients, navigation and social networking tools and very reliable video calls, allowing to connect instantly with anyone anytime.

Beyond the previous mentioned improvements, many other signs of progress exists. Better communication infrastructure, more advanced technology, and price reductions all contributed to where we are now.

The average number of hours viewed by users and the resolution of the videos have both increased significantly as a result of the widespread adoption of video consumption, making the bandwidth devoted to the presentation of video footage the largest among all applications. In 2023 it is expected that the number of devices connected to IP networks will surpass by three times the number of the global populations [1], this for sure will lead to an increase in video traffic. The previous fact plus the introduction of new technologies that offer innovative audiovisual experiences, such 360º films, virtual reality, augmented reality, 4K, 8K UHD, and 16K, only serves to emphasize the requirements of video needed today. To illustrate, if a 2-hour 4K video was filmed at a resolution of 4096 x 2160 (DCI 4K) and in the RGB colour space quantised with 8 bits, the required storage would be around 1.39 TBs. Since 59 GBs of data would be loaded every frame, consumer-level computers would not be able to handle this volume of raw data.

Prior to transmission or storage, a digital video's data requirements are reduced by the process of video compression, while maintaining a minimal loss of quality. This process is also known as video encoding. Before a video is displayed, the complementary procedure of decoding or decompressing recovers a digital video signal from a compressed representation. Spatial and temporal characteristics are leveraged in this procedure to take advantage of the many redundancies present in a video file. In the bellow image, many applications of this tool can be seen.

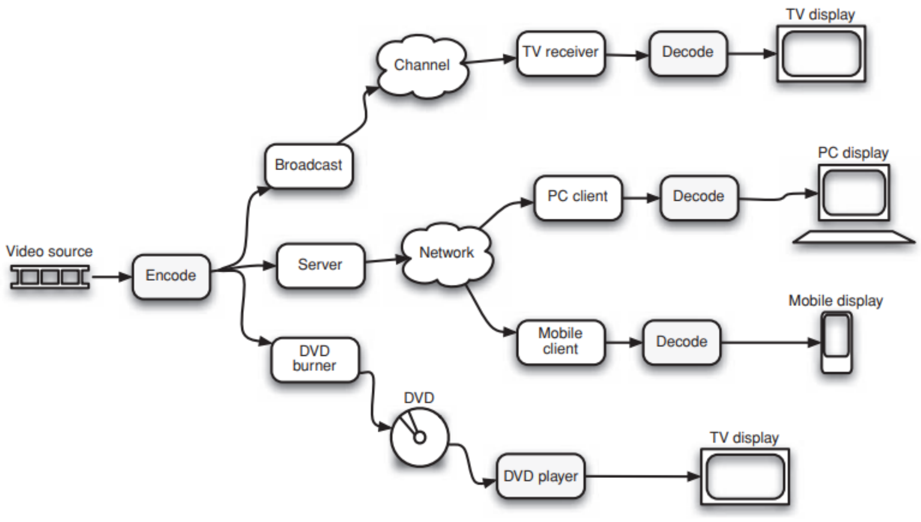


Figure 1.1: Video coding scenarios [2]

Even if it is now simple to execute algorithms quickly thanks to advancements in computing power, the continuous growth of video resolutions and demand for video technologies do not stop. Thus the complexity of the encoding process cannot be dealt only based on faster processors. Large resolution and high storage needs increase the processing time of encoding greatly since there is more data to be analysed. Problems would still arise if one just relied on faster processing units. For mobile devices with small batteries, such as smartphones or laptops, the usage of encoding techniques would still be unsuitable due to excessive power consumption.

Some hardware or software solutions can be used with these extremely complicated algorithms to expand the reach of this cutting-edge video technology without compromising its usability. Regarding the software strategy, they all aim to optimise a certain step in the encoding process. For example, heuristics is a technique that produces better results by making a decision based on a set of expected outcomes. Therefore, many of the building pieces of these algorithms may be improved by incorporating this method. An additional method is to utilise statistics to find the most important patterns to reduce complexity. This is done by creating models that are less complicated by computing several indicators from a vast amount of research data. The new most trendy approach to solve many of nowadays's problems, ranging from classification to regression, is by using Machine Learning. By applying these algorithms, it is possible to learn patterns that help reduce the number of computations needed to encode a video file. In addition to improving pure software compression and decompression methods, there has been a strong emphasis on creating dedicated hardware for such encoding tools. Since such co-processors often display a superior performance than generic CPUs, this strategy might resolve many of the issues that were previously raised.

There are two main types of machine learning paradigms: supervised and unsupervised. Computers can learn from data using both strategies. Unsupervised learning do not use labelled data to improve its parameters, in contrast to supervised learning, which does. Supervised learning should be an alternative when there are limited but well-labelled data. Unsupervised learning would frequently work and yield better results for large unlabeled datasets [3]. However, the best approach is still goal dependent. The huge volume of video data makes it obvious that video encoding will greatly benefit from this field.

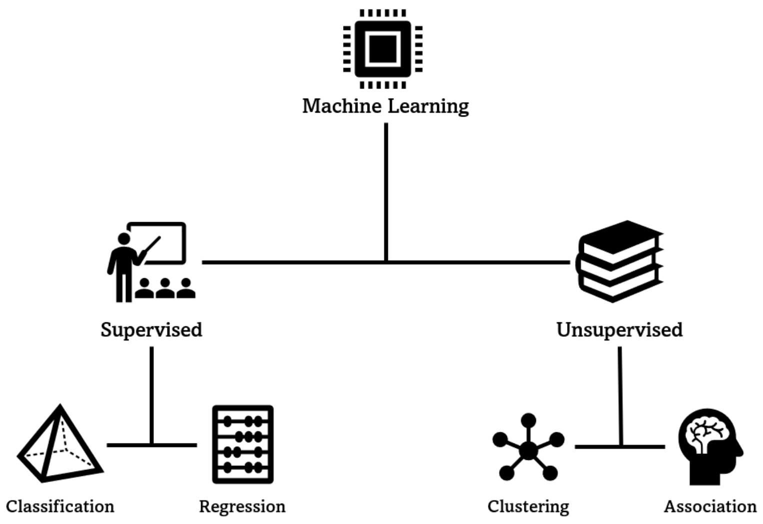


Figure 1.2: Machine Learning applications

Several video compression algorithms have been employed during the past ten years before being quickly superseded by newer versions that provide more compression advantages with less quality loss, such as the replacement of HEVC by VVC. It's crucial to keep in mind that although compression rates are increasing, complexity is also rising at a similar rate.

1.2 Scope and Learning-based Approach

In this section, the purpose of this dissertation will be discussed and also the steps to solve the problem in hand.

1.2.1 Versatile Video Coding (VVC)

Versatile Video Coding (VVC), also referred to as H.266, is a video compression standard that was put into place on July 6, 2020 by the Joint Video Experts Team (JVET), a collaboration between the VCEG working group and another working group from MPEG. High Efficiency Video Coding (HEVC), often known as H.265, was replaced by it. It was created with two main objectives in mind: better compression performance and support for a huge variety of applications [4].

VVC

Figure 1.3: VVC logo [5]

This novel standard presents a significant improvement compared to its predecessor, HEVC, having a compression rate 44.4% higher [6], while maintaining the same quality. This encoder not only offers compression advantages but also enables a variety of resolutions and settings, living up to the term "versatile" that appears in its name. The VVC Test Model (VTM), a reference software codebase with a basic set of coding tools, has been used in the development of VVC. Despite the advances made with the emergence of this standard, with it also came an increase in complexity that reaches 2000% when compared to HEVC [7]. This is due to the introduction of new concepts, including two new transforms, 32 new intra-angular prediction modes and a new block partition scheme. The partitioning strategy introduced four new structures. Although it is now possible to break apart a frame with more detail due to this update, this requires an exhaustive search in order to find the optimal way of organizing the data. This new scheme is responsible for the majority of the increase in encoding process time, accounting for approximately 97% of the total time [8].

In light of all mentioned in this section, this thesis proposes a potential strategy for simplifying this standard, more specifically its block partitioning procedure. The speeding up of the compression and decompression of VVC, enables it to have a wider range of use cases, such as real-time applications.

1.2.2 Machine Learning Approach

The machine learning field has been offering an increasing number of answers to various challenges over the past ten years. Many regression and classification problems, as well as anomaly detection and clustering problems, started to be solved. In order to lessen the complexity issue burden of the VVC standard, it was chosen to leverage the many tools that this area offers while also taking into account the present state-of-the-art. For this, the Multi-Stage Exit Convolutional Neural Network (MSE-CNN) was studied [10]. This network uses Deep Learning concepts (ConvNets, LSTMs and ResNets) to simplify the partitioning process. Bearing in mind what was mentioned above, the strategy to simplify the decision process required to achieve optimal partitioning involves:

- firstly, coding, processing and structuring a dataset with diversified and sufficient information;
- then implement a set of neural networks that to be trained in a supervised manner, using the data mentioned above;
- after obtaining a collection of models that went through validation and the fine-tuning process, choose the one that yields the best results;
- finally, the neural network should be merged with the VVC Test Model (VTM), in order to verify the performance of the standard with this modification.

In addition to these steps, it is necessary to internalize a number of concepts and technologies in order to implement the algorithm that will be covered later in this document.

1.3 Outline

In the preceding sections, the problem and a learning-based solution for this dissertation were introduced. The remaining of this document is divided in 6 main chapters.

In chapter 2, an explanation of the theoretical foundations necessary for a better understanding of the developed solution will be made, concepts from codecs to ML will be elaborated.

In the following chapter, the database, composed of pictures and videos, will be described. Topics such as its acquisition, the generation of labels through it and its processing to create appropriate data structures used for the development of a model will be addressed.

The procedure to implement the model and how the training was done will be described in chapter 4.

In chapter 5, the results obtained from the model will be presented, along with an discussion.

Finally, summary and conclusions of what was done in this work is presented in the last chapter, in addition to some suggestions for improving the results as well as the main contributions of this work.

References

[1] "Cisco Annual Internet Report - Cisco Annual Internet Report (2018–2023) White Paper," Cisco, Mar. 09, 2020. https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html (accessed Oct. 05, 2022).

[2] I. E. Richardson, The H. 264 Advanced Video Compression Standard, 2nd ed. 2010, p. 3.

[3] I. H. Sarker, "Machine Learning: Algorithms, Real-World Applications and Research Directions," SN Computer Science, vol. 2, no. 3, pp. 1–21, Mar. 2021, doi: 10.1007/s42979-021-00592-x.

[4] B. Bross et al., "Overview of the Versatile Video Coding (VVC) Standard and its Applications," IEEE Transactions on Circuits and Systems for Video Technology, vol. 31, no. 10, pp. 3736–3764, Oct. 2021, doi: 10.1109/tcsvt.2021.3101953.

[5] "H.266:Versatile video coding," H.266:Versatile video coding. https://www.itu.int/rec/T-REC-H.266 (accessed Oct. 05, 2022).

[6] Í. Siqueira, G. Correa and M. Grellert, "Rate-Distortion and Complexity Comparison of HEVC and VVC Video Encoders," 2020 IEEE 11th Latin American Symposium on Circuits & Systems (LASCAS), 2020, pp. 1-4, doi: 10.1109/LASCAS45839.2020.9069036.

[7] F. Bossen, X. Li, A. Norkin, K. Sühring, "JVET AHG report: Test model software development (AHG3)," Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11 document JVET-O0003, 15th JVET meeting, Gothenburg, Sweden, July 2019.

[8] A. Tissier, A. Mercat, T. Amestoy, W. Hamidouche, J. Vanne and D. Menard, "Complexity Reduction Opportunities in the Future VVC Intra Encoder," 2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP), 2019, pp. 1-6, doi: 10.1109/MMSP.2019.8901754.

[9] I. E. Richardson, The H. 264 Advanced Video Compression Standard, 2nd ed. 2010

[10] Li, T., Xu, M., Tang, R., Chen, Y., & Xing, Q. (2021). DeepQTMT: A deep learning approach for fast QTMT-based CU partition of intra-mode VVC. IEEE Transactions on Image Processing, 30, 5377-5390.

Chapter 2

Fundamental concepts

The two primary subjects of this work, Video Codecs (VVC, specifically) and Machine Learning, will be introduced in this chapter. Colour spaces, video formats, compression quality, prediction models, spatial models, entropy encoders, and VVC will all be covered regarding codecs. As for ML, points related to this thesis will be addressed in more detail: Artificial Neural Networks, Convolutional Neural Networks, Recurrent Neural Networks, Residual Networks, metrics used in ML and the current applications of this field. The description of the theoretical ideas behind the domains that make up this thesis not only assists the readers in better grasping what was done, but also reflects the research carried out to develop this work.

2.1 Video formats

Recording a video involves obtaining all the colours or contrasts that compose a visual environment through a time interval. This is achieved by numerical representation that facilitates representing a scene's colour or contrast and storing its description in a specific digital format. For example, to describe colour three elements are typically used to illustrate an entire colour space. Although one component is enough to represent the different shades of the image, such as a monochromatic frame. Regarding storing the video, the most popular formats are the 'intermediate', Standard Definition (ITU-R 601) and High Definition. However, the right choice for a format will not just depend on the medium capacity but also on the end user specifications.

2.1.1 Colour Spaces

In the world of computing, a colour space is required in order to represent colours. This space indicates different characteristics that a given pixel must have, such as brightness and colour. Monochromatic images do not require a parameter to specify the colour because it can be described just by its brightness values. Since not all devices represent colours in the same way and different spaces provide particular advantages, there are different types of colour spaces (RGB, HSV, YCbCr, CIELAB, among others). The conversion between different spaces can result in loss of information, thus in image quality.

RGB

RGB is one of the main colour spaces that are everywhere in our day-to-day devices, from our TVs to our smartphones. Composed of red, green, and blue, by varying the proportions of these three, it is possible to create any colour. This space is very appropriate for capturing and displaying images. For a frame to be captured, each individual colour from a scenery has to be filtered out. This is achieved through the use of sensor arrays that absorb specific wavelengths. Regarding the displaying, an RGB image has each pixel component (red, green and blue) illuminating with different intensities. The result of this, from a normal viewing distance, is the merging of the components into a single colour. Usually the values for each component are stored with 8 bits, but more can be used for higher resolution.

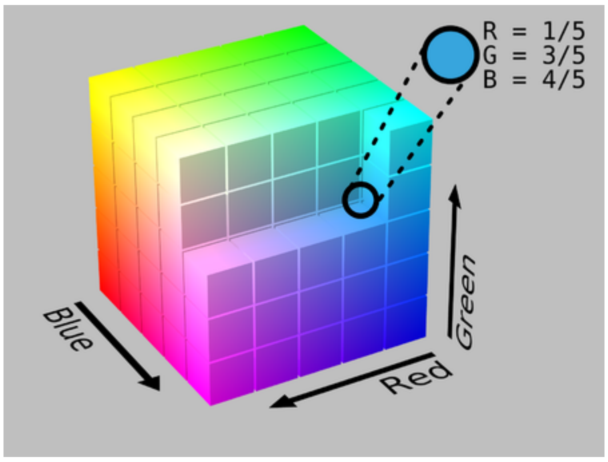


Figure 2.1: RGB cube [1]

YCbCr

Figure 2.2 shows a depiction of the elements that make up the YCbCr colour space which is commonly used in video codecs. The first image, starting from the top, is the original image, which includes all the components merged together. The second contains only the Y component, which consists of the contrast variation in the image. For this reason, this picture is in black and white. The last two images, contains the chroma blue (C_b) and chroma red (C_r) components. These two contain the colour information from the main image. It is relevant to mention that YCbCr is the digital version of YUV.

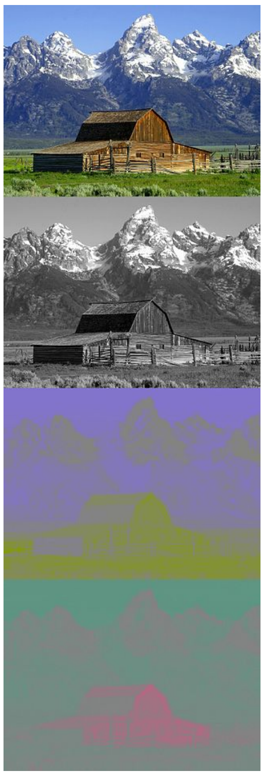


Figure 2.2: Pictures with different YCbCr components [2]

Unlike RGB, YCbCr is a system that takes human vision into account. Since the human visual system (HVS) has less problem distinguishing contrasts than colours, YCbCr takes advantage of this feature to save memory. As a result, more bits can be allocated to the luma weight rather than to C_b and C_r (chroma), without being visually noticeable.

This type of representation minimises the amount of storage space required to save an image. The parameters Y, C_b and C_r can be obtained directly from the RGB colour space from the following equation,

$$\begin{aligned} Y &= k_r * R + k_b * B + k_g * G \\ C_r &= R - Y \\ C_b &= B - Y \\ C_g &= G - Y \end{aligned} \tag{2.1}$$

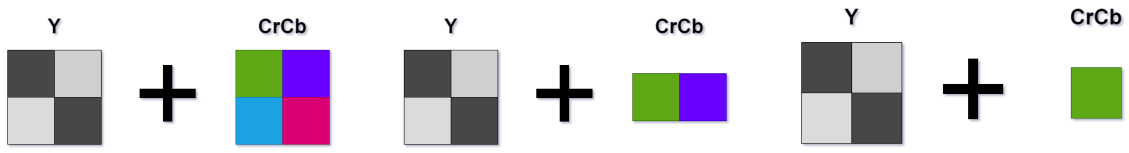
Y: Luma component

C_r, C_b, C_g: Chroma components

$k_r = 0.299$, $k_b = 0.587$, $k_g = 0.114$ [3]: Constants

The variables k_r, k_b, and k_g in the equation above are used to create a weighted average of the parameters R, G, and B [3]. Of the three chroma values, only two are used (Cr and Cb). This is because $C_r + C_g + C_b$ is constant, implying that the component C_g can be determined given the values of the other two and the luma.

There are 3 types of formats in YCrCb that propose different ways of grouping luma and chroma samples. The numbers in the format naming convention indicate the sampling rate of each component in the horizontal direction. In the format 4:4:4, for 4 samples of Y, there are 4 for Cr and Cb. A balance between chroma and luma is achieved in this configuration. For a 4:2:2 arrangement, there are two samples of each chroma component for every four samples of Y. Then there is the 4:2:0, which has a vertical and horizontal resolution that is half that of Y for both Cr and Cb. This means that for 4 luma values, only 1 value is used for the blue and the red chroma. 4:2:0 is widely used in various contexts, from digital television to DVDs. This fact is due to the storage space it occupies, requiring less memory than 4:4:4, 4:2:2 and RGB. It should be noted that this specific format does not follow the same naming logic as the previous two.



(a) 4:4:4 (b) 4:2:2 (c) 4:2:0

Figure 2.3: YCbCr types

2.1.2 Video resolutions

In order to transmit or store video, the proper format has to be chosen so that the needs for a specific application are met. The frame quality, available bandwidth, and the end user's specifications are all critical to deciding the most suitable format or resolution.

Before compression or transmission, it is common to use an 'Intermediate format' to capture or convert video footage. Common Intermediate Format (CIF) is part of a group of popular formats for these types of applications. In the bellow table, they are listed.
Regarding the use for each, in devices where the display resolution and bitrate are confined, QCIF or SQCIF are the fittest; CIF and QCIF are the most utilised in videoconferencing applications; for DVD-video and standard-definition television, 4CIF is the ideal one.

Format	**Luma resolution (horz. x vert.)**	**Bits/frame (4:2:0, 8 bits sample)**
Sub-QCIF (SQCIF)	128x96	147456
Quarter CIF (QCIF)	176x144	304128
CIF	352x288	1216512
4CIF	704x576	4866048

Table 2.1: Formats based on CIF [4]

Another format to be discussed is the Standard Definition, ITU-R Recommendation BT.601-5. This format is used for converting analogue television signals to digital and back and has been employed as an approach for most digital consumer video formats. It uses the 4:2:2 YCrCb with the luminance component being sampled at 13.5MHz and the chrominance at 6.75MHz. Depending on the frame rate, the specifications of the video will be different. Thus, the parameters used for the NTSC and PAL standards are distinct, since the former's rate is 30Hz and the latter's 25Hz. Despite this dissimilarity, both norms have the same total bitrate of 216Mbps, resulting from the lower frame rate of PAL being compensated by its higher spatial resolution. The active area is the true region shown on the screen, so it is smaller than a frame's actual resolution. One more characteristic of this format is that its samples use 8 bits for quantification. The values 0 and 255 are reserved for synchronisation, and the luma range is 16 to 235 (the former number corresponding to black and the latter to white).

	30Hz frame rate	**25Hz frame rate**
Field per second	60	50
Lines per complete frame	525	625
Luminance samples per line	858	864
Chrominance samples per line	429	432
Bits per sample	429	432
Total bitrate	216Mbps	216Mbps
Active lines per frame	480	576
Active samples per line (Y)	720	720
Active samples per line (Cr,Cb)	360	360

Table 2.2: Standard definition formats specifications [5]

The High Definition (HD) formats are widely used in applications associated with heightened quality video viewing, from films in the cinema to online streaming. In the following table, the most popular ones are presented.

Format	**Progressive or Interlaced**	**Horz. Pixels**	**Vert. Pixels**	**Frames or field per second**
720p	Progressive	1280	720	25 frames
1080i	Interlaced	1920	1080	50 fields
1080p	Progressive	1920	1080	25 frames

Table 2.3: High definition formats specifications [6]

Comparing SD formats with HD formats, it is possible to see a big difference in the amount of data being transmitted. A SD video has 10368000 displayed pixels per second, while a 1080p HD video has 51840000. The latter format takes five times more storage than its counterpart. For this reason, compressing algorithms must be used to make these formats more feasible.

Format	**Progressive or Interlaced**	**Horz. Pixels**	**Vert. Pixels**
4K	Progressive	3840	2160
8K	Progressive	7680	4320

Table 2.4: Ultra High Definition (UHD) formats [7] [8]

In Table 2.4, the resolutions of Ultra High Definition (UHD) are listed. The higher pixel density of these formats emphasises the need for more robust codecs. In today's world, these are the common used formats.

2.2 Video encoding techniques

According to the Shannon-Hartley Theorem, for a given communication channel that has a certain bandwidth and noise, there is a maximum value for the transmission rate on that same channel. Taking this theorem into account, if video files were sent without any kind of compression, they would take too long to reach the receiver. For this reason, the appearance of codecs (en**CO**der/**DEC**oder) came to facilitate transmission and storage of information. This is done through a sequence of encoding, transmitting/storing and decoding the data (1.1). Devices that capture, receive, store, or transfer video must thus have these codecs.

In order to compress the data, codecs exploit redundancies in the information, searching for spatial correlations in the same image and/or temporal correlations in image sequences. When it comes to spatial correlations, the algorithms search for similarities/redundancies between pixels that are close to each other. In any of the frames from the Figure 2.4, it is noticeable that certain sections of the characters' suits are similar in colour. On the other hand, in the temporal domain, the fact that successive frames have a lot of identical characteristics between each other is exploited. As an illustration, the background of the sequence of frames in the previously mentioned Figure remains unchanged.

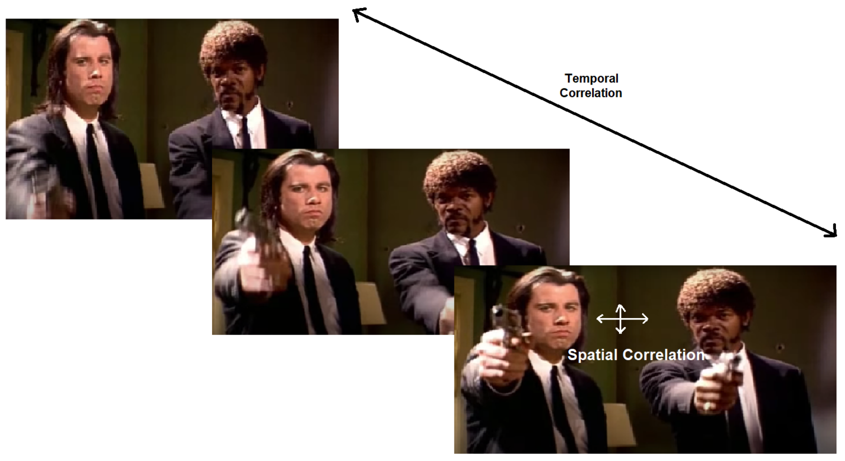


Figure 2.4: Redundancies in frames

The compression of an image can lead to its subsequent reconstruction being very different or not from the original file. Depending on which case it is, we will be dealing with a lossless or lossy codec. When compressing a video in a lossless way no information is lost. In other words, after decoding the resulting output is an exact copy of the input. However, in this type the encoding gains are modest. Contrarily, in lossy the compression ability is much higher than the previous type, but the quality loss of the reconstructed file is higher. Due to existing deterioration, these algorithms take into account HSV, thus allowing the quality reduction to be less noticeable.

In this section, the most relevant concepts of codecs, in the context of the solution developed in this thesis, will be mentioned. Namely the following topics: Codec Model, Prediction Models, Partitioning, Spatial Model and Entropy Encoding.

2.2.1 Codec Model

The goal of any codec is to describe a collection of picture sequences in a more simplified design. All of them have the same basic structure, which is composed of: a prediction model, a spatial model and an entropy encoder.

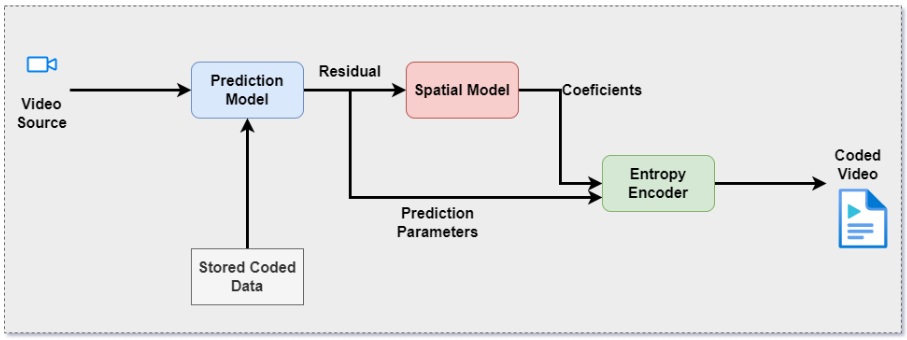


Figure 2.5: General Model of a Encoder

These algorithms accept one or more uncompressed images as input, which are then sent to the prediction model. By exploring similarities between frames and/or samples of the same picture, the encoder can predict a section or the entire image. If samples from the same frame are used, we are dealing with an intra prediction. Whereas in the case where sequences are used, the prediction is called inter or motion compensated. At the output of the prediction model, there is the residual frame/block, resulting from the subtraction between the forecast and the image/reference block, and also a set of parameters with information about the configurations of the executed process. Then this residual data is forwarded to the spatial model. Inside this block, the residue is prepared in order to reduce spatial redundancy. For this purpose, it is typically applied to data transforms and quantisation. The use of the transforms allows the conversion of samples to a new domain. Quantization makes it possible to remove insignificant values and characterize the residue more compactly. This stage outputs a set of quantised transform coefficients. In the last phase, the results of the prediction model and the spatial model are compressed by the entropy encoder. This encoder gathers all the information that has been manipulated, removes redundant content and generates a sequence of bits ready to be stored or transmitted.

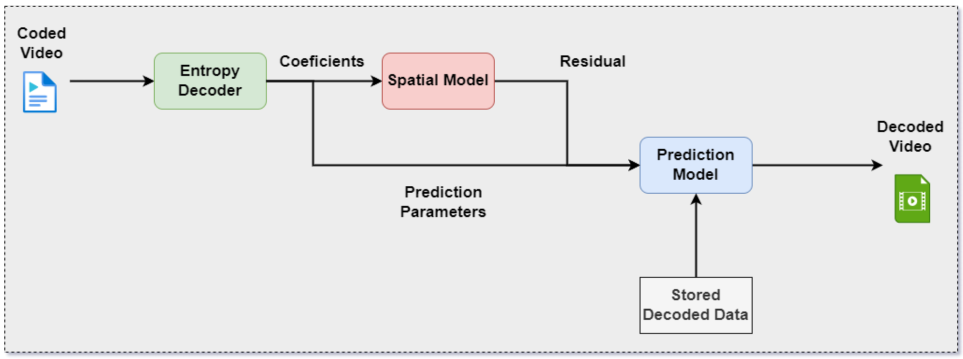


Figure 2.6: General Model of a Decoder

Since the decoder may only use the data provided by the encoder to reconstruct a frame sequence, it must do the opposite procedure from the encoder to rearrange the information. After the coded coefficients and parameters are read, they pass through the entropy decoder where they are interpreted. Then, they go through the spatial model in order to obtain the residual frame/block. In the end, using the prediction parameters and previously decoded pixels, the various images that make up the file are retrieved.

2.2.2 Prediction Model

Most of the codecs' compression gains come from the prediction model, which plays a key role in video encoding. Given one or more images it can use spatial and/or temporal correlations to build a simplified representation of the input sequences. The final product of this process is a set of residues, resulting from the difference between the reference and predicted samples. The more accurate the prediction, the smaller the variance of the residual block, thus allowing a reduction in the number of bits to represent the information. With the residual data and the configuration to make the prediction, it becomes possible to reconstruct the original video.

Intra Prediction

In intra prediction, previously encoded blocks are compared with other sections of the same reference image. This procedure is used to identify the blocks that have the most similarities with one another, taking advantage of spatial redundancy. This idea consists of the fact that the pixels that are closer to each other have identical characteristics. For this reason, some regions of a picture can be used to represent other parts, without significant differences between the original and final image. Figure 2.7 illustrates this point with a portion of a picture of a blue sky. The pixels inside the 4x4 block are extrapolated by using the pixels above and to the left of the block. The prediction pixels are often located quite near the block being predicted.

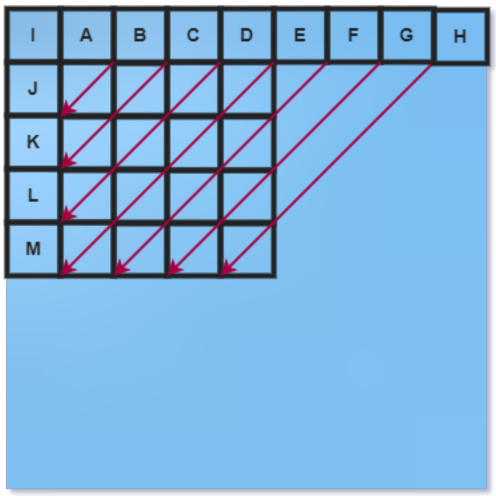


Figure 2.7: Intra Prediction

When a section is selected using this method, to it a prediction mode is assigned and also a motion vector is calculated for it. These parameters provide the encoder with information about the reference pixels and the location of the anticipated block. For example, in VVC there are 67 intra modes, of which 65 are directional, one planar and one DC. This implies that 67 different prediction scenarios are tested for each block in order to determine the combination of blocks that are most similar to each other.

Inter Prediction

In this type of prediction, the encoder takes advantage of temporal redundancy. A video is composed of several frames organised through time, so sequences close to each other share roughly the same amount of information. Thus using these similar sections from various images, it is possible to reduce the amount of storage needed to represent the file. Inter prediction allows the codec to use blocks from different frames to represent other areas. After finding a match with the most similarities (motion estimation), a motion compensation prediction is computed, and a motion vector is calculated. All this is then transmitted to the next stage of the encoder.

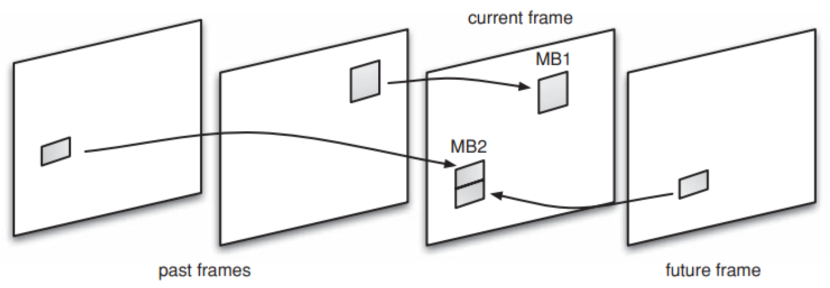


Figure 2.8: Inter Prediction [9]

It is important to note that pixels from past, present and future can be used for prediction, as shown in Figure 2.8

Residual

The metric used to obtain the degree of similarities between blocks of pixels is the SAE (Sum of Absolute Errors):

$$SAE = \sum_{i=1}^{N} |O_i - P_i| \tag{2.2}$$

O: vector containing the block's pixels for prediction

P: vector with the predicted block's pixels

By calculating this metric, the residual is obtained. The selection of the most appropriate mode for a set of predictions is based on the mode that leads to the minimization of the residual, i.e., the most appropriate mode will be the one that has the smallest SAE.

2.2.3 Partitioning

Before the intra or inter prediction is executed, the video frames are divided into rectangular structures, known as macroblocks, Coding Tree Units or superblocks. This process is called partitioning. Since using large blocks to divide an image isn't always superior, the use of smaller structures is sometimes required to fit the needs of regions that have more texture or detail. The use of small shapes makes the prediction more exact, leading to a less distorted outcome of the encoding result. Yet, the excessive use of it leads to the increase of bits to represent a frame, because more blocks mean more instructions. This methodology consists of finding the best manner of splitting frames into blocks in order to find a good balance between the bitrate and distortion.

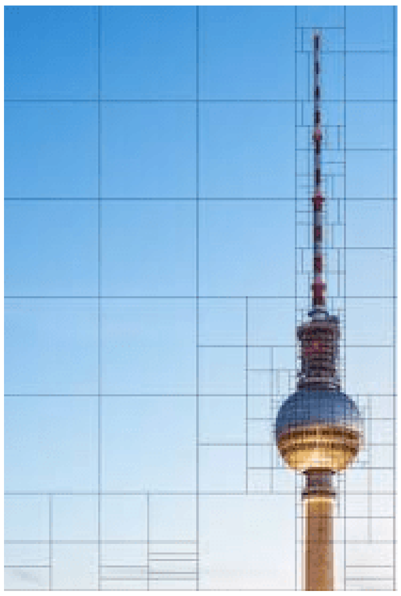


Figure 2.9: Partitioning [10]

Coding Tree Units

A CTU typically is composed of three Coding Three Blocks: one for luma and two for croma. The initial size of the CTUs is defined within the codec and can be either 128x128 or 64x64. The choice of these dimensions will affect the coding of the video file. It is expected that for ample pictures the use of large sizes of superblocks yields better results since they can take advantage of big areas with little colour variation in the pixels. After the partitioning of the image into macroblocks, they can be subdivided recursively into smaller structures called Coding Units (CUs).

Rate-Distortion Optimization

The metric that is used to obtain optimal block partitioning in each frame of a video sequence is called Rate-distortion Cost (RD cost), specified as

$$J = D + \lambda * R \tag{2.3}$$

D: the distortion

λ: Lagrangian weighting factor (depends on the QP)

R: bitrate

As noted before, this RD cost represents the trade-off between the number of bits used to encode and the quality loss of the frames. Depending on the application, this balance tends towards one facet or the other. For instance, in the case that an HD video is required, the amount of smaller blocks will increase, thus the distortion will be less impacted. However, the compression ability will be reduced.

The optimal value for the RD cost is obtained through a process of Rate-Distortion Optimization (RDO). Through this procedure, the encoder recursively searches for the proper way of breaking the frames into pieces until it finds the global minima for the RD cost. A trial-and-error approach is conducted to find the best block partitioning. The use of RDO imposes a real challenge in the codec speed, because of the huge computational complexity needed to minimise the RD cost being.

2.2.4 Spatial Model

In this stage, the goal is to reduce the amount of redundant data existing inside the residual obtained from the prediction. In other words, the objective is to decrease the number of bits needed to represent the information. To achieve this reduction, transforms and quantisation are typically used. Moreover, the process of using transforms is to enable quantisation in order to remove irrelevant data from the residual. The following sections explain these two concepts with more clarity.

Transforms

Data sometimes can not be easily handled in specific domains. So to ease the use of the information, transforms are used to represent the information as sets of uncorrelated coefficients. The most significant advantage of this tool is that it is lossless and reversible (inverse transform). Its usage allows it to manage data in the transform domain and convert it back to its original one without losing any information.

There are two main categories of transform in video coding: block-based and image-based. Block-based needs less memory to function and are well fitted in the compression of block-based motion compensation residuals, though they suffer from artefacts at block edges. On the other hand, image-based transforms memory requirements are high and are meant to operate with an entire image or a large section — this category shows better compression results in still images than the previous one.

The primary video codec transforms are the Discrete Cosine Transform (DCT) and the Discrete Sine Transforms (DST). These two fall into the category of block-based. They receive as input NxN residual blocks and output a new one with frequency components. Each frequency represents a coefficient. The higher the value, the more variation there is in the block's pixels. These types of components are the ones that are more helpful to humans since we are more sensitive to their change. For this reason, these coefficients are less quantised.

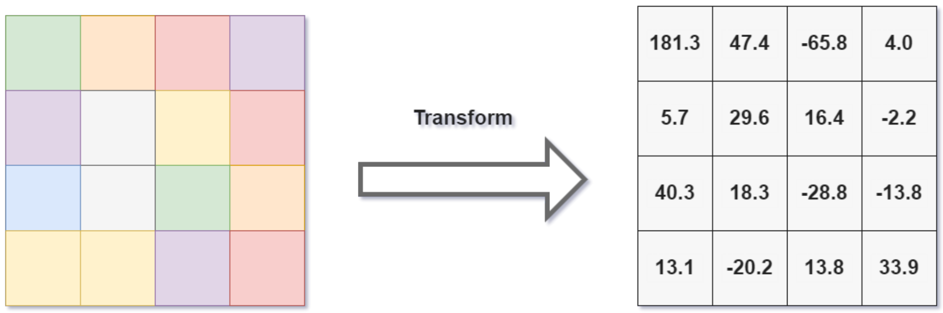


Figure 2.10: Illustration of application of transform

Quantisation

During quantisation, insignificant coefficients from the transform block are removed. This technique aims to map a signal that is represented with X amount of bits with fewer bits. The usage of this method causes a reduction in the precision of the image data. The reduction is accomplished by dividing the values from the transform by the Quantisation Parameter (QP), obtaining the integers from the result, and afterwards removing small coefficients, like numbers near zero. If QP is a large number, the compression level will be higher, and the amount of loss data will also increase. However, if it is a small parameter, the opposite happens. There will be more proximity between the original values and the re-scaled ones (less data loss), and the number of bits required to represent the information increases. The choice of the QP has to take these principles into account.

After quantisation, the blocks are composed mainly of zero-valued coefficients and a few non-zero ones. In order to increase clusters of similar values, these numbers are re-ordered in 1-dimension. Subsequently, the zero-valued weights are efficiently encoded, taking advantage of clusters previously created.

2.2.5 Entropy Encoding

Entropy encoding is a lossless process that takes as input a series of symbols describing components of a video sequence and outputs a appropriate compressed bitstream for transmission/storage. The input ranges from motion vectors and macroblock headers to quantised transform weights and other configuration parameters. The data is compressed by taking advantage of frequently occurring input symbols and representing them by a more straightforward structure, allowing fewer bits to be used. For instance, previously encoded motion vectors can be utilised to compute others. Then, instead of transmitting the vector, the instructions to generate it are provided.

2.3 Subjective and Objective Quality

After encoding, evaluation of compression quality is necessary in order to rate the codec performance. Visual quality measurement is associated with the HVS. Since this measurement depends on the condition of the HVS, it is inherently subjective. The clarity of a video is determined by elements like spatial fidelity, which describes how noticeable a distortion is, and temporal fidelity, which depicts how smoothly a transition occurs between successive frames.

Given the subjectivity associated with measuring quality, developers of codecs and video processing systems end up using a more objective metric, the Peak Signal to Noise Ratio (PSNR).

$$PSNR = 10 * \log_{10} \frac{(2^n - 1)^2}{MSE} \tag{2.4}$$

n: number of bits per image sample

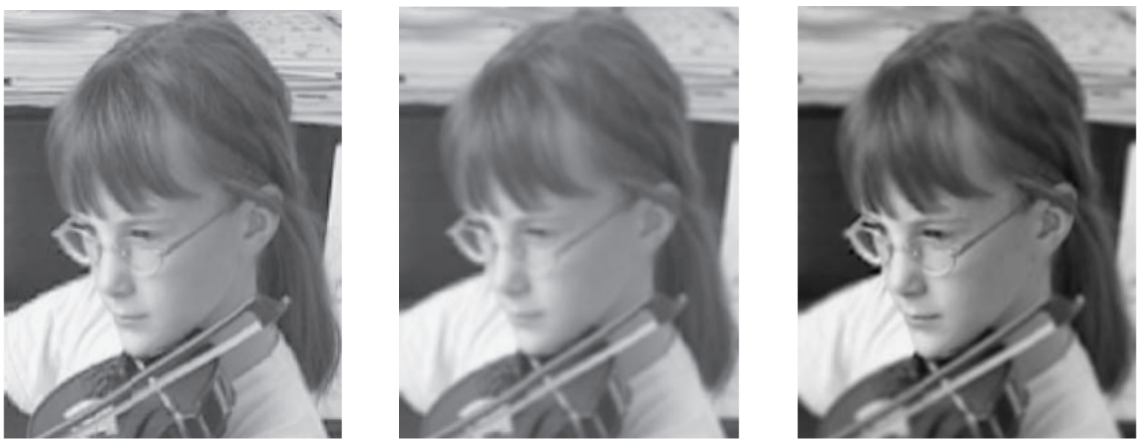
The PSNR is represented on a logarithmic scale and results from the square between the maximum possible value in the signal and the Mean Squared Error (MSE) between the original and the compressed image.

$$MSE = \frac{1}{M * N} \sum_{i=0}^{M-1} \sum_{k=0}^{N-1} [O(i,k) - C(i,k)]^2 \tag{2.5}$$

O: original image

C: compressed image

Despite being widely used in the scientific community, the PSNR is still not a good metric. The Figure 2.11b and 2.11c have a PSNR of 28.3dB and 27.7dB, respectively, compared to the original image (2.11a). The higher the PSNR, the higher the quality denotation. On one hand, through a visual (subjective) analysis, it appears that the picture on the right is the one with the greatest clearness, namely due to the sharpness of the face. However, on the other hand, through a (objective) comparison between the two using the PSNR, it appears that the one on the left is the superior one. These two examples highlight the dichotomy between subjective and objective quality.



(a) Original image (b) 28.3dB (c) 27.7dB

Figure 2.11: Subjective vs Objective quality [11]

2.4 Standard Codec

In this Chapter, some more particular topics about VVC will be covered and a comparison between it and its predecessor will be made. The concept of partitioning in VVC will be detailed, as it has a vital role in the work done in this thesis.

2.4.1 Block-based Codec

In a block-based hybrid codec, an image is divided into several blocks of pixels and forwards this information to the prediction model. It is called hybrid because it supports intra and inter operating modes. The strategy implemented by this type of encoder makes space for a simple and practical algorithm, which allows the use of block-based transforms (such as the Discrete Cosine Transform), and establishes a competent temporal model. Despite the advantages, this method brings with it some problems, all associated with the fact that real-life objects have a complex appearance and motion. VVC, HEVC, AVC, and their predecessors are hybrid block-based encoders. All of them contain a prediction model, a transform application and quantisation stage, and an entropy encoder. However, although they have functional blocks in common, the behaviour of each block in the standards is different, with the newer versions having the highest amount of complexity due to their greater diversity.

2.4.2 Partitioning in VVC and HEVC

The key objective of partitioning is to divide frames into pieces in a way that results in a reduction of the RD cost. To achieve a perfect balance of quality and bitrate, numerous image fragments combinations must be tested, which is computationally expensive. Due to the intensive nature of this process, a high compression rate can be attained. Partitioning contributes heavily to both the complexity and compression gains in both VVC and HEVC. When comparing this method in each, it is possible to verify that the levels of both mentioned metrics are higher in H.266. This achievement and downside are the results of a more refined partitioning scheme.

Comparing H.266 with H.265, some similarities can be seen when it comes to partitioning. Both codecs, since they are block-based, organize a video sequence in many frames that are divided into smaller pieces. First, pictures are split into CTUs, and then they are divided into CUs. After this procedure, the behaviour is standard dependent. For the luma channel, the largest CTU size in VVC is 128x128 while the largest CTU size in HEVC is 64x64. Regarding the minimum size of a CU, it is 4x4 for the former and 8x8 for the latter.

The partitioning procedures for HEVC and VVC start to differ after the conversion of a batch of frames into CUs. In HEVC, CUs are always split in a quarternary tree (quad-tree) partition, meaning that all of them will have square shapes. The sizes of these structures range from 64x64 to 8x8, with powers of two and the same value for both width and height. Such setups restrict the codec ability to predict since it has few options for determining the proper CU size to match a particular area of an image. A prediction unit (PU) is created as a result of the splitting of a CU. It can have the same dimensions as a CU plus a 4x4 size, and predictions are made using it.

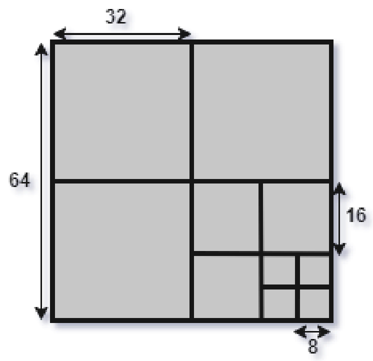


Figure 2.12: Partitions in HEVC

On the other hand, VVC uses a different type of partitioning scheme. A quad-tree is initially applied to the CTUs in the first level, and then a quad-tree with nested multi-type tree (QTMT) is applied recursively (Figure 2.13). This innovation makes it possible to split CUs in different rectangle forms besides simply square ones. Splitting a CU into three rectangles with a ratio of 1:2:1 results in a ternary tree (TT), with the center rectangle being half the size of the original CU. When applied horizontally it is called a horizontal ternary tree (HTT), and vertical ternary tree (VTT) when it is done vertically. A binary tree (BT) partitions a block into two symmetrical structures. Like the case of the TT, depending on the way the split is done, it can be called either a vertical binary tree (VBT) or a horizontal binary tree (HBT). The association of BT and TT is named a multi-type tree. The introduction of BT and TT partitions enables the creation of various new types of forms, with heights and widths that can be a combination between 128, 64, 32, 16, 8 and 4. The increased number of possible CUs boosts the ability of the codec to fragment an image more efficiently, allowing better predictions. Thus, increasing the compressing power.

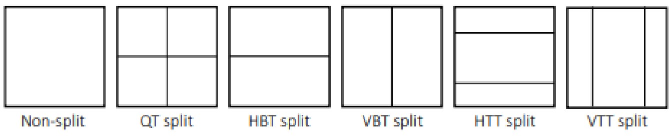
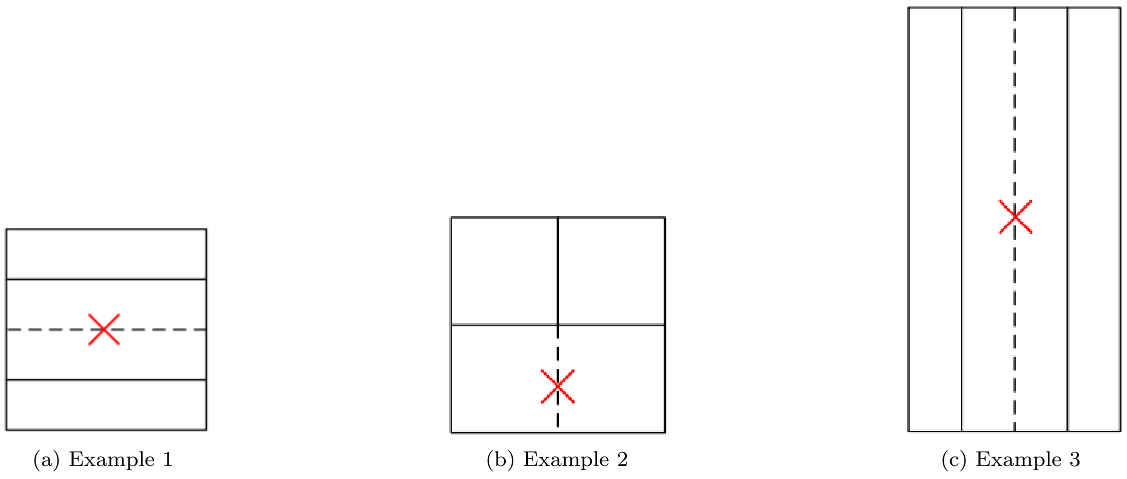


Figure 2.13: Partitions in VVC [12]

Furthermore, a set of guidelines are followed during the partitioning process to prevent the development of the same CU through a different sequence of splits. For instance, after a QT node is split using TT the resulting middle sub-part can not be partitioned using a BT in the same direction (Figure 2.14a). Because it is possible to acquire the identical sub-parts by using BT followed by another BT in both resultant blocks, this sequence of splits is prohibited. A second case is when a QT node is split using BT and the top sub-part is applied VBT, the other portion cannot be partitioned in the same manner (Figure 2.14b). The reason for this is that the end set of CUs could also be obtained through the use of a QT partition. Similar restrictions like these are also predicted in the codec. Additionally, the QT partition can not be applied again after the MT is utilised, no matter whether the MT node sub-block shape is a square or not.



(a) Example 1 (b) Example 2 (c) Example 3

Figure 2.14: Partitioning Restrictions in VVC [13]

VVC complexity analysis

The VVC partitioning process is the most complex when compared with the previous standards. This is verified both empirically and through the theory outlined in the preceding paragraph. In [14] the effectiveness and complexity of VVC and HEVC's coding were compared by the authors using a variety of omnidirectional video sequences, along with various QPs. It was determined that the H.266 standard takes 4.07 times longer to encode video sequences than its predecessor on average. Another comparable research was done in [15], except in this one, multiple modules from both codecs were examined using standard video sequences. The findings from these authors were consistent with the previous mentioned paper. Without Single Instruction/Multiple Data (SIMD), VVC saw a complexity increase of 15.88, while with SIMD, it experienced a gain of 10.17. In this context, SIMD refers to a type of parallel processing that optimizes the coding process. This investigation also came to the conclusion that Filtering and intra prediction were the two modules with the biggest complexity increase. The complexity reduction potential for three VVC tools—Multiple Transform Selection (MTS), Intra Mode Prediction (IMP), and Block Partitioning (BP)—was investigated by the writers of [16]. In this experiment, the encoding time was recorded while each of the codec components were alternately disabled and enabled. A decrease of up to 97% was shown to be attainable for BP, 55% for MTS, and 65% for IMP, it was determined. With this result, it is evident that the partitioning process could be greatly simplified for VVC.

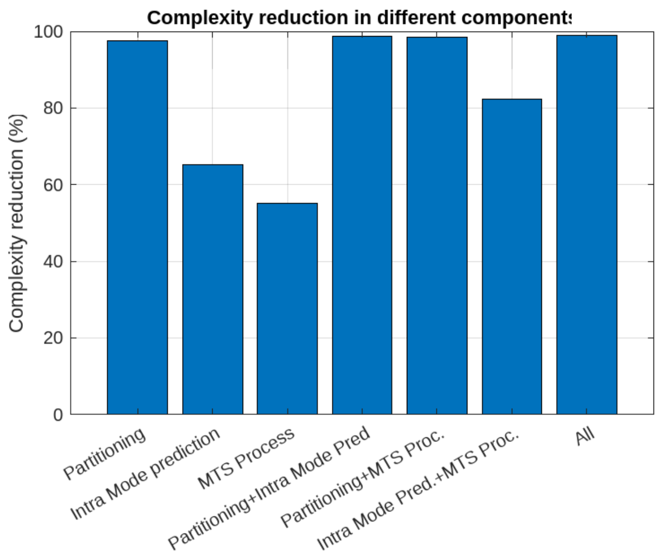


Figure 2.15: Codec components and possible complexity reduction when they are disabled [16]

2.5 Deep Learning

Frank Rosenblatt is considered the father of Deep Learning (DL) since he built and researched the fundamental component of this field, the perceptron [17]. Although his discoveries were made in the 60s, it took 20 years for more research and interest to develop in this field of study.

While machine learning (ML) is a branch of Artificial Intelligence (AI), DL is a ML sub-branch (Figure 2.16). The data that they use is what makes DL and ML so different from one another. While ML leverages labelled and structured data, DL uses data that is more in its raw state. In order to generate a model with the former, certain attributes from the data must be manually chosen. The latter, however, can extract the features required to complete a certain task. Both ML and DL teaches computers to do what comes naturally to humans: learn by example. DL algorithms self-learn to identify hidden patterns in data using enormous datasets and a large computing capacity. Driverless vehicles, chatbots, and other innovations are part of the many applications in this field.

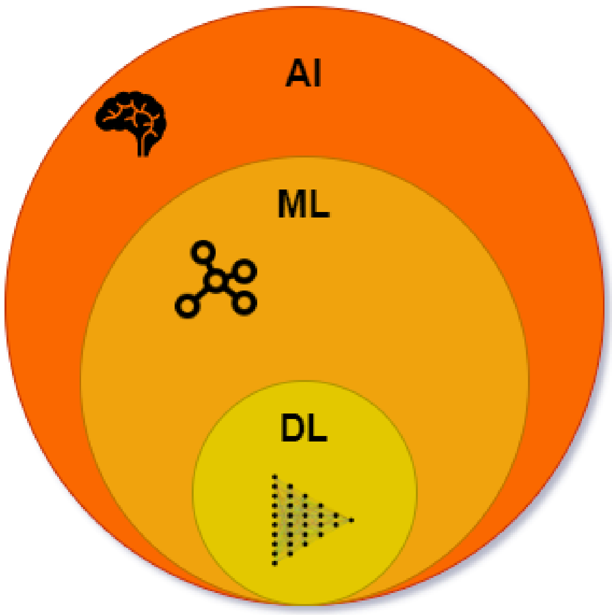


Figure 2.16: AI, ML and DL

ML models may learn in a variety of ways, which the main two are: supervised learning and unsupervised learning. To categorise or make predictions, supervised learning uses labelled datasets; this involves some sort of human interaction to accurately label input data. Unsupervised learning, in contrast, does not require labelled information; instead, it analyses the data for patterns and groups them according to a specific identifying traits.

Implementing a learning mechanism will help a model grasp what the output should be given a certain input. In order to increase the accuracy of the results, learning entails changing the weights of the network. The observed mistakes are minimised to achieve this. Learning is complete when looking at more data does not improve the algorithm performance. A cost function, or loss function, is designed in order to train the network and improve its capability for sound decision-making. The more the outputs of the model are close to the expected value, the lower the cost function value is. The majority of learning models may be seen as a simple integration of statistical estimates and optimization theory.

Backpropagation is a technique used to modify the weights in order to reduce the error during learning. For a single input-output example, backpropagation calculates the gradient of the loss function with respect to the network weights and updates them. Since this procedure is gradient-based, techniques that resemble gradient descent can be employed to update the weights.

Modern DL networks can have many layers and parameters. As an example, Google's GoogleNet has around 6.8M parameters. A staggering amount of computations must be performed for these enormous models. In the last 4 years, large-scale cloud AI training has mostly moved away from CPUs and onto Graphic Processing Units (GPUs) [18]. The capacity of a GPU to do computations in parallel is one of its most valued features, making it suitable for the learning process of DL algorithms. Comparing GPU capabilities to those of a CPU, the paralleling ability makes the execution of many calculations at the same time more advantageous than the versatility abilities of a CPU.

It is important to note that DL, although a sub-field of ML, encompasses a wide range of research fields. Therefore, this Section will only discuss the pertinent subjects that were crucial to the creation of this thesis's proposed solution. There will be an introduction to Artificial Neural Networks, Convolutional Neural Networks, Residual Neural Networks, and Metrics. A summary of the most recent Deep Learning approaches is presented at the end of this section.

2.5.1 Artificial Neural Networks (ANNs)

Artificial Neural Networks (ANN) are powerful models whose main idea is derived from the nervous system of animals. The nervous system's basic element is the neuron and they can be arranged into circuits that handle particular types of information. An example of a neuron is seen in the Figure 2.17. The axon, dendrite, myelin, and cell body are this cell's primary elements. The configuration of neural circuits varies widely depending on the desired function, but all sets have common characteristics.

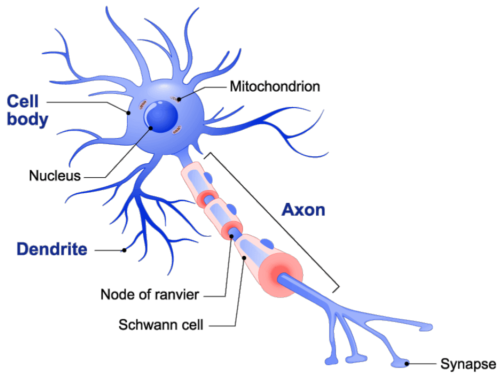


Figure 2.17: Neuron [19]

With the core concepts of how these biological structures function, the artificial neuron (also known as a perceptron or node) was created. Just like its biological counterpart, one or more inputs are given to the perceptron and their aggregate results in an activation. A non-linear function known as an activation function is typically applied after each input is individually weighted, as represented in Figure 2.18. Depending on the application, this structure can have different inputs and outputs. Images, audio, or text can all be used as input, while the output might be any real number.

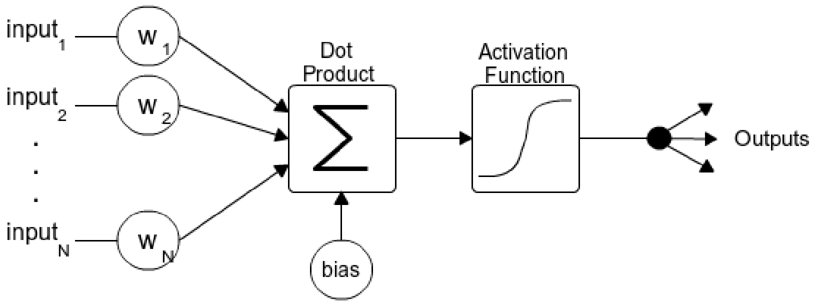


Figure 2.18: Artificial Neuron [20]

A collection of nodes organized in a specific way is considered an ANNs. In this organization, the output of a neuron is passed down to another one until the final output is obtained. Depending on the complexity of the network it can have a lot of layers and many nodes. The first layer is called Input Layer, the last one Output Layer and the ones in between are Hidden Layers (Figure 2.19).

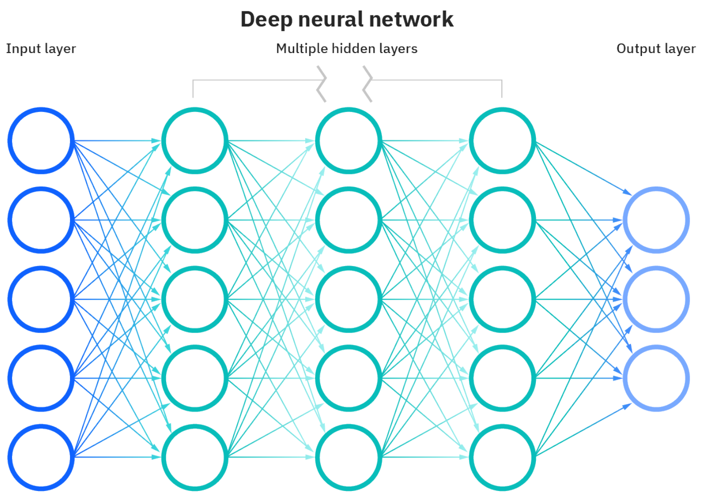


Figure 2.19: Artificial Neural Network [21]

Depending on the number of layers of an ANN, it can be considered either a Shallow Neural Network or a Deep Neural Network. Although there isn't a particular depth at which shallow and deep learning are mutually exclusive, most researchers agree that deep learning requires at least three layers. It has been demonstrated that an approximator with this many layers may estimate any function [22]. Deep Neural Networks help in decoupling abstractions and identifying the features that enhance performance. As a result, bigger networks are better equipped to identify patterns in the data. Figure 2.20 demonstrates this concept by illustrating how the first layers begin to detect fundamental traits and the last layers more intricate ones.

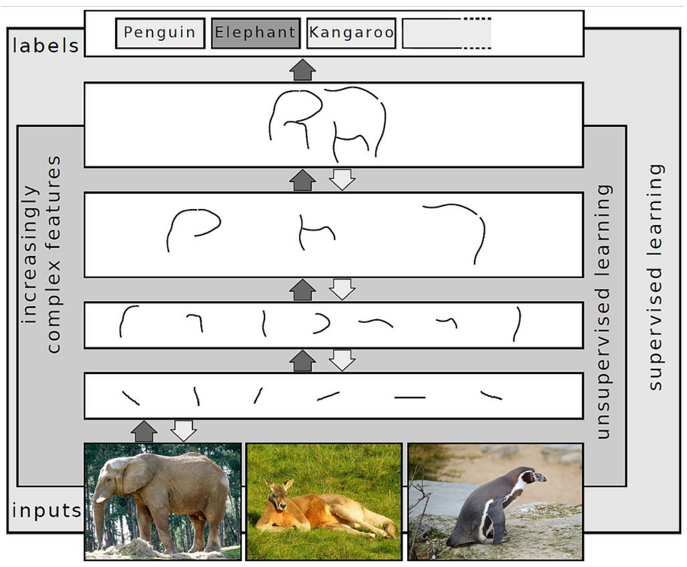


Figure 2.20: Feature extraction [23]

Transfer learning in deep learning is the act of transferring the weights from one model designed for a specific purpose to another. The model with the updated weights is typically subsequently modelled to match the new problem, in a process known as fine-tuning. Since a network's earliest layers extract fundamental features and its final levels build on that knowledge, it is common for the final layers to undergo a second training phase using fresh data in order to better serve their new function. This strategy not only speeds up the training process, by just using the last layers rather than the complete network, but it also offers a great deal of flexibility by reusing previously trained weights.

2.5.2 Residual Neural Networks (ResNet)

Sometimes the size of an ANN needs to be dramatically increased in order to address extremely difficult challenges, however when we add more layers, a typical issue in deep learning known as the vanishing/exploding gradient arises. This means that the gradient tends to zero or to a very large number. Therefore, the training and test error rate similarly increases as the number of layers is increased.

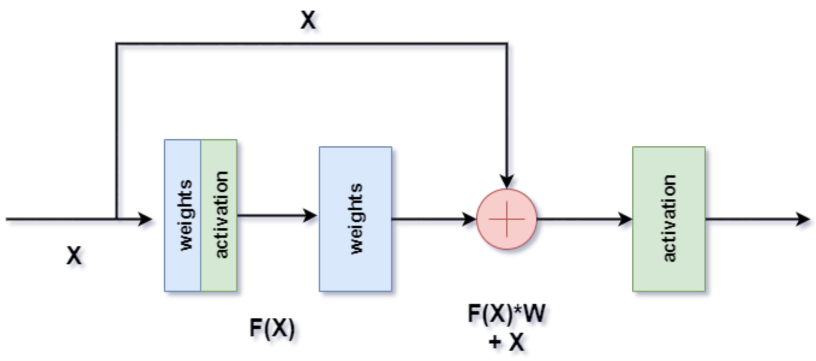


Figure 2.21: Residual Block

To solve the vanishing/exploding gradient problem, residual blocks [24] were developed. The skip connections technique is used by this network. The skip connection links the activation of one layer to the output of the weights of a subsequent one, creating a residual block, as seen in Figure 2.21. To build ResNets, these blocks are stacked.

2.5.3 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are neural networks that use the mathematical technique known as convolution. Convolution is a particular sort of linear operation where kernels or filters move along input data and produce a group of values that collectively make up a feature map. This model is in essence a neural network with convolution used in at least one layer instead of standard matrix multiplication. Since the use of multi-layer perceptrons in certain problems may lead to giant networks and the fact that they are prone to overfitting, CNNs become a viable solution for those problems. CNNs become a more practical option because the usage of multi-layer perceptrons may occasionally result in massive networks. Before CNNs, identifying objects in pictures required the use of laborious manual feature extraction techniques. By extracting patterns in an image, CNNs now provide a more scalable method for tasks such as object detection and image classification. However, they can be computationally demanding, requiring GPUs for model training.

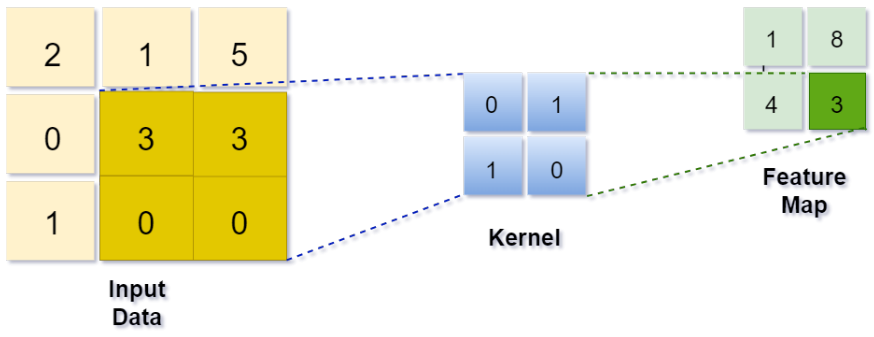


Figure 2.22: Convolution operation

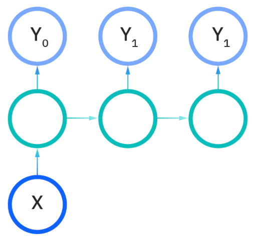
Figure 2.22 illustrates the convolution operation in 2D data. Starting from the top left of the input data, the four weights in the kernel multiply with the numbers in the input data that are in the same location as them. Following that, the outputs of the four processes are added together to get the feature map initial value, which is $2 * 0 + 1 * 1 + 1 * 0 + 3 * 0 = 1$. The next phase is a slide to the right by the kernel, followed by the same computations but with different numbers. When the input data's rightmost limit is reached, the kernel moves downwards and returns to the leftmost section. The process is then repeated up until the right bottom portion of the input grid is reached. Data that are in other dimensions are handled using a similar approach.

Regarding the kernel, there are some relevant hyperparameters to note: the stride, the number of filters and the padding. The stride is the number of pixels that the kernel moves in each step it takes (rightwards and downwards) on the input matrix. The number of filters affects the depth of the feature map. Finally, padding is done by appending zero values to the input data boundaries.

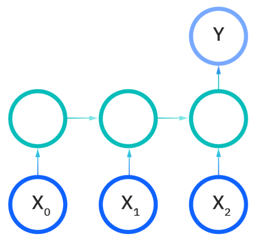
Convolutional layers are only one of a few layers that a CNN contains. In order to reduce the number of parameters in the input, the Pooling layer performs dimensionality reduction. The main difference between this layer and the convolutional one is that the former do not contain weights. In these layers, a choice is made that affects the entire region the filter overlaps. Max-pooling is the process of moving a filter over the input and choosing the pixel with the highest value. In contrast, average pooling outputs the average value found inside the receptive field. The fully-connected (FC) layer is a further layer that is constituent of a CNN. This layer serves as the output layer of the model and consists of collection of perceptrons.

2.5.4 Recurrent Neural Networks (RNNs)

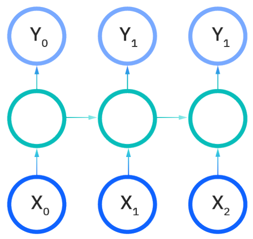
An RNN is a type of neural network that results from many similar nodes linked together sequentially. During calculation, RNNs have structures in which information about the computation is stored. Due to the ability to store previous obtained knowledge, a RNN can handle sequential data and can accept both the current input data and the ones received in the past. Natural language processing (NLP), speech recognition, and other temporal problems and matters where the order of the data is relevant are frequently addressed by these networks. In the Figure 2.23, there are various types of RNNs presented. The sequential nature of this network can be seen in the configurations 2.23a, 2.23b, and 2.23c, where the output of a node is influenced not only by its input $X_{0,1,2}$, but also by the activation of the previous node. In this Image, it is also possible to see that each node can have its own output.



(a) One-to-Many



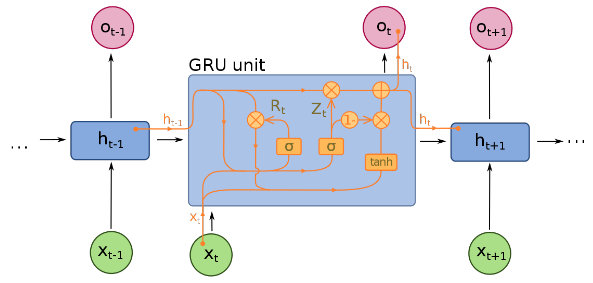
(b) Many-to-one



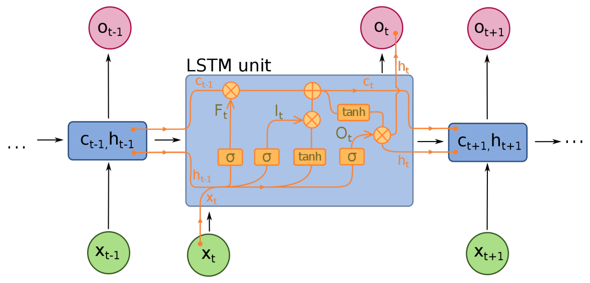
(c) Many-to-Many

Figure 2.23: Types of RNN [25]

Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GTU) are the two main RNN variants. LSTM is a more sophisticated recurrent neural network architecture that was created to simulate more accurately chronological sequences and their long-range relationships. This variant prevents the vanishing gradient challenges posed by traditional RNNs [26]. Regarding the GRUs, they are comparable to LSTMs in terms of structure, but they contain fewer parameters. In Figure 2.24, it is possible to see that the GRU as a simpler architecture.



(a) GRU



(b) LSTM

Figure 2.24: Main versions of RNNs [27]

2.5.5 Metrics

Every machine learning pipeline has quality measurements. Progress in the training process and the quality of model itself, is quantified through these metrics. These indicate what characteristics in the results of the model are most important. Therefore, it is essential to use the appropriate criterion when assessing any model. Metrics may be classified into many different categories, including regression metrics, classification metrics, ranking metrics, statistical metrics, and many more. In the context of this thesis, only classification metrics will be addressed.

It is important to understand binary classification in order to properly comprehend the classification metrics used in ML. This technique separates a collection of data into two groups, one of which is thought to be positive and the other to be negative. Four categories were established in order to specify a relationship between the model's predictions and the actual ground-truth:

- True Positives (TP): this happens when the actual class and the projected class are both members of the positive class.
- True Negatives (TN): occurs when both the actual class and the projected class are members of the negative class.
- False Positive (FP): the situation in which the actual class belongs to the negative class while the predicted class belongs to the positive one.
- False Negative (FN): this occurs when the predicted class is from the negative and the actual class is from the positive.

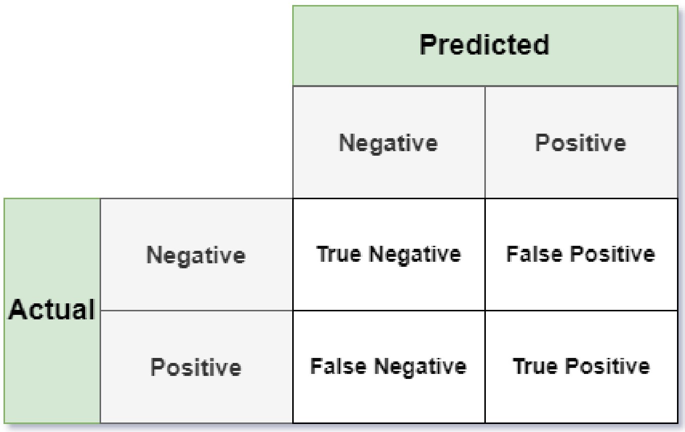


Figure 2.25: Binary classification

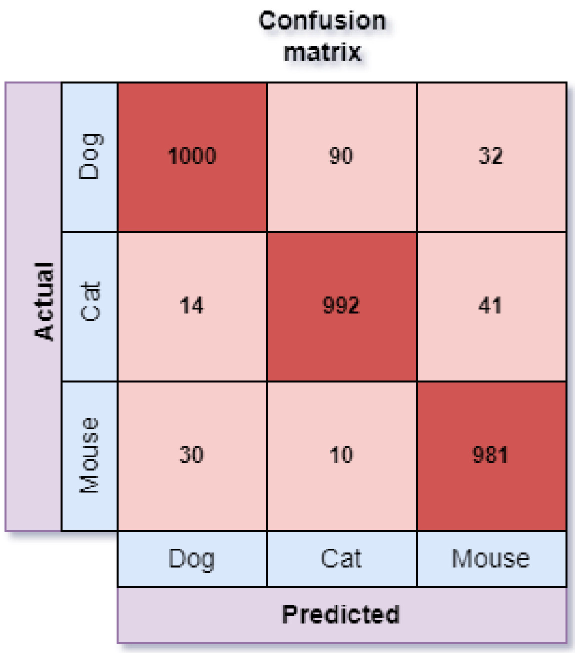
It is crucial to be aware that accuracy might be a deceptive measure. In issues with unbalanced classes, if a class makes up 90% of the total data and the model's ability to predict it is enormous when compared to the other classes, this measure will not reflect the model's robustness regarding prediction. The other metrics presented in the Table 2.5 are capable of showcasing the imbalances in the data.

Metric	**Description**	**Equation**
Accuracy	The proportion of accurate predictions to all predictions	$Accuracy = \frac{TN+TP}{TN+TP+FP+FN}$
Precision	The proportion of the correctly predicted classes from all the classes that were considered positive	$Precision = \frac{TP}{TP+FP}$
Recall	The percentage of samples from the positive class that the model successfully predicts	$Recall = \frac{TP}{TP+FN}$
Specificity	The proportion of samples from the negative class that the model correctly predicts	$Specificity = \frac{TN}{TN+FP}$
F1-Score	The weighted average for precision and recall. Provides a broad overview of the model's ability to predict both positive and negative classes	$F1_{score} = \frac{2*Precision*Recall}{Precision+Recall}$

Table 2.5: Classification metrics

A confusion matrix is a summary table showing the number of accurate and wrong predictions generated by a classifier. These can only be used in supervised learning algorithms where the output distribution is known.

Confusion matrices are extensively utilised because they provide a more precise representation of a model's performance than accuracy. In the example in Figure 2.26, a confusion matrix is shown. The forecasts are in the columns, while the ground-truth of the model is in the lines. This table displays the frequency with which each category was assigned to a certain class. In the example, it is clear that 992 of the "cat" samples were correctly labelled, while 14 and 41 of the other samples were labelled as dogs and mice, respectively. Similar information may be derived from the other classes. This data can alternatively be expressed in a normalised format.



Confusion matrix

Actual \ Predicted	Dog	Cat	Mouse
Dog	1000	90	32
Cat	14	992	41
Mouse	30	10	981

Figure 2.26: Confusion matrix example

This matrix's information may be utilised to calculate the previously mentioned metrics, Recall, Precision, and F1-Score. This is done by picking one class from each group to be the positive class and the others to be the negative classes.

2.5.6 Machine Learning Applications

Year after year, it becomes evident that ML and the algorithms that comprise it can tackle several problems in different fields. Numerous classification and regression problems can be solved by supervised learning algorithms such as Random Forest (RF), Decision Trees (DT), Support Vector Machines (SVM), K Nearest Neighbors (KNN), Naive Bayes (NB), Logistic Regression (LR), Linear Regression, Deep Learning Networks, and many others. Several of these algorithms improved fields ranging from health to the environment, with some of them outperforming conventional tactics. The abundance of data available now is one of the primary reasons for the growth of this domain. With the increase of amount of information available, several ML algorithms are able to generalise more effectively.

Even if ML is still in its infancy, the accomplishments that have already been made are tremendous. Providing evidence that it has become a permanent part of human existence. In this portion, the present state-of-the-art of this groundbreaking scientific discipline will be reviewed.

Energy

Attempts have been made to apply ML to the topic of energy. In [28], the novel applications of ML in renewable energy data are discussed. Linear regression, LR, DT, SVM, NB, KNN, K-Means, RF, DRA, and GBA were identified as the most prevalent techniques used to work with energy data. Additionally, some application examples of a couple of these methods are provided. In [29], for instance, a strategy to improve the accuracy and sensitivity of Heating, Ventilation, and Air Conditioning (HVAC) power consumption forecast was developed. This was accomplished by employing clustering techniques on the data, constructing a model by converting the clustered hourly data to monthly data, and applying the NB classifier to categorize the hourly data under different operating conditions into the energy consumption model with the smallest prediction error. Moreover, a comparison was done between an ANN, the NB, and a Multi-Variable regression (MVR) model; in this instance, the NB generated the best results, achieving a normalised mean bias error (NMBE) of 0.73%. Furthermore, [28] guides how to approach modelling in this setting. For example, short, medium, and long-term power projections are required due to the necessity for efficient grid operation and management. The most popular method for forecasting medium and long-term trends based on historical and socioeconomic factors is the MVR. Statistical approaches and DL methods such as neural networks and other from the family are recommended for short-term forecasts.

Health

Concerning health, several ML strategies were also utilised to tackle obstacles. In [30], a variety of DL applications in the health field are described. One of the applications mentioned was in [31], in which a CNN was used to predict whether digitised film mammograms contained cancer or not. On the Digital Database for Screening Mammography (CBIS-DDSM) test set, the best model scored an AUC of 0.88 per picture, but on the INbreast database test set, the best model achieved an AUC of 0.95 per picture. It is essential to note that several models were also developed. Using a combination of the built models to categorise the photos yielded an AUC of 0.91 for the first dataset and 0.98 for the second. Additionally, the constructed CNNs were based on two renowned topologies, the ResNet50 and the VGG-16. To use these architectures, pre-trained weights and transfer learning techniques were applied. The implementation of these network designs demonstrates the adaptability of these models, as they were not created for this kind of application. Moreover, in [30], DL approaches are the most used in medical imaging, despite the reduced availability of annotated data in this area.

Environment

The environmental area has also profited from the contributions of machine learning. ML techniques were utilised in [32] to estimate the Air Quality Index (AQI). The data was collected from major industrialised cities in India, including Bangalore, Delhi, and Ahmedabad. In this investigation, the researchers assessed the performance of DT, LR, RF, and GB models, as well as hybrid models comprised of LR+DT and DT+GB. The metrics selected for the comparison were the Mean Absolute Error (MAE), the Root Mean Square Error, and the correlation coefficient (r). In contrast to classical ML models, the error levels of hybrid models were shown to be considerably different. The MAE, RMSE, and correlation coefficients for the hybrid between LR and DT were 0.885, 4.948, and 0.9990, respectively. In comparison, the other combination yielded 0.91, 5.153, and 0.9991 correspondingly. RF and DT were the two standalone ML algorithms with the best performance. This research not only demonstrates the efficacy of ML approaches in forecasting the AQI, but also demonstrates that the combination of these techniques is superior to using the models individually. Similar to [31], the ResNet50 architecture and several others were utilised for categorisation purposes in [33]. The goal of this study was to classify various types of household garbage using CNNs. Using a dataset of images labelled with five distinct classes, the ResNet50 architecture achieved an accuracy of 87.5%. It is important to highlight that the accuracy metric for each class was above 80%.

Video coding

Next, within the scope of this thesis, some proposals about the use of ML in video coding (VVC) will be presented. In [34], a comprehensive assessment of current practises is provided. According to this investigation, as already mentioned in previous chapters, to obtain the highest possible compression efficiency, it is important to determine, for each coding tool, the optimal trade-off between rate and distortion. As previously stated in [16], the tools that take the most processing time are BP, IMP, and MTS; ordered from the most complex to the least. Consequently, they are the focus of most of complexity reduction strategies mentioned in the literature, with the majority of current research concentrating on block partitioning. Using heuristics and machine learning are the two primary methods for simplifying this standard. Using statistical or non-statistical properties of the VVC tools, algorithms are built utilising heuristics to predict the optimal outcome for each instrument. In the second strategy, the usage of ML is established by developing a model to forecast the result of the steps of the VVC coding process and then applying a huge set of data to train the model on how to perform the predictions. According to their relationship with the coding procedure, [34] notes that there are two primary ways for data extraction and training. In the in-loop technique, data is extracted and models are trained during the encoding process itself. Due to the complexity overhead introduced by the model training process, this method often provides less complexity reduction. The alternative is the off-loop, in which data and models are trained in advance. This technique leads to a greater loss of coding efficiency since the models do not adjust to the time-varying behavior of video. A thorough analysis of several approaches to lessen the VVC complexity is also included in [34]. The main takeaways from these comparisons are: although ML-associated techniques produced superior results in terms of complexity reductions, the Bjontegaard Delta rate (BD-rate) loss was 0.29% worse than the heuristic approaches regarding BP in intra coding; even though block partitioning is the primary contributor to the VVC complexity, approaches that focus on low-complexity intra mode estimate can achieve bigger reductions than others; methods based on machine learning that concurrently target intra and inter coding are the most effective. Furthermore, in this research, an RF model was developed to simplify the complexity of the VVC BP tool. Regarding complexity reduction, 73.10% was attained. This decrease is the greatest among those listed in this literature. Regarding BD-rate loss, 5.89% was attained.

The research that prompted this thesis may be found in [35]. To estimate the CU partition for intra prediction in accordance with the dynamic QTMT structure at multiple levels, the authors of this article suggest a multi-stage exit CNN (MSE-CNN) model with an early-exit mechanism. Since the RDO method is used by VVC, several CUs combinations are explored in order to locate the one that leads to the optimal partitioning scheme. MSE-CNN intends to exclude many of these tested CUs in order to decrease the complexity of this standard. In order to accomplish this, a database was created to train the model. This database contains a large number of YCbCr 4:2:0 image and video files. It can utilised not only for the training of the MSE-CNN, but also for other approaches. This method outperformed several previous methods demonstrated in the scientific literature, resulting in a complexity reduction of up to 66.88% with a BD-rate as low as 1.322%.

Final Remarks

Reviewing all that has been discussed in this section, one can conclude that ML brings several solutions, not only to the video coding sector, but also to many others. Current ML approaches, such as DL (CNNs and ANNs in particular), other more traditional techniques (with an emphasis on RFs and MVR), and a mixture of the traditional techniques constitute the state of the art in ML. Although some studies indicate that DL is now superior to other methodologies, as suggested in [36] and [37], the success of these algorithms is significantly dependent on the application for which they are being employed. Regardless, it is evident that DL is the ideal choice for image-related fields.

References

[1] “RGB color spaces - Wikipedia,” RGB color spaces - Wikipedia, Dec. 01, 2014. https://en.wikipedia.org/wiki/RGB_color_spaces (accessed Oct. 05, 2022).

[2] ”YUV - Wikipedia”. En.Wikipedia.Org, 2022, https://en.wikipedia.org/wiki/YUV.

[3] “Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios,” Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios, Mar. 2011. https://extranet.itu.int/brdocsearch/R-REC/R-REC-BT/R-REC-BT.601/R-REC-BT.601-7-201103-I/R-REC-BT.601-7-201103-I!!PDF-E.pdf (accessed Oct. 11, 2022).

[4] I. E. Richardson, The H. 264 Advanced Video Compression Standard, 2nd ed. 2010, p. 17.

[5] I. E. Richardson, The H. 264 Advanced Video Compression Standard, 2nd ed. 2010, p. 18.

[6] I. E. Richardson, The H. 264 Advanced Video Compression Standard, 2nd ed. 2010, p. 19.

[7] “A beginner’s guide to video resolution — Adobe,” A beginner’s guide to video resolution — Adobe. https://www.adobe.com/creativecloud/video/discover/video-resolution.html

[8] Digital Cinema Initiatives, “Digital Cinema System Specification,” Wayback Machine. https://web.archive.org/web/20160527180135/http://dcimovies.com/specification/DCI_DCSS_v12_with_errata 1010.pdf

[9] I. E. Richardson, The H. 264 Advanced Video Compression Standard, 2nd ed. 2010, p. 87.

[10] B. Bross, “Versatile Video Coding (VVC) on the final stretch,” VVC-bross-ITU-10-2019, Oct. 2019. https://www.itu.int/en/ITU-T/Workshops-and-Seminars/20191008/Documents/Benjamin_Bross_Presentation.pdf (accessed Oct. 05, 2022).

[11] I. E. Richardson, The H. 264 Advanced Video Compression Standard, 2nd ed. 2010, p. 22.

[12] Y.-W. Huang et al., “Block Partitioning Structure in the VVC Standard,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 31, no. 10, pp. 3818–3833, Oct. 2021, doi: 10.1109/tcsvt.2021.3088134.

[13] Y. Fan, J. Chen, H. Sun, J. Katto, and M. Jing, “A Fast QTMT Partition Decision Strategy for VVC Intra Prediction,” IEEE Access, vol. 8, pp. 107900–107911, 2020, doi: 10.1109/access.2020.3000565.

[14] J. Filipe, J. Carreira, L. Tavora, S. Faria, A. Navarro, and P. Assunção, “Omnidirectional Video Coding: New Coding Tools and Performance Evaluation of VVC,” Conference on Telecommunications - ConfTele, 2019, Published.

[15] Í. Siqueira, G. Correa and M. Grellert, ”Rate-Distortion and Complexity Comparison of HEVC and VVC Video Encoders,” 2020 IEEE 11th Latin American Symposium on Circuits & Systems (LASCAS), 2020, pp. 1-4, doi: 10.1109/LASCAS45839.2020.9069036.

[16] A. Tissier, A. Mercat, T. Amestoy, W. Hamidouche, J. Vanne and D. Menard, ”Complexity Reduction Opportunities in the Future VVC Intra Encoder,” 2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP), 2019, pp. 1-6, doi: 10.1109/MMSP.2019.8901754.

[17] C. C. Tappert, “Who is the father of deep learning?,” International Conference on Computational Science and Computational Intelligence (CSCI). IEEE, Dec. 2019.

[18] “GPUs Continue to Dominate the AI Accelerator Market for Now,” InformationWeek, Nov. 27, 2019. https://www.informationweek.com/ai-or-machine-learning/gpus-continue-to-dominate-the-ai-accelerator-market-for-now (accessed Oct. 05, 2022).

[19] ”Foc.Us”, https://foc.us/wp-content/uploads/2020/03/neuron-labels.png (accessed Oct. 05, 2022).

[20] ”Cluttered-Code.Github.Io”, http://cluttered-code.github.io/images/artificial-neuron.png (accessed Oct. 05, 2022).

[21] I. C. Education, “What are Neural Networks?,” What are Neural Networks? — IBM, Aug. 17, 2020. https://www.ibm.com/cloud/learn/neural-networks (accessed Oct. 05, 2022).

[22] S. Sugiyama, Human Behavior and Another Kind in Consciousness. 2019.

[23] H. Schulz and S. Behnke, “Deep Learning,” KI - Künstliche Intelligenz, vol. 26, no. 4, pp. 357–363, May 2012, doi: 10.1007/s13218-012-0198-z.

[24] K. He, X. Zhang, S. Ren and J. Sun, ”Deep Residual Learning for Image Recognition,” 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[25] I. C. Education, “What are Recurrent Neural Networks?,” What are Recurrent Neural Networks? — IBM, Sep. 14, 2020. https://www.ibm.com/cloud/learn/recurrent-neural-networks (accessed Oct. 05, 2022).

[26] D. Ravì et al., ”Deep Learning for Health Informatics,” in IEEE Journal of Biomedical and Health Informatics, vol. 21, no. 1, pp. 4-21, Jan. 2017, doi: 10.1109/JBHI.2016.2636665.

[27] “Recurrent neural network - Wikipedia,” Recurrent neural network - Wikipedia, Aug. 01, 2019. https://en.wikipedia.org/wiki/Recurrent_neural_network (accessed Oct. 05, 2022).

[28] M. Lahby, A. Al-Fuqaha, and Y. Maleh, Computational Intelligence Techniques for Green Smart Cities. Springer, 2022, pp. 41–66. doi: 10.1007/978-3-030-96429-0.

[29] C.-M. Lin, S.-F. Lin, H.-Y. Liu, and K.-Y. Tseng, “Applying the naïve Bayes classifier to HVAC energy prediction using hourly data,” Microsystem Technologies, vol. 28, no. 1, pp. 121–135, Jun. 2019, doi: 10.1007/s00542-019-04479-z.

[30] M. Lahby, A. Al-Fuqaha, and Y. Maleh, Computational Intelligence Techniques for Green Smart Cities. Springer, 2022, pp. 169-186. doi: 10.1007/978-3-030-96429-0.

[31] L. Shen, L. R. Margolies, J. H. Rothstein, E. Fluder, R. McBride, and W. Sieh, “Deep Learning to Improve Breast Cancer Detection on Screening Mammography,” Scientific Reports, vol. 9, no. 1, Aug. 2019, doi: 10.1038/s41598-019-48995-4.

[32] M. Lahby, A. Al-Fuqaha, and Y. Maleh, Computational Intelligence Techniques for Green Smart Cities. Springer, 2022, pp. 249-269. doi: 10.1007/978-3-030-96429-0.

[33] M. Lahby, A. Al-Fuqaha and Y. Maleh, Computational Intelligence Techniques for Green Smart Cities. Springer, 2022, pp. 271-293. doi: 10.1007/978-3-030-96429-0.

[34] M. Lahby, A. Al-Fuqaha and Y. Maleh, Computational Intelligence Techniques for Green Smart Cities. Springer, 2022, pp. 351-379. doi: 10.1007/978-3-030-96429-0.

[35] T. Li, M. Xu, R. Tang, Y. Chen, and Q. Xing, “DeepQTMT: A Deep Learning Approach for Fast QTMT-Based CU Partition of Intra-Mode VVC,” IEEE Transactions on Image Processing, vol. 30, pp. 5377–5390, 2021, doi: 10.1109/tip.2021.3083447.

[36] M. Lahby, A. Al-Fuqaha and Y. Maleh, Computational Intelligence Techniques for Green Smart Cities. Springer, 2022, pp. 149-168. doi: 10.1007/978-3-030-96429-0.

[37] M. Lahby, A. Al-Fuqaha and Y. Maleh, Computational Intelligence Techniques for Green Smart Cities. Springer, 2022, pp. 3-39. doi: 10.1007/978-3-030-96429-0.

[38] I. E. Richardson, The H. 264 Advanced Video Compression Standard, 2nd ed. 2010

[39] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016, pp. 180–184. [Online]. Available: http://www.deeplearningbook.org

[40] “Quality metrics overview,” Quality metrics overview. https://www.ibm.com/docs/en/cloud-paks/cp-data/3.5.0?topic=openscale-quality-metrics-overview (accessed Oct. 2022).

Chapter 3

Dataset Creation

As stated in Section 2.5.6, the MSE-CNN [1] will serve as the research model for this dissertation, with the goal of simplifying the partitioning scheme in the intra mode of VVC while maintaining low quality loss. This model falls within the category of supervised learning algorithms. Therefore, it requires training data to comprehend how to make predictions.

Since a robust database was produced in [1], it was also leveraged to train the model in this work.

This Chapter will provide a full overview of the aforementioned database, an explanation of the process used to produce the labels, and a summary of the contributions made regarding this data.

3.1 Data Analysis

The sequences and video files used to train, assess, and conduct research on the MSE-CNN model was created by the same authors that created the method [1].

The database comprises around 124 GB of data. It consists of 204 video segments and 8000 pictures compiled from a variety of sources [2]- [6]. These files were separated into three subgroups: 6,400 images and 160 sequences for training, 800 images and 22 sequences for validation, and 800 images and 22 sequences for testing. It is worth noting that just 182 of the training and validation sequences, as well as all 8 000 photos, are free to use for non-commercial purposes. Additionally, data modifications were made for storage and compatibility reasons. Videos longer than 10 seconds were shortened to 10 seconds, and NTSC sequences were cropped to a size of 720x480. As previously stated, all data is in its raw format, in the YCbCr 4:2:0 colour space.

Source	**Resolution**	**Num. of images/sequences**	**Total num. of CTUs**	**Total num. of CUs**
Raw Image Dataset (RAISE) [2]	2880×1920	2,000	2,640,000	372,692,745
	2304×1536	2000	1,728,000	242,719,640
	1536×1024	2000	768,000	173,216,005
	768×512	2000	192,000	58,271,751
Facial Video [3]	1920×1080 (1080p)	6	72,960	9,660,712
Consumer Digital Video Library [4]	1920×1080 (1080p)	30	622,080	139,216,238
	640×360 (360p)	59	40,520	20,699,422
Xiph.org [5]	2048×1080 (2K)	18	95,232	21,108,370
	1920×1080 (1080p)	24	471,840	125,995,868
	1280×720 (720p)	4	30,600	15,913,824
	704×576 (4CIF)	5	12,400	5,411,228
	720×486 (NTSC)	7	10,545	4,765,478
	352×288 (CIF)	25	14,368	8,603,450
	352×240 (SIF)	4	688	753,882
Aggregated		8,182	6,699,233	1,199,028,613

Table 3.1: Configuration of CPIV Database [1]

The specifics of the data that can be used for non-commercial purposes are shown in Table 3.1. It is clear to observe from the data that there are about 6.7 million CTUs and over 1 billion CUs among the 8182 images/sequences. This quantity of information is more than sufficient to train the CNN that is being suggested.

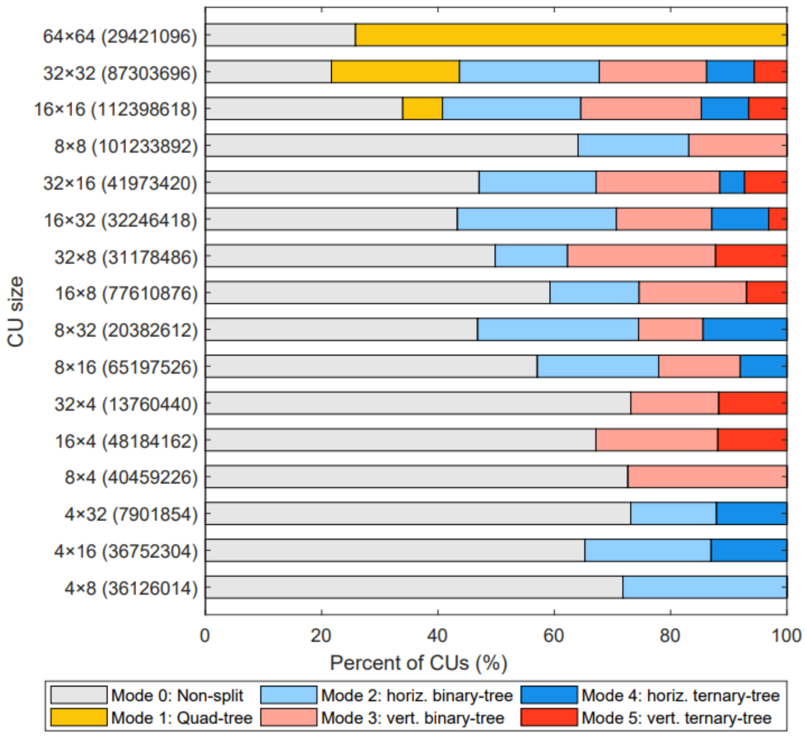


Figure 3.1: CU types proportions using various split options for the luminance channel [1]

Figure 3.1 depicts the proportion of partitions for various CU sizes in the luma channel. Since VTM-7.0, the encoder used in this dissertation (which will be discussed later), forces CTUs to split in QT, the percentages for this case are not shown. This diagram illustrates various facts that should be taken into account:

1. the split modes range from 2 to 6 depending on the CU type;
2. the minimum size of CU is 4x4; this is because for samples with width or height already equal to 4, the split mode used to partition them never affects the axis with a length of 4.;
3. structures with already asymmetric dimensions (16x8, 32x16, etc) are never divided in QT.

All of these statements are consistent with the comments stated in Section 2.4.2. The information provided in this graphic gives an understanding of the proportions of the partitions available for each sample dimension, which is essential for ML applications (balanced or unbalanced dataset), as will be discussed in the next section.

Another relevant Figure can be seen at 3.2. This graphic offers a new perspective on the data by displaying the percentage of split mode types per depth. Similar to the previous image, the first depth is disregarded.

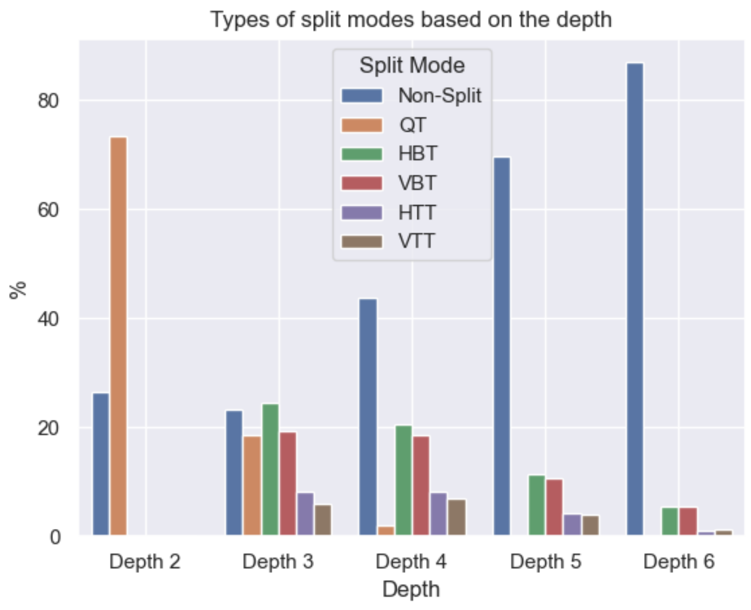


Figure 3.2: Proportions of CU partitions per depth (luma channel) in [2]

Prior to the third depth, the outcomes and conclusions are identical to image 2.4.2. The reason for this is that the CU sizes for the second and third levels are 64x64 and 32x32, respectively. From this graph, the following conclusions may be drawn:

1. in the third level, the most frequent partitions are QT and BT;
2. in the fourth stage, the far more common partition is BT, followed by Non-Split;
3. within the last two depths, the most prevalent partition mode is Non-Split;
4. there are no QT partitions in the last two levels;
5. and the difference in the amount between vertical and horizontal versions of TT and BT is no more than 6%, i.e, the vertical and horizontal versions happen more a less equally.

As previously said, visualising data in this manner enables one to better comprehend the data, future results, and methods to improve the model.

3.2 Labels

Annotated data must be created to train an ML model. Given that a trained network includes inputs and outputs with certain specifications, the data that is supplied to the model must be adapted to meet those requirements. This modification involves certain data processing steps to create information that can directly facilitate network training.

Although the authors of [1] produced a database, they did not provide the labelled data necessary for the MSE-CNN's training. Therefore, it was necessary to design a method for extracting this information from the provided data. In order to process the data, a five-step method was designed, as seen in the picture below.

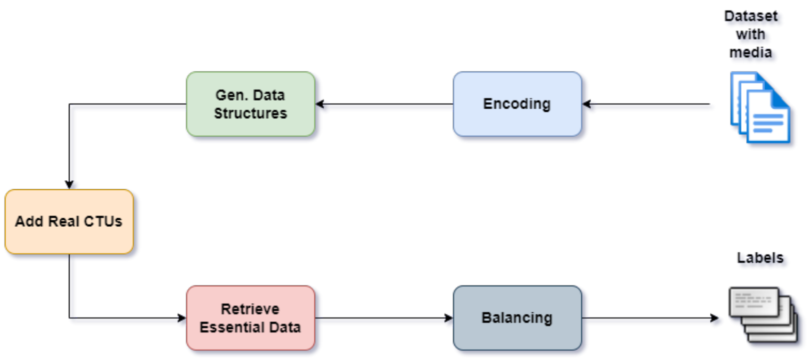


Figure 3.3: Architecture to generate annotated data

The first step is to encode the complete database using the reference software, VTM-7.0, which the authors have made accessible. The All Intra (AI) configuration (with the file enconder intra vtm.cfg) and a QP of 32 were used to encode the data. In the subsequent step, the output from the previous stage is transformed into a more manageable data structure. Following that, actual CTUs are added to the data structure. Afterwards, specific CUs are chosen from all of the previous data. Finally, for data balance purposes, the data is either upsampled or downsampled. The next Section will elaborate on these six processing steps.

3.2.1 Generating Labels

Some statements regarding the MSE-CNN are required to comprehend the rationale behind this pipeline's configuration. The following are the most important considerations to better comprehend this pipeline:

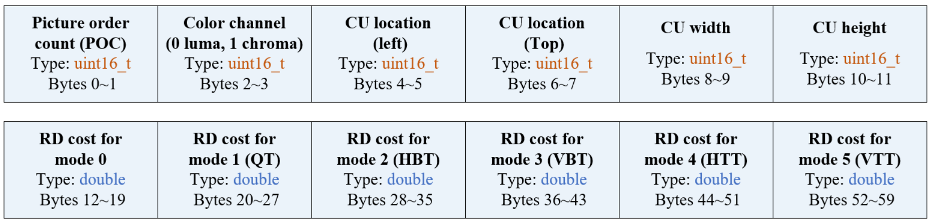
- the model can be seen as an hybrid of an RNN and a CNN, with a collection of CNNs connected to extract features, decide on the optimal split, and send the feature maps to the next depth level;
- the MSE-CNN is composed of six stages;
- the first stage receives a CTU as input, while all of the others receive the preceding stage's feature map output as input;
- while passing through the various stages, the input CTU is partitioned;
- each stage, which corresponds to a specific depth in CTU the partitioning scheme, successively determines which of the six potential split modes is optimal for its input;
- individual types of CUs are utilised to train each model/stage independently.

The next Chapter will go into further detail on this MSE-CNN model.

Now that the essential factors that shaped this label-generating pipeline have been outlined, its constituent components will be addressed next.

Encoding

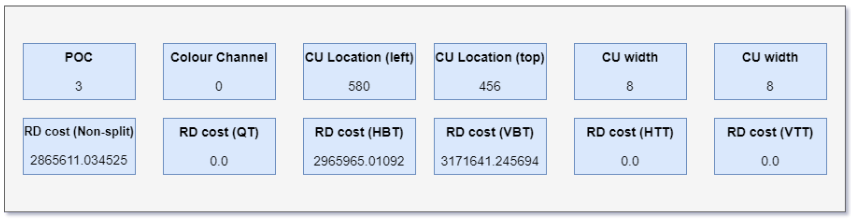
To find the best strategy to split the CTUs in the database supplied in [1], all of the data in it must be encoded. The authors of the paper also made accessible the VVC reference software they used, VTM-7.0, for this purpose. This tool encodes the database and determines the optimum approach for partitioning the sequences within it. In addition to providing a compressed version of the input data, the encoder also produces the RD costs for each CU that makes up the data. A binary file is then created that describes the partitioning scheme for each file in the database. This binary file is composed of a collection of records that include all of the information required to identify and access a CU's optimum split mode. The parameters of a record are depicted in Figure 3.4.



Picture order count (POC) Type: uint16_t Bytes 0~1	**Color channel (0 luma, 1 chroma)** Type: uint16_t Bytes 2~3	**CU location (left)** Type: uint16_t Bytes 4~5	**CU location (Top)** Type: uint16_t Bytes 6~7	**CU width** Type: uint16_t Bytes 8~9	**CU height** Type: uint16_t Bytes 10~11
RD cost for mode 0 Type: double Bytes 12~19	**RD cost for mode 1 (QT)** Type: double Bytes 20~27	**RD cost for mode 2 (HBT)** Type: double Bytes 28~35	**RD cost for mode 3 (VBT)** Type: double Bytes 36~43	**RD cost for mode 4 (HTT)** Type: double Bytes 44~51	**RD cost for mode 5 (VTT)** Type: double Bytes 52~59

Figure 3.4: Record structure [7]

The first two bytes of this structure provide information for the Picture Order Count (POC), which is the location of a given frame within the sequence. The next two bytes represent the channel to which the CU belong. Because the image format is YCbCr, the CUs contents can be in either the luma or chroma channel. The next four bytes reveal the location of the CU in the picture. The first two bytes represent the CU's leftmost horizontal axis position, whereas the second two represent the CU's uppermost vertical axis position. The dimensions are described in the next four bytes. The first two bytes provide width information, whereas the second two contain height information. All of the values mentioned thus far are unsigned 16-bit integers. The following data is stored as an 8-byte double. The remaining 48 bytes reflect the RD cost of each split mode. The first 8 bytes include the RD cost for the non-split split mode, followed by the QT, HBT, VBT, HTT, and VTT. By comparing these six RD costs and selecting the lowest one, the ideal partition for a certain CU is identified.



POC 3	**Colour Channel** 0	**CU Location (left)** 580	**CU Location (top)** 456	**CU width** 8	**CU width** 8
RD cost (Non-split) 2865611.034525	**RD cost (QT)** 0.0	**RD cost (HBT)** 2965965.01092	**RD cost (VBT)** 3171641.245694	**RD cost (HTT)** 0.0	**RD cost (VTT)** 0.0

Figure 3.5: Example of the data in a record. When the RD cost as a value equal to zero, it indicates that the calculation was skipped by the encoder.

Using the VTM-7.0 to apply this step to all 8000 images in the Raw Image Dataset took about 7 weeks. The length of time required for this operation clearly shows the complexity of this coding standard. This method was carried out on a machine equipped with an Intel Xeon E5-2650 v4 @2.20GHz and 64GB of RAM. Since there was an abundance of data among the 8000 images, only those were encoded. Moreover, this data was encoded using a QP of 32.

In summary, the encoding procedure accepts as inputs all sequences in the database as .yuv files and generates .dat files detailing each CU and its RD cost for each partition.

Generating Data Structures

Since the encoded data is a huge number of files holding information about particular CUs without correlation, the data must be structured. To achieve this goal, at this stage the CUs are structured according to how they will be divided at different levels. Figure 3.6 depicts the flow from the CTU to the final partition. The new data structure adheres to this design in order to fit the MSE-CNN's architecture, which takes into account the partition depth.

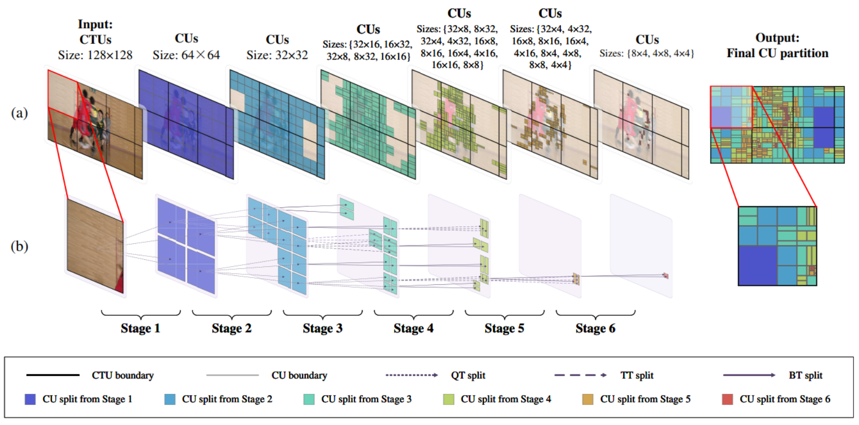


Figure 3.6: Partitioning from a CTU to the last CU [1]

The CUs are first extracted from the .dat files and a simple structure is created for them (image 3.7).

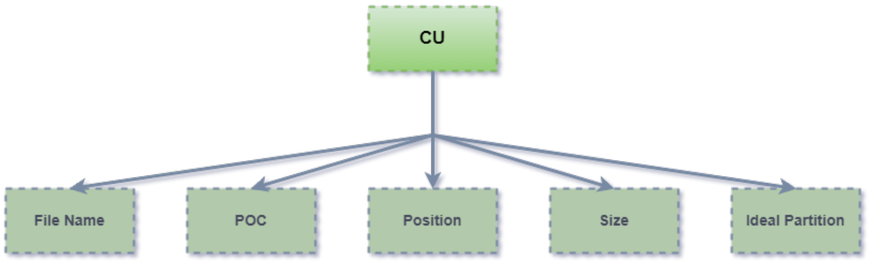


Figure 3.7: CU structure implemented

This structure holds the CU's location, size, file name, POC, and optimum split mode. Because it is necessary to identify a CU among the various files, the name and POC are stored. In terms of the ideal partition, this is computed by selecting the partition with the lowest RD cost. After turning the binary data into structures simpler to manage, the next step is to generate a sequence indicating how a CTU is partitioned from beginning to end. This structure has eight properties, as seen in Figure 3.8. The six fields described as stages 1, 2, 3, 4, 5, and 6 are CUs. They are in accordance with the progression of partitions that a CTU undergoes. Stage 1 is the first depth, which corresponds to a CTU. The sixth stage is the last block to be partitioned. Furthermore, the name of the file containing the CUs and the POC are attributes of this structure that are used to find them.

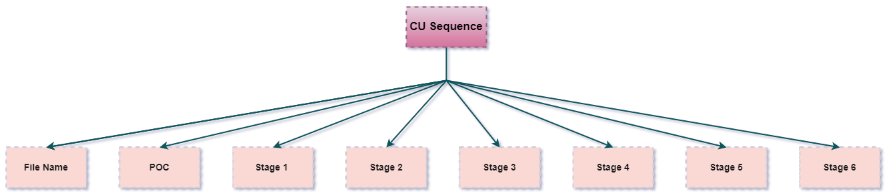


Figure 3.8: Structure implemented with CU sequences

To determine the optimal pattern for CU division, the following steps are taken:

1. select a CTU from the list of CTU structures and save it as part of stage 1;
2. based on the CTU's position, file name, POC, size, and optimal split mode (always QT for CTU), compute the next CU's position and size;

3. using the computed positions and shapes, as well as the file name and POC, a matching CU is sought for;
4. when the CU is identified, its CU structure is stored as part of the second stage;
5. now, much like step 2, the information from this CU will be utilised to find the next one;
6. this procedure is repeated until the CU corresponding to the sixth stage is found.

The outcome of this step, is a collection of files containing a list of sequences.

Adding Raw CTUs

Since the input of the MSE-CNN is the actual pixels of an image that match to a CTU, it is necessary to add this information to the labels. This is accomplished by utilising the superstructure created in the previous phase, which includes all of the information required to obtain the correct CTU from the database. Another property, termed "raw CTU", is added to the sequence structure, which includes the actual CTU splitting structure from the pictures.

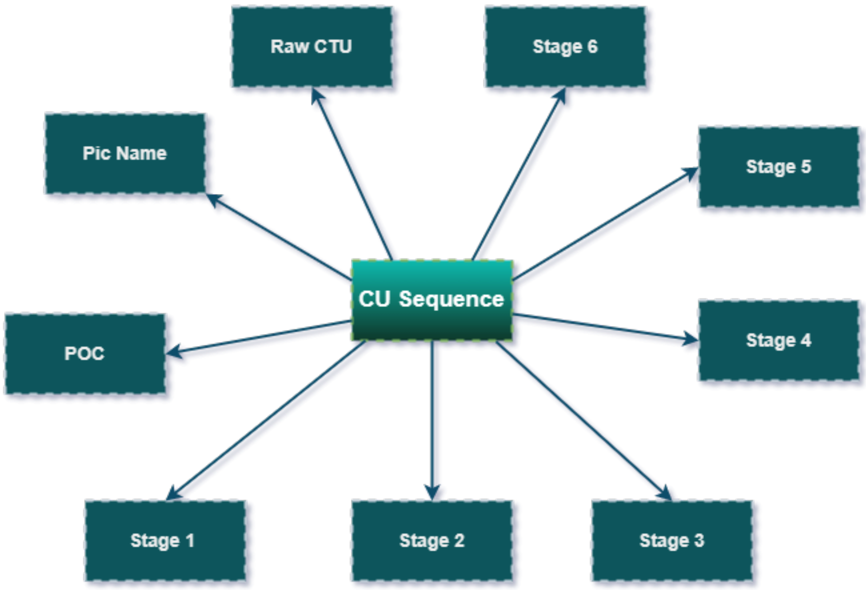


Figure 3.9: Structure of a dataset sample after adding the raw CTUs

The output of this step is a collection of files containing the same sequences as in the previous step, but with the actual CTUs added as a new property. It is worth noting that following this operation, the actual files containing the images are no longer required.

Retrieving Essential Data

In [1], it is proposed that this network is trained stage by stage and with specific CU types. This means that not all sorts of CUs will be required simultaneously throughout training. For example, when the third stage is trained with CUs of size 32x32, the CUs that follow and any other CUs that may emerge in this stage are unnecessary.

To organise the data in this fashion, a search of the data from the previous step is performed and certain changes are done to the data structure containing the sequences. To discover CUs for training a certain stage, one must search through the sequences list using the attribute corresponding to the stage being trained. When this data is discovered, a new structure is created to store it. This new structure contains all prior CUs that led to the current CU used to train the network, the current CU's data, and the raw CTU's data.

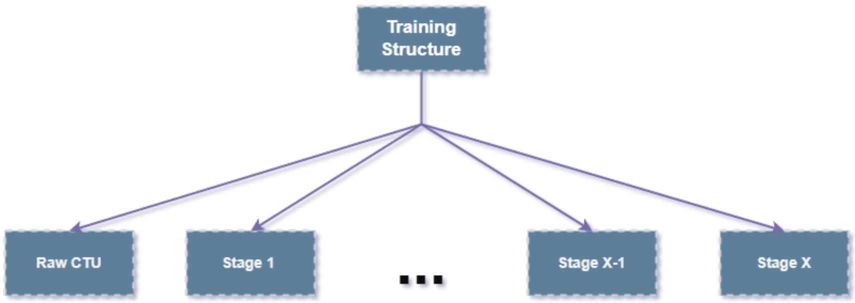
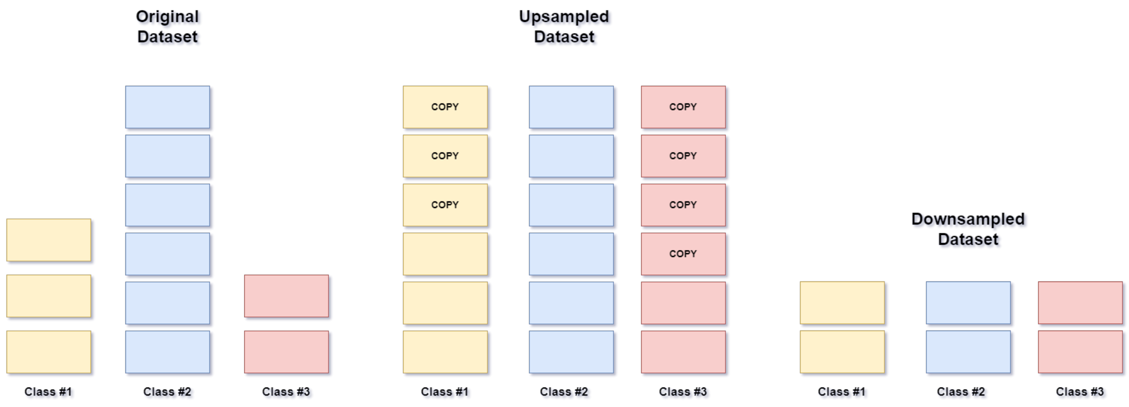


Figure 3.10: Final form of the data that is fed to the MSE-CNN. It is relevant to notice that the number of properties will depend in the stage being trained.

At the end of this phase, a collection of files containing essential data is already ready to be directly used to train a specific stage of the MSE-CNN.

Data balancing

Up to this point, the data in the partitions is not spread out in the same way. As seen in Figure 3.2, QT comprises the bulk of the entire number of split modes for depth 2. Feeding this data to the mode may prevent it from learning how to predict underrepresented partitions. As a consequence, the network predicts each class to be the class with the highest representation in the dataset. To address this difficulty, data balancing is a common technique in DL. The data is either upsampled (oversampled) or downsampled (undersampled) in this procedure (image 3.11). Upsampling consists of matching the number of underrepresented classes to the number of overrepresented classes. This is done by repeating the data in the classes that are not as well-represented to get more of them. Downsampling is similar to upsampling, but rather than increasing the quantity of data for the fewer defined classes, it matches the amount of information for the other categories to the number of samples of the class with the smallest sample size. This is accomplished by deleting items from the more populous classes.



(a) Original Dataset (b) Upsampled Dataset (c) Downsampled Dataset

Figure 3.11: Data balancing

In this last step, the output is a set of files containing the same data as the previous phase, but with an equal quantity of each partition type.

3.3 Implementation remarks

To facilitate fast training, labels must be organised in a manner that supports the usage of batch sizes greater than one. The present data processing method permits not only the gathering of data for feeding the model but also the use of training batches with sizes higher than one. It is feasible to build batches, however the training of this waterfall-shaped model requires batches with the same structure. For instance, while the third stage is being trained, a batch must include CU of the same dimensions; otherwise, mechanisms must be created to separate the various CU sizes inside the model. Implementing this solution would not help train the MSE-CNN because it would require a lot of for loops, which would slow down the training. As a result, before feeding a batch of data to the model, the batch is organised according to the CU shape. In other words, identically sized CU sequences are clustered. Furthermore, using this strategy will result in variable batch sizes.

Since the stages of the MSE-CNN will be trained independently, data is processed for each. Thus, each element of the data stream comprises all the information for the stage being trained and for prior stages to guide the feature maps to that stage. Regarding the data balancing method employed, downsampling was applied.

References

[1] T. Li, M. Xu, R. Tang, Y. Chen, and Q. Xing, "DeepQTMT: A Deep Learning Approach for Fast QTMT-Based CU Partition of Intra-Mode VVC," IEEE Transactions on Image Processing, vol. 30, pp. 5377–5390, 2021, doi: 10.1109/tip.2021.3083447.

[2] D.-T. Dang-Nguyen, C. Pasquini, V. Conotter, and G. Boato, "RAISE: A raw images dataset for digital image forensics," in Proceedings of the 6th ACM Multimedia Systems Conference, 2015, pp. 219–224.

[3] M. Xu, X. Deng, S. Li and Z. Wang, "Region-of-interest based conversational HEVC coding with hierarchical perception model of face," IEEE JSTSP, vol. 8, no. 3, pp. 475–489, Jun. 2014.

[4] CDVL.org, "Consumer digital video library," https://www.cdvl.org, 2019.

[5] Xiph.org, "Xiph.org video test media," https://media.xiph.org/video/derf, 2017.

[6] J. Boyce, K. Suehring, X. Li and V. Seregin, "JVET common test conditions and software reference configurations," in JVET-J1010, San Diego, US, Apr. 2018

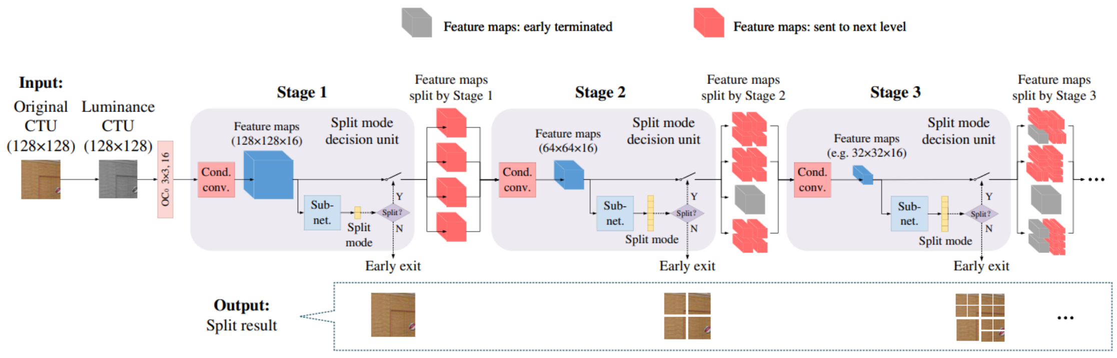
[7] tianyili2017. "CPIV/Data_Format.png at Master · Tianyili2017/CPIV." GitHub, github.com/tianyili2017/CPIV/blob/master/Data_Format.png. Accessed 9 Oct. 2022.

Chapter 4

Deep Learning Model

Multi-Stage Exit Convolutional Neural Network (MSE-CNN) is a DL model that seeks to forecast CUs in a waterfall architecture (top-down manner). This structure takes a CTU as input, extracts features from it, splits the CU into one of at most six possible partitions (Non-split, QT, HBT, VBT, HTT, and VTT), and then sends it to the next stage. This model has CTUs as inputs in the first stage, either in the chroma or luma channel, and feature maps in the subsequent stages. Furthermore, it generates feature maps and a split decision at each level. In the event that one of the models returns the split decision as Non-Split, the partitioning of the CU is ended immediately.

This Chapter will talk about the MSE-CNN model constituents, how it reduces the complexity of the intra mode VVC and the way in which this model's training was executed.



Feature maps: early terminated
Feature maps: sent to next level
Input:
Original CTU (128×128)
Luminance CTU (128×128)
OC₀ 3x3, 16
Stage 1
Feature maps (128×128×16)
Split mode decision unit
Cond. conv.
Sub-net.
Split mode
Split?
Y
N
Early exit
Feature maps split by Stage 1
Stage 2
Feature maps (64×64×16)
Feature maps split by Stage 2
Stage 3
Feature maps (e.g. 32×32×16)
Feature maps split by Stage 3
Output:
Split result

Figure 4.1: MSE-CNN [1]

4.1 MSE-CNN

Initially, this model adds more channels to the input of this network to create more attributes from it. This is accomplished by utilising simple convolutional layers. To extract more characteristics from the data, the information is then passed through a series of convolutional layers. These layers were named Conditional Convolution. At the end, a final layer is employed to determine the optimal manner of partitioning the CU. This layer is a blend of fully connected and convolutional layers.

In this section, it will be discussed the blocks that constitute the MSE-CNN. Namely, the Overlapping Convolution block, the Conditional Convolution and the Sub-Networks.

4.1.1 Overlapping Convolution block

At the beginning of this model, a simple convolutional layer was added to make large and diverse feature maps. After a CTU passes through this layer, the number of channels increases from one to sixteen. Increasing the amount of channels, also increments the number of filters that must be utilised, which results in an higher number of coefficients. All of these factors result in a prediction-capable model that is also more susceptible to overfitting. This is comparable to increasing the number of neurons in an ANN.

This layer is made up of sixteen 3x3 filters. It works with an overlapping convolution with a stride of one and non-zero padding. This means that the input CTU, after passing through the layer, will still have the same 2D dimensions. Also, the activation function used after the convolution operation is a PReLU, which has a trainable parameter. The picture below shows an illustration of this layer.

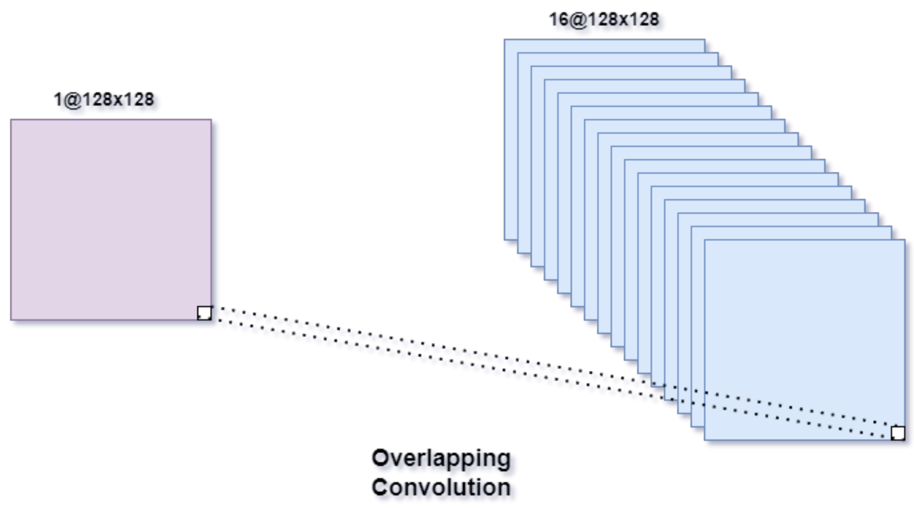


Figure 4.2: Overlapping Convolution block input and output

4.1.2 Conditional Convolution

Because different types of CUs will flow through the various stages of this network, with the largest diversity of CUs travelling through stages 5 and 6, the models must be able to extract more or fewer features based on the dimensions of the CUs to make the right decision. This is done by deepening the MSE-CNN using the Conditional Convolution block. This block is at the beginning of each stage, as seen in Figure 4.1. Since the CU size can change a lot at the same stage, due to them being split in a variety of ways through the depths, it would be best to have a block that can adjust to this. For this, the MSE-CNN uses ResNets [2]. Depending on the type of CU present at the input of a given stage, the number of residual units will be either larger (2) or smaller (0).

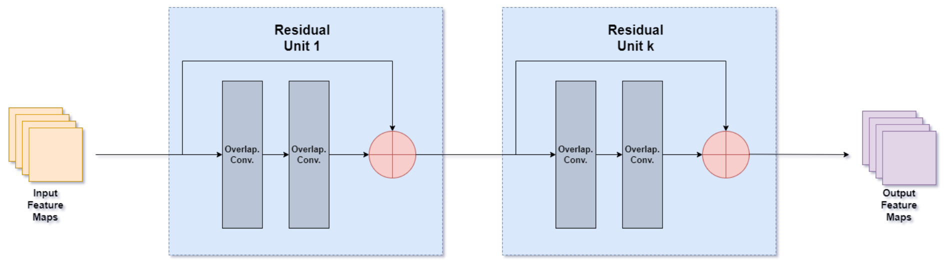


Figure 4.3: MSE-CNN ResNet

The conditional convolution is achieved through the following steps:

1. calculate the minimal dimension of the input CU, selecting the dimension with the least value between height and width;

2. using the current minimum dimension and the minimum dimension of the parent CU, compute the number of residual units that will be used,

$$n_r = \begin{cases} \log_2(\frac{a_p}{a_c}) & 4 \leq a_c \leq 64 \\ 1 & ac = 128 \end{cases} \tag{4.1}$$

3. lastly, run the input through the number of residual units calculated in the preceding step.

This strategy enables each stage to be adaptable and more efficient. In addition, these ResNets employ the same convolutional layers and activation function as the prior mentioned block.

4.1.3 Sub-Network

Following the feature extraction phase of this model, the following step is to determine the optimal partition for the CU; hence, a block is necessary to accomplish this prediction. To solve this issue, a sub-network was developed. This block gets its input from the conditional convolution. It then outputs a one-hot vector with the most likely split modes.

Figure 4.4 demonstrates that, depending on the width and height of the input, different sub-networks are utilised within the stages. Although these sub-networks have distinct layers, their designs are comparable.

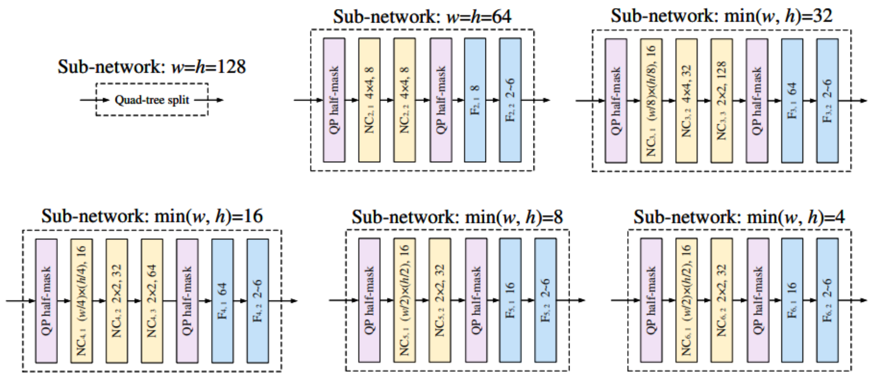


Figure 4.4: Sub-networks of the MSE-CNN [1]

The QP half-mask comprises the initial layer of these networks. QP has a substantial impact on the CU partition. The smaller it is, the higher the number of partitions. In these layers, this information is added by performing a half-mask operation to the network's input, wherein half of the feature maps are multiplied by a normalised QP value. The following equation is used for normalisation,

$$\tilde{q} = \frac{QP}{51} + 0.5 \tag{4.2}$$

First, the QP applied to encode the database is divided by 51 (the maximum QP value in VVC). This operation ensures that the value falls inside the range]0, 1]. Then the previous value is increased by 0.5. This adjusts the normalisation range to]0.5, 1.5].

A sequence of convolutional layers are applied to the data after the QP Half-mask. All convolutional layers have kernels whose width and height are integer powers of 2, like 2x2 and 4x4. In addition, the kernels are non-overlapping, unlike the ones described previously. As shown in Figure 4.4, certain sub-networks are capable of estimating a split mode for different types of CUs. In addition, these layers can find non-overlapping CUs in the final partition, according to [1]. This is accomplished by employing these kernels sequentially. When the initial kernel is applied, the kinds of CUs that can be located correspond to the kernel's size. After this, since the output of the last operation is smaller than its input, it becomes easier to locate larger CUs. This is because the first operation collects information from the input, allowing the next process to cover a larger area, even with smaller kernels.

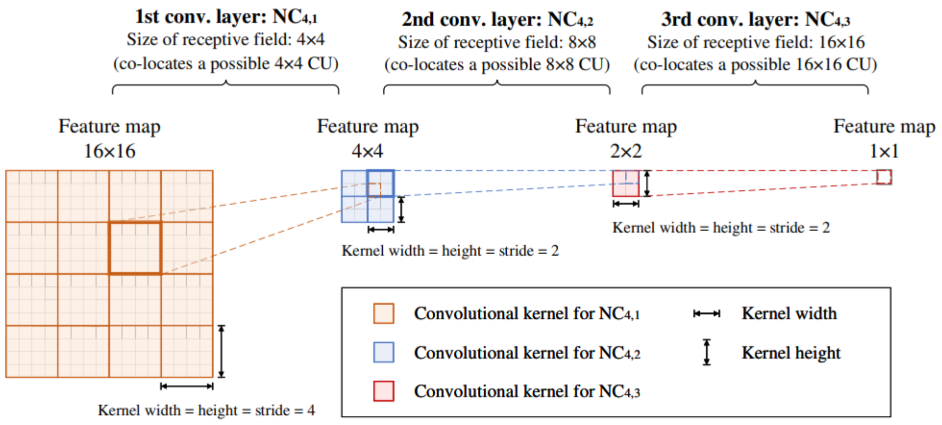

1st conv. layer: NC4,1
Size of receptive field: 4×4
(co-locates a possible 4×4 CU)
2nd conv. layer: NC4,2
Size of receptive field: 8×8
(co-locates a possible 8×8 CU)
3rd conv. layer: NC4,3
Size of receptive field: 16×16
(co-locates a possible 16×16 CU)
Feature map 16×16
Feature map 4×4
Feature map 2×2
Feature map 1×1
Kernel width = height = stride = 4
Kernel width = height = stride = 2
Kernel width = height = stride = 2
Convolutional kernel for NC4,1
Convolutional kernel for NC4,2
Convolutional kernel for NC4,3
Kernel width
Kernel height

Figure 4.5: Convolution in the sub-networks [1]

For example, in Figure 4.5, when a 16x16 input is combined with a 4x4 kernel, the output is a 4 times smaller version of the input. This means that when a 2x2 kernel is applied to the output, it can find 8x8 CUs. Similarly, in the subsequent layer, the input feature map has been shrunk by a factor of 8; hence, a 2x2 kernel corresponds to a 16x16 receptive field, co-locating a 16x16 CU. Similarly to the preceding section, the PReLU was also used as an activation function.

The last part of the sub-networks consists of two fully connected layers. This type of layer is required to get the probabilities for each of the 6 possible partitions. In the first FC layer, there are between 8 and 64 nodes. Also, since predicting the optimal CU partition is a classification problem, the last layer's activation function is a SoftMax [3], while the first FC's activation function is a PReLU. In addition, the input for the sub-network is transposed if its height is greater than its width in order to ensure that the feature map shapes that pass through the sub-network are substantially equivalent. For example, if the size of a CU is given by hxw, where h is the height and w is the width, and $h = 32$ and $w = 8$, then the CU is transposed and becomes 8x32.

4.1.4 Loss function

As mentioned briefly in Section 2.5, a loss function is one of the most fundamental building pieces of a DL-proposed solution [4]. The definition of a loss function has a significant effect on how the model works because it influences how well the network's parameters are tuned for the problem at hand. In addition, the behaviour of a loss function may be utilised to discover problems during the network training and determine the optimal solution for them. In the case of overfitting, for instance, the loss function decreases during training while it rises with validation data. Overall, a loss function not only tells a model how to adapt to the problem but can also be used as a performance evaluation tool.

In the MSE-CNN, the loss function was designed to tackle the fact that the data was not evenly distributed, that different CUs have different numbers of partitions for each stage and that distinct split modes have different RD costs. Even though the problem of unbalanced data was solved in Chapter 3 of this work by using techniques to balance the data, the approach in [1] will still be discussed in this section.

The loss developed for the MSE-CNN is the result of two other functions, as defined in the equation below.

$$L = L_{CE} + \beta L_{RD} \tag{4.3}$$

The first member is a modified cross-entropy (CE) function. In equation 4.4, a conventional CE function takes as inputs the estimated output ($\hat{y}_{n,m}$) of the model and the ground-truth ($y_{n,m}$). Both the actual and predicted outputs are vectors that contain the probabilities of each split mode type. The values of the integer "m" correspond to the position of a particular split mode probability, $m\varepsilon\{0, 1, 2, 3, 4, 5\}$. The number "n" indicates the output utilised inside the mini-batch. The size of the mini-batch utilised in each iteration is denoted by "N" in the formula. Since the predicted values fall inside the range [0, 1], the logarithm returns negative integers. Thus, a minus sign is used to keep the loss function from being negative.

$$L_{CE} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{m\varepsilon Partitions} y_{n,m}\log(\hat{y}_{n,m}) \tag{4.4}$$

This function was modified to handle the imbalanced data problems. In the expression below, an element was added to address this.

$$L_{CEmod} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{m\varepsilon Partitions} (\frac{1}{p_m})^{\alpha} y_{n,m}\log(\hat{y}_{n,m}) \tag{4.5}$$

The ratio of each partition in the whole dataset is stored in the vector parameter p_m. Thus, $\sum_{m\varepsilon Partitions} p_m = 1$. This allows for the application of distinct penalty weights that impacts more the less represented split modes in the dataset. In addition, $\alpha\varepsilon[0, 1]$ is a scalar that may be adjusted to define the significance of penalty weights. $\alpha = 0$ implies that no penalty is imposed, whereas $\alpha = 1$ indicates that the weight of each punishment is proportional to the inverse of p_m. When α is too little, the model may be improperly trained, since it tends to predict just the most common split mode. On the other hand, for higher values of α each penalty weight is proportional to the opposite of p_m. This keeps the model from being poorly trained. However, utilising high values may reduce the accuracy of predictions. As a result, there is a trade-off between the accuracy and reliability of predictions. As previously stated, since the labels created are already balanced, the proportions for each split mode will be the same. Consequently, it is not possible to apply distinct penalties to each partition type since the values in the p_m vector are all the same.

Concerning the second member of the MSE-CNN loss function, this constituent gives the network the ability to also make predictions based on the RD Cost.

$$L_{RD} = \frac{1}{N}\sum_{n=1}^{N}\sum_{m\varepsilon Partitions} \hat{y}_{n,m}\frac{r_{n,m}}{r_{n,min}} - 1 \tag{4.6}$$

In the above equation, the RD costs $r_{n,m}$ uses the same notation for "n" and "m" as the previous equation. Regarding $r_{n,min}$, it is the minimal RD cost for the nth CU among all split modes and $\frac{r_{n,m}}{r_{n,min}} - 1$ is a normalised RD cost. As a relevant note, $r_{n,min}$ is equal to the RD cost of the best partition mode. Consequently, the result of $\hat{y}_{n,m}\frac{r_{n,m}}{r_{n,min}} - 1$ ensures that CU's partitions with greater erroneously predicted probability values ($\hat{y}_{n,m}$) or greater RD cost values ($r_{n,m}$) are more penalised. In $\frac{r_{n,m}}{r_{n,min}} - 1$, the ideal partition has a normalised RD cost of zero, but the other partitions do not. Therefore, the only way for the loss to equal zero is if the probability for all other modes also equals zero. Consequently, the learning algorithm must assign a greater probability to the optimal split mode while reducing the probabilities for the rest. In addition, a parameter, β, is utilised to regulate the impact of this loss on the whole loss function.

4.1.5 Multi-threshold Decision

If the best prediction of the MSE-CNN model was always chosen as the ideal split mode for a given CU, many incorrect CU partitions would be considered best, reducing RD performance. As a result, a multi-threshold decision scheme is employed in order to strike a balance between encoding complexity and RD performance. This method allows the encoder to analyse the most likely partitions rather than just the far more probable one, which eliminates the repetitive testing of CUs in the original RDO procedure. Therefore, certain partitions are omitted, decreasing the complexity of encoding.

In the multi-threshold decision scheme, each stage uses a different decision threshold. These thresholds are used on the output of each stage to choose the most probable partitions. Moreover, like with the outputs, the thresholds range between 0 and 1. The notation for this scheme is $\{\tau_s\}_{s=2}^{6}$. "s" is the stage index, whereas τ is the threshold value. The index begins at stage 2 since stage 1 of the present VTM encoder is deterministic and does not need to be predicted by MSE-CNN. Let $\hat{y}_{n,max}$ denote the highest predicted probability of the output of a stage. The modes that would be picked from the output would be given by $\hat{y}_{n,m} \geq \tau_s \hat{y}_{n,max}$. These would be the partitions that the encoder checks, the rest would be skipped. Moreover, using large values for τ means that the encoder relies on fewer partitions, resulting in the least amount of encoding complexity but the greatest RD performance reduction. While lower values have the reverse effect, encoding complexity is increased and RD deterioration is minimised.

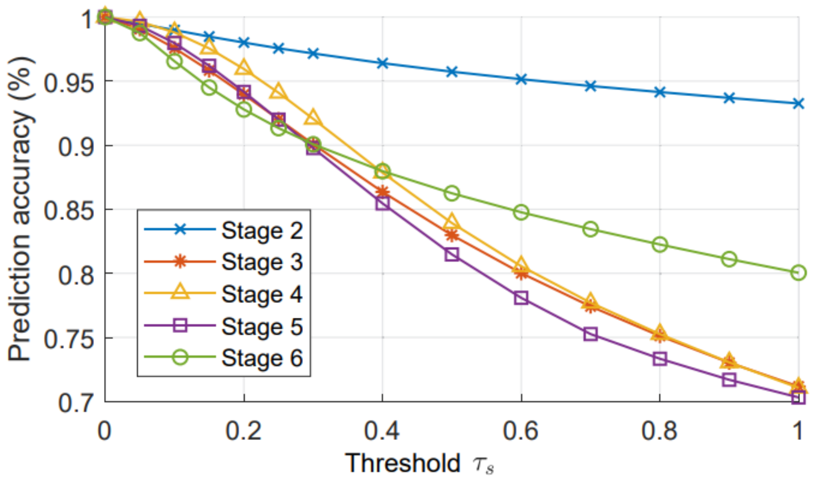


Figure 4.6: Threshold variation and accuracy [1]

In the above Figure, it is possible to see the accuracy variation given different values for the threshold. The following conclusions can be made from this Figure:

- for the lowest threshold, all stages provide maximum accuracy;
- for higher thresholds, Stage 2 always yields the highest prediction accuracy;
- when the thresholds are high, Stage 6 has the second-best prediction accuracy, but when the thresholds are near 0, it performs relatively poorly;
- the change in accuracy is minor for stages 3, 4, and 5.

Depending on the desired performance, better quality or complexity, the authors of the MSE-CNN developed two configurations for these thresholds:

- For less complexity,

$$\frac{1}{5}\sum_{s=2}^{6}\tau_s \geq 0.4, \tau_2 \geq \tau_6 \geq \tau_3 \approx \tau_4 \approx \tau_5 \tag{4.7}$$

- For better quality,

$$\frac{1}{5}\sum_{s=2}^{6}\tau_s < 0.4, \tau_2 \geq \tau_4 \approx \tau_3 \approx \tau_5 \geq \tau_6 \tag{4.8}$$

4.2 Training and Configurations

Proper training is required to achieve the best possible results with a DL-based approach. Ensuring the usage of dependable hardware and software tools improves the process of achieving excellent and rapid outcomes. In addition to the tools, it is crucial to use the appropriate methodologies. A simple training and validation approach was used to ensure that the MSE-CNN model and concepts were operating appropriately. This Section will go over the specifics of this step.

As indicated in one of the preceding chapters, the VTM-7.0 encoder was utilised to encode the data with a QP of 32. A machine with an NVIDIA GeForce RTX 3090, 252 GB of RAM, an Intel(R) Xeon(R) Gold 6336Y @ 2.40GHz, and Ubuntu 20.04.5 LTS as the operating system was utilised for training. Python [5] was the language used to construct the MSE-CNN architecture. As a consequence, several existing tools and frameworks for this language for deep learning and other fields were used. To create the MSE-CNN and training script, Pytorch [6] was utilised. To manage the data and structures, the Pandas [7], Numpy [8] and Python internal libraries were used. Using the Scikit-Learn [9] package, the model was evaluated and metrics were obtained. For tuning hyperparameters, Optuna [10] was utilised. Tensorboard [11] was used to observe and record the model's behaviour throughout training.

Regarding the hyperparameters, 100 epochs and a maximum mini-batch size of 32 [13] were selected. The network was trained using Mini-Batch Gradient Descent, as shown by the terminology used in the previous phrase. This method provides a compromise between speed and computational efficiency [12]. The hyper-parameter α associated with the loss function was set to zero since the labels are already balanced. About beta, the decisions made about it will be talked about in the next Chapter. The learning rate was always first set to a value and then dropped exponentially by 1% every 45 epochs. Using validation data, Optuna was utilised to fine-tune this parameter for each level in order to determine the optimal learning rate for each stage. After model optimization, the test dataset is used to evaluate the model. Moreover, the Adam algorithm [14] is used as the optimizer. Also, when training from scratch, all weight and bias parameters were set randomly with Xavier Initialization [15].

The strategy used to train the MSE-CNN was very similar to the one used in [1]. The first parts of the model to be trained were the first and second stages, in which 64x64 CUs were passed through the second depth. Afterwards, transfer learning was used to pass certain coefficients of the second stage to the third. Then, the third stage was trained with 32x32 CUs flowing through it. After this step, a similar process was done to the following stages, as described in Figure 4.7. It is worth noting that, beginning with stage 4, various CUs forms are at the models' input. This means that these stages were fed different kinds of CUs.

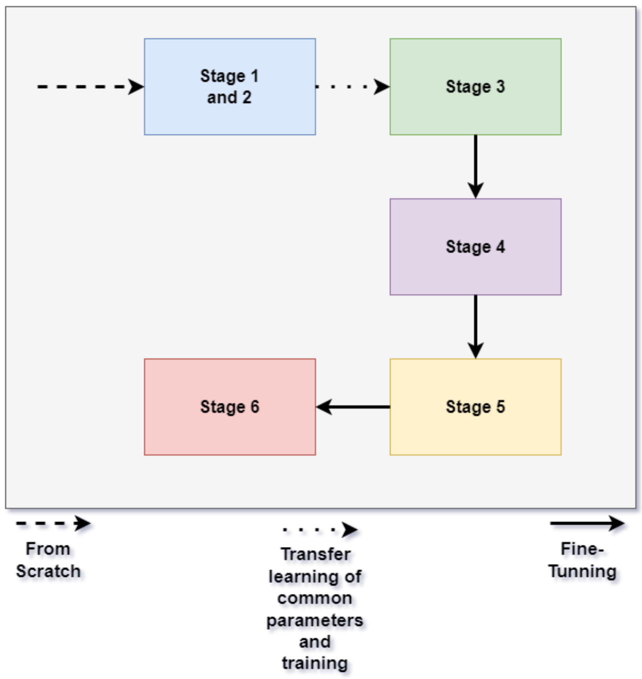


Figure 4.7: Training flow used for training

At the end of training, 6 models were obtained one for each partitioning depth in the luma channel. Although models for the luma and chroma channels could be created for all the shapes of CUs that are possible, rather than just for each depth, as shown in Figure 4.8, only six were trained for the sake of assessing the model behaviour in a simpler and more understandable configuration.

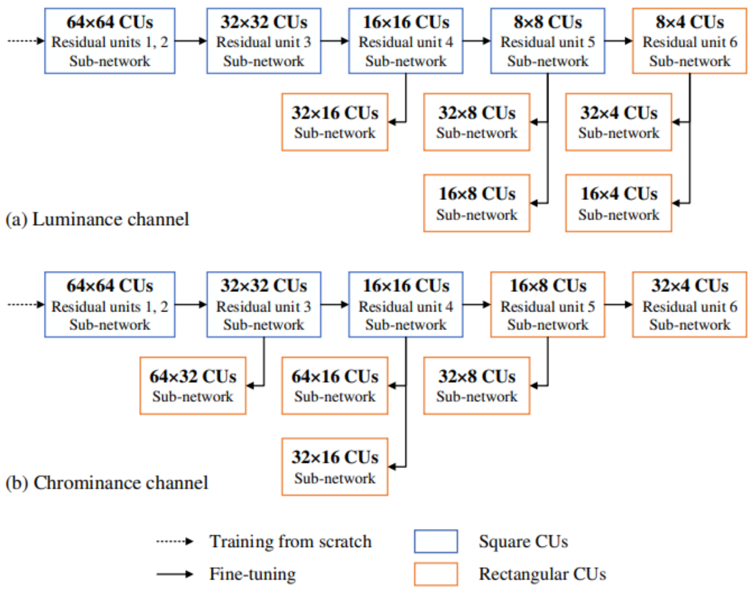


Figure 4.8: Training flow in [1]

As a side note, when one stage is being trained, the others are used in an evaluation mode where the stage's parameters do not change. In addition, the training data consisted of four files with a sequence of pictures. Table 4.1 shows the amount of information contained in these files. This data was divided into three distinct groups: one for training, one for optimising parameters, and one for the model's final validation [16]. 80% of the data was used for training, and the remaining was evenly split for validation and testing [17]. The splitting of the information was done before the data balancing procedure. When training the final model, the training and validation data were combined. The time it took to train each stage was around 24 hours.

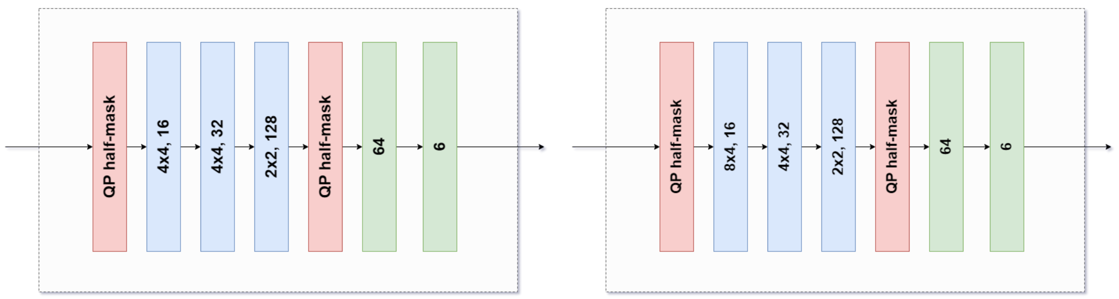
Partitions types	**Stage 2**	**Stage 3**	**Stage 4**	**Stage 5**	**Stage 6**
Non-split	149,676	373,248	128,4208	2,477,349	2,061,549
Quad-tree	389,124	276,070	54,475	0	0
Horizontal Binary tree	0	386,123	590,399	390,137	120,710
Vertical Binary tree	0	302,338	531,970	367,921	120,236
Horizontal Ternary tree	0	126,755	229,741	144,667	22,695
Vertical Ternary tree	0	89,919	193,988	140,361	27,722
All partitions	538,800	1,554,453	6,114,469	4,830,261	2,352,912

Table 4.1: Data quantity in the RAISE_Test files, before the data balancing process

4.3 Code Implementation Remarks

Due to the deterministic nature of the first stage, where CTUs are always partitioned with a QT, it was implemented together with the second stage. If it was done separately, the training for the first two stages would have to be done at the same time. Consequently, two distinct optimisers would need to be employed, which could result in unpredictable training behaviour.

When implementing the sub-networks on code, those that were meant to cater for varying CU sizes were further implemented separately. For example, in the case of the sub-network utilised when the minimum width or height is 32, two variants of the first two layers were built (Figure 4.9). This was done because 64x32 and 32x32 CUs can flow across this block. Because of this, the first two layers were implemented separately from the entire block. Then, they were used in conjunction with the remaining layers based on the dimensions of the input CU. The same procedures were followed for the other types of sub-networks.



(a) Sub-network 1 (b) Sub-network 2

Figure 4.9: 32 minimum axis size sub-networks. In red, the QP half-mask are represented. In blue, the convolutional layers. In which the first group of numbers corresponds to the kernel size and the last number to the number of channels the output has. The blocks with the colour green represent the fully connected layers and the number inside them are the quantity of outputs.

When the network was being trained, some of the RD costs from the input data had very high values. Consequently, the RD loss function value skyrocketed, resulting in extremely huge gradients during training. As a result, the maximum RD cost was hard coded at 10^{10}. This amount is large enough to be more than the best partition's RD cost and small enough to address this issue.

The code developed for this work contains more than 20000 lines of code. It can be accessed in the following link: https://github.com/raulkviana/MSE-CNN-Implementations.

References

[1] T. Li, M. Xu, R. Tang, Y. Chen, and Q. Xing, "DeepQTMT: A Deep Learning Approach for Fast QTMT-Based CU Partition of Intra-Mode VVC," IEEE Transactions on Image Processing, vol. 30, pp. 5377–5390, 2021, doi: 10.1109/tip.2021.3083447.

[2] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[3] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016, pp. 180–184. [Online]. Available: http://www.deeplearningbook.org

[4] Q. Wang, Y. Ma, K. Zhao, and Y. Tian, "A Comprehensive Survey of Loss Functions in Machine Learning," Annals of Data Science, vol. 9, no. 2, pp. 187–212, Apr. 2020, doi: 10.1007/s40745-020-00253-5.

[5] "Welcome to Python.org," Python.org, Oct. 11, 2022. https://www.python.org/ (accessed Oct. 17, 2022).

[6] "PyTorch," PyTorch. https://pytorch.org/ (accessed Oct. 17, 2022).

[7] "pandas - Python Data Analysis Library," pandas - Python Data Analysis Library. https://pandas.pydata.org/ (accessed Oct. 17, 2022).

[8] "NumPy," NumPy. https://numpy.org/ (accessed Oct. 17, 2022).

[9] "scikit-learn: machine learning in Python &mdash; scikit-learn 1.1.2 documentation," scikit-learn: machine learning in Python &mdash; scikit-learn 1.1.2 documentation. https://scikit-learn.org/stable/index.html (accessed Oct. 17, 2022).

[10] "Optuna - A hyperparameter optimization framework," Optuna. https://optuna.org/ (accessed Oct. 17, 2022).

[11] "TensorBoard — TensorFlow," TensorFlow. https://www.tensorflow.org/tensorboard/ (accessed Oct. 17, 2022).

[12] I. C. Education, "What is Gradient Descent?," What is Gradient Descent? — IBM, Oct. 27, 2020. https://www.ibm.com/cloud/learn/gradient-descent (accessed Oct. 21, 2022).

[13] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," arXiv, 2012, doi: 10.48550/ARXIV.1206.5533.

[14] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," arXiv, 2014, doi: 10.48550/ARXIV.1412.6980.

[15] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in AISTATS, vol. 9, 2010, pp. 249–256.

[16] "Validation Set: Another Partition Machine Learning Google Developers," Google Developers. https://developers.google.com/machine-learning/crash-course/validation/another-partition (accessed Oct. 23, 2022).

[17] A. Gholamy , V. Kreinovich, and O. Kosheleva , "Why 70/30 or 80/20 Relation Between Training and Testing Sets: A Pedagogical Explanation," ScholarWorks@UTEP, [Online]. Available: https://scholarworks.utep.edu/cgi/viewcontent.cgi?article=2202&context=cs_techrep

Chapter 5

Results and Discussion

In this Chapter, the approach proposed for training the MSE-CNN and its implementation will be studied based on its outputs.

5.1 Rate-Distortion Loss Function

In order to test the RD loss function during the training of the model, an experiment was performed using solely it. Figure 5.1 depicts the findings of this experiment.

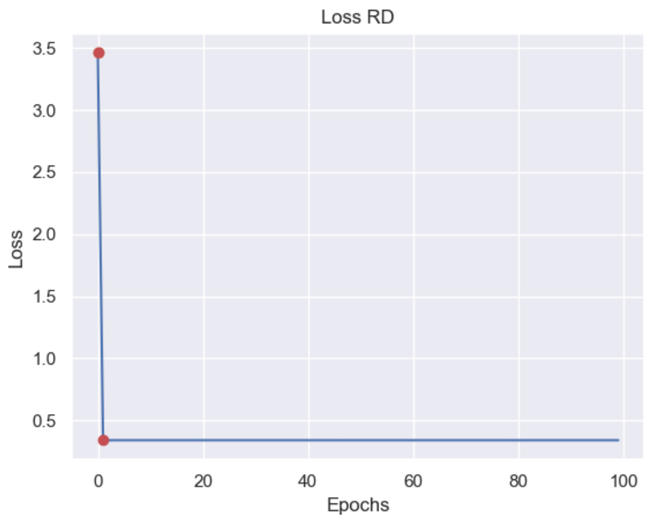


Figure 5.1: Rate-distortion Loss evolution during training. The red dots corresponds to the loss values for the first and second epochs

The trajectory of the loss across the epochs reveals that after the first epoch, the loss immediately converges and continues with a value around 0.34. A variety of training attempts yielded the same result. The confusion matrix for this experiment is shown in the picture below. It demonstrates that the model classified the entire input data as non-split.

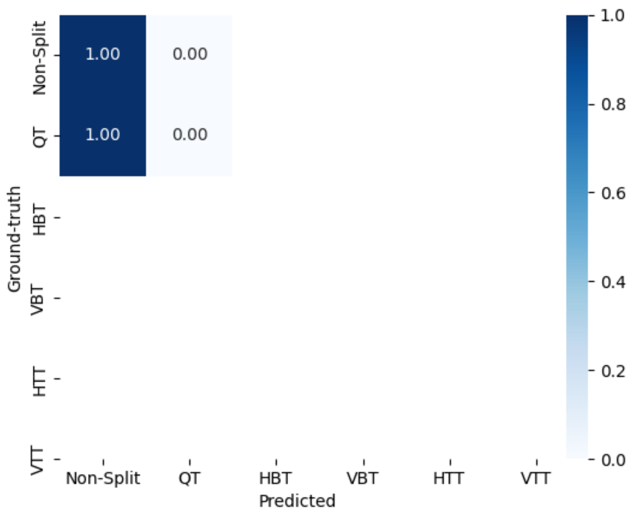


Figure 5.2: Confusion matrix after training the stage 2 with only rate-distortion loss

These findings suggest that the loss function, at least by itself, does not enable the model to understand how to make predictions since it discovered the incorrect way for doing so. This led to the execution of another experiment. This experiment attempted to compare the results of stage 2 by training the network with both losses and subsequently with only the cross-entropy loss. The purpose was to investigate if the RD loss improves the MSE-CNN's prediction abilities. The weighted average of the F1-score across all classes is shown in the table below, along with the weighted averages of the precision and recall. These metrics were obtained from the validation data. By observing the Table, it is clear that the model trained with the cross-entropy loss produces slightly better results. However, due to the insignificance of the difference between the results, another experiment was carried out by trying to find the optimal value of β.

Condition	**Precision**	**Recall**	**F1-score**
Both losses ($\beta = 0.1$)	0.9028	0.9027	0.9027
Only with CE loss	0.9112	0.9111	0.9111

Table 5.1: Weighted average of F1-score, recall and precision for comparing two different strategies for training stage 2

Optuna was used to optimise the β parameter of the loss function in this new investigation. In other words, the model was trained with a variety of β until the best one was discovered. The aim of this optimization was to maximise the F1-score in the validation data. At the end of this process, the optimiser gave β a value of 0.0008. This means that for the F1-score to be higher, β needed to be low. This finding is consistent with earlier ones, demonstrating that this hyperparameter does not lead to an improved MSE-CNN.

Because of the unusual nature of these findings, extra research was carried out. This time, it was hypothesised that because the previous results were for the second stage, and this stage only has two possible partition types, the benefits of the Rate-Distortion Loss function could not be observed. As such, stage 3 was trained with both losses and also trained with just the CE loss.

Condition	**Precision**	**Recall**	**F1-score**
Both losses ($\beta = 0.1$)	0.5465	0.4793	0.4351
Only with CE loss	0.5770	0.5767	0.5624

Table 5.2: Weighted average of F1-score, recall, and precision for comparing two distinct training strategies for stage 3

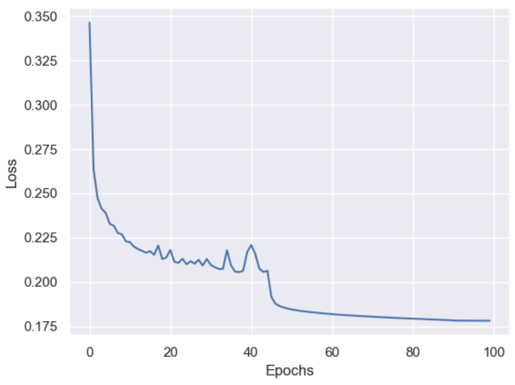
Table 5.2 shows that the same conclusions can be drawn as in Table 5.1. For all of the measures compared, the model that was only trained with the CE loss function outperformed its counterpart.

The preceding conclusions show that the RD loss function is ineffective while training the MSE-CNN. Although it is indicated in [1] that employing this function produces superior results, this was not confirmed in this study.

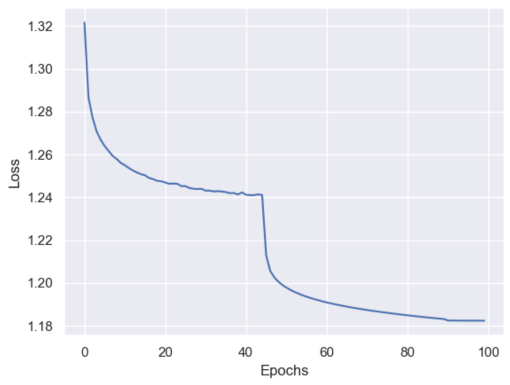
Due to the aforementioned considerations, the Rate-Distortion loss function was not used to obtain the next set of results. That is, β was set to zero.

5.2 Cross-Entropy Loss

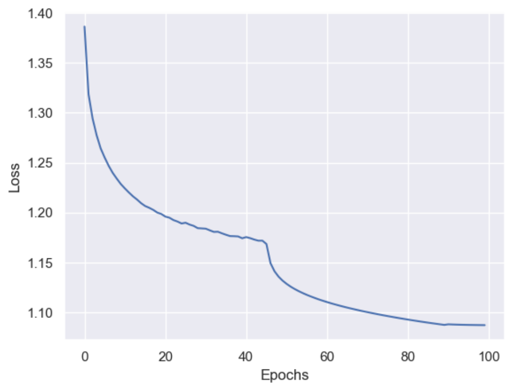
Figure 5.3 depicts the evolution of the loss functions for each stage throughout training. One may observe that all the losses nearly converge, despite the fact that none of them truly do. This leads to the conclusion that the number of epochs should be increased and researched further, even though the potential results would be minimal. Furthermore, based on the loss values, it is feasible to assume that the depths that are the most difficult to train are the third, fourth, and fifth. The fundamental reason for this is the variety of partition types that are possible in those specific depths. Another reason is the variety of CUs that can be at the input of a specific stage, which is greater for stages 4 and 5. As a result, finding the proper coefficients to deduce a generic classification model is challenging for algorithms. The stages with the lowest losses have two things in common: a minimal diversity of CU types at their input and a limited number of split modes to forecast. Consequently, the results obtained are within expectations.



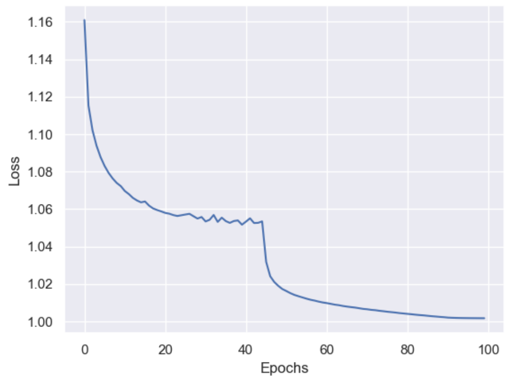
(a) Stage 2



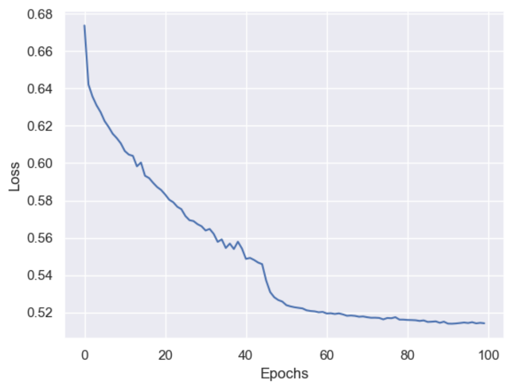
(b) Stage 3



(c) Stage 4



(d) Stage 5



(e) Stage 6

Figure 5.3: Evolution of losses during training for different depths

In order to analyse the presence of overfitting in the results, the loss in the testing data was also evaluated. The occurrence of overfitting was confirmed in the final three stages (Figure 5.4). This is evidenced by a decrease in the loss in the training data and an increase in loss in the testing data. Consequently, it is not advisable to use the model's parameters produced at the end of training. Since this was anticipated, the best coefficients for the model were saved throughout the epochs. The metric used to choose the best model was F1-score. As a result, the presence of overfitting did not affect the ability to obtain the optimal model. Furthermore, given the amount of information acquired in this work, the most logical answer to the occurrence of overfitting would be to train these stages with more information. This would allow the models to tune their coefficients more generically.

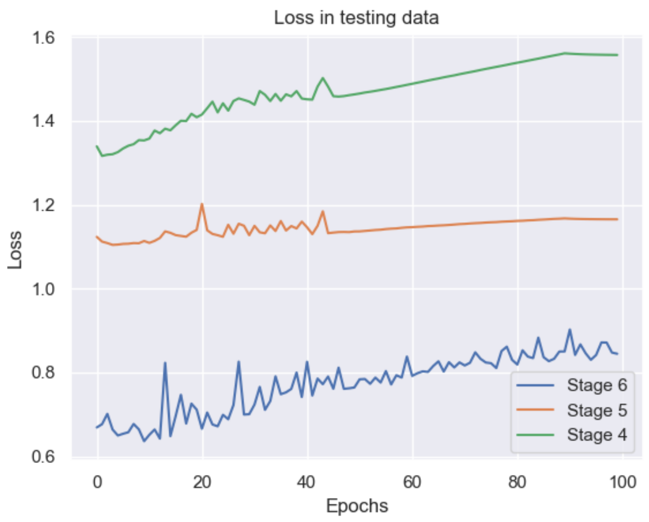


Figure 5.4: Loss in testing data for stages 4, 5 and 6

5.3 Classification Capabilities Assessment

5.3.1 F1-score, Recall and Precision

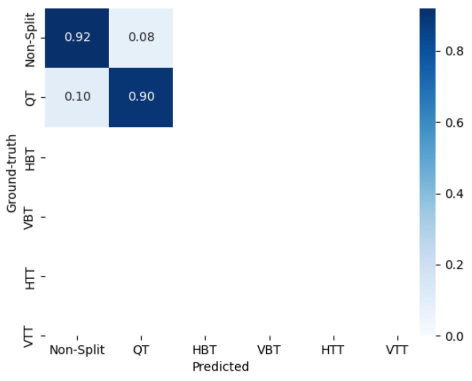
Stage	**F1-score**	**Recall**	**Precision**
Stage 2	0.9111	0.9111	0.9112
Stage 3	0.5624	0.5767	0.5770
Stage 4	0.4406	0.4581	0.4432
Stage 5	0.5143	0.5231	0.5184
Stage 6	0.7282	0.7411	0.7311

Table 5.3: F1-score (weighted average), recall (weighted average) and precision (weighted average) results

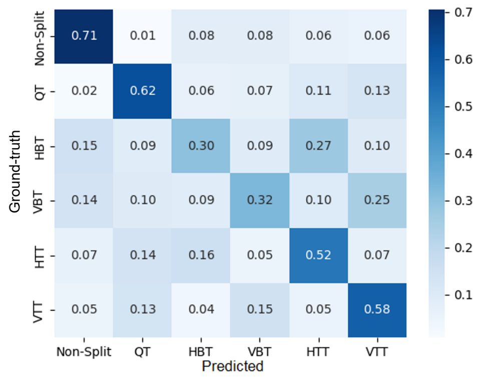
The F1-score, recall, and precision findings for each stage are shown in Table 5.3. The results show that the stages with the best performances in estimating the correct split modes are the second and sixth stages, with an F1-score of 91.11% and 72.82%, respectively. This is in conformity with the results reported about the training losses. The poorest performance was recorded in stage 4, with an F1-score of 44.06%. As discussed in the Section on losses, the stages with the most CU varieties at their inputs and partition types at their outputs have the most problems tuning their parameters to satisfy all of the potential circumstances. Because of this, stages 2 and 6 are superior to the others.

5.3.2 Confusion Matrix

Due to the fact that a confusion matrix facilitates a better understanding of the prediction skills of a multi-classification model, examining it would provide a different perspective on the obtained findings than the previously reported weighted averages for the F1-score, recall, and precision.



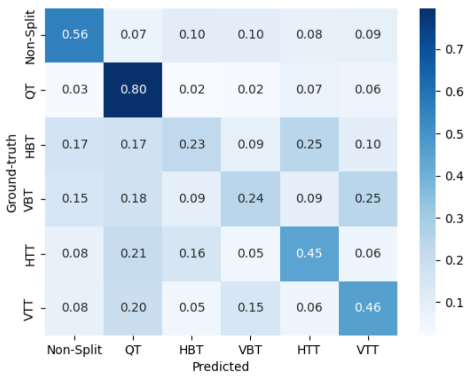
(a) Stage 2



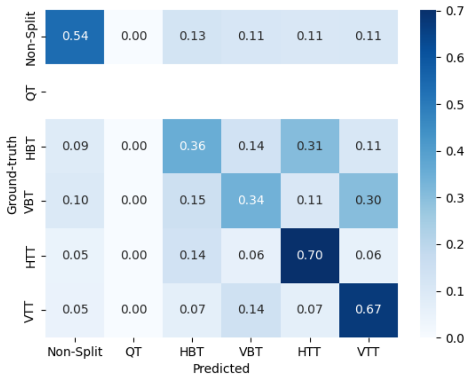
(b) Stage 3

Figure 5.5: Confusion matrix results in with the testing data in stage 2 and 3

Figure 5.5 depicts the confusion matrix for the results obtained from the testing data for phases 2 and 3. The results from stage 2 show that it do not have much trouble finding when the right split mode is a non-split and when it is a QT. Furthermore, the predicting abilities of this stage are more in favour of the non-split class, since 10% of all splits that should have been predicted as QT were predicted as non-split. Concerning stage 3, the model generally computed the right partition, since the diagonal squares of the matrix are darker than the others. It is also conceivable to conclude that the non-split and quad-tree divisions are the easiest to predict, possibly because they inherited coefficients from stage 2. The classes that were the most difficult to forecast were HBT and VBT. These two split modes were frequently confused with their counterparts, HTT and VTT, respectively. Although binary trees were frequently confused with ternary trees, the opposite was not as common. The number of times a VTT or HTT was incorrectly classified as a VBT or HBT (respectively) was the same as the number of times a TT was incorrectly classified as a QT. Furthermore, the fact that the results of ternary tree classification are higher than those of binary tree classification shows that not enough features are extracted for the BT class, with most of the characteristics used overlapping with those of the TT.



(a) Stage 4



(b) Stage 5

Figure 5.6: Confusion matrix results in with the testing data in stages 4 and 5

The conclusions drawn from the stage 4 results, which are depicted in Figure 5.6a, are similar to those of stage 3. Both stages offer great precision for the non-split and QT split modes but are unable to forecast BT partitions correctly. Even if there are similarities between them, stage 4's general forecasting abilities are less than those of stage 3, even though this stage's precision concerning the QT split is better. Given that the QT partition is not present in the results of the fifth stage (as demonstrated in Chapter 3), the model's parameters could be adjusted to better suit the existing classes. Thus, the accuracy for all classes improved. Despite an increase in precision, the capacity to discern a BT from a TT remained unchanged. In addition, Figure 5.6b shows that, although statistically insignificant, the model occasionally predicted classes as QT.

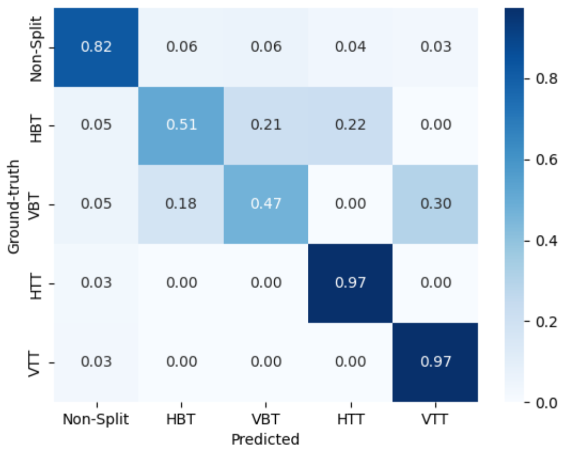


Figure 5.7: Confusion matrix results in with the testing data in stage 6

Similar to stage 5, QTs split modes do not exist. Consequently, this division is absent from Figure 5.7, as the model never predicted or expected an output with this split. As in the previous stages, it was hard to predict the BT class correctly, but this time the situation is different. In this stage, besides the already occurring problems regarding BTs and TTs, there were problems distinguishing between the vertical and horizontal variants of a BT. Despite the model's BT split categorization skills, the precision for the other partition types was very high.

The purpose of MSE-CNN is to submit a group of the most likely partitions to the encoder rather than just the best one. Consequently, using metrics such as those described above is insufficient to comprehend how the model would behave when merged with the encoder. It is required to determine whether the correct split mode can be identified within the group of transmitted split modes. In the following sections, the precision of the multi-threshold scheme will be evaluated.

5.3.3 Multi-thresholding scheme

Table 5.4 shows the percentage of times the model's output, after multi-thresholding, contains the right prediction. The thresholds used to evaluate this scheme were 0.3 and 0.5. For comparison, the top 2, 3, and 4 splits from MSE-CNN's predictions for each stage was taken and examined to see how many times the correct split mode could be found in that group. In addition, the average number of remaining splits, following the multi-thresholding approach for both thresholds, is included.

Stage	**Top 2 acc.**	**Top 3 acc.**	**Top 4 acc.**	**0.3 (MT)**	**0.5 (MT)**	**Avg. number splits (MT 0.3)**	**Avg. number splits (MT 0.5)**
Stage 2	1	1	1	0.9714	0.9503	1.132	1.076
Stage 3	0.8061	0.9114	0.9672	0.8746	0.7791	2.292	1.736
Stage 4	0.6929	0.8462	0.9392	0.7525	0.6408	2.302	1.720
Stage 5	0.7785	0.9209	0.9849	0.8488	0.7423	2.293	1.759
Stage 6	0.9448	0.9962	0.9995	0.8668	0.8094	1.291	1.222

Table 5.4: Multi-thresholding (MT) results and comparisons

Although the findings of the previously mentioned metrics, such as the confusion matrix, suggest rather poor predictive capacities for certain stages, it is possible to conclude that the top 2 best predictions for any stage provide the optimal partition mode at least 69.29% of the time. In other words, the accuracy would be acceptable if the encoder checked the two best splits outputted by the models. However, when it comes to high compression rates, this may not be sufficient. Using a threshold of 0.5 for stages 3, 4, and 5 of the multi-thresholding strategy may not be sufficient to get satisfactory compression results. The findings with a threshold of 0.3 provide a minimum accuracy of 75.25% for all stages, even though this requires the encoder to evaluate more split types. Regarding the average number of splits after the multi-thresholding scheme, it is crucial to mention that for the most conservative criterion, the highest average number of splits left after the thresholding process is 2.292. This means that the encoder would not have to verify an average of six partition modes for every CU, but only two. It is important to note that these results illustrate the dichotomy between complexity and compression rates, as well as the need to modify the threshold for each stage to get the desired outcomes. Furthermore, employing the multi-thresholding scheme with a constant value of 0.3 results in an average accuracy of 86.28% and a reduction in the number of splits that must be analysed by the encoder to 1.862. This result outlines a complexity-quality commitment that leans more towards fast encoding.

5.4 Objective Quality, Complexity Reduction and Bitrate

To understand the impact of this model in the encoder, the model's predictions were fed into the encoder to evaluate the complexity reduction by comparing the overall time to encode the test data with only the VTM-7.0 and with the MSE-CNN together with the encoder. Additionally, the quality loss was compared by analysing the compression achieved with and without the integration of the model. It is also worth noting that the model was run on GPU while the encoder was run on CPU.

Threshold values				
τ_2	τ_3	τ_4	τ_5	τ_6
0.45	0.3	0.25	0.25	0.25

Table 5.5: "Medium" configuration for the MT process [1]. Each τ is correspondent to a specific stage.

In Table 5.6, the results are shown, these were obtained using the dataset for testing. Additionally, the multi-thresholding approach was utilised with values from Table 5.5 (values for a "medium" configuration, as described in [1]).

Metric	**VTM-7.0**	**VTM-7.0+Model**	**Gain**
Bitrate	3810.192 kbps	4069.392 kbps	6.80%
Y-PSNR	35.7927 dB	35.5591 dB	-0.65%
Complexity	1792.88 s	1048.95 s	-41.49%

Table 5.6: Complexity, quality and bitrate regarding VTM-7.0 standlone and VTM-7.0 with MSE-CNN

In the above Table, it is possible to see that the Y-PSNR loss (objective quality measure for the luma channel) was 0.65%, while the complexity reduction decreased by 41.49%. Additionally, a 6.80% increase in bitrate was attained. It is vital to remember that these values compare the results achieved only with the encoder to those obtained with the encoder integrated with the model.

In terms of complexity, it is quite similar to the average found in [1], 44.65%, for a multi-thresholding with the values shown in Table 5.5. The complexity is similar since both the model employed in this study and the one in the original paper feature a maximum sequence of six stages. Moreover, since the complexity result from [1] is an average regarding a large set of data, it is reasonable to say that the values presented here are very similar with those of the mentioned paper. Concerning the quality and bitrate results, because the ones reported in [1] are for the BD-Rate and the Bjontegaard Delta PSNR (BD-PSNR), it is not possible to compare them to the bitrate and Y-PSNR, since the former metrics are produced by comparing the outcomes with different QPs. Nevertheless, the results attained are extremely close to the ones obtained with VTM-7.0 for both the bitrate and the Y-PSNR. Furthermore, while comparing the results of HEVC with the solution provided by this research, Table 5.7, it is obvious to see that the MSE-CNN with the VTM software is still superior to HEVC in terms of quality and bitrate. The bitrate savings and enhanced image quality are still higher for the approach given in this thesis, with more than 10 times bitrate reduction and more than 7 times quality superiority, as shown in the table below. When compared to the predecessor of VVC, these results suggest that using VVC with the MSE-CNN is still more than adequate.

Metric	**VTM-7.0**	**HEVC**	**Gain**
Bitrate	3810.192 kbps	6545.2 kbps	71.78%
Y-PSNR	35.7927 dB	34.0438 dB	-4.89%
Complexity	1792.88 s	3.165 s	-98.23%

Table 5.7: Complexity, quality and bitrate regarding HEVC and VTM-7.0 with the MSE-CNN

References

[1] T. Li, M. Xu, R. Tang, Y. Chen, and Q. Xing, “DeepQTMT: A Deep Learning Approach for Fast QTMT-Based CU Partition of Intra-Mode VVC,” IEEE Transactions on Image Processing, vol. 30, pp. 5377–5390, 2021, doi: 10.1109/tip.2021.3083447.

Chapter 6

Conclusions

6.1 Conclusion

In this study, the MSE-CNN was investigated. It started with the analysis of the data that the model would need in order to train it. Afterwards, a pipeline to generate labels was designed and applied to the database that was made available by the authors of MSE-CNN. Following that, the actual network was built. Many aspects of specifications of this network had to be understood before its implementation. Finally, the model was evaluated using a large number of dependable criteria to demonstrate the reliability of its results. All of this was done with the objective of decreasing the complexity of intra mode VVC while preserving image quality loss. It was possible to achieve an accuracy of 86.28% using the multi-thresholding scheme with a constant threshold of 0.3 and an average number of partition types that have to be examined by the encoder of 1.862 split modes. Moreover, in terms of coding metrics, a complexity reduction of 41.49%, a Y-PSNR loss of 0.65%, and a bitrate gain of 6.80% were observed. In addition to the development of this network, comments on code implementation and a more straightforward training process was suggested.

6.2 Future Work

Regarding future work, this model is amenable to several adjustments, additions, and evaluations. Since this network only uses convolutional and fully connected layers, adding pooling layers could change the results and possibly make them better. Specifically, incorporating these layers into the sub-networks portion of the MSE-CNN. The reason for this is that the dimensionality of the feature maps is reduced in this layer, and the final features are retrieved to determine the appropriate split. According to [1], when small translations are made to the input, the pooling layers make representations that stay mostly the same. When this layer's results are added to those of the convolutional layer, the outputs of this matching could be more accurate. Another intriguing change would be to consider the split mode ratio for each depth when making predictions. By putting the partitions that happen less often into one prediction class and the ones that happen more often into separate ones, the model could figure out more quickly which features are most important for this new smaller group of classes. In addition, grouping the VTT and VBT, as well as the HTT and HBT, into two classes could be an abstraction that results in better forecasting capabilities. In the results Chapter, it was found that stages with less partitions to predict generate better results. After grouping some classes, new models could be designed to be specialised in forecasting these groups. Also, the parameter p_m from the modified cross-entropy loss function, which was not applied in this investigation, could be used in different ways. Instead of using all of the partitions in the dataset to compute its proportions, this computation could be done stage-wise. In other words, the ratio of split modes that are stored in the p_m could be calculated for each stage. Thus, training penalties would be more accurately administered. Moreover, all DL models benefit being trained with more data, for this reason training this model with more data would improve its performance. Besides increasing the amount of information fed to the model, techniques such as K-fold cross-validation [2] could benefit the classification abilities of this model. In addition, hyperparameters such as the number of epochs must be tuned with the learning rate in order to improve the performance of this model. Finally, hard coding the model to eliminate unrealistic partition types such as partitioning a 32x16 CU in QT would be essential.

It would also be important to compare more the results of the model developed in this work with the results of [3]. For instance, evaluate the model with more sequences, using different multi-threshold values and training the model with data encoded with different QPs.

Concerning the developed code, optimizations could be made. These would reduce the amount of time it takes to process the data to get the labels and also speed up the training process. Furthermore, a possible hardware implementation of the VVC with the MSE-CNN could also be a interesting approach to reduce significantly the complexity.

6.3 Contributions

This research made some contributions to the scientific community in the field of video coding. The first big contribution was the actual network implementation, training and evaluation scripts, stages coefficients, data processing code, and numerous other useful functions for interacting with the model and data. This data is significant because it gives a possible interpretation of the model, since it was not made available in [3]. Some information, such as how the batch sizes bigger than one was fed into the model, is missing from the cited study. For this reason, the code developed may allow other researchers to examine the MSE-CNN and compare their results with those presented in this dissertation, the original publication, and other models. Since the database provided by the developers of MSE-CNN had not previously been encoded, the work done in this thesis also made these files available. Aside from this, all files created throughout the data processing are accessible. Files containing all of the structures mentioned in chapter 3 are available for use by the research community. The processed data can mostly be used for approaches that take into account the different splits that happen when an image is partitioned with VVC. In terms of the reference paper [3], this dissertation compares some of its findings and validates what was established in it. Regarding the unbalanced data issues, a different strategy was implemented. While on [3], a modification to the loss function was made to remedy this; in this research, downsampling was applied to the data. Furthermore, a different approach regarding generating the stages for the MSE-CNN was followed, in which 6 stages were created for the luma channel. Although the possible results of this approach are not comparable with the original methodology, it provides a more lightweight and straightforward way of solving this problem. Overall, this thesis offered a technical description of how the MSE-CNN can be implemented and a discussion of the most influencing factors and parameters on its performance.

References

[1] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016, pp. 342. [Online]. Available: http://www.deeplearningbook.org

[2] “sklearn.model_selection.KFold,” scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.model
(accessed Oct. 2022).

[3] T. Li, M. Xu, R. Tang, Y. Chen, and Q. Xing, “DeepQTMT: A Deep Learning Approach for Fast QTMT-Based CU Partition of Intra-Mode VVC,” IEEE Transactions on Image Processing, vol. 30, pp. 5377–5390, 2021, doi: 10.1109/tip.2021.3083447.