Politecnico di Torino

ScuDo
Scuola di Dottorato ~ Doctoral School
WHAT YOU ARE, TAKES YOU FAR

Doctoral Dissertation
Doctoral Program in Electronics and Telecommunication Engineering (34^{th}cycle)

Advances in visual data compression for video and remote-sensing hyper-spectral imaging applications

By

Nicola Prette

******

Supervisor(s):
Prof. Enrico Magli, Supervisor
Prof. Tiziano Bianchi, Co-Supervisor

Doctoral Examination Committee:
Prof. Marco Cagnazzo, Università degli Studi di Padova
Prof. Riccardo Leonardi, Università Degli Studi Di Brescia
Prof. Paolo Bestagini, Politecnico di Milano
Prof. Lorenzo Galleani, Politecnico di Torino
Prof. Maurizio Martina, Politecnico di Torino

Politecnico di Torino
2022

Declaration

 I hereby declare that, the contents and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

Nicola Prette
2022

* This dissertation is presented in partial fulfillment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo).

I would like to dedicate this thesis to my family, in particular my brother Luca and his wife Cristina, and my friends Enrico and Eros, for the support they have given me through these years.

Acknowledgements

I would like to thank my supervisors Enrico Magli, Tiziano Bianchi and the collaborators Diego Valsesia, Martina Cilia, Attilio Fiandrotti and Matteo Naccari for the guidance they have given me. This research was done thanks to the collaboration with RAI—Radiotelevisione Italiana and the support of interdepartmental center SmartData@PoliTO.

Abstract

This thesis presents several works aimed at the advancement of the field of data compression across several types of signals. In particular it deals with the development of new techniques for the compression of video, multi-spectral images and SAR raw data.

In all of these fields the advancement of the compression methodologies is fundamental. In the case of video, the ease of access to high-resolution cameras and the resulting deluge of content in the current social media environment makes the creation of new video-compression techniques necessary to respond to this explosion of data generated. For multi-spectral images and SAR raw data, compression is quite important especially when in a remote-sensing setting, in which the throughput for the communication channel can be quite limited.

The first few chapters describe our work on video coding. Due to the recent advancements achieved with deep learning, especially for image and video processing problems, we decided to develop a deep learning algorithm for video compression. In particular, we developed tools to improve the inter-prediction performance of current video coding standards, like H.265/HEVC. Inter-prediction is a fundamental step in most video compression algorithms, whose aim is to take advantage of the correlation between different frames in a video sequence.

This problem was tackled in two ways: first we developed a filter-generating network capable of predicting a given frame starting from previous ones. We then considered a different approach and concentrated on taking the estimates already provided by the motion-compensation algorithm and enhancing them using previous frames as guide. The designed network is a CNN united with an optical flow network used to align the previous frames to the one that is being enhanced. This method was implemented in the standard H.265/HEVC and we were capable to achieve an average reduction of the Bjøntegaard metric for rate-distortion of -1.69 %.

The following chapters describe our experiments with the low-complexity coding standard for multi-spectral and hyper-spectral images CCSDS 123.0 B-2.

First, we experimented its use for the compression of SAR raw data. SAR is a form of radar, so its raw captures are not actual images, but are instead a grid of complex numbers which describe the echoed signal from the environment in response to an emitted impulse from the sensor. These samples are difficult to compress as they have very limited correlation with each other. Furthermore, in remote sensing settings like earth observation from satellites, the algorithm must be low-complexity, due to hardware limitations.

For this reason, we tried to test the performance achieved by the standard CCSDS 123.0 B-2. This is advantageous because this standard would already be available in modern satellites, and the overall processing architecture would not be burdened by another compression algorithm for SAR data. After few experiments we managed to beat the de-facto standard for the task: block-adaptive quantization. This methodology was adopted in the Horizon 2020 project EO-Alert.

The last work proposed concerns the reduction of optical data sent to ground segments from satellites by skipping the pixels covered by clouds. In remote sensing, clouds are problematic as they represent regions of a captured image which do not provide useful information to the ground segment. We successfully designed multiple techniques to effectively skip the pixels of the image covered by clouds, and to correctly signal the skipped areas to the ground segment. This was achieved by replacing the cloudy regions with dummy values designed to minimize the rate in the file compressed using CCSDS 123.0 and by finding ways to transmit a map of the pixels affected by clouds to the ground. Thanks to this work is possible to significantly reduce the amount of data sent from satellites.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Topic of the thesis

Compression is the search for compact representations of data, for purposes such as the decrease in memory requirements, and faster transmission between source and destination. The advancement and development of new compression techniques is quite important in order to respond to the explosion of content generated every day, both from humans and automatic sensors.

In this thesis three problems are examined: video compression, data-reduction for multi-spectral earth observation images (via cloud screening) and compression for SAR raw data.

1.1.1 Video compression

In our works concerning video compression, we focused on the use of deep learning, after the great successes such techniques achieved in the last few years, particularly in the fields of image and video processing.

Video compression is a very complex task and trying to create a deep learning algorithm capable of beating the performance of the current generation of video coding standards would have been unfeasible. For this reason, we concentrated our efforts in developing deep learning tools to support and improve the performance achieved by already existing video compression standards. It has to be noted that the

tools we developed were tested to work for the current standard H.265/HEVC [1], but, since all the modern video coding standards share the same basic architecture, the techniques we designed are not really dependent on any particular codec, and could be implemented in any of the currently available compression algorithms.

Both of the video coding tools we describe tackle a particular step of the video compression architecture: inter-prediction. Inter-prediction is the algorithm that the codec uses to remove the redundancy present in video due to the high amount of correlation between neighboring frames. This inter-frame correlation is the greatest source for compression in video signals.

Inter-prediction is used to generate an estimate of a frame based on previously compressed ones, so that the encoder has to transmit just the difference between the real frame and the estimate. This difference is called the residual. The prediction is generated by a process called motion-compensation, which is a rudimentary algorithm that produces very artifacted estimates and requires the transmission of side information (the so-called motion-vectors) to function.

Frame prediction

Identifying these limitations of the currently employed inter-prediction algorithms, we designed two alternatives. First, we created a convolutional neural network (CNN [2]) designed to predict the current frame based on previous ones. This approach would be advantageous in multiple ways: first, since the prediction would not be dependent on the use of side information, it could reduce the dimension of the final compressed file. Secondly, by using a more advanced algorithm for prediction, it would be possible to predict more complex forms of motion than the ones modelled by motion-compensation.

Taking inspiration from [3] we created a network capable of generating pixel-adaptive convolutional filters, that are applied on the previous frame to generate an estimate of the current one. The filters were estimated by using the three frame that precede the one that is being predicted.

Motion-compensated frame enhancement

The previous idea was reworked: instead of designing a network which tries to fully generate the predicted frame, we decided that it was more practical to keep the motion-compensation algorithm of the codec and to use a neural network to improve the quality of the generated prediction. We called this network MMCE-Net (Multiframe Motion-Compensation Enhancement Network).

MMCE-Net generates the enhanced version of the motion-compensated frame (MC-frame) by taking as input the motion-compensated frame itself, plus two previous frames of the video sequence. These two previous frames are first registered to the motion-compensated frame using the optical flow network PWC-Net [4]. The MC-Frame plus the registered versions of the two previous frames is fed to a Dn-CNN [5].

This algorithm was implemented inside the video compression algorithm, and we were capable to achieved an improvement in terms of rate-distortion measured using the Bjøntegaard metric [6] of -1.69%.

1.1.2 SAR raw data compression

The following section describes our work in the field of SAR raw data compression. SAR stands for Synthetic Aperture Radar, and since it is a radar architecture, the raw captures are not images, but are instead the echoed signal from the examined environment in response to an impulse emitted by the sensor. SAR raw data takes the form of a grid of complex samples weakly correlated to one another, which makes compression arduous.

In our case, we needed to choose a SAR raw data compression algorithm to be employed on-board of a satellite designed for the Horizon 2020 project EO-Alert [7]. To accommodate the hardware requirements of the satellite, the chosen algorithm needed to have low complexity.

Following the work in [8], we tested the performance of low-complexity standard for multi-spectral and hyper-spectral image compression CCSDS 123.0-B-2 [9] on SAR raw data captures. The use of this algorithm is advantageous because it was already implemented on-board of satellites for the compression of multi-spectral and hyper-spectral images. Its viability for SAR raw data compression would allow to

simplify the processing architecture of the satellite, by using a single compression algorithm to work on multiple forms of data.

After extensive testing we verified that CCSDS 123.0-B-2 can be used for SAR raw data compression, managing to achieve compression performance which surpass the de-facto standard for the task, block adaptive quantization (BAQ [10]). This methodology was then employed in the Horizon 2020 project EO-Alert.

1.1.3 Data reduction for multi-spectral images via cloud screening

The last work described in the thesis, involves again the use of standard CCSDS 123.0-B-2. One great problem in earth observation is that portions of the transmitted data from the satellites are not usable due to the presence of clouds, which obscure the land the satellite is trying to observe.

In this work we developed multiple techniques to allow the transmission only of the valid pixels captured by the satellite, and to effectively skip the pixels affected by clouds. The correct reconstruction on ground is made possible by correctly signaling the skipped pixels. All the techniques documented in this chapter were designed to be compliant with the standard CCSDS 123.0-B-2.

1.2 Thesis organization

The rest of this document is structured in the following way:

- Chapter 2 provides some useful background information about the basics of deep learning, the inner workings of the compression standards H.265/HEVC for video, and CCSDS 123.0-B-2 for multi-spectral images.
- Chapter 3 describes the design of a network for frame prediction for the purpose of video compression.
- Chapter 4 describes the evolution of the work presented in chapter 3: a neural network for the enhancement of motion-compensated frames for the purpose of video compression.

- Chapter 5 describes our work for SAR raw data compression using the CCSDS standard.
- Chapter 6 describes the work on cloud-screening for data reduction in a remote sensing setting, again inside the framework of the CCSDS standard.

1.3 Publications

These are the published paper throughout the duration of the PhD:

1. Nicola Prette, Enrico Magli, and Tiziano Bianchi. Using CCSDS image compression standard for SAR raw data compression in the H2020 EO-Alert project. In *European Workshop on On-Board Data Processing (OBDP2019)*, 2019.
2. Martina Cilia, Nicola Prette, Enrico Magli, Bernhard Sang, and Stefano Pieraccini. Onboard data reduction for multispectral and hyperspectral images via cloud screening. In *IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium*, pages 6230-6233, 2020.
3. Nicola Prette, Diego Valsesia, and Tiziano Bianchi. Deep multiframe enhancement for motion prediction in video compression. In *28th IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*, pages 1-6, 2021.
4. Nicola Prette, Diego Valsesia, Tiziano Bianchi, Enrico Magli, Matteo Naccari, and Attilio Fiandrotti. Deep motion-compensation enhancement in video compression. *Electronics Letters*, 58, 04 2022.

Chapter 2

Background

2.1 Video Compression and the H.265/HEVC coding standard

This chapter provides a basic description of the video coding standard H.265/HEVC. This will be useful since this standard has a central role in much of our work on video compression. A general overview can be found in [1], while for a more in-depth look the book [11] is recommended. The explantion provided in this chapter are based on these two sources.

H.265/HEVC was designed jointly by the ITU-T Video Coding Experts Group (VCEG) and by the ISO/IEC Moving Pictures Expert Group (MPEG). Together they form the Joint Collaborative Team on Video Coding (JCT-VC). H.265/HEVC is the successor of H.264/MPEG-4 AVC [12], and it has been followed by H.266/MPEG-4 VVC [13].

2.1.1 Overview

Despite the years of iteration on the design, in broad strokes H.265/HEVC shares the same kind of architecture as H.261 [14] (the first of the video coding standards developed by ITU-T) and all its successors. It can be described as a sort of Differential Pulse-Code Modulation (DPCM), in the sense that the compressed samples are generated by quantizing and encoding a difference between the original sample

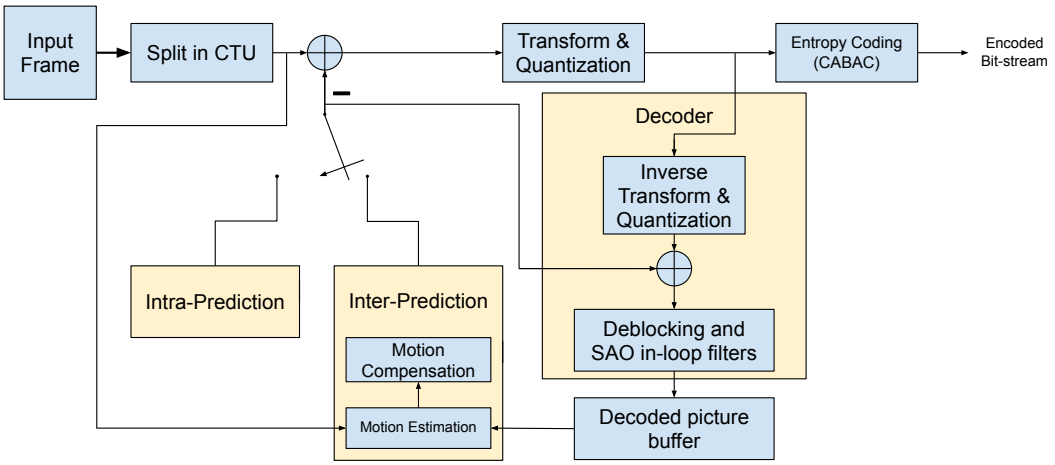

Fig. 2.1 Simplified diagram of the architecture of video compression standard H.265/HEVC

and an estimate computed by the algorithm. This estimate is constructed based on previous samples.

It is an hybrid architecture, because it employs together two different forms of prediction: intra-prediction and inter-prediction. Intra-prediction generates the estimated sample by taking advantage of the correlation present between samples inside the same frame, while inter-prediction uses the correlation between different frames.

A diagram displaying the various components of the architecture of H.265/HEVC can be seen in figure 2.1.

Each input frame is split in square sections called Coding Tree Units (CTU) which are processed sequentially. The encoding of the first frame of the image is done purely through intra-prediction, while for the following frames each CTU can be coded using either intra- or inter-prediction (but mostly the latter of the two).

Inter-prediction is achieved in two steps. First, the motion between a reference picture, chosen from a collection of previously encoded frames, and the target one is extracted. This motion information is then used to generate a prediction through a process called motion-compensation.

At this point, the predicted CTU is subtracted from the input one to generate a residual signal. A linear spatial transform is applied to this residual, and then the obtained coefficients are quantized and entropy coded. This compressed stream is sent to the decoder together with the prediction information.

In order to make the decoding of the file possible, the encoder needs to generate every prediction with information that would be available at the decoder while processing. For this reason, the structure of the decoder is replicated inside the encoder: an inverse transform is applied to the residual which is then summed to the predicted signal. This first reconstruction is then enhanced using the de-blocking and SAO filters to obtain the final reconstructed frame. These reconstructed frames go to form the Decoded Picture Buffer, which is used for the purpose of inter-prediction.

In the following section we will provide a broad description of some of the main components of this architecture, posing most of the emphasis on the inter-prediction segment, which will be the focus of our work on video compression.

2.1.2 Components of the architecture

Coding units and other partitions of the frame

As it was mentioned in the overview, inside the architecture of H.265/HEVC the frame is split in square block sections called Coding Tree Units (CTU). This is a more flexible version of what in the previous iterations of the standard were called macroblocks. The CTU is composed by three Coding Tree Blocks (CTB), each associated with one band (one luma channel and two chroma channels). This naming convention is kept for all the other ways to partition the frame: unit refers to a partition across all channels while block refers to its corresponding partition over a single channel. The luma CTB can be created in three possible sizes in pixels: 64×64, 32×32, 16×16.

The CTU is further partitioned in what are called Coding Units (CU) using a quad-tree structure (see Fig. 2.2). The smallest CU that it is possible to generate has size 8×8.

During the prediction stage of the algorithm, the CU is further partitioned in Prediction Units (PU). The rules for the partition in PUs change depending on whether inter- or intra- prediction is used. For inter-prediction the quad-tree structure is not followed anymore, and the PU can also assume rectangular shapes of different sizes.

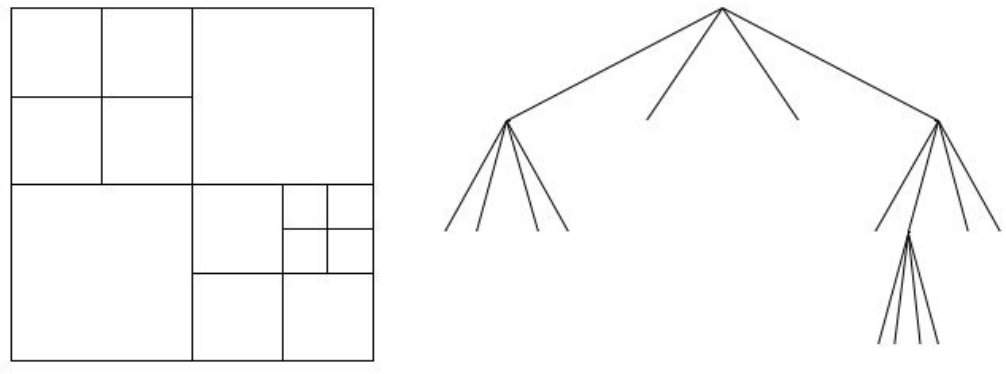

Fig. 2.2 Quad-tree structure used to partition a CTU into multiple CU and to partition a CU into multiple TU.

Similarly during the transform coding stage the CU is partitioned again to form the Transform Units (TU), again following the quad-tree structure down to a size of 4×4.

Intrapicture Prediction

Intra-prediction is used to predict the content of a PU by using neighboring PUs in the same frame. There are three forms of prediction:

- Directional prediction: the predicted PU is generated by taking pixels from already decoded PUs which touch its boundaries, and their value is extrapolated following one between 33 possible directions of prediction (Fig. 2.3).
- DC prediction: the full PU is filled with a constant value which is the average of the boundary samples.
- Planar prediction: the PU generated is a linear color gradient through the surface, whose characteristics are determined by the samples on the boundaries.

Interpicture Prediction

Inter-prediction generates the content of the PUs using information contained in previously encoded frames. In H.265/HEVC, motion between frames is modelled as

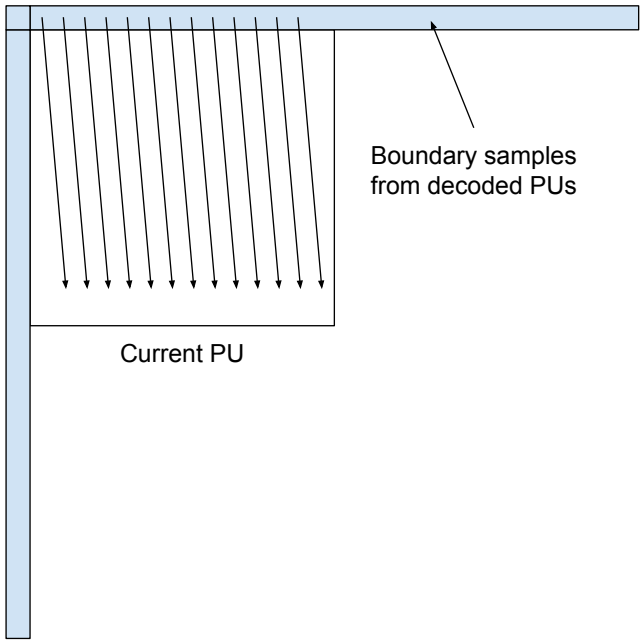

Fig. 2.3 Diagram for directional intra-prediction: the boundary pixels are extended to generate the predicted PU. One of 33 directions can be chosen for the extension.

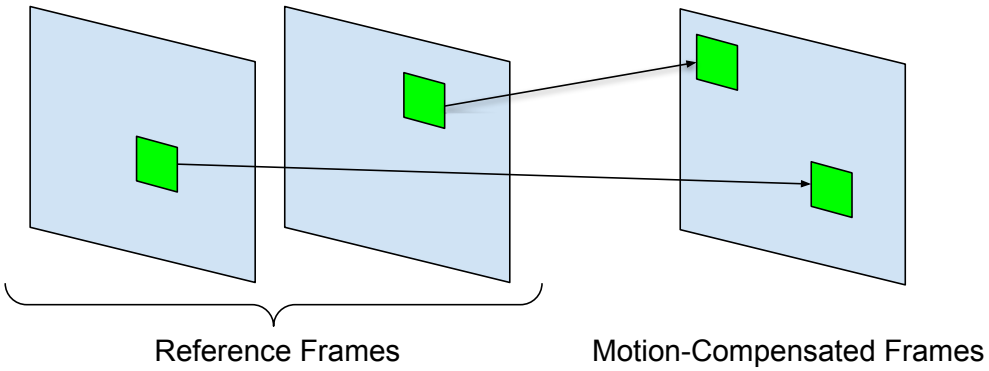

Fig. 2.4 Motion-Compensation mechanism for inter-prediction: the predicted frame is obtained by assembling blocks from pictures in the decoded picture buffer. These blocks are moved to the right position in the predicted frame by using the motion vectors

a translation of blocks through time (Fig. 2.4). Motion is assumed homogeneous for the blocks, so it can be described succinctly as an offset which is applied to all the pixels of the block. These offsets are the motion vectors.

The motion vectors are generated through a process called motion-estimation. Given a PU inside the frame, the motion-estimation algorithm iteratively looks for blocks inside other temporally-neighboring frames which correlate with the region corresponding to the PU in the original image. The algorithm used for the motion-estimation is not defined in the standard, and the choice is left to the discretion of who implements the encoder. The temporally neighboring frames are found in the decoded pictures buffer and are called reference frames.

Whenever a match is found, the offset between the PU that is being predicted and the matching block is the estimated motion vector. In order to obtain more accurate predicted frames, the motion vectors are estimated with an accuracy equal to one quarter of a pixel. This is achieved by up-scaling both the reference frames and the original by a factor of four (both vertically and horizontally) during the motion estimation process.

There are two kinds of inter-prediction: uni-prediction, and bi-prediction. In uni-prediction all the reference frames used to generate the PU precede the current frame in display order. Instead, in bi-prediction both future and past frames are used. Furthermore, in bi-prediction two reference picture lists are used to generate two predictions. These two predicted blocks are then merged together, either by

averaging or by using a weighted sum. The frames coded using uni-prediction are referred to as P-frames, while the ones obtained through bi-prediction are the B-frames.

In order to make possible the generation of the prediction blocks at the decoder side, the motion vectors need to be transmitted as side information. Since the motion of a prediction block is correlated with the motion estimated for other neighboring blocks it is possible to compress motion information. The compression of the motion information is done using an algorithm called advanced motion vector prediction (AMVP), which is an improved version of similar mechanisms present in previous versions of the standard. In this mechanism, a prediction of a motion vector is generated (called motion vector predictor or MVP) and what is transmitted is the difference between the real motion vector and the MVP. This residual between the two is the motion vector difference (MVD).

In cases where the same motion information is shared by (spatially or temporally) contiguous prediction blocks, H.265/HEVC does not send the motion vector to the decoder; what is sent instead is an index which points to the contiguous block which has the same motion-vector information. This is called merged mode as it creates regions composed of different blocks which share the same motion information.

Transform, and Quantization

The generated residual then undergoes a process called transform coding. The CU is partitioned in multiple TU using a quad-tree structure. The biggest size allowed for the TU is 32×32 while the smallest is 4×4 pixels. The transformed coefficients of the block are computed by using two 1-D transforms, one vertical and the other horizontal. The transform coefficients are integer approximations of the coefficients of a discrete cosine transform (DCT), except when the TU has dimensions 4×4, in this case an approximated version of the discrete sine transform is used.

The transformed coefficients are then quantized. This is done by dividing the samples by a value called quantization step, and by rounding it up to the nearest integer. The quantization step size $\Delta_Q(QP)$, is proportional to the quantization parameter (QP). The QP can be chosen with values between 0 and 51, and the relationship between the quantization step and the QP is the following:

$$\Delta_Q(QP) = (2^{1/6})^{QP-4} \tag{2.1}$$

Using this equation, when the QP is equal to 4 the quantization step is equal to 1, and there is a doubling in size of the quantization step for an increase of the QP by a value of 6.

Both transform coding and quantization are skipped in the so-called lossless mode.

Entropy Coding

H.265/HEVC uses a entropy coding algorithm called CABAC (context-based adaptive binary arithmetic encoder) [15] an algorithm originally developed for H.264/MPEG-4 AVC. In broad terms, this algorithm is a arithmetic coder connected to a probability estimator to adapt to changes in the distribution of the data to compress through time.

In-Loop Filters

The reconstructed frames are affected by various kinds of artifacts, like noticeable square borders due to the partition of the frame in blocks during compression. To mitigate these problems some enhancing algorithms are applied to the reconstructed frames, which are called in-loop filters. There are two enhancing processes which are applied to the reconstructed frames: the deblocking filter (DBF) and the sample adaptive offset (SAO).

DBF is devoted to the reduction of the discontinuities generated by the compression algorithm (block artifacts), which are generated by the partition of the frame in CUs, PUs and TUs. It consists of a blurring filter which is applied selectively across the reconstructed frame. The block artifacts are especially noticeable in smooth areas of the original frame, so it is in these areas that the deblocking filter is applied. In sections where the original frame is not smooth the artifacts are less noticeable, and the application of the de-blocking filter is avoided. Furthermore, the use of the DBF in such areas would filter away legitimate information from the original samples and make the final frame blurry.

SAO is a general enhancing filter which reduces the presence of other imperfections, mainly ringing artifacts deriving from the use of larger sizes for the transform. There are two types of SAO:

- Edge Offset (EO): Which tries to limit the effect of the Gibbs phenomenon (ringing artifacts). This is achieved by adding offsets to the interested samples, which compensate for the peak and valleys attributed to the detected artifact. To detect the rings and decide which offset to apply to the sample, a classification algorithm, which uses the neighboring pixels and the original frame as input, is applied. The results of these classifications have to be sent to the decoder as side information to allow for the correct reconstruction of the frame.
- Band Offset (BO): In this algorithm the offset is provided to the sample is based on if it falls inside a determined range of values (band). The offset is thus determined only by the value of the pixel itself, without examining its neighbors. The offset to be applied for each band are determined by the encoder to further minimize the distortion on the reconstructed image without adding much side information.

2.2 Low-complexity image compression for remote sensing applications

2.2.1 Introduction

Hyper-spectral and multi-spectral images are optical images characterized by a greater number of bands, compared to the usual three (red, green and blue). This makes it possible to capture the electromagnetic spectrum for each pixel over a large span of wavelengths and with an high resolution. In certain applications the captured wavelengths can reach thousands [16].

This detailed representation of the electromagnetic spectrum is useful for several scientific applications and thus hyper-spectral and multi-spectral sensors are often present in satellites used for Earth Observation. Hyper-spectral images can be useful for purposes such as agriculture, monitoring and alerting for calamities of various

nature, mining and many others [17, 18, 7]. For this reason new architectures, enhancing algorithms and applications continue to be developed for this field [19].

The great number of channels makes this kind of images very large in terms of memory occupation. This is a big problem since the available capacity of the channel between the satellite and the ground segment is limited, which makes necessary to compress the data. However, the hardware available on-board of satellites is limited in terms of performance so it is necessary to design compression techniques of limited complexity. Another caveat is that in scientific applications it is necessary to capture data with no distortion, or where distortion is guaranteed to be lower than a specified upper bound, so only lossless and near-lossless techniques can be used.

The *Consultative Committee for Space Data Systems* (CCSDS) is a committee devoted to the design and diffusion of standards useful for applications in space-flight communications, and also deals with the design of standards for compression of hyper-spectral and multi-spectral images. The latest iteration regarding this topic is called *"Low-Complexity Lossless and Near-Lossless Multispectral and Hyperspectral Image Compression* [20]" (CCSDS 123.0-B-2). An in-depth overview of this standard, which was used as basis for this chapter, can be found in [9].

2.2.2 The CCSDS 123.0-B-2 Standard

Overview

CCSDS 123.0-B-2 is a compression standard based on the principle of DPCM (Differential Pulse-Code Modulation), which means that it compresses the images by first generating estimates of its samples (using the correlation between samples inside the signal), and then by quantizing and entropy coding the residual. It is a backward compatible extension of CCSDS 123.0-B-1 (where the last digit indicates the number of Issue of the standard). This new issue introduces, among other things, the possibility of performing near-lossless compression (lossy compression with an upper bound on distortion for every pixel). This is achieved by adding a quantization step to be applied in the prediction loop, since the quantization of the input image before prediction would be sub-optimal [21]. It also introduces a new mode of entropy coding denominated *hybrid* coding, which is capable of reducing the bit-rate

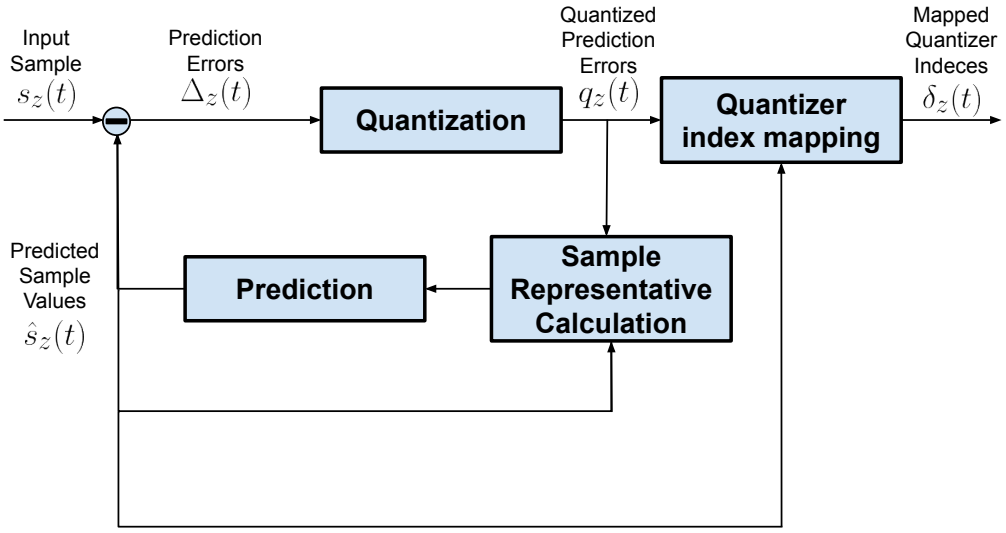

Fig. 2.5 Structure of the CCSDS 123.0-B-2 predictor

to values smaller than 1 bit-per-pixel by encoding multiple input pixels at the same time.

An overview of its architecture is displayed in figure 2.5: an image is fed to a prediction loop which generates a prediction error (which is quantized in the near-lossless configuration). This prediction error is then sent to an entropy coder. Three entropy coder algorithms can be selected: Block-Adaptive Coder, Sample-Adaptive Coder, and the Hybrid-Coder.

Predictor Stage

The role of this component is to predict the value of a pixel conditioned on previously encoded input pixels. The assumption is that in optical images the values in the same area of the image are correlated with each other, so the values of previously encoded pixels can be used to predict the future ones.

This prediction is then subtracted from the input value to generate a prediction error (also called residual):

$$\Delta_z(t) = s_z(t) - \hat{s}_z(t) \tag{2.2}$$

where $s_z(t)$ is input sample (z is the index of the band that is being encoded and t is the index which designates a pixel inside the image navigated in raster scan order), $\hat{s}_z(t)$ is the predicted value and $\Delta_z(t)$ is the residual.

This residual is then quantized to reduce the number of bits on which it is represented. A uniform quantization is employed, which means that the bins used for the quantization are all equal dimensions. The quantization is parametrized in two ways: either by setting the absolute error limit $m_z(t)$ or by setting the relative error limit r_z. $m_z(t)$ is used to bin the residual into $2m_z(t)+1$ values and the quantized output is obtained with the formula:

$$q_z(t) = sgn(\Delta_z(t)) \cdot \left\lfloor \frac{|\Delta_z(t))| + m_z(t)}{2m_z(t)+1} \right\rfloor \tag{2.3}$$

The relative limit is instead used to estimate the absolute error limit using this formula:

$$m_z(t) = \left\lfloor \frac{r_z(t)|\hat{s}_z(t)|}{2^D} \right\rfloor \tag{2.4}$$

where D is the dynamic range for the input. These error limits can be set for each spectral band of the image.

The quantized values $q_z(t)$ are then mapped to non-negative values $\delta_z(t)$ which are called *mapped quantized indeces*.

In order to obtain the same predicted value both at the encoder and the decoder, the estimation must be generated starting from the so-called sample representatives $s''_z(t)$. These are reconstructed values obtained by combining the residual with the predicted samples (using a more advanced method than simply summing the two). The predicted values are generated via simple interpolation schemes between previously encoded samples in the same band and samples in the same pixel position in the image but in previous bands.

Which samples are chosen is determined by the mode of prediction selected. There are four possible modes, which are the combination of the two choices between *narrow*/*wide* mode and *neighbor*/*column* - *oriented* mode. The choice between *narrow* and *wide* mode determines whether the sample to the left of the one that is being encoded ($s''_{z,y,x-1}$) is used for the purpose of prediction. In *narrow* mode this

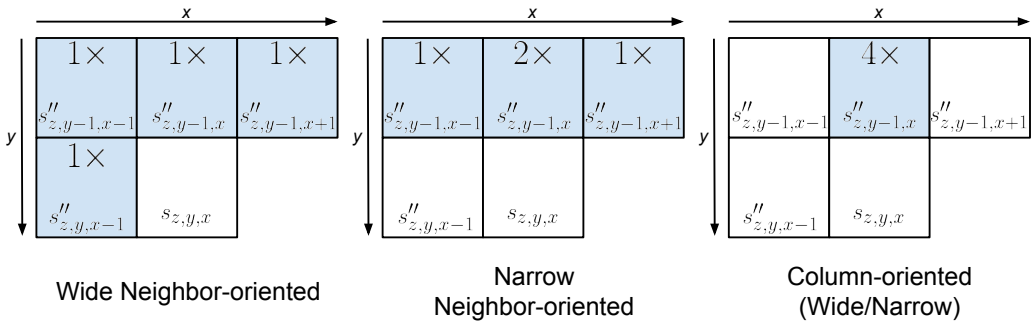

Fig. 2.6 Local sum modes available in CCSDS 123.0-B-2

is skipped to make the processing of the sample of a row of the image completely independent on previous samples of the same row, which is a useful property for purposes of parallelization on hardware. The impact of skipping this sample is limited [22]. Regarding *column-oriented* vs. *neighbor-oriented* mode, the first uses for prediction only the sample exactly above the current one while the latter also uses the sample on upper-right corner and the sample in the upper-left corner. This is displayed in Fig. 2.6.

The modes described above are used to compute a value denominated local sum $\rho_{z,y,x}$. This value is in turn used to generate the so-called local differences:

$$\begin{aligned} d_{z,y,x} &= 4s''_{z,y,x} - \rho_{z,y,x} \\ d^N_{z,y,x} &= 4s''_{z,y-1,x} - \rho_{z,y,x} \\ d^W_{z,y,x} &= 4s''_{z,y,x-1} - \rho_{z,y,x} \\ d^{NW}_{z,y,x} &= 4s''_{z,y-1,x-1} - \rho_{z,y,x} \end{aligned} \tag{2.5}$$

where $d_{z,y,x}$ is denominated the central local difference, while $d^N_{z,y,x}$, $d^W_{z,y,x}$ and $d^{NW}_{z,y,x}$ are the directional ones (respectively north, west and north-west). The predicted central local difference $\hat{d}_{z,y,x}$ is computed as a weighted sum of local differences.

There are two possible modes to generate $d^N_{z,y,x}$: *full* mode and *reduced* mode. In *full* mode $d^N_{z,y,x}$ is a weighted sum between the three directional differences for the sample ($d^N_{z,y,x}$, $d^W_{z,y,x}$, $d^{NW}_{z,y,x}$) and the central differences from neighbors located in the same position (x,y) but belonging to the previous P spectral bands $d_{z-1,y,z},\ldots,d_{z-P,y,z}$, where P is a parameter set by the user. The weights for the sum evolve through time

based on the evolution of the prediction error in the previously encoded samples. In *reduced* mode the directional local differences are not used.

Finally, the predicted central local difference $\hat{d}_{z,y,x}$ and the local sum $\rho_{z,y,z}$ are used to generate the predicted values $\hat{s}_{z,y,x}$.

Encoder Stage

The generated quantized indexes are then passed to an entropic coder. CCSDS 123.0-B-2 allows to choose between three possible algorithms: *Block-Adaptive Coder*, *Sample-Adaptive Coder* and the *Hybrid Coder*.

The *Block-Adaptive Coder* is a Rice coding based algorithm first described in the standard CCSDS 121.0-B-2 [23]. The input samples are partitioned in blocks of fixed dimensions between 8 and 64 samples. Each of these blocks is coded using one of five different coding methods. Each method is tested on the samples and the one that gives the best results is chosen.

The *Sample-Adaptive Coder*, first maps each input sample to different code-words which belong to a GPO2 family (Golomb power of 2). These code-words (which denote as $\mathfrak{R}_k(\delta_z(t))$) are generated based on an index k which is itself generated based on the statistics of previously encoded samples.

Given the user-specified GPO2 length limit U_{max}, if $\lfloor \delta_z(t)/2^k \rfloor < U_{max}$ then the selected $\mathfrak{R}_k(\delta_z(t))$ is composed by a series of $\lfloor \delta_z(t)/2^k \rfloor$ zeroes followed by a 1 and then the last k bits of $\delta_z(t)$. If $\lfloor \delta_z(t)/2^k \rfloor \geq U_{max}$ then $\mathfrak{R}_k(\delta_z(t))$ is made up of U_{max} zeroes, followed by $\delta_z(t)$ written in binary over D bits, where D is the dynamic range of the input image.

Finally, the *Hybrid Coder*, is an evolution of the *Sample-Adaptive Coder* which gives the possibility of encoding parts the signal with bit-rates which are less than 1 bit-per-pixel. The samples where this is done, which are denominated low-entropy samples, are mapped using variable-to-variable length codes (which means that groups of variable length of the input samples are mapped to variable length codes). The codes used and a more thorough description of the *Hybrid Coder* can be found in [24].

2.3 Deep Learning

In this chapter we provide an overview of the fundaments of deep learning. For a more extensive explanation of the basics of this field we suggest the book [25]. This book was used as basis for this chapter.

Deep learning is a family of algorithms in the field of machine-learning. As such, deep learning algorithms are not completely designed by the programmer, but are instead optimized to solve a certain task by taking large datasets of examples and extracting patterns from it (the algorithm "learns" from data, hence the name machine learning). Classical machine learning is dependent on having the data be represented in forms which makes the learning easier, a process called feature engineering.

Deep learning does away with this process by using large multi-layer (e.g. deep) architectures which are capable of learning both how to extract the useful features from the data and how to use them to solve the task. The simplest example of a deep learning algorithm is the feed-forward neural network (also known as multilayer perceptron or MLP).

2.3.1 Multilayer Perceptron

The objective of a feed forward neural network is to approximate an unknown function $y = f^*(x)$, which represents the task we need to learn. The structure of the network defines a function $y = f(\theta; x)$, where θ are tunable parameters which define the function.

The basic building block of the MLP is the neuron, which is a simple affine operator that combines several inputs using a series of weights and a bias parameter. The obtained output is then fed to a non linear function called activation.

$$y = f_{nl}(Wx + b) \tag{2.6}$$

where $x \in \mathbb{R}^m$ is the input vector, $W \in \mathbb{R}^{m \times n}$ is a weight matrix, $b \in \mathbb{R}^n$ is the bias vector and f_{nl} is the activation. In the architecture of the feed-forward neural network multiple neurons are assembled together in a multi-layer structure, displayed in chapter 2.7.

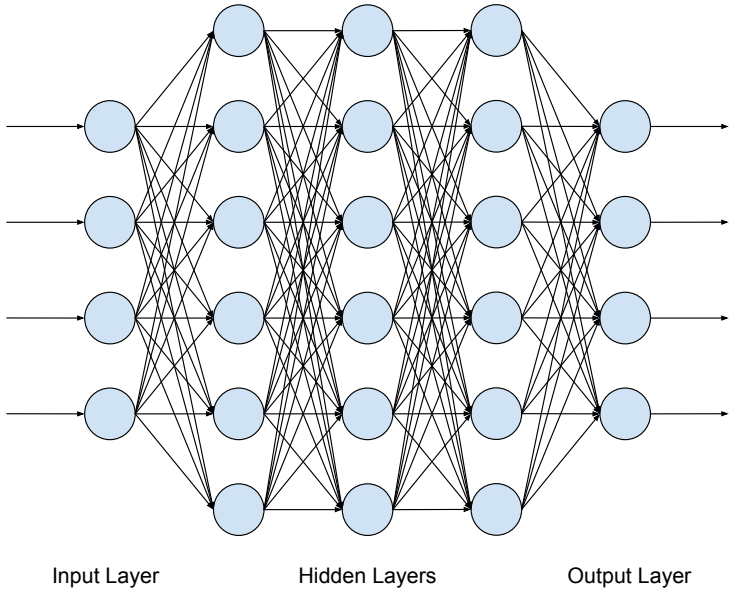

Fig. 2.7 Structure of the fully connected multilayer perceptron

Each layer outputs a new representation of the data in output from the previous layer. The multi-layer structure is used because it allows to create very complex functions with simple building blocks, by stacking them one on top of the other. The internal layers of the network which create these representations are called hidden layers, and the number of the layer is called the depth of the model, while the number of neurons in the layer is the width of the layer. In general, the greater the depth, the more the network is complex, and thus capable of approximating more complex functions.

The structure described in this chapter, where every neuron in a layer is connected with the all the neurons in the following layer is also called a fully connected network.

Activation functions

The activation function is applied to the outputs of the neurons in order to make to overall function described by the neural network non-linear. This makes the network more expressive in the sense that in this way it can be used to approximate also non-linear functions. Through the years many possible activation functions were proposed but the most used are:

- The sigmoid: $f(x) = \frac{1}{1+e^{-x}}$

- The ReLU (Rectified Linear Unit) function: $f(x) = max(0,x)$ [26]
- The Leaky ReLU: $f(x) = max(\alpha x, x)$ with α being a small value $\in (0,1)$ [27]

which are chosen depending on several criteria, including the computational efficiency of computing the function and its derivative.

2.3.2 Training and optimization

This structure is thus capable of creating arbitrary functions $y = f(\theta;x)$ by changing the values of its parameters θ. In order to approximate $y = f^*(x)$ the parameters need to be learned from data. This is done through a process of optimization called gradient-descent, which minimizes the value generated by a cost function.

Cost function

In the beginning, the weights of a neural networks are initialized to random values. Then the proximity of the neural network to the target $y = f^*(x)$ is evaluated by using a so-called cost function. A large dataset of examples of inputs is fed to the network, then the cost function is applied on the generated outputs. In most settings the dataset also includes for every input a corresponding desired output for that input, which is called a label. When the label is used in the cost function to train the network, the process is denominated supervised learning.

One approach to derive a cost function is to define the distribution $p(y|x;\theta)$ in the supervised setting, where y is modelled as a output of the true function affected by noise, and to maximize the likelihood. In this case the cost function becomes the negative log-likelihood and the objective of the training becomes its minimization.

$$C(\theta) = -\mathbb{E}_{x,y\sim\hat{p}_{data}}[log p_{model}(y|x)] \tag{2.7}$$

where p_{data} is the probability distribution of the data and p_{model} is the probability distribution assumed from the model. By choosing $p_{model}(y|x)$ to be a gaussian with the output value of the network as the mean

$$p_{model}(y|x) = \mathcal{N}(y; f(x,\theta), I) \tag{2.8}$$

and estimating the log-likelihood we obtain the mean square error (MSE) cost function:

$$C(\theta) = \frac{1}{2}\mathbb{E}_{x,y\sim\hat{p}_{data}}\left[||y - f(x;\theta)||^2\right] + const. \tag{2.9}$$

Other cost functions can be derived for different objectives and assumption on the data.

Back-Propagation

The process of feeding inputs to the network and the evaluation of the cost on the output is called forward propagation. Once the scalar cost is evaluated, this information is used to move the weights closer to a configuration of values which minimizes the cost function, thus getting the network closer to realizing the task is getting trained for. This is done through a process called back-propagation [28].

Back-propagation consist in evaluating the gradient of the cost function with respect to every parameter of the network. Thanks to the chain rule of derivation this can be evaluated in steps going backward from the output up to the beginning of the network, with simple operations (hence the name back-propagation). Once the value of the gradient in respect to the parameter, these can be updated, by moving in the opposite direction of the gradient. The dimension of the update is determined by a parameter called learning rate.

Optimization

In order to work correctly, deep learning needs very large datasets, which makes the estimation of the gradient of the cost function over all of it not practical in terms of time and memory. Usually, what is done instead, is to sample mini-batches from the dataset and optimize the cost function only over them. After every iteration of training, one mini-batch is discarded and another one, made up of different samples, is extracted. Once a round of training over every sample of the dataset is done, a length of time which is referred to as an epoch, the cycle starts over. The training continues up to convergence, which is the moment in which the cost function stops decreasing. This optimization over random mini-batches is denominated stochastic gradient descent (SGD).

What has been described up to now are the basics of training and optimization, but through the years more advanced techniques to minimize the cost function have been designed and are commonly used during training. One example is ADAM [29], which uses the same principle of SGD, but instead of keeping the learning rate fixed, it changes its value based on an evaluation of moving averages of the gradient and the squared gradient. In this way, the magnitude of the update for each iteration adapts to the conformation of the cost function in a particular region, and limits the risk of, for example, jumping over minima present in the region due to a learning rate which is too high. For this reason, the use of Adam usually speeds up the training process. Another example is AdaDelta [30], which is another algorithm to obtain an adaptive learning rate.

Dataset

The training is achieved by minimizing the cost function over mini-batches extracted from the what is called the training set. To avoid the risk of the network not generalizing to data outside of the input dataset (a phenomenon known as over-fitting), its performance are assessed over a portion of a separate dataset not used for training called the validation set, while the training progresses. The validation is used to iterate and modify the architecture and the hyper-parameters of the neural network, so indirectly the information it contains spills out in the final design of the network. For this reason, another dataset is also used, called the test set, to determine the final performance of the network.

Regularization

As anticipated in the previous section, there is always a risk for a trained network to perform worse when used on data outside of the training set. One way to ensure that this does not happen is to introduce regularizations.

Regularizations are a group of methodologies which use prior knowledge on the data and the function that has to be approximated, to help the neural network to achieve better results. This is usually realized by adding penalties on the cost function or constraints on the parameters. Among the most used techniques there is weight decay, which is a penalty on the cost function to reduce the magnitude of the

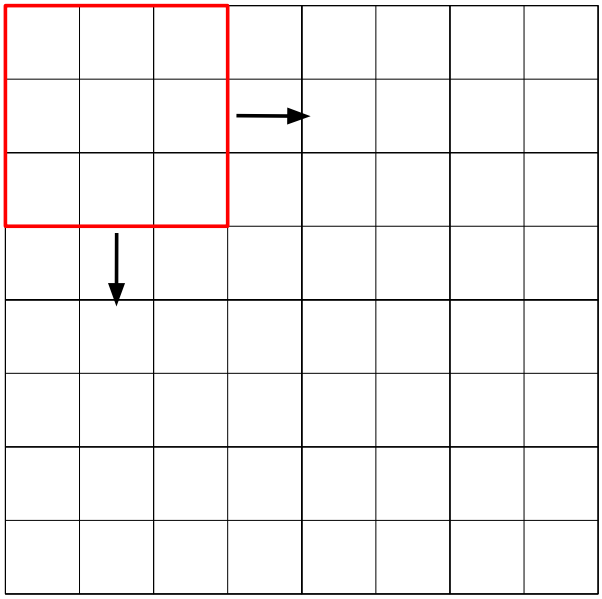

Fig. 2.8 Diagram of a convolution between a 2D grid-structured signal and a kernel

weights, and the L1 regularization, which imposes the minimization of the L1 norm of the parameters and is used to have more sparsity.

Another very important form of regularization is data augmentation. This consists in the generation of more data from the input dataset, usually by introducing simple transformations, like cropping and translations in the case of images. This is particularly used in tasks such as classification since it is useful to make the trained network more stable and invariant to transformation on the input. Another form of augmentation is the introduction of noise on the input data to provide robustness.

2.3.3 Convolutional Neural Networks

General Structure

Convolutional neural networks (CNN) [2] are a type of neural network specialized for data set up in grid structures, like images. As the name implies the main building block of this kind of network is the convolution operation (Fig. 2.8). This chapter will describe networks using 2D-convolutions, but this structure can be extended on any number of dimensions.

In this kind of neural network, what is learned are the coefficients of a convolutional filter. This is advantageous in multiple ways: it greatly reduces the number of parameters to be shared, as the dimensions of a kernel are much smaller than the input signal. Furthermore, this makes the network usable on inputs of arbitrary dimensions. For example, for problems involving images, the same convolutional network can be used on images of different shapes and resolutions.

For the CNN to work correctly, some assumption needs to be met: first, the data needs to be stationary, in this way the same filter can be relied upon to capture the useful features in any part of the input signal. Also, the useful features needed to accomplish the final objective should depend on localized regions of neighboring samples in the signal, so that they can be captured by convolutional filters. These properties are usually satisfied by natural signals.

The convolutional filters are also denominated kernels, while the output of each layer is called a feature map, since it provides a map in which the highest values indicate positions in the input where the kernel is matched. The span of input data captured by the kernel at every shift, determined by the size of the kernel, is called the receptive field.

Pooling Layers

When multiple layers of convolutions are stacked one upon the other, there is usually an operation which is executed between layers called pooling. These pooling layers reduce the size of the feature map and feed this down-scaled version of the input to the following layer. This is done for multiple reasons: first, it is useful in cases where the final output of the network is of smaller dimension then the input. Secondly, it allows to increase the receptive fields for the deeper layers and to capture characteristics which span larger and larger regions of the input. This is especially used in tasks such as regression and classification, where the output is a scalar or a vector, which need to be generated by taking information from the entire input. This also allows the network to be more robust to slight variations in the positions of the features.

The most used pooling layers are the average-pooling, which generates the down-sampled feature maps by locally averaging, and max-pooling, which extracts the maximum patch-by-patch.

Chapter 3

Deep Frame Extrapolation for Video Compression

3.1 Introduction

In a media landscape flooded by content due to the introduction of streaming services, sharing through social networks, and the increase in resolution in modern video files, the need for better video compression algorithms is ever-increasing. This poses a great challenge since, after decades of iteration on the design of new video coding tools, it becomes more and more difficult to develop methodologies that further exploit the redundancies present in video signals. Deep learning represents an attractive approach to design advanced architectures for a video compression algorithm.

As illustrated in the previous chapter, deep learning is a very powerful tool, and through the utilization of such techniques, the scientific community has been capable of achieving feats that would have appeared impossible just a few years ago. Even though there are still limitations compared to the currently employed techniques (especially in terms of complexity), deep learning shows great promise for video compression, and standardization committees are starting to investigate this field as an avenue for the development of the algorithms of the future.

In this context, our objective was to design a novel deep learning algorithm with the purpose of increasing the efficacy of already existing video coding standards,

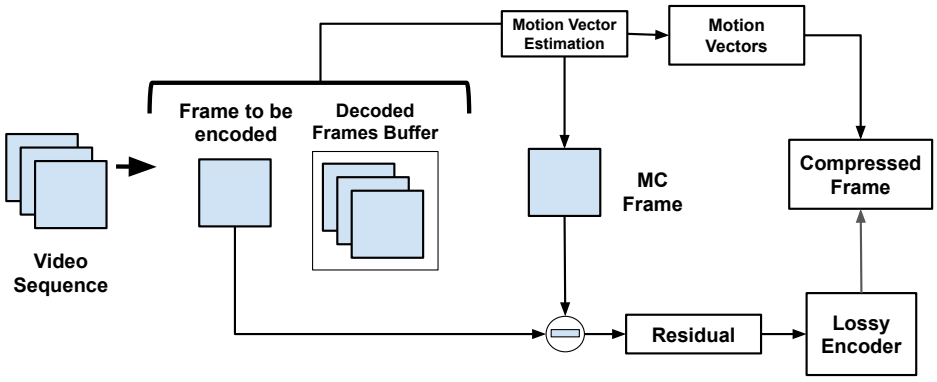

Fig. 3.1 Processing chain of hybrid coding schemes (like H.265/HEVC)

like the H.265/HEVC (High Efficiency Video Coding) [1]. In particular, our aim was to improve the temporal prediction aspect of existing codecs, carried out by the so-called inter-prediction algorithm. As illustrated in the background chapter, inter-prediction is primarily achieved through a step called motion-compensation, which is the construction of an approximated version of the frame that is being compressed by assembling square-shaped portions of past frames in order of encoding (the scheme is illustrated in Fig. 3.1).

Motion-compensation is a very simple mechanism that creates only rough estimates of the predicted frame. The question we investigated is the following: would it be possible, using the power of deep learning, to predict future frames based on previously transmitted ones, without transmitting motion parameters to the decoder? In this way inter-prediction could be executed without needing to transmit side information. Furthermore, such a network could learn to model more advanced forms of motion compared to motion-compensation (which can only compensate for translations of blocks of pixels) and thus, generate estimates which are more accurate.

3.2 Related work

Several papers have been created which use of machine learning and in particular deep learning for the purpose of compressing video. Broadly, the adopted algorithms in this field can be divided into two families: end-to-end compression, in which the neural network strives to perform the whole compression task on its own, and

compression tools, in which the aim is to improve the performance of an existing codec. In this section, some relevant techniques from both categories are described, but a more in-depth survey can be found in [31].

3.2.1 End-to-End Compression

Though in the beginning these promising works seemed limited in scope and unable to reach competitive performance, several end-to-end approaches have recently been proposed, which are closing the performance gap compared to conventional video coding standards such as H.265/HEVC.

Chen et al. [32] proposed PixelMotionCNN, an architecture that tries to re-create in a neural network framework the predictive coding workflow employed by coding standards such as H.265/HEVC. The network generates a prediction of a block of the frame that is being encoded by fusing information taken from previous frames together with the already encoded parts of the current frame. The residual between the predicted block and the original one is then compressed by using an iterative scheme based on an auto-encoder network, reaching results that are comparable to H.264/AVC.

Along the same lines, Lin et al. [33] tried to recreate the structure of modern codecs using a neural network. The proposed network is called M-LCV and extends the scope of previous approaches by including, among other things, multiple reference frames and motion vectors (instead of having the deep learning algorithm try to completely predict the frames). The architecture uses several different networks to recreate the various steps of hybrid video compression, such as motion estimation and residual and motion vector compression. These different segments are jointly optimized using a loss function based on rate-distortion minimization. This work manages to outperform H.265/HEVC for the low-latency mode.

Han et al. [34] proposed a generative approach based on the variational autoencoder architecture. It extends the variational autoencoder to work for sequential information to create a model where it is possible to condition the prior distribution of a frame based on the preceding frames. The obtained latent variable corresponding to the frame is then quantized and arithmetically encoded following the prior model. This very ambitious work is only limited to very low-resolution signals with

specialized content such as video game character sprites due to the complexity of the task of creating good models for video.

3.2.2 Compression Tools

Most of the works in the field fall in the second category, i.e. deep learning compression tools. These tools are used to improve the performance of existing codecs, usually by tackling some specific parts of its architecture and trying to improve upon them. This makes the goal of achieving state-of-the-art performance in the field of video-compression way easier compared to building a complete compression algorithm from scratch. Different techniques try to tackle different steps of the video-compression algorithm.

A popular topic is the use of neural networks as an in-loop filter, which allows the use of video enhancement techniques for compression. For example, Li et al. [35] propose a multiframe enhancement of the reconstructed frame. First, it selects a portion of previous frames from the reference picture list, choosing the one that more closely resembles the frame being encoded. Then these suitable reference frames are motion-compensated by a network and then fed together with the current frame to a neural network based on DenseNet [36]. Also, a map of how the frame is partitioned in coding and transform units is given to the network to help with artifact removal. Similarly, Zhang et al.[37] focus on reducing artifacts on intra-predicted frames by using a recursive residual network.

Several proposed schemes exploited the success of neural networks in the field of image compression and concerned themselves with intra-prediction. Li et al. [38] propose the use of a fully connected network. The model takes as input multiple reference lines of already encoded pixels, that surround the block and whose content is highly correlated with it. In contrast, H.265/HEVC uses only one line of reconstructed samples. In [39] all intra-coding modes are substituted with neural networks. Fully connected networks are used for small blocks, while for sizes 16×16 and 32×32 convolutional networks are used. More recently, [40] proposed the use of generative adversarial networks, based on the reasoning that intra-prediction can be seen as an inpainting task since it consists of the completion of a partial image which is a task usually accomplished with such networks.

Some works try to tackle inter-prediction. Among them, we remember Zhao et al. [41], which use the structure for interpolation between frames proposed in paper Deep Voxel Flow [42] to construct better reference frames denominated "virtual reference frames". The idea of enhancing the reference frames is also used in Lee et al. [43], with the difference that this structure is studied for the uni-prediction setting.

3.3 Proposed method

3.3.1 Problem setting

As anticipated in the introduction there are limitations in the capabilities of inter-prediction based on motion-compensation. First, motion-compensation is very limited: it works on square sections of pixels at a time and can only correctly model translations. Thus, it is impossible for motion-compensation to represent more complex forms of movement through time such as rotations and deformations of objects. Furthermore, motion-compensation makes it necessary to send side information together with the encoded stream in the form of the spatial offsets needed to retrieve the predictor from the reference frames (the so-called motion-vectors).

To solve all these problems we conceived of a network that predicts a frame taking previous ones as input. Such a network could both eliminate the need to send side information and could possibly compensate for more advanced modes of motion present inside video signals.

3.3.2 Network Structure

Base Model

The starting point for the approach we propose was the model described in the paper [3]. This method, presented at CVPR 2018, is a convolutional network for the purpose of video super-resolution. The proposed method utilizes the temporal context of a frame (e.g. the frames that immediately precede and follow the examined one) to create an up-sampled version of it.

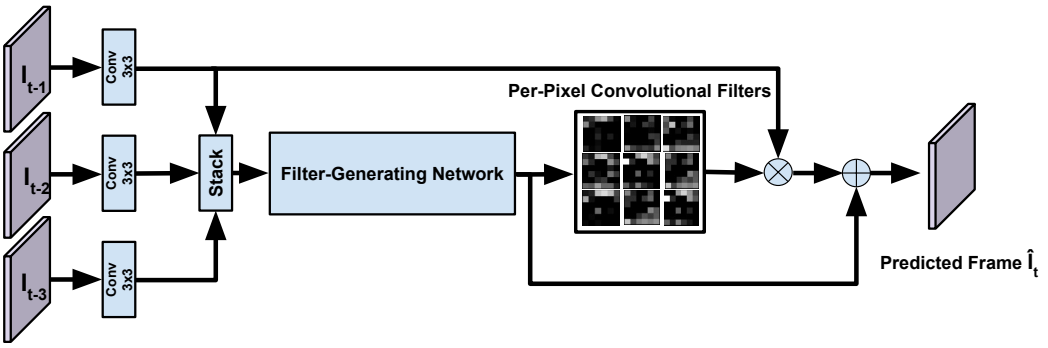

Fig. 3.2 Proposed approach

In most other approaches for video super-resolution, the input is fed together with the temporal context frames, and the job of the network is to merge internally the information to come up with the super-resolved output. In addition, usually the temporal context frames are first motion-compensated with respect to the frame that has to be super-resolved ([44, 45]). This alignment makes the work of the network simpler because in this way it can find the objects represented in the video in the same positions across the various frames. This has the downside that the alignment of the previous frames can distort them and possibly destroy part of the information they contain. Also, there is a great dependence on the quality of the motion-compensation to have good results in terms of super-resolution.

[3] introduces the Dynamic filter generation network, a different paradigm in which the network uses the context frames and the input to create convolutional filters that are applied to the output frame in order to obtain the super-resolved one. As the word "Dynamic" in the name of the network suggests, a different up-sampling filter is generated for each pixel of the input so that based on the location inside the frame the surrounding pixels are adaptively interpolated based on which part of the image is being examined.

In our case, we applied the idea of a filter generating network to the problem of inter-frame prediction: three consecutive frames are fed to a filter-generating network and the obtained filters are applied to the last of the frames to obtain the predicted one.

The filter generating network is a CNN using 3D-convolutional filters to capture both spatial and temporal information from the stack of frames. We used the simplest model described in the paper [3]: each layer uses batch normalization [46] followed

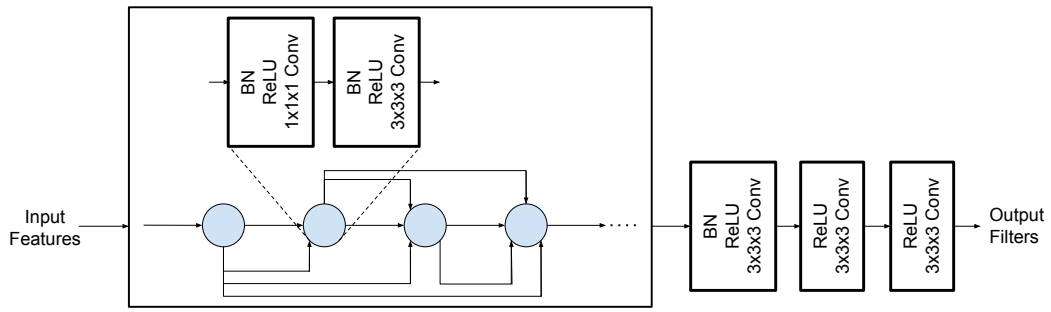

Fig. 3.3 Internal structure of the filter generating network.

by ReLU activations, a convolutional layer with a filter with size 1×1×1, then batch normalization again, ReLU, and another convolutional layer with size 3×3×3.

Each following layer employs as input the features extracted by all the previous layers. This is done by concatenating the outputs of the previous layers over the channel axis. This follows the structure proposed in the paper [36].

We started based on the smallest of the models proposed in [3], a 6-layer structure in which each layer adds 32 feature channels to the following ones. The output of the sixth layer is a feature map with 256 channels. This is fed to another Batch Norm, ReLU, and a 2D convolution 3x3. After this, the network separates into two branches, one branch for generating filters, and the other for generating a residual that is added to the filtered output. This residual is used to make the generated prediction sharper since the output of the filtering operation tends to generate estimates which are smoothed. Both branches comprise two blocks of ReLU followed by a Conv 1x1, but the branch which generates the filter further employs a SoftMax block. This makes the values of the filter sum to one and makes the output of the filter depend mostly on one pixel in the receptive field. In this way the filtering acts more as a motion vector applied to one or few neighboring pixels and not interpolation across the elements of the receptive field of the filter. The structure is illustrated in figures 3.2 and 3.3.

3.4 Dataset and Training

3.4.1 Dataset Creation

A dataset to train the network was created starting from raw video sequences provided by *RAI - radiotelevisione Italiana*. This collection of videos consisted of 24 sequences in total: 12 sequences at resolution 1080p and other 12 at resolution 2160p. All of them were captured at 50 fps, with variable lengths. 5 of these sequences are widely available (*CrowdRun.yuv*, *ParkJoy.yuv*, *DucksTakeOff.yuv*, *InToTree.yuv*, *OldTownCross.yuv* can be found on the Xiph.org website [47]), while the others were captured by *RAI* (some examples can be seen in image 3.4). It was important to use raw sequences to avoid having compression artifacts in the training datasets. The downside is that the resulting data is very large in terms of memory, since even sequences under a minute of length at 1080p can occupy more than 1 GB of memory. The 24 sequences cited occupy 144 GB of memory in total. The training dataset was obtained starting from this collection of videos by extracting short 4-frames sequences, in order to use the first 3 frames as input and the last one as the label. Since it was not feasible to work with full-frames as it would require too much memory (and would make it impossible to train on batches of inputs), random crops of size 144×144, were extracted. In total, 6.6×10^4 sequences were collected.

At first, it was contemplated to extract the crops based on the amount of motion contained in them (selecting thus only sequences with noticeable motion), in order to promote the learning of a motion model. This was discarded as we did not want to create biases in the training, leaving the use of a dataset selected using this criterion for possible future tests.

3.4.2 Training Setup

The network was implemented in *python 2.7* using the platform for machine learning *TensorFlow v1* [48]. It was trained using one *GPU Nvidia TITAN RTX 6000*, which uses 24 GB. The batch size used was 16 images and the network was optimized using ADAM [29]. Learning rates were tested in the ranges between 1×10^{-4} and 1×10^{-5} and using a scheduler with exponential decay every 4×10^3 iterations.

Fig. 3.4 Frames from the sequences provided by RAI. On top is a frame from the sequence *FountainLady*, in the middle there is a frame from *LupoBoa* and in the last row there is a frame from *RainFruits*.

Several loss functions were tested: we started from L2 loss, then, since this loss tends to generate blurry estimates we tested L1 and the Hueber loss [49], which are more robust. Also, a variation of the loss function described in the paper [50] was tested (this loss was adapted to work in this case where prediction is computed on a single frame). This loss, called Gradient Difference Loss (GDL) is designed to favor the generation of sharper estimates:

$$L_{gdl}(I_{t-1}, I_t, \hat{I}_t) = ||I_{t-1} - I_t| - |I_{t-1} - \hat{I}_t||^{\alpha} \tag{3.1}$$

where I_t is the label, $\hat{I}_t$ is the estimated predicted frames and I_{t-1} is the video frame corresponding to the time $t-1$. This loss was tested for the values of $\alpha = 1, 2$, which we denominate "Gradient L1" and "Gradient L2".

3.4.3 Experiments

During our work on this topic, two different architectures were examined.

Frame Prediction and Super-Resolution

The method described in [3] is designed for up-sampling so it has to generate multiple filters for every pixel on the low-resolution input. For example, when an image is up-sampled by a factor of 2 (which means that both height and width are doubled), 4 pixels need to be generated for every pixel of the input.

At the beginning of our work, we kept a similar structure to the one described in [3] and designed a network enacting both frame prediction and super-resolution. This meant that the input frames needed to be down-scaled to a lower resolution while the predicted frames generated by the network were kept at full size (see Fig. 3.5). In our experiments we used a factor for down-scaling the image of 4 (as it was done in [3]).

This approach of working on down-sampled versions of the input was tried because it allowed us to easily capture large motions across the frame. But, after a few experiments, we determined that this method was ineffective as the network after training learned only to act as super-resolution for the previous frame while failing to capture and estimate motion. As can be observed in figure 3.5, given the

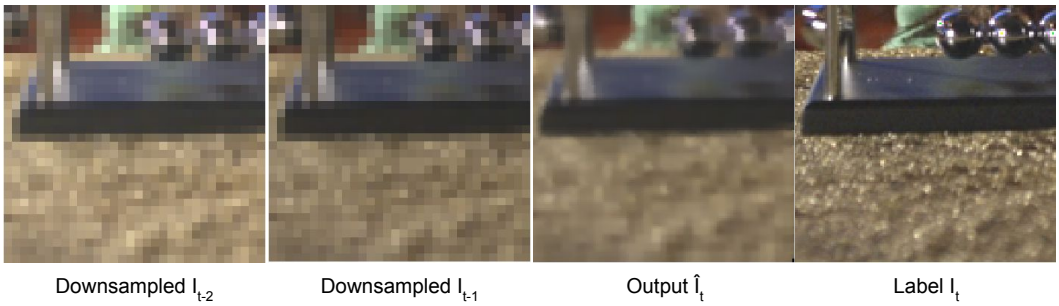

Fig. 3.5 Example of output from the first version of the network. The output is low-quality and the network seems to struggle in predicting a future frame at a higher resolution.

down-sampled input, the priority is the generation of a somewhat faithful estimate of the frames at the original resolution, while predicting the temporal evolution becomes secondary. For this reason, we decided to change the architecture of the network.

Frame prediction without up-sampling

The previous approach was abandoned in favor of creating a network that just operated as frame prediction without being burdened by also having to learn an effective way to accomplish the task of super-resolution. In this version, the inputs were kept at full resolution, and thus only one filter per pixel had to be generated. This approach focused the structure and we started to see the network generating estimates of a future frame.

After a while, though, we determined that there was a limitation to the capabilities of the network to compensate for motion. This was due to the dimensions of the generated convolutional filter, which determines the greatest vertical and horizontal pixel offsets that the network is capable of compensating. If a filter is too small, it means that it cannot compensate for motions that are bigger than its receptive field, and since we were now working with inputs at full resolution, this was a common occurrence.

Simply generating bigger filters was not feasible, though, because it resulted in a great increase in the memory required by the network. To remedy this problem, we decided to modify the network to make it generate linearly separable filters.

Linearly separable filters are filters that can be obtained as a scalar product of 1-dimensional vectors. In this way the network does not need to generate $n \times n$ coefficients for every pixel anymore, and instead only outputs two vectors of dimension $1 \times n$ per pixel, greatly reducing the memory requirements. Of these two vectors, one can be thought as dictating the horizontal offset, while the other dictates the vertical one. The two filters are multiplied with each other using a scalar product to generate the final 2-D filter.

This method has the downside of limiting the amount of expressiveness of the generated filters, because it cannot generate any possible 2-D filter anymore, and is now restricted to just the ones that can be decomposed in two $1 \times n$ vectors. It is nevertheless necessary because it makes possible to create much larger filters. Using this architecture, we were able to generate filters with dimensions of 17×17, greatly increasing the compensation capabilities compared to the previous architecture where the filters were of dimensions 5×5.

Since the new network uses inputs at the same resolutions as the outputs, for memory reasons a new dataset was extracted, where each element has dimensions 64×64 instead of 144×144.

The network was tested with several configurations, and the best results for the training were achieved using L1 loss, learning rate 1×10^{-4} with exponential decay every 4×10^3 iterations. Training reached convergence after 1.4×10^5 iterations.

With this streamlined architecture we started seeing promising results. The network managed to generate predictions of new frames based on previous ones with accuracy in terms of MSE closer to the one achieved using motion-compensation.

Figure 3.6 shows a comparison between the outputs generated by the trained network and a baseline motion-compensation algorithm. This baseline estimate was created using motion vectors extracted using a Matlab [51] function called *vision.BlockMatcher* with blocks of dimensions 4×4, an exhaustive search for the blocks, and no sub-pixel precision for the motion vectors. As it can be observed, the developed method generated predicted frames which started to approximate the performance of this simple version of motion-compensation, with the biggest problem being a general blurriness of the estimates. This can be explained by, among other things, the problem that given a set of previous frames there is not necessarily a singular possible future frame completely determined by those previous frames. For

Fig. 3.6 Example of the output from the network with 1-D separable filters. The first row shows the input frames, the label and the output generated by the network, while the second row shows the same input and label but the output is generated using a basic motion compensation algorithm.

this reason, the network generates a blurry estimate to create a sort of "averaging" between the different temporal evolutions.

3.5 Performance analysis

To evaluate the quality of the prediction of the network we compared its performance to the ones of the motion-compensated estimates generated H.265/HEVC on some of the *Joint Video Experts Team* (*JVET*) test sequences selected for this codec [52]. H.265/HEVC was tested in lossless mode with a setting that only uses uni-directional prediction (Low-Delay P) and uses a value for the quantization parameter QP equal to 22. The motion-compensated frame was extracted by modifying the code of H.265/HEVC official implementation HM [53]. The comparison was done by evaluating the PSNR in the Y-Channel in the YUV color space.

As it can be seen in Table 3.1, the method is working correctly but there is still a significant gap in the prediction quality, as the network was not capable of

Sequence Name	HEVC Lossless Y-Channel [dB]	Network Y-Channel [dB]
BQSquare	33.19	28.35
BasketballPass	32.75	27.12
ParkScene	33.89	29.13
RaceHorses	33.4	28.11

Table 3.1 Comparison on the average the quality of prediction in terms of PSNR between the inter-prediction algorithm used by H.265/HEVC and the prediction generated by the network.

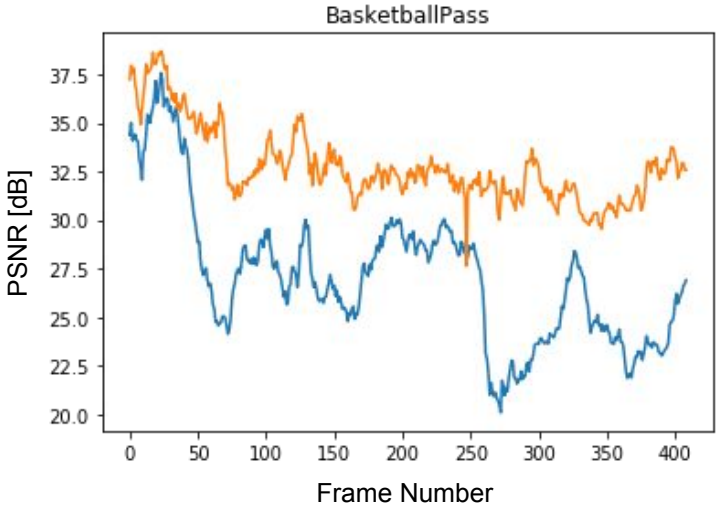

Fig. 3.7 Comparison in the quality of prediction in terms of PSNR between H.265/HEVC and the the network for every frame of the sequence BasketballPass using QP equal to 10. The yellow line represents the motion-compensated frames generated by H.265/HEVC while the blue one represents the predictions from the network.

satisfyingly compensating for larger and more complex forms of motion without generating blurry or artifacted predictions.

There is great variability in the quality of the prediction from frame to frame (Fig. 3.7). As it can be observed in Fig. 3.8, the performance drops especially in frames where there is more fast movement (like frame 250 from the sequence *"BasketballPass"*, portrayed in the left column of the figure).

This can be connected to the fact that compensation capabilities of the network are still limited since even after the use of the separable filters, the maximum span of

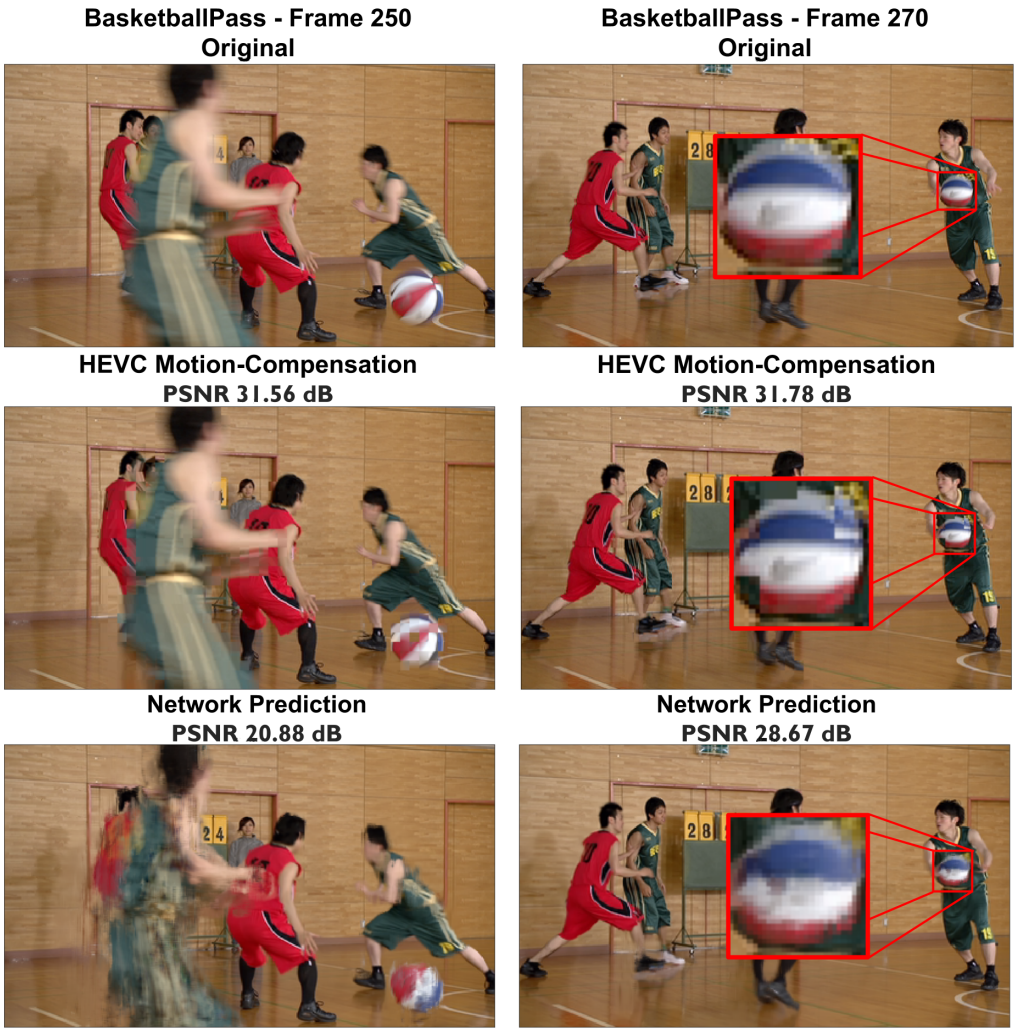

Fig. 3.8 Comparison between the predictions generated from H.265/HEVC and the prediction generated by the network for selected frames of the sequence BasketballPass. The left column shows frame 250 while the right one shows frame 270.

movement that was possible to compensate was only 8 pixels vertically and 8 pixels horizontally.

We also tried to create rate-distortion curves for some tests using sequences from the Xiph.org website [47]. In particular the sequences chosen were *mobcal* and *shields*. Since this method was not implemented inside HEVC the rate was evaluated using an indirect methodology: a residual signal was obtained by subtracting the predicted frame from the original one. This residual was then treated as an image and compressed using the image compression standard JPEG [54].

This was seen as a fair method to simulate the effects of the compression algorithm for the prediction residual followed by H.265/HEVC since in both cases the principle employed is basically the same: the compressed file is the block-based quantization of the coefficients obtained after applying a DCT transform. Furthermore, we were interested in the relative change in terms of the rate before and after the employment of the prediction network, and not the absolute values, which makes the differences between the algorithms less impactful.

In order to use JPEG for this purpose, we needed to format the prediction residual between *H.265/HEVC* as an image, with pixel values represented on 8 bits. Since the residual is the difference between the MC-frame and the original one, we first made the values positive by adding a constant offset equal to half of the range (e.g. 128), and then we clipped the values to fit the range $[0, 255]$.

The test was done by compressing the sequences over a range of values for the QP: the values chosen were $\{10, 15, 20, 25, 30\}$. To simulate more realistically the relationship between rate and distortion that would be obtained using H.265/HEVC, the residuals were compressed using parameters for JPEG which matched the amount of distortion obtained for a certain value of QP. Specifically, we searched for matching values for a parameter of JPEG called quality factor q. The conversion table shown in table 3.2 was derived.

Table 3.2 Compression parameters for the experiments.

Conversion Table					
QP	10	15	20	25	30
Quality factor q	95	90	85	80	75

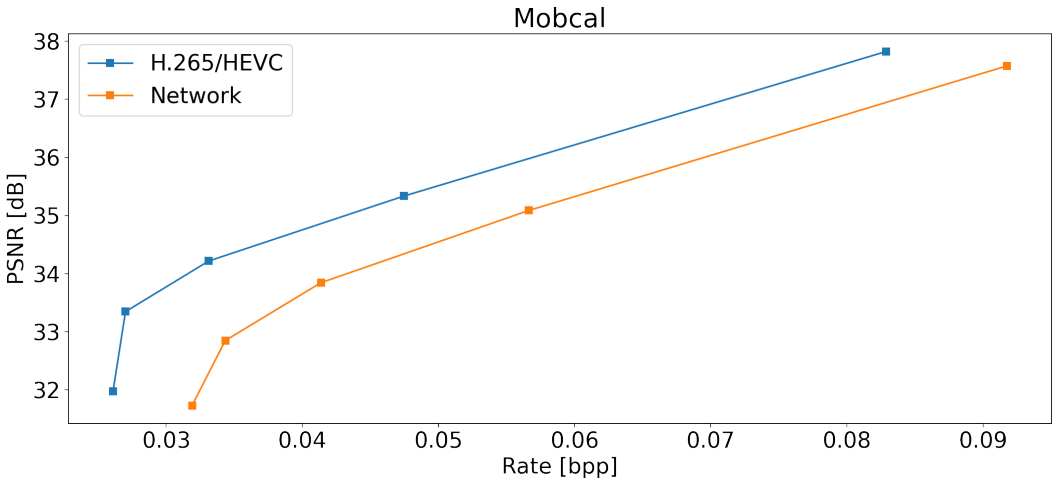

Fig. 3.9 Rate-distortion curves comparing the performance of the frame prediction network (blue line) and H.265/HEVC (red line) for the *mobcal* sequence (upper graph)

Using this indirect method, we were able to build rate-distortion curves for some tested sequences. The rate was evaluated by averaging the dimensions of the compressed residuals files, while the distortion was evaluated by measuring the PSNR between the residual after compression and the original ones. The obtained rate-distortion curves can be seen in Fig. 3.9 and 3.10.

As can be observed in the rate-distortion graphs, there was still a significant performance gap compared to the predictions of H.265/HEVC, due to the limitations described above.

3.6 Conclusion

At the same time we were working on this concept, a similar method has been published in [55]. This work describes a CNN based on U-Net [56] which takes as input two frames from the reference frame buffer of HEVC, and tries to create an inter-predicted frame. Similarly to our work, it employs a filter generating network, but their algorithm supports both uni- and bi-directional inter-prediction and was fully implemented inside the H.265/HEVC codec. For this reason, we decided to slightly change the aim of our work. How the work evolved will be described in chapter 4.

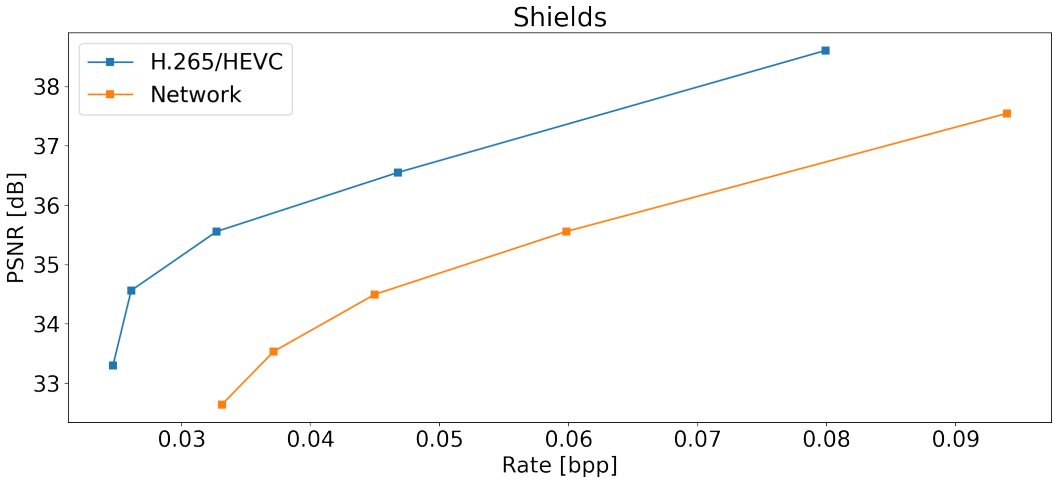

Fig. 3.10 Rate-distortion curves comparing the performance of the frame prediction network (blue line) and H.265/HEVC (red line) for the *shields* sequence (upper graph)

Chapter 4

Deep Motion-Compensation Enhancement in Video Compression

4.1 Introduction

At this point, we changed the objective of our work. Again we dealt with inter-prediction, but in this case, instead of completely replacing motion-compensation we envisioned a different approach. In this new design, the network takes as input the motion-compensated frame from the codec and it is tasked with creating an enhanced version of it.

A more accurate predicted frame reduces the entropy of the residual, and thus improves the compression performance, without any additional side information needed. Furthermore, using this architecture, we would be able to compensate for more advanced forms of motion, compared to the block-wise translations modeled by the algorithm used by H.265/HEVC. The enhancement is aided by the use of previously reconstructed frames. In order to maximize the usefulness of past frames, they are first registered to the motion-compensated frame using an optical-flow network and a warping module. We called this method the M (MMCE-Net). The work described in this chapter has been previously published in [57, 58].

The remainder of this chapter is organized as follows: first, we describe the architecture of the MMCE-Net technique, then an explanation on how it was inte-

grated within H.265/HEVC reference implementation and the performances obtained; finally, ablation tests are provided to verify the selected design.

4.2 Proposed method

4.2.1 Problem setting

MMCE-Net aims at improving compression performance by focusing on the enhancement of the motion-compensation step. In codecs such as the H.265/HEVC standard, the motion estimation step is achieved by iteratively shifting the block being encoded across a collection of previously reconstructed frames (the reference frames), searching for the best matching location which minimizes a given cost function.

Such motion-compensated frames can exhibit a lot of artifacts [59], especially in areas characterized by a large amount of activity. In particular, motion-compensation introduces blocking artifacts, which are sharp rectangular-shaped discontinuities, derived from the fact that motion-compensation stitches together pieces of different images. These discontinuities create high frequencies in the residual, which leads to reduced coding efficiency. Furthermore, if the artifact is not completely removed after adding the quantized residuals and applying the in-loop deblocking filter, it can propagate to future frames as false edges if the interested region is used as a predictor.

4.2.2 Network Architecture

The proposed method employs a neural network that acts on a set of frames, composed of the motion-compensated version of the frame to be encoded along with a few reference frames, and attempts to produce the best possible estimation of the frame to be encoded by removing such artifacts.

Recent advances in video processing with multiframe deep neural networks [60–62] showed an impressive ability at capturing the temporal dynamics of the sequence. Hence, the rationale of the proposed method is that the network should be able to uncover more complex motion patterns within the input set of frames and

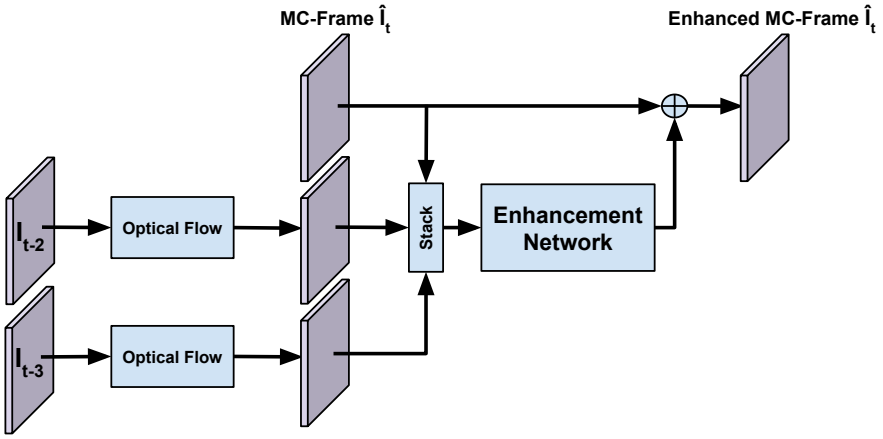

Fig. 4.1 Structure of MMCE-Net: An MC-Frame is concatenated with two previous frames and fed to an enhancing Network. To make the enhancing job easier the frames are first warped using an optical flow network, so they spatially match the MC-Frame.

correct the motion-compensated prediction so that it is more similar to the original frame. A better estimate produces a residual with a smaller amount of entropy and, therefore, improves the compression performance.

General Structure

The architecture of the our model is illustrated in Fig. 4.1. A convolutional neural network (CNN), which we denominate the enhancement network, is fed with three inputs. The first is the target of the enhancement: the motion-compensated frame (from here on out called P_t or MC-frame). Stacked upon it are two further images: I_{t-2} and I_{t-3}. These are two previously encoded frames that are added to provide information useful to reconstruct the artifacted parts of the MC-frame; the intuition is that, by using 3 different captures of the same scene at different moments in time, the network can find a way to merge the information present in all of them and use it to reduce the strength of the artifacts present in the MC-frame.

To this end, I_{t-2}, I_{t-3} are first pre-processed: for both an optical flow relative to the MC-frame is extracted and used to warp them. This is obtained by re-sampling the images, using the offsets provided by the optical flow as grid and applying cubic interpolation in the sampling points. This is done in order to obtain a rough registration between the contents of I_{t-2}, I_{t-3}, and the MC-Frame, and makes the work of the network easier.

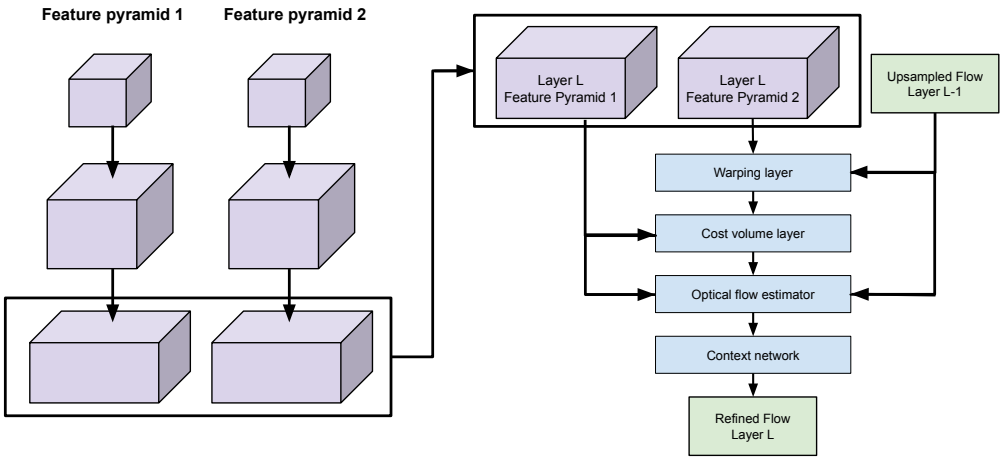

Fig. 4.2 Structure for PWC-Net: a Pyramid of features is extracted from the two input frames. Each of these layers is fed to a pyramid plus an up-sampled version of the optical flow extracted from the previous layer is fed to a system of CNNs enacting different roles.

Optical Flow Network

The optical-flow was estimated using a neural network called PWC-Net [4]. This network was chosen because it was the state of the art in the field of optical flow networks at the time. Its name stands for Pyramid, Warping and Cost volume network, which are the basic building blocks of this technique.

A diagram of the structure of the network can be seen in Fig. 4.2. First, a CNN extracts a feature-pyramid from both inputs. This feature-pyramid is like an image pyramid but where each layer is not created by simply down-sampling the input image at different scales. Instead, the feature maps for each layer are extracted using convolutional layers.

Then, the same steps are repeated on each layer of the feature pyramid. One of the two frames is warped using optical flow generated by a lower-level layer (opportunely up-sampled to work at a larger scale). The warped output and the other input are then fed to a network that extracts a cost volume [63] between them, which is an estimation of the correlation between the two feature maps. The cost volume, the input frames, and the optical flow are then fed to a CNN that extracts the optical flow for this layer of the pyramid. This optical flow is further enhanced using another CNN-denominated context network.

In our scheme, we employed a pre-trained 6-layer version of PWC-Net, which constrained us to use inputs with sizes bigger than 64×64 since each layer of the pyramid reduces the spatial resolution by half.

It has to be noticed that we do not use the decoded frame $I_{t\text{-}1}$ in this network. This was done because P_t is too similar to I_t. In fact, in the first approximation, P_t is already obtained as a motion-compensation of $I_{t\text{-}1}$, which makes the use of I_t redundant.

Fig. 4.3 shows an example of a warped frame using the optical flow extracted using PWC-Net. The obtained warped frame is aligned to the target, except in parts where the input is left as is, or areas that are completely black. This happens in the parts of the input image where is not possible to match the target image.

Enhancement Network

As mentioned before, the network needs to employ the information present in the three input frames in order to find a way to enhance the MC-Frame. To achieve this, the three frames were stacked one upon the other across the dimension of the color channels. It is then the work of internal structure of the CNN to merge the information in a suitable way to achieve the final goal.

This tensor is then fed to a Dn-CNN architecture [5] which gives the enhanced frame as output. This network employs residual learning and skip-connections, and its composed of several layers of convolution, batch-norm and ReLU activations. Since it uses residual learning, it does not estimate directly the output frame, but instead it generates a residual to be added to the MC-frame to generate the desired output. As it will be explained later, the whole processing was carried only on the luminance component of all the inputs, ignoring the chrominance component.

4.2.3 Dataset

The dataset used to train the network was obtained starting from VIMEO 90K [62]. This dataset is composed of 90,000 videos taken from the streaming website VIMEO, where each of the sequences is made up of seven frames with a resolution of 448×256. This collection of sequences contains scenes depicting a great variety

Fig. 4.3 Example of the effects of warping an image using an optical flow extracted with PWC-Net.

of subjects and types of movement, while requiring a reduced amount of memory to store it.

For every frame, the color space was changed from RGB to YCbCr following the color primaries described in ITU-R BT.709. The raw video files were created by concatenating together the frames belonging to the same sequence. These raw files were then compressed using x265, a very popular (and optimized) implementation of the H.265/HEVC [64]. All sequences were encoded setting a constant quantization level. This means that the same quantization parameter (QP) is used on every block of every frame, and the adaptation mechanism to keep the rate constant is disabled. Furthermore, the sequences were compressed disabling bi-directional prediction, so all frames were predicted using temporally preceding frames (this is a setting similar to the one described by the Low-Delay P configuration in HM).

At first, one dataset was created using QP $= 22$. Further trainings were executed using a second dataset obtained by using $QP = \{27; 32; 37\}$, choosing one of these values randomly with equal probability for each. The MC-frames were then obtained from the compressed video files using a software for video analysis called StreamEye [65], while the reconstructed files were obtained using the shell tool ffmpeg. Using this procedure, we were able to obtain all the images needed for the training: one MC-frame P_t and two reconstructed frames ($\hat{I}_{t\text{-}2}$ and $\hat{I}_{t\text{-}3}$), and one original frame used as label I_t.

4.3 Training and implementation details

As explained previously, MMCE-Net is composed of two parts, i.e. the optical flow pre-processing and the actual multiframe enhancement part.

The optical flow is extracted using a pre-trained TensorFlow implementation [66] of PWC-Net. The `LG-6-2 MULTISTEPS CHAIRSTHINGSMIX` model was employed because it is the trained model which achieves the best performance, although it is also the one which requires more memory.

The enhancement network architecture chosen is a 20-layer Dn-CNN with 2D convolutions initialized using Glorot technique [67]. The network was trained using an Adam optimizer [29] and minimizing the loss between the label and the output of

the network. The chosen loss function is L2, and the learning rate was set to 10^{-4}. The code was implemented using the TensorFlow platform.

The network was trained for 10^6 iterations using an NVIDIA Tesla V100 SXM2 GPU with CUDA 11.0 installed. The batch size was set to 16 and the training was executed on square crops of the dataset of size 256×256.

The network was first trained on the dataset created fixing the value QP $= 22$, then it was fine-tuned on the multi-QP dataset. This was done to make the final trained network function well independently from the QP level.

4.4 Experiments

4.4.1 Full-Frame Enhancement

Before fully committing to this method and implementing it inside H.265/HEVC, we tested its performance with a set of experiments aimed at determining the reduction of the entropy of the residual before and after the introduction of the enhancement.

A testing set was built from the standard test sequences provided by JVET [52]. The sequences were compressed using H.265/HEVC using the x265 library, configured for uni-directional prediction and constant quantization with QP $= \{22, 27, 32, 37\}$. Then the motion-compensated, reconstructed, and original frames were extracted (in the same way as it was done for the dataset that was extracted from VIMEO 90K), and we applied MMCE-Net on this testing set of images.

Since we could not directly estimate the change in rate that we would have obtained by using this technique inside H.265/HEVC, we tried to assess the entropy of the prediction residual by treating it as an image and compressing it using the image compression standard JPEG [54], similarly to what we did in chapter 3. The level of quantization was set using the ffmpeg parameter *qscale* with values $\{4, 7, 10, 20\}$. These parameters were chosen to roughly approximate the levels of distortion obtained by using the QP values of $\{22, 27, 32, 37\}$.

The general gain in terms of rate-distortion was evaluated by employing the BD-rate metric [6]. Table 4.3 displays the obtained results, while Fig. 4.4 show some of the corresponding rate-distortion curves.

Table 4.1 Performance of MMCE-Net full-frame enhancement on JVET test sequences, estimated using JPEG compression:

Sequence Name	**BD Rate** [%]
BasketballDrive	-5.83
ParkScene	-4.14
Kimono1	-4.12
Cactus	-4.89
BQTerrace	-6.19

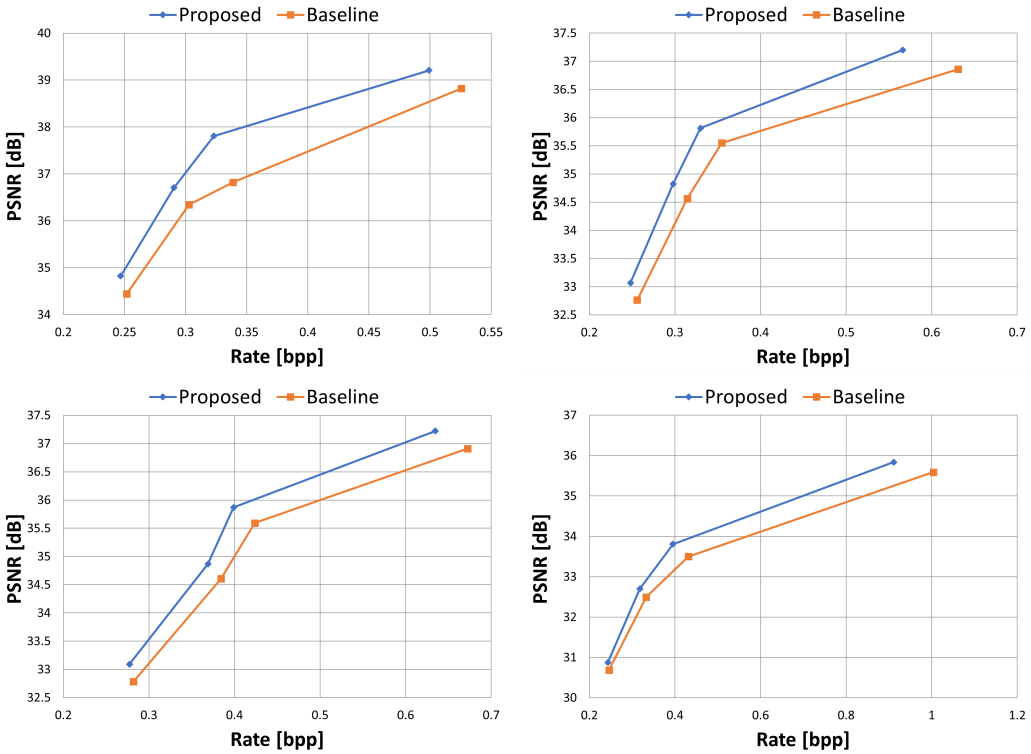

Fig. 4.4 Rate-Distortion curves for full-frame enhancement for the four JVET test sequences, estimated using JPEG compression. The first row contains the curves for sequences Kimono and BasketBallDrive, while the lower row contains the curves for sequences ParkScene and BQTerrace.

We observed that MMCE-Net provided a gain in terms of BD-Rate up to 6.19%. The enhanced MC-Frames brought both a reduction in rate-distortion on all the sequences examined at every level of quantization tested. It seems that most of the improvement is seen at lower levels of quantization, while it shrinks for QP $=$ $\{32,37\}$. This is further displayed in figure 4.5: there is a reduction of the rate on every frame, but the amount of gain is smaller at higher levels of quantization.

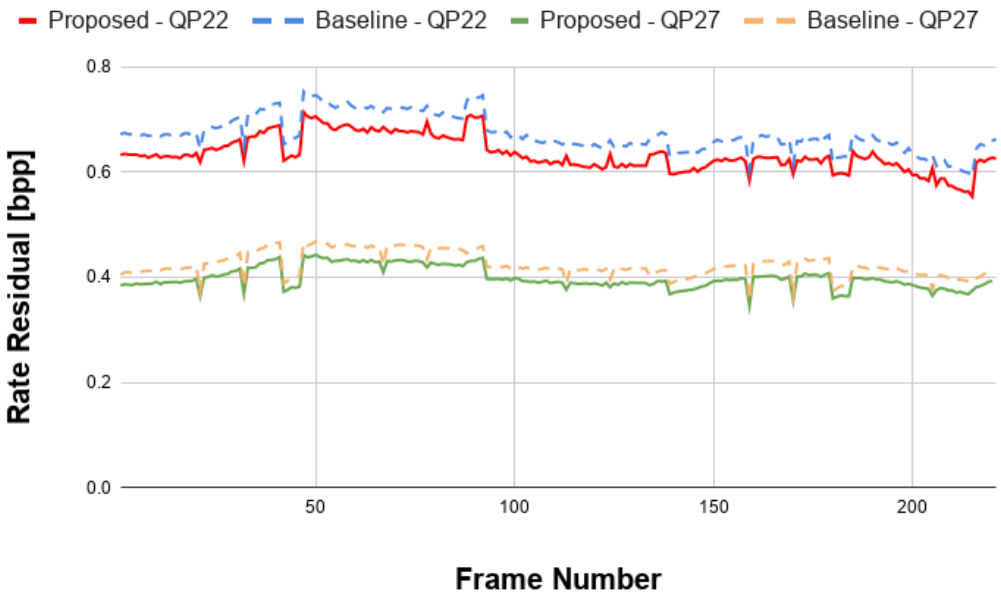

Fig. 4.5 Graph representing the rate frame-by-frame for the sequence ParkScene. The solid lines represent the rates obtained using MMCE-Net while the dotted lines correspond to the curves obtained using the H.265/HEVC. The two upper lines were obtained using a QP=22, while the lower ones use QP=27.

In this configuration we also executed an ablation test: we examined the performance obtained by applying the Dn-Cnn on the MC-frame alone, without using the previous frames. The results, shown in table 4.2, confirm the impact of the multi-frame approach.

Figure 4.6 shows some examples of the kind of enhancement the network provides and how the residual changes before and after the enhancement process. Thanks to MMCE-Net the impact of the artifacts generated by the motion-compensation algorithm used by H.265/HEVC MC-frames is reduced.

Table 4.2 This table displays the results of the ablation test in which the warped frames are not used and the enhancement is done using only the MC-frame

Sequence Name	**BD Rate** [%]
BasketballDrive	-2.96
ParkScene	-0.54
Kimono1	-2.15
Cactus	-1.39
BQTerrace	-1.44

Fig. 4.6 Examples of outputs from the network. Starting from the left: the target frame, the output of H.265/HEVC motion-compensation, the output of MMCE-Net, the residual generated from H.265/HEVC motion-compensation and the residual generated by the use of MMCE-Net.

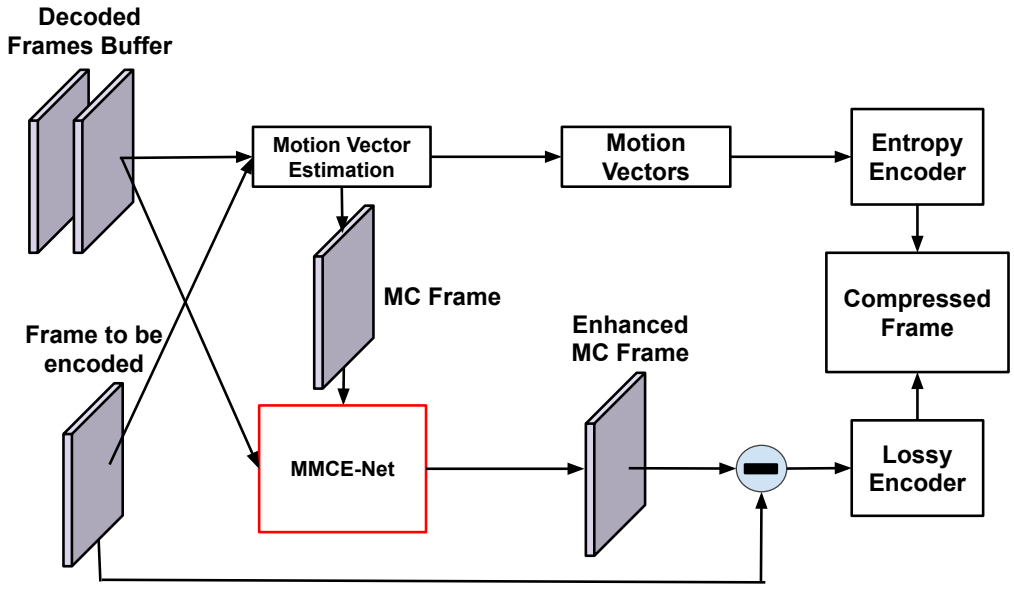

Fig. 4.7 This scheme illustrates how the proposed architecture is integrated into the flow of a video coding algorithm.

4.4.2 Integration inside the H.265/HEVC reference model

As mentioned in the introduction the network was implemented as an optional enhancement for the motion-compensated frame generated by H.265/HEVC (see Fig 4.7).

The network was integrated inside HM (version 16.22) [53]. After examining the inner working of H.265/HEVC we had to abandon the idea of applying MMCE-Net to the full motion-compensated frame. This is because in H.265/HEVC the residual is computed and encoded for each coding unit, and the processing of the following coding units is dependent on the previous ones. For this reason, it is not possible to obtain the full motion-compensated frame without also having already encoded the residual. To allow the integration of MMCE-Net inside H.265/HEVC, we had to apply it on the coding units separately, rather than on full frames, as we did on the previous set of tests. This was not a problem in terms of implementation, but it reduced the gains achieved by the network in the previous tests. This is because the network cannot correct artifacts occurring at the borders between frames coding units and it can only correct the distortions which are due to the effect of motion-compensation inside each coding unit.

After a few experiments, we observed that most of the improvement was experienced by using CU with dimensions 64×64, which is the largest possible size allowed by the codec. Using it on CUs of other sizes produced minor gains on most sequences.

For efficiency reasons, the scheme was only applied on the Y channel during processing, because when we tried to use it on the Cb and Cr channels (the chrominance values) it gave only minor performance improvements.

MMCE-Net was integrated into HM as an additional option for the compression algorithm. In this way, it is used only when it brings an improvement in terms of rate-distortion. In particular, the criteria chosen to determine whether the network is selected or not was the evaluation of how the residual changes before the after the enhancement of the MC-Frame; a smaller variance of the residual leads to a bit-rate reduction, for the same quality of the decoded frame. For simplicity, the variance has been approximated using the SAD metric. If the distortion does not decrease, the enhancement scheme is not chosen, and the original motion-compensation is kept.

Since MMCE-Net was implemented as an additional mode, it is necessary to send information to the decoder to communicate whether it was used or not on a certain block. A binary flag was inserted for the purpose of signaling the employment of MMCE-Net. The flags are encoded using CABAC [15], with a single context to estimate the probability distribution.

The context is initialized by setting uniform probability for the two states of the flag. The maximum level of depth of the quad-tree where the enhancement is applied is also signaled with two bits in fixed-length coding in the Sequence Parameter Set (SPS, [1]) of the encoded video. In this way it is possible to extend the use of the technique also on smaller coding units, although in our case it was applied only on CUs of size 64×64.

MMCE-Net was implemented as a python code employing the TensorFlow library [48], while the reference implementation for H.265/HEVC was written in c++, so there was not a direct way to integrate the codes in a single piece of software. For this reason, the two pieces of software were kept separate, and they were made to communicate using the UDP protocol [68].

The communication procedure that was designed is the following: MMCE-Net is launched first and waits for incoming signals on a predetermined port. Then HM

is launched, and it processes the first three frames as usual. When it comes to the fourth frame onwards the processing stops whenever a CU of dimensions 64x64 is encountered, as the processed CU in question has to be sent to MMCE-Net to undergo the enhancement procedure. The Y-channel of this CU is converted to a string of characters that are transmitted on the designed port where MMCE-Net is listening. The same is done with the co-located sections extracted from frames I_{t-2} and I_{t-3}. The 3 frames are received, reformatted into images, and processed by MMCE-Net which gives as output the enhanced CU. The enhanced CU is then transmitted back to HM, where execution resumes. At this point HM compares the quality of the enhanced CU against the original one and decides which one to choose, setting the associated flag accordingly.

4.5 Performance analysis

4.5.1 Comparison against H.265/HEVC baseline

The performance of the network was tested on the test sequences provided by JVET [52]. As explained before, our network was designed to enhance only in the uni-prediction scenario, so our tests were done on the HM Low-Delay P configuration (for details see [52]), which uses only uni-prediction for all the frames.

Table 4.3 compares MMCE-Net integrated in the standard with the baseline HM (version 16.22) using the Bjøntegaard metric [6].

Depending on the sequence, the amount of gain in performance given by the employment of the MMCE-Net architecture changes significantly. The classes B and E seems the one in which more advantages are seen. The greatest peaks are seen for sequences BQTerrace with a BD-Rate of -7.49% and Johnny, where we see a drop of -6.08%. In the other classes of sequences the gains are less significant, but we nevertheless see that MMCE-Net manages to beat the performance of vanilla H.265/HEVC, even for problematic sequences like PeopleOnStreet and BasketballPass.

Most of the reduction in the bit-rate is observed with values for the QP which are lower: (QP Levels $= 22, 27$), while the effects are less strong with higher values. On the B-class, the network reaches an average reduction of 4 % when the QP = 22.

Table 4.3 BD-rate between MMCE-Net and HM-16.2 in the Low-Delay P configuration.

Class	Sequence	fps	BD-Rate (%)
A - 2560 × 1600	PeopleOnStreet	30	-0.37
	Traffic		-1.75
B - 1920 × 1080	BQTerrace	60	-7.49
	BasketballDrive	50	-1.47
	Cactus		-1.22
	Kimono	24	-2.75
	ParkScene		-0.67
C - 832 × 480	BQMall	60	-1.28
	BasketballDrill	50	-0.44
	PartyScene		-0.71
	RaceHorsesC	30	-0.60
D - 416 × 240	BQSquare	60	-1.08
	BasketballPass	50	-0.12
	BlowingBubbles		-0.38
	RaceHorses	30	-0.12
E - 1280 × 720	FourPeople	60	-1.04
	Johnny		-6.08
	KristenAndSara		-2.77
Total Average			-1.69

For QP = 37 goes to an average value of -2%. For class E the phenomenon is more extreme: at QP=22 the BD-rate reaches negative 7%, which shrinks to less than 1% for QP = 37.

Fig. 4.8 shows some examples of the kind of outputs generated by MMCE-Net. It seems that most of the gain corresponds to regions affected by block artifacts connected to the proces of motion-compensation. The network focuses on such areas and tries to mitigate the effects of the artifacts on the entropy of the residual by either blurring or by estimating more plausible values for the pixels given the context of the frame.

Most of the gain is seen in high-resolution sequences, while the improvements are thinner in the smaller resolutions, especially when the sequence contains large amounts of movement. We conjecture that this is motivated by the fact that at low-resolutions CUs of size 64×64 are rare, especially in the regions with a lot of movement, which require partition in smaller blocks. In high-resolution videos CU with the required dimensions are much more common. Also one must consider that when a sequence of frame has fast movement in it, the frames tend to be less

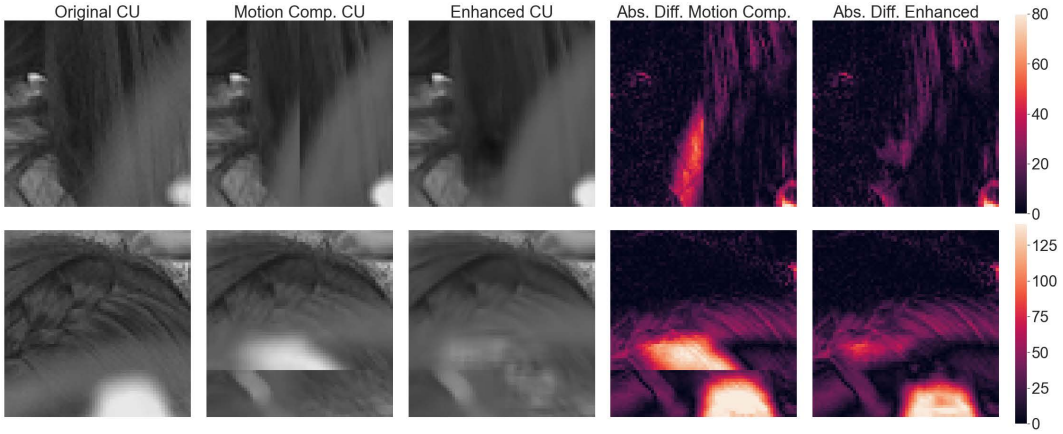

Fig. 4.8 Examples of the enhanced blocks generated by MMCE-Net. Using this architecture there is a reduction of artifacts such as the block-shaped borders resulting from the use of the motion-compensation algorithm.

correlated between each other (due to a larger amount of change between frames), and are thus less helpful for the enhancement procedure.

4.5.2 Ablation Tests

To assess the contributions of the various components of MMCE-Net, several ablation tests were performed. For these tests, variations in the architecture were obtained by removing the various constituent parts.

Four alternatives were constructed: a version in which the whole multiframe structure is discarded and the Dn-CNN is applied directly to the MC-frame using the original frame as a label, denominated "*Single Frame*". A version in which the other two past frames are employed but the registration is not carried out (so the de-artifacting network is applied to the stacking of the MC-frame plus $\hat{\mathrm{I}}_{\mathrm{t\text{-}2}}$ and $\hat{\mathrm{I}}_{\mathrm{t\text{-}3}}$ without any warping applied); this network was denominated "*Multiframe No Warping*". Another version in which the whole network (optical flow and warping included) is used, but with one previous frame instead of two. It was called "*Single Warped Frame*". A final version is denoted as "*Enhancement Always On*", which applies the enhancement to all blocks without using any rate-distortion optimization.

All the networks were trained until convergence using the same procedure followed for the original network.

Table 4.4 BD-Rates (%) for the various ablation tests.

Class	Sequence	**MMCE-Net**	Single Frame	Single Warped Frame	Multiframe No Warping	Enhancement Always On
B - 1920 × 1080	ParkScene	**−0.67**	0.00	−0.04	−0.07	−0.36
	Kimono	**−2.75**	−2.00	−1.41	−1.65	−2.19
C - 832 × 480	BQMall	**−1.28**	−0.16	−1.13	−0.27	−1.22
	PartyScene	−0.71	−0.43	−0.74	**−0.77**	−0.61
D - 416 × 240	BQSquare	−1.08	−0.28	−1.30	**−1.40**	−1.04
	BlowingBubbles	−0.38	−0.10	−0.37	**−0.69**	−0.34
E - 1280 × 720	Johnny	**−6.08**	−1.57	−2.73	−3.02	−5.60

Table 4.4 displays the performance of the ablations.

Single frame enhancement

First, we wanted to test the advantages of using a multiframe structure for enhancement. This is done by using only the Dn-CNN part of the architecture and applying it directly to the motion-compensated frame, in a mode we denominated "*Single Frame*".

As expected, the performance of this version of the network is severely lacking compared to the MMCE-Net since the previous frames provide additional information which is quite useful for the reconstruction of parts of MC-frames affected by artifacts. As it can be observed in Table 4.4, the "*Single Frame*" category has results which are significantly surpassed by every other version of the network.

Removal of one previous frame

The second aspect we wanted to test was the impact of the number of previous frames inside the structure of MMCE-Net. In this ablation, we trained the network to work with only the $\hat{\mathrm{I}}_{\mathrm{t\text{-}2}}$ warped reference frame. This version of the network was denominated "*Single Warped Frame*". In this case, the results more closely resemble the ones of the proposed network as can be seen in Table 4.4, but the addition of the further previous frame gives a small edge in terms of performance.

Removal of frame registration

We then tested the impact of the registration mechanism using optical flow and the warping module. This was achieved by building a version of the network (denoted as "*Multiframe No Warping*") in which the previous frames were fed to the enhancement network as they are, stacking them to the MC-frame without applying any warping.

On the tested sequences, it can be observed that for classes B and E (which correspond to the resolutions of 1080×1920 and 720×1280 respectively) the absence of the registration creates a large drop in the compression performance. Instead, in classes C and D, where the enhancement network tends to be less effective anyway, the removal of the warping pre-processing seems to reduce the BD-rate by a smaller amount. This can be explained by the fact that at small resolutions (since class C and class D contain 832×480 and 416×240 sequences respectively) the size of the CUs have dimensions which are similar to the size of objects in movement. This is problematic because it means that the MC-frame can be radically different compared to the previous frames used for the enhancement. The registration step, when employed in cases such as this, generates warped frames which are very unnatural and distorted, which are not helpful for the enhancement. These issues could possibly be made less impactful by training different networks for the different resolutions.

MC-frame enhancement used on every coding unit

A further configuration examines how the enhanced predicted frame is used inside the H.265/HEVC standard. In this ablation, denoted as "*Enhancement Always On*", we tested the possibility of applying the enhancement to all blocks, doing away with the mechanism for determining the amount of distortion before and after it. This means that the signaling procedure can be skipped, because it is not necessary to send the side information connected to the flags. The down-side is that in this configuration MMCE-Net is used also on-blocks where MMCE-Net does not provide an improvement.

As can be observed in Table 4.4, after this modification the performance did not massively drop, especially for the C and D classes of videos. The greatest difference is for the sequence Kimono (where there is an increase of 0.55% in BD-Rate). This can be explained by the fact that the enhancement network is selected almost always

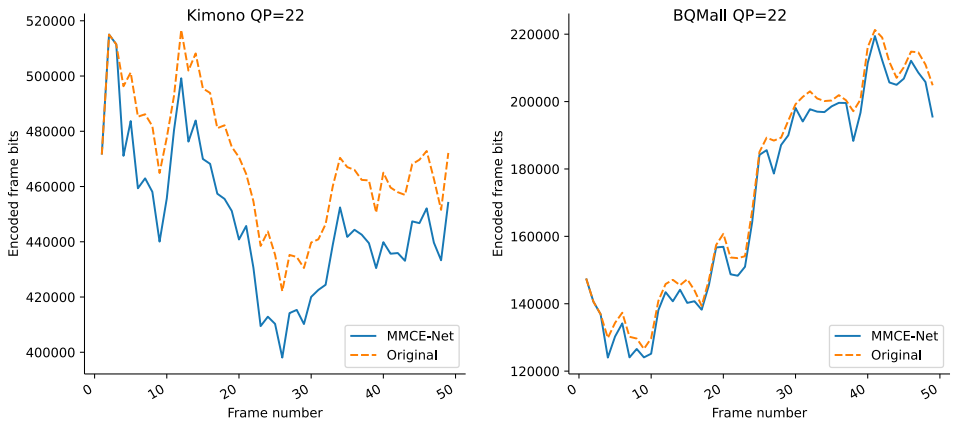

Fig. 4.9 Frame-by-frame comparison of the dimensions of the encoded stream before and after the enhancement for the sequences Kimono and BQMall using the ablation network "*Enhancement Always On*". The encoding was done using quantization level QP=22.

and thus the implementation of the network as an optional mode gives only a small benefit.

Class	Sequence	MMCE-Net Success Rate (%)
B - 1920×1080	ParkScene	83.26
	Kimono	77.39
C - 832×480	BQMall	85.74
	PartyScene	84.18
D - 416×240	BQSquare	66.62
	BlowingBubbles	89.53
E - 1280×720	Johnny	73.33
Total Average		80.01

Table 4.5 Percentage of 64×64 CU enhanced by MMCE-Net using the "*Enhancement Always On*" configuration. The count was carried out on the first 50 frames of each sequence using quantization level QP=22.

Figure 4.9 shows a few examples where we can observe that the proposed technique usually introduces an overall improvement on the frames of the sequence where it is applied and very rarely yields a worse estimate compared to the original motion-compensated frame. This is confirmed at the coding unit level in Table 4.5 where it can be observed that MMCE-Net produces a gain in terms of accuracy of the CUs 80% of the times. Furthermore, Fig. 4.10 shows a scatter plot of the quality of the original MC-frame (horizontal) versus the enhanced predicted frame (vertical) in

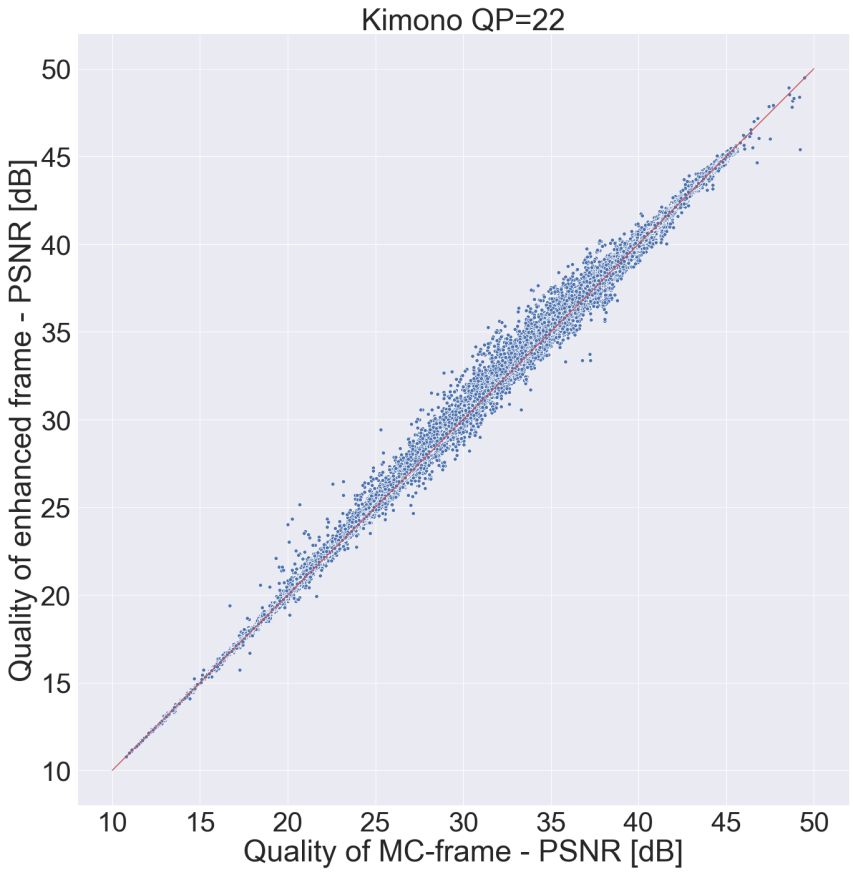

Fig. 4.10 This scatter plot illustrates how the MMCE-Net changes the quality in prediction on a block-by-block basis. The axes report the PSNR between the original frame and the MC-frame (horizontal) and between the original frame and the enhanced frame (vertical). These results were produces using QP=22.

terms of the PSNR between the predicted and the original frame. The results suggest that even in cases where MMCE-Net reduces the accuracy of a motion-compensated CU (represented in the figure by the points below the diagonal), the losses are very small.

Conclusions and Future Work

In this paper, we introduced a deep neural network methodology to improve the compression performance of the H.265/HEVC video coding standard, by improving the accuracy of the motion-compensation step of the algorithm. While it was tested for H.265/HEVC, the proposed technique does not depend on this particular standard,

and conceptually could integrated in other codecs (for example H.264/AVC and H.266/VVC)

MMCE-Net improves the performances of the standard in a significant manner on all the test sequences provided by JVET, with peaks of peformance for the B and E classes. Future work will concern the extension of this technique for bi-directional prediction and enabling testing on other configurations besides Low-Delay P, especially at high quality levels. Other avenues for extending the technique are the investigation of different methods for the fusion of the frames (such as in [60]) or introducing memory elements like in [69] and [70].

Chapter 5

Synthetic Aperture Radar Raw Data Compression

5.1 Introduction

In the following two chapters we examine the works we have done in the field of compression for remote sensing applications. In particular, in this chapter we describe our work on Synthetic Aperture Radar (SAR) raw data compression, in connection with the H2020 project EO-ALERT [7]. The work was previously published in [71].

5.1.1 EO-ALERT

The EO-ALERT project has introduced a next-generation data processing chain designed for the purpose of Earth Observation (EO) using satellites. An overview on the project can be found in [72].

In the usual scenario, the orbiting satellite collects the data from its sensors on-board and transmits them to the ground segment with limited processing, which is mainly taken care of on ground. There is great latency on EO applications between the moment in which the data is captured and when the final product on ground is obtained. Even in the most modern applications, this can reach values in the 15-30 minutes range. The reason for this large delay is the limited throughput in the

channel between the satellites and ground segments, in conjunction with the large amount of raw data that needs to be transmitted before it can be used to form the final products on-ground. The latency is especially problematic for tasks which require real time observations.

The goal of EO-ALERT is to develop a new architecture for data processing to reduce the latency below the 5 minutes threshold, with most communications having around 1 minute of latency. This is achieved by moving the generation of processed products on-board the satellite by taking advantage of the increased capabilities of currently available hardware. The generated products contain much less redundancy and thus reduce the latency needed to transmit them to Earth. In particular one objective is to generate alerts for time-sensitive tasks such as ship detection and extreme weather monitoring (hence the name EO-ALERT).

5.1.2 SAR raw data compression

EO-ALERT was designed to capture and process both optical and SAR data. Even though the stated purpose of the architecture is to process data on-board the satellite, it is nevertheless necessary to send the raw data acquired by the sensors to the ground segment, so that they can be stored and analyzed with greater precision on Earth at a later moment. Due to hardware constraints on-board, low-complexity compression algorithms are required for this task. Concerning the design of the optical data compression algorithm the CCSDS 123.0-B-2 standard in [9] is a natural choice. The standard has been designed for lossless and near-lossless compression of multi-spectral images in the remote sensing setting and already satisfies the constraints of the on-board architecture.

Compared to optical data, compression of SAR raw data is much more challenging. SAR is a form of active data collection, which means that the data is captured by emitting high-frequency signals and by analyzing the echoes generated by the environment in response to the stimulus. The echoes are detected, and the quadrature and in-phase components of the signal are sampled and stored as complex numbers. As it can be observed in Fig. 5.1, these samples are very uncorrelated, and compressing them using a low-complexity algorithm is challenging. Please note that the underlying scene is not directly visible from SAR raw data. To visualize the data

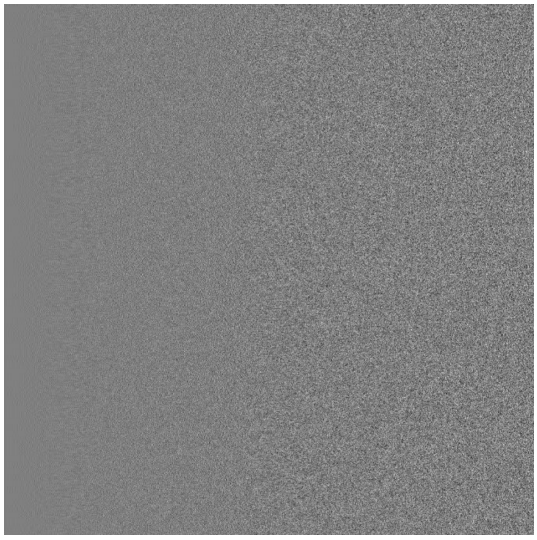

Fig. 5.1 Example of the real part of a raw SAR data capture. The samples have low correlation and are usually modeled as a Gaussian random process with slowly varying variance

in a way that the eye can interpret, SAR raw data have to undergo a process called "focusing" [73].

Historically, the most commonly used method to tackle the task of SAR raw data compression is Block Adaptive Quantization (BAQ) [10]. BAQ models the captured data as a random process $s \in \mathbb{C}^{H \times W}$ where the complex samples have independent real and imaginary parts that are distributed according to a zero-mean Gaussian distribution with a slowly-varying variance.

$$\Re\{s_{i,j}\}, \Im\{s_{i,j}\} \sim \mathcal{N}(0, \sigma_{i,j}), \forall i, j \in (H \times W) \tag{5.1}$$

Using this model, compression is simply achieved by partitioning the data in blocks where the process can be modeled as stationary i.e. $\sigma = const.$. Then the local variance of the block is estimated and used to set the threshold levels for a Max-Lloyd non-uniform quantizer (designed to minimize the MSE for signals with gaussian distribution). The resulting quantization indeces plus the variances for the blocks compose the final compressed file.

Through the years several evolutions of this technique were proposed, like Block Adaptive Vector Quantization [74] which introduces vector quantization, and Entropy Constrained Block Adaptive Quantization [75] which introduces a low complexity entropy coder in the architecture. Other methods employ transforms to

take advantage of the correlations between values and map the signal to a domain where compression is easier. For example, [76] employs the wavelet transform, while [77] compares the effects of using the Fourier, discrete cosine and the Hadamard transforms. All these evolutions bring improvements in terms of rate-distortion but have the downside of increasing the complexity.

In this chapter, we investigate the viability of using the CCSDS 123.0-B-2 standard for the purpose of SAR raw data compression.

5.1.3 Motivations for the use of CCSDS on SAR raw data

Even though CCSDS 123.0-B-2 was designed for the compression of optical images, its utilization on SAR raw data is justified. In essence, CCSDS 123.0-B-2 can be characterized as a Differential Pulse Code Modulation scheme (DPCM) with a quantizer in the loop. The algorithm generates a prediction of the value of a pixel for a certain spectral channel based on spatially neighboring pixels in the same spectral channel and in adjacent spectral channels. The compressed sample is then generated by applying quantization and entropy coding on a residual $\delta_{i,j}$ between the input and the prediction.

$$\delta_{i,j} = s_{i,j} - \hat{s}_{i,j} \tag{5.2}$$

The effectiveness of using a DPCM-based algorithm was shown in the paper [8], in which a technique of this kind is shown to improve the performance in terms of rate-distortion by taking advantage of the correlation between samples without massively increasing complexity. For this reason, we deemed CCSDS 123.0-B-2 a reasonable candidate for this task.

Since this algorithm is already employed for the purpose of optical image compression, re-using it also for SAR raw data would avoid adding further complexity on the processing chain on-board the satellite.

5.2 Proposed method and experimental results

CCSDS 123.0-B-2 was applied to the SAR raw data by separating the real and imaginary parts and processing them as two separate images. CCSDS 123.0-B-2 works with positive values so an offset equal to half of the dynamic range of the input data is added to the input to make the samples positive. Two settings were examined: compression of the SAR raw data after normalization, and compression without pre-processing.

5.2.1 Dataset

The experiments were conducted using 3 raw data acquired by the SIR-C/X-SAR mission [78], denoted as *Innsbruck* (2001×2136 complex samples), *Jesolo* (2001×2469) and *Matera* (2001×4000). Since these images were received from a real satellite, they were already compressed, using 6 bits for the quantization of the image *Innsbruck* and 4 bits for the images *Jesolo* and *Matera*.

5.2.2 Experimental setup

In all tests CCSDS 123.0-B-2 was set up using the parameters listed in table 5.1: The fields of the table refer to parameters that were not described in the background chapter. The explanation of these parameters can be found in [20]. All the results were compared to the BAQ algorithm using a partition in blocks of size 32×32 on which the random process is assumed stationary. The comparison was made in terms of distortion, by fixing the compression rate to the same value for both systems. Since the dataset is constituted by images already compressed with bit-rates up to 4 bpp, we tested only lower bit rates, i.e., 2 bpp and 3 bpp.

For the BAQ algorithm fixing the bit rate is simply a matter of changing the number of bits used for the quantization. On the other hand, to match these bit-rates for the files compressed using CCSDS 123.0-B-2 we had to make multiple attempts by tuning the value called maximum absolute error (MAE). This parameter sets an upper bound on the distortion on the reconstructed files for each sample.

The distortion was evaluated by measuring the SNR between the original SAR data, and the one after decompression. The formula used to estimate the SNR is:

$$SNR[dB] = 10\log_{10}\left(\frac{\sum_{i=0}^{H}\sum_{j=0}^{W} s_{i,j}^2}{\sum_{i=0}^{H}\sum_{j=0}^{W} r_{i,j}^2}\right) \tag{5.3}$$

where r is the difference between the original SAR samples and the reconstructed ones.

5.2.3 Normalized SAR raw data

In a first experiment, SAR raw data were normalized in the same way as in the BAQ baseline, using a block size of 32×32.

The standard deviation was estimated on the different blocks and the blocks were normalized to have a standard deviation equal to 1. The data was then uniformly quantized at either 16 bpp or 8 bpp, using as maximum and minimum values the range $\{-4\sigma, 4\sigma\}$. This was done to have the normalized SAR raw data in the same integer format as optical data, so that the same compression algorithm can be used without modifications on both kinds of data. The value 16 bpp was chosen because

Table 5.1 Compression parameters for the experiments.

Parameter	**Value**
Number of bands for prediction	$P = 3$
Register size (in bits)*	$R = 64$
Weight resolution*	$\omega = 19$
Weight update scaling exponent change interval*	$t_{inc} = 64$
Initial weight update scaling exponent parameters*	$v_{min} = -1$
Final weight update scaling exponent parameters*	$v_{max} = 3$
Prediction mode	Full Wide/Neighbor Oriented
Sample representative parameters	All set to 0
Error limit	Absolute and Non-band dependent
Encoder	Sample adaptive
Unary length limit	$U_{max} = 18$
Initial count exponent*	$\gamma_0 = 1$
Accumulator initialization constant*	$K = 3$
Rescaling counter size*	$\gamma* = 6$

it is the maximum dynamic range used for optical images. The value 8 bpp was used as a reasonable trade-off between resolution and complexity.

The results of these tests (listed in Tables 5.2, 5.3), show that both at 2 and 3 bpp the files compressed using CCSDS 123.0-B-2 are reconstructed with levels of distortion in terms of SNR which are slightly higher than the ones achieved using BAQ. These performance seem to not be affected by the number of bits used to represent the input as there is only slight variation when we change between 16 and 8 bits.

Image	Input Dyn. Range	SNR BAQ [dB]		SNR CCSDS [dB]		MAE
		Real	Imag.	Real	Imag.	
Innsbruck	16 [bpp]	9.85	9.80	8.99	9.13	5.1×10^3
	8 [bpp]			8.74	8.89	2.0×10^1
Jesolo	16 [bpp]	8.88	8.50	8.56	8.41	5.9×10^3
	8 [bpp]			8.77	8.61	2.2×10^1
Matera	16 [bpp]	9.88	9.85	8.01	9.70	5.8×10^3
	8 [bpp]			8.07	9.77	2.2×10^1

Table 5.2 Comparison in terms of SNR between BAQ and CCSDS 123.0-B-2 with normalized input at bitrate 2 bpp

Image	Input Dyn. Range	SNR BAQ [dB]		SNR CCSDS [dB]		MAE
		Real	Imag.	Real	Imag.	
Innsbruck	16 [bpp]	15.40	15.30	15.53	15.68	2.4×10^3
	8 [bpp]			15.42	15.56	9.0×10^1
Jesolo	16 [bpp]	13.03	12.75	15.03	14.88	2.8×10^3
	8 [bpp]			15.39	15.23	1.0×10^1
Matera	16 [bpp]	14.52	14.48	14.88	16.22	2.8×10^3
	8 [bpp]			14.70	16.60	1.0×10^1

Table 5.3 Comparison in terms of SNR between BAQ and CCSDS 123.0-B-2 with normalized input at bitrate 3 bpp

5.2.4 SAR raw data with no pre-processing

In a second experiment, SAR raw data were fed to the CCSDS 123.0-B-2 algorithm with no pre-processing. The rationale of this second experiment is that the block-wise

normalization is actually not needed, since CCSDS 123.0-B-2 can adapt to changes in the distribution of the inputs thanks to its prediction mechanism, which estimates the sample based on local observations. Skipping the pre-processing step is also advantageous as it further reduces the complexity of the architecture.

For this set of experiments, it was not possible to match the exact bit-rates of BAQ as in the previous experiments. This is because the input data is already represented on a small number of bits for every sample (6 bits for Innsbruck, 4 bits for Jesolo and Matera), and small changes of MAE produce large changes in the bit rate, with the bit rate rapidly dropping to 1 bpp for every small values of MAE.

As it can be observed in tables 5.4 and 5.5, this approach outperforms BAQ with gains up to 2 dB in SNR when we compress at a bit-rate of 2 bpp and achieving lossless compression at 3 bpp for the sequences Jesolo and Matera. For the sequence Jesolo there is a drop of 1 dB at bit-rate 2 bpp, which can be justified by the fact that the bit-rate was approximated very coarsely in this case.

Image	SNR BAQ [dB]		SNR CCSDS [dB]		MAE	Bit-rate CCSDS [bpp]	
	Real	Imag.	Real	Imag.		Real	Imag.
Innsbruck	9.85	9.80	11.49	11.31	4	2.06	2.08
Jesolo	8.88	8.50	7.55	7.36	1	1.73	1.72
Matera	9.88	9.85	10.45	10.50	1	1.8	1.8

Table 5.4 Comparison in terms of SNR between BAQ and CCSDS 123.0-B-2 without input normalization at bitrate 2 bpp

Image	SNR BAQ [dB]		SNR CCSDS [dB]		MAE	Bit-rate CCSDS [bpp]	
	Real	Imag.	Real	Imag.		Real	Imag.
Innsbruck	15.40	15.30	16.47	16.29	2	2.76	2.74
Jesolo	13.03	12.75	Inf.	Inf.	0	3.15	3.14
Matera	14.52	14.48	Inf.	Inf.	0	2.90	2.91

Table 5.5 Comparison in terms of SNR between BAQ and CCSDS 123.0-B-2 without input normalization at bitrate 3 bpp

5.3 Conclusion

This chapter illustrated the advantages of using the CCSDS 123.0-B-2 standard for the purpose of SAR raw data compression on-board the satellite. The main advantage of this design choice is the possibility to streamline the computing architecture by utilizing the same algorithm to compress both optical and SAR data. Furthermore, when employed without input normalization the CCSDS 123.0-B-2 compression algorithm achieves better performance than BAQ, which is the de-facto standard for SAR raw data compression.

Due to these properties, the proposed technique was implemented in the architecture of the H2020 project EO-ALERT as the reference SAR raw data compression algorithm.

Chapter 6

On-board data reduction for multi-spectral and hyper-spectral images via cloud screening

6.1 Introduction

This chapter describes the work done on the topic of cloud screening for multi-spectral and hyper-spectral applications in a remote sensing setting, for the purpose of data reduction.

As already highlighted in chapter 5, communication in the remote sensing setting is a challenging problem. The low throughput between satellites and ground stations makes the use of compression of the transmitted data necessary, but with the caveat that the compression algorithm cannot be excessively complex, due to the low capabilities of the currently available hardware on-board satellites.

Through the years several low-complexity compression algorithms were developed precisely to tackle this problem [79], but it nevertheless remains necessary to reduce the amount of data generated on-board the satellite by limiting the frequency of acquisition to whatever level the transmission channel and the data processing architecture of the satellite allows.

This situation is further complicated by the fact that a portion of the transmitted data is not usable for analysis on the ground because of clouds, which may cover

large portions of an examined region. To avoid wasting time and processing resources by transmitting useless data it can be advantageous to devise a methodology to detect clouds on-board the satellite and skip the transmission of regions obscured by them.

An approach of this kind was already proposed in [80]. In this work, the pixels obscured by clouds are detected by analyzing their multi-spectral content, and whenever the number of pixels affected inside a line of the image exceeds a threshold the whole line is not transmitted. The blocks of consecutive lines skipped are then signaled to the receiver. This approach is computationally cheap and makes the communication of which parts of the image are not transmitted very simple. The downside is that a portion of valid pixels (e.g. pixels not covered by clouds) are discarded together with the non-valid ones.

For this reason, it can be useful to design a more advanced approach that is capable to preserve all the valid pixels without increasing complexity. This chapter describes our efforts to accomplish this purpose. This work was published in [81].

6.2 Proposed methods

This work takes as starting point the already cited CCSDS 123.0-B-2 standard for lossless and near-lossless compression for multi-spectral and hyper-spectral images [9]. All of the proposed techniques were implemented using this algorithm as the backbone.

The main idea was to develop techniques that allowed to skip only the pixels actually obscured by clouds, following exactly a cloud map obtained using a cloud screening technique. We also wanted to remain compliant with the CCSDS 123.0-B-2 standard, so any receiver which employs it can still correctly decode the data.

The proposed solution was to set the non-valid (e.g cloudy) pixels to a dummy value while also sending the complete cloud mask to the receiver. The dummy values have to be chosen carefully in order to make the bit-rate necessary to encode the cloudy regions negligible. Also, the cloud masks have to be sent to the receiver to allow for the correct identification of the real values compared to the dummy ones. To accomplish this end several methods to generate the dummy values and transmit the cloud mask were examined.

6.2.1 Transmission of the Cloud Mask

Two methods for the transmission of the cloud mask were devised.

Table Insertion

This first solution takes advantage of the fact that CCSDS-123.0-B-2 gives the possibility to send, in the header of the transmitted file, up to 15 user-defined tables for side-information. The content of the cloud mask is encoded using a 1 bpp representation (valid pixels are signaled by a 1 while non-valid pixels are signaled with a 0) in a supplementary table.

This solution is simple to implement and is compliant with the standard. It has the downside of sending the cloud mask not compressed, without taking advantage of the correlation between samples present inside of it.

Band Insertion

The second approach tested was to concatenate the cloud mask as an additional color channel for the image. In this way, the compression algorithm treats the values like other regular pixels and can compress them. The color map needs to be encoded losslessly and its values are not helpful for the prediction of the rest of the pixels of the image. For this reason, it needs to be encoded separately from the rest of the image. This approach will be referred to as Band Insertion.

6.2.2 Pixel replacement

The pixels covered by clouds needed to be replaced by dummy values. The criteria for the choice of the values was the minimization of the dimension of the final compressed file. To reach this objective two methods were devised.

Pixel replenishment

The first approach attempted was to minimize the residual between the prediction and the dummy by choosing values that are easily predictable by the algorithm. For

this reason, the pixels in cloudy regions were replaced by averages of local pixels. In particular, approximating the prediction methodology of CCSDS 123.0-B-2, to generate a dummy value for a pixel in position (x,y,z) of the multi-spectral image we employ the neighboring pixels on the upper-left corner plus a pixel from the preceding color channel but at same location (x,y).

$$I(x,y,z) = \frac{1}{4} \times [I(x-1,y-1,z) + I(x-1,y,z) + I(x,y-1,z) + I(x,y,z-1)] \quad (6.1)$$

These dummy pixels are generated sequentially across the image using the same scanning order as the one which CCSDS 123.0-B-2 follows. For this reason, inside the cloudy region the generated values become averages of previously generated dummy pixels.

Zero Residual

In this second method, denominated *zero residual*, the generation of real dummy values in the cloudy region of the image is avoided. What is to act inside the compression algorithm and force the residual to zero whenever a cloudy pixel is being compressed. This is equivalent to using the prediction generated by CCSDS 123.0-B-2 as a dummy value, but this version does not require to pre-process the input image.

6.3 Experimental results

6.3.1 Dataset

The various designs were tested on the Landsat 8 ETM+ dataset [82], which provides a corresponding cloud mask for every multi-spectral image in input. This mask also contains other information, such as the position of bodies of water and rivers and the shadows generated by the clouds, but for our purposes only the cloud information was kept and was mapped on binary images. An example can be seen in Fig. 6.1.

The images are separated in three different categories based on the percentage of cloudy pixels present: *Clear* images, with less than 35 % of cloud pixels, *MidCloud*

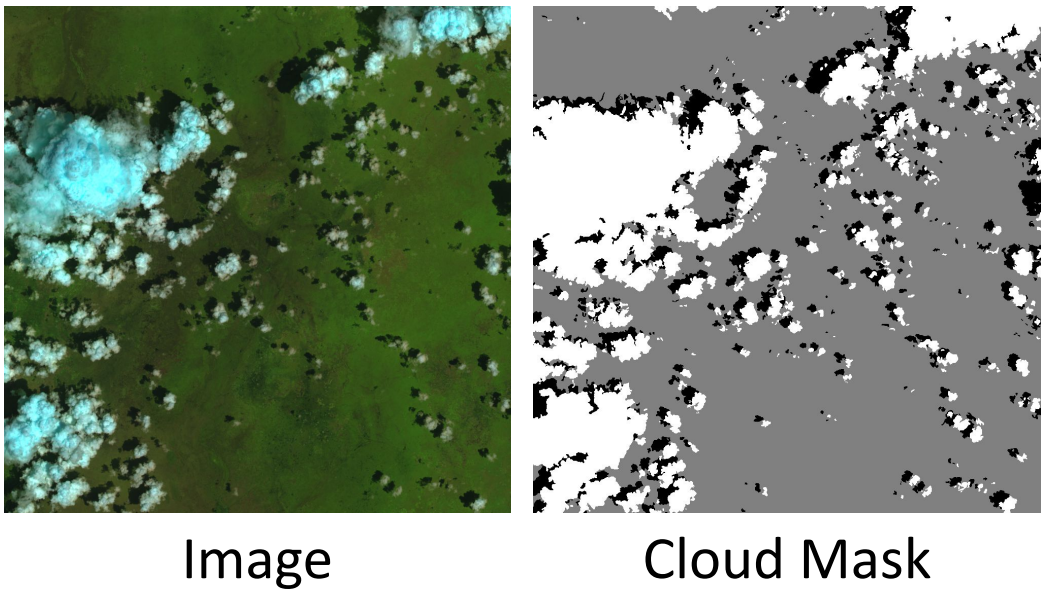

Fig. 6.1 Example of an image with the corresponding cloud mask from the Landsat 8 dataset. The mask data from the original dataset contains superfluous information, like the shadow generated by the clouds in this example. All this further information was discarded, and the cloud masks were transformed into binary value images.

for the range between 35 and 65 % and *Cloudy* for images with more than 65 % of covered images. The test set was built by selecting 13 images from the dataset selecting at random from the different categories.

This dataset has the problem of containing already processed images instead of raw captures, but since we are examining the effects of skipping a region on the dimension of the final file, it can be argued that the reduction in bit-rate would be by a comparable amount if we were working on raw images. Moreover, as illustrated in projects like the already cited EO-ALERT [7], in modern architectures some forms of processing of the acquired images are done on-board of satellites so the tested situation can be considered realistic.

Nevertheless, tests were also done using AVIRIS raw captures. These images do not have clouds and associated cloud masks, so for our experiments, we generated artificial cloud masks by taking a profile of a cloud and copying and pasting it multiple times until the desired amount of cloud coverage is reached. Since the algorithm follows the cloud mask to skip the data, we did not have to modify the optical images. An example of one of the artificial cloud masks is in Fig. 6.2.

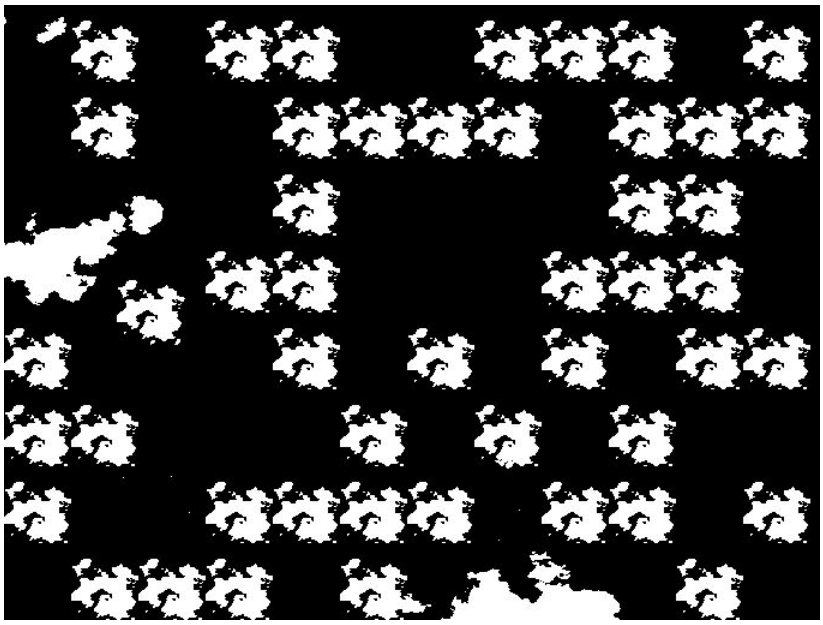

Fig. 6.2 Example of one of the artificial cloud mask that was used in the tests on the AVIRIS images.

6.3.2 Experimental setup

The four possible combinations of the two Cloud mask transmission techniques and the two pixel replacement techniques, were tested.

In the following sections these settings will be denominated:

1. **PT**: Pixel replenishment with Table insertion
2. **PB**: Pixel replenishment with Band insertion
3. **ZT**: Zero residual with Table insertion
4. **ZB**: Zero residual with Band insertion

The CCSDS 123.0-B-2 algorithm was configured with the same parameters as it was done in the chapter on SAR compression: prediction mode was set to Full Wide/Neighbor Oriented, with 3 bands used for prediction, the sample representative parameters were all set to zero and the sample adaptive entropy coder was adopted.

The algorithm was tested across multiple values for the Maximum Absolute Error $= \{0 \text{ (Lossless)}, 2, 8, 32\}$

6.3.3 Results

The methods were compared to the performance of CCSDS 123.0-B-2 compression without using the various data reduction techniques. The results for this baseline will be labeled "CCSDS" in the result tables of this chapter. The results were also be compared with the ones obtained using the technique proposed in [80].

Tests on Landsat dataset

The first tests were done on three examples extracted from the LANDSAT dataset, each with a different percentage of cloud coverage, respectively 15%, 51% and 65%. For the problem of data reduction, the gain in performance is basically related to the percentage of clouds so even if the testing set is small these results are representative of all images which have a similar percentage of cloud coverage. The results of these tests are shown in Tab. 6.1. The results are computed in bit per pixel per band (bpppb).

Table 6.1 Comparison in terms of bitrate between the different algorithms proposed (bpppb) using three examples from the LANDSAT dataset.

% Cloud	MAE	CCSDS	PT	PB	ZT	ZB	[80]
15 %	0	8.83	9.05	8.77	8.99	8.72	8.88
	2	6.52	6.74	6.46	6.68	6.41	6.57
	8	4.80	5.03	4.75	4.97	4.70	4.85
	32	3.15	3.38	3.11	3.34	3.06	3.19
51 %	0	8.20	5.98	5.68	5.74	5.45	4.30
	2	5.88	4.35	4.06	4.23	3.93	3.10
	8	4.15	3.25	2.95	3.17	2.87	2.21
	32	2.48	2.26	1.96	2.23	1.93	1.34
65 %	0	8.33	5.67	5.39	5.38	5.10	2.13
	2	6.02	4.17	3.88	4.02	3.74	1.57
	8	4.35	3.18	2.90	3.10	2.81	1.16
	32	2.80	2.32	2.04	2.28	2.00	1.68

As it can be observed on the image with 15 % cloud coverage, on low cloud percentages table insertion (used in columns PT and ZT) is not advantageous. This is because in this situation there is a small amount of possible data reduction which is not enough to compensate for the overhead needed to transmit the cloud mask in the header. In general, on all images and at every level of quantization, band insertion seem to reach superior performance over table insertion, while the choice between zero residual and pixel replenishment does not seem to affect the result significantly.

Obviously, the gain is proportional to the percentage of cloud coverage, as more data is skipped. It is interesting to observe that the gains tend to reduce with higher levels of quantization, which can be attributed to the fact that at higher compression rates the cloudy region is represented by fewer bits of the compressed file, so with their removal the gain is smaller.

Generally, the performance of [80] are higher than the ones reached by our techniques, but the fact that [80] discards non-cloudy pixels has to be taken into account.

Once we determined that whether to use zero residual or pixel replenishment does not affect the performance of the compression, we tested the performance over the full Landsat dataset. The performance were averaged across the following ranges of cloud coverage percentages = $\{0-25\%, 26-50\%, 51-75\%\}$.

In this comparison, ZB seems to be the superior method, since it always provides greater gains compared to ZT. The patterns observed in table 6.1 are confirmed in this more extensive set of experiments, with an increase in performance with the increase in the percentage of clouds, and a decrease for higher levels of quantization.

Tests on AVIRIS images

At this point we wanted to test the performance of these techniques on hyper-spectral images, to more accurately model the possible data reduction on-board of remote sensing satellites. The LANDSAT 8 dataset does not contain hyper-spectral images, and there are no available datasets that contain cloudy images with the associated cloud masks which are also hyper-spectral. For this reason, we decided to use a dataset of hyper-spectral images (captured by the AVIRIS sensor [83]) and decided to manually add clouds to create the desired setting.

Table 6.2 Comparison in terms of bitrate on the full Landsat dataset for algorithms Zero residual with Table insertion and Zero residual with Band insertion

% Cloud	*CCSDS*	**ZT**	**ZB**
0−25%	0	0.17	0.46
	2	0.11	0.39
	8	0.03	0.32
	32	-0.15	0.20
26−50%	0	1.53	1.82
	2	1.15	1.44
	8	0.79	1.08
	32	0.34	0.62
51−76%	0	2.72	3.02
	2	1.86	2.15
	8	1.16	1.45
	32	0.41	0.70

This was done by selecting clouds already present in the images and by copying and pasting them multiple times across the image until the desired percentage of cloud coverage is reached. Three images were generated using this methodology, with cloud coverage percentages of $5\%, 24\%, 40\%$. The tests were then conducted on these generated images, with the results shown in table 6.3.

Due to the higher number of channels of these images (the AVIRIS capture has 220 color channels), table insertion is a viable approach in this setting, even at low percentages of cloud coverage. This is because the overhead due to the transmission of the cloud mask is negligible compared to the total size of the file, and even the removal of a small region can compensate for it. This means that in the hyperspectral setting all the proposed methods are basically equivalent for the purpose of data reduction.

6.4 Conclusions

In this chapter we illustrated our work on the topic of data reduction through cloud screening for hyper-spectral images. The techniques proposed are capable of reducing the data transmitted by only discarding the pixels which do not provide

Table 6.3 Comparison in terms of bitrate between the different algorithms proposed (bpppb) using three examples from the AVIRIS dataset.

% Cloud	**MAE**	**CCSDS**	**PT**	**PB**	**ZT**	**ZB**	**[80]**
5%	0	5.83	5.77	5.75	5.77	5.75	5.86
	2	3.55	3.49	3.47	3.49	3.47	3.58
	8	2.00	1.95	1.94	1.95	1.94	2.03
24%	0	6.43	5.64	5.63	5.65	5.63	6.44
	2	4.14	3.37	3.36	3.36	3.35	4.15
	8	2.54	1.94	1.92	1.92	1.91	2.55
40%	0	6.68	5.42	5.41	5.44	5.43	4.7
	2	4.39	3.17	3.16	3.16	3.15	3.05
	8	2.76	1.86	1.84	1.84	1.83	1.90

information to the cloud segment and are compliant with the low-complexity compression standard CCSDS 123.0-B-2.

Chapter 7

Conclusions

7.1 Overview of the work

This thesis describes several works with the shared objective of advancing the field of data compression. Three forms of data were examined: video, multi-spectral images, and SAR raw data.

Chapter 3 and chapter 4 explored the possibilities of deep learning for the purpose of video compression. First, we explored the use of CNNs for frame prediction as an alternative for the inter-prediction algorithms used in the current generation of video compression standards. Then, after meeting some limitations in this approach, we designed MMCE-Net: a network for the enhancement of the predicted frames generated by motion-compensation. This structure was successfully trained and implemented inside the video compression standard H.265/HEVC, and we were capable to obtain an average reduction of the BD-rate on the standard ITU-T test sequences of -1.69%.

The following two chapters describe our work with the standard CCSDS 123.0 B-2. In chapter 5 we tested the effects of using this standard for the compression of SAR raw data, and demonstrated that this approach achieves better performance than the de-facto standard for the task BAQ. The viability of this algorithm for this task allows to streamline the processing architecture of satellites for remote sensing, and this approach was adopted for the Horizon 2020 project EO-Alert.

Chapter 6 proposes multiple techniques to efficiently remove data related to clouds in earth observations. The proposed techniques allow to reduce the data transmitted without the loss of any useful information from the receiver side and produce compressed files which are compliant with the standard CCSDS 123.0 B-2.

7.2 Open Problems

There are a lot of avenues for future work regarding the topic of data compression, especially when speaking about the possibilities for deep learning in the field of video compression. For example, the work proposed in chapter 4 could be expanded by adapting to more configurations (for example bi-prediction), or by inserting memory elements to take advantage of long-term correlations inside video-sequences. Finally, it would be interesting to investigate the efficacy of a structure where the enhancing network is more tightly integrated in the motion-compensation algorithm. In such a scheme, a complex algorithm could decide to switch between different configurations of the motion-compensation algorithm and reduce the amount of side information needed at the receiver side, while keeping the entropy of the residual constant using the enhancing network.

References

[1] G. J. Sullivan, J. Ohm, W. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[2] Y. Lecun, *Generalization and network design strategies*. Elsevier, 1989.

[3] Y. Jo, S. W. Oh, J. Kang, and S. J. Kim, "Deep video super-resolution network using dynamic upsampling filters without explicit motion compensation," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3224–3232.

[4] D. Sun, X. Yang, M. Liu, and J. Kautz, "PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8934–8943.

[5] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3142–3155, 2017.

[6] G. Bjontegaard, "Calculation of Average PSNR Differences between RD-curves," 2001.

[7] M. Kerr, S. Tonetti, S. Carnara, J. I. Bravo, R. Hinz, A. Latorre, F. Membibre, A. Ramos, S. Wiehle, O. Koudelka, E. Magli, R. Freddi, S. Fraile, and C. Marcos, "Eo-alert: A satellite architecture for detection and monitoring of extreme events in real time," in *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS*, 2021, pp. 168–171.

[8] E. Magli and G. Olmo, "Lossy predictive coding of sar raw data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 41, no. 5, pp. 977–987, 2003.

[9] M. Hernández-Cabronero, A. B. Kiely, M. Klimesh, I. Blanes, J. Ligo, E. Magli, and J. Serra-Sagristà, "The ccsds 123.0-b-2 "low-complexity lossless and near-lossless multispectral and hyperspectral image compression" standard: A comprehensive review," *IEEE Geoscience and Remote Sensing Magazine*, vol. 9, no. 4, pp. 102–119, 2021.

[10] R. Kwok and W. Johnson, "Block adaptive quantization of magellan sar data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 27, no. 4, pp. 375–383, 1989.

[11] V. Sze, M. Budagavi, and G. J. Sullivan, *High Efficiency Video Coding (HEVC): Algorithms and Architectures*. Springer Publishing Company, Incorporated, 2014.

[12] "Advanced Video Coding for Generic Audio-Visual Services, ITU-T Rec. H.264 and ISO/IEC 14496-10 (AVC), ITU-T and ISO/IEC JTC 1, May 2003 (and subsequent editions)."

[13] "Versatile Video Coding, Standard ISO/IEC 23090-3, ISO/IEC JTC 1, Jul. 2020."

[14] "Video Codec for Audiovisual Services at px64 kbit/s, ITU-T Rec. H.261, version 1: Nov. 1990, version 2: Mar. 1993."

[15] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the h.264/avc video compression standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 620–636, 2003.

[16] "IASI level 1: Product guide, EUMETSAT, Tech. Rep. EUM/OPS-EPS/MAN/04/0032, Darmstad, Germany," 2019.

[17] M. J. Khan, H. S. Khan, A. Yousaf, K. Khurshid, and A. Abbas, "Modern trends in hyperspectral image analysis: A review," *IEEE Access*, vol. 6, pp. 14 118–14 129, 2018.

[18] J. Theiler, A. Ziemann, S. Matteoli, and M. Diani, "Spectral variability of remotely sensed target materials: Causes, models, and strategies for mitigation and robust exploitation," *IEEE Geoscience and Remote Sensing Magazine*, vol. 7, no. 2, pp. 8–30, 2019.

[19] M. Parente, J. P. Kerekes, and R. Heylen, "A special issue on hyperspectral imaging [from the guest editors]," *IEEE Geoscience and Remote Sensing Magazine*, 2019.

[20] "Low-Complexity Lossless and Near-Lossless Multispectral and Hyperspectral Image Compression, Consultative Commitee for Space Date Systems (CCSDS) Standard CCSDS 123.0-b-2, Feb. 2019. [online]. Available: https://public.ccsds.org/Pubs/123x0b1ec1s.pdf."

[21] D. Valsesia and E. Magli, "High-throughput onboard hyperspectral image compression with ground-based cnn reconstruction," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 12, pp. 9544–9553, 2019.

[22] I. Blanes, A. Kiely, M. Hernández-Cabronero, and J. Serra-Sagristà, "Performance impact of parameter tuning on the ccsds-123.0-b-2 low-complexity lossless and near-lossless multispectral and hyperspectral image compression standard," *Remote Sensing*, vol. 11, p. 1390, 06 2019.

[23] "Lossless Data Compression, Consultative Commitee for Space Date Systems (CCSDS) Standard CCSDS 121.0-b-2, Apr. 2012. [online]. Available: https://public.ccsds.org/Pubs/121x0b1ec3s.pdf."

[24] I. Blanes, A. Kiely, M. Hernández-Cabronero, and J. Serra-Sagristà, "The hybrid entropy encoder of ccsds 123.0-b-2: Insights and decoding process," in *7th International Workshop on On-Board Payload Data Compression (OBPDC)*, September 2020, pp. 1 – 10.

[25] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016, http://www.deeplearningbook.org.

[26] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?" in *2009 IEEE 12th International Conference on Computer Vision*, 2009, pp. 2146–2153.

[27] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[28] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[29] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 12 2014.

[30] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[31] S. Ma, X. Zhang, C. Jia, Z. Zhao, S. Wang, and S. Wang, "Image and video compression with neural networks: A review," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 6, pp. 1683–1698, 2020.

[32] Z. Chen, T. He, X. Jin, and F. Wu, "Learning for video compression," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 2, pp. 566–576, 2020.

[33] J. Lin, D. Liu, H. Li, and F. Wu, "M-lvc: Multiple frames prediction for learned video compression," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 3543–3551.

[34] J. Han, S. Lombardo, C. Schroers, and S. Mandt, "Deep probabilistic video compression," *CoRR*, vol. abs/1810.02845, 2018. [Online]. Available: http://arxiv.org/abs/1810.02845

[35] T. Li, M. Xu, C. Zhu, R. Yang, Z. Wang, and Z. Guan, "A Deep Learning Approach for Multi-Frame In-Loop Filter of HEVC," *IEEE Transactions on Image Processing*, vol. 28, no. 11, pp. 5663–5678, 2019.

[36] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2261–2269.

[37] S. Zhang, Z. Fan, N. Ling, and M. Jiang, "Recursive residual convolutional neural network- based in-loop filtering for intra frames," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 7, pp. 1888–1900, 2020.

[38] J. Li, B. Li, J. Xu, R. Xiong, and W. Gao, "Fully connected network-based intra prediction for image coding," *IEEE Transactions on Image Processing*, vol. 27, no. 7, pp. 3236–3247, 2018.

[39] H. Sun, L. Yu, and J. Katto, "Fully neural network mode based intra prediction of variable block size," in *2020 IEEE International Conference on Visual Communications and Image Processing (VCIP)*, 2020, pp. 21–24.

[40] L. Zhu, S. Kwong, Y. Zhang, S. Wang, and X. Wang, "Generative adversarial network-based intra prediction for video coding," *IEEE Transactions on Multimedia*, vol. 22, no. 1, pp. 45–58, 2020.

[41] L. Zhao, S. Wang, X. Zhang, S. Wang, S. Ma, and W. Gao, "Enhanced motion-compensated video coding with deep virtual reference frame generation," *IEEE Transactions on Image Processing*, vol. 28, no. 10, pp. 4832–4844, 2019.

[42] Z. Liu, R. A. Yeh, X. Tang, Y. Liu, and A. Agarwala, "Video frame synthesis using deep voxel flow," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 4473–4481, 2017.

[43] J.-K. Lee, N. Kim, S. Cho, and J.-W. Kang, "Deep video prediction network-based inter-frame coding in hevc," *IEEE Access*, vol. 8, pp. 95 906–95 917, 2020.

[44] A. Kappeler, S. Yoo, Q. Dai, and A. K. Katsaggelos, "Video super-resolution with convolutional neural networks," *IEEE Transactions on Computational Imaging*, vol. 2, no. 2, pp. 109–122, 2016.

[45] X. Tao, H. Gao, R. Liao, J. Wang, and J. Jia, "Detail-revealing deep video super-resolution," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 4472–4480.

[46] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.

[47] "Xiph.org video test media [derf's collection]," https://media.xiph.org/video/derf/.

[48] "Tensorflow," https://www.tensorflow.org/.

[49] P. J. Huber, "Robust Estimation of a Location Parameter," *The Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 73 – 101, 1964. [Online]. Available: https://doi.org/10.1214/aoms/1177703732

[50] M. Mathieu, C. Couprie, and Y. LeCun, "Deep multi-scale video prediction beyond mean square error," *CoRR*, vol. abs/1511.05440, 2016.

[51] "Matlab," https://ch.mathworks.com/products/matlab.html/.

[52] F. Bossen *et al.*, "Common test conditions and software reference configurations," *JCTVC-L1100*, vol. 12, p. 7, 2013.

[53] K. McCann, C. Rosewarne, B. Bross, V. Naccari, K. Sharman, and G. Sullivan, "High efficiency video coding test model 16 (hm 16) reference software," *Doc. JCTVC N14970*, vol. 1010.

[54] G. Wallace, "The jpeg still picture compression standard," *IEEE Transactions on Consumer Electronics*, vol. 38, no. 1, pp. xviii–xxxiv, 1992.

[55] H. Choi and I. V. Bajić, "Deep frame prediction for video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 7, pp. 1843–1855, 2019.

[56] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[57] N. Prette, D. Valsesia, and T. Bianchi, "Deep multiframe enhancement for motion prediction in video compression," in *2021 28th IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*, 2021, pp. 1–6.

[58] N. Prette, D. Valsesia, T. Bianchi, E. Magli, M. Naccari, and A. Fiandrotti, "Deep motion-compensation enhancement in video compression," *Electronics Letters*, vol. 58, 04 2022.

[59] K. Zeng, T. Zhao, A. Rehman, and Z. Wang, "Characterizing perceptual artifacts in compressed video streams," in *Electronic Imaging*, 2014.

[60] J. Tong, X. Wu, D. Ding, Z. Zhu, and Z. Liu, "Learning-based multi-frame video quality enhancement," in *2019 IEEE International Conference on Image Processing (ICIP)*, 2019, pp. 929–933.

[61] R. Yang, M. Xu, Z. Wang, and T. Li, "Multi-frame quality enhancement for compressed video," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6664–6673.

[62] T. Xue, B. Chen, J. Wu, D. Wei, and W. T. Freeman, "Video enhancement with task-oriented flow," *International Journal of Computer Vision (IJCV)*, vol. 127, no. 8, pp. 1106–1125, 2019. [Online]. Available: http://toflow.csail.mit.edu/

[63] C. Rhemann, A. Hosni, M. Bleyer, C. Rother, and M. Gelautz, "Fast cost-volume filtering for visual correspondence and beyond," in *CVPR 2011*, 2011, pp. 3017–3024.

[64] "x265 documentation," https://x265.readthedocs.io/en/master/.

[65] "Streameye," https://www.elecard.com/products/video-analysis/streameye.

[66] P. Ferriere, "Optical Flow Prediction with Tensorflow," https://github.com/philferriere/tfoptflow.

[67] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, Y. W. Teh and M. Titterington, Eds., vol. 9. Chia Laguna Resort, Sardinia, Italy: PMLR, 13–15 May 2010, pp. 249–256. [Online]. Available: http://proceedings.mlr.press/v9/glorot10a.html

[68] J. Postel, "Rfc0768: User datagram protocol," 1980.

[69] X. HoangVan and H.-H. Nguyen, "Enhancing quality for vvc compressed videos with multi-frame quality enhancement model," in *2020 International Conference on Advanced Technologies for Communications (ATC)*, 2020, pp. 172–176.

[70] M. Lu, M. Cheng, Y. Xu, S. Pu, Q. Shen, and Z. Ma, "Learned quality enhancement via multi-frame priors for hevc compliant low-delay applications," in *2019 IEEE International Conference on Image Processing (ICIP)*, 2019, pp. 934–938.

[71] N. Prette, E. Magli, and T. Bianchi, "Using ccsds image compression standard for sar raw data compression in the h2020 eo-alert project," in *European Workshop on On-Board Data Processing (OBDP2019)*, 2019.

[72] M. Kerr, S. Tonetti, S. Cornara, J. Bravo, R. Hinz, A. Latorre, F. Membibre, C. Solimini, S. Wiehle, H. Breit, B. Tings, O. Koudelka, F. Teschl, E. Magli, T. Bianchi, A. Migliorati, P. Motto Ros, M. Caon, R. Freddi, M. Benetti, F. Milani, G. Curci, S. Fraile, L. Garcia, C. Marcos, and A. Fiengo, "Eo-alert: A novel architecture for the next generation of remote sensing satellites supporting rapid civil alerts," 2020, 71st International Astronautical Congress : IAC 2020, IAC 2020 ; Conference date: 12-10-2020 Through 14-10-2020. [Online]. Available: https://iac2020.vfairs.com

[73] R. Bamler, "A comparison of range-doppler and wavenumber domain sar focusing algorithms," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 30, no. 4, pp. 706–713, 1992.

[74] A. Moreira and F. Blaser, "Fusion of block adaptive and vector quantizer for efficient sar data compression," in *Proceedings of IGARSS '93 - IEEE International Geoscience and Remote Sensing Symposium*, 1993, pp. 1583–1585 vol.4.

[75] T. Algra, "Data compression for operational sar missions using entropy-constrained block adaptive quantisation," in *IEEE International Geoscience and Remote Sensing Symposium*, vol. 2, 2002, pp. 1135–1139 vol.2.

[76] V. Pascazio and G. Schirinzi, "Wavelet transform coding for sar raw data compression," in *IEEE 1999 International Geoscience and Remote Sensing Symposium. IGARSS'99 (Cat. No.99CH36293)*, vol. 4, 1999, pp. 2251–2253 vol.4.

[77] U. Benz, K. Strodl, and A. Moreira, "A comparison of several algorithms for sar raw data compression," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 33, no. 5, pp. 1266–1276, 1995.

[78] M. Zink and R. Bamler, "X-sar radiometric calibration and data quality," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 33, no. 4, pp. 840–847, 1995.

[79] I. Blanes, E. Magli, and J. Serra-Sagrista, "A tutorial on image compression for optical space imaging systems," *IEEE Geoscience and Remote Sensing Magazine*, vol. 2, no. 3, pp. 8–26, 2014.

[80] D. R. Thompson, R. O. Green, D. Keymeulen, S. K. Lundeen, Y. Mouradi, D. C. Nunes, R. Castaño, and S. A. Chien, "Rapid spectral cloud screening onboard aircraft and spacecraft," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 52, no. 11, pp. 6779–6792, 2014.

[81] M. Cilia, N. Prette, E. Magli, B. Sang, and S. Pieraccini, "Onboard data reduction for multispectral and hyperspectral images via cloud screening," in *IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium*, 2020, pp. 6230–6233.

[82] "U.S. Geological Survey, L8 sparcs cloud validation masks," 2016.

[83] W. M. Porter and H. T. Enmark, "A System Overview Of The Airborne Visible/Infrared Imaging Spectrometer (Aviris)," in *Imaging Spectroscopy II*, G. Vane, Ed., vol. 0834, International Society for Optics and Photonics. SPIE, 1987, pp. 22 – 31. [Online]. Available: https://doi.org/10.1117/12.942280