Aalto University
School of Science

Master’s programme in ICT Innovation

Content Adaptive NN-Based In-Loop Filter for VVC

Hyperparameter Pruning

Zhijie He

Master’s Thesis
2023

Aalto University
School of Science

Author Zhijie He	
Title Content Adaptive NN-Based In-Loop Filter for VVC — Hyperparameter Pruning	
Degree programme ICT Innovation	
Major Data Science	
Supervisor Prof. Juho Kannala	
Advisor Dr. Francesco Cricri	
Collaborative partner Nokia	
Date 24 September 2023	**Number of pages** 50 **Language** English

Abstract

The most recent video coding standard VVC contains five in-loop filters to reduce compression artifacts that come from the common drawbacks of block-based hybrid compression framework. However, those traditional in-loop filters are insufficient to deal with the complicated compression artifacts. The emergence of Neural Networks (NNs) has brought significant advancements in the realm of image and video processing, offering a promising avenue for improving video compression. Many prior studies in this domain have focused on training models on large datasets to achieve generalization, rather than catering to specific content characteristics. In this work, we introduced a content-adaptive in-loop filter for Versatile Video Coding (VVC) working with other in-loop filters. The content adaptation is achieved by over-fitting a pre-trained model at the encoder side on the test data. To reduce the bitrate overhead, the Neural Network Compression and Representation (NNR) standard has been introduced which focuses on compressing NNs efficiently. Furthermore, rather than over-fitting all parameters within the NN model, we introduce a set of learnable parameters known as multipliers, which serve to further reduce the bitrate overhead. The proposed model takes auxiliary information including Boundary Strength (BS) and Quantization parameter (QP) as input. Additionally, we have conducted a comprehensive series of experiments to identify the optimal combination of hyperparameters for this approach. The results indicate coding gains of -2.07% (Y), -5.54% (Cb), -1.95% (Cr) Bjøntegaard Delta rate (BD-rate) for Class B and -1.34% (Y), -1.88% (Cb), -0.52% (Cr) Bjøntegaard Delta rate (BD-rate) for Class D with respect to the Peak Signal-to-Noise Ration (PSNR) on top of the Versatile Video Coding (VVC) Test Model (VVC) 12.0 with NN-based Video Coding (NNVC) 5.0, in Random Access (RA) configuration.

Keywords Neural network , in-loop filter, video compression , Versatile Video Coding (VVC) , NNR

Preface

As I sit down to write this preface, I am filled with a sense of accomplishment and gratitude. The journey leading to the completion of this master's thesis has been challenging, yet immensely rewarding.

Throughout this endeavor, I have had the privilege of receiving guidance, support, and inspiration from a multitude of sources. I would like to take this opportunity to express my sincere gratitude to all those who have played a pivotal role in shaping this research and my academic journey as a whole.

First and foremost, I am deeply indebted to my thesis supervisor, Juho Kannala, and thesis advisor, Francesco Cricri, whose guidance, expertise, and unwavering support have been instrumental in shaping this research. Your insightful feedback and encouragement have been invaluable, and I am grateful for the opportunity to learn from you.

I would also like to thank my colleagues, María Santamaría, Ruiying Yang, and József-Hunor Jánosi, for their valuable input and constructive feedback, which have greatly improved the quality of this work.

I extend my appreciation to the Aalto University faculty and staff for providing a conducive academic environment and access to resources essential for this research.

To my friends and family, who have provided unwavering support and understanding during this demanding journey, I am eternally grateful. Your belief in me has been my source of strength.

Lastly, I want to express my gratitude to the participants and organizations who generously contributed their time and resources to this study. Your willingness to engage in this research made a significant impact on the quality and depth of my findings.

This thesis is a testament to the collaborative spirit that permeates academia, and I am humbled by the collective effort that has brought it to fruition. It is my hope that this work will contribute to the body of knowledge in video compression and inspire future research in this area.

Thank you to everyone who has been a part of this journey, directly or indirectly. Your support and encouragement have made this thesis possible, and I dedicate this work to all of you.

Otaniemi, 9 September 2023

Zhijie He

Contents

Symbols and abbreviations

Symbols

Y	luminance (Y)
Cb, **Cr**	Red and Blue chrominance color channels
r	reference (original) images or videos
c	compressed images or videos
$\triangle$	delta symbol
$\sum$	summation symbol
$\lvert\cdot\rvert$	Absolute value

Abbreviations

AI	All-intra
ALF	Adaptive Loop Filter
BD-rate	Bjøntegaard Delta rate
CABAC	Context-Adaptive Binary Arithmetic Coding
CCALF	Cross-Component Adaptive Loop Filter
chrominance or chroma	Colour difference component
CNNs	Convolutional Neural Networks
Codec	Coder-Decoder pair
CPU	Central Processing Unit
CTC	Common Test Conditions
CTU	Coding Tree Unit
CU	Coding Units
DBF	De-Blocking Filter
DCT	Discrete Cosine Transform
GPU	Graphics Processing Unit
H.264/AVC	Advanced Video Coding
H.265/HEVC	High-Efficiency Video Coding
H.266/VVC	Versatile Video Coding
IEC	International Electrotechnical Commission
ISO	International Standards Organization
ITU	International Telecommunication Union
JPEG	Joint Photographic Experts Group
JVET	Joint Video Experts Team
LMCS	Luminance Mapping Chrominance Scaling
luma	represents brightness information in image
MPEG	Moving Picture Experts Group
NN	Neural Network
NNR	Neural Network Compression and Representation
NNVC	Neural Network-based Video Coding
PSNR	Peak Signal-to-Noise Ratio
PU	Prediction Units
QP	Quantization Parameter
RA	Random Access
ReLU	Rectified Linear Unit
SADL	Small AdHoc Deep Learning Library
SAO	Sample Adaptive Offset
TU	Transform Units
VTM	Versatile Video Coding Test Model
VCEG	Video Coding Experts Group

List of Figures

List of Tables

1 Introduction

In the ever-evolving landscape of multimedia technology, video compression stands as a vital pillar that bridges the gap between the ever-increasing demand for high-quality video content and the limited resources available for its storage, transmission, and playback.

Following the successful standardization of video coding standards such as Advanced Video Coding (H.264/AVC) [1] and High-Efficiency Video Coding (H.265/HEVC) [2] in the field of video compression technology, Versatile Video Coding (H.266/VVC) [3] emerged as the next generation of video compression technology, aiming to further advance the state of the art in video coding.

This achievement was made possible through the collaborative efforts of the Joint Video Experts Team (JVET), a collaborative group formed by the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG). In July 2020, VVC was officially ratified. The VVC standard represents a significant advancement in video compression technology compared to its predecessor. In Random Access (RA) Common Test Conditions (CTC) [4], VVC has demonstrated an impressive improvement in compression efficiency, achieving a reduction in bit rates by approximately 37% when compared to HEVC.

These standards brought about substantial improvements in video compression efficiency, paving the way for high-quality video delivery over various networks and devices.

1.1 How to resolve compression artifacts?

Versatile Video Coding (H.266/VVC) [3], which includes several fundamental modules prediction, transform, quantization, and entropy coding, follows the traditional block-based hybrid video coding structure, just like previous video coding standards [1, 2], as depicted in Figure 1, which presents a simplified diagram of a blocked-based hybrid video coding structure [5].

Within the hybrid coding framework, the encoding process begins with block partitioning, which involves dividing a video frame into discrete, non-overlapping coding blocks. These individual blocks serve as the foundational coding units (CU) and become the building blocks for subsequent processing steps, including prediction units (PU), transform units (TU), and so on. Compared to previous standards, extensive partitioning options and more adaptable partitioning combinations are introduced to VVC that provide higher coding efficiency [6].

This hierarchical structure allows for a systematic and efficient approach to video compression, where each coding block undergoes specific operations to achieve optimal compression results. A block-based coding system is distinguished by its ease of development and hardware compatibility. It simplifies code and works well with modern hardware designs, making it a viable option for video compression solutions. Furthermore, this method effortlessly supports critical coding characteristics, particularly parallelization, which is required for efficient video encoding and decoding.

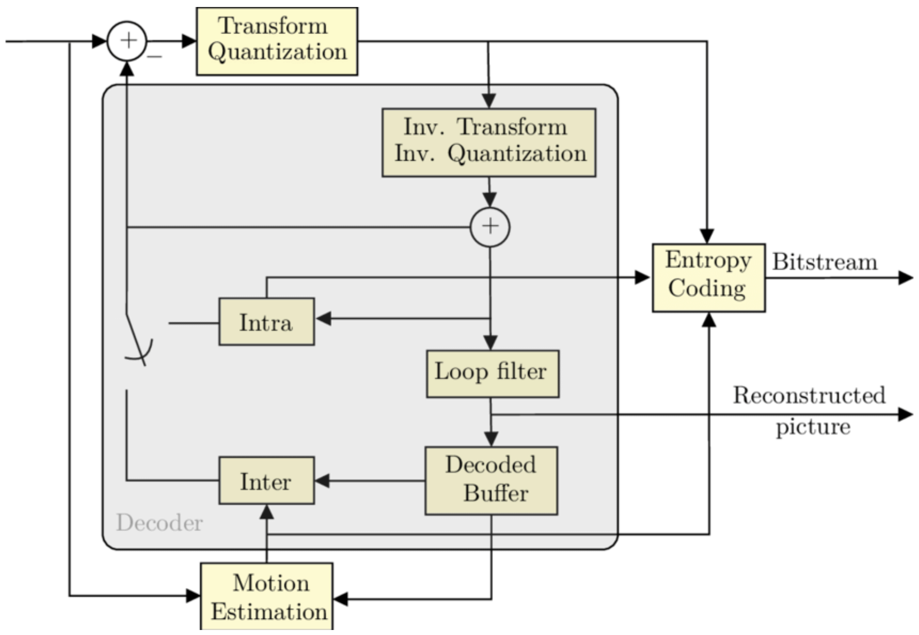

Figure 1: Diagram of a block-based hybrid video coding system

However, it's worth noting that the block-based prediction and quantization techniques employed in these existing compression frameworks [1, 2, 3] come with certain drawbacks, including the emergence of discontinuities near the boundaries of these blocks, the propagation of error information around texture contours, the loss of high-frequency details. These problems correspond to the appearance of blocking, ringing, and blurring artifacts, respectively. Addressing these artifacts is a central focus in the ongoing evolution of video compression technologies.

Over the past several decades, considerable research efforts have been devoted to addressing and mitigating compression artifacts in video coding. Hence, filters play a crucial role in mitigating compression artifacts and enhancing the reconstruction quality of the decoded video. There are two main categories of filters employed to mitigate compression artifacts: in-loop filters and post-processing filters. The key distinction between them lies in their application and impact on the video compression process. In-loop filters, as the name suggests, are applied within the encoding and decoding loops. They directly influence the compression process itself, and the filtered data serves as a reference for subsequent stages. On the other hand, post-processing filters come into play after the data has been decoded, hence the term "post-processing". Their primary purpose is to enhance the visual quality of the decoded content.

In particular, extensive research has yielded a variety of in-loop filters [7, 8, 9, 10] designed to mitigate artifacts and distortions in video coding. The VVC standard incorporates five essential in-loop filters, each with a specific purpose to enhance the quality of compressed videos. These filters are integral to mitigating various compression artifacts and distortions: The luma mapping with chroma scaling (LMCS) [11] aims at adaptively modifying the distribution of the coded sample for improved coding efficiency [12], The de-blocking Filter (DBF) [13] is primarily utilized to alleviate blocking artifacts, The sample Adaptive Offset (SAO) [14] is to minimize ringing artifacts, The adaptive Loop Filter (ALF) [15] is dedicated to refining the reconstructed signal with a particular focus on luma (brightness) samples concerning the original signal, and The Cross-Component Adaptive Loop Filter (CCALF) aims to correct adaptive clipping [12]. However, traditional filtering techniques often face challenges in effectively reducing compression artifacts in videos.

In recent times, deep learning (DL) [16], particularly Convolutional Neural Networks (CNNs), has achieved huge success in the field of image processing tasks, including image recognition and classification, which provides a promising way for image/video compression. Neural Network (NN) filters [17, 18, 19, 20, 21], leveraging the power of artificial intelligence, offer promising solutions to this issue. NN filters have demonstrated the ability to address multiple artifacts concurrently, making them a valuable tool in enhancing video quality during the compression process. However, most of the NN filters proposed in prior research are trained on large video datasets, emphasizing the importance of generalization across various content rather than specializing in content-specific adaptation.

1.2 Content Adaptation

Several previous studies have explored content adaptation by employing a selection mechanism to choose the most appropriate CNN filter from a collection of candidate models [18, 22]. In some cases, the selection of the optimal CNN filter is determined using an additional learning-based NN model. In [9], the in-loop filters utilize a discriminative Convolutional Neural Network (CNN) to choose the optimal NN filter. Likewise, [22] treated filtering as a decision process and trained an agent to select the best NN filter from a set of CNN filters with various architectures. These approaches leverage machine learning to intelligently determine the suitable filter for artifact reduction.

Meanwhile, several strategies have tackled content adaptation through NN over-fitting [23, 24, 25]. In these approaches, the weight updates of neural networks are transmitted to the decoder side along with the video bitstream. This allows the decoder to adapt its processing based on the specific content of the video, enhancing the overall compression and quality of the encoded video. In [23, 24], encoder-side over-fitting was investigated to facilitate content adaptation for a pre-trained post-processing CNN filter. This approach involves training or fine-tuning a pre-trained neural network on the encoder side to adapt it to the specific characteristics of the video content being processed. The weight updates resulting from this over-fitting process are then transmitted along with the video bitstream to the decoder side. On the decoder side, these weight updates are used to reconstruct the over-fitted CNN model before applying it to the filtering process.

However, it's important to take into account the additional bitrate overhead associated with signaling weight updates. Neural Networks (NNs) have evolved into more complex structures with a growing number of parameters, making them increasingly demanding in terms of transmission between devices. Applications like encoder-side over-fitting require efficient compression techniques to reduce bandwidth requirements.

1.3 Compress NN parameters

In response to this need, the Moving Picture Experts Group (MPEG) has developed the Neural Network Compression and Representation (NNR) standard [26]. The NNR codec design encompasses a range of compression-efficient quantization methods, which include uniform reconstruction, codebook, and dependent scalar quantization. Additionally, it leverages the Deep Context-Adaptive Binary Arithmetic Coding (DeepCABAC) [27] arithmetic coding method as a core encoding and decoding technology. Furthermore, NNR provides NN pre-processing tools like sparsification, pruning, low-rank decomposition, unification, batch norm folding, and local scaling, which further optimize the compression of NN parameters.

Moreover, the NNR standard's high-level syntax is equipped with mechanisms that facilitate parallel decoding at both the block-row and sub-tensor levels. These comprehensive features collectively contribute to the efficiency, effectiveness, and adaptability of the codec, catering to a wide range of applications and use cases.

1.4 Research Proposal

This work introduces a content-adaptive in-loop filter for video compression working with other VVC in-loop filters. Figure 3 illustrates the complete video over-fitting pipeline. The content adaptation is achieved by over-fitting a pre-trained model at the encoder side during the test phase using the input video sequence. Only a set of new learnable parameters called multipliers [25] are over-fitted and signaled. Both the encoder and decoder are expected to know the pre-trained models. To reduce the bitrate overhead, the weight update resulting from the over-fitting process is efficiently encoded using the Neural Network Compression and Representation (NNR) standard and transmitted alongside the video bitstream. On the decoder side, the over-fitted model is reconstructed before video decoding takes place. The proposed approach was evaluated on the VVC Test Model (VTM) 12.0 with NN-based Video Coding (NNVC) 5.0 in the Random Access Common Test Conditions (RA CTC) for NNVC [4].

The results indicate coding gains of -2.07% (Y), -5.54% (Cb), -1.95% (Cr) Bjøntegaard Delta rate (BD-rate) for Class B and -1.34% (Y), -1.88% (Cb), -0.52% (Cr) Bjøntegaard Delta rate (BD-rate) for Class D with respect to the Peak Signal-to-Noise Ration (PSNR).

1.5 Structure of the thesis

The remainder of this work is structured as follows:

2. **Background** briefly summarizes the existing research and literature and discusses the traditional in-loop filter limitations.

3. **Research Methodology** details the methodology for the content adaptation concept, including video over-fitting pipeline, Neural network architecture, and hyperparameter pruning.
4. **Simulation Results and Discussions** we present and explain our results.
5. **Conclusion** concludes the main findings and suggests paths that are worthwhile for future research activities.

2 Background

In this section, we will begin by reviewing the evolution of traditional video compression standards. Subsequently, we will introduce the traditional in-loop filters incorporated into the latest video compression standard, VVC (Versatile Video Coding). Finally, we will summarize some research papers that delve into the application of neural networks in the context of in-loop filters within the field of video compression.

2.1 Video Compression

2.1.1 The general path of video coding development

Over the past few decades, advancements in internet technology and device proliferation have greatly facilitated the global transmission of information. Beyond the internet speed and bandwidth capacity, the size of the data is equally pivotal, underscoring the significance of data compression.

The history of video compression is a journey marked by relentless innovation and continuous adaptation to the evolving needs of multimedia technology. Historically, video compression began with analog methods aimed at reducing the bandwidth required for transmitting video signals. Techniques such as time-division multiplexing and frequency-division multiplexing allowed multiple video signals to share a common channel. However, the digital revolution heralded a new era, bringing with it techniques that would shape the future of video compression.

Among the pivotal milestones in this journey was the introduction of the Discrete Cosine Transform (DCT) [28] in the 1970s. The DCT marked a watershed moment, enabling spatial compression by converting image blocks into frequency domain coefficients. This technique found its first major application in the JPEG standard [29] for image compression and later became the foundation for video compression standards like MPEG-1.

In the early 1990s, MPEG-2 extended the capabilities of video compression, enabling the storage and transmission of digital video content on DVDs, digital television, and satellite broadcasts. The compression efficiency of MPEG-2 was a significant leap forward, allowing for the delivery of high-quality video within limited bandwidth constraints.

The next significant milestone in the evolution of video compression after MPEG-2 was the development of H.264, also known as Advanced Video Coding (AVC) [1]. H.264 represented a substantial improvement in compression efficiency over its predecessors. It introduced advanced techniques such as inter-frame prediction, variable block sizes, and entropy coding, which significantly reduced the bitrates required for high-quality video.

H.264 quickly became the dominant video compression standard and played a pivotal role in enabling high-definition video streaming and video conferencing over the Internet. Its efficiency made it possible to deliver high-quality video content even in environments with limited bandwidth.

Following H.264, the next major advancement was the introduction of High-Efficiency Video Coding (HEVC or H.265) [2] in 2013. HEVC further improved compression efficiency, making it possible to transmit 4K and 8K video content with reduced bitrates. It became the foundation for modern video streaming services and high-resolution video formats.

Indeed, following High-Efficiency Video Coding (HEVC or H.265), the next significant advancement in video compression technology is the introduction of Versatile Video Coding (VVC) [3], also known as H.266. VVC is the latest generation video compression standard that aims to push the boundaries of compression efficiency even further. Versatile Video Coding (VVC) was introduced to address the growing demand for high-resolution video content, including 4K, 8K, and beyond while optimizing data transmission and storage. VVC employs a range of innovative techniques, including more advanced intra-frame and inter-frame coding, improved motion compensation, and enhanced entropy coding. These advancements collectively result in higher compression ratios and improved video quality.

Looking forward, the video compression field continues to evolve. Emerging technologies such as the Alliance for Open Media's AV1 codec, which offers royalty-free and open-source compression, are poised to make significant contributions to the industry. Additionally, research into machine learning-based compression algorithms and real-time adaptability is ongoing, promising even more efficient and flexible video compression solutions for the future.

2.1.2 Discrete Cosine Transform

The Discrete Cosine Transform (DCT) stands as a pivotal mathematical technique in the domain of multimedia compression, serving as a cornerstone in numerous image and video compression standards. Its significance lies in its capacity to transform spatial pixel data into the frequency domain, thereby facilitating efficient data compression while preserving essential visual information. Through its block-based processing approach, where data is partitioned into manageable blocks, such as 8x8 pixel segments, the DCT enables the removal of less perceptually significant data, leading to the reduction of file sizes. This technique, coupled with quantization and optional entropy coding, renders DCT-based compression typically lossy, balancing compression ratios with acceptable image quality. As a fundamental component of standards like JPEG for images and MPEG for video, the DCT plays a vital role in multimedia storage, transmission, and display, underscoring its enduring relevance in the digital age.

2.1.3 Versatile Video Coding(VVC)

Versatile Video Coding (VVC), also known as H.266, is considered the most advanced and contemporary video compression standard. The development of Versatile Video Coding (VVC) was primarily motivated by the ever-increasing demand for higher-resolution video content. This demand has been driven by the widespread adoption of ultra-high-definition displays and the emergence of 4K and 8K video streaming technologies. VVC was designed to address the need for more efficient compression methods to deliver such high-resolution content over various networks and platforms. In response to this evolving landscape, VVC has emerged as a transformative force, pushing the boundaries of what's achievable in video compression. It goes beyond its predecessors by optimizing compression efficiency to an unprecedented level while simultaneously elevating video quality. This remarkable achievement is the result of extensive research and innovation, encompassing novel encoding techniques, improved algorithms, and adaptive tools designed to address the diverse challenges posed by modern multimedia applications. As the successor to High-Efficiency Video Coding (HEVC), VVC not only inherits the legacy of its predecessors but also sets a new standard for versatility and adaptability, making it an indispensable cornerstone in the ever-evolving landscape of digital video technology.

2.2 Traditional In-Loop Filters for VVC

To mitigate the compression artifacts induced by the block-based compression process, numerous in-loop filtering techniques have been proposed. VVC defines a set of in-loop filters, each designed to address specific compression artifacts, ultimately enhancing the quality of the encoded video [12].

In Figure 2 [30], we can observe the simplified VVC (Versatile Video Coding) decoder block diagram, which places particular emphasis on the in-loop filter coding blocks, highlighted by the red dashed line. This diagram offers a visual representation of the complex procedures within VVC decoding and underscores the essential function performed by the in-loop filters in improving both video quality and compression efficiency. These filters play a critical role in refining the reconstructed video signal. Before storing the reconstructed samples in the decoded picture buffer, five sequential processing steps are employed. This process begins with the Luma Mapping with Chroma Scaling (LMCS) Process, followed by the utilization of the deblocking filter (DBF). Next in line is the application of the Sample Adaptive Offset (SAO) filter, and lastly, the deployment of both the adaptive loop filter (ALF) and the Cross-Component Adaptive Loop Filter (CC-ALF).

Here's an overview of these in-loop filters:

- Luma Mapping with Chroma Scaling (LMCS) [11]: The LMCS step involves adjusting the luma (brightness) values of the video samples and simultaneously scaling the chroma (color) information. LMCS is a technique used to optimize the visual quality and compression efficiency of the video by adjusting the luminance and chrominance components separately.

- De-Blocking Filter (DBF) [13]: The DBF is a critical component for mitigating discontinuities that frequently manifest at block boundaries. These discontinuities arise due to the block-based prediction and transformation processes used in video compression. DBF plays a pivotal role in ensuring that adjacent blocks blend seamlessly, thus reducing the visual artifacts commonly associated with block-based compression.

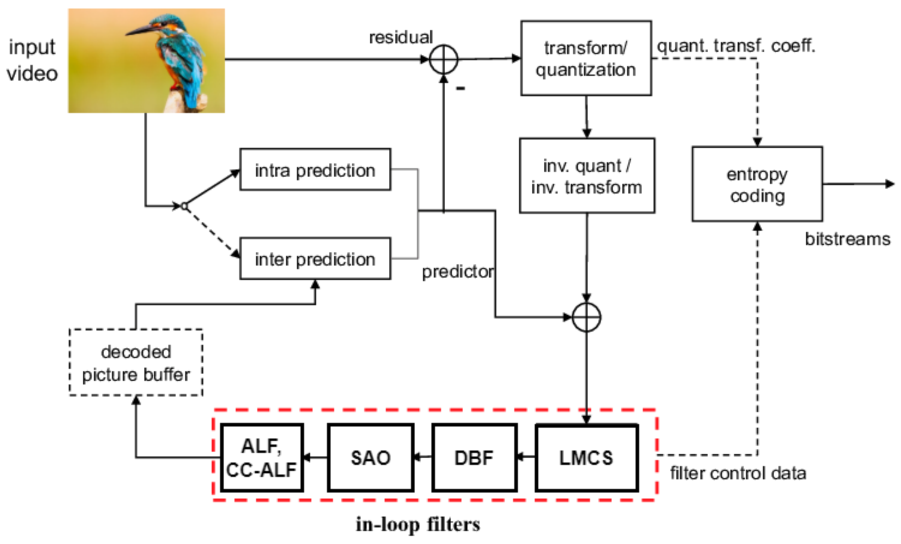

Figure 2: In-loop filters in VVC

- Sample Adaptive Offset (SAO) [14]: SAO is a crucial tool in the fight against ringing artifacts, a common consequence of the quantization step in compression. By applying SAO, these unwanted visual distortions within individual blocks are effectively diminished, resulting in a smoother and more pleasing visual experience.

- Adaptive Loop Filter (ALF) [15]: ALF takes video quality enhancement to the next level by precisely refining the luma (brightness) samples in the reconstructed signal concerning the original input. This meticulous adjustment minimizes discrepancies and further enhances the perceived image quality, ensuring that viewers receive a more faithful representation of the source content.

- Cross-Component Adaptive Loop Filter (CC-ALF) [31]: In addition to luma refinement, CC-ALF extends its capabilities to chroma (color) samples. By meticulously adjusting both luma and chroma components, CC-ALF contributes to the harmonization of color information, delivering a reconstructed signal that remains faithful to the original source across all aspects of the video.

In addition to the in-loop filters that are integrated into video coding standards, there have been studies [32, 33, 34] exploring various traditional filters aimed at reducing compression artifacts. For instance, [32] leveraged non-local image information by imposing a low-rank constraint on similar image patches, with the goal of reducing compression noise. On the other hand, [33] introduced an adaptive wavelet domain filter designed to suppress quantization noises in coded blocks within the frequency domain. Furthermore, [34] utilized a bilateral filter to further diminish coding artifacts, with particular emphasis on addressing ringing artifacts. However, it's worth noting that traditional filters typically make ideal assumptions about the nature of compression artifacts and may not be capable of effectively addressing all situations. Due to the complexity and diversity of compression artifacts stemming from video coding, the performance improvements achieved by traditional filters remain relatively limited.

2.3 Neural Network Based In-Loop Filter for Video Coding

The successful utilization of Convolutional Neural Networks (CNNs) in various low-level computer vision tasks, such as image restoration [35], recognition, and classification, has propelled extensive research into CNN-based in-loop filters. These investigations aim to significantly enhance the efficiency and effectiveness of in-loop filters within the realm of video coding. By harnessing the potential of CNNs, advancements in in-loop filtering have garnered substantial attention and effort, contributing to a more refined and optimized approach to video coding. The success of NNs has spread to the field of video coding, where they have demonstrated the ability to remove compression artifacts either as in-loop filters [10, 18, 19, 25, 20, 21, 17] or post-filters [36, 37, 38]. In [20], a highly sophisticated recursive residual convolutional neural network (RRCNN) is introduced as a solution for the reconstruction of intra-frames. Additionally, in another study highlighted in [17], an innovative approach employing a deep residual highway convolutional neural network (RHCNN) is presented. This approach seamlessly integrates RHCNN into the in-loop filtering process within the High-Efficiency Video Coding (HEVC) standard, effectively addressing both intra and inter-mode filtering. However, in most frameworks, a single offline trained NN does the filtering, in other frameworks, a set of candidate NNs are evaluated and the one that offers the best results is selected.

NN over-fitting has emerged as a method to achieve content adaptation, leading to enhanced coding efficiency. Figure 3 provides an overview of the complete over-fitting pipeline.

On the encoder side, a pre-trained NN model is over-fitted on the input video data. The pre-trained NN model is selected out of four candidate models based on specific metrics, such as the model that achieves the highest PSNR gain. Rather than transmitting the entire over-fitted NN model, a weight update is created by calculating the difference between the over-fitted model parameters and the pre-trained model parameters. This weight update is then compressed and encoded using the NNR standard before being transmitted along with the video bitstream. Since over-fitting only occurs on the encoder side, it is assumed that both the encoder and decoder know the pre-trained models.

On the decoder side of the video compression system, the weight update that was initially signaled from the encoder is received and undergoes a crucial sequence of steps. The signaled weight update is first decompressed using the NNR (Neural Network Representation) standard. Following the successful decompression, the next step involves the careful integration of the received weight update with the corresponding pre-trained NN model to reconstruct the over-fitted model before the filtering process.

Earlier methods in this field primarily focused on addressing the bitrate overhead incurred during the transmission of weight updates, with a particular emphasis on mitigating over-fitting biases. The aim was to strike a balance between maintaining video quality and reducing the data required for transmitting weight updates. However, the landscape of video compression and neural network-based encoding has witnessed a significant advancement with the introduction of innovative techniques, as exemplified in [39]. In this groundbreaking work, a novel set of multipliers takes center stage as a game-changing approach. These multipliers are strategically employed to further minimize the already reduced bitrate overhead while simultaneously enhancing the overall performance of the system.

3 Research Methodology

This section will discuss the proposed content adaptation scheme in detail, including the base model network architecture, dataset, over-fitting process, and inference.

3.1 Over-fitting Video pipeline

The over-fitting video pipeline is visually represented in Figure 3, and it can be divided into two distinct parts: the encoder and the decoder side. The process of content adaptation is accomplished by over-fitting a pre-trained neural network model to the input video data. This adaptation step plays a pivotal role in optimizing the video compression process to suit the specific characteristics of the content being encoded or decoded. The pre-trained neural network (NN) model utilized as the default low-complexity neural network loop filter in NNVC 5.0 was detailed in JVET-AD156 [40]. After the over-fitting process, the calculated weight updates (overfitted model parameters - pre-trained model parameters) are coded with the MPEG-NNR standard, and the bitrate overhead is considered in the BD-rate calculation. Then the weight update will be signaled with the compressed video bitstream to the decoder. Since only weight updates are signaled, not the entire NN models, the pre-trained NN models are expected to be known by both the encoder and decoder sides.

The inference process was executed using the Small AdHoc Deep Learning library (SADL) [41] and quantization was applied to achieve int16 precision. The evaluation was performed on the JVET-NNVC common test conditions (CTC) [4] in Random Access (RA) configurations. Notably, this evaluation enables neural network intra-prediction and the (overfitted) low-complexity neural network loop filter.

On the encoder side, two essential signals are signaled to the decoder side: the NNR bitstream (compressed weight updates) and the video bitstream. The decoder side utilizes the signaled NNR bitstream to reconstruct the overfitted neural network (NN) model. This reconstructed NN model is then employed to reconstruct the video content, contributing to the enhancement of video quality during the decoding process. A scaling factor is also signaled for each color component in picture headers which is used to scale the residues before being added to the input data. The residues are the difference between the input data and the reconstructed NN model output data.

3.2 Network Architecture

The pre-trained NN (NN-based loop filter) model network architecture is based on JVET-AD0156 [40] (Figure 4) which is the default (low-complexity) neural network loop-filter in NNVC 5.0. The key difference is the integration of multiplier layers after each convolutional block, serving to implement content adaptation and simultaneously reduce the size of the signaled NNR bitstream.

This model comprises a total of N filter blocks, with $N - 2$ of them being hidden layers (Figure 4). Within each hidden filter block, a configuration of two 1x1 convolution layers is connected by a leaky Rectified Linear Unit (ReLU) activation

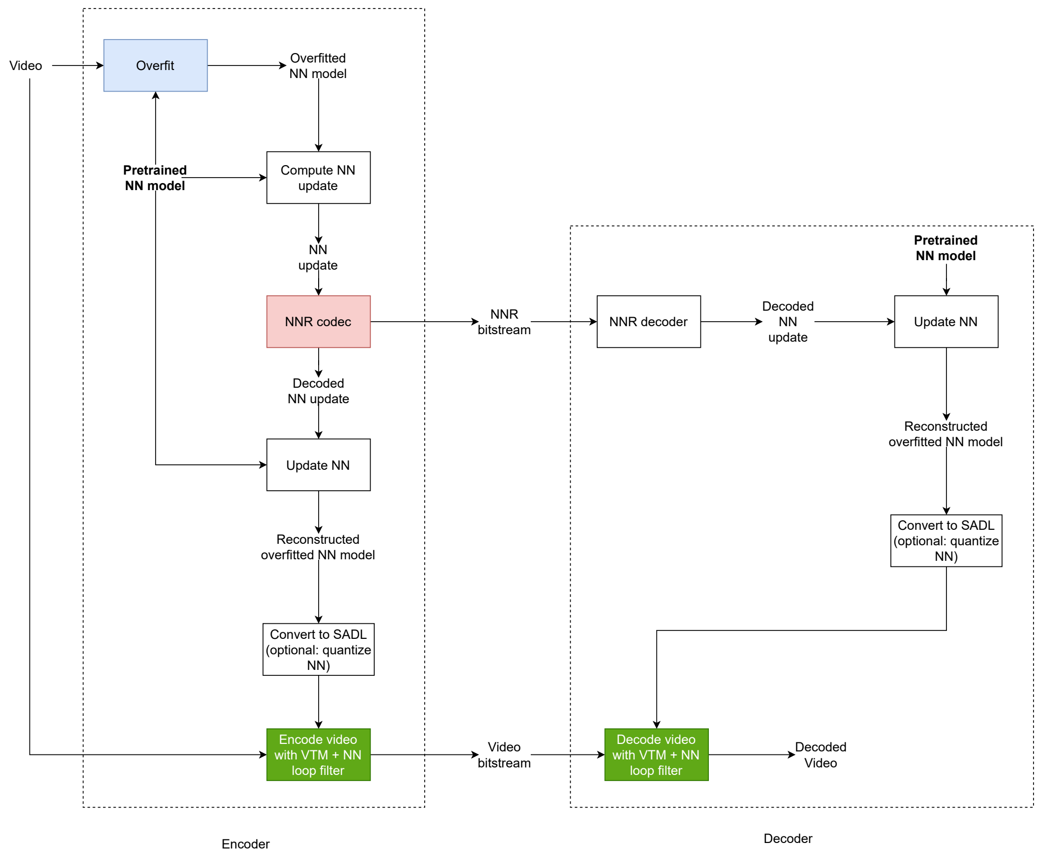

Figure 3: Over-fitting video pipeline

layer [42], followed by a subsequent 4 layers with rank R followed by fusion of adjacent 1x1 convolution:

1. $1x1xKxR$ pointwise convolution
2. $3x1xRxR$ separable convolution
3. $1x3xRxR$ separable convolution
4. $1x1xRxK$ pointwise convolution

The base model takes as input a block of reconstructed samples measuring 128x128 CTU, along with 8 neighboring samples on each side of the CTU. Furthermore, frame-level parameters, such as the Quantisation Parameter (QP) step and Deblocking boundary strength (BS), are incorporated as additional input planes into the neural network. The luma samples are interleaved into four 72x72-sized blocks, matching the size of the chroma blocks. The model generates an output consisting of six filtered blocks, sized at 64x64, comprising four luma blocks and two chroma blocks.

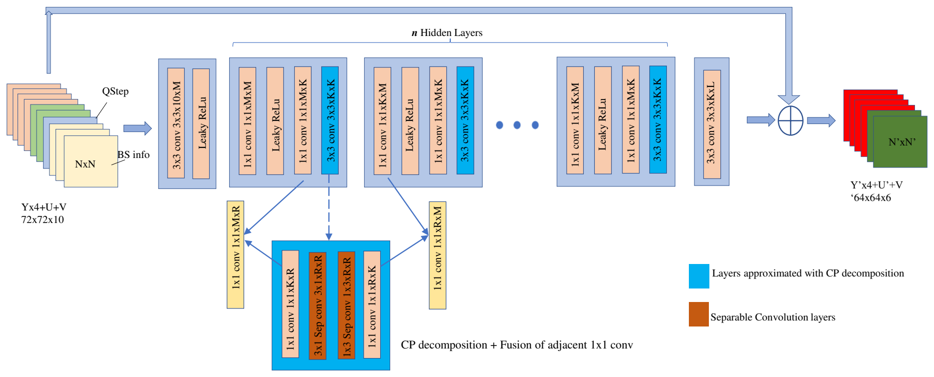

Figure 4: NN architecture from JVET-AD0156

3.3 Training Dataset

The pre-trained NN models are trained using two datasets: DIV2K [43] for All-Intra (AI) and the BVI-DVC dataset [44] for Random Access (RA). AI involves compressing each video frame independently, while RA enables quick and direct access to specific frames within a video sequence. BVI-DVC provides video sequences containing 64 frames per frame, encompassing a range of resolutions from 2160p down to 240p. DIV2K comprises 800 high-resolution images and six sets of low-resolution images produced at various scales, each generated using distinct methods. The original image and video data underwent compression using the VTM 12.0 NNVC 5.0. Consequently, four pre-models were generated, comprising two models trained on AI-compressed data and an additional two models trained on RA-compressed data. The pre-trained models from this offline training stage are referred to as base models.

3.4 Proposed filter position in VVC

A Neural Network (NN) based loop filter is a neural network model that becomes part of the loop filter chain within a video codec. It offers the flexibility to either replace one of the existing loop filters or be seamlessly integrated into the existing loop filter structure at any chosen position. Typically, the most commonly adopted positions for its integration are just prior to (or concurrently with) the deblocking filter (DBF) and similarly before (or in parallel with) the Sample Adaptive Offset (SAO) filter.

For a visual representation of this concept, Figure 5 offers a simplified overview of the in-loop filter process and clearly illustrates the placement of the proposed NN-based filter, which is positioned in parallel with the DBF filter. When an NN-based loop filter operates in parallel with a non-NN loop filter, its outputs are combined using a weighted sum, such as 0.5 times the NN output plus 0.5 times the DBF output, as an example of blending.

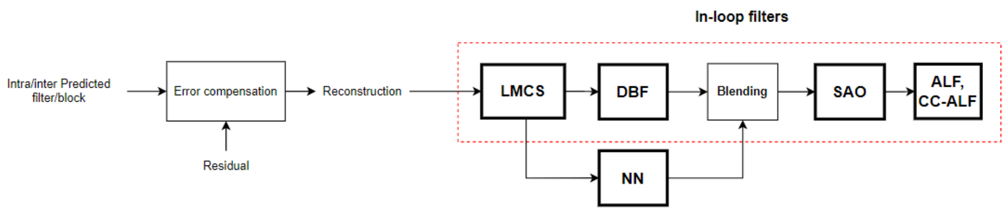

Figure 5: Proposed filter position in VVC

3.5 Over-fitting

The test video sequences as shown in Table 1 are the mandatory sequences from JVET common test conditions for NNVC [4]. The test video sequences are encoded with the VTM 12.0 NNVC5.0 and the JVET CTC RA for NNVC. In this way, the reconstruction video sequences are used as input to the base model to be overfitted. During over-fitting, each reconstructed video sequence will utilize five distinct quantization parameter (QP) values, specifically: 22, 27, 32, 37, and 42.

Four offline base models were generated using the mentioned training dataset. The weight updates resulting from the over-fitting need to be signaled to the decoder, and thus represent a bitrate overhead, only one base model was over-fitted per test (sequence and QP) to reduce such overhead [45]. The optimal base model for over-fitting selection is done by selecting the one providing the highest Peak Signal-to-Noise Ratio (PSNR) gain on the first RA segment of the input video sequence. The selected optimal base model will be over-fitted at the encoder side using the test video sequence as input data. Since the optimal base model has a relatively large number of parameters, only a subset of them are over-fitted to further lower the weight-update bitstream overhead. The over-fitted parameters are a new set of learnable parameters, named multipliers [39]:

$$\delta\left((W * x + b) * m\right)$$

where W is the kernel, $*$ is the convolution operator, x is the input, b is the bias, m is the multiplier and δ is the activation function.

Following the over-fitting process, NNR is employed to encode and compress the weight-update information.

Table 2 shows the Network Information during the over-fitting stage.

3.5.1 Model Evaluation Metrics

We use multiple objective image quality metrics to measure the compressed videos. Given a reference image r and the corresponding compressed image c, both of size MxN.

- MSE (Mean Squared Error): The Mean Squared Error between the reference and compressed images. The MSE between r and c is defined by:

$$MSE(r,c) = \frac{1}{MN}\sum_{1}^{M}\sum_{1}^{N}\left(r_{ij} - g_{ij}\right)^2$$

Table 1: JVET Mandatory Test video sequences

Class	Sequence name	Frame count	Frame rate	Bit depth
A1	Tango2	294	60	10
A1	FoodMarket4	300	60	10
A1	Campfire	300	30	10
A2	CatRobot1	300	60	10
A2	DaylightRoad2	300	60	10
A2	ParkRunning3	300	50	10
B	MarketPlace	600	60	10
B	RitualDance	600	60	10
B	Cactus	500	50	8
B	BasketballDrive	500	50	8
B	BQTerrace	600	60	8
C	RaceHorses	300	30	8
C	BQMall	600	60	8
C	PartyScene	500	50	8
C	BasketballDrill	500	50	8
D	RaceHorses	300	30	8
D	BQSquare	600	60	8
D	BlowingBubbles	500	50	8
D	BasketballPass	500	50	8
E	FourPeople	600	60	8
E	Johnny	600	60	8
E	KristenAndSara	600	60	8
F	BasketballDrillText	500	50	8
F	ChinaSpeed	500	30	8
F	SlideEditing	300	30	8
F	SlideShow	500	20	8

- PSNR (Peak Signal-to-Noise Ratio) [46]: Measures the ratio of the peak signal power to the noise power and is typically expressed in decibels (dB). The formula for PSNR is as follows:

$$PSNR = 10 \times \log_{10}\left(\frac{MAX^2}{MSE}\right)$$

  where MAX is the maximum possible pixel value of the image or video (e.g., 255 for 8-bit images).

- Delta PSNR (dPSNR): A measure of the change in PSNR before and after applying the over-fitting process to an image or video. It is calculated as the difference in PSNR values. The formula for delta PSNR is:

$$dPSNR = PSNR_{after} - PSNR_{before}$$

Table 2: Network Information for NN-based Video Coding Tool Testing in Over-fitting Stage

Network Information in Over-fitting Stage		
Mandatory	GPU Type	GPU NVIDIA A100-SXM-80GB
	Framework	TensorFlow 2.8.0
	Number of GPUs per Task	1
	Epoch	100 and 200 epochs
	Batch size	64
	Loss function	Weighted-MAE and Weighted MSE
	Training time (for 1 model)	
	Training data information	JVET CTC RA mandatory sequences
	Training configurations for generating compressed training data (if different to VTM CTC)	
Optional	Patch size	72x72
	Learning rate	
	Learning rate update strategy	Constant Learning Rate, ExponentialDecay and ReduceLROnPlateau
	Optimizer	ADAM
	Preprocessing	Convert 144x144 YUV420 signal to 6 72x72 blocks. (Normalize to 0 ~1)

- Positive delta PSNR (Positive dPSNR): A variation of delta PSNR that focuses on the positive change in PSNR, indicating improvements in image or video quality. It is calculated as the difference in PSNR values when the PSNR improvement is greater than zero (positive values indicating better quality):

$$Positive\, dPSNR = max(0, dPSNR)$$

3.5.2 Hyperparameter Selection

To find the optimal hyperparameter combination, we conducted experiments encompassing various aspects, including the selection of:

- Over-fitting Color Weight: We employed two color weight configurations: 4:1:1 and 12:1:1, allocating weights to the luminance (Y), chroma blue (Cb), and chroma red (Cr) components, respectively.

- Loss Function: Weighted Mean Absolute Error (MAE) and Weighted Mean Squared Error (MSE). The weight comes from the pre-defined over-fitting color weight, the formula for Weighted MAE loss is as follows:

  $$Weighted\, MAE = \frac{Y}{YCbCr} MAE_Y + \frac{Cb}{YCbCr} MAE_{Cb} + \frac{Cr}{YCbCr} MAE_{Cr}$$

  where MAE is the Mean Absolute Error. The MAE between reference image r and compressed image c is defined by:

  $$MAE(r, c) = \frac{1}{MN} \sum_{1}^{M} \sum_{1}^{N} \left| r_{ij} - g_{ij} \right|$$

  We calculate separate MAE values for each color channel, denoted as MAE_Y, MAE_{Cb}, MAE_{Cr} corresponding to the luminance (Y), chroma blue (Cb), and chroma red (Cr) channels, respectively.

  Similarly, the formula for weighted MSE is:

  $$Weighted\, MSE = \frac{Y}{YCbCr} MSE_Y + \frac{Cb}{YCbCr} MSE_{Cb} + \frac{Cr}{YCbCr} MSE_{Cr}$$

- Learning Rate: A set of learning rates that are determined empirically.

- Learning Schedule: Constant Learning Rate, ExponentialDecay and ReduceLROnPlateau.

- Model Save Metric: This parameter defines the metric utilized for selecting the optimal model checkpoint during the over-fitting process. The available options include Loss, PSNR, delta PSNR, and Positive delta PSNR concerning color channels (Y, Cb, Cr).

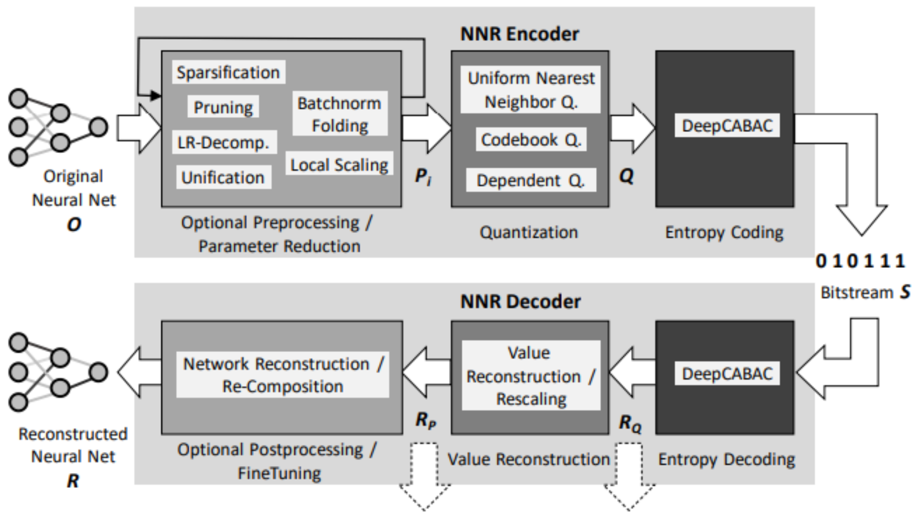

Figure 6: NNR pipeline

3.6 Weight-update compression NNR

As shown in Figure 3, base models (pre-trained NN models) are expected to be known by both the encoder and decoder sides. Considering that over-fitting only occurs at the encoder side, the NN weight update needs to be sent to the decoder side to reconstruct the over-fitted base model. In order to compress NN weight update efficiently, the Neural Network Compression and Representation standard (NNR) is introduced which is the first international standard for efficient compression of neural networks (NNs)[47, 26].

Figure 6 describes an overview of the NNR coding and decoding process which is structured into three key stages [26]:

1. Parameter Reduction Methods: Obtain a more compact representation of the model.

2. Parameter Quantization Methods: Discretize or represent parameter values with lower precision, optimizing the efficiency of encoding.

3. Entropy Coding Methods: Encode the quantized parameters effectively which ensures that the compressed data can be efficiently transmitted or stored while minimizing information loss.

In this work, the weight updates were coded with the Neural Compression Test Model (NCTM) [26]. The coding process involved uniform quantization, a QP density of 2, and the CABAC unary length set to 10. The selection of the best QPs was carried out through the application of the Inference-Optimized Quantization (IOQ) technique [48] based on the performance of the NN at the inference stage. The iterative IOQ process initiates with a small QP value. In each iteration, the NN weights are quantized and encoded, resulting in a quantized NN. Subsequently, this quantized NN is evaluated based on a specific performance criterion. The algorithm ends when the performance gets lower than a given threshold, and the QP used in the previous iteration is chosen as the optimal one.

In the context of this work, the performance criterion utilized is the △PSNR, which represents the Peak Signal-to-Noise Ratio (PSNR) difference between the reconstructed output after the application of the over-fitted filter and the original input prior to the filter's application.

3.6.1 Hyperparameter Selection

- NNR Color Weight: We employed two color weight configurations: 4:1:1 and 12:1:1, allocating weights to the luminance (Y), chroma blue (Cb), and chroma red (Cr) components, respectively.

- Max Degradation: The Maximum allowed degradation represents the difference between the reference PSNR (typically the PSNR of the output of the non-quantized NN model) and the quantized PSNR. This Max Degradation value can be adaptative based on the bitrate of the video, which includes factors like the quantization parameter (QP) and the resolution of the video.

3.7 VTM Inference

During the inference phase, SADL (Selective AdaDeep Learning) employs int16 precision, which means that the model's parameters and intermediate activations are quantized to int16 precision. The approach of quantization was presented in JVET-AD0156 [40] by means of normalization of weights and bias by scaling factors to reduce the weights with large values. The scaling factor is signaled in the picture headers for each color component and it is applied to scale the difference between the input samples and the NN filtered samples, known as residues. These scaled residues are then added to the input samples as part of the processing procedure.

After this process, the weights are quantized using a naïve approach, and the quantizers of bias, multipliers, input, and output are 11, 13, 11, and 13 respectively.

More information during the inference stage is shown in Table 3.

Table 3: Network Information for NN-based Video Coding Tool Testing in Inference Stage

Network Information in Inference Stage		
Mandatory	GPU Type	CPU only
	Framework	SADL
	Number of GPUs per Task	
	Number of Parameters (Each Model)	53724
	Total Number of Parameters (All Models)	107448
	Parameter Precision (Bits)	Int16
	Memory Parameter (MB)	0.21
	Multiply Accumulate (kMAC/pixel)	16.5
	Calculation Method	on a block basis
Optional	Total Conv. Layers	46
	Total FC Layers	0
	Total Memory (MB)	
	Patch size	72x72
	Changes to network configuration or weights required to generate rate points	
	Peak Memory Usage (Total)	

4 Simulation Results and Discussions

This section presents the results of performing the over-fitting video pipeline and explores the results of minor modifications to the model and NNR configuration based on those initial values. Section 4.1 presents the outcomes of the over-fitting video pipeline with the default configuration. Section 4.2 examines the influence of various loss functions. Section 4.3 investigates the impact of fixed learning rates versus adaptive learning rates with different schedules on the final BD rate. Section 4.4 delves into the consequences of various model save metrics. Section 4.5 studies the Max Degradation impact on NNR results and the final BD rate. Section 4.6 illustrates the process of selecting the optimal base model based on various insights. Finally, section 4.7 summarizes the best hyperparameter combination.

The over-fitting phase was conducted on GPU NVIDIA A100-SXM-80GB with TensorFlow version 2.8.0. Subsequently, the inferences are performed on the JVET-NNVC common test conditions in RA configurations [4]. The video encoding and decoding processes were executed on CPU cores, specifically the Intel Xeon Gold 6154 clocked at 3.00 GHz.

The weight update compression was done with the Neural Compression Test Model (NCTM) [26]. For IOQ the initial QP was -40. The coding used the default configuration, uniform quantization, and QP density 2.

The anchor is NNVC 5.0, where both the default neural network low-complexity filter and the neural network intra-filter are enabled. In the test, the neural network intra-filter and the (overfitted) low-complexity loop filter are enabled. The base models were obtained based on data coded by the NNVC 5.0 anchor.

The video quality is measured with the PSNR and the BR-rate [49] quantifies the coding performance (lower means better). While the NNR bitstream is signaled independently from the video bitstream, its bitrate was taken into account in the BD-rate computation.

4.1 Default Configuration

Table 4 displays the default configuration of our over-fitting proposals while Table 5 presents the coding performance is improved by the over-fitted models [50]. For the simulation results shown in Table 5, one of the base models is replaced with an over-fitted one. The overall results are averages computed across classes A1, A2, B, and C. The over-fitting process, conducted on a GPU, took approximately 11 hours for classes A1 and A2, 7 hours for class B, 2.5 hours for class C, 1 hour for class D, and 4 hours for class F.

Due to time constraints, our forthcoming tests will primarily focus on the evaluation of classes C and D.

4.2 Color Weight Test

In this section, we examine how different color weight configurations affect the final BD-rate. Two color weight configurations were employed: 4:1:1 and 12:1:1,

Table 4: Default configuration

Hyperparameter Selection	
Over-fitting Color Weight	4:1:1
NCTM Color Weight	4:1:1
Loss Function	Weighted-MSE
Epoch	200 epochs for class C, D 100 epochs for class A, B,F
Learning Rate	1e-3
Learning Schedule	Constant Learning Rate
Model Save Metric	Loss YCbCr
NCTM Max Degradation	0.05
Optimal Base Model	Highest PSNR

Table 5: BD-rate of the default configuration. Anchor: NNVC 5.0

Class	Y -PSNR	U-PSNR	V-PSNR	EncT	DecT CPU
A1	-1.59%	-3.96%	-7.97%	97%	107%
A2	-0.22%	-7.77%	-6.84%	97%	106%
B	-1.70%	-8.42%	-5.13%	98%	107%
C	-1.06%	-4.80%	-3.17%	99%	106%
Overall	-1.21%	-6.43%	-5.52%	98%	107%
D	-0.80%	-7.86%	-7.23%	98%	107%
F	-1.07%	-7.51%	-5.16%	98%	107%

Table 6: Over-fitting Color Weight Test. Anchor: NNVC 5.0

Over-fitting Color Weight Test	BD-rate Over NNVC-5.0				
	Y-PSNR	U-PSNR	V-PSNR	EncT	DecT CPU
4:1:1	-0.80%	-7.86%	-7.23%	98%	107%
12:1:1	-1.03%	-5.42%	-4.36%	98%	106%

with weights allocated to the luminance (Y), chroma blue (Cb), and chroma red (Cr) components, respectively. Indeed, Y (luminance) is typically considered more important than Cb and Cr (chrominance) in video coding and processing. This is because the human visual system is more sensitive to changes in brightness (luminance) than changes in color (chrominance). As a result, video compression algorithms often allocate more bits and resources to preserving the quality of the luminance channel to ensure that the overall video quality is visually acceptable.

4.2.1 Over-fitting Color Weight Test

Table 6 presents the results of testing various over-fitting color weight configurations and their effects on the BD-rate.

Boosting the weight on the Luma (Y) component while reducing the weight on the Cb and Cr components results in a notable improvement in the BD rate of Y-PSNR. This adjustment places more emphasis on enhancing the luminance component, which leads to better Y-PSNR performance. However, it comes at the cost of reduced performance in the Cb and Cr components. This trade-off allows for more tailored optimization of video quality based on priorities in different color channels.

4.2.2 NNR Color Weight Test

This experiment aims to find the most suitable NNR color weight configuration for optimizing video quality during the coding process. Different weight combinations for color channels (e.g., Y, Cb, Cr) are tested to understand how they affect the overall video quality. Table 7 displays the results of testing different nnr color weight configurations and their impact on the BD-rate.

Using a 12:1:1 color weight setting has shown significant improvements in Y-PSNR (luma channel) while causing fewer changes in the chroma channels, namely U-PSNR and V-PSNR. This indicates that the 12:1:1 color weight configuration prioritizes enhancing the performance of the luma channel while accepting some trade-offs in the chroma channels. In contrast, the over-fitting color weight setting sacrifices performance in the chroma channels to achieve more gains in the luma channel, making the 12:1:1 setting a more acceptable compromise.

4.3 Learning Rate and Learning Schedule Test

Table 8 provides the results of experiments testing different learning rates and learning schedules. Three types of learning schedules were employed:

Table 7: NNR Color Weight Test. Anchor: NNVC 5.0

NNR Color Weight Test	BD-rate Over NNVC-5.0				
	Y-PSNR	U-PSNR	V-PSNR	EncT	DecT CPU
4:1:1	-1.03%	-5.42%	-4.39%	99%	107%
12:1:1	-1.08%	-5.41%	-4.36%	98%	106%

- **Constant Learning Rate** The learning rate remains constant throughout training.
- **Exponential learning rate decay** In this schedule, the learning rate starts with an initial value and is then reduced exponentially over time. The formula for this decay is typically:

  $$learning\,rate\,(t) = initial\,learning\,rate \times decay\,rate^{t}$$

  where t represents the current epoch. The decay rate is a hyperparameter that controls the rate at which the learning rate decreases.
- **ReduceLROnPlateau** This learning rate schedule dynamically adjusts the learning rate based on the model's performance. If the monitored metric (e.g., MSE loss) stops improving or plateausing for a specified number of consecutive epochs (controlled by the "patience" parameter), it triggers a reduction in the learning rate.

For the Exponential learning rate decay and ReduceLROnPlateau schedules, a common decay rate of 0.9 was used. These different learning rate schedules were evaluated to determine their impact on training and coding efficiency. The choice of learning rate and schedule can significantly influence the convergence and performance of a neural network model.

Table 8 illustrates the impact of different learning rates and learning schedules on the final BD-rate in video compression. The results indicate when using a bigger learning rate (e.g., 1e-03), using learning schedules could help us improve model performance, but when using a smaller learning rate, a constant learning rate works best.

Table 8 provides insights into the impact of various learning rates and learning schedules on the final BD-rate in video compression. The results suggest that the choice of learning rate and learning schedule can significantly influence model performance:

- When using a larger learning rate (e.g., 1e-03), employing learning schedules such as Exponential Decay and ReduceLROnPlateau can lead to improvements in model performance. These schedules help adjust the learning rate during over-fitting, allowing for better convergence.
- In contrast, when using a smaller learning rate, a constant learning rate appears to work best. This finding suggests that with a sufficiently small learning rate, the model's weights are updated optimally without the need for learning rate decay strategies.

4.4 Loss Function Test

Both MAE (Mean Absolute Error) loss and MSE (Mean Squared Error) loss are commonly used in video compression for evaluating the quality of compressed video. These loss functions measure the difference between the original (reference) video

Table 8: Learning Rate and Learning Schedule Test. Anchor: NNVC 5.0

Learning Rate and Learning Schedule Test			BD-rate Over NNVC-5.0				
			Y-PSNR	U-PSNR	V-PSNR	EncT	DecT CPU
Constant Learning Rate	1e-02	Class C	-1.09%	-1.28%	-0.58%	99%	107%
		Class D	-0.83%	-4.87%	-4.07%	99%	107%
	1e-03	Class C	-1.28%	-1.68%	-0.52%	99%	106%
		Class D	-1.08%	-5.41%	-4.39%	99%	107%
	2e-04	Class C	**-1.35%**	**-1.94%**	**-0.56%**	98%	106%
		Class D	**-1.11%**	**-4.30%**	**-3.77%**	98%	106%
ExponentialDecay	1e-02	Class C	-1.28%	-1.82%	-0.39%	98%	107%
		Class D	-0.97%	-4.83%	-4.10%	98%	107%
	1e-03	Class C	-1.33%	-1.51%	-0.52%	98%	106%
		Class D	-1.11%	-4.69%	-4.29%	98%	106%
	2e-04	Class C	-1.15%	-0.87%	-0.37%	98%	106%
		Class D	-1.00%	-3.23%	-3.03%	98%	106%
ReduceLROnPlateau	1e-02	Class C	-1.29%	-1.83%	-0.40%	98%	107%
		Class D	-0.98%	-5.64%	-4.87%	98%	106%
	1e-03	Class C	-1.30%	-1.53%	-0.40%	98%	107%
		Class D	-1.01%	-5.14%	-4.07%	98%	106%
	2e-04	Class C	-1.14%	-0.89%	–0.36%	98%	107%
		Class D	-1.01%	-3.39%	-3.33%	98%	106%

Table 9: Loss function Test. Anchor: NNVC 5.0

Loss Function Test	BD-rate Over NNVC-5.0				
	Y-PSNR	U-PSNR	V-PSNR	EncT	DecT CPU
MSE	-1.14%	-4.93%	-4.23%	98%	107%
MAE	-0.65%	-6.40%	-5.13%	98%	107%

and the compressed video. We calculate separate loss values for each color channel, denoted as $LossFunction_Y$, $LossFunction_{Cb}$, $LossFunction_{Cr}$ corresponding to the luminance (Y), chroma blue (Cb), and chroma red (Cr) channels, respectively.

- MAE (Mean Absolute Error): MAE loss calculates the absolute difference between corresponding pixels in the reference and compressed videos and then takes the mean of these absolute differences. It gives equal weight to all errors, regardless of their magnitude. The formula for Weighted MAE loss is as follows:

$$Weighted\,MAE = \frac{Y}{YCbCr}MAE_Y + \frac{Cb}{YCbCr}MAE_{Cb} + \frac{Cr}{YCbCr}MAE_{Cr}$$

- MSE (Mean Squared Error): MSE loss calculates the squared difference between corresponding pixels in the reference and compressed videos and then takes the mean of these squared differences. It gives more weight to larger errors, making it sensitive to outliers. The formula for weighted MSE is:

$$Weighted\,MSE = \frac{Y}{YCbCr}MSE_Y + \frac{Cb}{YCbCr}MSE_{Cb} + \frac{Cr}{YCbCr}MSE_{Cr}$$

Based on the results from previous tests, it's recommended to use the following configuration settings:

- Learning Rate: 2e-4
- Learning Schedule: Constant learning rate
- Over-fitting and NNR Color Weight: 12:1:1

Table 9 illustrates the impact of different loss functions on the final BD-rate in video compression. The results indicate that MSE loss outperforms MAE loss in terms of achieving a lower BD-rate, suggesting that MSE is more effective in this specific over-fitting (content adaptation) scenario.

4.5 Model Save Metric Test

Typically, the default model save metric is loss YCbCr. As we analyze in the Color Weight Test, the luminance (Y) component is more important than other color channels. Therefore, three types of model save metrics that are experimented with:

Table 10: Model Save Metric Test. Anchor: NNVC 5.0

Model Save Metric Test			BD-rate Over NNVC-5.0		
			Y-PSNR	U-PSNR	V-PSNR
Constant Learning Rate 2e-04	Loss Y	Class C	-1.33%	-1.89%	-0.64%
		Class D	-1.14%	-4.93%	-4.28%
	dPSNR Y	Class C	-1.34%	-1.87%	-0.54%
		Class D	-1.14%	-4.98%	-4.25%
	Positive dPSNR Y	Class C	-1.34%	-1.88%	-0.54%
		Class D	-1.10%	-4.87%	-4.28%
ExponentialDecay 1e-03	Loss Y	Class C	-1.28%	-1.29%	-0.40%
		Class D	-1.15%	-4.83%	-4.10%
	dPSNR Y	Class C	-1.31%	-1.39%	-0.53%
		Class D	-1.12%	-4.52%	-4.10%
	Positive dPSNR Y	Class C	-1.31%	-1.39%	-0.51%
		Class D	-1.07%	-5.19%	-4.15%
ReduceLROnPlateau 1e-03	Loss Y	Class C	-1.27%	-1.59%	-0.59%
		Class D	-1.05%	-5.14%	-4.15%
	dPSNR Y	Class C	-1.27%	-1.31%	-0.34%
		Class D	-1.05%	-5.18%	-4.09%
	Positive dPSNR Y	Class C	-1.30%	-1.53%	-0.40%
		Class D	-1.06%	-5.14%	-4.12%

1. **Loss Y** This model save metric focuses on the luminance (Y) component's loss during training. It prioritizes minimizing the loss in the Y channel, which is typically considered the most important for visual quality. Saving the model based on Loss Y means that the model version with the lowest loss in the Y channel will be selected for further use or evaluation.

2. **dPSNR Y (delta PSNR Y)**: dPSNR Y represents the change in PSNR specifically for the Y (luminance) channel. It measures how much the PSNR of the Y component improves compared to the previous model checkpoint. Saving the model based on dPSNR Y ensures that selecting the model checkpoint provides the most significant improvement in Y channel quality.

3. **Positive dPSNR Y** Similar to dPSNR Y, Positive dPSNR Y measures the improvement in PSNR for the Y channel. However, it focuses on positive improvements only. This means it considers only the cases where the Y channel quality has improved, disregarding any negative changes. Saving the model based on Positive dPSNR Y aims to prioritize models that consistently enhance the Y channel's quality.

Table 10 provides insights into how various model save metrics affect the final BD-rate in video compression. The findings suggest that the positive dPSNR Y metric yields the most favorable outcomes.

4.6 NCTM Max Degradation Test

The Max Degradation is defined by $ref_{dPSNR} - quantized_{dPSNR}$. It could be adaptive with respect to the bitrate of the video, e.g., adaptive wrt QP and resolution. For example, for high bitrate video (low QP and/or high resolution), a lower max degradation value may be used, allowing only for a small drop in accuracy and accepting a bit more bitrate overhead from NNR. Conversely, for lower bitrate video (high QP and/or low resolution), we cannot afford much bitrate overhead from NNR, thus we use a high "max degradation" value, i.e., we allow for a bigger accuracy drop.

Table 11 presents the results of fixed max degradation experiments on class D, and Table 12 displays the results of fixed max degradation experiments on class C. These experiments involved testing max degradation values ranging from 0.001 to 0.006 for all QPs (22, 27, 32, 37, 42) and using the same max degradation settings for each QP.

Table 13 provides the results of adaptive max degradation experiments on class D. The max degradation is adaptive with respect to QP, and the value is defined as follows:

$$adpative\, max\, degradation = default + QP_{index} * interval$$

This adaptive max degradation approach takes into account the QP value, allowing for adjustments based on the video's bitrate and resolution characteristics.

In the experiments, it was found that for fixed max degradation, the best max degradation for class C is 0.002, and the best max degradation for class D is 0.005. This result aligns with expectations, as for videos with lower bitrates (class D), a higher max degradation can be tolerated. When using adaptive max degradation, the best max degradation for class D was found to be $default = 0.001$ with an interval of 0.002. But as we could see from the max degradation experiment results, compared to fixed max degradation, adaptive max degradation performs slightly better. Besides, the adaptive max degradation with respect to the content of the video may be a future topic, for example, in class D, the video Racehorses final BD rate improves a lot with the max degradation increasing.

Overall, using adaptive max degradation with respect to QP or resolution does show some improvement in performance, but not to the extent that was expected. When comparing the default setting with a fixed max degradation of 0.005, for class D BD rate Y-PSNR, the improvement ranged from -1.04% to -1.06%, and for class C BD rate Y-PSNR, it ranged from -1.34% to -1.36%.

4.7 Optimal Base Model Test

The default base model selection involves choosing the model with the highest PSNR gain and subsequently over-fitting that selected model. However, it's possible that the chosen model is already performing well and cannot be over-fitted significantly. Therefore, we propose another alternative method for selecting the base model.

Table 14 provides insights into the impact of different base model selection metrics. The findings demonstrate that over-fitting the base model with the highest PSNR gain outperforms the model with the lowest PSNR gain. This discrepancy can

Table 11: Fixed Max Degradation Test on Class D. Anchor: NNVC 5.0

NCTM Fixed Max Degradation Test Class D		BD-rate Over NNVC-5.0				
		Y-PSNR	U-PSNR	V-PSNR	EncT	DecT CPU
0.001	All	-0.98%	-5.06%	-4.22%	97%	106%
	BasketballPass	-0.58%	-3.62%	-3.64%		
	BQSquare	-2.55%	-8.70%	-8.01%		
	BlowingBubbles	-0.64%	-2.35%	-2.30%		
	RaceHorses	-0.15%	-5.57%	-2.95%		
0.002	All	-1.01%	-5.06%	-4.19%	97%	106%
	BasketballPass	-0.61%	-3.56%	-3.51%		
	BQSquare	-2.61%	-8.75%	-8.01%		
	BlowingBubbles	-0.66%	-2.37%	-2.27%		
	RaceHorses	-0.15%	-5.57%	-2.95%		
0.003	All	-1.03%	-5.11%	-4.22%	96%	106%
	BasketballPass	-0.61%	-3.56%	-3.51%		
	BQSquare	-2.61%	-8.75%	-8.01%		
	BlowingBubbles	-0.68%	-2.31%	-2.37%		
	RaceHorses	-0.23%	-5.80%	-3.00%		
0.004	All	-0.98%	-5.01%	-4.18%	97%	106%
	BasketballPass	-0.54%	-3.45%	-3.49%		
	BQSquare	-2.62%	-8.78%	-7.94%		
	BlowingBubbles	-0.68%	-2.31%	-2.37%		
	RaceHorses	-0.07%	-5.49%	-2.91%		
0.005 (default setting)	All	**-1.04%**	**-5.23%**	**-4.04%**	97%	106%
	BasketballPass	-0.57%	-3.76%	-3.45%		
	BQSquare	-2.62%	-8.78%	-7.94%		
	BlowingBubbles	-0.68%	-2.31%	-2.37%		
	RaceHorses	-0.31%	-6.05%	-2.40%		
0.006	All	-1.04%	-5.23%	-4.04%	97%	106%
	BasketballPass	-0.57%	-3.76%	-3.45%		
	BQSquare	-2.62%	-8.78%	-7.94%		
	BlowingBubbles	-0.68%	-2.31%	-2.37%		
	RaceHorses	-0.31%	-6.05%	-2.40%		

Table 12: Fixed Max Degradation Test on Class C. Anchor: NNVC 5.0

NCTM Fixed Max Degradation Test Class C		BD-rate Over NNVC-5.0				
		Y-PSNR	U-PSNR	V-PSNR	EncT	DecT CPU
0.001	All	-1.35%	-1.76%	-0.45%	100%	107%
	BasketballDrill	-1.46%	-4.87%	-2.04%		
	BQMall	-0.89%	-0.37%	1.23%		
	PartyScene	-1.94%	-0.70%	-0.76%		
	RaceHorses	-1.13%	-1.12%	-0.25%		
0.002	All	**-1.36%**	**-1.82%**	**-0.56%**	100%	107%
	BasketballDrill	-1.47%	-4.89%	-2.03%		
	BQMall	-0.90%	-0.44%	1.18%		
	PartyScene	-1.94%	-0.69%	-0.76%		
	RaceHorses	-1.14%	-1.27%	-0.64%		
0.003	All	-1.36%	-1.82%	-0.53%	100%	107%
	BasketballDrill	-1.46%	-4.92%	-1.94%		
	BQMall	-0.91%	-0.39%	-1.23%		
	PartyScene	-1.94%	-0.69%	-0.76%		
	RaceHorses	-1.14%	-1.27%	-0.64%		
0.004	All	-1.34%	-1.88%	-0.49%	100%	107%
	BasketballDrill	-1.42%	-5.28%	-1.81%		
	BQMall	-0.90%	-0.48%	-1.81%		
	PartyScene	-1.93%	-0.63%	-0.75%		
	RaceHorses	-1.12%	-1.12%	-0.63%		
0.005 (default setting)	All	-1.34%	-1.87%	-0.54%	98%	106%
	BasketballDrill	-1.45%	-5.45%	-1.98%		
	BQMall	-0.90%	-0.48%	1.23%		
	PartyScene	-1.88%	-0.41%	-0.77%		
	RaceHorses	-1.12%	-1.12%	-0.63%		
0.006	All	-1.34%	-1.87%	-0.54%	98%	106%
	BasketballDrill	-1.45%	-5.45%	-1.98%		
	BQMall	-0.90%	-0.48%	1.23%		
	PartyScene	-1.88%	-0.41%	-0.77%		
	RaceHorses	-1.12%	-1.12%	-0.63%		

Table 13: Adaptive Max Degradation Test on Class D. Anchor: NNVC 5.0

NCTM Adaptive Max Degradation Test Class D		BD-rate Over NNVC-5.0				
		Y-PSNR	U-PSNR	V-PSNR	EncT	DecT CPU
default: 0.001 interval: 0.001	All	-0.99%	-5.04%	-4.20%	99%	107%
	BasketballPass	-0.61%	-3.56%	-3.51%		
	BQSquare	-2.62%	-8.78%	-7.94%		
	BlowingBubbles	-0.72%	-2.46%	-2.43%		
	RaceHorses	-0.03%	-5.36%	-2.91%		
default: 0.001 interval: 0.002	All	**-1.06%**	**-5.15%**	**-4.06%**	100%	107%
	BasketballPass	-0.63%	-3.45%	-3.52%		
	BQSquare	-2.62%	-8.78%	-7.94%		
	BlowingBubbles	-0.68%	-2.31%	-2.37%		
	RaceHorses	-0.31%	-6.05%	-2.40%		
default: 0.001 interval: 0.003	All	-1.04%	-5.15%	-4.06%	99%	107%
	BasketballPass	-0.55%	-3.44%	-3.51%		
	BQSquare	-2.62%	-8.78%	-7.94%		
	BlowingBubbles	-0.68%	-2.31%	-2.37%		
	RaceHorses	-0.31%	-6.05%	-2.40%		
default: 0.001 interval: 0.004	All	-1.05%	-5.22%	-4.05%	99%	107%
	BasketballPass	-0.55%	-3.44%	-3.51%		
	BQSquare	-2.62%	-8.78%	-7.94%		
	BlowingBubbles	-0.68%	-2.31%	-2.37%		
	RaceHorses	-0.31%	-6.05%	-2.40%		
default: 0.002 interval: 0.001	All	-1.05%	-5.11%	-4.22%	98%	107%
	BasketballPass	-0.63%	-3.45%	-3.52%		
	BQSquare	-2.62%	-8.78%	-7.94%		
	BlowingBubbles	-0.68%	-2.31%	-2.37%		
	RaceHorses	-0.28%	-5.89%	-3.03%		
default: 0.002 interval: 0.002	All	-1.04%	-5.15%	-4.06%	100%	107%
	BasketballPass	-0.55%	-3.44%	-3.51%		
	BQSquare	-2.62%	-8.78%	-7.94%		
	BlowingBubbles	-0.68%	-2.31%	-2.37%		
	RaceHorses	-0.31%	-6.05%	-2.40%		
default: 0.002 interval: 0.003	All	-1.05%	-5.22%	-4.05%	98%	106%
	BasketballPass	-0.58%	-3.74%	-3.49%		
	BQSquare	-2.62%	-8.78%	-7.94%		
	BlowingBubbles	-0.68%	-2.31%	-2.37%		
	RaceHorses	-0.31%	-6.05%	-2.40%		

Table 14: Optimal Base Model Test. Anchor: NNVC 5.0

Optimal Base Model Test		BD-rate Over NNVC-5.0				
		Y-PSNR	U-PSNR	V-PSNR	EncT	DecT CPU
The model achieves the highest PSNR gain	Class C	-1.34%	-1.88%	-0.52%	98%	107%
	Class D	-1.11%	-4.93%	-4.28%	97%	107%
The model achieves the lowest PSNR gain	Class C	0.37%	1.25%	1.29%	99%	105%
	Class D	0.57%	-1.92%	-1.79%	99%	104%

be attributed to the fact that the model with the lowest PSNR gain requires a greater number of epochs to converge when compared to the model with the highest PSNR gain.

- The model achieves the highest PSNR gain (default setting). Then we have 3 base models and one over-fitted model.
- The model achieves the lowest PSNR gain. Then we have 2 suboptimal models and 2 good models (one is the base model with the highest PSNR gain, and the other is the overfitted model derived from the base model with the lowest PSNR gain).

4.8 Hyperparameter Selection

This section provides a summary of the results from the preceding tests, presenting the optimal hyperparameter combination in Table 15. Furthermore, Table 16 illustrates the BD rate results achieved using the configuration outlined in Table 15.

It's worth noting that the over-fitting process, which was conducted on a GPU, required approximately 2 days for classes A1 and A2, 1 day for class B, 7 hours for class C, 3 hours for class D, and 8 hours for class F. These processing times provide an overview of the computational resources involved in the over-fitting experiments.

As observed from the preceding experiments, the following key insights and summaries can be derived:

- Color weight Impact: Allocating more weights to the luma channel can lead to a higher Y-PSNR in BD-rate. This is significant because, compared to the other two color channels, the luma channel holds greater importance. Human eyes are more sensitive to changes in brightness, making the quality of the luma channel crucial in video compression and perception.
- Base Model Selection Impact: The choice of the base model significantly influences the over-fitting process.
- Content-Adaptive Over-fitting: Over-fitting the base model to suit the specific characteristics of the input video sequence improves performance. Content adaptation, achieved through over-fitting, enhances coding efficiency and perceptual quality.

Table 15: Optimal Configuration

Hyperparameter Selection	
Over-fitting Color Weight	12:1:1
NCTM Color Weight	12:1:1
Loss Function	Weighted-MSE
Epoch	200 epochs
Learning Rate	2e-4
Learning Schedule	Constant Learning Rate
NCTM Max Degradation	adaptive Max Degradation
Movel Save Metric	Positive PSNR Y
Optimal Base Model	Highest PSNR

Table 16: BD-rate of the Optimal Configuration. Anchor: NNVC 5.0

Class	Y -PSNR	Cb-PSNR	Cr-PSNR	EncT	DecT CPU
B	-2.07%	-5.54%	-1.95%	98%	107%
C	-1.34%	-1.88%	-0.52%	99%	106%
D	-1.11%	-4.93%	-4.28%	98%	107%

- NNR for Weight Compression: Utilizing the Neural Network Compression and Representation (NNR) standard for compressing weight updates is an effective strategy to minimize bitrate overhead while maintaining performance.
- Optimal Hyperparameters: Extensive experiments on hyperparameters for both the over-fitting and NNR processes are crucial. Finding the optimal set of hyperparameters is essential for achieving the best possible results.

5 Conclusions

In this work, we introduced a content-adaptive NN-based in-loop filter for VVC. Content adaptation is achieved by over-fitting the base model selected from four candidate models based on the input video sequence. As content adaptation only occurs at the encoder side, weight updates need to be signaled to the decoder side to reconstruct the over-fitted base model. For this purpose, we employed NNR, which is the NN model parameters compression standard, to code and compress the weight updates. Furthermore, rather than over-fitting all parameters of the base model, a set of parameters known as multipliers is introduced. These multipliers play a crucial role in reducing the overall bitrate overhead while simultaneously enhancing the overall performance. Additionally, we understand that the success of our approach relies not only on content adaptation but also on the careful selection and fine-tuning of hyperparameters. Thus, we conducted a series of extensive experiments to identify the optimal hyperparameters for both the over-fitting process and the NNR compression. These experiments played a pivotal role in shaping the performance of our content-adaptive in-loop filter. The results demonstrate the success of content adaptation and the significant improvements achieved through hyperparameter pruning.

References

[1] T. Wiegand, G.J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the h.264/avc video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):560–576, 2003.

[2] Gary J. Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1649–1668, 2012.

[3] Benjamin Bross, Ye-Kui Wang, Yan Ye, Shan Liu, Jianle Chen, Gary J. Sullivan, and Jens-Rainer Ohm. Overview of the versatile video coding (vvc) standard and its applications. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(10):3736–3764, 2021.

[4] Jill Boyce, Karsten Suehring, Xiang Li, and Vadim Seregin. Jvet-j1010: Jvet common test conditions and software reference configurations. 07 2018.

[5] Mohsen Abdoli. *Intra Coding Tools for Versatile Video Coding (VVC)*. PhD thesis, 06 2019.

[6] Yu-Wen Huang, Jicheng An, Han Huang, Xiang Li, Shih-Ta Hsiang, Kai Zhang, Han Gao, Jackie Ma, and Olena Chubach. Block partitioning structure in the vvc standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(10):3818–3833, 2021.

[7] Tsu-Ming Liu, Wen-Ping Lee, and Chen-Yi Lee. An in/post-loop deblocking filter with hybrid filtering schedule. *Circuits and Systems for Video Technology, IEEE Transactions on*, 17:937 – 943, 08 2007.

[8] Siwei Ma, Tiejun Huang, Cliff Reader, and Wen Gao. Avs2 ? making video coding smarter [standards in a nutshell]. *IEEE Signal Processing Magazine*, 32(2):172–183, 2015.

[9] Andrey Norkin, Gisle Bjontegaard, Arild Fuldseth, Matthias Narroschke, Masaru Ikeda, Kenneth Andersson, Minhua Zhou, and Geert Van der Auwera. Hevc deblocking filter. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1746–1754, 2012.

[10] Ming-Ze Wang, Shuai Wan, Hao Gong, and Ming-Yang Ma. Attention-based dual-scale cnn in-loop filter for versatile video coding. *IEEE Access*, 7:145214–145226, 2019.

[11] Taoran Lu, Fangjun Pu, Peng Yin, Sean McCarthy, Walt Husak, Tao Chen, Edouard Francois, Christophe Chevance, Franck Hiron, Jie Chen, Ru-Ling Liao, Yan Ye, and Jiancong Luo. Luma mapping with chroma scaling in versatile video coding. In *2020 Data Compression Conference (DCC)*, pages 193–202, 2020.

[12] Marta Karczewicz, Nan Hu, Jonathan Taquet, Ching-Yeh Chen, Kiran Misra, Kenneth Andersson, Peng Yin, Taoran Lu, Edouard François, and Jie Chen. Vvc in-loop filters. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(10):3907–3925, 2021.

[13] Xiaoyan Sun, Feng Wu, Shipeng Li, and Wen Gao. In-loop deblocking filter for block based video coding. In *6th International Conference on Signal Processing, 2002.*, volume 1, pages 33–36 vol.1, 2002.

[14] Chih-Ming Fu, Elena Alshina, Alexander Alshin, Yu-Wen Huang, Ching-Yeh Chen, Chia-Yang Tsai, Chih-Wei Hsu, Shaw-Min Lei, Jeong-Hoon Park, and Woo-Jin Han. Sample adaptive offset in the hevc standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1755–1764, 2012.

[15] Chia-Yang Tsai, Ching-Yeh Chen, Tomoo Yamakage, In Suk Chong, Yu-Wen Huang, Chih-Ming Fu, Takayuki Itoh, Takashi Watanabe, Takeshi Chujoh, Marta Karczewicz, and Shaw-Min Lei. Adaptive loop filtering for video coding. *IEEE Journal of Selected Topics in Signal Processing*, 7(6):934–945, 2013.

[16] Yann LeCun, Y. Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–44, 05 2015.

[17] Yongbing Zhang, Tao Shen, Xiangyang Ji, Yun Zhang, Ruiqin Xiong, and Qionghai Dai. Residual highway convolutional neural networks for in-loop filtering in hevc. *IEEE Transactions on Image Processing*, 27(8):3827–3841, 2018.

[18] Chuanmin Jia, Shiqi Wang, Xinfeng Zhang, Shanshe Wang, Jiaying Liu, Shiliang Pu, and Siwei Ma. Content-aware convolutional neural network for in-loop filtering in high-efficiency video coding. *IEEE Transactions on Image Processing*, PP:1–1, 01 2019.

[19] Dandan Ding, Lingyi Kong, Guangyao Chen, Zoe Liu, and Yong Fang. A switchable deep learning approach for in-loop filtering in video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(7):1871–1887, 2020.

[20] Shufang Zhang, Zenghui Fan, Nam Ling, and Minqiang Jiang. Recursive residual convolutional neural network- based in-loop filtering for intra frames. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(7):1888–1900, 2020.

[21] Jian Yue, Yanbo Gao, Shuai Li, Hui Yuan, and Frédéric Dufaux. A global appearance and local coding distortion based fusion framework for cnn based filtering in video coding. *IEEE Transactions on Broadcasting*, 68(2):370–382, 2022.

[22] Zhijie Huang, Jun Sun, Xiaopeng Guo, and Mingyu Shang. Adaptive deep reinforcement learning-based in-loop filter for vvc. *IEEE Transactions on Image Processing*, 30:5439–5451, 2021.

[23] Yat Hong Lam, Alireza Zare, Çağlar Aytekin, Francesco Cricri, Jani Lainema, Emre B. Aksu, and Miska M. Hannuksela. Compressing weight-updates for image artifacts removal neural networks. *ArXiv*, abs/1905.04079, 2019.

[24] Yat-Hong Lam, Alireza Zare, Francesco Cricri, Jani Lainema, and Miska M. Hannuksela. Efficient adaptation of neural network filter for video compression. In *Proceedings of the 28th ACM International Conference on Multimedia*, MM '20, page 358–366, New York, NY, USA, 2020. Association for Computing Machinery.

[25] Maria Santamaria, Francesco Cricri, Jani Lainema, Ramin G. Youvalari, Honglei Zhang, and Miska M. Hannuksela. Content-adaptive neural network post-processing filter with nnr-coded weight-updates. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 2251–2255, 2022.

[26] Heiner Kirchhoffer, Paul Haase, Wojciech Samek, Karsten Müller, Hamed Rezazadegan-Tavakoli, Francesco Cricri, Emre B. Aksu, Miska M. Hannuksela, Wei Jiang, Wei Wang, Shan Liu, Swayambhoo Jain, Shahab Hamidi-Rad, Fabien Racapé, and Werner Bailer. Overview of the neural network compression and representation (nnr) standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(5):3203–3216, 2022.

[27] Simon Wiedemann, Heiner Kirchhoffer, Stefan Matlage, Paul Haase, Arturo Marban, Talmaj Marinč, David Neumann, Tung Nguyen, Heiko Schwarz, Thomas Wiegand, Detlev Marpe, and Wojciech Samek. Deepcabac: A universal compression algorithm for deep neural networks. *IEEE Journal of Selected Topics in Signal Processing*, 14(4):700–714, 2020.

[28] N. Ahmed, T. Natarajan, and K.R. Rao. Discrete cosine transform. *IEEE Transactions on Computers*, C-23(1):90–93, 1974.

[29] G.K. Wallace. The jpeg still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1):xviii–xxxiv, 1992.

[30] Zhijie Huang, Jun Sun, Xiaopeng Guo, and Mingyu Shang. Adaptive deep reinforcement learning based in-loop filter for vvc. *IEEE Transactions on Image Processing*, PP:1–1, 06 2021.

[31] Kiran Misra, Frank Bossen, and Andrew Segall. On cross component adaptive loop filter for video compression. In *2019 Picture Coding Symposium (PCS)*, pages 1–5, 2019.

[32] Xinfeng Zhang, Ruiqin Xiong, Weisi Lin, Jian Zhang, Shiqi Wang, Siwei Ma, and Wen Gao. Low-rank-based nonlocal adaptive loop filter for high-efficiency

video compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(10):2177–2188, 2017.

[33] Suhong Wang, Xiang Zhang, Shanshe Wang, Siwei Ma, and Wen Gao. Adaptive wavelet domain filter for versatile video coding (vvc). In *2019 Data Compression Conference (DCC)*, pages 73–82, 2019.

[34] Jianle Chen, Marta Karczewicz, Yu-Wen Huang, Kiho Choi, Jens-Rainer Ohm, and Gary J. Sullivan. The joint exploration model (jem) for video compression with capability beyond hevc. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(5):1208–1225, 2020.

[35] Vardan Papyan and Michael Elad. Multi-scale patch-based image restoration. *IEEE Transactions on Image Processing*, 25(1):249–261, 2016.

[36] Mingze Wang, Shuai Wan, Hao Gong, Yuanfang Yu, and Yang Liu. An integrated cnn-based post processing filter for intra frame in versatile video coding. In *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1573–1577, 2019.

[37] Ren Yang, Mai Xu, Tie Liu, Zulin Wang, and Zhenyu Guan. Enhancing quality for hevc compressed videos. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(7):2039–2054, 2019.

[38] Fan Zhang, Chen Feng, and David R. Bull. Enhancing vvc through cnn-based post-processing. In *2020 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, 2020.

[39] Maria Santamaria, Ruiying Yang, Francesco Cricri, Honglei Zhang, Jani Lainema, Ramin G. Youvalari, Hamed R. Tavakoli, and Miska M. Hannuksela. Overfitting multiplier parameters for content-adaptive post-filtering in video coding. In *2022 10th European Workshop on Visual Information Processing (EUVIP)*, pages 1–6, 2022.

[40] H. Wang, J Chen, Reuze A.M.Kotra, and M. Karczewicz. Ee1-1.4: Tests on neural network-based in-loop filter with constrained computational complexity. *Journal of Advanced Video Compression*, 2021.

[41] N. Liu, E. Segall, and R.-L. Liao. Jvet common test conditions and evaluation procedures for neural network-based video coding technology. 2021.

[42] Andrew L. Maas. Rectifier nonlinearities improve neural network acoustic models. 2013.

[43] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[44] Di Ma, Fan Zhang, and David Bull. Bvi-dvc: a training database for deep video compression. *IEEE Transactions on Multimedia*, 2021.

[45] H. Wang, J Chen, Reuze A.M.Kotra, and M. Karczewicz. Ee1-1.4: Tests on neural network-based in-loop filter with constrained computational complexity. *Journal of Advanced Video Compression*, 2021.

[46] Alain Horé and Djemel Ziou. Image quality metrics: Psnr vs. ssim. In *2010 20th International Conference on Pattern Recognition*, pages 2366–2369, 2010.

[47] Compression of neural networks for multimedia content description and analysis. Standard, ISO/IEC, 2022.

[48] Paul Haase, Daniel Becking, Heiner Kirchhoffer, Karsten Müller, Heiko Schwarz, Wojciech Samek, Detlev Marpe, and Thomas Wiegand. Encoder optimizations for the nnr standard on neural network compression. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 3522–3526, 2021.

[49] Gisle Bjøntegaard. Calculation of average psnr differences between rd-curves. 2001.

[50] Ruiying Yang, Maria Santamaria, Francesco Cricri, Honglei Zhang, Jani Lainema, and Miska M. Hannuksela. Ahg11: Content-adaptive neural network loop-filter. In *Journal of Advanced Video Compression*, 2023.