OPTICAL FLOW-BASED MEDIA COMPRESSION

AFSANA AHSAN JENY

ISTANBUL 2022

OPTICAL FLOW-BASED MEDIA COMPRESSION

A THESIS SUBMITTED TO THE

GRADUATE SCHOOL

OF

BAHÇEŞEHİR UNIVERSITY

BY

AFSANA AHSAN JENY

IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS

FOR

THE DEGREE OF MASTER OF COMPUTER ENGINEERING

IN THE DEPARTMENT OF GRADUATE SCHOOL

ISTANBUL, 2022

T.C.
BAHCESEHIR UNIVERSITY
GRADUATE SCHOOL

...../...../.....

MASTER THESIS APPROVAL FORM

Program Name:	Computer Engineering (Thesis)
Student's Name and Surname:	Afsana Ahsan Jeny
Name Of the Thesis:	Optical Flow-based Media Compression
Thesis Defense Date:	12/08/2022

This thesis has been approved by the Graduate School which has fulfilled the necessary conditions as Master thesis.

Prof. Dr.Ahmet ÖNCÜ
Institute Director

This thesis was read by us, quality and content as a Master's thesis has been seen and accepted as sufficient.

	Title/Name	**Signature**
Thesis Advisor's	Assist. Prof, Dr. Md Baharul Islam	
Member's	Assist. Prof. Dr. Tarkan Aydin	
Member's	Assist. Prof. Dr. A. F. M. Shahen Shah	

DEDICATION

To my family and professor, for their endless support and for always being there for me.

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

Name, Surname: Afsana Ahsan Jeny

Signature:

ACKNOWLEDGEMENTS

I would like to take a moment and thank the following people, without whom I would not have been able to complete this thesis, and without whom I would not have made it through my master's degree!

The Faculty of Engineering and Applied Sciences at BAHÇEŞEHİR UNIVERSITY, especially to my supervisor ASSIST. PROF, DR. MD BAHARUL ISLAM whose insight and knowledge into the subject matter steered me through this research. And special thanks to my peer and friend Masum Shah Junayed who helped me in tough times and sailed me through rough waves.

And my biggest thanks to my family (especially my beloved mother) for all the support you have shown me through this degree, the culmination of two years of distance learning. For my parents who are an inspiration for me and who make me strive harder in life. Thanks for all your support, without which I would have stopped these studies a long time ago, you have been amazing.

İstanbul, 2022 Afsana Ahsan Jeny

ABSTRACT

OPTICAL FLOW-BASED MEDIA COMPRESSION

Afsana Ahsan Jeny

Master of Science in Computer Engineering

Thesis Supervisor: Assist. Prof. Dr. MD BAHARUL ISLAM

August 2022, 103 Pages

Recently, learned image compression algorithms have shown incredible performance compared to classic hand-crafted image codecs. Despite its considerable achievements, the fundamental disadvantage is not optimized for retaining local redundancies, particularly non-repetitive patterns, which negatively influence the reconstruction quality. On the other hand, research in video compression has seen significant advancement in the last several years. However, the existing learning-based algorithms continue to be plagued by erroneous motion compression and ineffective motion compensation architectures, resulting in compression errors with a lower rate-distortion trade-off. This thesis paper introduces the autoencoder-style network-based efficient image compression method, which contains three novel blocks, i.e., adjacent attention block, Gaussian merge block, and decoded image refinement block, to improve the overall image compression performance. The transformer-based model is also introduced in this paper to compare with CNN. After that, to overcome these challenges, we present an end-to-end video compression method through a set of primary operations (e.g., motion estimation, motion compression, motion compensation, residual compression, and artifact contraction) differently. A comprehensive ablation study demonstrates the effectiveness of the proposed blocks and modules for image and video compression. Experimental results show the state-of-the-art and competitive performance of the proposed method on various benchmark datasets.

Keywords: Convolutional Neural Network, Transformer, Attention Block, Artifact Reduction.

ÖZET

OPTİK AKIŞ TABANLI MEDYA SIKIŞTIRMA

Afsana Ahsan Jeny

Bilgisayar Mühendisliği Bilim Ustası

Tez Danışmanı: Assist. Prof. Dr. MD BAHARUL ISLAM

Ağustos 2022, 103 Sayfa

Son zamanlarda öğrenilen görüntü sıkıştırma algoritmaları, klasik el yapımı görüntü kodeklerine kıyasla inanılmaz bir performans göstermiştir. Önemli başarılarına rağmen, temel dezavantaj, yeniden yapılanma kalitesini olumsuz etkileyen, özellikle tekrarlamayan kalıplar olmak üzere, yerel fazlalıkları korumak için optimize edilmemiştir. Öte yandan, video sıkıştırma araştırmaları son birkaç yılda önemli ilerlemeler kaydetti. Bununla birlikte, mevcut öğrenme tabanlı algoritmalar, hatalı hareket sıkıştırma ve etkisiz hareket dengeleme mimarileri tarafından rahatsız edilmeye devam ediyor ve bu da daha düşük oran-bozulma dengesi ile sıkıştırma hatalarına neden oluyor. Bu tez makalesi, genel görüntü sıkıştırma performansını iyileştirmek için üç yeni blok, yani bitişik dikkat bloğu, Gauss birleştirme bloğu ve kodu çözülmüş görüntü iyileştirme bloğu içeren otomatik kodlayıcı tarzı ağ tabanlı verimli görüntü sıkıştırma yöntemini tanıtmaktadır. Transformatör tabanlı model de bu yazıda CNN ile karşılaştırmak için tanıtılmıştır. Bundan sonra, bu zorlukların üstesinden gelmek için, farklı bir dizi birincil işlem (örneğin, hareket tahmini, hareket sıkıştırma, hareket telafisi, artık sıkıştırma ve yapay daralma) yoluyla uçtan uca bir video sıkıştırma yöntemi sunuyoruz. Kapsamlı bir ablasyon çalışması, görüntü ve video sıkıştırma için önerilen blokların ve modüllerin etkinliğini gösterir. Deneysel sonuçlar, önerilen yöntemin çeşitli kıyaslama veri kümeleri üzerindeki en gelişmiş ve rekabetçi performansını göstermektedir.

Anahtar Kelimeler: Evrişimsel Sinir Ağı, Transformatör, Dikkat Bloğu, Artifakt Azaltma.

CONTENTS

TABLES

FIGURES

1. INTRODUCTION

IoT devices will account for 50% of all networked devices and connections around the globe by the year 2023, as stated in recent publications [1, 2] published by Cisco, a significant business in the market for networking technologies. There will be 3.6 gadgets linked to the internet for every human being, and these devices will be sharing data with the rest of the world. Such devices have various applications that depend on pictures, such as video surveillance, automation, and self-driving vehicles. They will all contribute significantly to the rise of internet traffic. This growth will be driven in large part by these devices. On the other hand, images have taken on an increasingly important role in how we communicate and take in information. This is true on social media platforms such as Instagram and Snapchat, communication platforms such as Skype and Zoom, and advertising platforms such as YouTube and Netflix. Videos and images will represent a significant portion of the global IP traffic, which, according to Cisco, is projected to increase by a factor of three between 2018 and 2023. This is due primarily to the proliferation of smartphones and the growing proportion of the global population that has access to these devices. All of this "Big Data" presents a difficulty for data management (storage and transmission). However, it also offers a treasure trove for machine learning (ML) and computer vision, which can provide valuable strategic information and take advantage of its potential worth.

1.1. Problem Definition Of Image and Video Compression

Compressing an image brings about a reduction in the amount of spatial redundancy found in the image as well as an optimization of the amount of bandwidth and storage space used in various applications. These applications include video compression, online advertising, and the exchange of professional photographs. In order to achieve higher levels of

compression efficiency, traditional image compression algorithms (Wallace, 1992) (Skodras et al., 2001) (fau,) (Bellard,) rely on hand-crafted processes with extensive dependencies. For instance, JPEG uses the discrete cosine transform (Wallace, 1992) (DCT). On the other hand, JPEG2000 employs discrete wavelet transformations (DWT) to shift an image pixel to the frequency domain and deconstruct multi-scale decomposition into spectral bands accordingly. This is cited in (Skodras et al., 2001). On the other hand, they produce artifacts at the picture boundaries, which become undetectable at high bit rates. In more recent video codecs, such as VVC (fau,), an intra prediction and an in-loop filter are included for intra-frame coding. It is also used in BPG (Bellard,), an image codec, to remove redundant and unnecessary features and enhance the reconstruction frame's quality. On the other hand, traditional compression methods cannot be improved from beginning to finish, which hinders their overall rate-distortion (RD) optimization performance (especially in similarity index) and their learning capabilities.

In the present era, image compression approaches (Ballé et al., 2016) (Ballé et al., 2018) (Kim et al., 2020) (Minnen et al., 2018) (Li et al., 2020)(Lee et al., 2018) based on deep learning surpass standard algorithms in terms of rate-distortion (RD) performance. For instance, Balle et al. (Ballé et al., 2016) present an end-to-end picture compression utilizing a convolutional neural network (CNN) based autoencoder. In particular, context-adaptive entropy models for learning image compression are becoming more popular due to their ability to provide superior performance when compared to standard codecs. The entropy model was given a hyperprior in the research (Ballé et al., 2018) so that more bits may be added to it to better properly characterize it. Minnen et al. (Minnen et al., 2018) employed the auto-regressive prior information to develop an appropriate entropy model, which allowed them to obtain compression effectiveness comparable to or even greater than that of the traditional codec (Bellard,). A very comparable concept was presented in the work

that was done in (Lee et al., 2018), which achieved a context-adaptive entropy model by taking into consideration two distinct types of contexts: bit-consuming contexts (also known as hyperprior) and bit-free contexts (also known as an auto-regressive model). Although these methods improve compression performance, they also significantly increase compression artifacts (Pham et al., 2020). This is because the compression artifacts are caused by the quantization process that occurs during the entropy coding, and the corresponding fields in latent space are stacked as a result.

However, estimating the optical flow (key contributor in video compression) of light across a scene is one of the first steps in video analysis. Optical flow is the method by which the movements, surfaces, and edges of an item in a video sequence are described. Optical flow is produced by the relative velocity of an observer and a visual scene. Optical flow in computer vision has a wide variety of real-world applications, some of which include video segmentation, action identification, surveillance systems, tracking, and autonomous vehicles. These are only a few examples of the many possible applications of optical flow. Getting an accurate reading of the optical flow rate is still a difficult problem to solve.

The problem of large displacement, caused by things moving quickly, has been a persistent one in the field of optical flow for a good number of years. A number of scholars attempted to solve this problem by using a method that went from broad to specific. For instance, (Li et al., 2017) attempted to enhance the coarse-to-fine network by matching descriptors and making use of closest neighbor attributes (NNF). a regular grid construction of the mappings field is presented by Chen et al. (Chen and Koltun, 2016), which is aimed to optimize the energy function in the optical flow; nevertheless, they ran into an occlusion issue. In addition to the issue of excessive displacement, this problem is becoming a substantial barrier to overcome when attempting to estimate optical flow. A

forward-backward consistency evaluation approach is used by a few different strategies (Chen and Koltun, 2016) (Tu, 2015) (Yang and Soatto, 2017) in order to address the occlusion region. However, despite the fact that this method possesses a number of benefits, it also results in a number of drawbacks, including the following: resizing the image and the flow results in errors caused by interpolation (Tu, 2015), and fine structure over smoothing results in an inability to detect "tiny and quicker objects," which poses a significant risk to the accuracy of the method (Yang and Soatto, 2017).

On the other hand, in order to provide high-quality video services despite the limits imposed by transmission networks and storage space, video compression is also essential. Over the course of the last several years, a great deal of research and development has gone into the creation of a variety of well-known video compression techniques, such as H.264 (Wiegand et al., 2003), H.265 (Sullivan et al., 2012), and VVC (Bross et al., 2019). These approaches depend on handmade modules such as block motion estimation and the Discrete Cosine Transform (DCT) to reduce the high amount of redundancy in video sequences, which may be spatial or temporal. Nevertheless, during block-wise operation, undesired artifacts, most notably blocking, ringing, and blurring artifacts, are always formed along the boundaries (Lim and Park, 2011). This rippling phenomenon has a detrimental effect on the deterioration of video quality and negatively influences the user experience. In addition, these algorithms are unable to optimize in a way that considers the whole process. Because of this, it is necessary to improve the compression performance, which calls for more research. Recently, deep neural network (DNN) based image compression methods (Theis et al., 2017) (Ballé et al., 2018) (Minnen et al., 2018) (Lee et al., 2018) have attracted a substantial amount of academic interest for two reasons: (i) they permit the utilization of non-linear transformations, and (ii) they do not need handcrafting attributes. These investigations have built a conceptual framework for

deep autoencoders in image codecs to enhance the rate-distortion trade-off (e.g., low MSE or high SSIM). Its primary emphasis is on the development of frameworks that are optimized from beginning to finish while concurrently working to improve all modules (e.g., transformation, quantization, entropy estimation).

1.2. Motivation

The attention mechanism is being used to acquire more details from the latent space while repressing unimportant information to allocate more bits (Liu et al., 2019) (Cheng et al., 2020) (Chen et al., 2021) to improve the overall effectiveness of image compression. It has been shown that the non-local attention mechanism (Song et al., 2021) is successful in a variety of visual activities (i.e., semantic segmentation). The authors Liu et al. (Liu et al., 2019) reference employ non-local attention to construct implicit significance masks in order to drive the adaptive processing of latent characteristics. On the other hand, Cheng et al. (Cheng et al., 2020) eliminate the non-local section so that image compression may be learned in a more straightforward manner. The most recent study, conducted in (Chen et al., 2021), used non-local attention processes to improve the adaptive processing of latent information. It enables the compression algorithm to assign extra bits to locations that are difficult to compress (e.g., edges and textures). Nevertheless, there are a few problems with this approach. Firstly, their non-local attention (working in a single direction) has no influence on the vertical and horizontal weights to build the corresponding broad field and acquire useful characteristics to boost RD efficiency. This is because they are working in a single direction. Second, in entropy coding, a single mask cannot remove redundant latent feature data even if many masks are used. Thirdly, the compression artifacts are greatly exacerbated as a consequence of allocating bits to places that are not critical, which ultimately results in unsatisfactory recreated pictures.

As a result of the success of the transformer in the field of natural language processing

(Vaswani et al., 2017) (Wu et al., 2021) (Yan et al., 2021) (Liu et al., 2021b), researchers have been investigating its potential for use in conjunction with CNNs in the field of image compression (Bai et al., 2021) (Lu et al., 2021). CNNs and transformers have been used in a model for end-to-end image compression recently proposed by Bai et al. (Bai et al., 2021). Instead of using the patch stem of the vision transformer, they used a CNN image encoder to encrypt the feature information (e.g., image splitting and embedding). The usage of the compressed features accomplishes image classification by the ViT decoder. A deconvolutional neural network is used to rebuild the picture once the features have been input into it. However, because of its limited receptive field, long-range correlations problems are caused, and the complexity of the self-attention layer is quadratic, which raises both the computational and memory costs. Image compression was performed more recently by Lu et al. (Lu et al., 2021) using primary and hyper encoder-decoders that were generated by neural transformation units (NTUs). Each NTU has a convolution layer in addition to Swin Transformer Blocks (STBs), (Liu et al., 2021b). After that, a casual attention module (CAM) is constructed to use both hyper and autoregressive priors for adaptive context modeling (Minnen et al., 2018) (Cheng et al., 2020) (Chen et al., 2021). Nonetheless, there are several obstacles on their way: (a) the autoregressive model caused heavy computational expenses during decoding as a result of the serial coding strategy; (b) the adaptive context modeling does not enable global latent feature relationship, which particularly affects with entropy optimization; and (c) the computational complexity is greater as a consequence of the self-attention layer.

On the other hand, the U-Net-based works were created (Dosovitskiy et al., 2015) (Ilg et al., 2017) (Bailer et al., 2015) (Schuster et al., 2018) (Xu et al., 2017) (Yang and Ramanan, 2019) (Jayasundara et al., 2021) (Kong et al., 2021) using the encoder-decoder architecture of CNN and aimed to eliminate the preceding shortcomings. However, these

architectures based on the U-Net are experiencing certain difficulties. Because of the compressing and stretching of the feature map in the encoder and decoder portions, "many particular aspects of an image might be lost from the pixels," which is harmful to dense motion estimation and the production of specific flow fields. Some other works, such as (Ranjan and Black, 2017) (Sun et al., 2018) (Hui et al., 2018) (Sun et al., 2019) (Zhao et al., 2020) (Eldesokey and Felsberg, 2021) utilized spatial pyramid networks to solve the prior problems through the application of many traditional concepts. These concepts included spatial pyramiding, warping, and post-processing. However, as they were extracting features from the spatial pyramid networks, they encountered losses in each pyramid while computing the sequential flow. This was a challenge for them. As a direct consequence of this, the rate of these approaches was poor, and they were unable to accurately maintain motion within the feature map.

Recent works to end-to-end trainable optical flow that are notable include PWC-Net (Sun et al., 2018) (Sun et al., 2019), LiteFlowNet (Hui et al., 2018), and LiteFlowNet2 (Hui et al., 2020). These architectural designs made use of many constraint correlations layers, each of which worked on a flow estimate and bent the features at the corresponding level. They used the iterative refining procedure so that they could bring the flow field up to date. However, the most significant issue with the methods currently being used is the flow estimate inaccuracy. This mistake is brought about by the boundary blur that occurs on a multi-scale picture in pyramid networks while the features are extracted. As a result of the utilization of the iterative refinement approach in these ways, the created mistakes have shown themselves in each and every iterative flow field. Because of this, incorrect estimates of optical flows are produced.

Recent deep learning-based video compression methods, such as (Lin et al., 2020) (Wu et al., 2020) (Feng et al., 2020) (Yang et al., 2020b) (Hu et al., 2021) (Li et al., 2021b)

(Sheng et al., 2021) concentrate on different modules to enhance its overall performance. For example, to improve the system's performance and visual quality of the reconstructed frame, Lin et al. (Lin et al., 2020) utilized multi-reference frames and MV fields for a more accurate current frame and two refinement networks to remove the MV and residual errors. Wu et al. (Wu et al., 2020) introduced a post-processing step in the baseline (Lu et al., 2019a) and Feng et al. (Feng et al., 2020) focused on enhancing the residual frame for implementing residual autoencoders. Yang et al. (Yang et al., 2020a) proposed a hierarchical technique and a recurrent enhancement module to improve the overall performance. In contrast, Li et al. (Li et al., 2021b) and Sheng et al. (Sheng et al., 2021) gave priority to improving the residual prediction through the temporal contextual encoder-decoder network. However, no one explores the motion vector compression network, which is the primary factor of the video compression system. The earlier methods employed the variational autoencoder style network (Ballé et al., 2018), where a new latent vector from the encoder is sent to the decoder at each time interval. In this case, the code vector must include all of the input image features in the decoder, which burdens CNN's memory. Though (Hu et al., 2021) (Feng et al., 2020) executed residual blocks to acquire additional deep features, unfortunately, this is inadequate to reduce the computational load. As a result, the capacity of the model was limited. Therefore, a low-cost motion compression network is obvious for learning-based video compression. Besides, it is also significant to emphasize the other modules (e.g., motion compensation) and reduce compression errors (caused by the quantization process in the MV and residual encoder-decoder network).

1.3. Contributions

Our contributions to this thesis paper are summarized as follows:

- We present an end-to-end autoencoder-based image compression CNN model to

improve the overall image compression performance. Three new blocks, i.e., an adjacent attention block (AAB), a Gaussian merge block (GMB), and a decoded image refinement block (DIRB), are included in this model. Besides, we introduce an end-to-end image compression transformer that employs the main and hyper encoder-decoders, including two new blocks such as a multi-head spatial reduction self-attention (MHSRSA) and a Casual Global Anticipation Module (CGAM). It is implemented without convolutions and the global receptive field to tackle the long-range correlation issues in image compression. An extensive experiment is conducted on two publicly available datasets (KODAC (kod,) and CLIC2021 (cli, a)). Our method shows state-of-the-art performance in both datasets and simultaneously reduces computational complexity.

- We introduce a novel method for optical flow estimation based on a deep pyramid network, namely DeepPyNet. This method consists of a recurrent pyramid-based feature extractor, cost volume, and flow decoder network. To evaluate the proposed DeepPyNet, the two widely used datasets MPI-Sintel and KITTI 2015, are used. It achieves state-of-the-art results in both the clean and final pass of MPI-Sintel and shows the competitive performance when fine-tuning the large datasets FlyingChairs and FlyingThings3D.

- An end-to-end video compression method is proposed, including motion estimation, motion compression with deep residual attention split (DRAS) block, motion compensation with channel residual block (CRB), artifact contraction module (ACM), residual compression, and entropy coding employed based on the rate-distortion trade-off with a single loss function. As a result, our proposed method may readily include new state-of-the-art methodologies for optical flow estimations, image compression, motion and residual frame prediction, and rate control. We per-

formed an extensive ablation study and experiments on four datasets (i.e., HEVC, UVG, VTL, and MCL-JCV), showing competitive performance. Besides, it outperforms the first baseline method, DVC, and a recent RLVC, FVC, and DCVC in terms of PSNR and MS-SSIM at lower bit rates.

2. LITERATURE REVIEW

2.1. Image Compression Methods

Classical methods Image compression techniques are primarily concerned with reducing the levels of spatial redundancies present in images. For example, converting photos from the pixel domain to the frequency domain is simpler to compress. For instance, JPEG (Wallace, 1992) applies the discrete cosine transform. In contrast, JPEG2000 (Skodras et al., 2001) applies the discrete wavelet transform, which is handcrafted. To reduce data redundancy, high-frequency information is separated from low-frequency information, and bits are allocated according to the signal significance. Entropy coding such as Huffman (Liu et al., 2022) (Ranjan, 2021), hashing (Monga and Evans, 2006), and arithmetic coding (Witten et al., 1987) (Guo et al., 2021) is also utilized to increase the lossless compression performance of the image.

Currently, the intra-prediction approach (Bellard,) (fau,), which is often used in video compression, has also been employed for image compression. For example, the BPG (Bellard,) standard is based on the HEVC/H.265 (Sullivan et al., 2012) image compression standard, which delivers the highest possible image compression results in comparison to prior methods, such as JPEG and JPEG2000. The prediction-transform approach is used in the BPG standard (Bellard,), and 35 encoding options are utilized to create the reconstructed image, which also decreases redundant data. Then, bigger computing units, more forecast methods, more transform varieties, and more coding facilities are all supported by VVC (fau,). Furthermore, the hybrid techniques employ both conventional compression techniques and the most current learning super-resolution strategies, such as (Cheng et al., 2018), to achieve higher compression ratios. However, traditional algorithms are created by hand-crafted components (such as entropy coding). On the other

hand, our deep learning-based approach is end-to-end and can learn, resulting in improved compression efficacy.

Deep learning-based methods Deep neural networks (DNNs) have shown to be useful for various computer vision applications in recent years, namely super-resolution, denoising, and object recognition. Some recent studies have attempted to conduct excellent neural networks representation capabilities to improve the performance of image compression (Toderici et al., 2015) (Ballé et al., 2015) (Ballé et al., 2016) (Toderici et al., 2017) (Theis et al., 2017) (Agustsson et al., 2017) (Johnston et al., 2018) (Ballé et al., 2018) (Minnen et al., 2018) (Li et al., 2018) (Mentzer et al., 2018) (Lee et al., 2018) (Lu et al., 2019b) (Agustsson et al., 2019) (Cheng et al., 2020). Toderici et al. (Toderici et al., 2015) developed the first learning-based image compression framework based on a recurrent neural network (RNN). Various bitrates may be generated using a single model in their method. Compared to BPG, (Johnston et al., 2018) introduces more complex RNN components and efficient reconstruction approaches to obtain equivalent or even superior MS-SSIM (Wang et al., 2003) results. Although some of these approaches (Toderici et al., 2015) (Toderici et al., 2017) (Johnston et al., 2018) are aimed to reduce the bitrates, the rate-distortion (RD) trade-off is not considered.

By improving the RD performance, Balle' et al. (Ballé et al., 2015) introduced a CNN-based framework with the generalized divisive normalization (GDN) layer, which is effective for simulating nonlinear transformations that have been frequently employed in subsequent approaches (Ballé et al., 2016) (Ballé et al., 2018) (Minnen et al., 2018) (Lee et al., 2018) (Cheng et al., 2019) (Cheng et al., 2020) (Chen et al., 2021). However, to improve the RD performance, these methods conduct the Gaussian Model (GM) distribution that is still short of encoding latent features by effectively estimating the conditional statistics. According to Rippel et al., (Rippel and Bourdev, 2017), a feature pyramid network

(FPN) was introduced to obtain more valuable features. However, this would also lead to redundant information since convolutional methods exchange features. Li et al. (Li et al., 2018) suggested the use of a significance map to alter the bit allocation of images, which they found to be effective. A three-layer convolutional neural network branch was trained to create the significance map. However, the explicit learning material requires weight, which raises the computing cost. It is also tricky to adaptively assign bits for in-depth features, as described in (Li et al., 2018).

In the training process, some methods (Agustsson et al., 2017) (Agustsson et al., 2019) employed an adversarial network (GAN) as a distortion assessment to lead the decoder to create more feasible pattern structures, which tends to result in reconstructed images of decent visual quality. But the pattern structures obtained in this way are not actual textures and lack fidelity. Recent studies on adaptive learning of feature significance have shown that attention strategies are quite effective. Considerable progress has been achieved in areas like as natural language processing (Jain and Wallace, 2019) and semantic segmentation (Song et al., 2021). Moreover, noise removal and super-resolution efficiency can be dramatically improved by incorporating non-local block (NLB) into neural networks (Zhang et al., 2019) (Sun and Tappen, 2011). In image compression, some methods (Zhou et al., 2019) (Liu et al., 2019) (Cheng et al., 2020) (Chen et al., 2021) employ attention mechanisms that allow spatially adaptive feature response for more difficult locations (i.e., patterns, saliency) in order to allocate more bits. For example, (Zhou et al., 2019) introduced an improvement unit that functions on full-resolution photos to eliminate compression artifacts by filtering the reconstructed images using a simple neural network. (Liu et al., 2019) (Cheng et al., 2020) (Chen et al., 2021) employed residual non-local attention mechanisms to improve the RD performance and compression artifacts due to the quantization procedure. However, these proposed attention mechanisms can't be exploited

features in both directions (vertical and horizontal) because of their one-way weight allocation. Therefore, allocating more bits in complex regions (i.e., patterns, edges) is not fully explored to improve the final reconstructed image.

In contrast, we propose an adjacent attention block that uses distinct weights in the horizontal and vertical directions for feature maps to maintain only the most relevant information while eliminating unnecessary information, such as a complicated natural background, which significantly impacts the performance of RD. Furthermore, in order to decrease compression artifacts, we have included a refinement block, which is capable of smoothing out and improving the visualization of the reconstructed image.

2.2. Video Compression Methods

Classical methods Traditional video codecs such as H.264, H.265, and VVC are highly effective and work well in real applications. Their design is based on a predictive coding architecture that requires the integration of many modules. Every module is designed independently. There is no possibility of improving the whole system simultaneously. Numerous DNN-based improvements have been created to increase the performance of classic codecs, including entropy coding (Song et al., 2017), mode determination (Liu et al., 2016), intra-prediction, and residual coding (Chen et al., 2017). These modifications boost the performance of adjacent blocks only, requiring the development of further end-to-end compression techniques. A block-based learning strategy was devised; however, the model was plagued by block artifacts at the boundary, resulting in poor performance. Additionally, the motion information is sent via classic block motion estimation, which results in inferior performance (Wiegand et al., 2003) (Sullivan et al., 2012) (Bross et al., 2019) (Lim and Park, 2011) (Chen et al., 2019). However, this network is not end-to-end optimized, and motion information is not compressed utilizing deep techniques in this system.

Deep-learning based methods Recently, deep learning-based approaches have been utilized for creating an end-to-end video compression system (Kin and Coker, 2017) (Lu et al., 2019a; Djelouah et al., 2019) (Habibian et al., 2019) (Lin et al., 2020) (Agustsson et al., 2020) (Lu et al., 2020) (Hu et al., 2020) (Wu et al., 2020) (Hu et al., 2021). For example, Kin et al. (Kin and Coker, 2017) proposed an overfitted bidirectional recurrent convolutional neural network (RCNN) based on an autoencoder. Their RCNNs can learn the temporal information from consecutive frames. But, the RCNN didn't obtain more accurate motion information for the final reconstructed frame due to low feature channel. As a result, they are unable to attain state-of-the-art reduction rates and performances. Following this, Lu et al. (Lu et al., 2019a) introduced the first end-to-end optimal video compression system, in which deep neural networks are employed to substitute all of the essential modules of the H.264/H.265. Especially, to obtain the MV, they employed pixel-level optical flow network (Ranjan and Black, 2017) and improved the rate-distortion trade-off; they utilized deep auto-encoder type networks for MV compression, which were initially used in image compression studies (Ballé et al., 2016). Then, Djelouah et al. (Djelouah et al., 2019) presented a network that integrates motion compression with image synthesis using an interpolation-based video compression technique. Accurate motion estimation has been implemented to achieve decent compression performance, and state-of-the-art optical flow estimation networks (Dosovitskiy et al., 2015) (Ranjan and Black, 2017) (Sun et al., 2018) (Hui et al., 2018) (Hui et al., 2020) have been used in order to accomplish these networks. (Habibian et al., 2019) suggested a 3D autoencoder method that does not rely on optical flow to compensate for motion. On the other hand, their method is still constrained to detecting fine-scale motions and also suffered time-consuming problem due to suing an auto-regressive probability model. To further reduce redundancy, Lin et al.(Lin et al., 2020) employed numerous frames in various modules utilizing the prior image compression auto-encoder network and pixel-level

network to further minimize redundancy. Even for motion compensation and residual compression they employed the same architecture like (Lu et al., 2019a). Their training technique for many frames was quite difficult, yet they were able to increase the RDO performance over (Lu et al., 2019a). The previous methods (Lu et al., 2019a) (Djelouah et al., 2019) (Habibian et al., 2019) (Lin et al., 2020) include estimating pixel-level optical flow maps and compressing the pixel-level residual information by image compression autoencoder network (too simple architecture). However, producing trustworthy pixel-level flow maps, and motion compression by an autoencoder style network for generating the predicted frame and residual frame may be challenging, which can reduce the compression performance of learning-based video codecs.

Several researchers (Agustsson et al., 2020) (Lu et al., 2020) (Hu et al., 2020) have proposed utilizing deep learning and classical methods in an end-to-end network to alter certain portions for video compression. For example, in (Hu et al., 2020), the RaFC block compresses the flow maps like the traditional block-based methods. Instead of the pixel-wise optical flow map, RaFC-frame determines the correct flow map resolution for each local block of motion features of a video frame in an automated manner. Lu et al. (Lu et al., 2020) focused on improving the reconstruction error of the video compression system by changing autoencoder style structures; however, the features lost from the encoder portion might be necessary for the predicted frame. Next in Agustsson et al. (Agustsson et al., 2020) replace the bilinear warping operation with a scale-space flow algorithm that learns to blur the reference content adaptively for improved warping outcomes. These algorithms (Agustsson et al., 2020) (Lu et al., 2020) (Hu et al., 2020), on the other hand, are tailored to specific prediction modes, resulting in a lack of flexibility throughout the whole compression system and produced compression artifacts.

Afterward, Wu et al. (Wu et al., 2020) proposed another method for video compression us-

ing P-frame compression architecture that can be learned end-to-end. They claimed that their MV-Residual prediction network could simultaneously predict the motion vectors and residual information. However, the popular benchmark datasets (i.e., UVG, HEVC) are not employed in this experiment to show the performance of their proposed model since the Rectified Linear Unit (ReLU) (not deep rectifiers) is used to optimize the rate-distortion trade-off into their MV-Residual prediction network. After that, the most recent network, Hu et al. (Hu et al., 2021), proposed a video coding network where all major operations (i.e., motion estimation, compression, and compensation are performed in the feature space. The deformable compensation modules are used instead of optical flow to obtain accurate motion information. Moreover, the multi-frame feature fusion module utilizes the nonlocal attention mechanism to combine the reconstructed frames. As a result, by using deformable convolution and nonlocal attention mechanisms in every fusion, their proposed video compression model became more complex and showed the computational complexity due to producing more parameters (26M). Furthermore, for motion and residual compression, this model employed the previous architecture (Ballé et al., 2016) without carrying any modification to boost the RDO algorithms. In contrast, our network executes motion compression and compensation module in a novel method using the deep residual attention mechanisms and channel residual block, respectively, to enhance the rate-distortion trade-off for the overall compression system. This results in a higher compression ratio. In addition, to modify the decoded or rebuilt frame artifact, we offer a new module ACM that uses the residual swin convolution UNet block. This module considerably influences the final frame, which helps improve the reconstruction quality.

3. Proposed CNN Model for Image Compression

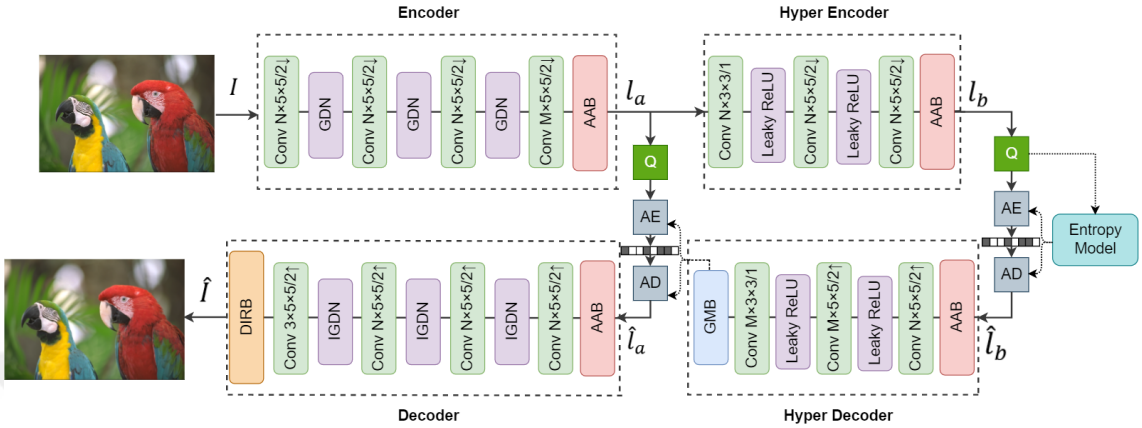

Figure 3.1: The proposed architecture. The AAB, GMB, and DIRB represent the adjacent attention block, Gaussian merge block, and decoded image refinement block. Q, AE, and AD correspondingly indicate the quantization, arithmetic encoder, and arithmetic decoder. The parameters of Conv (convolution) layers indicate the number of filters $\times$ height $\times$ weight of kernel $\times$ stride (up or downsampling). Here, upsampling and downsampling are represented by $\uparrow$ and $\downarrow$, respectively. For the values of N and M, we employ 320 and 192, respectively.

This section presents the proposed deep image compression framework in detail. In Figure 3.1, the architecture is shown. We adopt the network of the autoencoder type, motivated from (Ballé et al., 2016) (Ballé et al., 2018) for learning-based image compression with three new blocks. In particular, four modules are employed in the proposed system, which is the main encoder and decoder, as well as the hyper-encoder and the hyper-decoder network, respectively. The proposed attention mechanism, referred to as the adjacent attention block (AAB), is included in each architecture module. Two additional blocks, the Gaussian merge block (GMB) and decoded image refinement block (DIRB) are introduced to increase the overall performance of the RD and improve the reconstructed image, respectively.

At first, the original image I is taken through the main encoder network and creates the corresponding latent representations l_a by employing four convolutional layers with non-

linear functions (e.g., GDN). After that l_a is quantized to $\hat{l_a}$. The quantized latent forms $\hat{l_a}$ are delivered to the decoder network to generate the final reconstructed image $\hat{I}$ after arithmetic encoding (AE) and decoding (AD). Similarly, we utilize the same quantization method as (Ballé et al., 2018) (Minnen et al., 2018) with some modifications. When it comes to image compression, the goal is to obtain high-quality reconstructed images at a certain bitrate, and the entropy model is utilized to predict the bitrate target. In the suggested approach, we adopt the pipeline described in (Ballé et al., 2018) and (Minnen et al., 2018) and employ the hyper-encoder and hyper-decoder components to determine the parameters of the entropy of the model. The hyper-encoder module received the hyper-prior information from the latent forms $\hat{l_a}$ and encoded them into latent representations l_b. After that, it is quantized to $\hat{l_b}$ and passed to the hyper-decoder after AE and AD process. The hyper-decoder module retrieves the hyper-prior information from $\hat{l_b}$ and estimates the relevant entropy model parameters ψ accordingly. In the following three subsections, we will go through the framework's proposed three blocks, i.e., AAB, GMB, and DIRB.

Below is how the rate-distortion trade-off (Υ) is taken into account while optimizing the learning-based image compression framework:

$$\Upsilon = \lambda \mathrm{D} + \mathrm{R} = \lambda \mathrm{d}(\mathrm{I}, \hat{I}) + \mathrm{H}\left(\hat{l}_a\right) + \mathrm{H}\left(\hat{l}_b\right) \tag{3.1}$$

The D and R are the distortion and bitrate, respectively, in this equation. The trade-off parameter is denoted by λ. The distortion measure (MS-SSIM (Wang et al., 2003)) is denoted by d (.). H is the bitrate utilizing for encoding the latent visualization $\hat{l}_a$ and $\hat{l}_b$, respectively. The suggested technique approximates the bitrate by utilizing the entropy of the respective $\hat{l}_a$ and $\hat{l}_b$ latent interpretations, i.e., $\mathrm{H}\left(\hat{l}_a\right) = \mathrm{E}\left[-\log_2\left(E_{\hat{l}_a|\hat{l}_b}\left(\hat{l}_a \mid \hat{l}_b\right)\right)\right]$ and $\mathrm{H}\left(\hat{l}_b\right) = \mathrm{E}\left[-\log_2\left(E_{\hat{l}_b}\left(\hat{l}_b\right)\right)\right]$ respectively. Both $E_{\hat{l}_a|\hat{l}_b}\left(\hat{l}_a \mid \hat{l}_b\right)$ and $E_{\hat{l}_b}\left(\hat{l}_b\right)$ reflect the probability distributions of $\hat{l}_a$ and $\hat{l}_b$, correspondingly.

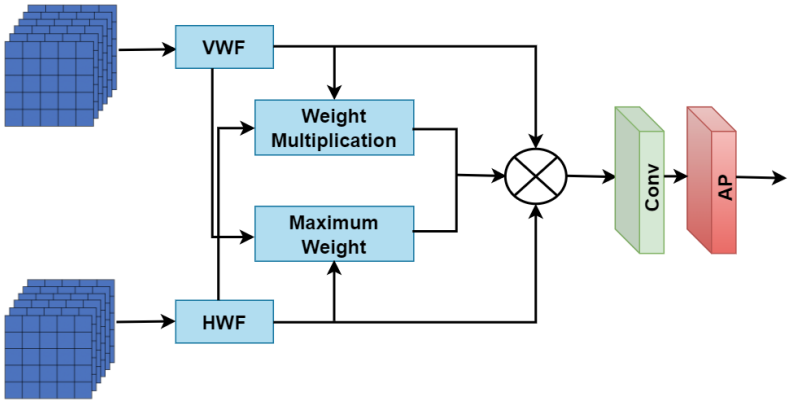

Figure 3.2: The proposed architecture of AAB. VWF and HWF represent two features, namely, vertical weight features and horizontal weight features, respectively. The softmax function generates the weight coefficient, then extended by the weight multiplication block and the maximum weight block. After that, the weight coefficients are passed to the convolution layer (Conv) and average pooling layer (AP).

3.1. Adjacent Attention Block

In deep neural networks, the attention mechanism is an effort to emulate the similar behavior of deliberately focusing on a few significant elements while disregarding the rest. Nowadays, there are now three primary techniques to include attention mechanisms: spatial (Zhu et al., 2019), channel (Hu et al., 2018), and Convolution Block Attention Module (CBAM) (Woo et al., 2018). In the meanwhile, several researchers have adapted spatial attention processes by non-local blocks (Wang et al., 2018) to image compression (Chen et al., 2021) and (Liu et al., 2019), intending to reduce spatial redundancy. However, these methods concentrate only on building deep networks to increase the models' representation capability, which results in high computation and memory demands. Besides, in most cases, the conventional spatial attention mechanism (Li et al., 2021a) only provides one-direction weight allocation (Liu et al., 2019) (Cheng et al., 2020) (Chen et al., 2021), which results in the loss of vital information up to a specific level.

We propose an attention mechanism, AAB, which allocates weights based on distinct

methods from both the vertical and horizontal channels. In addition to successfully suppressing irrelevant information, it may also ensure that the loss of critical information is kept to an absolute minimum. Besides, it concentrates the texture on the edges of the image with much contrast and allocates additional bits to them. Figure 3.2 depicts the proposed structure of AAB. Three parts are included in the structure.

- First, the adjacent attention mechanism may create weight characteristics in both the vertical and horizontal channels. It works crosswise to obtain more stable features for allocating more bits in edge areas.
- Second, it multiplies the two types of weight features, i.e., vertical weight features (VWF) and horizontal weight features (HWF), through the structure's weight multiplication (WM) module to increase the weight coefficient (for example, a tiny weight could be 0.1 x 0.2, while the highest weight could be 0.9 x 0.7).
- Third, it produces the highest possible value by combining the VWF and HWF via the maximum weight (MW) module, which supports the weight coefficient result obtained from the second portion [for instance, max (0.1, 0.9)].

To connect and merge the weights of the three components of the model, they are arranged as follows:

$$w_i = \sum_{l=1}^{n} \frac{a^{a_{i,l}}}{\sum_{q=1}^{n} a^{a_{i,q}}} d_l \tag{3.2}$$

$$wm = concat\left(\left[w_s, w_r, \left(w_s * w_r\right), \left(w_s, w_r\right)\right]\right) \tag{3.3}$$

The weights (w_i) of VWF and HWF allocated by the attention process are denoted by $a^{a_{i,l}}$, pixel i and I denote the feature at a specific instant and the sequential feature, and the hidden layer characteristics of the feature sequence I are indicated by d_l. In equation $2,\ \mathrm{m}$ indicates the weight multiplication, and w_s represents the weight coefficient of VWF

in the feature space $(w_s = [w_1, w_2, \ldots . \ w_{i-1}, w_i])$. Then the weight operation of WM and MW is denoted by $(w_s * w_r)$, and (w_s, w_r), respectively. After completing all the weight operations of VWF and HWF, one convolutional layer (Conv) and average pooling (AP) layer produce the deep feature.

According to Figure 3.1, for high-quality compression, the suggested AAB is incorporated into the encoding, decoding, hyper-encoding, and hyper-encoding networks for leveraging the channel relationship. The re-weighted feature map from AAB is included in the subsequent quantization and entropy coding components.

Figure 3.3: The architecture of GMB. For each layer, N specifies the hyper-parameter that determines how many channels will be available, and C indicates how many different Gaussian models will be available.

3.2. Gaussian Merge Block

Estimating bit rates is critical in learning-based image compression techniques. (Minnen et al., 2018) and (Lee et al., 2018) demonstrate learning-based systems in which the hyper-prior compression technique is employed and a Gaussian Model (GM) distribution is used to represent the latent representations $\left(\hat{l}_a\right)$ in the model.

$$E_{\hat{l}_a|\hat{l}_b}\left(\hat{l}_a \mid \hat{l}_b\right) \sim \partial(\varphi, \vartheta) \tag{3.4}$$

where $E_{\hat{l}_b}(\hat{l}_b)$ denotes the quantized entropy model (Ballé et al., 2016). The purpose of the hyper-encoder and hyper-decoder is to predict the parameters (φ, ϑ) of the GM. Though the one GM-based entropy model has significantly improved over prior work (Ballé et al., 2016), the representation capabilities of one GM are still inadequate, particularly for complicated components. As a result, we conduct the Gaussian Merge Block (GMB) to boost the image compression performance. In our proposed GMB, the $\hat{l}_a$ is expressed as below:

$$E_{\hat{l}_a|\hat{l}_b}\left(\hat{l}_a \mid \hat{l}_b\right) \sim \sum_{i=1}^{G} W_i \partial\left(\varphi_i, \vartheta_i\right) \tag{3.5}$$

where W_i and G denote the weights assigned to various GMs and the number of GMs, respectively. To estimate the parameters (ψ) of the GMB, we generate three convolutional layers with three LeakyReLU layers, as illustrated in Figure 3.3. In our proposed GMB, the value of G is set to three. A total of $6 \times \mathrm{N}$ output channels are employed, with the first $5 \times \mathrm{N}$ channels being used for predicting the mean and variance of three GMs. A sigmoid layer is included in the output of the final N channels in estimating the weights of every GM. For example, the weight of the first GM is denoted by W, and the next one will be (1-W), respectively. Furthermore, by creating $G(G \geq 4)$ GMs, we may increase the number of output channels on the GMB block to $4 \times G \times N$ $(\mathrm{C} = 4 \times \mathrm{G})$. For G of GMs, the mean and variance parameters are estimated by the first $3 \times \mathrm{G} \times \mathrm{N}$ channels in the same manner. The softmax layer is utilized to figure out each GM weight after the final $\mathrm{G} \times \mathrm{N}$ channels.

3.3. Decoded Image Refinement Block

The proposed compression approach for the entropy model employs a quantization procedure. As a result, compression artifacts may appear in the reconstructed image. Thus, a proposed DIRB, at the decoder side, is adjoined after image reconstruction, which signif-

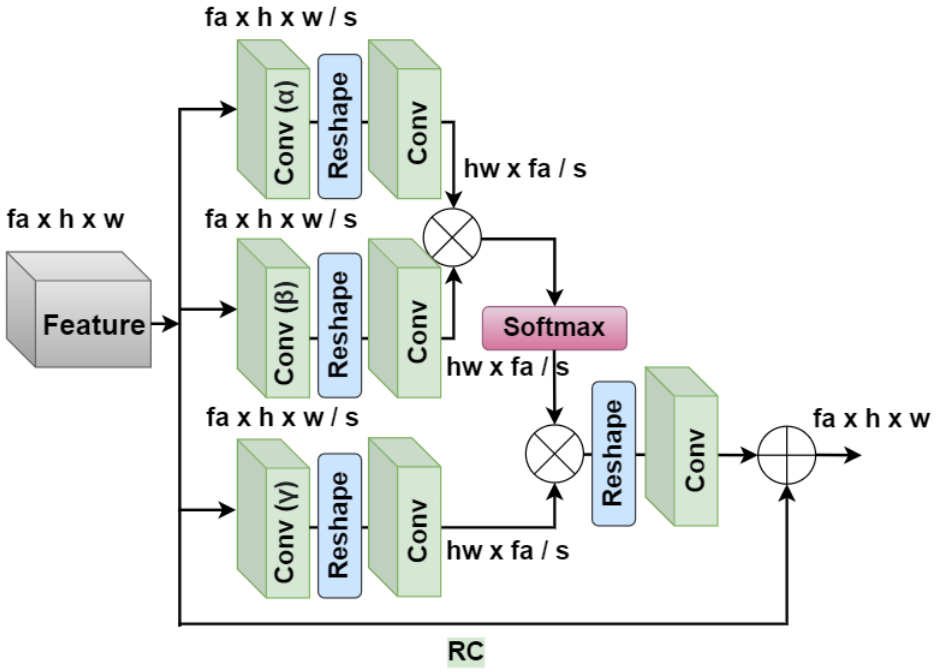

Figure 3.4: The architecture of DIRB. Conv indicates the convolution layer and RC means residual connection. See Section 3.4 for more details.

icantly improves the performance of the decoded image. To improve the representations of feature maps, the proposed refinement block uses a self-similarity measure and inter-spatial relationship information. The following is a concept of a deep neural network process:

$$I_i = \frac{1}{f(O)} \sum_{\omega_j} F\left(O_i, O_j\right) \gamma\left(O_j\right) \tag{3.6}$$

$$f(O) = \sum_{\omega_j} F\left(O_i, O_j\right) \tag{3.7}$$

Where i is the position reference of the feature reaction awaiting to be computed, and j is the counted position reference of input features. The input and output signals are represented by I and O, respectively, with the same area and channel number. At the input feature map, $F(.)$ calculates the similar reaction between j and all j. The response is multiplied by the matching features representation calculated by γ (.) after normalizing

with a coefficient $f(O)$. Refinement block can extract the long-distance dependency between multiple places by calculating the reaction matrix, which may efficiently enlarge the receptive fields of deep convolution layers. It solves the shortcomings of traditional standard convolution operations, which can only gather minimal data from nearby regions. Figure 3.4 depicts our proposed DIRB for obtaining spatial relevant information in a feature space.

$$F\left(O_i, O_j\right) = \alpha\left(X_0\right)^t \beta\left(X_0\right) \tag{3.8}$$

where X_0 represents the input features and $F\left(O_i, O_j\right)$ represents the reaction weight vector for every position. Convolution operations $(\alpha(.)$ and $\beta(.))$ are used to produce the features descriptions, which are multiplied to create the matching matrix.

$$X_{IF} = \text{softmax}\left(\alpha\left(X_0\right)^t \beta\left(X_0\right)\right) \bigotimes \gamma\left(X_0\right) \tag{3.9}$$

softmax (.) and X_{IF} denote the normalized operation and improved features, respectively. Improved features X_{IF} are calculated by multiplying the reaction weight vector for the feature representations, produced by the 1×1 convolution operation $\gamma(.)$.

$$X_{out} = X_0 \bigoplus X_{IF} \tag{3.10}$$

In the refinement block, we included a residual connection that constructs similar to a residual learning network by combining input feature X_0 and improved feature X_{IF}. It enables the component to concentrate on improving high-frequency information rather than low-frequency information.

Comparing the ways of gradually expanding the receptive fields in typical regular con-

volution procedures, our proposed refinement block can acquire the spatial dependency between any two locations for the purpose of further refining and improving the flow of gradients and information. Our DIRB can also add global information to the features that allow our network to utilize better the promising information contained within the low-resolution reconstructed images.

3.4. Proposed Transformer Model For Image Compression

Fig. 3.5 depicts the proposed transformer-based end-to-end architecture for image compression (TICM). It includes patch partitioning, patch embedding, the main transformer encoder-decoder (denoted by M_E and M_D) and hyperprior autoencoders (represented by H_E and H_D). Each M_E and H_E consist of transformer block and deformable patch merging module (DPMM), on the other hand, every M_D and H_D compose of a transformer block and deformable patch expanding module (DPEM). The main encoder-decoder of transformer and hyperprior autoencoders are connected through a CGAM in order to exploit spatial relationships of latent features. The following subsections provide a more in-depth description of the main diagram.

3.4.1. Encoder-Decoder

In this work, images are compressed using both main and hyper encoders to assess and capture the most relevant information. In contrast, decoders perform operations similar to the encoders in order to create a pixel domain reconstructed image from the compressed bitstream.

Firstly, before encoding, the proposed model uses a patch splitting module, like ViT, to divide input RGB images (I $\in \mathbb{R}^{H\times W\times 3}$ where H and W represent height and width, respectively) into non-overlapping patch partitions. Each patch is regarded as a "token," with its feature specified as a concatenation of the RGB image raw pixel values. A linear

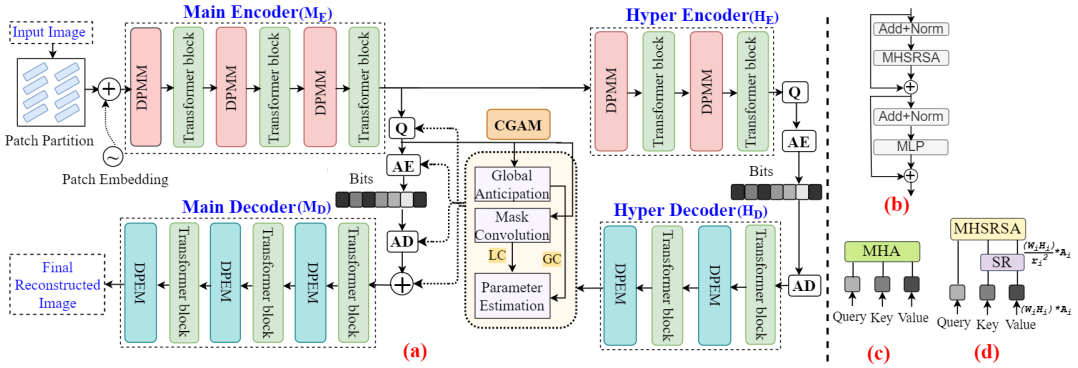

Figure 3.5: Proposed image compression model (TICM). The model (a) consists of transformer-based main (M_E and M_D) and hyper encoder-decoders (H_E and H_D). Each encoder contains a deformable patch merging module (DPMM) and transformer block, on the other hand, every decoder contains a deformable patch expanding module (DPEM) and transformer block. The employed transformer block is displayed in (b). The (c) and (d) represent the traditional MHA layer and our proposed MHSRSA layer. The modified CGAM is devised to connect the main and hyper-encoder-decoders, which can boost the RDO algorithm. The CGAM combines mask convolution, global prediction, local context (LC), and global context (GC) to make long-range correlations in the latent space.

embedding layer is employed to project this raw-valued feature to an arbitrary dimension (represented as A). Following that, each patch is projected into dimension $A1$ using a linear embedding layer, which serves as the first input for the pipeline that follows. In our approach, we utilized the patch size of 4×4, which resulted in a feature dimension of 4×4×3 = 48 for each patch and obtained feature map ($P1$) of size $\frac{H}{4} \times \frac{H}{4} \times$ A1. These embedded patches with a position embedding are fed into a transformer block with DPMM (Pan et al., 2021), and the output is molded into a $\frac{H}{8} \times \frac{H}{8} \times$ 2A1 size feature map $P2$ which is exclusively used to encode local representations. The DPMM is conducted to create a hierarchical representation, which reduces the number of tokens and non-uniformly merges significant patches in an adaptive way as the network becomes deeper. The DPMM equation can be expressed in the following way:

$$DPMM(I_{fp}) = GELU(BN(DC(I_{fp}))) \tag{3.11}$$

where BN indicates batch normalization, GELU is the non-linear activation function, and I_{fp} is the input feature map. The remaining two transformer blocks with DPMM to manage longer correlation dependencies. Each pair of 2 $\times$ 2 adjacent patches are concatenated by the initial DPMM, and a linear layer is applied to the concatenation of the $4A$ dimensional features. The number of tokens is reduced by a multiple of 2$\times$2 = 4 (2$\times$ downsampling of resolution) and the output dimension is defined to 2A. Similarly, using the extracted features maps from the previous step as input, $P3$ with the output size of $\frac{H}{16} \times \frac{H}{16}$ is obtained.

Table 3.1: Detail description of the encoder part where H, W, DPMM, TB, S, HD, ML, P, N denote height, width, deformable patch merging module, transformer block, reduction ratio of the MHSRSA, head number of the MHSRSA, MLP blocks, patches, number of channels, respectively.

Block	Output Size	Layers	Configuration
1	$\frac{H}{4} \times \frac{W}{4}$	DPMM	S1=8, HD1=1, ML1=3, P1=4, N1=64
		TB	
2	$\frac{H}{8} \times \frac{W}{8}$	DPMM	S2=4, HD2=2, ML2=3, P2=2, N2=128
		TB	
3	$\frac{H}{16} \times \frac{W}{16}$	DPMM	S3=2, HD3=6, ML3=4, P3=2, N3=320
		TB	
4	$\frac{H}{32} \times \frac{W}{32}$	DPMM	S4=1, HD4=6, ML4=4, P4=2, N4=512
		TB	
5	$\frac{H}{64} \times \frac{W}{64}$	DPMM	S5=1, HD5=6, ML5=8, P5=2, N5=704
		TB	

The output feature maps from the M_E are fed into hyperprior modules. The hyperprior module is conducted to minimize the bit rate, calculate the hidden representations' probability distribution, and impose entropy restrictions. Employing the identical methods as in M_E, the feature map for H_E, we received $P4$ and $P5$ with the output sizes of $\frac{H}{32} \times \frac{H}{32}$, and $\frac{H}{64} \times \frac{H}{64}$, respectively. In Table 3.1, the transformer encoder part and its patches and dimensions are described in detail.

The symmetric decoder, which corresponds to the encoder, is being developed. Instead of DPMM, we employed DPEM in both M_D and H_D to up-sample the extracted deep features. To decrease the feature dimension, the DPEM reconfigures the feature maps of

neighboring dimensions into a higher resolution feature map (2 $\times$ upsampling of resolution) and minimizes the feature dimension to half of the original dimension. From H_D, the output sizes of $\frac{H}{64} \times \frac{H}{64}$ and $\frac{H}{32} \times \frac{H}{32}$, are acquired respectively. Next, these feature maps are passed into M_D after applying the CGAM. See Section 2.3 for a detailed description of the CGAM. Eventually, the final reconstructed image is acquired from M_D, with its feature map $\frac{H}{16} \times \frac{H}{16}$, $\frac{H}{8} \times \frac{H}{8}$, and $\frac{H}{4} \times \frac{H}{4}$, respectively.

3.4.2. Transformer block

In this article, the ViT transformer block is employed for both main and hyperprior autoencoders with some modifications. Each block contains two normalization (Add+Norm) layers, our proposed multi-head spatial reduction self-attention (MHSRSA) layer by replacing the multi-head attention (MHA) layer of ViT, residual skip connection, and a MLP layer with GELU non-linearity. This can be formulated as:

$$I_l = I_{l-1} + M_{lp}\left(\mathrm{Nm}\left(I'_{l-1}\right)\right) \tag{3.12}$$

$$I'_{l-1} = I'_{l-1} + \mathrm{MHSRSA}\left(\mathrm{Nm}\left(I_{l-1}\right)\right) \tag{3.13}$$

Where M_{lp}, Nm, MHSRSA indicates MLP layer, normalization layer, and MHSRSA layer, respectively.

Our proposed MHSRSA takes a query q, a key k, and a value v as input and returns a refined feature, similar to MHA (see Fig.3.5 (c)). The distinction is that our SRA minimizes the spatial scale of k and v before performing the attention operation (see Fig. 3.5 (d)), reducing the computational/memory cost significantly. The equation is formulated

as follows:

$$\mathrm{MHSRSA}(q, k, v) = \textbf{Concat}(h_0, \ldots, h_{N_i})w^o \tag{3.14}$$

$$\mathrm{h}_j = \textbf{Attention}(qw_j^q, \mathrm{SD}(k)w_j^k, \mathrm{SD}(v)w_j^v) \tag{3.15}$$

where the concatenation operation (Vaswani et al., 2017) is represented as Concat $(\cdot)$. $w_j^q \in \mathbb{R}^{A_i \times d_\mathrm{h}}$, $w_j^k \in \mathbb{R}^{A_i \times d_\mathrm{h}}$, $w_j^v \in \mathbb{R}^{A_i \times d_\mathrm{h}}$, and $w^o \in \mathbb{R}^{A_i \times A_i}$ indicate linear projection parameters. The head number of the attention layer in level i is represented by N_i. Consequently, the dimension of each head (i.e., d_h) is the same as $\frac{A_i}{N_i}$. The spatial dimension of the input sequence (i.e., k or v) is reduced using the SD operation, which is expressed as:

$$\mathrm{SD}(i_n) = \mathrm{Norm}\left(\mathrm{Reshape}\left(i_n, r_i\right) w^s\right) \tag{3.16}$$

Here, a input sequence is represented by $\mathbf{i_n} \in \mathbb{R}^{(h_i w_i) \times A_i}$, and the lowering ratio of the attention layers is calculated by r_i in level i. The input sequence i_n is reshaped into a new sequence of size $\frac{h_i w_i}{r_i^2} \times (r_i^2 A_i)$ using the function Reshape(i_n, r_i). The dimension of the input sequence is minimized to A_i by a linear projection, $w_s \in \mathbb{R}^{\left(r_i^2 A_i\right) \times A_i}$. Normalization layer is referred to as Norm. Our attention operation Attention is computed in the same way as the original Transformer (Vaswani et al., 2017):

$$\mathit{Attention}(\mathbf{q}, \mathbf{k}, \mathbf{v}) = \mathrm{Softmax}\left(\frac{\mathbf{q}\mathbf{k}^\top}{\sqrt{d_\mathrm{h}}}\right)\mathbf{v}. \tag{3.17}$$

These calculations reveal that our attention operation's computational and memory costs are r_i^2 times lower than traditional MHA, allowing our MHSRSA to manage greater input feature maps and sequences while using less computing and memory.

3.4.3. CGAM block

The CGAM block performs as an adaptor between the main and hyperprior autoencoders to capture spatial dependencies in the latent space. Additionally, it helps to reduce the amount of computation cost required in the model. Previous works (Ballé et al., 2018) (Minnen et al., 2018) (Chen et al., 2021) (Lu et al., 2021) employed adaptive context modeling to create an entropy model for end-to-end RDO algorithms. However, these models do not capture latent spatial correlations at a global scale, and cross-channel latent interactions are not investigated. As a result, latent-separated global prediction causes a decoding consistency issue, which raises decoding complexity. Due to this problem, we used a modified version of the Casual Context Module (Guo et al., 2020) in our study, which we called CGAM.

In the previous work (Guo et al., 2020), throughout channels, the causal context model divides the latent $\hat{l}_n$ in spatial location c into two groups. Decode $\hat{l}_n$ is predicted as typical by using the distribution of the first channel group $\hat{l}_{n,1}$. This channel group uses a mask convolution $m_{\text{conv},1}$, to create adjacent context $a_{n,1}$ and then integrates it with hyperpriors $\hat{z}$. After that the second group is predicted to $\hat{l}_{n,2}$, improved the mask to $m_{\text{conv},2}$, and adjacent contexts is $a_{n,2}$. Generally, the whole procedure may be stated as follows:

$$
\begin{aligned}
a_{n,1} &= m_{conv,1}\left(\hat{l_{i_1}}, \hat{l_{i_2}} \ldots \hat{l_{i_k}}\right), \\
Q_{\hat{l}_{n,1|\hat{z}}}\left(\hat{l}_{n,1|\hat{z}}\right) &\leftarrow E_{s,1}\left(\hat{z}, a_{n,1} \mid \theta_E\right), \\
a_{n,2} &= m_{conv,2}\left(\hat{l_{i_1}}, \hat{l_{i_2}}, \ldots \hat{l_{i_k}}, \hat{l}_{n,1}\right), \\
Q_{\hat{l}_{n,2|}|\hat{z}}\left(\hat{l}_{n,2|\hat{z}}\right) &\leftarrow E_{s,2}\left(\hat{z}, a_{n,2} \mid \theta_E\right)
\end{aligned}
$$

This causal context model extracts the cross-channel interaction well. However, it simply extracts local relationships and does not consider global relationships. For this reason, a

modification is carried out here. We present a causal global anticipation model (CGAM) that uses the long-range interactions of the latent in order to make anticipation.

In our CGAM, the latent variables $\hat{l}$ are also divided into two groups. The compression procedure for the first channel group $\hat{l}_{n,1}$ remains identical, with the exception of the use of hyperprior and the local context model, as seen in Fig.3.5 (a) CGAM portion. While predicting the distribution of the second latent group, both the optimized adjacent context $a_{n,2}$ and the global context $a_{n,3}$ is integrated with the entropy model together. The optimized adjacent context $a_{n,2}$ is created by the causal context model $m_{\text{conv},2}$, and the global context $a_{n,3}$ is derived by the causal global anticipated model $m_{\text{conv},3}$. Therefore, with the previous process, the new steps can be adjusted as follows:

$$a_{n,3} = m_{conv,3}\left(\hat{l_1}, \hat{l_2}, \ldots\ldots\ldots\ldots \hat{l}_{n-1}, \hat{l}_{n,1}\right),$$
$$Q_{\hat{l}_{n,2|\hat{z}}}\left(\hat{l}_{n,2|\hat{z}}\right) \leftarrow E_{s,2}\left(\hat{z}, a_{n,2}, a_{n,3} \mid \theta_E\right).$$

Due to the CGAM, the decoder is able to construct reliable global reference information through latent separation, to reduce the decoding time complexity, as shown in the ablation study.

3.4.4. Loss function

After transforming the input x into latent features y, the encoder quantizes the latent features y to $\hat{y}$, which are subsequently entropy-coded into bitstreams using predetermined distribution models. The decoding section repeats the encoding processes by analyzing the encoded bitstream and reconstructing pixel blocks to generate decoded $\hat{x}$. Therefore, the rate-distortion cost equation should be:

$$L = R(\hat{y}) + \lambda D(x, \hat{x}) \tag{3.18}$$

where R is the compressed bit rate of $\hat{y}$, and D denotes the mean square error (MSE) between the ground truth x and the restored output $\hat{x}$. λ is modified to achieve a rate-distortion trade-off at different bit rates.

Figure 3.6: Qualitative Visualization. At low bitrates, visual assessment on (a) KODIM 15 and (b) KODIM 07 from the KODAK dataset as well as (c) and (d) represent the final output of the reconstructed images.

4. Proposed Model for Optical Flow

The complete architecture of DeepPyNet is shown in Figure 4.1. It consists of three main steps: firstly, a deep feature pyramid extraction method retrieves features from the feature map of the input images. Then we utilize a multi-channel cost volume for all possible pairings of pixels in the last two levels of the feature map. Finally, an iterative residual refinement method is used to update a flow field iteratively.

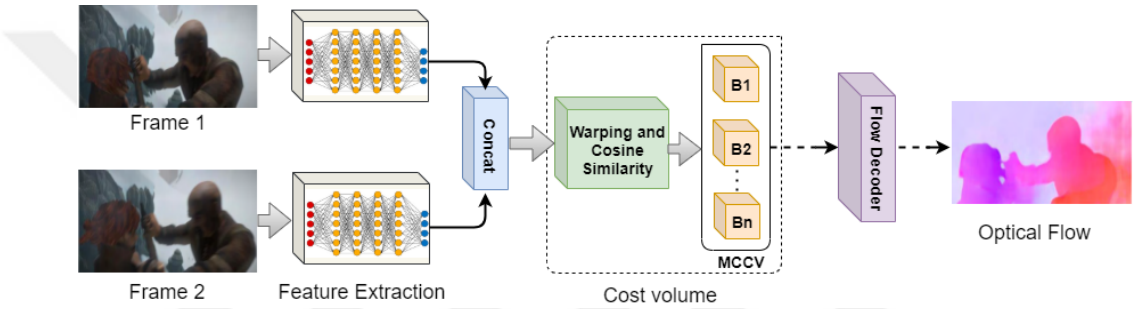

Figure 4.1: There are three main steps of DeepPyNet; first, a feature extraction method based on the recurrent feature pyramid networks is used. The feature extractor method extracts every frame and then concatenates and passes into the multi-channel cost volume (MCCV) of the last two levels of the feature map. Here, B1, B2, and Bn represent the multi-channel cost volumes. Finally, an iterative residual refinement technique is used to update a flow field iteratively through a flow decoder network.

4.1. Feature Extractor

The proposed feature extractor method is depicted in Figure 4.2. DeepPyNet includes a four-stage feature pyramid, with each level consisting of two residual blocks with width and height resolutions that are $1/2, 1/4, 1/8$, and $1/16$ of the original image resolution, respectively. For each pair of images, features are extracted top to bottom through four routes, the left and right pyramids using the WFD, while the middle pyramid is constructed using the WFD with a max-pooling layer. The left pyramid is referred to as the foundation pyramid. Modifying the actual pyramid design when integrating DeepPyNet into various optical flow estimation techniques in different ways is necessary.

Furthermore, the foundation pyramid sends the features to the middle pyramid to repair information because of the max-pooling. In turn, the middle pyramid updates the features to rebuild the right pyramid. Note that we did not add the MAF and ABF functions at every feature extraction level. Moreover, we have discovered that adding these functions is adequate to achieve high performance only at certain levels near the base of the pyramid.

4.2. Feature Pyramid Map

In this part, we discuss the thorough feature map analysis of our proposed DeepPyNet to reduce the blurry flow fields and loss in every pyramid level. At first, an M-level bottom-up route is generated by a feature pyramid from a pair of RGB images, B_t and B_{t+1}, where the bottom level corresponds to the initial image, which is denoted, $O_t^0 = B_t$. Each feature map O_t^l is usually produced using a $3x3$ convolutional filter, including a stride of two pixels with regard to the previous feature map O_t^{l-1}. Recent works (Dosovitskiy et al., 2015) have used the last four feature map pairings $\left\{\left(O_t^3, O_{t+1}^3\right), \left(O_t^4, O_{t+1}^4\right), \left(O_t^5, O_{t+1}^5\right),\right.$ and $\left.\left(O_t^6, O_{t+1}^6\right)\right\}$ to calculate the cost volume, resulting in an architecture with six layers of convolutional blocks. Thus, the cost volume can be expressed at level l^{th}, as follows:

$$U = \mathrm{s}\left(O_t^l, v\left(O_{t+1}^l\right)\right) \tag{4.1}$$

The warping procedure v employs bi-linear interpolation to warp B_{t+1}, including the optical flow to B_t. An element-wise dot product calculates the similarity between the two feature maps through the s (the correlation operation).

In the feature pyramid, at each level I, the output pixel value $n_{i,j}^l$ is a weighted sum of the

pixel values from the preceding level

$$n_{i,j}^{l} = \sum_{a,b} n_{i+a,j+j}^{l-1} p_3(a,b) \tag{4.2}$$

Where a, b are the pixel indices, p_3 is the 3×3 convolutional kernel, and $-1 \leq a, b \leq 1$ (3×3 kernel). It can be referred as weighted feature downsampling (WFD) pyramid (Sun et al., 2018) (Hui and Loy, 2020) (Hur and Roth, 2019). We found that blurred motion limits are a significant consequence of WFD. For example, in the original images, two little things move in various ways. B_t and B_{t+1} following WFD in multiple levels of convolution. The ground resolution considers both items as a whole and predicts both objects in the same direction, which is wrong and opposes potentially accurate higher resolution estimations. As a consequence, the overall forecast will be wrong.

To remove this error, we utilized the max-pooling layer with the WFD pyramid in the feature extraction method of DeepPyNet. As a result, they can learn to enhance edge regions since the max-pooling layer is anticipated to retain heavier weight pixels for the following level, resulting in improved edge areas. Although the max-pooling layer may not contain all important information, it may lead the WFD pyramid to the additional information via the residual structure. Therefore, the WFD with the max-pooling layer work like another pyramid, extracting the features via an addition operation. The value of the pixel at Level M

$$n_{i,j}^{l} = \sum_{a,b} M(n)_{i+a,j+j}^{l-1} p_1(a,b) + n_{i+a,j+j}^{l-1} p_3(a,b) \tag{4.3}$$

After downsampling to restore feature maps, we have used two functions: a multiplicative attention function (MAF) that outputs a shape attention mask, and an additive bias function (ABF), which produces a bias mask. If there are two pyramids, then the feature map

of the right pyramid at level $1 + 1$ needs to consider the feature map of the left pyramid, as shown in

$$O_{\text{right}}^{l+1} = \text{Conv}\left(O_{\text{right}}^{l} MAF(O)_{\text{left}}^{l+1} + ABF(O)_{\text{left}}^{l+1}\right) \tag{4.4}$$

We also offer additional routes with the above two functions (MAF and ABF) for the features, which operate at a certain level of the various pyramids as determined.

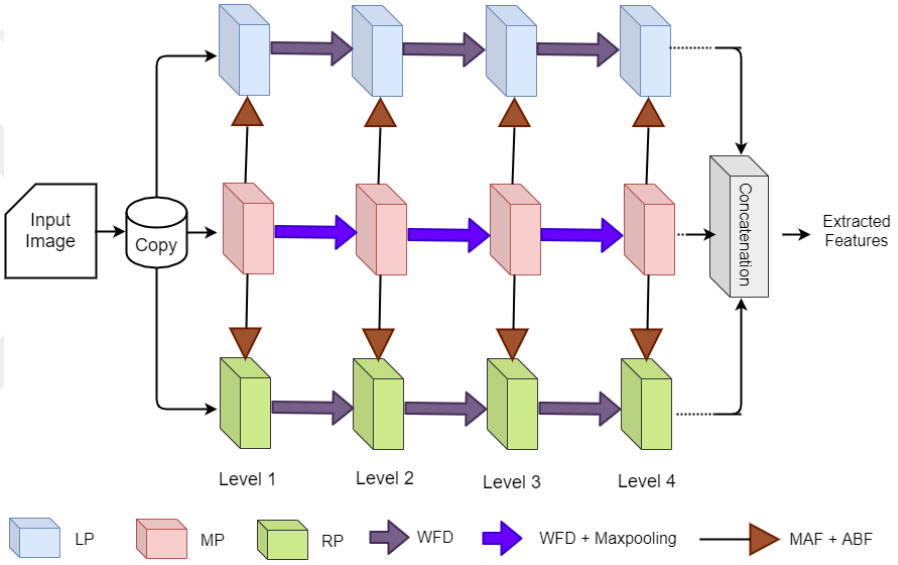

Figure 4.2: The proposed feature extractor method is based on a deep pyramid network. Here, LP = Left Pyramid, MP = Middle Pyramid, RP = Right Pyramid, WFD = Weighted Feature Downsampling, MAF = Multiplicative Attention Function, and ABF = Additive Bias Function.

4.3. Multi Channel Cost Volume

Cost volume may usually incorporate the difference between the two feature extractors, yielding an extra dimension channel $|\mathrm{G(a)}|$. In our instance, the volume would be unreasonably enormous. For this reason, we utilized a multichannel cost volume (Yang and Ramanan, 2019), which is like a "moderate" strategy between a conventional and a modern (deep) feature volume. It took all of the pairings of pixels in the feature map of the

last two levels of DeepPyNet.

Let $G^1, G^2 \in K^{\mathrm{de} \times Hi \times Wi}$ denote the d-dimensional pixel-wise embeddings of the source and destination images. By calculating the cosine similarity of every pixel in the images in Hi to Wi source, we constructed a $4\mathrm{D}$ cost volume with many potential targets in a search window of S and T.

$$B(b,a) = \frac{G^1(a) \cdot G^2(a+b)}{\|G^1(a)\| \cdot \|G^2(a+b)\|}, B(b,a) \in K^{S \times T \times Hi \times Wi} \tag{4.5}$$

where $\mathrm{a} = (\mathrm{e}, \mathrm{f})$ represents the source pixel position and $\mathrm{b} = (\mathrm{g}, \mathrm{h})$ represents the pixel displacement. People re-identification and appearance authentication are both accomplished via cosine similarity as a substitute for the dot product(Liu et al., 2017). By extracting channel-wise products between every pair of possible matches, we captured n similarities between n distinct feature embeddings that have been trained together. While this may be thought of as n separate cost volumes, we appended them into a multi-channel 4D cost volume where n is regarded as a featured channel whose dimension is maintained constant throughout the filtering process. In this multi-channel cost volume process, we used 9×9 search windows on a feature pyramid with the strides: $64; 32; 16; 8; 4$. At each scale, we maintained $n = 16; 16; 16; 16; 12$ hypotheses.

4.4. Flow Decoder Network

We have used the Iterative Residual Refinement (IRR) (Hur and Roth, 2019) method in which data are retrieved from the multiscale $4\mathrm{D}$ cost volumes, and an optical flow field is repeatedly updated using a similar network block with separated weights. The four-level DeepPyNet is gradually updated throughout the pyramid levels, using separate decoders for every level. Using the IRR method, we replaced numerous decoders with just one typical decoder that optimizes the output incrementally across all the pyramid layers. In

the initial pipeline, we specified the number of the iterations double to the number of pyramid levels, with fewer parameters and an increasingly concise illustration:

$$F_{fl}^{l} = \mathrm{De}\left(O^{l}\left(G_1\right), \mathrm{co}\left(O^{l}\left(G_1\right), w\left(O^{l}\left(G_2\right), \hat{F}_{fl}^{l-1}\right)\right), \hat{F}_{fl}^{l-1}\right) + \hat{F}_{fl}^{l-1} \tag{4.6}$$

$$\hat{F}_{fl}^{l-1} = 2. \uparrow \left(F_{fl}^{l-1}\right) \tag{4.7}$$

Where, F_{fl}^{i}, G_1, and G_2 represent the optical flow, reference frame, and target frame respectively. The feature map O^{l} at pyramid level I, De is the decoder, co represents the cost volume, and $\uparrow$ conducts 2 times bilinear upsampling to produce a flow field with double the resolution of the preceding one, respectively. As the dimension grows, we likewise increased the volume of the flow in proportion equation (4.7).

A 2×2 convolution layer is added after the input feature map $O^{l}\left(G_1\right)$ in order to ensure that the number of feature maps supplied to the decoder De is the same throughout all pyramid levels. This allows us to utilize a single common decoder with a set number of input streams throughout the pyramid, rather than many decoders, and update the flow field as well. The flow of the output is $1/16$ of the input image resolution. The estimated flow fields are upsampled to match the ground truth resolution throughout training and evaluation.

4.5. Loss Function

In order to effectively control our network, we used the I1 distance between the predicted and ground truth flow across the whole series of predictions, namely, $\{F_1 \ldots\ldots\ldots F_n\}$ with increasingly high weights over the entire scene. The loss (f) is formulated as having

for a ground truth flow F_{gt} :

$$\mathrm{E} = \sum_{i=1}^{n} \delta^{n=i} \left\| F_{gt} - F_i \right\| \tag{4.8}$$

Where the $\delta = 0.10$ is set in our experiments.

5. PROPOSED MODEL FOR VIDEO COMPRESSION

Fig. 5.1 depicts the proposed optimized video compression architecture, which includes motion estimation (ME), MV compression (MVC), motion compensation (MC), and residual compression (RC) network. However, the MVC and MC modules (Lu et al., 2019a) are significantly improved through the motion vector compression network with the deep residual attention split (MVCDRAS) block and motion compensation with the channel residual block (MCCRB) to optimize the efficiency of the video compression system. Furthermore, we proposed the artifact contraction module (ACM) that makes the reconstruction frame more pleasing. These components are tuned together and utilize a single rate-distortion loss, as shown via experimental findings and extensive ablation studies. These modules are separately discussed below.

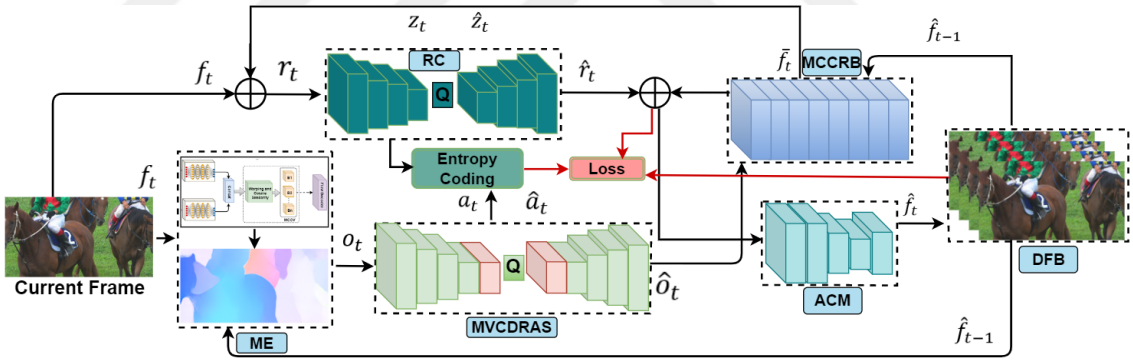

Figure 5.1: An overview of the proposed video compression architecture. To generate the raw optical flow, o_t, we first send f_t along with $\hat{f}_{t-1}$ to the motion estimation (ME) module. Then it is sent to the motion vector compression network with deep residual attention split (MVCDRAS) to compress o_t and receive $\hat{o}_t$. Next, $\hat{o}_t$ is passed through the proposed motion compensation with the channel residual block (MCCRB) network to obtain $\bar{f}_t$. After that, f_t and $\bar{f}_t$ are concatenated and obtained as r_t. Additionally, r_t is fed into the residual compression (RC) network, which obtains $\hat{r}_t$. Due to artifact from the quantization process, the concatenated frame, $C(\hat{f}_t, \hat{r}_t)$, is fed into the proposed artifact contraction module (ACM) to get $\hat{f}_t$. A decoded frame buffer (DFB) is employed as an online buffer to create a clear reconstructed frame.

Notations. Let $\mathrm{F} = \{f_1, f_2, \ldots f_{t-1}, f_t\}$, these symbols represent the original video se-

quences. When the time step t occurs, f_t, $\bar{f}_t$, and $\hat{f}_t$ represents the original, predicted, and reconstructed/decoded frame, respectively. The difference between the predicted frame $\bar{f}_t$ and the original frame f_t is represented by r_t (the residual frame). The final reconstructed/decoded residual frame is denoted by $\hat{r}_t$. This work uses the reconstructed and decoded words in the same context. Motion information obtained from pixel-wise optical flow estimation is essential for eliminating temporal redundancy. Therefore, motion vector/information (MV) or optical flow value is represented by o_t. The final reconstructed/decoded values of the MV fields at time step t are represented by $\hat{o}_t$. Since an autoencoder illustrates transformation, consequently, the residual frame r_t and the MV frame o_t are transferred to z_t and a_t. After the quantization in MV and residual autoencoder, $\hat{z}_t$ and $\hat{a}_t$ denotes the quantized forms of z_t and a_t, respectively.

5.1. Motion Estimation

Video compression performance significantly depends on motion estimation. Optical flow finds the temporal correlation between video sequences. Several recent learning-based optical flow estimation algorithms (Dosovitskiy et al., 2015) (Ranjan and Black, 2017) (Sun et al., 2018) (Hui et al., 2018) (Hui et al., 2020). These methods can compute the pixel-level motion information precisely. However, more bits are needed to compress motion information for conventional video compression methods because of the higher data capacity. The recent deep recurrent feature pyramid-based network, namely DeepPyNet (Jeny et al., 2021) is used to find optical flow. It includes a pyramid-based feature extractor (not handcrafted), multi-channel cost volume, and flow decoder. In this method, the previous reference frame $\hat{f}_{t-1}$ and current frame f_t are taken as inputs and produce the motion information o_t, which is compressed later through the autoencoder network. This motion estimation network is correctly optimized to reduce the rate-distortion trade-off. Adapting the DeepPyNet (Jeny et al., 2021) optical flow map with our video compression

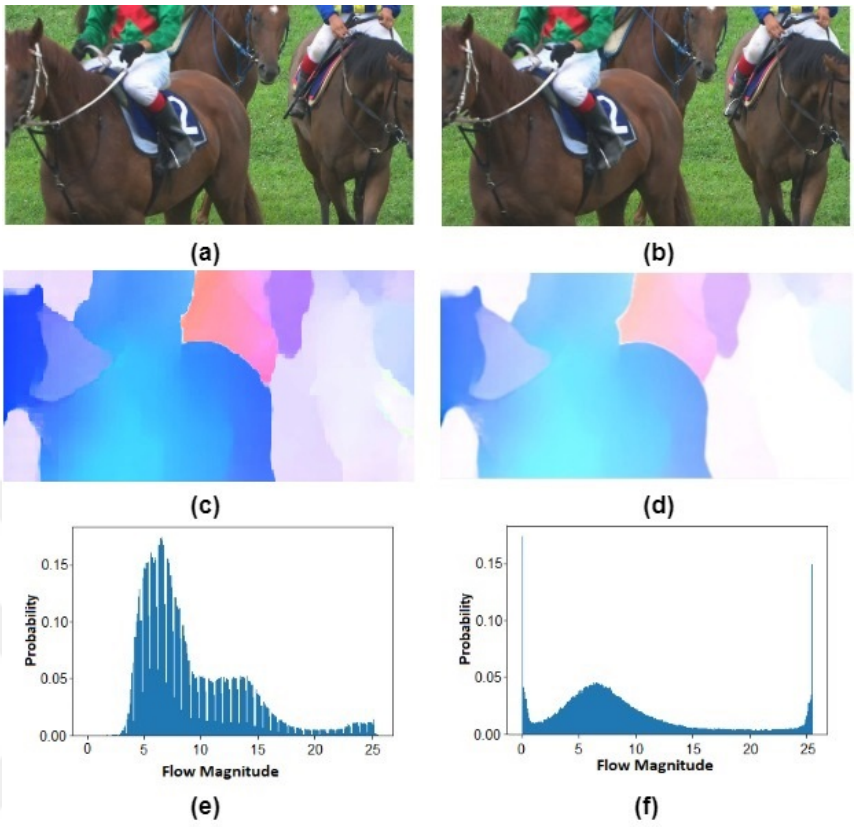

Figure 5.2: Optical flow with statistical analysis. (a) and (b) represent two consecutive frames from the HEVC Class C dataset, (c) and (e) represent the raw optical flow and its flow magnitude, (d) and (f) optical flow in the video compression system and its flow magnitude.

approach makes it much more efficient.

Fig. 5.2 shows the optical flow estimation maps of two RaceHorse video sequences (sequence 82 in (a) and 83 in (b)) from the HEVC Class C dataset (Sullivan et al., 2012) with and without joint training. The associated probability distributions of the optical flow magnitudes are shown in (e) and (f), respectively. The reconstructed optical flow map (d) obtained by cooperative training of the video compression model includes more pixels with zero value than the original optical flow map (c) counterpart. Thus, the reconstructed optical flow map needs fewer bits for encoding. For example, encoding the optical flow map in Fig. 2(c) requires 0.043 Bpp, whereas encoding the optical flow map in Fig. 2(d)

requires just 0.031 Bpp, saving around 27 percent of the bits.

5.2. MV Compression With Deep Residual Attention Split

The key contributor to improving RDO performance is optical flow compression. To compress it, the earlier works (Lin et al., 2020) (Wu et al., 2020) (Feng et al., 2020) (Yang et al., 2020b)(Li et al., 2021b) (Sheng et al., 2021) employed the variational autoencoder network (Ballé et al., 2018) for compressing the motion vector. It takes significant time to capture abstract features for the decoder. Although residual blocks are utilized by (Hu et al., 2021) (Feng et al., 2020) to acquire more deep features, this was insufficient to lessen the computational burden. However, attention mechanisms with residual blocks (Wang et al., 2017) may perform better since an attention mechanism is used to assign greater attention to certain image regions to create more relevant features for the decoder and reduce the computational load. Besides, a model with computable linear-nonlinear transformations shows superior efficiency in end-to-end optimization architecture.

Inspired by this, we propose an autoencoder-style network with deep residual attention and split (DRAS) block to compress the optical flow, as shown in Fig. **??** (a). It consists of four DRAS blocks (2 each encoder and decoder), four convolution and deconvolution layers, followed by the Generalized Divisive Normalization (GDN) or inverse GDN, except for the last layer. The convolution and deconvolution layer sizes are considered by the three parameters, $\mathrm{k} \times \mathrm{n} \times \mathrm{s}$ that represent the size of the kernel, filter maps, and the stride, respectively. At first, the motion representation a_t is generated by the MV encoder, which is then quantized via uniform scalar quantization to $\hat{a}_t$. After that $\hat{a}_t$ is sent to the MV decoder, which rebuilds at $\hat{o}_t$. The decoder's architecture is just the reverse of the encoder's.

DRAS block. A novel part of our autoencoder network is the DRAS block, as shown in

Fig. **??** (b). Due to the high computational cost, typical non-local attention (Liu et al., 2019) is ineffective for this task. Our proposed DRAS block utilizes both split attention (SA) (Zhang et al., 2020) and residual GDN (ResGDN) blocks (in Fig. **??** (c)) to achieve greater efficiency. Here, the SA can capture cross-channel relationships in the features for better representations of the optical flow-based motion vector. On the other hand, the paradigm of identity shortcut connection in the ResNet (He et al., 2016) is added to certain GDN layers, represented by the ResGDN, to enable deeper learning of the network. At first, the input features go through convolution and ResGDN layers and are fed into the SA layer for splitting the features. After that, these features are concatenated and passed to inner attention (IA) layer (inside the red dotted line in Fig. **??** (b)). This IA layer consists of a Max-Pooling layer, two convolution layers with a 1×1 kernel, two types of rectifiers (i.e., GDN and Sigmoid), and one SA block. The output of SA is concatenated with the element-wise production features of the previous SA. Before concatenating the last single 1×1 convolution layer and input residual connection, the last features from SA layers (the initial SA and SA from IA) are concatenated with some residual skip connections. These residual connections with the IA layer increase the channel dimensions and allow for the prediction of accurate motion vectors, which can minimize the bit rate and distortion (please refer to section V (ablation study)).

5.3. Motion Compensation With Channel Residual Block

The predicted frame $\bar{f}_t$ is obtained by the motion compensation (MC) network, which is supposed to be similar to f_t. In the MC module, $\hat{f}_{t-1}$ is warped to f_t depending on $\hat{o}_t$ as Eq. 5.1.

$$\hat{f}^w_{t-1} = \text{Warp}\left(\hat{f}_{t-1}, \hat{o}_t\right) \tag{5.1}$$

where, $\hat{f}^w_{t-1}$ represents the warped frame and $Warp$ is the backward warp operation (Jaderberg et al., 2015). However, there are still artifacts in $\hat{f}^w_{t-1}$. Therefore, $\hat{f}^w_{t-1}$ and

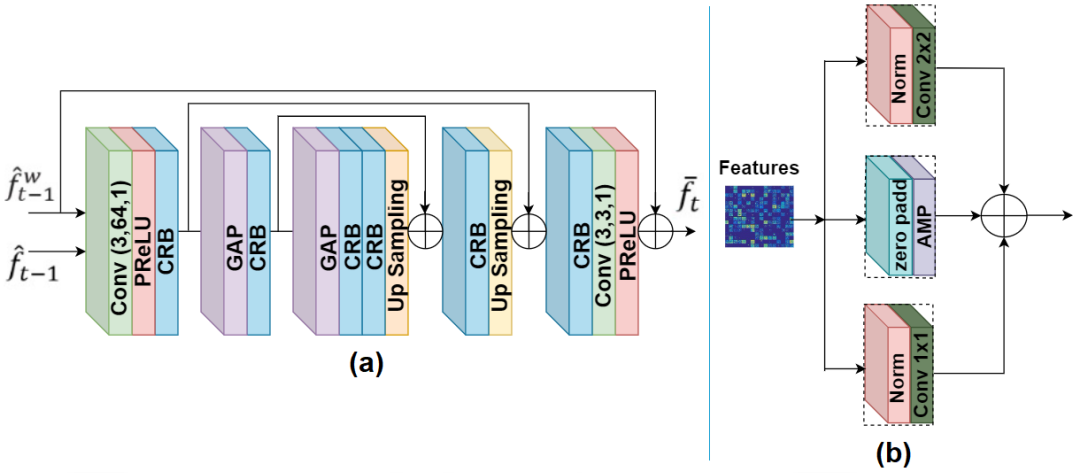

Figure 5.3: a) The proposed architecture for the motion compensation module, where CRB, GAP, and PReLU define the channel residual block, global average pooling, and parametric ReLU, respectively; (b) indicates the proposed channel residual block (CRB), where Norm and AMP represent normalization and average max-pooling.

$\hat{f}_{t-1}$ are concatenated as the input to eliminate the artifact and passed to a CNN model in MC.

Some works (Lin et al., 2020) (Feng et al., 2020) (Hu et al., 2020) (Lu et al., 2020) utilized a CNN model from the first baseline paper (Lu et al., 2019a) that used a modified UNet (Ronneberger et al., 2015) for artifact reduction. However, we observe that the generated predicted frame delivers blurry results because of ignoring the intermediate layers in UNet. Therefore, learning may slow down and put the network at risk of skipping the layers that capture abstract features. Later researchers (Hu et al., 2021) (Yang et al., 2020a) (Yang et al., 2020b) employed multiple reference frames for generating predicted frames and faced time complexity issues. Motivated by these, we propose a CNN model with a channel residual block (CRB) in the MC module, as illustrated in Fig. 5.3 (a). The MCCRB aims to increase the system's effectiveness and improve compression efficiency while simultaneously reducing the number of parameters. In particular, by including a CRB in feature maps, additional channels are assigned to the features in the lower layers

of the network, allowing for the equivalent computational cost. Our proposed MCCRB consists of two convolution layers, two activation functions (PReLU) and global average pooling layers (GAPs), two up-sampling layers, and six CRB blocks. Some residual connections are added to the network, with the final activation after the summation.

Channel residual block (CRB). Our special CRB block is depicted in Fig. 5.3 (b). Three sub-blocks are added to CRB to accept the feature maps separately. The first sub-block contains a normalization layer, followed by a convolution layer with a kernel size of 2×2 and a fixed feature map dimension $(64, 128)$ that executes linear transformation on the input feature map to retain the channel size. An average max-pooling layer and a zero-padding layer are employed in the second sub-block to improve feature map dimensions and shorten the sequence length of feature maps. In the third sub-block, a layer of normalization using a convolution (kernel size of 1×1, 32 filters, and stride of 2 followed by a PReLU) enhances the feature map dimension while decreasing the sequence length of feature maps. To better visualize the predicted frame and boost the RDO performance, we concatenate the outputs of all three sub-blocks using residual connections and their updated feature maps. Moreover, this predicted frame is also able to boost the performance of the residual frame in the system.

5.4. Residual Compression Network

The r_t is derived between f_t and $\bar{f}_t$. Then it is encoded and decoded by the image compression network (Wu et al., 2020). It is significant to map r_t to a smaller domain to compress data with high efficiency using the residual encoder. Besides, the residual decoder is employed to rebuild $\hat{r}_t$. Furthermore, on the basis of some convolution layers and a GDN/IGDN-based autoencoder type network, the r_t compresses (Ballé et al., 2018). Similar to the motion compression network, r_t is converted to z_t after the residual encoder. Through the quantization procedure, it is transferred to $\hat{z}_t$, and after the residual

decoder, it is turned into $\hat{r}_t$.

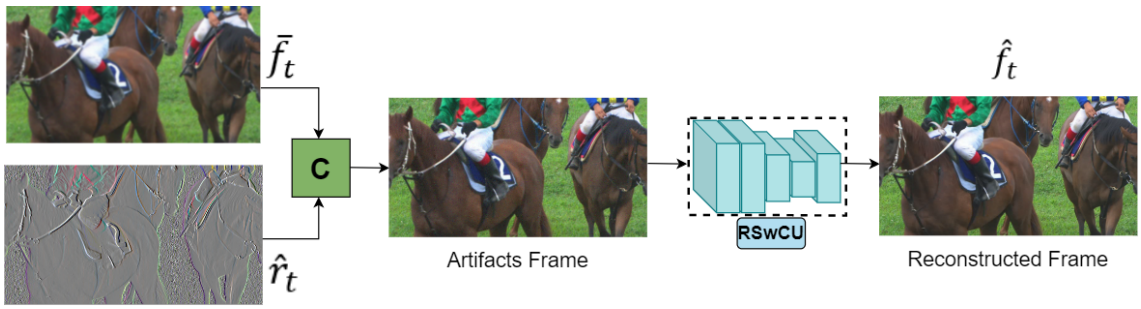

Figure 5.4: Our artifact contraction module (ACM). The RSwCU represents the proposed residual swin convolution UNet block.

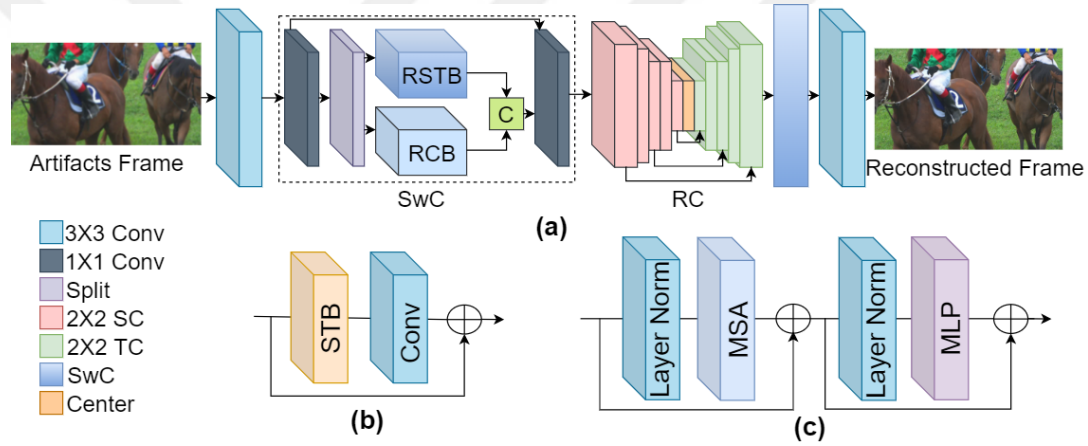

Figure 5.5: The proposed residual swin convolution UNet (RSwCU) block for ACM. (a) denotes the RSwCU block, (b) residual swin transfer block (RSTB), and (c) swin transfer block (STB) (Liu et al., 2021b). The terms RCB, SwC, C, RC, MSA, and MLP refer to the residual convolution block, swin convolution block, concatenation, residual connection, multi-head self-attention, and multi-layer perception, respectively. The notations "SC" and "TC" refer to a 2×2 strided and transposed convolution with stride 2.

5.5. Artifact Contraction Module

We discovered that there are still compression artifacts in $\hat{f}_t$ and $\hat{r}_t$ owing to quantization in the encoder-decoder network, which resulted in a low-quality reconstructed frame. Aside from that, it is also a key contributor to the enhancement of RDO performance. Therefore, we propose a compression artifact contraction architecture capable of improving the reconstructed frame while increasing the RDO performance. Fig. 5.4 depicts the

proposed artifact Contraction Module (ACM) module. At first, $\hat{f}_t$ and $\hat{r}_t$ are concatenated and produce $C(\hat{f}_t,\ \hat{r}_t)$ which has compression artifact. After that $C(\hat{f}_t,\ \hat{r}_t)$ is fed into our proposed artifact contraction architecture to produce $\hat{f}_t$. Below, the proposed artifact contraction architecture is explained in detail.

Residual-swin-convolution-uNet. Fig. 5.5 (a) depicts the proposed artifact contraction architecture, referred to as Residual-Swin-Convolution-UNet (RSwCU). The RSwCU is primarily composed of two 3×3 convolution layers, two swin convolution blocks (SwCs), and one UNet (Ronneberger et al., 2015) block. There are four levels of the UNet block in our proposed RSwCU. Each level consists of a residual link between 2×2 strided convolution (SC) (down-sampling) and transposed convolution (TC) (up-sampling). There are 16, 32, 64, and 128 channels at each layer from the first to fourth levels.

In Fig. 5.5 (a), a SwC block connects the residual swin transformer block (RSTB) (see Fig. 5.5) (b) and the residual convolutional block (RCB) (Lim et al., 2017) through two 1×1 convolutions, split-concatenation processes, and a residual connection. A 1×1 convolution is first applied to an input feature tensor F of an artifact image. Then, it is divided into two equal feature map groups: F_1 and F_2 by split layer. We define such a procedure as follows.

$$F_1, F_2 = Split(Conv1 \times 1(F)). \tag{5.2}$$

Then, F_1 and F_2 are passed into an RSTB (consists of a convolution layer and swin transformer block (Liu et al., 2021b), see Fig. 5.5 (b, c)) and RCB, respectively, resulting in the appearance of

$$M_1, M_2 = RSwinTransformer(F_1), RConv(F_2). \tag{5.3}$$

Subsequently, M_1 and M_2 are concatenated and utilized as the input of a 1×1 convolution

with a residual link to the original input, F. As a result, the SwC block's ultimate output is defined by

$$N = Conv1 \times 1(Concat(M_1, M_2)) + F. \tag{5.4}$$

It is worth noting that our suggested RSwCU has several advantages over artifact images. First and foremost, the SwC block combines the regional capabilities of the RCB with the non-regional capabilities of the RSTB. Here, the residual connection of RCB and RSTB establishes an identity-based link in the SwC block, enabling the aggregation of features at multiple levels. Second, the variational UNet enhances RSwCU's ability to represent local and non-local phenomena of artifact images. Third, to decrease the computation burden (e.g., parameters), the split and concatenation procedures are employed as group convolution with 2 groups. Please refer to the ablation study to see the effectiveness of the proposed RSwCU.

5.6. Loss Function And Entropy Coding

Loss Function. Our goal is to use the smallest feasible number of bits for video encoding while reducing distortion between f_t and $\hat{f}_t$. For training, we use the following rate-distortion function.

$$\alpha \mathrm{D} + \mathrm{R} = \alpha \mathrm{d}\left(f_t, \hat{f}_t\right) + \left(\mathrm{N}\left(\hat{a}_t\right) + \mathrm{N}\left(\hat{z}_t\right)\right) \tag{5.5}$$

where the distortion between f_t and $\hat{f}_t$ is denoted by $\mathrm{d}\left(f_t, \hat{f}_t\right)$, and the mean square error (MSE) or multi-scale structure similarity (MS-SSIM) is utilized to measure this distortion. $\mathrm{N}(.)$ denotes the number of bits utilized for encoding. The bitstreams encode both $\hat{a}_t$ and $\hat{z}_t$ representations, and a trade-off between bit numbers and distortion is specified by α, the Lagrange multiplier. The quantization step is needed prior to entropy coding, which needs a_t and z_t. As long as the quantization process is not differentiated, end-to-end

training is infeasible. To tackle this issue, we employ the approach in (Wu et al., 2018b) during training where the quantization procedure is substituted by the uniform noise, i.e, $\hat{a}_t = a_t + U$, and $\hat{z}_t = z_t + U$. We directly use the rounding technique in the predicted step, i.e, $\hat{a}_t = round(a_t)$, and $\hat{z}_t = round(z_t)$, respectively.

Entropy coding. A bit rate estimation network is needed to improve the network by considering both distortion and bit numbers to determine the latent representations. The entropy of the appropriate latent representation is the most precise way to estimate the bit rate. Therefore, we employ the hyperprior entropy model in (Ballé et al., 2018) to precisely estimate the bit rate $(\mathrm{N}(\hat{a}_t)$ and $\mathrm{N}(\hat{z}_t))$ of $\hat{a}_t$ and $\hat{z}_t$. In order to ensure that the decoding procedures are compatible with parallelization, we do not employ the auto-regressive entropy model (Minnen et al., 2018).

5.7. Decoded Frame Buffer

It is significant to use $\hat{f}_{t-1}$ in the ME and MCCRB networks during f_t compression, as illustrated in Fig. 5.1. Thus, the encoding operation establishes a chain of dependencies. We have implemented an automated update technique, a decoded frame buffer (DFB), to tackle this issue and make the training process more efficient. More specifically, each iteration of the training stores $\hat{f}_t$ in a buffer. When encoding f_{t+1}, the buffer containing $\hat{f}_t$ is utilized for the ME and MCCRB networks. As a result, every training sample in the DFB will be modified once for each epoch. We can optimize and save the previous frame at each iteration, which allows us to complete the process faster and provide precise reference frames.

6. EXPERIMENTAL SETUP

6.1. Dataset And Evaluation Matrix

The experimental datasets are primarily separated into two types: training data and test data. We randomly select $300k$ images from the Open Images dataset (Krasin et al., 2017) and crop them to a 256×256 pixel size for training. For testing, the KODAK image dataset (Kodak, 1993) and CLIC professional validation dataset (cli, b) are employed, including high-resolution natural images. The KODAK dataset comprises 24 photos with a resolution of 512×768 pixels and a broad range of contents and patterns, which are artifact-sensitive (restricted color gradients). As a result, it's frequently been employed to test image compression techniques. The CLIC dataset (cli, b) includes 41 pictures acquired by mobile phones and professional cameras. The images have greater resolutions, with an average size of 1913×1361 pixels for mobile shots and 1803×1175 pixels for professional photos.

Our proposed DeepPyNet is evaluated on two challenging datasets: MPI-Sintel (Butler et al., 2012) and KITTI 2015 (Geiger et al., 2013), and also fine-tuned other datasets, FlyingChairs (Dosovitskiy et al., 2015) and FlyingThings3D (Mayer et al., 2016). We have utilized all training images from 'Clean' and 'Final' to train the model for the MPI-Sintel dataset, which includes 1041 trained image pairs generated in two different passes ('Clean' and 'Final'). For the KITTI datasets 2015, the model is trained using 28,058 image pairs, followed by (FlyingChairs) and FlyingThings3D to finalize the model. The ground truth is utilized solely for validation. For the experiment, photometric and spatial augmentation is performed. Torchvision ColorJitter is used with brightness 0.3, contrast 0.3, saturation 0.4, and hue 0.8 /π. In terms of KITTI 2015, the augmentation degree is decreased to brightness 0.2, contrast 0.2, saturation 0.3, and hue 0.5 /π. The color

increase is done separately on each image with a probability of 0.2. To rescale and stretch the images, the spatial augmentation is performed with the probability of 0.9 in the range $2^{[-0.3,2.0]}$ where MPI-Sintel $2^{[-0.3,1.0]}$, and KITTI 2015 $2^{[-0.2,2.0]}$ respectively.

In training, we utilize the Vimeo-90K dataset (Xue et al., 2019) comprises 89,800 video clips, each with seven frames at a size of 448×256 pixels. The video sequences are randomly cropped to 256×256 before training. We used the following four datasets to test our method's compression performance.

- **HEVC Test Sequences (Sullivan et al., 2012):** It is the most common test sequence for assessing video compression performance. Our experiment uses Class B (1920×1080), Class C (832×480), and Class D (352×288) datasets.
- **MCL-JCV Dataset (Wang et al., 2016):** It consists of 24 video clips with a 1920×1080 resolution. This dataset is used to judge the quality of the video.
- **Ultra Video Group (UVG) Dataset (Mercat et al., 2020):** It is a video dataset with a high frame rate ($120fps$), in which the motion between adjacent frames is limited. According to the settings described in (Wu et al., 2018a) (Habibian et al., 2019) (Lu et al., 2019a), we conduct our studies of video sequences with 1920×1080 resolutions.
- **Video Trace Library (VTL) Dataset (Hu et al., 2021):** It comprises many raw YUV sequences utilized for low-level computer vision applications. In our studies, we employ 20 video sequences with 352×288 resolutions. The maximum duration of the video clips is set at 300 frames.

6.1.1. Evaluation metrics

This article evaluates the rate-distortion in bits per pixel (bpp) while the model is optimized by employing the PSNR (Hore and Ziou, 2010) and MS-SSIM (Wang et al., 2003). To show their coding efficiency, rate-distortion (RD) curves are generated. We followed the same set of (Bégaint et al., 2020), and for MS-SSIM, the lambda values are fixed to 2.41, 5.24, 8.31, 15.65, 30.43, and 60.56. For transformer experiment, λ is 0.014, 0.028, 0.044, 0.058, 0.064, and 0.077 for training the network.

The average endpoint error (EPE) (Alhersh et al., 2020) and the proportion of erroneous pixels (F1) are used to evaluate the optical flow estimation.

To analyze the rate-distortion effectiveness between the reconstructed frame and the ground-truth frame, we employ the PSNR, MS-SSIM (Wang et al., 2003), and Bjøntegaard delta bit rate (BDBR) (Bjontegaard, 2001). The PSNR/MS-SSIM of each video sequence is obtained by averaging the PSNR/MS-SSIM of all reconstructed frames. The average number of bits needed for motion and residual coding in each frame is measured using Bpp (bits per pixel). The learning-based approaches have a fixed Group of Pictures (GOP) configuration but not traditional codecs. There is no restriction on the GOP size to compare the learning-based video codec to the conventional one. For example, the GOP size for the UVG, MCL-JCV, and VTL datasets is set to 12. Furthermore, the equivalent GOP size for H.265/H.264 in these works is 12 or 10. The x265 LDP high-speed mode is employed for H.265, and the settings are followed from (Yang et al., 2020a). To provide a fair comparison, the results of the methods (i.e., DVC-CVPR19 (Lu et al., 2019a), Djelouah-CVPR19 et al. (Djelouah et al., 2019), Hu-ECCV20 et al. (Hu et al., 2020), M-LVC-CVPR20 (Lin et al., 2020), Lu-ECCV20 et al. (Lu et al., 2020), FVC-CVPR21 (Hu et al., 2021), DCVC-NeurIPS21 (Li et al., 2021b), RLVC-JSTSP21 (Yang et al., 2020a), Sheng-21 (Sheng et al., 2021)) are obtained from their respective articles. To demonstrate

the effectiveness of our method, experiments with larger GOP sizes were also carried out in the ablation study.

6.2. Implementation Details

For image compression, all experiments are carried out on a Windows 10 workstation with an Intel Core i7 processor, 32GB of RAM, and a single NVIDIA GeForce RTX 2070 GPU with 8GB of memory running under the CUDA 10.0. We used Python 3.7.0 with a Conda environment to finish the experiment's code. Pytorch 1.0.0 is used as the deep learning framework. For the model implementation process, the Adam optimizer (Kingma and Adam, 2015) is conducted to train all models for 1.8M steps with a batch size of 8. For the first $110k$ iterations, the learning rate is determined to 0.0003, then reduces to 0.00003 for the other $35k$ iterations, and finally to 0.00001 for the final $35k$ iterations. The channel numbers of the latent and hyper latent variables are set in the proposed model at 320 and 192, respectively.

Using the FlyingChairs and FlyingThings3D dataset, DeepPyNet is trained for 200k iterations with a batch size of 3, and the learning rate is 0.0001 on both MPI-Sintel and KITTI 2015. MPI-Sintel and FlyingChairs are combined on MPI-Sintel for another 150k iterations with batch size 6. Finally, we fine-tuned on KITTI-2015 for an additional 50k iterations, utilizing the weights on MPI-Sintel. For the implementation, we used PyTorch, and each of the modules is randomly initialized. We used an 32 GB memory with Nvidia 2070GPU. We also used Google Colab for the ablation studies. We utilized the AdamW (Loshchilov and Hutter, 2017) optimizer during training, and a total of 6.1M parameters were employed. The average endpoint error (EPE) (Alhersh et al., 2020) and the proportion of erroneous pixels (F1) are used to evaluate the optical flow estimation.

For video compression, different α values are utilized for training. For example, α = 64,

128, 256, and 512 are used for PSNR, and α = 16, 32, 64, and 128 are employed for MS-SSIM. The proposed model is trained in two steps. Firstly, we set α = 1024 and train the model for 2,000,000 steps using mean square error at a high bit rate. We fine-tune the pre-trained model for a further 500,000 iterations for α = 64, 128, 256, and 512. In the second stage, models are further fine-tuned for roughly 80,000 steps using the MS-SSIM criteria as the distortion term to improve MS-SSIM performance. The previous learning rate for newly added modules is $5e-5$, and the final learning rate is $2e-5$ throughout the fine-tuning phases with batch size 4.

7. RESULTS AND DISCUSSIONS

7.1. Qualitative Results Of CNN Model

We present some visualization outcomes to make the efficiency of our approach more apparent. Figure 7.1 demonstrates the qualitative comparisons (final reconstructed images) of some images from KODAK dataset (Kodak, 1993) dataset. In Figure 7.1, we compare our results with existing methods (Skodras et al., 2001) (fau,) (Bellard,) (Ballé et al., 2016). To illustrate the efficiency of our proposed technique, we highlight a few specific areas of the reconstructed images for a more in-depth examination. Our reconstructed images have a higher PSNR, i.e., 37.9dB (Kodim 23.png), 34.21dB (Kodim 24.png), and 30.01dB (Kodim 19.png), and maintain around the same bit rates as other methods. Besides, the texture of the images is more vibrant (especially the patterns around the eyes of the birds (row 1), the drawing (row 2), and the window (row 3), allowing us to preserve the finer feature pleasingly.

Usually, textured areas (high contrast) are allocated more bits than non-textured areas (low contrast), resulting in better visual quality at the same bit rate. To display the efficiency of the proposed AAB, the visualizations of kodim23.png, kodim19.png, and kodim24.png from the KODAK (Kodak, 1993) dataset are depicted in Figure 7.2. From Figure 7.2 (b and c), it can be shown that AAB distributes weights vertically and horizontally to suppress irrelevant information effectively. As a result, in latent (Figure 7.2(b)), it assigns more bits to regions of high contrast (objects) while assigning fewer bits to regions of low contrast (background). A vivid representation of the allotted bits can be seen in Figure 7.2 (c), particularly the surface regions of the objects.

In order to demonstrate the efficiency of our proposed DIRB, the visualizations are portrayed in Figure 7.3 of kodim23.png, kodim19.png, and kodim24.png from KODAK (Ko-

Figure 7.1: The qualitative performance comparison of the our reconstructed images with existing methods, such as Balle et al. (Ballé et al., 2016), BPG444 (Bellard,), JPEG (Skodras et al., 2001), and VTM 8.0 (fau,). These images are taken from KODAK(Kodak, 1993) dataset.

dak, 1993) dataset. The result after the last convolution layer is shown in Figure7.3 (b), and after applying the DIRB, the final reconstructed images are shown in Figure 7.3 (c).

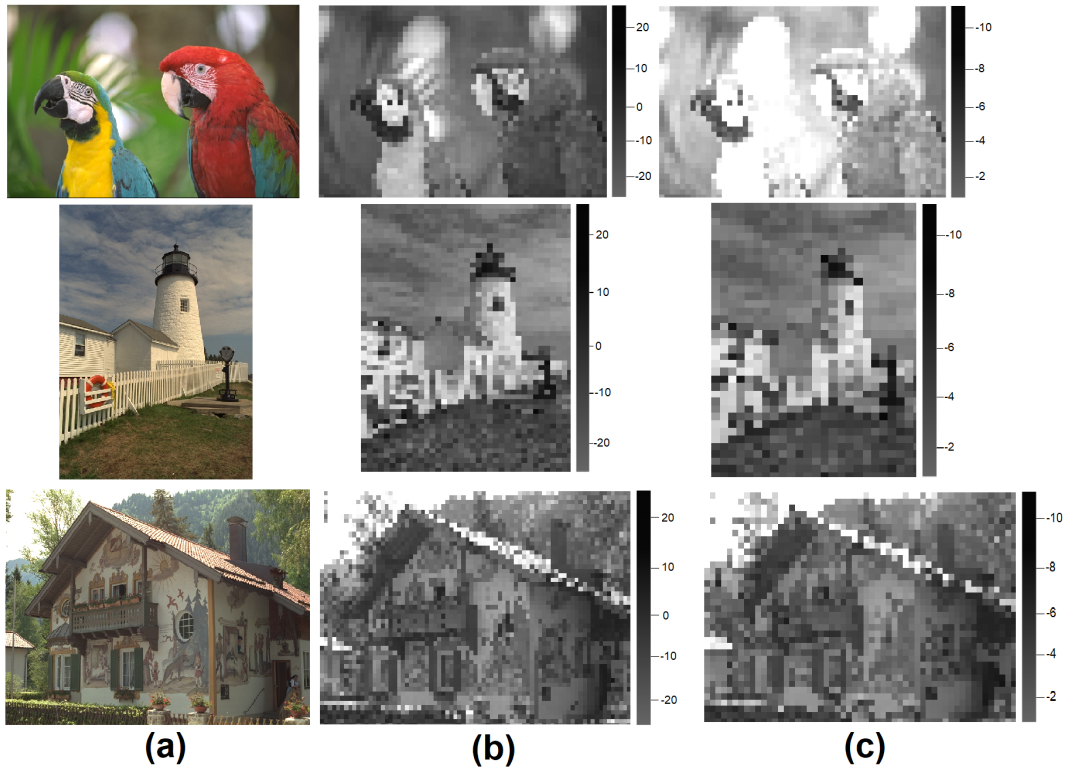

Figure 7.2: The visualization result of AAB regarding kodim23.png, kodim19.png, and kodim24.png from KODAK (Kodak, 1993) dataset. (a) original image, (b) Latent, and (c) Allocated bits (see the edges of the objects).

We can see that the learned residual images (Figure 7.3 (b)) include a disproportionate amount of high-frequency information. On the other hand, the final reconstructed images (Figure 7.3 (c)) also aid in perfectly predicting the spectral analysis of the images with a better display.

7.2. Qualitative Results Of Transformer Model

Fig. 7.5 shows a qualitative comparison of the reconstructed images between our proposed model with two popular methods (Minnen et al., 2018) (Bellard,). From the PSNR value, our reconstructed image exhibits sharp textures and fewer artifacts at low bit rates (0.093Bpp) with high PSNR (32.91dB). On the other hand, the reconstructed images from BPG444 and Minnen et al. produced blurry results (e.g., the flower region) and compres-

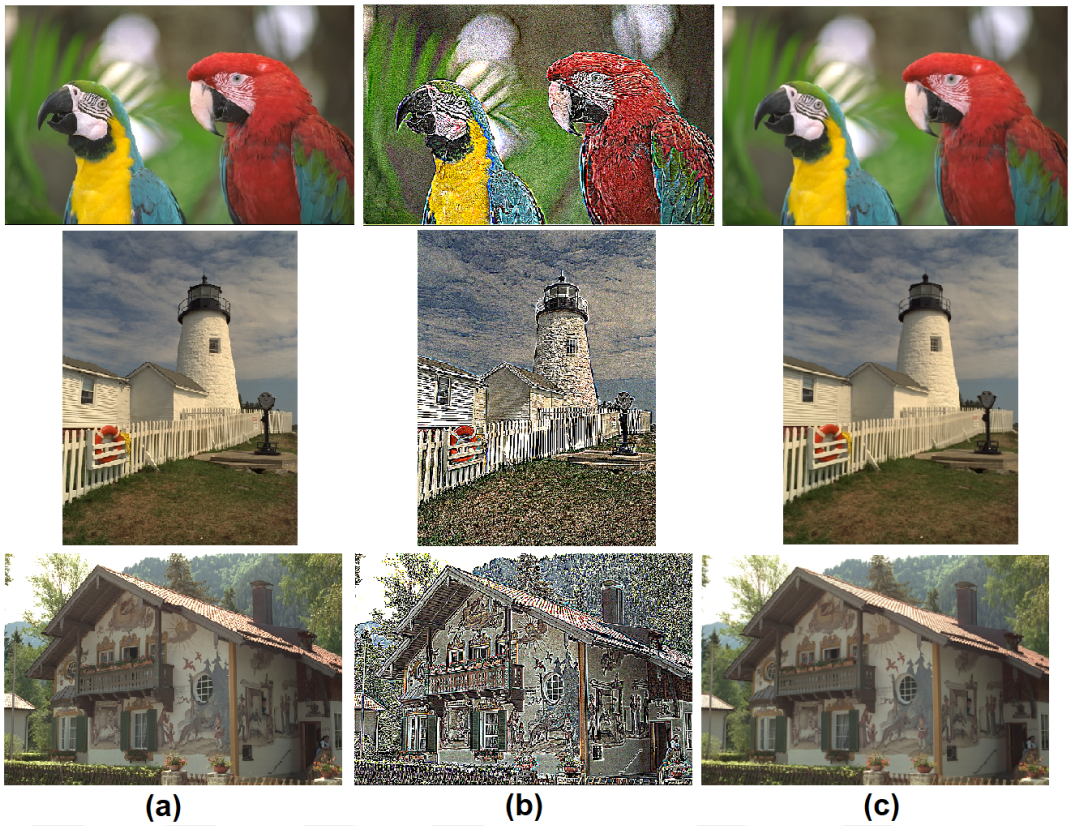

Figure 7.3: The visualization result of DIRB in terms of kodim23.png, kodim19.png, and kodim24.png from KODAK (Kodak, 1993) dataset. (a) original image, (b) the reconstructed image without applying DIRB, and (c) the reconstructed image after applying DIRB.

sion artifacts with low PSNR compared to ours (32.91 vs. 31.02, and 32.91 vs. 30.92, respectively).

7.3. Qualitative Results Of Optical Flow Model

Figure 7.6 represents the flow predictions of the MPI-Sintel and KITTI 2015 test set. DeepPyNet provides a clean and accurate optical flow, which comes from the learning capacity of the iterative refinement flow method.

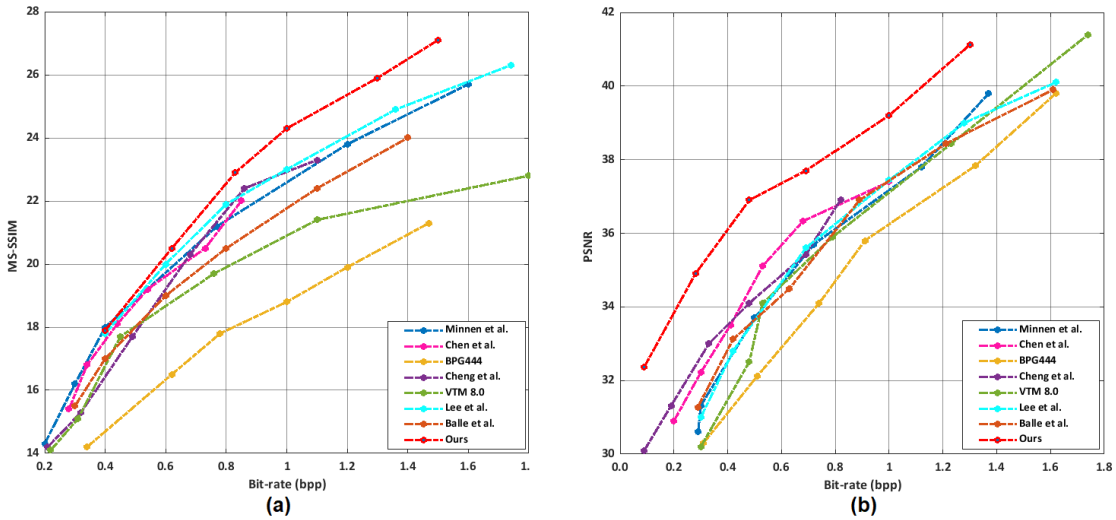

Figure 7.4: The RD performance assessment on the KODAK dataset(Kodak, 1993). (a) the performance of MS-SSIM in decibels (i.e., $-10log_{10}(1-MS-SSIM)$), and (b) the performance of PSNR.

7.4. Qualitative Results Of Video Compression Model

This sub-section provides qualitative results of predicted, residual, and final reconstructed frames using our proposed blocks (i.e., DRSA, CRB) and the ACM module with RSwCU. The qualitative comparison of the original and our reconstructed frames on all datasets (UVG, HEVC, MCL-JCV, and VTL) is shown in Fig. 7.7. The Bpp represents the average bits per pixel of each video frame. From the MS-SSIM values and Bpp from Fig. 7.7, our reconstructed frames exhibit fewer compression artifacts, fewer bit rates, and retain more abstract features. As a result, our method has a higher visual quality while creating fewer fuzzy contours. Although the frame in the VTL dataset has a lower quality, our proposed method still produces a 0.954 MS-SSIM with 0.211 Bpp.

Fig. 7.8 shows the qualitative results of the reconstructed/decoded frame with and without considering the ACM module with RSwCU in the proposed method. It is noticeable that the decoded results without applying the ACM module have more artifacts with noise and are less smooth. In contrast, the outcomes of our technique (by considering the ACM

Figure 7.5: Qualitative performance. At low bit rates, visual assessment comparison with two earlier methods (Minnen et al., 2018) (Bellard,) on KODAK dataset.

module) are comparatively much smoother (i.e., the grasses in Fig. 7.8(c) upper part and the hand Fig. 7.8(c)) lower part with fewer artifacts and better visualization. Besides, the PSNR/MS-SSIM value for (b) is 31.44dB/0.912 (upper part) and (b) is 31.67dB/0.927 (lower part), while it is significantly high for (c) at 35.11dB/0.981 (upper part) and (c) at 33.98dB/0.976. Thus, our ACM module can improve more accurate results with the RSwCU network.

We proposed a DRSA block in the autoencoder network for the MVC module to boost the system's performance. The visualization of DRSA blocks with the raw optical flows is presented in Fig. 7.9. Our proposed DRSA can learn optical flow with abstract features and reduce unnecessary information (see Fig. 7.9 (c, d)), which can increase the RDO performance and make it faster. In Fig.7.10, we also show the MC module's predicted frame using our proposed CRB block. It can be shown that without the CRB block in Fig.

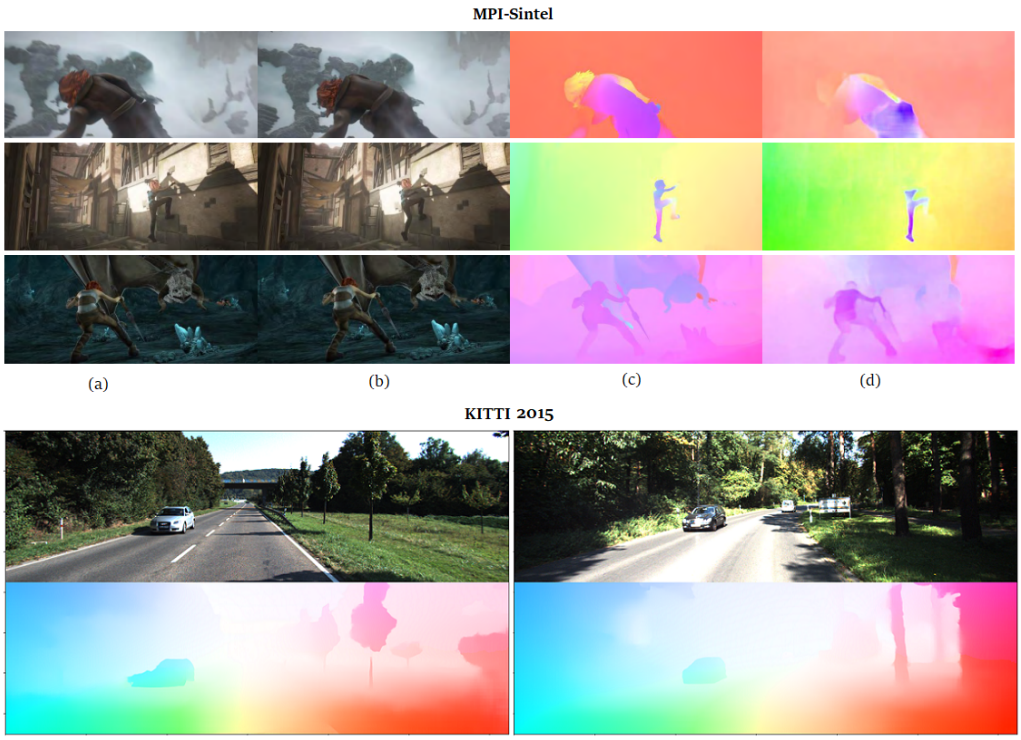

Figure 7.6: The examples of optical flow estimation. Here, the upper half and bottom half portions are the predictions of the flow result from the MPI-Sintel and KITTI 2015 dataset, respectively. In the upper half portion: (a) and (b) represent the frames of the MPI-Sintel test dataset. (c) represents the ground truth of these frames and (d) the output of the optical flow through the DeepPyNet. The frames of the test dataset of KITTI 2015 and their optical flow represents in the bottom half portion.

7.10 (b, c), the objects in the residual frame (c) cannot be identified, and the predicted frame (b) is also of lower quality. However, the predicted frame (d) gets smoother after applying the CRB, while the residual frame becomes more apparent (e), which assists in optimizing compression performance as well. For a more detailed look at the efficacy of DRSA and CRB blocks, please refer to the ablation study in Table 7.10.

7.5. Quantitative Results Of CNN Model

To evaluate our proposed model, the RD performance is computed. We employ the PSNR as the quality measure, as illustrated in Figure 7.4 (a). Our technique is evaluated against a variety of well-known image compression algorithms (both classical and deep learning-

Table 7.1: The RD performance (MS-SSIM and PSNR) of the CLIC (cli, b) Professional Validation dataset. Bold indicates the highest performance and underline indicates the second highest.

Approaches	MS-SSIM	PSNR
JPEG(Skodras et al., 2001)	17.6	35.4
BPG44 (Bellard,)	22.3	41.2
VTM 8.0 (fau,)	22.6	**42.5**
Lee et al.(Lee et al., 2018)	26.6	40.4
Balle et al.(Ballé et al., 2018)	22.5	40.1
Minnen et al.(Minnen et al., 2018)	<u>27.1</u>	40.0
Cheng et al.(Cheng et al., 2020)	24.1	36.4
Ours	**27.4**	<u>40.8</u>

based), including (Bellard,) (fau,) (Minnen et al., 2018) (Chen et al., 2021) (Cheng et al., 2020) (Lee et al., 2018)(Ballé et al., 2016). When compared to (Bellard,) (Minnen et al., 2018) (Ballé et al., 2016) (Cheng et al., 2020) (Lee et al., 2018) (Ballé et al., 2016), our method outperforms them by a large margin, specially from the most popular methods Chen et al. (Chen et al., 2021) (around 41.12 vs. 37.4), and Cheng et al. (Cheng et al., 2020) (around 41.12 vs. 37.1). However, the bit rates are slightly lower (about 0.7%) when comparing our approach to the traditional method (fau,) (around 41.4 vs. 41.12).

The experiments are also carried out using the MS-SSIM quality measure, as seen in Figure 7.4 (b). We provide MS-SSIM values in decibels (i.e., $-10log_{10}(1 - MS - SSIM)$) to better illustrate the progress. It is clearly said that our method shows state-of-the-art performance against both the traditional methods, including (Bellard,), and (fau,), and deep learning-based methods, including, (Minnen et al., 2018) (Chen et al., 2021) (Cheng et al., 2020) (Lee et al., 2018) (Ballé et al., 2016). Therefore, we can say that the AAB, GMB, and DIRB we have presented significantly influence showing higher RD performance and improving the reconstructed image's similarity. Please refer the ablation study (in next sub-section) to get a better idea of the modules' efficacy.

We employ another CLIC (cli, b) professional validation dataset to confirm the robustness of our technique, and the results are shown in Table 7.1. It is noteworthy that our approach also yields state-of-the-art results in terms of MS-SSIM which we express in decibels (i.e., $-10log_{10}(1 - MS - SSIM)$). However, regarding PSNR, our method achieves the second-highest result (underline results in Table 7.1), which is approximately 4% less than the traditional method (fau,) (around 42.5 vs. 40.8). However, it outperforms all existing deep learning approaches.

7.6. Quantitative Results Of Transformer Model

Fig. 7.11 illustrates a quantitative comparison in terms of PSNR and MS-SSIM on the public Kodak dataset with earlier attempts based on deep learning and transformers. As shown in Fig. 7.11, our model shows the state-of-the-art performance in PSNR even from the recent works, for example, Lu et al. (approximately 40.1 vs. 40.7), Cheng et al. (roughly 36.9 vs. 40.7), and the popular work Minnen et al. (around 40.4 vs. 40.7). The MS-SSIM ratings are in decibels (dB) to enhance readability: $MS - SSIM_{dB}$ = -10 . $log_{10}(1 - MS - SSIM)$. As far as we know, our method likewise exhibits the state-of-the-art when compared in MS-SSIM to other current approaches. However, this performance also indicates the efficacy of our proposed CGAM compared to the earlier entropy models employed in (Ballé et al., 2018) (Minnen et al., 2018) (Chen et al., 2021) (Lu et al., 2021).

7.7. Quantitative Results Of Optical Flow Estimation Model

Our model is trained using the FlyingChairs and then evaluated with the Train Split validation on the MPI-Sintel dataset. The result is illustrated in Table 7.2, and then we divide the results using the training data. With various inputs, our proposed DeepPyNet network is trained three times and given results on the clean pass of MPI-Sintel. When utilizing FlyingChairs and FlyingThings3D for training, our technique beats all current methods

Table 7.2: The Results Comparison with the prior approaches on MPI-Sintel and KITTI 2015.

Training Data	Method	MPI-Sintel (train)		MPI-Sintel (test)		KITTI 2015 (train)		KITTI 2015 (test)
		Clean	Final	Clean	Final	F1-epe	F1-all	
Not mentioned	FlowFields (Bailer et al., 2015)	–	–	3.75	5.81	–	–	15.31
Not mentioned	FlowFields++ (Schuster et al., 2018)	–	–	2.94	5.49	–	–	14.82
MPI-Sintel	DCFlow (Xu et al., 2017)	–	–	3.54	5.12	–	–	14.86
MPI-Sintel	MRFlow (Wulff et al., 2017)	–	–	2.53	5.38	–	–	12.19
FlyingChairs and FlyingThings3D	SDeepPyNet (Ranjan and Black, 2017)	4.12	6.69	5.57	8.43	–	–	–
	HD3 (Yin et al., 2019)	3.84	8.77	–	–	13.17	24.0	–
	LiteFlowNet (Hui et al., 2018)	2.48	4.04	–	–	10.39	28.5	–
	PWC-Net (Sun et al., 2018)	2.55	3.93	–	–	10.35	33.7	–
	LiteFlowNet2(Hui et al., 2020)	2.24	3.78	–	–	8.97	25.9	–
	VCN (Yang and Ramanan, 2019)	2.21	3.68	–	–	8.36	25.1	–
	MaskFlowNet (Zhao et al., 2020)	2.25	3.61	–	–	–	23.1	–
	FlowNet 2.0 (Ilg et al., 2017)	2.02	3.54	–	–	10.08	30.0	–
	FlowCaps (Jayasundara et al., 2021)	2.13	2.51	–	–	7.83	–	–
	RAFT+NCUP (Eldesokey and Felsberg, 2021)	1.41	2.75	–	–	4.83	17.4	–
	OAS-Net (Kong et al., 2021)	2.55	3.65	–	–	–	–	–
	DeepPyNet (ours)	**1.40**	**2.50**	–	–	5.02	17.46	–
FlyingChairs, FlyingThings3D, and MPI-Sintel	FlowNet 2.0 (Ilg et al., 2017)	1.45	2.01	4.16	5.74	2.30	6.8	11.48
	HD3 (Yin et al., 2019)	1.87	1.17	4.79	4.67	1.31	4.1	6.55
	IRR-PWC (Hur and Roth, 2019)	1.92	2.51	3.84	4.58	1.63	5.3	7.65
	ScopeFlow (Bar-Haim and Wolf, 2020)	–	–	**3.59**	4.10	–	–	6.82
	DeepPyNet (ours)	**1.07**	**1.15**	4.01	**3.41**	**1.31**	**3.8**	**6.25**

Table 7.3: The comparison between the DeepPyNet and the prior works regarding the number of parameters and training iterations.

Method	Parameters (Millions)	Training Iterations
FlowNet2 (Ilg et al., 2017)	162.5M	7100 K
FlowNetS (Dosovitskiy et al., 2015)	38.7M	1700 K
FlowNetC (Dosovitskiy et al., 2015)	39.2M	1700 K
LiteFlowNetX (Hui et al., 2018)	0.9M	2000k
LiteFlowNet (Hui et al., 2018)	5.4M	2000k
IRR-PWC (Hur and Roth, 2019)	6.4M	850k
PWC-Net+ (Sun et al., 2019)	9.4M	1700k
DeepPyNet (ours)	**6.1M**	**220k**

regarding MPI-Sintel (train). Our approach obtains an average EPE (end-point-error) of 1.40 and 2.50 on the MPI-Sintel(train) Clean and Final pass, while FlowNet2 achieves 2.02 and 3.54, respectively. Then our proposed DeepPyNet is got $0.73(2.13$ vs. $1.40)$ and $1.15(2.55$ vs. $1.40)$ less EPE than the popular recent methods, Flowcaps and OAS-Net, respectively. It performs slightly worse in terms of KITTI 2015 (train) in both F1-epe and F1-all. However, the improvement on MPI-Sintel demonstrates that our pyramid feature extraction is more prevalent. On the other hand, when we combine the FlyingChairs, FlyingThings3D, and MPI-Sintel by evaluating the MPI-Sintel and KITTI 2015, DeepPyNet shows state-of-the-art performance. In MPI-Sintel (train), the Clean and Final pass EPE is only 1.07 and 1.15. Similarly, in KITTI 2015 (train), the EPE is 1.31 for F1-epe and 3.8 for F1-all which is $0.99(2.30$ vs. $1.31)$ and $3(6.8$ vs. $3.8)$ less than FlowNet 2.0.

Likewise, in the test both of MPI-Sintel and KITTI 2015, DeepPyNet demonstrates excellent performance, especially in terms of KITTI 2015 (test) F1-all, it is $5.23(11.48$ vs. $6.25)$ less EPE accuracy from FlowNet and $1.4(7.65$ vs. $6.25)$ less from IRR-PWC. The structure of our network is one of the reasons for improved applicability. With optical flow being restricted as the output of a succession of the same update steps, the network learns an iterative process of refinement that imitates updates of the first-order descent algorithm. It restricts the search area, lowers the chance of overfitting, and promotes quicker workouts.

As demonstrated in Table 7.3, DeepPyNet is trained with relatively fewer iterations (only 220k). It is also one of the most compact models available, with the fewest parameters (6.1M). On the other hand, LiteFlowNet used only 5.4M parameters while its iterations were nine times more than DeepPyNet $(2000\text{k}$ vs. 220k). One last point to mention is that DeepPyNet is the only optical flow network in this table that processes a "real" 4D cost volume rather than a "pseudo" multi-channel 2D cost volume as the others do.

Table 7.4: Comparing the findings for bjontegaard delta bit rate (BDBR), calculated by the PSNR/MS-SSIM of our method with state-of-the-art on H.265 (x265 LDP very fast) in different datasets. The bit-rate savings are shown by negative BDBR values, whereas positive BDBR values indicate higher bit-rate costs. The red and green colors represent the highest and second highest result, respectively.

Dataset	DVC(Lu et al., 2019a)	Hu(Hu et al., 2020)	Lu(Lu et al., 2020)	Agustsson(Agustsson et al., 2020)	HLVC (Yang et al., 2020b)	M-LVC(Lin et al., 2020)	RLVC(Yang et al., 2020a)	L
Class B	5.66/-2.74	-/-	-13.35/-7.93	-/-	-11.75/-37.44	green-36.55/-42.82	-24.20/-50.42	
Class C	25.88/-6.88	4.94/-32.44	-/-	-/-	7.83/-23.63	-/-	-4.67/-35.94	
Class D	15.34/-18.51	-/-32.43	-6.86/-	-/-	-12.57/-52.56	-13.87/-36.27	-27.01/-48.85	
MCL-JCV	-/-	-10.60/-34.10	4.21/-	-1.82/-33.61	-/-	-/-	-/-	
UVG	10.40/8.05	-/-	-7.56/-25.49	-8.80/-38.04	-1.37/-30.12	-12.11/-25.44	-13.48/-40.62	
VTL	-/-	-/-6.04	-16.05/-	-/-	-/-	-/-	-/-	
Average	8.03/-5.02	-2.83/-26.25	-7.92/-16.71	-5.31/-35.83	-4.47/-35.94	-20.84/-34.84	-17.34/-43.96	

7.8. Quantitative Results Of Video Compression Model

Our video compression method is evaluated using MS-SSIM, PSNR, and BDBR.

7.8.1. MS-SSIM evaluation

Fig. 7.12 demonstrates the rate-distortion curves of MS-SSIM for our method with state-of-the-art methods, e.g., conventional H.264/H.265 (Wiegand et al., 2003) (Sullivan et al., 2012), deep learning-based methods e.g., DVC-CVPR19 (Lu et al., 2019a), Djelouah-CVPR19 et al. (Djelouah et al., 2019), Hu-ECCV20 et al. (Hu et al., 2020), M-LVC-CVPR20 (Lin et al., 2020), Lu-ECCV20 et al. (Lu et al., 2020), FVC-CVPR21 (Hu et al., 2021), DCVC-NeurIPS21 (Li et al., 2021b), RLVC-JSTSP21 (Yang et al., 2020a), and Sheng-21 (Sheng et al., 2021) in all datasets. Our method shows state-of-the-art/competitive MS-SSIM performance in the HEVC Class C, UVG, VTL, and MCL-JCV datasets/HEVC Class B, and D datasets. In the HEVC class B and D datasets, our method received the same MS-SSIM result (around 0.99) as Sheng-21 (Sheng et al., 2021). However, in every dataset, our method outperforms the popular recent methods, i.e., FVC-CVPR21 (Hu et al., 2021), DCVC-NeurIPS21 (Li et al., 2021b), and RLVC-JSTSP21 (Yang et al., 2020a) and the conventional methods, i.e., (Wiegand et al., 2003) (Sullivan et al., 2012) by a large margin, which reflects the efficacy of our proposed approach.

7.8.2. PSNR evaluation

From the rate-distortion curves of PSNR in Fig. 7.13, our method also shows the state-of-the-art/competitive outcomes with the learning-based methods, e.g., DVC-CVPR19 (Lu et al., 2019a), Djelouah-CVPR19 et al. (Djelouah et al., 2019), Hu-ECCV20 et al. (Hu et al., 2020), M-LVC-CVPR20 (Lin et al., 2020), Lu-ECCV20 et al. (Lu et al., 2020), FVC-CVPR21 (Hu et al., 2021), DCVC-NeurIPS21 (Li et al., 2021b), RLVC-JSTSP21 (Yang et al., 2020a), Sheng-21 (Sheng et al., 2021) and conventional methods, H.264/H.265 (Wiegand et al., 2003) (Sullivan et al., 2012) in every datasets. In particular, in the HEVC Class D dataset, H.265 outperforms all methods. However, our method overcomes the existing deep learning methods (particularly the recent methods, i.e., FVC-CVPR21 (Hu et al., 2021), DCVC-NeurIPS21 (Li et al., 2021b), RLVC-JSTSP21 (Yang et al., 2020a), Sheng-21 (Sheng et al., 2021)). Similarly, in the HEVC Class C dataset, our method is able to yield almost equal performance (around 34.5) like H.265 and Lu-ECCV20 (Lu et al., 2020). As a result, we can infer that the module's proposed blocks significantly influence the system's overall performance.

7.8.3. BDBR evaluation

In Table 7.8, we present the BDBR (Bjontegaard, 2001) findings of our method in comparison to other state-of-the-art methods (DVC-CVPR19 (Lu et al., 2019a), Agustsson et al. (Agustsson et al., 2020), Hu-ECCV20 et al. (Hu et al., 2020), M-LVC-CVPR20 (Lin et al., 2020), Lu-ECCV20 et al. (Lu et al., 2020), HLVC-CVPR20 (Yang et al., 2020a), FVC-CVPR21 (Hu et al., 2021), RLVC-JSTSP21 (Yang et al., 2020a), Liu (Liu et al., 2021a)) where H.265 is set as an anchor. Our method saves around 2%, 28%, 8%, 30%, and 11% bit saving rates in terms of PSNR as well as 5%, 11%, 10%, 10%, and 6% in terms of MS-SSIM on HEVC Class B, MCL-JCV, UVG, and VTL datasets, respectively; these results demonstrate the superiority of our proposed method. It is also clear that our

method surpasses the recent deep learning methods, i.e., FVC-CVPR21 (Hu et al., 2021), RLVC-JSTSP21 (Yang et al., 2020a), Liu (Liu et al., 2021a) and also other existing deep learning methods. On the other hand, in the HEVC Class B dataset, regarding PSNR, it exhibits state-of-the-art results; nevertheless, in MS-SSIM, it has roughly an 18% gap compared to the FVC-CVPR21 (Hu et al., 2021). Additionally, in HEVC Class D it achieves the second highest result (approximately 28% less than RLVC-JSTSP21 (Yang et al., 2020a) and slightly less than HLVC-CVPR20 (Yang et al., 2020a)) in both PSNR and MS-SSIM.

7.9. Computational Performance

Our model is built on PyTorch with CUDA support. Experiments are conducted using a Windows 10 workstation with an NVIDIA RTX 2080Ti GPU. The initial training stage takes around four days, the second training stage takes approximately two days, and the fine-tuning step takes 1 day. When encoding videos with a resolution of per 1080P frame, the encoding speed is 0.612s (resp. 0.576s), which is almost 26% faster than Sheng-21(Sheng et al., 2021) (0.827 vs. 0.612), though the decoding speed has an 18% (0.472 vs. 0.576) gap. On the other hand, in FVC-CVPR21 (Hu et al., 2021), the authors stated that the overall coding speed is 0.548s. Despite this, they used the auto-regressive entropy model, which is not GPU-friendly. As a result, the coding performance was estimated to take up a significant amount of time. In fact, they did not mention whether or not the entropy coding speed is included in their overall coding speed. Meanwhile, our model has only 10.1M parameters, whereas FVC-CVPR21(Hu et al., 2021) is 26M and Sheng-21(Sheng et al., 2021) is 10.7M, which is around 157% and 7% higher than ours, correspondingly.

7.10. Ablation Study

7.10.1. CNN image compression model

We perform some ablation studies on the KODAK dataset (Kodak, 1993) to further illustrate the robustness and effectiveness of our proposed approach.

Table 7.5: The performance comparison of different prior attention blocks with our AAB in terms of PSNR with bit rates (bpp) on KODAK dataset (Kodak, 1993).

Approaches	λ	PSNR	bpp
Proposed method	2.41	27.23	0.201
Proposed method + attention module (Cheng et al., 2020)	2.41	31.89	0.211
Proposed method + NLAM (Chen et al., 2021)	2.41	32.21	0.202
Proposed method + AAB	2.41	**32.98**	**0.208**
Proposed method	8.31	27.51	0.657
Proposed method + attention module (Cheng et al., 2020)	8.31	33.87	0.678
Proposed method + NLAM (Chen et al., 2021)	8.31	34.01	0.651
Proposed method + AAB	8.31	**35.67**	**0.591**

In Table.7.5, we provide an investigation by adopting current attention modules replacing our suggested attention module in our proposed approach for two kinds of λ values. PSNR performance is relatively poor (27.23) for λ = 2.41 and 8.31 at low and high bit rates when the attention module is not included in the baseline model. When the attention modules of Cheng et al. (Cheng et al., 2020), Chen et al.(Chen et al., 2021), and ours are utilized, the PSNR values improve by around 15% (31.89 vs. 27.23 and 32.21 vs. 27.23) for (Cheng et al., 2020) and (Chen et al., 2021), and by about 17% (32.98 vs. 27.23) for ours at low bit rates, respectively. The PSNR improves significantly when λ= 8.31, for example, for our suggested adjacent attention module, the PSNR is improved by roughly 23% (35.67 vs. 27.51) and even by around 5% (35.67 vs. 33.87 and 35.67 vs. 34.01) over the prior modules of (Cheng et al., 2020) (Chen et al., 2021).

To further verify the effectiveness of our proposed three modules in the main architecture,

Table 7.6: The performance measurement of the proposed modules in trems of PSNR, MS-SSIM, and Inference Time on KODAK dataset (Kodak, 1993). The AAB, GMB and DIRB indicate Adjacent Attention Block, Gaussian Merge Block, and Decoded Image Refinement Block, respectively.

S.N.	Baseline	AAM	GMB	RB	PSNR	MS-SSIM	Inference Time (ms)
1	✓	×	×	×	26.27	18.71	522
2	✓	✓	×	×	27.95	19.95	734
3	✓	×	✓	×	26.89	18.78	591
4	✓	×	×	✓	30.33	20.32	822
5	✓	✓	✓	×	32.51	23.41	883
6	✓	×	✓	✓	31.23	21.87	861
7	✓	✓	×	✓	33.96	24.45	901
8	✓	✓	✓	✓	**35.89**	**26.01**	**1067**

we have carried out another experiment in trems of PSNR, MS-SSIM, and Inference Time by replacing and adding the modules in Table 7.6. The PSNR and MS-SSIM performance of the baseline model are 26.27 and 18.71, respectively, when the three modules are not included as well as the inference time of 522ms. The value of PSNR and MS-SSIM dramatically increases with the inference time of 1067ms when all components are considered. For example, the improvement in PSNR and MS-SSIM is roughly 27% (35.89 vs. 26.27) and 28% (26.01 vs. 18.71). Among them, the proposed adjacent attention module and refinement block are able to boost the RD performance more, for instance, approximately 23% (33.96 vs. 26.27) in PSNR and 23% (24.45 vs. 18.71) in MS-SSIM. Eventually, it can be concluded that our proposed modules are very effective in boosting the state-of-the-art RD performance.

7.10.2. Transformer image compression model

To investigate the efficacy of the proposed modules, MHSRSA and CGAM, their performance is evaluated with the multi-head attention layer (MHSA), and the previous famous entropy model (Minnen et al., 2018) on the KODAK dataset in Table 7.7. When the MHSRSA and CGAM are utilized together for five blocks (M_E and H_E), the FLOPs, number

Table 7.7: Ablation study on different modules computed in terms of computation cost, PSNR, and MS-SSIM on KODAK dataset.

No.	No. of blocks (Transformer)	CGAM	Attention	FLOPs	#Param	PSNR	MS-SSIM
1	**5 (ME:3, HE:2)**	✓	**MHSRSA**	**16.7**	**27.3M**	**40.72**	**27.41**
		×	MHSRSA	14.3	25.6M	33.91	22.53
		✓	MHSA	19.4	37.1M	36.53	24.81
		×	MHSA	18.6	35.7M	31.41	21.14
2	6 (ME: 4, HE: 2)	✓	MHSRSA	19.1	34.3M	40.75	27.42
		×	MHSRSA	17.6	31.8M	30.44	20.31
		✓	MHSA	22.9	41.5M	33.33	21.81
		×	MHSA	20.5	39.2M	29.01	19.65

Table 7.8: The number of pyramids with multiplicative attention function (MAF) and additive bias function (ABF)

Types of pyramids	Trained on FlyingChairs		
	Testing	MPI-Sintel(clean)	MPI-Sintel(final)
Left and Middle pyramid	0.80	3.11	4.50
Left and Middle pyramid, MAF, and ABF	0.78	2.67	3.67
Left, Middle and Right pyramid, MAF, and ABF	0.75	2.73	3.20
Left, Middle and Right pyramid, MAF, and ABF	**0.71**	**2.16**	**3.10**

of parameters, PSNR, and MS-SSIM are obtained at 16.7, 27.3M, 40.72dB, and 27.41dB, respectively. However, when CGAM is removed, and MHSRSA is used with this(Minnen et al., 2018), the FLOPs are less, but parameters, PSNR, and MS-SSIM are decreased significantly (6.22% (27.3 vs. 25.6), 16.72% (40.72 vs. 33.91), and 17.8% (27.41 vs. 22.53), respectively). Utilizing MHSA in both steps, the performance is still lower/reduced. Once the six transformer blocks are used in case 2, the FLOPs and parameters are increased by around 13% (19.1 vs. 16.7) and 21% (34.3 vs. 27.3), respectively. Besides, the PSNR and MS-SSIM are improved slightly. Thus, our proposed MHSRSA and CGAM significantly influence the model.

7.10.3. Optical flow estimation model

In the ablation study, the clean and final training data of MPI-Sintel is evaluated using the training and validation data of FlyingChairs. The multiplicative attention function (MAF) and additive bias function (ABF) use extra fine-tuning on FlyingThings3D.

Table 7.9: The EPE performance of the DeepPyNet in the level of the pyramids utilizing the multiplicative attention function (MAF) and additive bias function (ABF) function.

Levels (pyramid)	Trained on FlyingChairs		
	Testing	MPI-Sintel(clean)	MPI-Sintel(final)
Level 1	0.76	2.11	3.50
Level 2	0.75	1.87	3.11
Level 3	0.78	1.73	2.78
Level 4	0.79	2.14	3.15
Level 1+2	0.72	2.16	3.10
Level 1+2+3	0.81	1.97	2.87
Level 1+2+3+4	0.83	1.31	2.32
Level 2+3+4	**0.70**	**1.30**	**2.10**

Figure 7.6 represents the flow predictions of the MPI-Sintel and KITTI 2015 test set. DeepPyNet provides a clean and accurate optical flow, which comes from the learning capacity of the iterative refinement flow method. We used four levels of the feature pyramid extraction method. The EPE results of two or three pyramids with the MAF and ABF functions are depicted in Table 7.8. FlyingChairs is fine-tuned to train on the MPI-Sintel (both clean and final pass) dataset and test the FlyingChairs dataset. It can be noted from this table that the EPE is decreasing at every level for utilizing the MAF and ABF functions.

7.10.4. Video compression model

We conduct an extensive ablation study on different modules and GOP sizes to evaluate the effectiveness of our proposed method.

Effectiveness of different modules

Table 7.10 shows the result of the ablation study in terms of PSNR, MS-SSIM, a number of parameters, and greeninference time (encoding and decoding speed in second (s)), on different modules across the entire network on the HEVC Class B dataset. When the baseline (RC, entropy coding, and DFB), ME, the proposed DRSA for MVC, CRB for MC, and ACM module are employed (Case 1) together with joint training, we received the

Table 7.10: The effectiveness of different modules with DRSA, CRB blocks on the proposed network in terms of PSNR, MS-SSIM, number of parameters (Param), and inference time (IT) greenin second (s) on HEVC Class B Dataset (Sullivan et al., 2012)

Case	Baseline	ME	MVC	MC	ACM	PSNR	MS-SSIM	Param (M)	IT (s)
1	✓	✓	with DRSA	with CRB	✓	**36.13**	**0.991**	10.15	0.826
2	✓	✓	no DRSA	with CRB	✓	34.78	0.956	11.78	1.026
3	✓	✓	with DRSA	no CRB	✓	33.98	0.951	13.01	1.196
4	✓	✓	with DRSA	with CRB	×	32.65	0.936	8.45	0.723
5	✓	×	with DRSA	with CRB	✓	30.79	0.926	9.12	0.798
6	✓	✓	no DRSA	no CRB	✓	32.01	0.951	13.22	1.401

.

highest PSNR (36.13dB) and MS-SSIM (0.991) of our system with 10.15M parameters and 0.826s. The PSNR is decreased by 1.35dB (from 36.13 to 34.78) and 2.15dB (from 36.13 to 33.98), and MS-SSIM is reduced by 3.5% (0.991 vs. 0.956), and 4% (0.991 vs. 0.951) for omitting the DRSA and CRB separately. However, parameters and inference time are also increased by around 14% (10.15 vs. 11.78) and 22% (10.15 vs. 13.01) in case 2 as well as 19% (0.826 vs. 1.026) and 31% (0.826 vs. 1.196) in case 3, respectively. In contrast, the system delivers too poor performance without DRSA and CRB, as seen by lowered PSNR and MS-SSIM and dramatically higher parameters and inference time (in case 6). To further illustrate the ACM module's effectiveness, we conducted an experiment where the ACM was removed from the architecture. As a result, the PSNR and MS-SSIM decreased significantly (in case 4), though the parameters and inference time were also decreased. An additional experiment is performed with the ME module in case 5. The system would be unable to provide a satisfying RDO performance without the ME module. In conclusion, we can claim that our proposed blocks, DRSA and CRB, as well as the proposed module ACM, have an unrivaled influence on improving the RDO performance of the system.

GOP evaluation

We further analyze the performance of the different group of pictures (GOPs) for our proposed method while compressing video. Two distinct GOP configurations are employed: the bi-directional IPPP (bi-IPPP) and the normal IPPP structure (uni-IPPP) (Yang et al.,

2020b). Fig. 7.14 demonstrates the PSNR performance of our proposed approach utilizing GOP = 10, 13, and 20 for both bi-IPPP and uni-IPPP. greenFor bi-IPPP configuration, we follow the configuration from (Yang et al., 2020b). We observe that when the GOP size is increased to 20 (bi-IPPP), the performance remains comparable with that of GOP = 13 (bi-IPPP) and GOP = 10 (bi-IPPP), with just a modest impairment efficiency. greenIt happens when there is a greater gap between the I-frames and the P-frames in bi-IPPP. However, for GOP size 20, the PSNR is decreased by only around 2% from GOP 10 (roughly 38.8 vs. 37.9) and around 1.5% from GOP 13 (approximately 38.5 vs. 37.9). On the other hand, when we analyze the uni-IPPP in terms of GOP = 10, 13, and 20, the performance of the uni-IPPP mode is quite comparable over a range of GOP sizes of the bi-IPPP mode. For example, by around 0.4dB, 1.1dB, and 1.4dB from GOP 10 (bi-IPPP). To summarize, the proposed method is compatible with a wide range of GOP sizes. It is remarkably adaptable to GOP = 20 (bi-IPPP or uni-IPPP) without significantly reducing compression performance.

Table 7.11: The effectiveness of output feature channels with DRSA and without DRSA block for the encoder in MVCA network in terms of PSNR, MS-SSIM, inference time, and a number of parameters on UVG (Mercat et al., 2020) dataset.

Feature Channel (FC)	32	64	128	256	512	1024	2048
PSNR (dB)	33.15	34.11	35.25	36.18	36.32	36.32	36.36
MS-SSIM	0.931	0.943	0.956	0.969	0.971	0.972	0.972
Inference time (ms)	25	27	29	34	40	55	78
Num. of parameters (M)	1.14	2.29	3.22	4.21	5.67	8.99	11.02
PSNR (dB)	35.50	36.22	37.78	38.06	38.80	38.80	38.81
MS-SSIM	0.951	0.962	0.978	0.989	0.991	0.991	0.991
Inference time (ms)	22	23	24	27	31	50	71
Num. of parameters (M)	0.89	2.03	3.01	3.99	4.97	7.85	9.79

Effectiveness of output feature channels with DRSA

Since our proposed autoencoder network downsamples frames at the encoder and then upsamples them in decoded frames, it must make predictions quickly enough to operate at the frame rate of the video on a low-power edge device. To assess the feasibility of our

model, we measure the end-to-end inference time per frame as a function of the number of output feature channels with DRSA and without DRSA in the autoencoder network for the encoder, as shown in TABLE 7.11. The upper part represents the result without DRSA and the lower part with DRSA. The PSNR, MS-SSIM, inference time and parameters are raised by increasing the feature channel (FC). However, the inference time and parameters significantly increase at FC of 512, while PSNR and MS-SSIM improve slightly for both parts of the table. For example, for FC of 1024 and 2048, the parameters are increased by around 37% (5.67M vs. 8.99M) and 49% (5.67M vs. 11.02M) while expanding the inference time by roughly 27% (40ms vs. 55ms) and 49% (40ms vs. 78ms), respectively in terms of the upper parts. The PSNR and MS-SSIM are improving slightly for the feature channels of 1024 and 2048. Therefore, we build the autoencoder model up to 512 output feature channels, assuring that inference on a single frame takes just 40 ms for encoding. However, in this case, we proposed the DRSA in an autoencoder network to reduce the computational cost further. Along with the DRSA up to 512 feature channels (lower part of the table), the parameters are minimized by approximately 12% (5.67M vs. 4.97M), and the inference time is reduced by roughly 23% (40ms vs. 31ms). Besides, the PSNR and MS-SSIM are also improved by around 6% (36.32 vs. 38.80) and 2% (0.971 vs. 0.991), respectively. Thus, we can conclude that the effectiveness of our proposed DRSA in the MVC module is unparalleled.

7.10.5. Limitations

Our proposed approach has not investigated the entropy model and residual compression module. For entropy coding, (Ballé et al., 2018) is applied in our experiment. As a result, the overall coding speed is 1.9s (0.712s for the entropy coding). In addition, the current traditional-based approach H.266/VTM (Bross et al., 2019) surpassed the recent deep learning-based methods. Therefore, the entropy model needs further investigation in the

future in order to reduce our overall coding speed, and a performance comparison using H.266/VTM should be carried out.

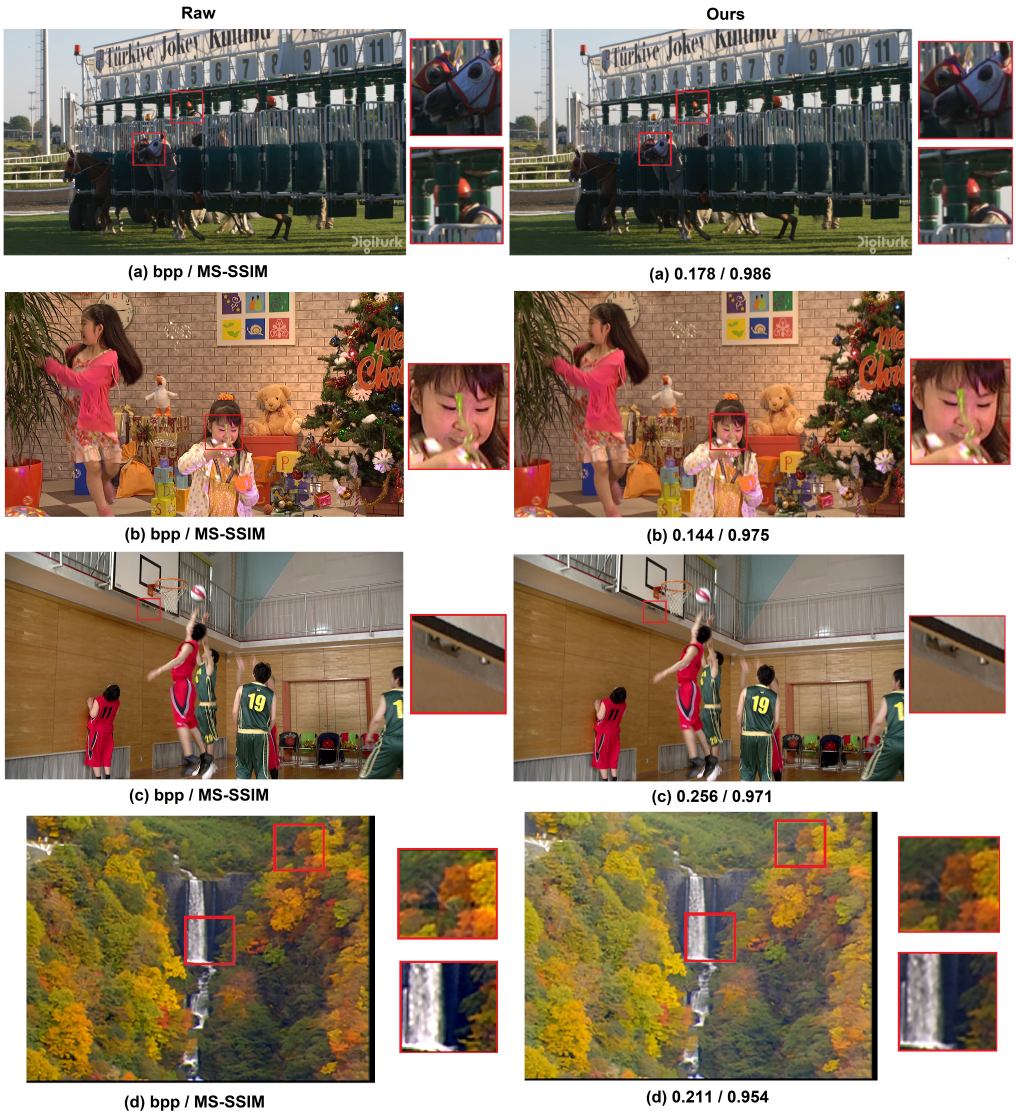

Figure 7.7: Qualitative comparison between the original and our reconstruction frames on the UVG (Mercat et al., 2020) (a), HEVC (Sullivan et al., 2012) (b), MCL-JCV (Wang et al., 2016) (c), and VTL (Hu et al., 2021) (d) datasets.

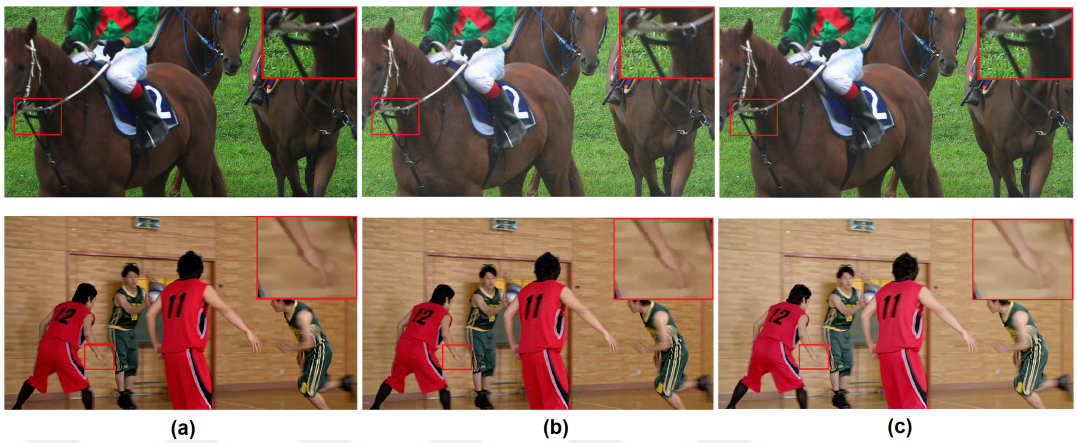

Figure 7.8: The visualization of the reconstruction frame with the proposed RSwCU. (a) represents the original frames derived from the HEVC Class C and D dataset (Sullivan et al., 2012), (b) the artifact frames after the concatenation of the predicted and residual frame, and (c) the final reconstructed frame by applying the ACM module with RSwCU.

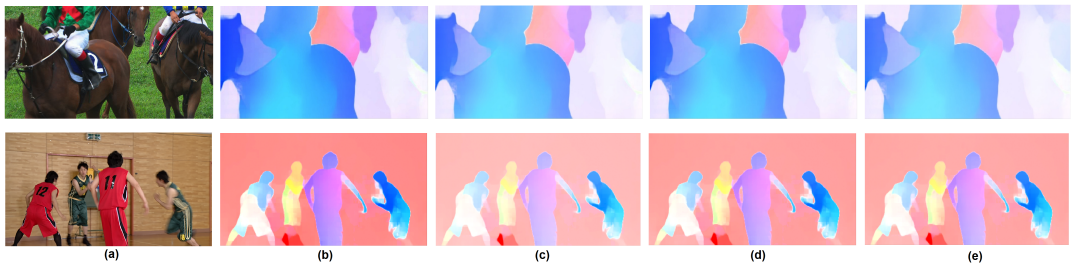

Figure 7.9: The visualization results of the motion compression using our proposed autoencoder-style network. (a), (b), (c), (d), and (e) represent the raw images from HEVC Class C and D datasets, the raw optical flow from the ME module using DeepPyNet (Jeny et al., 2021), visualization using 1 DRSA, 2 DRSA, and lastly, the visualization of the compressed optical flow/motion vector, respectively.

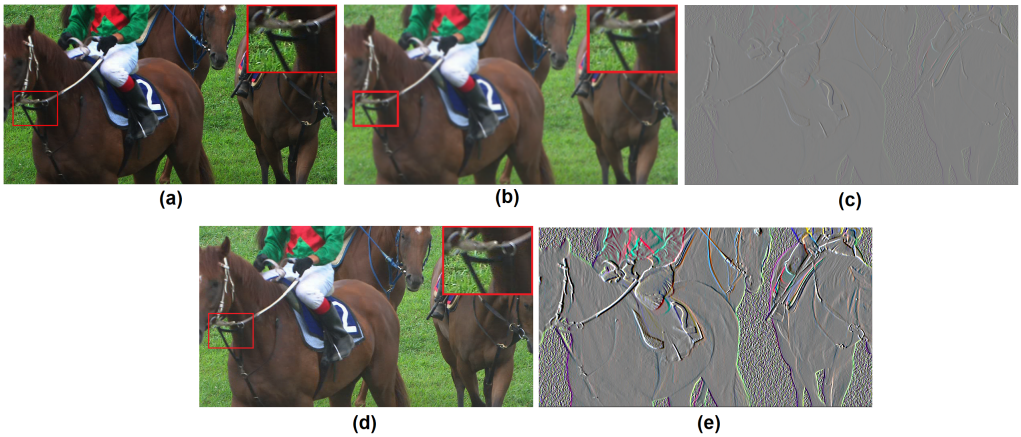

Figure 7.10: The visualization of the predicted frame from the MC module utilizing the CRB block. (a) represents the raw image, (b) and (c) the predicted frame and the residual frame when there is no CRB, and (d) and (e) the predicted frame and the residual frame when there is CRB, respectively.

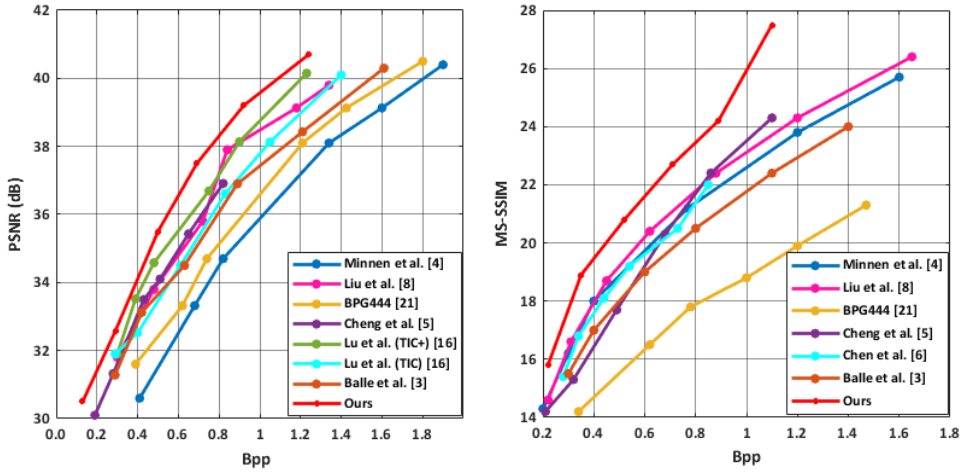

Figure 7.11: Quantitative performance comparison of the PSNR (left) and MS-SSIM (right) with state-of-the-art methods (Minnen et al., 2018) (Liu et al., 2020) (Bellard,) (Cheng et al., 2020) (Lu et al., 2021) (Ballé et al., 2018) (Chen et al., 2021).

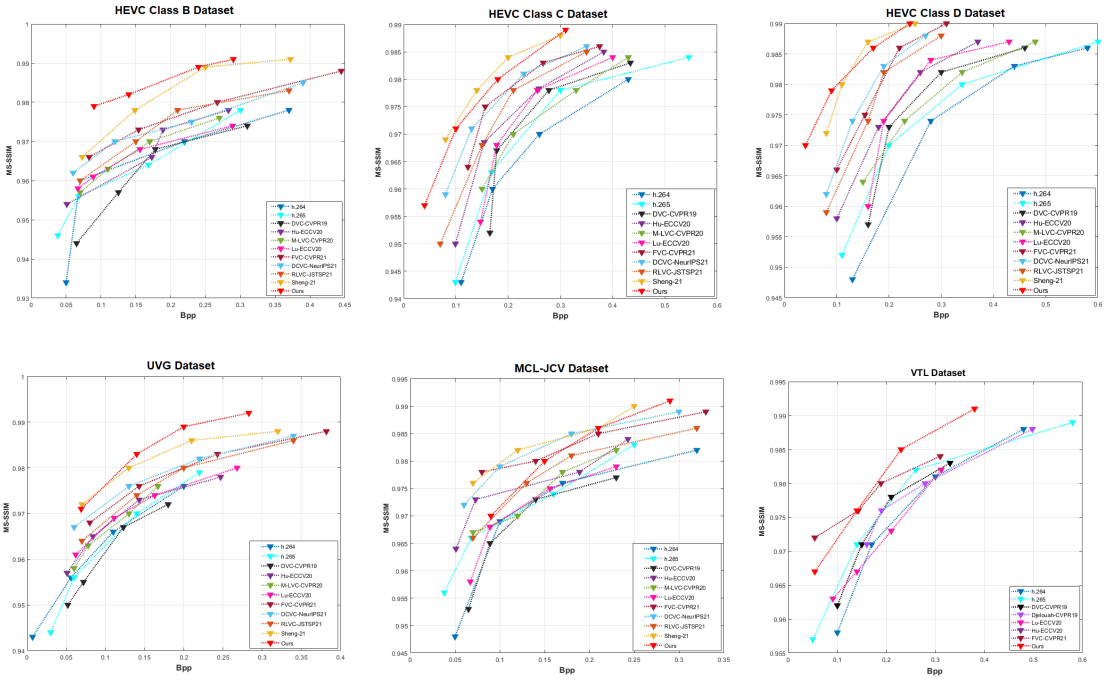

Figure 7.12: The MS-SSIM comparison of our method with the state-of-the-art learning-based methods, e.g., DVC-CVPR19 (Lu et al., 2019a), Djelouah-CVPR19 et al. (Djelouah et al., 2019), Hu-ECCV20 et al. (Hu et al., 2020), M-LVC-CVPR20 (Lin et al., 2020), Lu-ECCV20 et al. (Lu et al., 2020), FVC-CVPR21 (Hu et al., 2021), DCVC-NeurIPS21 (Li et al., 2021b), RLVC-JSTSP21 (Yang et al., 2020a), Sheng-21 (Sheng et al., 2021) and conventional methods, H.264/H.265 (Wiegand et al., 2003) (Sullivan et al., 2012) on HEVC Test Sequences (Class B, C, and D) (Sullivan et al., 2012), UVG (Mercat et al., 2020), MCL-JCV (Wang et al., 2016), and VTL (Hu et al., 2021) datasets.

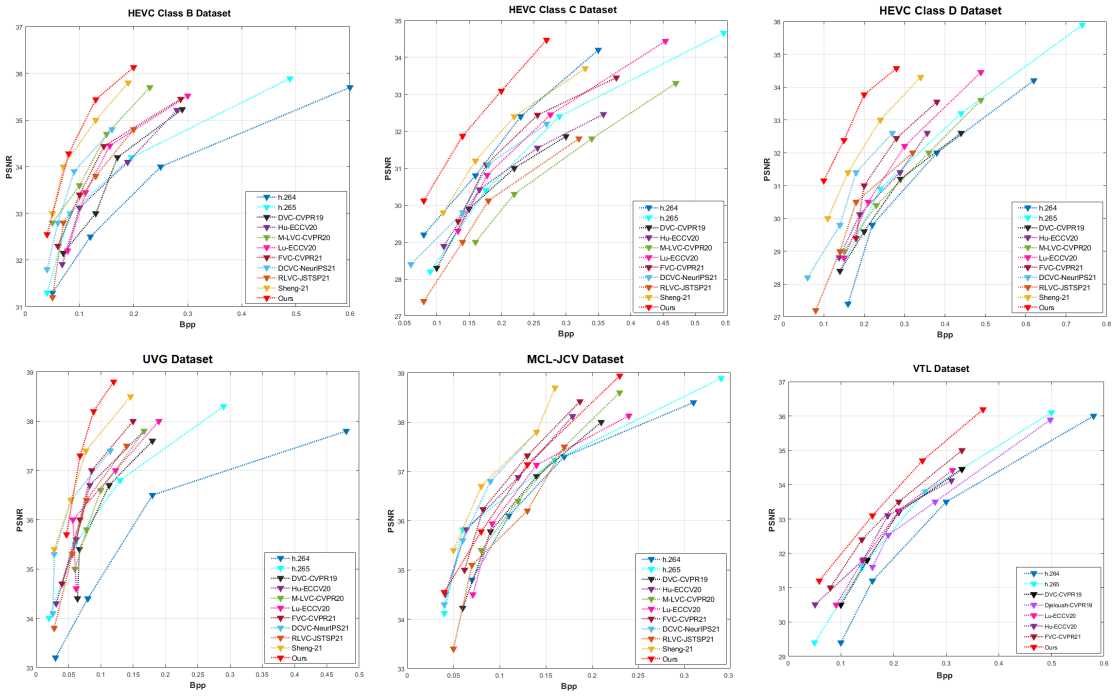

Figure 7.13: The PSNR comparison of our method with the state-of-the-art learning-based methods, e.g., DVC-CVPR19 (Lu et al., 2019a), Djelouah-CVPR19 et al. (Djelouah et al., 2019), Hu-ECCV20 et al. (Hu et al., 2020), M-LVC-CVPR20 (Lin et al., 2020), Lu-ECCV20 et al. (Lu et al., 2020), FVC-CVPR21 (Hu et al., 2021), DCVC-NeurIPS21 (Liu et al., 2021a), RLVC-JSTSP21 (Yang et al., 2020a), Sheng-21 (Sheng et al., 2021) and conventional methods, H.264/H.265 (Wiegand et al., 2003) (Sullivan et al., 2012) on HEVC Test Sequences (Class B, C, and D) (Sullivan et al., 2012), UVG (Mercat et al., 2020), MCL-JCV (Wang et al., 2016), and VTL (Hu et al., 2021) datasets.

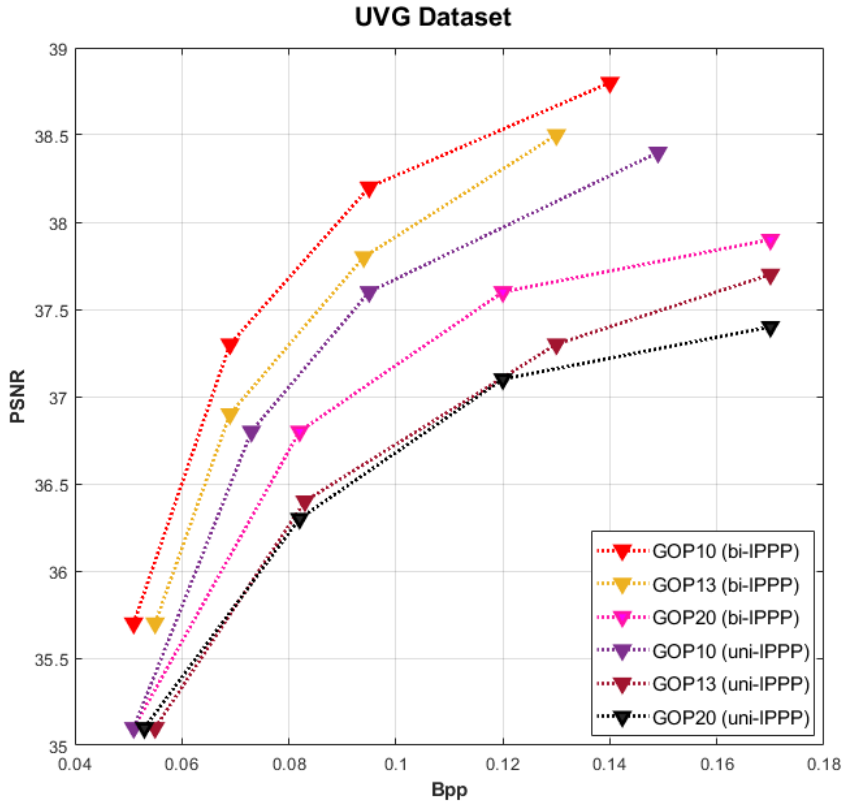

Figure 7.14: The effectiveness of a variety of GOP configurations on the UVG (Mercat et al., 2020) dataset.

8. CONCLUSION AND FUTURE WORK

Learned image compression techniques outperformed hand-crafted codecs. Despite its successes, it is not ideal for maintaining local redundancy, especially non-repetitive patterns affecting reconstruction quality. Video compression research has advanced in recent years. Existing learning-based algorithms are hampered by erroneous motion compression and inadequate motion correction structures, resulting in reduced rate-distortion compression mistakes. This thesis offers an autoencoder-style network-based efficient image compression approach with three innovative blocks: nearby attention block, Gaussian merging block, and decoded picture refining block. The neighboring attention block allocates extra bits to capture vertical and horizontal spatial correlations and delete useless data. Gaussian merging block helps rate-distortion optimization, whereas decoded image refinement block enhances low-resolution reconstructed pictures. An ablation study tests the model's qualitative and quantitative capabilities. Our technique beats state-of-the-art algorithms on the KODAK dataset (by 4dB and 5dB) and CLIC dataset (by 4dB and 3dB) in PSNR and MS-SSIM. This research compares CNN with a transformer-based approach. To tackle these issues, we provide an end-to-end video compression approach using several main operations (motion estimation, motion compression, motion compensation, residual compression, and artifact contraction). A flow-based paradigm is presented to improve video compression. Motion compression networks use a deep residual attention split (DRAS) block to pay more attention to particular picture areas, creating more effective decoder features and enhancing RDO efficiency. Motion compensation proposes a channel residual block (CRB) to enhance the residual frame with a more accurate forecast frame. Artifact contraction module (ACM) using residual swin convolution UNet block improves reconstruction quality by minimizing compression faults. A buffer improves the final frame by fine-tuning the prior reference frames. These modules com-

bine with a loss function to optimize video decoding. A detailed ablation investigation proves the blocks and modules' video compression efficiency. The suggested technique performs well on four benchmark datasets. In the future, an entropy model will be investigated for the both image and video compression in order to reduce the computational cost.

APPENDIX

PUBLICATIONS

1. **Afsana Ahsan Jeny**, Md Baharul Islam, Masum Shah Junayed, and Debashish Das, "**Improving Image Compression with Adjacent Attention and Refinement Block**", IEEE Access, IEEE. DOI: 10.1109/ACCESS.2022.3195295.

2. **Afsana Ahsan Jeny**, Md Baharul Islam, and Tarkan Aydin, "**DeepPyNet: A Deep Feature Pyramid Network for Optical Flow Estimation**", 36th International Conference on Image and Vision Computing New Zealand (IVCNZ-2021), Tauranga, New Zealand. 10.1109/IVCNZ54163.2021.9653193.

3. **Afsana Ahsan Jeny**, Masum Shah Junayed, and Md Baharul Islam, "**An Efficient End-to-End Image Compression Transformer**", The 29th IEEE International Conference on Image Processing (IEEE ICIP 2022), Bordeaux, France. (**Accepted**).

Bibliography

Clic · challenge on learned image compression. Available online: http://compression.cc/tasks/ evaluation (accessed on 15 December 2021).

Kodak image dataset. Available online: http://r0k.us/graphics/kodak/ evaluation (accessed on 15 December 2021).

Vtm10.0. vtm reference software for vvc. Available online: https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/-/tree/VTM-10.0 (accessed on 12 March 2022).

Workshop and challenge on learned image compression, 2020. Available online: https://www.compression.cc/ (accessed on 27 February 2022).

Agustsson, E., Mentzer, F., Tschannen, M., Cavigelli, L., Timofte, R., Benini, L., and Gool, L. V. (2017). Soft-to-hard vector quantization for end-to-end learning compressible representations. *Advances in neural information processing systems*, 30.

Agustsson, E., Minnen, D., Johnston, N., Balle, J., Hwang, S. J., and Toderici, G. (2020). Scale-space flow for end-to-end optimized video compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8503–8512.

Agustsson, E., Tschannen, M., Mentzer, F., Timofte, R., and Gool, L. V. (2019). Generative adversarial networks for extreme learned image compression. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 221–231.

Alhersh, T., Belhaouari, S. B., and Stuckenschmidt, H. (2020). Metrics performance analysis of optical flow. In *VISIGRAPP (4: VISAPP)*, pages 749–758.

Bai, Y., Yang, X., Liu, X., Jiang, J., Wang, Y., Ji, X., and Gao, W. (2021). Towards end-to-end image compression and analysis with transformers. *arXiv preprint arXiv:2112.09300*.

Bailer, C., Taetz, B., and Stricker, D. (2015). Flow fields: Dense correspondence fields for highly accurate large displacement optical flow estimation. In *Proceedings of the IEEE international conference on computer vision*, pages 4015–4023.

Ballé, J., Laparra, V., and Simoncelli, E. P. (2015). Density modeling of images using a generalized normalization transformation. *arXiv preprint arXiv:1511.06281*.

Ballé, J., Laparra, V., and Simoncelli, E. P. (2016). End-to-end optimized image compression. *arXiv preprint arXiv:1611.01704*.

Ballé, J., Minnen, D., Singh, S., Hwang, S. J., and Johnston, N. (2018). Variational image compression with a scale hyperprior. *arXiv preprint arXiv:1802.01436*.

Bar-Haim, A. and Wolf, L. (2020). Scopeflow: Dynamic scene scoping for optical flow. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7998–8007.

Bégaint, J., Racapé, F., Feltman, S., and Pushparaja, A. (2020). Compressai: a pytorch library and evaluation platform for end-to-end compression research. *arXiv preprint arXiv:2011.03029*.

Bellard, F. Bpg image format. Available online: https://bellard.org/bpg/ evaluation (accessed on 15 January 2022).

Bjontegaard, G. (2001). Calculation of average psnr differences between rd-curves. *VCEG-M33*.

Bross, B., Chen, J., Liu, S., and Wang, Y.-K. (2019). Versatile video coding (draft 5). *Joint Video Experts Team (JVET) of ITU-T SG*, 16:3–12.

Butler, D. J., Wulff, J., Stanley, G. B., and Black, M. J. (2012). A naturalistic open source movie for optical flow evaluation. In *European conference on computer vision*, pages 611–625. Springer.

Chen, Q. and Koltun, V. (2016). Full flow: Optical flow estimation by global optimization over regular grids. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4706–4714.

Chen, T., Liu, H., Ma, Z., Shen, Q., Cao, X., and Wang, Y. (2021). End-to-end learnt image compression via non-local attention optimization and improved context modeling. *IEEE Transactions on Image Processing*, 30:3179–3191.

Chen, T., Liu, H., Shen, Q., Yue, T., Cao, X., and Ma, Z. (2017). Deepcoder: A deep neural network based video compression. In *2017 IEEE Visual Communications and Image Processing (VCIP)*, pages 1–4. IEEE.

Chen, Z., He, T., Jin, X., and Wu, F. (2019). Learning for video compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(2):566–576.

Cheng, Z., Sun, H., Takeuchi, M., and Katto, J. (2018). Performance comparison of convolutional autoencoders, generative adversarial networks and super-resolution for image compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 2613–2616.

Cheng, Z., Sun, H., Takeuchi, M., and Katto, J. (2019). Deep residual learning for image compression. In *CVPR Workshops*, page 0.

Cheng, Z., Sun, H., Takeuchi, M., and Katto, J. (2020). Learned image compression with discretized gaussian mixture likelihoods and attention modules. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7939–7948.

Djelouah, A., Campos, J., Schaub-Meyer, S., and Schroers, C. (2019). Neural inter-frame compression for video coding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6421–6429.

Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Van Der Smagt, P., Cremers, D., and Brox, T. (2015). Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766.

Eldesokey, A. and Felsberg, M. (2021). Normalized convolution upsampling for refined optical flow estimation. *arXiv preprint arXiv:2102.06979*.

Feng, R., Wu, Y., Guo, Z., Zhang, Z., and Chen, Z. (2020). Learned video compression with feature-level residuals. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 120–121.

Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237.

Guo, Z., Fu, J., Feng, R., and Chen, Z. (2021). Accelerate neural image compression

with channel-adaptive arithmetic coding. In *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. IEEE.

Guo, Z., Wu, Y., Feng, R., Zhang, Z., and Chen, Z. (2020). 3-d context entropy model for improved practical image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 116–117.

Habibian, A., Rozendaal, T. v., Tomczak, J. M., and Cohen, T. S. (2019). Video compression with rate-distortion autoencoders. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7033–7042.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hore, A. and Ziou, D. (2010). Image quality metrics: Psnr vs. ssim. In *2010 20th international conference on pattern recognition*, pages 2366–2369. IEEE.

Hu, J., Shen, L., and Sun, G. (2018). Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141.

Hu, Z., Chen, Z., Xu, D., Lu, G., Ouyang, W., and Gu, S. (2020). Improving deep video compression by resolution-adaptive flow coding. In *European Conference on Computer Vision*, pages 193–209. Springer.

Hu, Z., Lu, G., and Xu, D. (2021). Fvc: A new framework towards deep video compression in feature space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1502–1511.

Hui, T.-W. and Loy, C. C. (2020). Liteflownet3: Resolving correspondence ambiguity for

more accurate optical flow estimation. In *European Conference on Computer Vision*, pages 169–184. Springer.

Hui, T.-W., Tang, X., and Loy, C. C. (2018). Liteflownet: A lightweight convolutional neural network for optical flow estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8981–8989.

Hui, T.-W., Tang, X., and Loy, C. C. (2020). A lightweight optical flow cnn—revisiting data fidelity and regularization. *IEEE transactions on pattern analysis and machine intelligence*, 43(8):2555–2569.

Hur, J. and Roth, S. (2019). Iterative residual refinement for joint optical flow and occlusion estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5754–5763.

Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., and Brox, T. (2017). Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2462–2470.

Jaderberg, M., Simonyan, K., Zisserman, A., et al. (2015). Spatial transformer networks. *Advances in neural information processing systems*, 28:2017–2025.

Jain, S. and Wallace, B. C. (2019). Attention is not explanation. *arXiv preprint arXiv:1902.10186*.

Jayasundara, V., Roy, D., and Fernando, B. (2021). Flowcaps: Optical flow estimation with capsule networks for action recognition. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3409–3418.

Jeny, A. A., Islam, M. B., and Aydin, T. (2021). Deeppynet: A deep feature pyramid

network for optical flow estimation. In *2021 36th International Conference on Image and Vision Computing New Zealand (IVCNZ)*, pages 1–6. IEEE.

Johnston, N., Vincent, D., Minnen, D., Covell, M., Singh, S., Chinen, T., Hwang, S. J., Shor, J., and Toderici, G. (2018). Improved lossy image compression with priming and spatially adaptive bit rates for recurrent networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4385–4393.

Kim, Y., Soh, J. W., and Cho, N. I. (2020). Agarnet: adaptively gated jpeg compression artifacts removal network for a wide range quality factor. *IEEE Access*, 8:20160–20170.

Kin, C. Y. S. and Coker, B. (2017). Video compression using recurrent convolutional neural networks.

Kingma, D. P. and Adam, J. B. (2015). A method for stochastic. *Optimization. In, ICLR*, 5.

Kodak, E. (1993). Kodak lossless true color image suite (photocd pcd0992). *URL http://r0k. us/graphics/kodak*, 6.

Kong, L., Yang, X., and Yang, J. (2021). Oas-net: Occlusion aware sampling network for accurate optical flow. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2475–2479. IEEE.

Krasin, I., Duerig, T., Alldrin, N., Ferrari, V., Abu-El-Haija, S., Kuznetsova, A., Rom, H., Uijlings, J., Popov, S., Veit, A., et al. (2017). Openimages: A public dataset for large-scale multi-label and multi-class image classification. *Dataset available from https://github. com/openimages*, 2(3):18.

Lee, J., Cho, S., and Beack, S.-K. (2018). Context-adaptive entropy model for end-to-end optimized image compression. *arXiv preprint arXiv:1809.10452*.

Li, B., Ren, H., Jiang, X., Miao, F., Feng, F., and Jin, L. (2021a). Scep—a new image dimensional emotion recognition model based on spatial and channel-wise attention mechanisms. *IEEE Access*, 9:25278–25290.

Li, J., Li, B., and Lu, Y. (2021b). Deep contextual video compression. *Advances in Neural Information Processing Systems*, 34.

Li, M., Zuo, W., Gu, S., Zhao, D., and Zhang, D. (2018). Learning convolutional networks for content-weighted image compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3214–3223.

Li, W., Sun, W., Zhao, Y., Yuan, Z., and Liu, Y. (2020). Deep image compression with residual learning. *Applied Sciences*, 10(11):4023.

Li, Y., Hu, Y., Song, R., Rao, P., and Wang, Y. (2017). Coarse-to-fine patchmatch for dense correspondence. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(9):2233–2245.

Lim, B., Son, S., Kim, H., Nah, S., and Mu Lee, K. (2017). Enhanced deep residual networks for single image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 136–144.

Lim, H. and Park, H. (2011). A ringing-artifact reduction method for block-dct-based image resizing. *IEEE transactions on circuits and systems for video technology*, 21(7):879–889.

Lin, J., Liu, D., Li, H., and Wu, F. (2020). M-lvc: Multiple frames prediction for learned

video compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3546–3554.

Liu, B., Chen, Y., Liu, S., and Kim, H.-S. (2021a). Deep learning in latent space for video prediction and compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 701–710.

Liu, H., Chen, T., Guo, P., Shen, Q., Cao, X., Wang, Y., and Ma, Z. (2019). Non-local attention optimized deep image compression. *arXiv preprint arXiv:1904.09757*.

Liu, J., Lu, G., Hu, Z., and Xu, D. (2020). A unified end-to-end framework for efficient deep image compression. *arXiv preprint arXiv:2002.03370*.

Liu, X., An, P., Chen, Y., and Huang, X. (2022). An improved lossless image compression algorithm based on huffman coding. *Multimedia Tools and Applications*, 81(4):4781–4795.

Liu, Y., Li, H., and Wang, X. (2017). Learning deep features via congenerous cosine loss for person recognition. *arXiv preprint arXiv:1702.06890*.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. (2021b). Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022.

Liu, Z., Yu, X., Gao, Y., Chen, S., Ji, X., and Wang, D. (2016). Cu partition mode decision for hevc hardwired intra encoder using convolution neural network. *IEEE Transactions on Image Processing*, 25(11):5088–5103.

Loshchilov, I. and Hutter, F. (2017). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Lu, G., Cai, C., Zhang, X., Chen, L., Ouyang, W., Xu, D., and Gao, Z. (2020). Content adaptive and error propagation aware deep video compression. In *European Conference on Computer Vision*, pages 456–472. Springer.

Lu, G., Ouyang, W., Xu, D., Zhang, X., Cai, C., and Gao, Z. (2019a). Dvc: An end-to-end deep video compression framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11006–11015.

Lu, M., Guo, P., Shi, H., Cao, C., and Ma, Z. (2021). Transformer-based image compression. *arXiv preprint arXiv:2111.06707*.

Lu, X., Wang, H., Dong, W., Wu, F., Zheng, Z., and Shi, G. (2019b). Learning a deep vector quantization network for image compression. *IEEE Access*, 7:118815–118825.

Mayer, N., Ilg, E., Hausser, P., Fischer, P., Cremers, D., Dosovitskiy, A., and Brox, T. (2016). A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4040–4048.

Mentzer, F., Agustsson, E., Tschannen, M., Timofte, R., and Van Gool, L. (2018). Conditional probability models for deep image compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4394–4402.

Mercat, A., Viitanen, M., and Vanne, J. (2020). Uvg dataset: 50/120fps 4k sequences for video codec analysis and development. In *Proceedings of the 11th ACM Multimedia Systems Conference*, pages 297–302.

Minnen, D., Ballé, J., and Toderici, G. D. (2018). Joint autoregressive and hierarchical priors for learned image compression. *Advances in neural information processing systems*, 31.

Monga, V. and Evans, B. L. (2006). Perceptual image hashing via feature points: performance evaluation and tradeoffs. *IEEE transactions on Image Processing*, 15(11):3452–3465.

Pan, Z., Zhuang, B., He, H., Liu, J., and Cai, J. (2021). Less is more: Pay less attention in vision transformers. *arXiv preprint arXiv:2105.14217*.

Pham, T. T., Van Hoang, X., Nguyen, N. T., Dinh, D. T., et al. (2020). End-to-end image patch quality assessment for image/video with compression artifacts. *IEEE Access*, 8:215157–215172.

Ranjan, A. and Black, M. J. (2017). Optical flow estimation using a spatial pyramid network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4161–4170.

Ranjan, R. (2021). Canonical huffman coding based image compression using wavelet. *Wireless Personal Communications*, 117(3):2193–2206.

Rippel, O. and Bourdev, L. (2017). Real-time adaptive image compression. In *International Conference on Machine Learning*, pages 2922–2930. PMLR.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

Schuster, R., Bailer, C., Wasenmüller, O., and Stricker, D. (2018). Flowfields++: Ac-

curate optical flow correspondences meet robust interpolation. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 1463–1467. IEEE.

Sheng, X., Li, J., Li, B., Li, L., Liu, D., and Lu, Y. (2021). Temporal context mining for learned video compression. *arXiv preprint arXiv:2111.13850*.

Skodras, A., Christopoulos, C., and Ebrahimi, T. (2001). The jpeg 2000 still image compression standard. *IEEE Signal processing magazine*, 18(5):36–58.

Song, Q., Li, J., Guo, H., and Huang, R. (2021). Denoised non-local neural network for semantic segmentation. *arXiv preprint arXiv:2110.14200*.

Song, R., Liu, D., Li, H., and Wu, F. (2017). Neural network-based arithmetic coding of intra prediction modes in hevc. In *2017 IEEE Visual Communications and Image Processing (VCIP)*, pages 1–4. IEEE.

Sullivan, G. J., Ohm, J.-R., Han, W.-J., and Wiegand, T. (2012). Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on circuits and systems for video technology*, 22(12):1649–1668.

Sun, D., Yang, X., Liu, M.-Y., and Kautz, J. (2018). Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8934–8943.

Sun, D., Yang, X., Liu, M.-Y., and Kautz, J. (2019). Models matter, so does training: An empirical study of cnns for optical flow estimation. *IEEE transactions on pattern analysis and machine intelligence*, 42(6):1408–1423.

Sun, J. and Tappen, M. F. (2011). Learning non-local range markov random field for image restoration. In *CVPR 2011*, pages 2745–2752. IEEE.

Theis, L., Shi, W., Cunningham, A., and Huszár, F. (2017). Lossy image compression with compressive autoencoders. *arXiv preprint arXiv:1703.00395*.

Toderici, G., O'Malley, S. M., Hwang, S. J., Vincent, D., Minnen, D., Baluja, S., Covell, M., and Sukthankar, R. (2015). Variable rate image compression with recurrent neural networks. *arXiv preprint arXiv:1511.06085*.

Toderici, G., Vincent, D., Johnston, N., Jin Hwang, S., Minnen, D., Shor, J., and Covell, M. (2017). Full resolution image compression with recurrent neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 5306–5314.

Tu, Z. (2015). *Variational Optical Flow Algorithms for Motion Estimation*. PhD thesis, University Utrecht.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Wallace, G. K. (1992). The jpeg still picture compression standard. *IEEE transactions on consumer electronics*, 38(1):xviii–xxxiv.

Wang, F., Jiang, M., Qian, C., Yang, S., Li, C., Zhang, H., Wang, X., and Tang, X. (2017). Residual attention network for image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164.

Wang, H., Gan, W., Hu, S., Lin, J. Y., Jin, L., Song, L., Wang, P., Katsavounidis, I., Aaron, A., and Kuo, C.-C. J. (2016). Mcl-jcv: a jnd-based h. 264/avc video quality

assessment dataset. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 1509–1513. IEEE.

Wang, X., Girshick, R., Gupta, A., and He, K. (2018). Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803.

Wang, Z., Simoncelli, E. P., and Bovik, A. C. (2003). Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. Ieee.

Wiegand, T., Sullivan, G. J., Bjontegaard, G., and Luthra, A. (2003). Overview of the h. 264/avc video coding standard. *IEEE Transactions on circuits and systems for video technology*, 13(7):560–576.

Witten, I. H., Neal, R. M., and Cleary, J. G. (1987). Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540.

Woo, S., Park, J., Lee, J.-Y., and Kweon, I. S. (2018). Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19.

Wu, C.-Y., Singhal, N., and Krahenbuhl, P. (2018a). Video compression through image interpolation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 416–431.

Wu, C.-Y., Zaheer, M., Hu, H., Manmatha, R., Smola, A. J., and Krähenbühl, P. (2018b). Compressed video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6026–6035.

Wu, H., Xiao, B., Codella, N., Liu, M., Dai, X., Yuan, L., and Zhang, L. (2021). Cvt: Introducing convolutions to vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22–31.

Wu, X., Zhang, Z., Feng, J., Zhou, L., and Wu, J. (2020). End-to-end optimized video compression with mv-residual prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 156–157.

Wulff, J., Sevilla-Lara, L., and Black, M. J. (2017). Optical flow in mostly rigid scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4671–4680.

Xu, J., Ranftl, R., and Koltun, V. (2017). Accurate optical flow via direct cost volume processing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1289–1297.

Xue, T., Chen, B., Wu, J., Wei, D., and Freeman, W. T. (2019). Video enhancement with task-oriented flow. *International Journal of Computer Vision*, 127(8):1106–1125.

Yan, H., Li, Z., Li, W., Wang, C., Wu, M., and Zhang, C. (2021). Contnet: Why not use convolution and transformer at the same time? *arXiv preprint arXiv:2104.13497*.

Yang, G. and Ramanan, D. (2019). Volumetric correspondence networks for optical flow. *Advances in neural information processing systems*, 32:794–805.

Yang, R., Mentzer, F., Gool, L. V., and Timofte, R. (2020a). Learning for video compression with hierarchical quality and recurrent enhancement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6628–6637.

Yang, R., Mentzer, F., Van Gool, L., and Timofte, R. (2020b). Learning for video compression with recurrent auto-encoder and recurrent probability model. *IEEE Journal of Selected Topics in Signal Processing*, 15(2):388–401.

Yang, Y. and Soatto, S. (2017). S2f: Slow-to-fast interpolator flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2087–2096.

Yin, Z., Darrell, T., and Yu, F. (2019). Hierarchical discrete distribution decomposition for match density estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6044–6053.

Zhang, H., Wu, C., Zhang, Z., Zhu, Y., Lin, H., Zhang, Z., Sun, Y., He, T., Mueller, J., Manmatha, R., et al. (2020). Resnest: Split-attention networks. *arXiv preprint arXiv:2004.08955*.

Zhang, Y., Li, K., Li, K., Zhong, B., and Fu, Y. (2019). Residual non-local attention networks for image restoration. *arXiv preprint arXiv:1903.10082*.

Zhao, S., Sheng, Y., Dong, Y., Chang, E. I., Xu, Y., et al. (2020). Maskflownet: Asymmetric feature matching with learnable occlusion mask. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6278–6287.

Zhou, L., Sun, Z., Wu, X., and Wu, J. (2019). End-to-end optimized image compression with attention mechanism. In *CVPR workshops*, page 0.

Zhu, X., Cheng, D., Zhang, Z., Lin, S., and Dai, J. (2019). An empirical study of spatial attention mechanisms in deep networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6688–6697.