Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Codificador de vídeo baseado em autoencoder variacional: uma análise de desempenho frente aos codificadores tradicionais HEVC e VVC

Luiz Gustavo Rodrigues Martins

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Orientador
Prof. Dr. Edson Mintsu Hung

Brasília
2021

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Codificador de vídeo baseado em autoencoder variacional: uma análise de desempenho frente aos codificadores tradicionais HEVC e VVC

Luiz Gustavo Rodrigues Martins

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Prof. Dr. Edson Mintsu Hung (Orientador)
ENE/UnB

Prof. Dr. Eduardo Peixoto F. da Silva
ENE/UnB

Prof. Dr. Renam Castro da Silva
IFSP

Prof. Dr. João José Costa Gondim
Coordenador do Curso de Engenharia da Computação

Brasília, 16 de novembro de 2021

Dedicatória

À memória de meu avô José, o qual sempre incentivou a busca pelo conhecimento, um bem compartilhado entre seres humanos e que estará junto a nós durante toda a vida e, quem sabe, além.

Agradecimentos

Agradeço à Universidade de Brasília por proporcionar uma educação pública de elevada qualidade a um número incontável de estudantes do qual tenho orgulho de fazer parte.

Aos professores que fizeram parte da minha formação durante essa etapa da graduação em Engenharia de Computação. Agradecimento especial ao meu orientador Mintsu por sua enorme paciência e sinceridade me guiando durante essa caminhada e, também, pela sua nobre atitude em prol de atividades voluntárias. Ao professor Bruno, por ter me apresentado à área de processamento de imagens e vídeo e despertado meu interesse por querer aprender mais sobre. Agradeço também ao professor Eduardo pela parceria, juntamente aos professores Mintsu e Bruno, durante o projeto a partir do qual foi possível o desenvolvimento deste estudo.

Muito obrigado ao Nilson e Henrique pelos ensinamentos e disposição em ajudar, ao "maninho" Matheus pela troca de conhecimento e companheirismo, e aos colaboradores Renam Castro, Vanessa Testoni e Pedro Garcia.

Por fim, agradeço à minha família. Meus pais Marleide e Luiz que sempre me apoiaram, meus irmãos Jéssica e André pelos momentos de descontração, meu cunhado Tainá pela imensa ajuda em colocar minha cabeça e ideias em ordem para poder escrever, nossos *pets* Wiccy e Pandora pelos carinhos, e meu sobrinho e afilhado Nhanderú por iluminar e trazer a alegria de criança aos meus dias.

Resumo

Codificadores de vídeo com aprendizado de ponta-a-ponta estão sendo desenvolvidos com a proposta de superar limitações dos codificadores tradicionais, os quais possuem algoritmos manuais que se tornam cada vez mais complexos para ganhos de desempenho cada vez mais limitados. Este estudo tem por objetivo realizar a análise de desempenho de um codificador de vídeo, baseado em uma arquitetura de aprendizado profundo conhecida como *autoencoder* variacional, o qual é composto de duas redes: uma intra-quadros e outra inter-quadros. Essas redes, embora dependentes, são treinadas de forma separada. Para a definição dos parâmetros a serem utilizados nos treinamentos dos modelos finais do codificador, primeiramente são realizados treinamentos testes das redes. Nos treinamentos testes da rede intra-quadros, foram realizados três grupos de treinamentos: (*i*) treinamento para gerar modelos a serem utilizados no treinamento teste da rede inter-quadros, (*ii*) treinamento com diferentes quantidades de filtros para se definir o número de filtros para os treinamentos finais, e (*iii*) treinamento com bases de dados diferentes para definir qual base será utilizada no treinamento final da rede intra-quadros e seus valores para o multiplicador Lagrangiano da função objetivo, conhecido como *lambda*. Em (*ii*) foram definidos 256 filtros para o treinamento final das redes e em (*iii*) foi definida a utilização de um banco de dados com imagens em formato *.png* (*Portable Network Graphics*) e dois valores de *lambda*, 10^{-2} e 10^{-3}. O treinamento teste da rede inter-quadros utilizou os modelos do treinamento teste (*i*) da rede intra-quadros e gerou modelos com diferentes combinações entre parâmetros *lambda* intra e inter. Analisando o comportamento dos modelos obtidos, foram definidos três valores de lambda para os modelos inter finais: $5 \cdot 10^{-3}$, 10^{-3} e $5 \cdot 10^{-4}$. Dessa forma, foram gerados seis modelos, com as combinações possíveis entre os *lambdas* intra e inter-quadros. Todos os treinamentos inter-quadros utilizaram o *UGC Dataset*. Com os modelos finais obtidos, foram realizadas compressões de sequências de vídeo do banco de dados JVET. Com isso, foram calculadas métricas PSNR, SSIM e MS-SSIM para o canal de luminância dos resultados obtidos. As mesmas métricas foram obtidas para compressões realizadas pelos codificadores tradicionais HEVC e VVC. Com métricas tanto para o codificador de vídeo neural quanto para os codificadores tradicionais, foi realizada a comparação de desempenho entre esses codificadores. Com as

análises realizadas, concluiu-se que o desempenho do codificador de vídeo analisado neste estudo é comparável ao modo de configuração *All Intra* dos codificadores tradicionais HEVC e VVC, principalmente ao se considerar a métrica MS-SSIM.

Palavras-chave: Compressão, Vídeo, Rede neural, Aprendizado profundo, Inteligência artificial

Abstract

Learning-based end-to-end video encoders are being developed with the purpose of overcoming limitations of traditional encoders, which have manual algorithms that are becoming more and more complex while performance gains are increasingly limited. In the present study, a video encoder with a variational autoencoder architecture, a kind of deep learning architecture, composed by an intra and inter-frame networks is subjected to analysis. These networks, although dependent, are trained separately. For setting parameters to the final models training, some test training of the networks is carried out. In the intra-frame test trainings, three groups of training were set: (*i*) training for generating models to be used in the inter-frame test training, (*ii*) training with different number of filters for choosing the best one for the final training, and (*iii*) training with different databases to choose the one that will be used in the final training of the intra-frame models and choose its Lagrangian multiplier's values of the objective function, known as *lambda* values. In (*ii*) 256 filters were choosen for the final training of the networks and in (*iii*) the database with *.png* (*Portable Network Graphics*) images was choosen and the values of 10^{-2} and 10^{-3} were defined for intra-frame *lambda*. The test inter-frame training used the models of the initial training in (*i*) and generated models with different combinations between intra and inter *lambda* parameters. Three values were chosen for inter-frame *lambda*: $5 \cdot 10^{-3}$, 10^{-3} and $5 \cdot 10^{-4}$. Thus, six final models were trained using intra and inter-frame *lambdas* combination. All inter-frame trainings used the *UGC Dataset*. In possession of the final models, compressions of some JVET video sequences were performed. Thus, PSNR, SSIM and MS-SSIM metrics were calculated for the lumma channel of the results. These metrics were also calculated for compressions performed by the traditional encoders HEVC and VVC. With metrics for both the learning-based video encoder and the traditional encoders, the performance comparison between these encoders was performed. After analysis, it was concluded that the performance of the video encoder analyzed in this study presents performance comparable to the *All Intra* mode of the traditional HEVC and VVC encoders, especially when considering the MS-SSIM metric.

Keywords: Compression, Video, Neural network, Deep learning, Artificial intelligence

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

AVC *Advanced Video Coding.*

BPP *Bits* por pixel.

CIC Departamento de Ciência da Computação.

CNN *Convolutional Neural Networks.*

DNNVC *Deep Neural Network based Video Compression.*

FT Faculdade de Tecnologia.

FUB Fundação Universidade de Brasília.

GOP *Group of Pictures.*

GPU *Graphics Processing Unit.*

HEVC *High Efficiency Video Coding.*

IEC *International Electrotechnical Commission.*

ISO *International Organization of Standardization.*

ITU-T *International Telecommunication Union Telecommunication Standardization Sector.*

JTC *Joint Technical Committee.*

JVET *Joint Video Experts Team.*

LISA Laboratório de Imagens, Sinais e Acústica.

MPEG *Moving Picture Experts Group.*

MS-SSIM *Multi-Scale Structural Similarity Index Measure.*

MSE *Mean Squared Error.*

PNG *Portable Network Graphics.*

PSNR *Peak signal-to-noise ratio.*

RGB *Red, Green, Blue.*

SC *Subcommittee.*

SSIM *Structural Similarity Index Measure.*

UnB Universidade de Brasília.

VVC *Versatile Video Coding.*

WG *Working Group.*

YUV *Luminance-Bandwidth-Chrominance.*

Capítulo 1

Introdução

1.1 Contexto

Formas de comunicação remota com qualidade se tornam cada vez mais necessárias e urgentes em uma realidade afetada por questões de saúde devido à pandemia COVID-19.

Serviços de vídeoconferência para reuniões e de *streaming* para consumo de conteúdo multimídia são cada vez mais requisitados. A qualidade da informação transmitida é importante para que a comunicação ocorra com qualidade e a experiência não seja afetada. Tudo isso remete à necessidade de se utilizarem técnicas de compressão eficientes, já que a capacidade de armazenamento de informação e de transmissão desses dados cresce de forma mais lenta que a demanda.

Codificadores tradicionais como H.264[Wiegand et al., 2003], H.265[Sullivan et al., 2012] e H.266[Bross et al., 2021] possuem abordagem híbrida com codificação baseada em transformadas, resíduos, e técnicas de compensação de movimento e de intra-predição. Essas técnicas são baseadas em algoritmos manuais.[de Oliveira et al., 2021]

Segundo [de Oliveira et al., 2021], nas últimas décadas, as técnicas tradicionais desses codificadores aumentaram o desempenho da compressão de vídeo em torno de 50% a cada 10 anos, com um custo cada vez maior de complexidade computacional, memória e consumo energético. Segundo [Seidel, 2020], há uma tendência de que cada novo padrão de codificação fique dez vezes mais complexo enquanto sua eficiência aumente somente duas vezes.

Frente as limitações dos codificadores tradicionais, técnicas de aprendizagem de máquina vem sendo empregadas na implementação de soluções na área de compressão de vídeo [Chen et al., 2020]. Essas soluções podem envolver codificadores completos totalmente baseados em aprendizado ou ferramentas baseadas em aprendizado que substituem ou auxiliam determinado módulo já existente em um codificador de mercado como o HEVC ou VVC.

O presente estudo envolve uma análise de desempenho de um codificador de vídeo baseado em *autoencoder* variacional. A proposta desse codificador é apresentada em [de Oliveira et al., 2021]. O codificador em questão foi desenvolvido pela equipe do projeto SAMSUNG/FUB/FT - Codificação de imagens e vídeos baseados em inteligência artificial para padrões futuros (*Deep Codec project*). Portanto, trata-se de uma implementação funcional baseada em aprendizado, o que demonstra a possibilidade de utilização de redes neurais em atividades de compressão de vídeo.

1.2 Problemática

Com uma demanda crescente pela utilização e consumo de conteúdo em forma de vídeo, técnicas de aprendizado profundo (*deep learning*, em inglês) surgem como alternativas para implementação de codificadores com aprendizado de ponta-a-ponta (*end-to-end*, em inglês) que atinjam taxas de compressão satisfatórias.[Ding et al., 2021]

Codificadores tradicionais exploram tanto correlação espacial quanto temporal. Porém, diante da complexidade e restrições computacionais, geralmente esses se limitam a transformações lineares e estimação de movimento translacional. Os codificadores neurais baseados em modelos *autoencoder* surgem como uma possível alternativa para lidar com essas limitações.

1.3 Objetivo

O objetivo geral do presente estudo é comparar o desempenho do codificador desenvolvido pela equipe do projeto, o qual é baseado em um modelo de *autoencoder* variacional, com o desempenho dos codificadores tradicionais mais atuais: *High Efficiency Video Coding* (HEVC) [Sullivan et al., 2012] e *Versatile Video Coding* (VVC) [Bross et al., 2021].

Os objetivos específicos são:

- Compreender o código desenvolvido pela equipe do projeto;
- Seleção dos parâmetros mais adequados para realização dos treinamentos completos das redes neurais intra e inter-quadros, por meio de testes de treinamento e comparação do desempenho destes;
- Realizar o treinamento de longo prazo para otimização das redes neurais desenvolvidas pela equipe;
- Analisar o desempenho do codificador neural desenvolvido em comparação aos codificadores tradicionais, por meio da realização de testes de compressão, cálculo de métricas e posterior comparação de resultados.

1.4 Justificativa

A utilização de redes neurais para a realização de compressão de vídeo está sendo cada vez mais explorada pelo campo de pesquisa científica.[Ding et al., 2021]

Novos codificadores de vídeo precisam passar por avaliação de desempenho para se verificar a viabilidade de utilização. Além disso, é interessante comparar esse desempenho com o dos codificadores de vídeo já consolidados no mercado, para uma visão mais realista e identificação de limitações.

Frente a isso, este estudo tem um significativo papel na análise de um codificador de vídeo baseado em *autoencoder* variacional, realizando comparações de desempenho em relação aos codificadores tradicionais mais atuais HEVC e VVC.

1.5 Organização do texto

No segundo capítulo, são apresentados conceitos teóricos necessários para fundamentação de tópicos relacionados à compressão de vídeo e redes neurais. Também, são abordados conceitos dos codificadores tradicionais mais atuais e das principais métricas empregadas para avaliação objetiva do desempenho de codificadores de vídeo. No capítulo três, é apresentada uma visão geral sobre o codificador de vídeo baseado em *autoencoder* variacional, o qual foi desenvolvido pela equipe do projeto *Deep Codec* (referenciada por equipe *Deep Codec* no decorrer do texto).

No quarto capítulo, é detalhada a metodologia do estudo descrevendo os treinamentos de redes neurais realizados e etapas para avaliação do desempenho do codificador. Em seguida, no quinto capítulo, são apresentados os resultados obtidos para os modelos treinados e análises de desempenho do codificador. Por fim, as conclusões e sugestões de trabalhos futuros são apresentadas no capítulo seis.

Capítulo 2

Fundamentos

Segundo [Sayood, 2012], o objetivo principal de qualquer algoritmo relacionado à compressão é representar informações de uma forma compacta. Cada vez mais pessoas possuem acesso à internet e o desenvolvimento da comunicação móvel tornou possível o consumo de dados multimídia em praticamente qualquer local do globo. Entretanto, isso só é possível devido a existência de técnicas de compressão.

Ainda segundo o autor, uma grande parte dos dados consumidos e enviados estão em formato digital, constituindo *bytes* de dados. A organização dessa informação digital é explorada pela compressão para identificar possíveis maneiras de tornar sua representação mais compacta.

A enorme quantidade de dados gerados sinaliza maior atenção aos conceitos de transmissão e armazenamento de dados. Embora novas tecnologias nessas áreas estejam surgindo para aumentar o desempenho, a demanda cresce de forma mais intensa, segundo apresentado por [Sayood, 2012]. Dessa maneira, a compressão de dados tem um papel fundamental para permitir que as estruturas existentes de transmissão e armazenamento possam ser utilizadas de forma mais otimizada.

Visto que este estudo discorre sobre a análise de desempenho de um codificador de vídeo baseado em redes neurais, o presente capítulo irá elucidar alguns conceitos e termos abordados no decorrer do texto.

O capítulo inicia com conceitos fundamentais de compressão de vídeo, como predição intra e inter-quadros, profundidade de bits, espaço de cores YUV e otimização de taxa e distorção. A seguir, as principais características dos codificadores tradicionais HEVC e VVC e as principais métricas utilizadas para avaliação de codificadores serão apresentadas.

Uma visão geral acerca de redes neurais, do propósito das arquiteturas de *autoencoders* e, também, de sua aplicação na realização de compressão de dados encerram este capítulo.

2.1 Conceitos de compressão de vídeo

Esta seção irá apresentar conceitos fundamentais relacionados ao campo de compressão de vídeo, como profundidade de *bits* de uma imagem, espaço de cores YUV, os principais tipos de predição de quadros e a relação de taxa e distorção, cuja otimização é importante para que um dado codificador apresente bom desempenho.

2.1.1 Profundidade de *bits*

As unidades constituintes de imagens digitais são os píxeis. Para um mapa de cores RGB, com a presença de três canais (vermelho, verde e azul), a combinação das intensidades das componentes constituintes de cada um dos canais definem a cor de um dado píxel. Quando todos as componentes tem valor máximo, a imagem será totalmente branca, ao passo que se todas as componentes do canal vermelho tiverem valor máximo e as componentes dos outros canais tiverem valor nulo, então a imagem resultante terá cor vermelha.

Ao se mencionar que um determinado vídeo possui 8 *bits*, isso significa que cada um de seus canais possuem $2^8 = 256$ níveis de intensidade para seus píxeis constituintes. Dessa forma, combinando-se os três canais resulta em um total de 256^3 cores possíveis, sendo superior a $16,7$ milhões de cores.

Quanto maior a quantidade de *bits* que representam a intensidade de um píxel, maior será a profundidade de *bits* e, também, maior qualidade de imagem. Muitas vezes, é necessário um maior espectro de cores e com isso pode-se aumentar a profundidade para 10 *bits*, por exemplo. Dessa maneira, haverá um total de $2^{10} = 1024$ intensidades por canal, resultando em 1024^3, um valor superior a 1 bilhão de cores.

Maior profundidade de *bits* gera arquivos maiores e com maior taxa de dados. Por exemplo, para um vídeo com resolução de 1920×1080 por quadro, com taxa de 30 quadros por segundo e profundidade de 8 *bits* resulta em aproximadamente $1,5$ *gigabits* por segundo. Caso a profundidade fosse de 10 *bits*, o resultado seria de aproximadamente $1,9$ *gigabits* por segundo. Com isso, a compressão se faz necessária.

2.1.2 Espaço de cores YUV e subamostragem de cores

Segundo [Weston, 2019], o olho humano é muito mais sensível à luminância presente em uma imagem do que às cores em específico. Por conta disso, ao se realizar processamento de vídeo, grande parte da informação de cores presente nos quadros pode ser descartada, mas mantendo a informação de luminância preservada.

Para trabalhar com as componentes de luminância e crominância separadamente, é comum a utilização do espaço de cor YUV. Quando todos os canais utilizam a mesma taxa

de amostragem, tem-se uma amostragem YUV 4:4:4. Muitas representações utilizam uma amostragem onde as componentes dos canais de crominância equivalem a um quarto (metade na vertical e metade na horizontal) do total de amostras de luminância, originando a amostragem 4:2:0. Para aplicações de maior resolução pode-se utilizar amostragem YUV 4:4:4 ou 4:2:2, além de maior profundidade de bits, conforme apresentado em 2.1.1.

2.1.3 Predições intra e inter-quadros

Dado que um vídeo é constituído de uma sequência de várias imagens denominadas quadros, há uma grande presença de redundância espacial e temporal. Em uma imagem, píxeis vizinhos apresentam muita similaridade devido à proximidade entre si estabelecendo uma correlação espacial. A correlação temporal advém do fato de, em um pequeno intervalo de tempo, existir um conjunto extenso de imagens capturadas para constituir esse período do vídeo, assim, as diferenças entre os quadros desse conjunto tendem a ser relativamente pequenas.

Ao se utilizar a predição intra para compressão de um quadro de vídeo, a imagem será comprimida baseada em suas próprias informações que já estejam disponíveis no decodificador, explorando apenas a redundância espacial. Essa predição dá origem aos quadros do tipo I (Intra), que são importantes pois permitem que quadros comecem a ser preditos a partir de outros quadros.

Segundo [JUNG, 2021], durante a predição intra nos codificadores H.264 e HEVC, blocos já decodificados são utilizados para estimação do próximo bloco, utilizando a ideia de redundância espacial. Funções são então aplicadas aos píxeis dos blocos vizinhos gerando modos de predição. O codificador então realiza uma análise desses modos e seleciona aquele de melhor resultado, considerando tanto a magnitude do resíduo, que é a diferença entre o bloco predito e o original, como também a quantidade necessária de *bits* para transmissão.

A redundância temporal é abordada na predição inter, a qual utiliza informações de outros quadros de vídeo para gerar a predição de um dado quadro. Geralmente, o codificador de vídeo trabalha sobre grupos de imagens (*Group of Pictures*, GOP) para realizar compressão. O primeiro quadro desse grupo é codificado, tipicamente, utilizando predição intra e os demais utilizam predicão inter. De forma geral, os quadros de predição inter são classificados em dois tipos: P (Predito) ou B (Bipredito ou Bidirecional). Quadros do tipo P são comprimidos utilizando somente quadros anteriores, enquanto os quadros B utilizam tanto quadros anteriores quanto posteriores na sequência do GOP.

Segundo [JUNG, 2021], para a realização de compressão inter, o quadro atual do vídeo é repartido em blocos e o codificador procura um bloco semelhante em uma região limitada presente no quadro de referência, o qual deve ser um quadro codificado que pode

estar tanto no passado quanto futuro dependendo do tipo de predição sendo utilizada. O deslocamento aparente do bloco é calculado gerando um vetor de movimento, processo chamado de estimação de movimento. Aplicando o vetor de movimento ao bloco de referência, realizando uma compensação de movimento, um bloco predito é obtido. A diferença entre o bloco sendo codificado e esse bloco predito é calculada gerando um resíduo. No fluxo de *bits* (do inglês *bitstream*), são codificados esse vetor de movimento, as coordenadas do bloco similar que deu origem a esse vetor e o resíduo. Esse *bitstream* contém menor entropia (informação) do que o quadro completo, tornando a compressão mais eficiente.

Após a estimação de movimento, deve ser realizada a compensação de movimento por meio da aplicação dos vetores de movimento nos quadros de referência dando origem a uma predição do quadro atual. O decodificador recupera os vetores de movimento e o resíduo a partir do *bitstream* e os utiliza para formar o quadro final reconstruído.

2.1.4 Otimização taxa-distorção

Otimização de taxa-distorção é fundamental em compressão de imagem e vídeo que envolvem perdas. Esse conceito indica a relação de troca existente entre taxa e distorção. Para maiores taxas, uma maior quantidade de *bits* é utilizada para representar a compressão de uma imagem ou quadro de um vídeo o que leva a uma melhor qualidade de imagem e menor distorção. Por outro lado, uma menor quantidade de *bits* gera imagens mais distorcidas e de menor qualidade.

Conforme [JUNG, 2021], seja uma taxa $R(X)$ de codificação de uma imagem X, $D(X, Y)$ a distorção entre uma imagem Y distorcida e sua imagem original X e λ (*lambda*) um parâmetro de ponderamento da relação de troca entre taxa e distorção, segue a definição do valor J que representa essa relação:

$$J = R(X) + \lambda \cdot D(X, Y) \tag{2.1}$$

Para codificadores de imagem e vídeo com aprendizado, é comum a definição inicial do parâmetro *lambda* na Equação 2.1, otimizando de maneira simultânea taxa e distorção, mas mantendo a relação entre elas. [JUNG, 2021]

2.2 Codificadores tradicionais

Segundo [Sullivan et al., 2012], os padrões de compressão de vídeo iniciaram uma evolução mais significativa a partir do desenvolvimento dos grupos de padronização ITU-T e ISO/IEC. O padrão de compressão H.264/MPEG-4 *Advanced Video Coding* (AVC) [Wi-

egand et al., 2003], desenvolvido conjuntamente entre o ITU-T e ISO/IEC, apresentou forte impacto e encontrou aplicação em uma ampla variedade de produtos ainda presentes no cotidiano, como transmissão de sinais de televisão de alta definição via satélite e cabo, aplicações de segurança, discos *Blu-ray* e aplicações de videoconferência. Atualmente, os codificadores mais recentes tradicionalmente utilizados são o *High Efficiency Video Coding* (HEVC) e o *Versatile Video Coding* (VVC), ambos desenvolvidos em conjunto por grupos de trabalho do ITU-T e ISO/IEC.

Segundo [Sullivan et al., 2012], o HEVC foi publicado em 2013, sendo desenvolvido para continuar atendendo aplicações do padrão anterior AVC e também focar em dois pontos importantes: vídeos de maior resolução e aumento na utilização de arquiteturas de processamento paralelo. Esse padrão oferece em torno de 50% de redução na taxa de dados (do inglês *bitrate*) sobre o padrão anterior AVC.

Em julho de 2020, o VVC foi finalizado como sendo o mais recente padrão de compressão de vídeo. Esse codificador apresenta em torno de 50% de redução de taxa de dados sobre o HEVC para uma mesma qualidade de vídeo e de 75% sobre o AVC.[Bross et al., 2021]

O VVC foi desenvolvido para ser altamente versátil e não somente reduzir a taxa de dados em relação aos padrões anteriores. Esse codificador busca trabalhar com diferentes necessidades no campo multimídia, envolvendo desde resoluções padrões até as mais elevadas como 8K, aplicações de vídeo imersivo 360 graus, realidade aumentada e, também, em aplicações de jogos online que requerem latência muito reduzida.[Bross et al., 2021]

Tanto o HEVC quanto o VVC ainda empregam em suas camadas de compressão de vídeo a convencional abordagem de codificação de vídeo híbrida baseada em blocos com técnicas de intra e inter-predição e transformação espacial em duas dimensões. [Sullivan et al., 2012], [Bross et al., 2021]

Segundo [Bross et al., 2021], o JVET vem estabelecendo grupos de pesquisa sobre compressão de vídeo baseada em redes neurais. Com o crescente sucesso da aplicação de aprendizado de máquina nas áreas de visão computacional e de processamento de imagens, pesquisadores estão investigando o uso dessa abordagem na área de compressão de vídeo. Os trabalhos de pesquisa são divididos em duas categorias: esquemas de compressão com aprendizado de ponta-a-ponta, e incorporação de ferramentas de codificação com aprendizado em esquemas de compressão convencionais, como do HEVC e VVC.

Técnicas de aprendizado de máquina empregadas nos esquemas dos codificadores tradicionais HEVC e VVC visando otimização estão cada vez mais presentes nos estudos e pesquisas. Para otimização do HEVC, [Bouaafia et al., 2021a] propõe modelo de aprendizado profundo para o algoritmo de decisão de inter-modos. Em [Bouaafia et al., 2021b] é apresentada uma abordagem de redes neurais convolucionais profundas para otimização

da convencional filtragem em laço (do inglês *in-loop filtering*) do VVC.

2.3 Métricas para avaliação

Para que seja possível avaliar e comparar reconstruções de vídeos que passaram pelo processo de compressão por algum codificador, é necessário determinar a qualidade dos vídeos apresentados. Medir a qualidade visual de imagens e vídeos é uma tarefa complexa devido a sua subjetividade. A utilização de critérios objetivos para se avaliar a qualidade de uma imagem traz acurácia e permite reprodutibilidade dos resultados, embora esse tipo de abordagem nem sempre reproduza fielmente a subjetividade que uma análise visual realizada por uma pessoa envolve.

Diante dessa complexidade da análise subjetiva, as métricas de qualidade objetivas são amplamente utilizadas. A métrica de relação sinal-ruído de pico (*Peak signal-to-noise ratio* - PSNR) é a mais comum, mas muito limitada. Com isso, métricas objetivas que tentam se aproximar dos resultados de testes de qualidade subjetivos também vem sendo desenvolvidas, dentre as quais estão o índice de similaridade estrutural (*Structural Similarity Index Measure* - SSIM) e o índice de similaridade estrutural multi-escala (*Multi-Scale Structural Similarity Index Measure* - MS-SSIM).

2.3.1 Relação sinal-ruído de pico (PSNR)

A relação sinal-ruído de pico é uma métrica simples e muito utilizada para avaliação de imagens comprimidas. Ela depende da presença da imagem original para comparação.

Essa métrica aplica uma função logarítmica sobre o erro quadrático médio (*Mean Squared Error* - MSE) entre a imagem original e sua versão distorcida. A equação para obtenção da PSNR é apresentada a seguir:

$$\text{PSNR} = 10 \cdot \log_{10} \frac{(2^n - 1)^2}{\text{MSE}} \tag{2.2}$$

Na Equação 2.2, o parâmetro n faz referência ao número que representa a profundidade de *bits* da imagem, conforme apresentado em 2.1.1. O resultado é informado em escala decibel. Maiores valores indicam uma melhor qualidade enquanto valores mais baixos, menor qualidade

Segundo [Richardson, 2010], a métrica PSNR tem muitas limitações, como o fato dela depender da presença da imagem original para comparação a qual pode nem sempre estar disponível e, também, a sua baixa correlação com resultados de medidas subjetivas de qualidade de vídeo.

Um exemplo que demonstra uma das limitações da PSNR é quando o valor de todos os píxeis de uma dada imagem tem sua intensidade reduzida em uma unidade ou então quando todos os píxeis são deslocados em uma das direções em apenas uma unidade. Ao realizar essas ações, a PSNR irá degradar bastante, mas, para uma pessoa que observa a imagem, é como se nada tivesse ocorrido pois essa deterioração não é percebida visualmente.

2.3.2 Índice de similaridade estrutural (SSIM)

Segundo [Wang et al., 2004], MSE e PSNR são as métricas mais comumente utilizadas pois apresentam maior simplicidade de cálculo, possuem um significado físico claro e são de maior facilidade para otimização. Entretanto, não são muito adequadas para avaliar a qualidade visual. Dessa forma, esse mesmo autor desenvolveu o índice de similaridade estrutural, uma métrica que procura levar em conta aspectos do sistema visual humano para medir a similaridade entre duas imagens. Essa métrica considera a combinação de três fatores: distorção de luminância, distorção de contraste e correlação entre as imagens.

Por ser uma métrica que leva em conta uma medida de similaridade estrutural e o sistema visual humano possuir forte tendência a extrair características estruturais de imagens, essa métrica procura estar mais relacionada com a qualidade perceptual e seus resultados tendem a representar isso.

Essa métrica apresenta um cálculo mais complexo. Entretanto, diversas bibliotecas de processamento de imagens e vídeos possuem funções implementadas para uso facilitado da métrica.

2.3.3 Índice de similaridade estrutural multi-escala (MS-SSIM)

O índice de similaridade estrutural multi-escala foi proposto por [Wang et al., 2003] como uma forma de determinação de qualidade de uma imagem. Essa proposta traz maior flexibilidade do que a abordagem em única escala da SSIM ao incorporar variações de resolução e de condições de visualização.

Segundo esse próprio autor, a percepção dos detalhes presentes em uma dada imagem depende de alguns fatores, tais como: densidade de amostragem do sinal da imagem, da distância do observador ao plano da imagem, e, também, do próprio sistema visual desse observador. O método multi-escala da métrica MS-SSIM é uma interessante abordagem para incorporar detalhes da imagem sob diferentes resoluções.

Dessa forma, a métrica MS-SSIM, uma extensão da SSIM, emprega uma abordagem multi-escala através de iterações em que há utilização de filtro passa-baixa e subamostragem na imagem.

Assim como para a métrica SSIM, bibliotecas de processamento de imagens e vídeos possuem funções implementadas para o uso da MS-SSIM.

2.4 Redes neurais

Este estudo tem por objetivo analisar o desempenho de um codificador de vídeo baseado em *autoencoder* variacional. Esta seção, irá apresentar conceitos gerais sobre redes neurais, envolvendo assuntos relacionados ao processo de treinamento de uma rede e sobre camadas convolucionais que são mais utilizadas em arquiteturas para processamento de imagens. Por fim, conceitos relacionados a *autoencoder* e *autoencoder* variacional são apresentados.

2.4.1 Treinamento de uma rede

Uma rede neural é constituída de uma série de funções interligadas e separadas em camadas. Essas funções são conhecidas como neurônios, em analogia à rede cerebral humana. O funcionamento geral consiste em um neurônio de uma dada camada receber como entrada um resultado fornecido por outro neurônio de uma camada anterior. Esse processo continua sucessivamente até que a última camada da rede neural forneça o seu resultado.

Cada neurônio que constitui uma camada possui como características um peso, um viés (do inglês *bias*) e uma função de ativação. Segundo [JUNG, 2021], para um ajuste dos pesos, é necessário realizar o treinamento de um modelo utilizando um conjunto de pares de valores (x, y), chamado de conjunto de treinamento. Para cada entrada x, a rede neural gera um valor predito $\hat{y} = f(\boldsymbol{\theta}, x)$, utilizando um conjunto de pesos $\boldsymbol{\theta}$. Esse processo de obtenção da saída e valores intermediários é conhecido por *forward pass*.

A diferença obtida entre valores preditos e os valores que se desejavam obter geram um erro ou perda (do inglês *loss*). Esse valor é avaliado por alguma métrica, como MSE, por exemplo. A utilização de uma métrica diferenciável permite o cálculo do gradiente da perda em relação aos pesos utilizados na rede neural. Dessa maneira, calcula-se a derivada de cada um dos pesos com relação as diferenças durante o passo de retropropagação (*backpropagation*, em inglês) e ajustam-se os pesos com base no gradiente durante o passo chamado de gradiente descendente. [JUNG, 2021]

Com pequenas alterações realizadas nos pesos, as saídas dos neurônios podem ser alteradas de forma muito significativa, o que atrapalha o processo de treinamento da rede. Dessa maneira, funções de ativação são empregadas nos neurônios. Essas funções tem o objetivo de realizar transformações não lineares nos dados de entrada, permitindo que a rede aprenda e execute tarefas mais complexas.[Academy, 2021b]

2.4.2 Camadas convolucionais

Camadas convolucionais fazem parte da arquitetura das redes neurais convolucionais (CNN), realizando operações de convolução sobre entradas recebidas de outras camadas no modelo.

Convoluções são aplicadas em uma imagem capturando traços marcantes e identificando padrões de repetição por meio da aplicação de filtros. Quanto maior a quantidade de filtros aplicados, mais profundas serão as camadas de convolução e, com isso, mais detalhadas serão as características identificadas pelos filtros.

Ao aplicar convoluções sobre uma imagem de entrada, a imagem de saída apresentará dimensão reduzida. Para evitar essa redução de dimensão, pode-se aplicar *padding* na imagem. Dessa forma, a borda da imagem será estendida com píxeis adicionais, os quais podem receber valores nulos, valores espelhados ou repetidos com mesmo valor do píxel da borda.

Segundo [JUNG, 2021], heuristicamente as CNNs apresentam melhor desempenho em menores resoluções e com muitos mapas de características do que com poucos mapas de características e maior resolução. Dessa forma, reduzir o tamanho das imagens e aplicar convoluções em menores resoluções tende a gerar melhores resultados.

2.5 *Autoencoder*

Autoencoder é uma arquitetura de rede neural constituída de um codificador e um decodificador que podem ser treinados pelo método de gradiente descendente. Os *autoencoders* tem sido utilizados com sucesso em tarefas de redução de dimensionalidade e recuperação de informação.

Segundo [Goodfellow et al., 2016], os *autoencoders* são restringidos de forma que não sejam capazes de copiar perfeitamente uma dada entrada. Com isso, são forçados a priorizar o aprendizado de determinados aspectos da entrada, levando a um aprendizado focado nas propriedades importantes de um determinado dado.

Segundo [Academy, 2021a], *autoencoder* variacional é um tipo de *autoencoder* cujo treinamento é regularizado para evitar sobreajuste e para que o espaço latente (espaço codificado) contenha propriedades que possibilitem processos generativos. A arquitetura desse tipo de *autoencoder* codifica entradas como distribuições ao invés de pontos e o espaço latente restringe as distribuições resultantes do codificador a uma gaussiana padrão.

Em [JUNG, 2021], o autor discorre que apesar dos *autoencoders* variacionais possuírem maior foco em geração de dados, eles podem ser empregados em uma ampla variedade de áreas, sendo a compressão de dados uma delas. Segundo o autor, uma das primeiras tentativas de utilização na área de compressão foi apresentada por [Ballé et al., 2016a], o

qual observou uma analogia entre a questão de otimização de taxa-distorção e a definição de um *autoencoder* variacional, treinando uma rede *autoencoder* para minimizar a entropia dos códigos gerados pelo modelo (cada modelo é treinado para um dado *lambda*) e também sua distorção que controla a posição do modelo na curva de otimização de taxa-distorção.

Capítulo 3

Codificador base

Este estudo foi realizado utilizando o codificador de vídeo baseado em *autoencoder* variacional implementado pela equipe *Deep Codec* como base para a realização dos treinamentos das redes neurais e experimentos de compressão.

Neste capítulo, será abordada uma visão geral e sucinta acerca das características e arquitetura desse codificador. Em [de Oliveira et al., 2021], é apresentada com mais detalhes a proposta desse codificador.

3.1 Sobre o codificador desenvolvido pela equipe do projeto

O codificador de vídeo proposto pela equipe *Deep Codec* consiste em um tipo de rede neural artificial conhecida como *autoencoder*, uma rede não supervisionada, pois não utiliza dados rotulados, que aprende como reduzir a informação de forma eficiente e reconstruir a entrada partindo dessa representação compactada. O *autoencoder* é responsável por codificar os intra-quadros e também os quadros preditos do tipo P, explorando, assim, correlações tanto espacial quanto temporal.

3.1.1 Grupo de imagens (GOP) e *bitstream* do codificador

O codificador possui o conceito de GOP, que também está presente em codificadores tradicionais. Na implementação, o codificador trabalha com um GOP de N quadros consecutivos, sendo que esse N não precisa necessariamente possuir um valor fixo, podendo variar. Não há qualquer dependência entre diferentes GOPs.

O primeiro quadro que constitui um dado GOP será codificado como um quadro intra e os demais serão codificados como quadros do tipo P. O quadro imediatamente anterior é utilizado como referência para a predição inter. O codificador aritmético percorre o GOP

em sequência do início ao fim para geração do *bitstream* relativo a esse grupo. Junto a esse *bitstream* é concatenado um cabeçalho com informações sobre o GOP em questão. Esse cabeçalho apresenta a quantidade de quadros que constitui o GOP (reforçando a possibilidade de variação do tamanho do grupo) e o tamanho do *bitstream* gerado pelo codificador aritmético.

O *bitstream* final do vídeo codificado contém um cabeçalho com informações gerais sobre o vídeo em questão, além dos *bitstreams* relativos a cada GOP do vídeo. Esse cabeçalho contém informações sobre as dimensões (altura e largura), a taxa de quadros por segundo e o formato YUV do vídeo de entrada. No codificador, o formato utilizado nas operações é YUV 4:4:4, então o vídeo de entrada será convertido para esse formato. A saída reconstruída será expressa no formato subamostrado YUV 4:2:0.

3.1.2 Codificador intra-quadros

O codificador intra-quadros é baseado no trabalho de [Ballé et al., 2016b], cuja arquitetura principal é apresentada na Figura 3.1. A arquitetura para codificação de intra-quadros é um *autoencoder* variacional com modelagem de entropia não-paramétrica.

O codificador é otimizado para minimizar a seguinte função objetivo:

$$L[g_a, g_s, p_{\tilde{\mathbf{y}}}] = -\mathbb{E}[\log_2 p_{\tilde{\mathbf{y}}}] + \lambda \mathbb{E}[d(\mathbf{z}, \hat{\mathbf{z}})] \tag{3.1}$$

O primeiro termo na Equação 3.1 é a expectativa de entropia (quantidade de informação) e o segundo termo é a expectativa de distorção, que faz referência à qualidade da reconstrução. O parâmetro λ (*lambda*) controla a relevância da reconstrução na otimização da função objetivo. [de Oliveira et al., 2021]

Segundo [de Oliveira et al., 2021], um codificador aritmético realiza a codificação do latente quantizado da rede utilizando tabelas fixas para cada um deles. Essas tabelas são reunidas e armazenadas durante o treinamento e então utilizadas em tempo de execução para codificar os coeficientes sem que ocorram perdas.

3.1.3 Codificador inter-quadros

O codificador inter-quadros codifica um dado quadro utilizando como única referência o quadro imediatamente anterior reconstruído. Ambos quadros são concatenados e utilizados como entrada do *autoencoder*. Uma visão geral da arquitetura desse codificador é apresentada na Figura 3.2.

Segundo [de Oliveira et al., 2021], com base na Figura 3.2 tem-se que o codificador (*Encoder*) é otimizado de forma conjunta para dar origem a um latente que contenha

tanto informações para a estimação de movimento quanto para a obtenção de resíduo entre quadros.

O decodificador de movimento (*Motion decoder*), tem o objetivo de gerar parâmetros para uma transformação afim de *warping* a ser aplicada sobre o quadro de referência, o qual foi previamente reconstruído utilizando o codificador intra apresentado em 3.1.2. Com a aplicação dessa transformada, é obtida a predição do quadro. [de Oliveira et al., 2021]

O decodificador de resíduo (*Residue decoder*) recupera o resíduo, o qual será adicionado ao quadro predito para se obter como resultado o quadro decodificado.

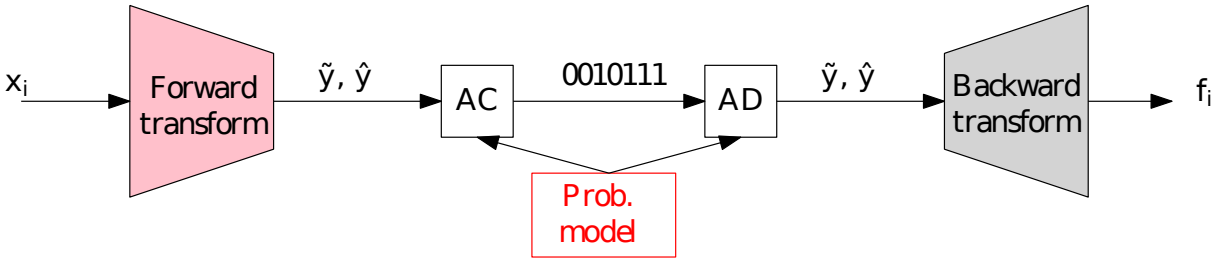

Figura 3.1: Arquitetura para codificador intra-quadros. AC e AD correspondem ao codificador e decodificador aritmético, respectivamente. Os termos x, y e f correspondem à entrada, ao espaço latente e à saída do codificador, respectivamente (Fonte: [de Oliveira et al., 2021]).

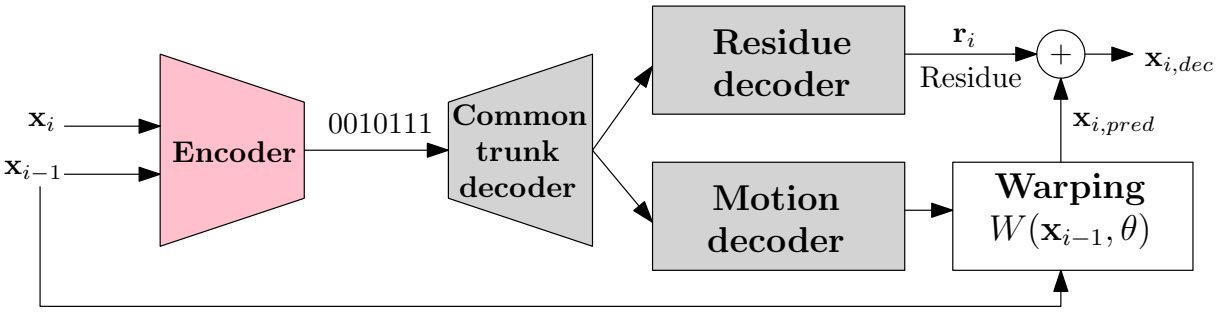

Figura 3.2: Visão geral da arquitetura do codificador inter-quadros (Fonte: [de Oliveira et al., 2021]).

Capítulo 4

Metodologia

Para a realização deste estudo, foram necessários alguns passos que envolvem desde o entendimento do código, passando pelos treinamentos das redes neurais do codificador e, por fim, realizando comparações entre os resultados de compressão obtidos pelo codificador neural e pelos codificadores tradicionais HEVC e VVC.

O codificador desenvolvido pela equipe *Deep Codec* é composto por duas redes neurais principais, uma para codificação de intra-quadros e outra para codificação de inter-quadros, também chamadas de redes intra e inter, respectivamente. A primeira trabalha na codificação de um único quadro e a última codifica um dado quadro baseada em um outro tomado como referência.

A etapa de treinamento das redes envolve a realização de testes (treinamentos iniciais) para uma análise prévia do comportamento de ambas redes, permitindo um melhor ajuste de parâmetros e definição das bases de dados que serão utilizadas durante os treinamentos finais de longo prazo.

Após a realização de todos os treinamentos, são realizados testes posteriores com o intuito de verificar o comportamento de taxa-distorção dos codificadores, neural e tradicionais, ao realizar a compressão de vídeos presentes em uma base de dados selecionada. Métricas são então calculadas sobre esses resultados, permitindo uma análise mais objetiva acerca do desempenho do codificador neural desenvolvido pela equipe *Deep Codec* frente aos codificadores HEVC e VVC.

Esses procedimentos metodológicos são apresentados em maior detalhe neste capítulo, dividido em treze passos, reunidos em três grandes fases: 4.1. Separação de código e treinamento teste das redes neurais, 4.2. Treinamento completo das redes neurais e 4.3. Realização de testes de compressão finais e comparação. Na Figura 4.1 é apresentado um fluxograma que sintetiza as etapas principais da metodologia apresentada neste capítulo.

Os treinamentos dos modelos neurais, sendo testes ou definitivos, foram realizados em placa de vídeo (GPU) NVidia GeForce RTX 2080 Ti, utilizando computadores da infraes-

trutura do Laboratório de Imagens, Sinais e Acústica (LISA) vinculado ao Departamento de Ciência da Computação (CIC) da Universidade de Brasília (UnB).

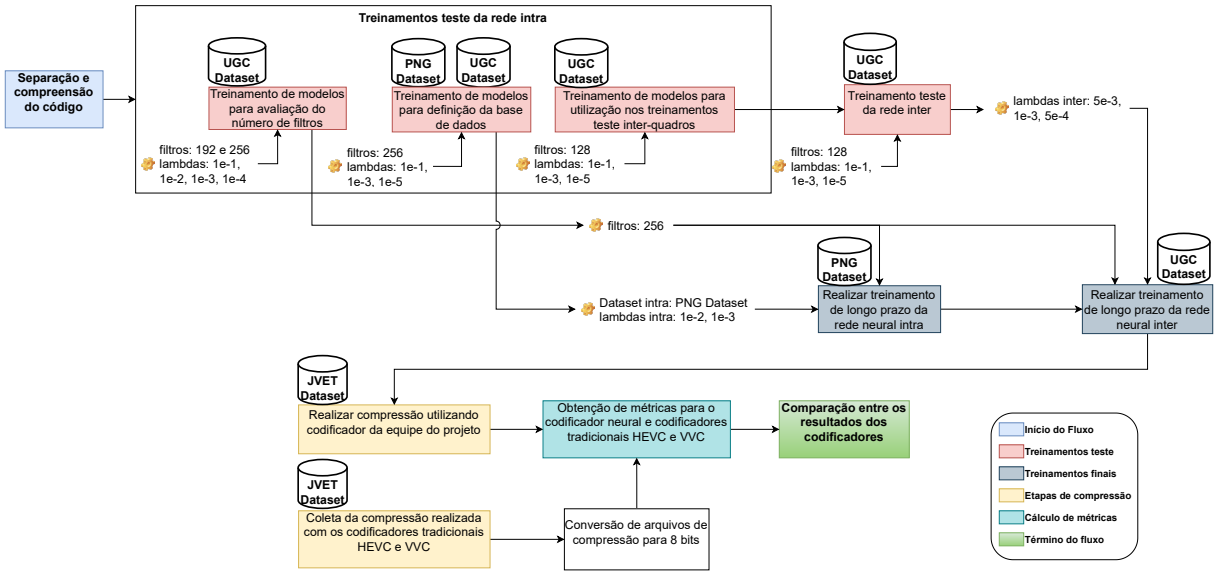

Figura 4.1: Fluxograma com passos representativos da metodologia adotada.

4.1 Separação de código e treinamento teste das redes neurais

Nesta seção, serão apresentados os principais passos necessários para a realização dos treinamentos das redes neurais do codificador. Envolverá explicação breve sobre como se dá a utilização dos módulos de treinamento das redes neurais, os bancos de dados utilizados nos treinamentos e como foram realizados os treinamentos iniciais para uma análise prévia de comportamento.

4.1.1 Separação e compreensão do código

Para possibilitar o treinamento dos respectivos modelos das redes neurais intra e inter, foi necessário, inicialmente, identificar os módulos do código, desenvolvido pela equipe *Deep Codec*, os quais são responsáveis por essa tarefa.

Tanto o módulo destinado ao treinamento intra quanto inter possuem abordagem semelhante para a ingestão dos parâmetros necessários para a inicialização do aprendizado das redes. É preciso informar um arquivo de configuração *.json* que contêm um parâmetro de comando (*command*) cujo valor indica que será realizado treinamento da rede e,

também apresenta demais parâmetros necessários para indicar, por exemplo, quantidade de filtros e o parâmetro de configuração *lambda* do modelo a ser treinado.

Um exemplo parcial da estrutura do arquivo de configuração para geração de modelos intra é apresentado na Figura 4.2. Já na Figura 4.3, é apresentado um exemplo da estrutura do arquivo de configuração para geração de modelos inter.

A seguir são caracterizados os parâmetros apresentados na Figura 4.2 e Figura 4.3:

- *command*: contém a *string "train"* que indica ao módulo que será realizado um treinamento da rede;
- *num_filters, intra_num_filters e inter_num_filters*: indica a quantidade de filtros utilizados nas camadas convolucionais constituintes da arquitetura do *autoencoder* variacional para cada um dos modelos, intra ou inter;
- *intra_checkpoint_dir e inter_checkpoint_dir*: caminho em disco onde está salvo o modelo intra-quadros e onde será salvo o modelo inter-quadros;
- *lmbda*: parâmetro *lambda* que faz referência ao multiplicador Lagrangiano para ponderamento entre taxa e distorção;
- *batchsize*: indica o número de amostras processadas antes de ocorrer uma atualização do modelo da rede neural;
- *patchsize*: tamanho dos recortes sobre as imagens de treinamento.

Os principais parâmetros também podem ser passados via linha de comando, assim, os informados via arquivo *.json* serão preteridos frente a esses. O arquivo de configuração é gravado na mesma pasta destinada ao modelo treinado, para que se mantenha um histórico do treinamento.

Com a finalidade de manter um histórico fiel dos parâmetros utilizados no treinamento das redes, adicionou-se um trecho de código que visa alterar os parâmetros do arquivo de configuração de acordo com os informados via terminal. Assim, os parâmetros descritos por linha de comando são atualizados no arquivo *.json* e essa configuração completa será salva juntamente ao seu respectivo modelo neural, facilitando posterior identificação dos parâmetros utilizados em determinado treinamento.

Com a identificação dos módulos do código os quais são responsáveis por implementar o treinamento das redes neurais, foi possível compreender a abordagem necessária para dar início ao treinamento dos modelos intra e inter-quadros do codificador.

```
{
    "command": "train",
    "num_filters": 256,
    "batchsize": 8,
    "patchsize": 256,
    "lmbda": 0.01,
    "last_step": 500000
}
```

Figura 4.2: Exemplo de código de configuração para treinamento da rede intra-quadros com os principais parâmetros.

```
{
    "command": "train",
    "intra_checkpoint_dir": "modelos_intra/modelo_1",
    "intra_num_filters": 256,
    "inter_checkpoint_dir": "modelos_inter/modelo_1",
    "inter_num_filters": 256,
    "lmbda": 0.01,
    "batchsize": 8,
    "patchsize": 256,
    "last_step": 300000
}
```

Figura 4.3: Exemplo de código de configuração para treinamento da rede inter-quadros com os principais parâmetros.

4.1.2 Seleção dos bancos de dados

Foram selecionados dois bancos de dados (*datasets*) distintos para o treinamento das redes neurais dos codificadores intra e inter.

Para o treinamento dos modelos finais de intra-quadros, foi utilizado um banco de dados de imagens de formato *.png* (referenciado como *PNG Dataset* no decorrer do texto) o qual consiste em uma união de diversos outros conjuntos de imagens. Com isso, o conjunto de treinamento se torna bem amplo e variado. Conforme apresentado em [de Oliveira et al., 2021], os seguintes bancos de dados foram utilizados:

1. *CLIC Professional dataset* com 628 imagens [CLIC, 2020]
2. *CLIC Mobile dataset* com 1111 imagens [CLIC, 2020]
3. *DIV2K dataset* com 902 imagens [Agustsson and Timofte, 2017]
4. *Ultra-Eye Ultra HD dataset* com 41 imagens [Nemoto et al., 2014]

5. *MCL-JCI* com 51 imagens [Jin et al., 2016]
6. *FLICKR2K dataset* com 2650 imagens [Lim et al., 2017]

Para cada um desses bancos de imagens, a equipe *Deep Codec* extraiu trechos de imagens sem sobreposição com tamanhos 256×256 píxeis, dando origem ao banco de dados final constituído de 88529 recortes (*patches*, em inglês) de imagens. [de Oliveira et al., 2021]

Já para o treinamento do codificador inter, que lida com predição de quadros consecutivos de um vídeo, a equipe *Deep Codec* decidiu por selecionar um banco de dados de vídeos com diversidade de conteúdo, tanto natural quanto sintético. Assim, o banco de dados *UGC Dataset* [Youtube, 2020] constituído de vídeos da plataforma *Youtube* foi escolhido. Esse *dataset* possui vídeos de diversas categorias, dentre as quais estão:

1. *Animation* com 22 vídeos
2. *CoverSong* com 22 vídeos
3. *Gaming* com 39 vídeos
4. *HowTo* com 17 vídeos
5. *Lecture* com 24 vídeos
6. *LiveMusic* com 21 vídeos
7. *Lyrics* com 2 vídeos
8. *MusicVideo* com 26 vídeos
9. *NewsClip* com 17 vídeos
10. *Sports* com 33 vídeos
11. *VerticalVideo* com 20 vídeos
12. *Vlog* com 32 vídeos

Novamente, para uma padronização do formato das imagens (constituídas de quadros de vídeos) de treinamento da rede neural de inter-quadros, os 275 vídeos do *UGC dataset* foram tratados pela equipe *Deep Codec*. Dessa forma, todo o conteúdo do banco de dados foi transformado para YUV 444 e normalizado no intervalo [0, 1]. Além disso, trechos de dimensão 256×256 píxeis foram extraídos de uma região de um dado vídeo de forma randômica e o mesmo deslocamento foi utilizado para todos os quadros. [de Oliveira et al., 2021]

O *UGC Dataset* também foi utilizado em alguns treinamentos iniciais da rede do codificador intra, apresentados em 4.1.3.

4.1.3 Treinamentos testes da rede intra

Os treinamentos testes do codificador intra foram realizados em três grupos, cada um tinha um dos seguintes objetivos:

- gerar modelos intra para utilização nos treinamentos testes inter e posterior definição dos *lambdas* inter finais;
- gerar modelos para avaliar número de filtros a ser utilizado nos treinamentos finais;
- gerar modelos treinados com bases diferentes para decidir qual base será utilizada no treinamento final intra e definição dos *lambdas* intra finais.

Treinamento de modelos para utilização nos treinamentos testes inter-quadros

Inicialmente, foram treinados três modelos com 128 filtros, utilizando o banco de dados com arquivos *.yuv* do *UGC Dataset* e foram escolhidos três valores de *lambda*: 10^{-1}, 10^{-3} e 10^{-5}. Esses modelos foram treinados por 500 mil iterações e foram utilizados como parâmetro de entrada no treinamento teste da rede inter a ser apresentado em 4.1.4.

Treinamento de modelos para avaliação do número de filtros

Para realizar uma análise referente ao número de filtros a serem utilizados no treinamento da rede, foram treinados quatro modelos tanto para 192 filtros quanto para 256 filtros, sendo utilizado o *UGC Dataset*. Os valores de *lambda* utilizados nesses treinamentos foram 10^{-1}, 10^{-2}, 10^{-3} e 10^{-4}, executando por 500 mil iterações.

Treinamento de modelos para definição da base de dados

Para se definir o banco de dados do codificador intra-quadros e também os valores dos parâmetros *lambda* a serem utilizados nos treinamentos dos modelos finais, foram treinados seis modelos intra, sendo três modelos utilizando como base de dados o *PNG Dataset* e os três demais modelos, o *UGC dataset*. Com isso, modelos treinados com o mesmo parâmetro *lambda* utilizando banco de dados distintos foram comparados para definir qual das bases seria utilizada para treinamentos de longo prazo. Esses treinamentos utilizaram 256 filtros, *lambdas* de 10^{-1}, 10^{-3} e 10^{-5} e passaram por 500 mil iterações, como os treinamentos anteriores.

4.1.4 Treinamento teste da rede inter

A rede inter-quadros utiliza uma rede intra treinada previamente, a qual será responsável pela codificação dos quadros âncoras (intra-quadros).

Utilizando-se os três modelos intra treinados com 128 filtros, indicados em 4.1.3, foram realizados treinamentos de nove modelos inter. Selecionaram-se três valores para o parâmetro *lambda* em concordância com os da rede intra: 10^{-1}, 10^{-3} e 10^{-5}. Desse modo, para cada *lambda* inter, foram utilizados os três modelos treinados da rede intra.

Para os treinamentos testes da rede inter, foram utilizadas 300 mil iterações e o *UGC dataset* como base de dados. Os nove modelos de codificadores inter, tem sua combinação exemplificada na Figura 4.4.

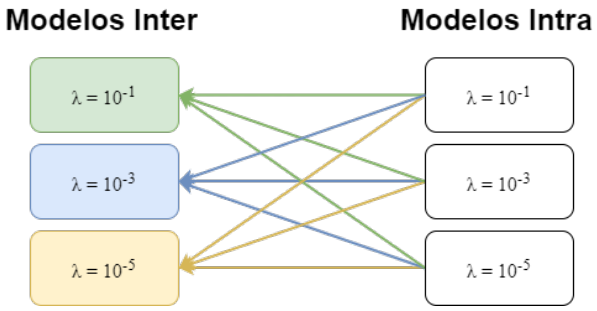

Figura 4.4: Ilustração das combinações realizadas durante o treinamento teste da rede neural inter. Para um dado valor de *lambda* da rede inter, foram treinados três modelos utilizando cada um dos três modelos intra treinados.

4.1.5 Análise dos resultados dos testes

Com o objetivo de selecionar os parâmetros finais a serem utilizados nos treinamentos finais dos codificadores intra e inter-quadros, foi necessário realizar uma análise do comportamento dos treinamentos testes realizados.

O comportamento dos treinamentos testes foi estudado com base na análise dos gráficos gerados em tempo de treinamento pela ferramenta *Tensorboard* [Tensorflow, 2021].

Após a realização dos treinamentos testes da rede intra, foi realizada análise dos gráficos gerados pelo *Tensorboard* para as métricas *Bits* por pixel (BPP) e Erro Quadrático médio (*Mean Squared Error* - MSE) dos modelos obtidos. Assim, foi possível realizar a comparação relativa ao número de filtros (5.1.1) e também definir a base de dados a ser utilizada nos treinamento finais e também os parâmetros *lambda* (5.1.2).

Com base nessas comparações, foi tomada a decisão de utilizar a base *PNG Dataset* para o treinamento dos modelos intra-quadros finais do codificador. Além disso, foram definidos os valores de 10^{-2} e 10^{-3} para o *lambda* a ser utilizado nesses modelos finais, assim como a utilização de 256 filtros.

Analisando-se o comportamento dos nove modelos inter, mencionados em 4.1.4, foram definidos três valores *lambda* para o treinamento dos modelos finais. Com base nos re-

sultados, que serão apresentados em 5.2.1, os parâmetros *lambda* da rede inter-quadros foram $5 \cdot 10^{-3}$, 10^{-3} e $5 \cdot 10^{-4}$.

4.2 Treinamento completo das redes neurais

Após a realização dos treinamentos iniciais para verificar o comportamento das redes neurais, foram definidos os valores do parâmetro *lambda*, o número de filtros e as bases de dados que foram utilizadas em cada um dos codificadores, intra e inter, nos treinamentos de longo prazo. Esses treinamentos foram realizados por uma maior quantidade de iterações, permitindo uma melhor convergência das redes neurais.

Essa atividade resultou na obtenção dos modelos finais para os codificadores intra e inter-quadros para posterior análise na realização de testes de compressão e obtenção de métricas.

4.2.1 Treinamento de longo prazo da rede neural intra

Os treinamentos definitivos dos modelos do codificador intra-quadros foram realizados utilizando o *PNG Dataset*. Para uma melhor otimização e estabilidade do modelo, foram realizadas 1 milhão de iterações. Os valores definidos para os parâmetros *lambda*, que representam o modelo, foram de 10^{-2} e 10^{-3}.

Em 5.1.3 serão apresentados os resultados para o comportamento desse treinamento.

4.2.2 Treinamento de longo prazo da rede neural inter

Os treinamentos de longo prazo dos modelos neurais inter-quadros mantiveram o uso do *UGC dataset*. Assim como no treinamento dos modelos definitivos intra-quadros, a quantidade de iterações utilizadas sofreu um aumento. Os modelos inter foram treinados por 500 mil iterações.

As 500 mil iterações foram divididas em duas etapas. Primeiramente, a rede inter-quadros foi treinada por 300 mil iterações e o comportamento foi verificado. Então, decidiu-se continuar o treinamento por mais 200 mil iterações para que a rede adquirisse maior estabilidade.

Foram definidos os seguintes valores para o parâmetro *lambda*: $5 \cdot 10^{-3}$, 10^{-3} e $5 \cdot 10^{-4}$. Dessa maneira, foram treinados seis modelos no total, onde para cada *lambda* inter foram treinados dois modelos, cada um utilizando um modelo intra com *lambda* diferente, os quais foram treinados de acordo com 4.2.1. O comportamento dos modelos finais inter será apresentado em 5.2.2.

4.3 Realização de testes de compressão finais e comparação

Esta seção irá apresentar como foi realizada a obtenção das curvas de taxa-distorção para o codificador neural, para o HEVC e VVC. Esse processo envolve desde a seleção da base a ser utilizada nos testes, passando pela etapa de compressão em si e, por fim, a geração dos gráficos de métricas para as sequências avaliadas. Com as métricas PSNR, SSIM e MS-SSIM expostas em gráficos de taxa-distorção, é possível realizar a comparação de desempenho entre os codificadores.

4.3.1 Seleção do banco de dados de testes (JVET)

A equipe *Deep Codec* estabeleceu o *dataset* JVET como base para a realização de testes de compressão dos codificadores. Especificamente, foram selecionadas sequências pertencentes as classes D e F desse agrupamento de vídeos.

Conforme apresentado em [de Oliveira et al., 2021], os parâmetros de codificação utilizados seguiram a recomendação das condições de teste definidas para atividades de codificação de vídeo baseadas em redes neurais profundas DNNVC da ISO/IEC JTC 1/SC 29/WG 11.[Boyce et al., 2018]

As seguintes sequências foram utilizadas nos testes de compressão:

- Sequências da Classe D e suas características
  - RaceHorses (300 *frames*, 30 fps, 416×240 *pixels*)
  - BQSquare (600 *frames*, 60 fps, 416×240 *pixels*)
  - BlowingBubbles (500 *frames*, 50 fps, 416×240 *pixels*)
  - BasketballPass (500 *frames*, 50 fps, 416×240 *pixels*)
- Sequência da Classe F e suas características
  - BasketballDrillText (500 *frames*, 50 fps, 832×480 *pixels*)

Os vídeos mencionados foram codificados com profundidade de 10 bits e a distância entre quadros âncoras (GOP *size*, em inglês) é constante para uma dada sequência e depende da taxa de quadros da fonte [de Oliveira et al., 2021]. Os valores do GOP *size* para cada um dos vídeos foram: 32 para RaceHorses, 64 para BQSquare e 48 para BlowingBubbles, BasketballPass e BasketballDrillText.

4.3.2 Realizar compressão utilizando codificador da equipe do projeto

Com a finalização dos treinamentos do codificador inter, foram realizadas compressões das sequências de vídeo do *dataset* JVET.

A equipe *Deep Codec* implementou um código que recebe um dado vídeo em formato *.yuv* e realiza a compressão gerando um arquivo de saída binário dessa codificação. Esse mesmo código também já realiza a reconstrução do vídeo comprimido, gerando assim uma outra saída em formato *.yuv* referente a essa reconstrução.

Em Figura 4.5 é apresentado o formato do código de configuração que irá definir os parâmetros necessários para a execução do módulo de compressão do codificador. Os parâmetros são caracterizados como:

- *command*: contém uma *string* que indica ao módulo que será realizada uma compressão de um vídeo informado como entrada;
- *input_data*: caminho em disco que contém o vídeo a ser codificado;
- *output_file*: caminho em disco onde será gravado o arquivo binário da codificação, assim como a reconstrução;
- *intra_checkpoint_dir e inter_checkpoint_dir*: os modelos intra e inter-quadros obtidos no treinamento do codificador de vídeo que se deseja utilizar na compressão;
- *intra_num_filters e inter_num_filters*: indica o número de filtros que foram utilizados no treinamento dos modelos intra e inter-quadros do codificador de vídeo;
- *intra_period*: indica o *GOP size* do vídeo que se deseja realizar a compressão;
- *num_frames*: número de quadros a serem codificados do vídeo de entrada.

Portanto, com o código de compressão, todas as sequências de vídeo apresentadas em 4.3.1 foram comprimidas utilizando o codificador neural. Com os resultados das compressões realizadas, foi possível obter métricas de distorção e qualidade para realizar análise de desempenho desse codificador.

```
{
    "command": "compress",
    "input_data": "jvet_sequences/class_d/
       BasketballPass_416x240_50fps_8bit_420.yuv",
    "output_file": "evaluation_class_d/modelo_1/
       BasketballPass_416x240_50fps_8bit_420.bin",
    "intra_checkpoint_dir": "modelos_intra/modelo_1",
    "intra_num_filters": 256,
    "inter_checkpoint_dir": "modelos_inter/modelo_1",
    "inter_num_filters": 256,
    "intra_period": 48,
    "num_frames": 500
}
```

Figura 4.5: Exemplo de código de configuração para realização de compressão utilizando o codificador neural.

4.3.3 Coleta da compressão realizada com os codificadores tradicionais HEVC e VVC

A equipe *Deep Codec* realizou a compressão das sequências de teste do JVET utilizando os codificadores tradicionais HEVC e VVC. As abordagens de compressão no modo *All Intra*, onde todos os quadros do vídeo são codificados utilizando somente predição do tipo intra, e no modo *Low-delay P*, onde se utiliza predição inter com quadros do tipo P, foram empregadas para se obter os resultados desses codificadores. Essas compressões seguiram as recomendações de condições de teste conforme mencionado anteriormente em 4.3.1.

O presente estudo tem por objetivo realizar comparações entre os resultados de compressão obtidos pelo codificador desenvolvido pela equipe *Deep Codec* e os obtidos por meio dos codificadores tradicionais HEVC e VVC. Dessa forma, foram coletadas e organizadas as compressões realizadas pelos codificadores tradicionais para posterior obtenção de métricas e comparações com o codificador neural.

4.3.4 Conversão de arquivos de compressão para 8 bits

Conforme mencionado em 4.3.1, as compressões realizadas nas sequências do *dataset* JVET, utilizando tanto o HEVC como o VVC, foram com profundidade de 10 bits.

O código desenvolvido pela equipe *Deep Codec* para o cálculo das métricas PSNR, SSIM e MS-SSIM, cujo funcionamento será descrito em 4.3.5, consegue apenas lidar com arquivos 8 bits, com isso, foi necessário realizar a conversão das compressões de 10 para 8 bits.

Portanto, foi desenvolvido um código de apoio para realizar essas conversões e permitir que o código de métricas fosse utilizado de forma adequada gerando os resultados para posterior análise. O código de conversão desenvolvido é apresentado no Apêndice A.

4.3.5 Obtenção de métricas para o codificador neural e codificadores tradicionais HEVC e VVC

O código implementado pela equipe *Deep Codec* contém um módulo para cálculo de métricas de uma forma facilitada. Para executar esse código, é necessário indicar os parâmetros listados no arquivo de configuração, apresentados na Figura 4.6.

De forma geral, é necessário informar os caminhos em disco do arquivo original do vídeo que foi comprimido, do arquivo binário de sua respectiva codificação, do arquivo da reconstrução desse vídeo em específico e, também, deve-se informar um caminho para que os arquivos de saída com as métricas sejam salvos.

As métricas calculadas são: MSE, PSNR, SSIM e MS-SSIM. Elas são calculadas tanto para os espaços de cor RGB quanto YUV, além de isoladamente para o canal Y, o qual representa a luminância.

Para os codificadores tradicionais, foram calculadas as métricas tanto para o modo de operação *All Intra* quanto para o *Low-delay P*, com tamanho do GOP conforme definido na seção 4.3.1, ao passo que, para o codificador neural, foram calculadas métricas somente para o modo com predição de quadros P.

```
{
    "original_video": "jvet_sequences/class_d/
       BasketballPass_416x240_50fps_8bit_420.yuv",
    "compressed_file": "evaluation_class_d/modelo_1/
       BasketballPass_416x240_50fps_8bit_420.bin",
    "reconstructed_video": "evaluation_class_d/modelo_1/
       BasketballPass_416x240_50fps_8bit_420_rec.yuv",
    "folder_to_save_files": "metrics/modelo_1/"
}
```

Figura 4.6: Exemplo de código de configuração para código de obtenção de métricas.

4.3.6 Comparação entre os resultados dos codificadores

Com os arquivos de métricas obtidos, conforme apresentado em 4.3.5, foram comparadas as métricas PSNR, SSIM e MS-SSIM para o canal de luminância Y.

Para uma montagem automatizada dos diversos gráficos de métricas para os variados vídeos selecionados das sequências do JVET, foi implementado um código que percorre pastas cujos nomes já representam o codificador que gerou os resultados de compressão sobre os quais foram calculadas as métricas. Assim, os nomes das pastas são aproveitados para a geração das legendas do gráfico. Esse código desenvolvido é apresentado em Apêndice B.

A apresentação das métricas em forma de gráficos permite uma análise acerca do desempenho dos codificadores em cada métrica específica, assim como permite compará-los. Os resultados das comparações serão apresentados na seção 5.3.

Capítulo 5

Resultados

Este capítulo iniciará apresentando resultados para o comportamento dos treinamentos das redes intra e inter-quadros, envolvendo desde os experimentos iniciais até o modelo final obtido para as redes.

Em sequência, para se atingir o objetivo principal deste estudo, o qual é analisar o desempenho do codificador neural frente aos codificadores tradicionais HEVC e VVC, serão apresentados os resultados das métricas PSNR, SSIM e MS-SSIM obtidas para as compressões realizadas pelo codificador baseado em *autoencoder* variacional implementado pela equipe *Deep Codec*. Os resultados dessas métricas serão confrontadas com aquelas obtidas pelos codificadores tradicionais, propiciando uma comparação crítica acerca do desempenho do codificador neural.

Os gráficos apresentados nas seções 5.1 e 5.2, que representam o comportamento dos treinamentos realizados, foram obtidos via *Tensorboard* [Tensorflow, 2021]. Para melhor visualização da tendência dos treinamentos, o parâmetro de suavidade da ferramenta utilizada foi configurado para seu valor máximo de $0,999$.

5.1 Rede Intra

Os treinamentos testes realizados para a rede neural intra serviram de base para a escolha de parâmetros que foram utilizados nos treinamentos definitivos do codificador de vídeo.

Nesta seção, serão apresentados e discutidos os resultados obtidos durante os treinamentos testes e definitivo referentes ao codificador intra-quadros.

5.1.1 Comparação entre números de filtros

Nessa etapa, foram gerados oito modelos intra, sendo quatro utilizando 192 filtros e os demais, 256 filtros. Os valores de *lambda* utilizados foram 10^{-1}, 10^{-2}, 10^{-3} e 10^{-4}.

Observa-se na Figura 5.1 que para os modelos de altas taxas, os quais possuem parâmetro *lambda* de maior valor, uma maior quantidade de filtros gera um maior valor de BPP.

Realizando-se uma referência entre os resultados dos modelos apresentados nas Tabelas 5.1 a 5.2, observa-se que o aumento de BPP vem acompanhado de uma redução na MSE, sendo mais evidente no modelo de *lambda* 10^{-1}, onde a BPP aumenta de $2,107$ para $2,437$ e a MSE reduz de $366,3$ para $359,0$ quando a quantidade de filtros aumenta. Porém, ainda assim esse comportamento ocorre para menores taxas embora em menor intensidade.

Apesar dos ganhos não serem tão expressivos para baixas taxas, eles ainda estão presentes. Visando atingir melhor desempenho do codificador neural implementado pela equipe *Deep Codec*, foi definida a utilização de 256 filtros para os treinamentos finais dos modelos intra e inter-quadros, uma vez que não representou impeditivo.

Tabela 5.1: BPP estimada obtida nos treinamentos testes dos modelos intra-quadros com a utilização de 192 e 256 filtros.

***Bits* por pixel (BPP)**		
Intra Lambda	**192 filtros**	**256 filtros**
10^{-1}	$2,107$	$2,437$
10^{-2}	$0,9928$	$1,013$
10^{-3}	$0,3038$	$0,317$
10^{-4}	$0,08912$	$0,09676$

Tabela 5.2: MSE obtida nos treinamentos testes dos modelos intra-quadros com a utilização de 192 e 256 filtros.

***Mean Squared Error* (MSE)**		
Intra Lambda	**192 filtros**	**256 filtros**
10^{-1}	$366,3$	$359,00$
10^{-2}	$401,4$	$401,70$
10^{-3}	$639,3$	$622,60$
10^{-4}	1327	1278

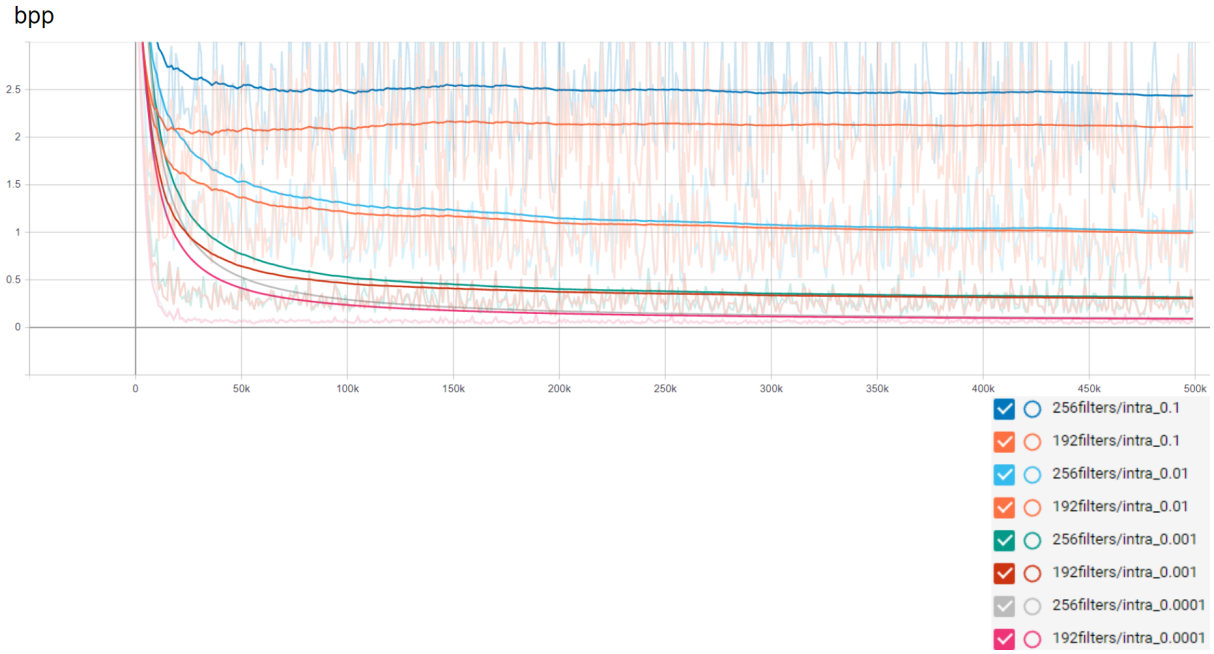

Figura 5.1: Visualização do comportamento da BPP em treinamentos intra-quadros realizados com número de filtros iguais a 192 e 256.

5.1.2 Análise dos valores de *lambda* e escolha da base de dados

Seis modelos intra-quadros foram obtidos, dentre os quais três foram treinados com o *PNG Dataset* e outros três foram treinados com o *UGC Dataset*.

Três valores de *lambda* foram utilizados: 10^{-1}, 10^{-3} e 10^{-5}. O comportamento do codificador intra, para cada um desses pontos de operação utilizando cada uma das bases de dados mencionadas, é apresentado nas Figuras 5.2 a 5.3. Com base nesses gráficos, observa-se que os modelos com parâmetro *lambda* de maior valor apresentam maiores taxas BPP e menores valores de MSE.

Mais detalhadamente nas Tabelas 5.3 a 5.4, modelos com *lambda* de 10^{-1} possuem valores de BPP considerados elevados para um codificador intra, mesmo que a MSE seja adequada. Por outro lado, os modelos de *lambda* igual a 10^{-5} possuem uma MSE muito elevada e uma BPP muito baixa, o que irá gerar imagens muito distorcidas prejudicando a desempenho do codificador intra, o qual será responsável pela geração dos quadros-âncoras a serem utilizados no modelo final do codificador de vídeo. Dessa forma, foi descartado o uso do parâmetro *lambda* de 10^{-5} e selecionou-se um valor de 10^{-2} para substituir o anterior maior valor de 10^{-1}. Com isso, foram definidos os parâmetros *lambda* de 10^{-2} e 10^{-3} como os pontos de operação a serem utilizados no modelo final do codificador intra-quadros.

Na Tabela 5.4, analisando a comparação entre modelos de mesmo *lambda* mas treinados com bases de dados distintas, é observado que os valores de MSE são bem semelhantes entre si. Com base na Tabela 5.3, os modelos treinados com *dataset* de imagens *.png* apresentam maior taxa. Vale ressaltar ainda que, conforme apresentado em 4.1.2, o *PNG Dataset* apresenta uma grande variedade de conteúdo, ao passo que o *UGC Dataset*, por ser uma base de vídeos, pode apresentar muitos quadros de transição e com pouco conteúdo, o que poderia prejudicar a predição intra em um teste real, embora tenha apresentado um bom comportamento durante o treinamento. Portanto, decidiu-se por utilizar o *PNG Dataset* para o treinamento definitivo dos modelos da rede intra-quadros.

Tabela 5.3: BPP estimada obtida nos treinamentos testes dos modelos intra-quadros utilizando-se tanto o *PNG Dataset* como também o *UGC Dataset*.

***Bits* por pixel (BPP)**		
Intra Lambda	**.png Dataset**	**UGC Dataset - .yuv**
10^{-1}	$2,721$	$2,417$
10^{-3}	$0,3517$	$0,299$
10^{-5}	$0,05216$	$0,05162$

Tabela 5.4: MSE obtida nos treinamentos testes dos modelos intra-quadros utilizando-se tanto o *PNG Dataset* como também o *UGC Dataset*.

***Mean Squared Error* (MSE)**		
Intra Lambda	**.png Dataset**	**UGC Dataset - .yuv**
10^{-1}	$474,3$	$510,10$
10^{-3}	$800,2$	$761,50$
10^{-5}	2952	2634

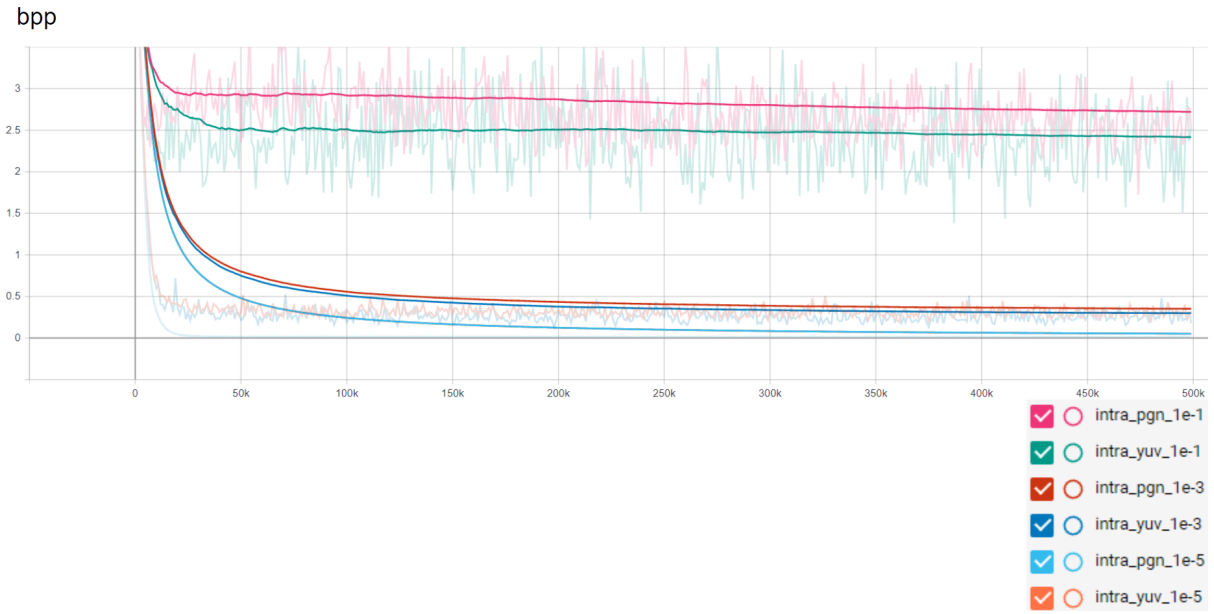

Figura 5.2: Visualização do comportamento da BPP em treinamentos intra-quadros realizados utilizando-se o *PNG Dataset* e também com o *UGC Dataset* composto de vídeos em formato *.yuv*.

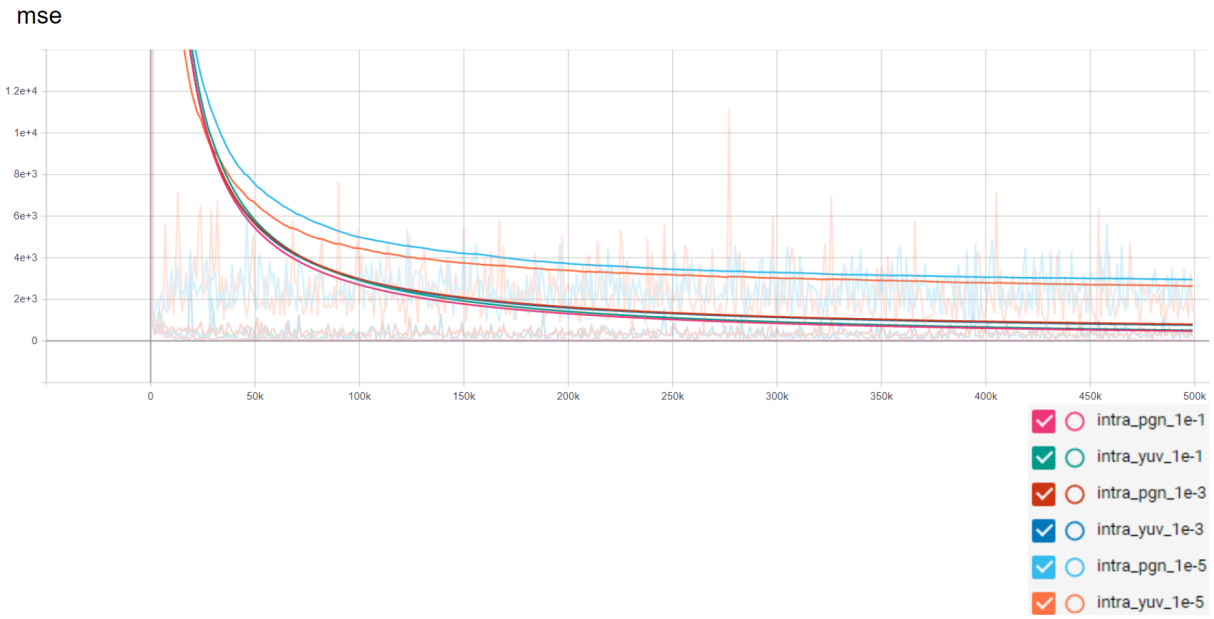

Figura 5.3: Visualização do comportamento da MSE em treinamentos intra-quadros realizados utilizando-se o *PNG Dataset* e também com o *UGC Dataset* composto de vídeos em formato *.yuv*.

5.1.3 Treinamentos finais dos modelos intra-quadros

Com a definição por utilizar 256 filtros, parâmetros *lambda* de 10^{-2} e 10^{-3} e o *PNG Dataset*, os treinamentos finais dos modelos intra-quadros foram realizados por 1 milhão de iterações.

Comparando o valor da BPP do modelo final de *lambda* 10^{-3}, apresentado na Tabela 5.5, com o respectivo valor obtido no treinamento teste usando o *PNG Dataset* apresentado na Tabela 5.3, verifica-se que ocorreu uma leve redução indicando uma melhor convergência do treinamento realizado por mais iterações. Isso se confirma ao se comparar uma maior redução dos valores de MSE desse mesmo modelo, passando de $800,2$ em Tabela 5.4 para $505,6$ em Tabela 5.6.

Além disso, os valores de BPP (Tabela 5.5) e de MSE (Tabela 5.6) para o modelo de maior *lambda* igual 10^{-2} apresenta uma taxa não muito elevada e uma menor distorção comparado ao modelo de 10^{-1} do treinamento realizado em 5.1.2.

Dessa forma, verifica-se que os treinamentos finais dos modelos intra-quadros apresentaram boa convergência e taxas dentro de um intervalo que não gera desperdício de taxa e também não apresenta grande distorção, indicando uma escolha acertada para os parâmetros *lambda* dos modelos.

Esses modelos finais intra-quadros serão utilizados para os treinamentos finais inter-quadros, determinando assim os pontos finais de execução do codificador de vídeo neural implementado pela equipe *Deep Codec*.

Tabela 5.5: BPP estimada obtida nos treinamentos definitivos dos modelos intra-quadros.

Intra Lambda	***Bits* por pixel (BPP)**
10^{-2}	$1,125$
10^{-3}	$0,3116$

Tabela 5.6: MSE obtida nos treinamentos definitivos dos modelos intra-quadros.

Intra Lambda	***Mean Squared Error* (MSE)**
10^{-2}	$230,8$
10^{-3}	$505,6$

5.2 Rede Inter

Os treinamentos testes realizados para a rede neural inter-quadros permitiram uma análise de comportamento dos modelos utilizando combinações entre *lambdas* intra e inter. Com isso, foi possível estabelecer os parâmetros *lambda* da rede inter os quais foram utilizados nos treinamentos finais.

Nesta seção, serão apresentados os resultados para os modelos de teste inter-quadros e, posteriormente, os resultados obtidos nos treinamentos dos modelos finais do codificador.

5.2.1 Análise das combinações dos *lambdas* Intra e Inter

O treinamento teste intra apresentado em 4.1.3, que utilizou 128 filtros e foi realizado com o *UGC Dataset*, gerou três modelos intra que foram utilizados para realização do treinamento teste dos modelos inter-quadros. O objetivo do treinamento teste inter foi o de analisar o efeito das combinações entre os *lambdas* intra e inter, de forma a permitir a escolha dos parâmetros *lambda* definitivos para o treinamento final dos modelos inter.

Nas Figuras 5.4 a 5.5, observa-se que para um dado *lambda* intra de valor elevado igual a 10^{-1}, uma redução no valor do *lambda* inter de um valor elevado igual a 10^{-1} para um intermediário de 10^{-3} gera uma brusca queda do valor da BPP e um aumento significativo da MSE, embora não muito discrepante. Isso é evidenciado nas Tabelas 5.7 a 5.8, onde a BPP reduz de $1,004$ para $0,2059$ e a MSE tem um aumento de $172,0$ para $297,6$.

Continuando a analisar o efeito da mudança no *lambda* inter para um dado *lambda* intra, observa-se nas Figuras 5.6 a 5.7 que, para um valor de 10^{-3} do *lambda* intra, uma redução no valor do *lambda* inter provocou uma brusca queda no BPP e um aumento grande na MSE. Assim, verifica-se que quanto menor o valor do *lambda* intra, uma mudança no valor de *lambda* inter provoca um aumento cada vez mais significativo da MSE, ocasionando maior distorção. Desse modo, um valor muito baixo para *lambda* intra não seria adequado, o que reforça a decisão tomada em 5.1.2 para os valores de 10^{-2} e 10^{-3} para o *lambda* intra. Além disso, os valores nas Tabelas 5.7 a 5.8 para os modelos com *lambda* inter de 10^{-5} indicam que esses modelos apresentam uma taxa muito reduzida e elevada distorção, o que torna inviável o uso dos modelos com esse valor de parâmetro.

Nas Figuras 5.8 a 5.9, observa-se que para um valor intermediário de *lambda* inter, uma redução no valor de *lambda* intra gera um leve aumento na BPP e praticamente mantém a mesma MSE, o que indica que o modelo inter gasta um pouco mais de bits para tentar manter a distorção mais estável. Entretanto, nas Figuras 5.10 a 5.11 observa-se que quando o *lambda* inter possui valor mais baixo, a robustez em manter a distorção ao se reduzir o valor do *lambda* intra não se mantém e a MSE apresenta um aumento brusco.

Dessa forma, valores intermediários para os parâmetros *lambda* dos modelos inter geram melhores comportamentos, pois valores elevados irão gerar altas taxas o que não é muito desejável em um codificador de vídeo e valores muito baixos diminuem a robustez da rede em manter a distorção ao se alterar os parâmetros do codificador intra-quadros. Assim, os parâmetros *lambda* de $5 \cdot 10^{-3}$, 10^{-3} e $5 \cdot 10^{-4}$ foram definidos para o treinamento dos modelos finais inter-quadros.

Tabela 5.7: BPP estimada obtida nos treinamentos dos modelos inter-quadros utilizando diferentes modelos intra-quadros como entrada.

***Bits* por pixel (BPP)**			
Intra Lambda	**Inter Lambda**		
	10^{-1}	10^{-3}	10^{-5}
10^{-1}	$1,004$	$0,2059$	$0,08292$
10^{-3}	$1,145$	$0,2397$	$0,08492$
10^{-5}	$1,292$	$0,2943$	$0,08588$

Tabela 5.8: MSE obtida nos treinamentos dos modelos inter-quadros utilizando diferentes modelos intra-quadros como entrada.

***Mean Squared Error* (MSE)**			
Intra Lambda	**Inter Lambda**		
	10^{-1}	10^{-3}	10^{-5}
10^{-1}	$172,0$	$297,6$	1711
10^{-3}	$181,1$	$328,4$	1707
10^{-5}	$230,2$	$330,2$	2413

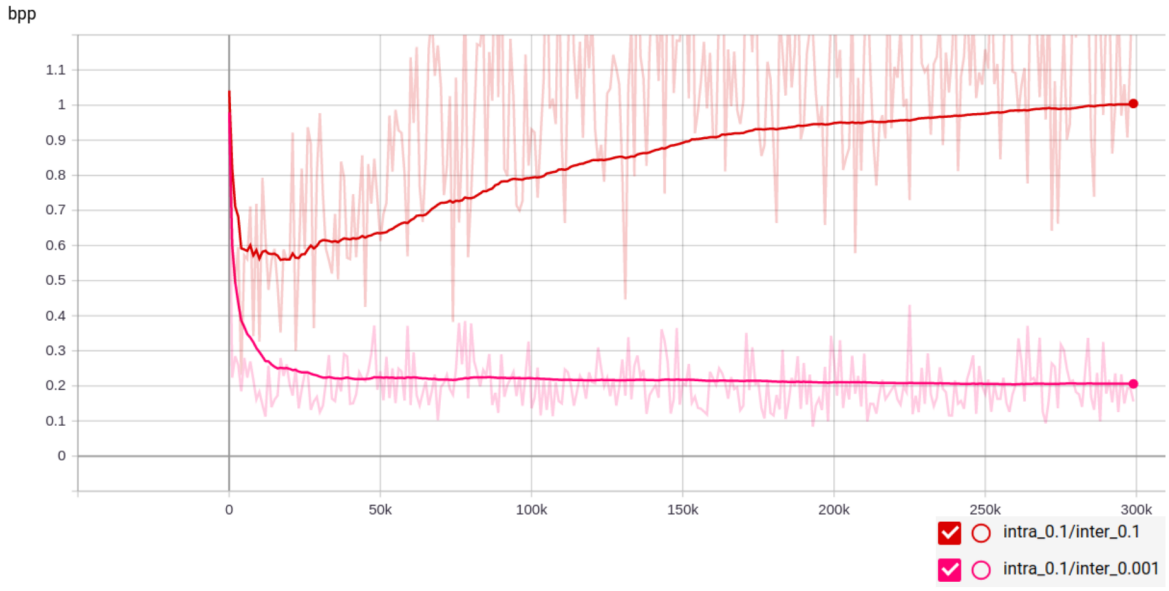

Figura 5.4: BPP estimada de diferentes modelos inter-quadros treinados com *lambdas* de 10^{-1} e 10^{-3} com base no mesmo modelo intra-quadros treinado com *lambda* de 10^{-1}.

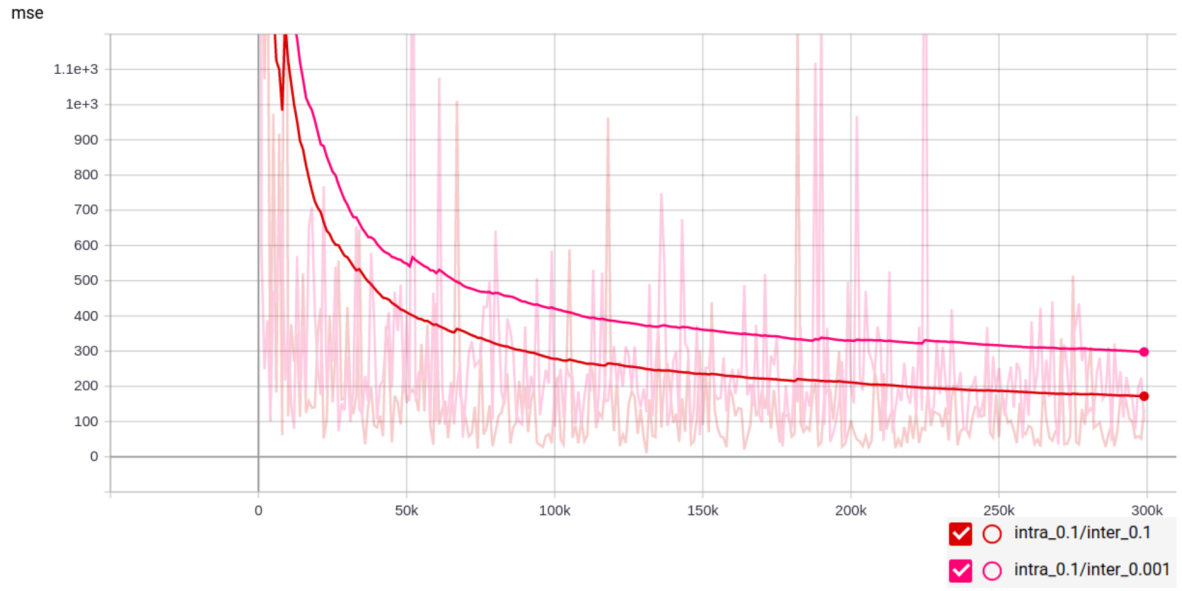

Figura 5.5: MSE para diferentes modelos inter-quadros treinados com *lambdas* de 10^{-1} e 10^{-3} com base no mesmo modelo intra-quadros treinado com *lambda* de 10^{-1}.

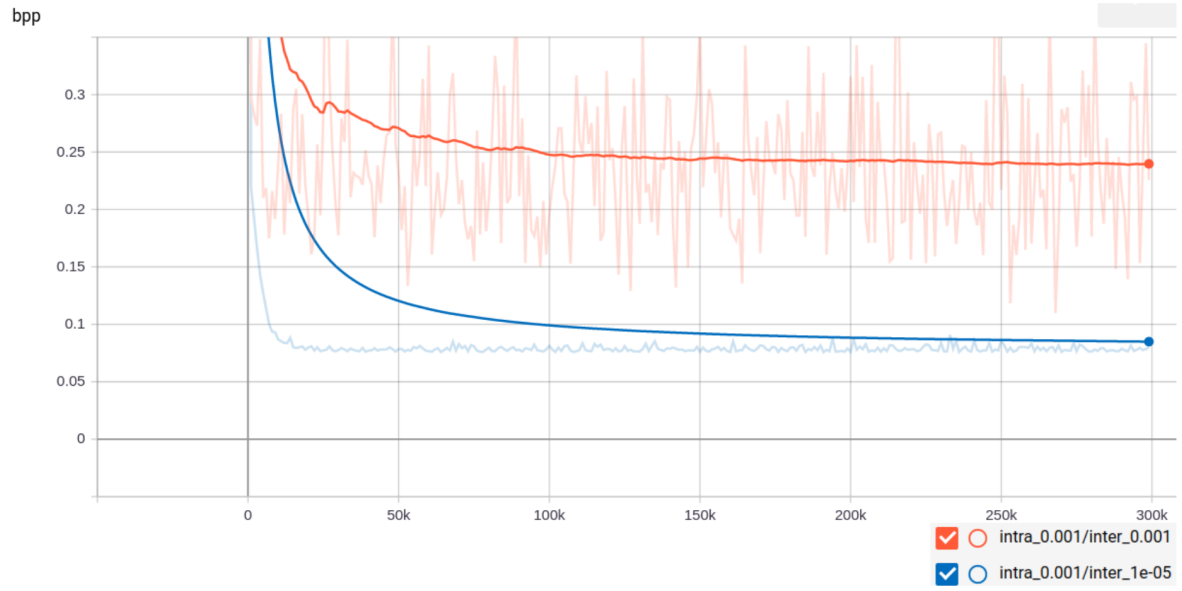

Figura 5.6: BPP estimada de diferentes modelos inter-quadros treinados com *lambdas* de 10^{-3} e 10^{-5} com base no mesmo modelo intra-quadros treinado com *lambda* de 10^{-3}.

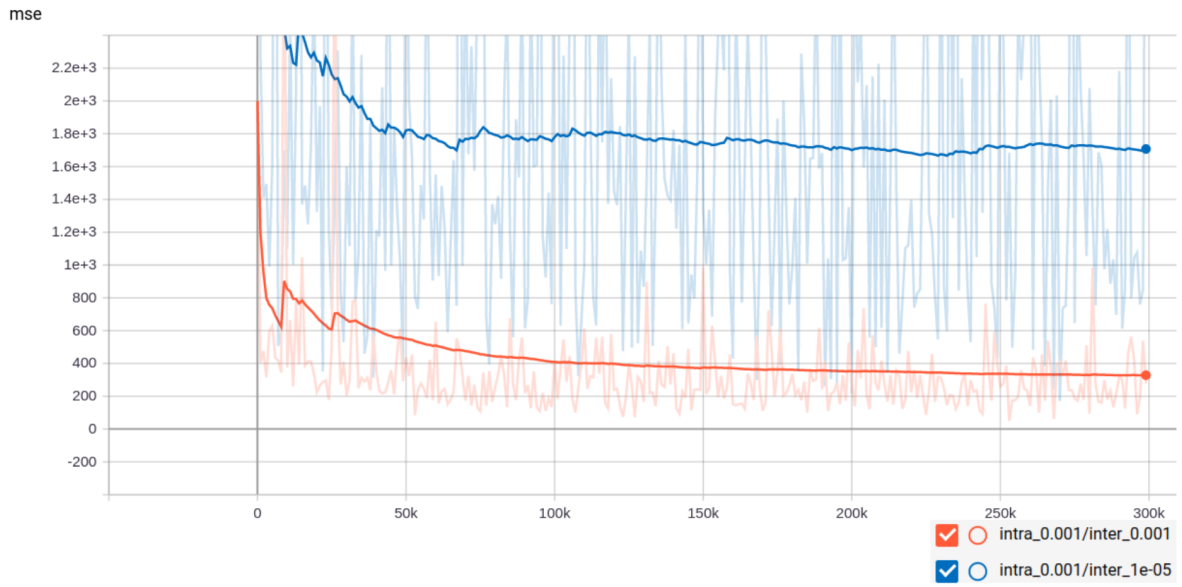

Figura 5.7: MSE para diferentes modelos inter-quadros treinados com *lambdas* de 10^{-3} e 10^{-5} com base no mesmo modelo intra-quadros treinado com *lambda* de 10^{-3}.

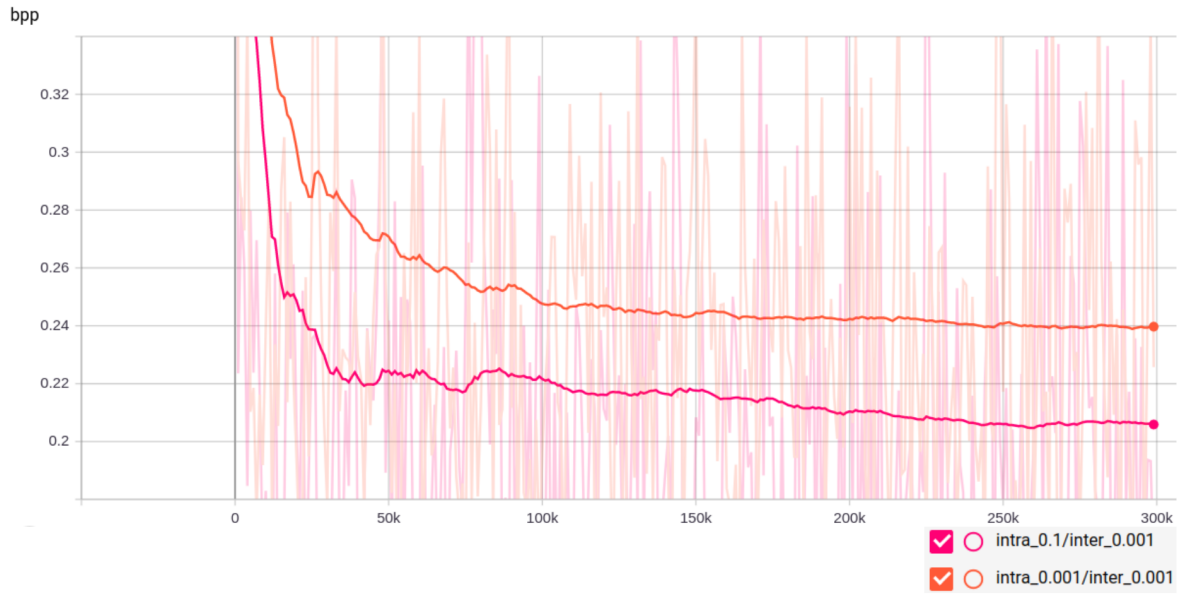

Figura 5.8: BPP estimada de diferentes modelos inter-quadros treinados com mesmo *lambda* de 10^{-3} e com base em modelos intra-quadros distintos treinados com *lambdas* de 10^{-1} e 10^{-3}.

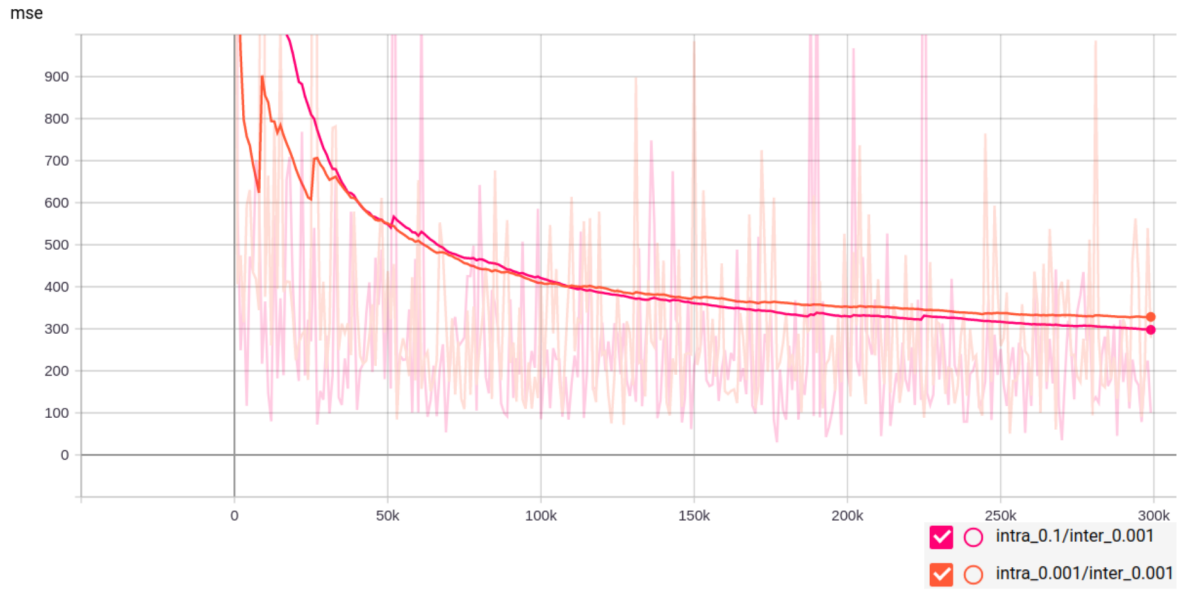

Figura 5.9: MSE para diferentes modelos inter-quadros treinados com mesmo *lambda* de 10^{-3} e com base em modelos intra-quadros distintos treinados com *lambdas* de 10^{-1} e 10^{-3}.

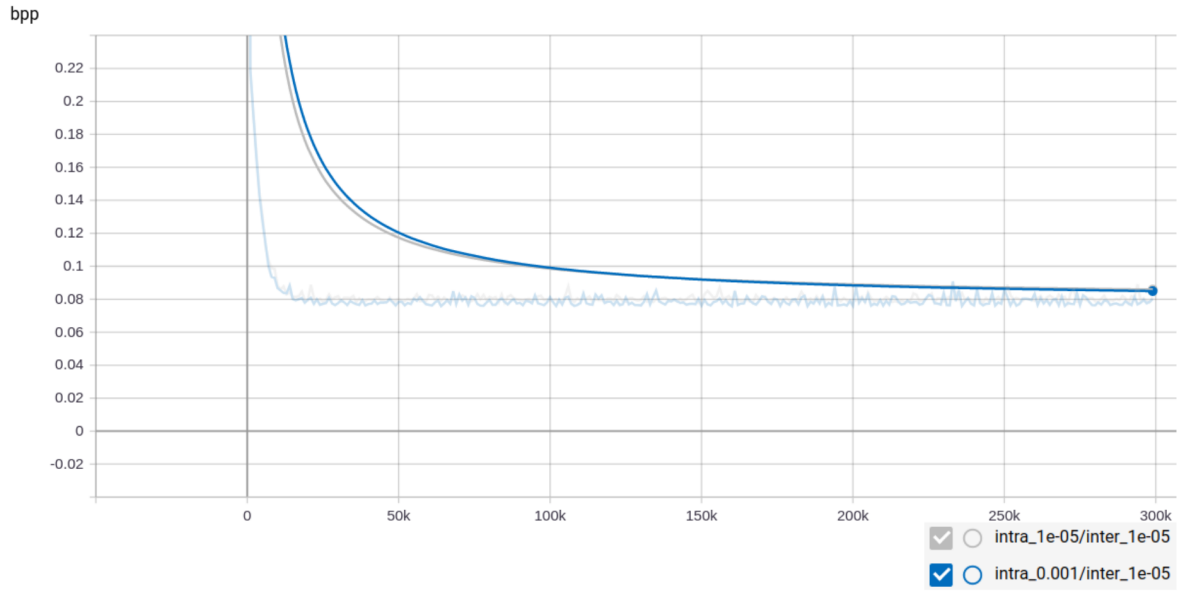

Figura 5.10: BPP estimada de diferentes modelos inter-quadros treinados com mesmo *lambda* de 10^{-5} e com base em modelos intra-quadros distintos treinados com *lambdas* de 10^{-3} e 10^{-5}.

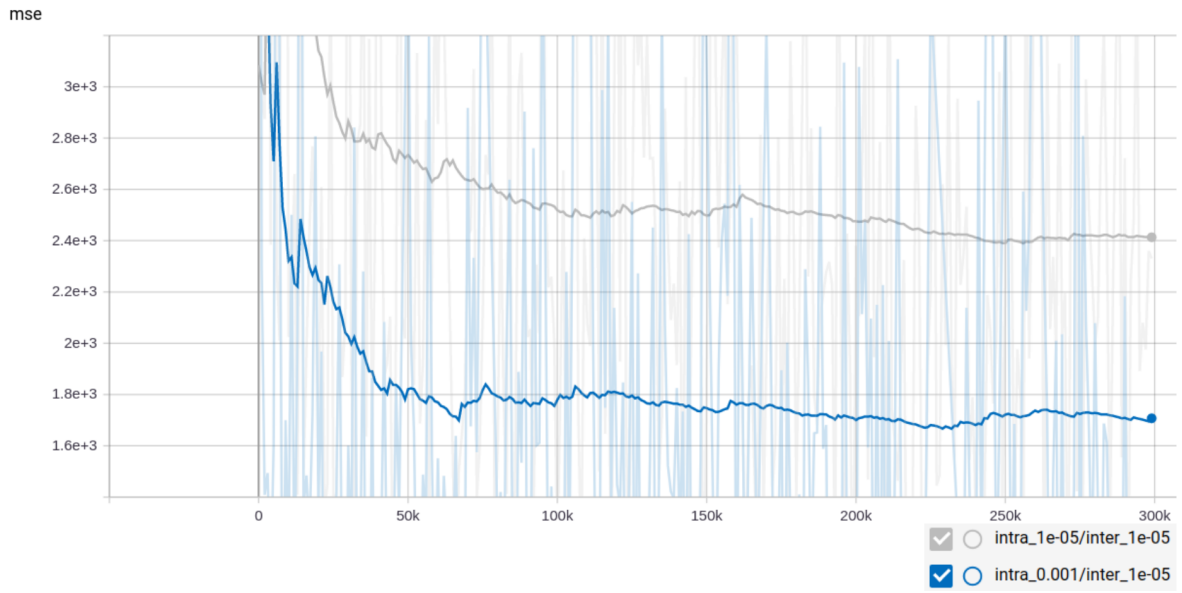

Figura 5.11: MSE para diferentes modelos inter-quadros treinados com mesmo *lambda* de 10^{-5} e com base em modelos intra-quadros distintos treinados com *lambdas* de 10^{-3} e 10^{-5}.

5.2.2 Treinamentos finais dos modelos inter-quadros

Os treinamentos finais inter-quadros originaram seis modelos. O treinamento ocorreu por um total de 500 mil iterações, o qual foi dividido em duas etapas: uma de 300 mil e outra com 200 mil iterações.

Na primeira etapa do treinamento, cujo comportamento da MSE é apresentado na Figura 5.12 e de forma resumida na Tabela 5.9, observou-se que a MSE divergiu para o modelo de *lambdas* inter e intra iguais a 10^{-3}, aumentando seu valor durante o intervalo de tempo entre 80 e 110 mil iterações aproximadamente. Dessa maneira, após o encerramento das 300 mil iterações, o modelo ainda não tinha atingido um nível de convergência satisfatório e foi necessário continuar o treinamento por mais 200 mil iterações.

Após a realização total das 500 mil iterações, o modelo apresentou convergência conforme observado nas Figuras 5.13 a 5.14. Os valores finais para BPP e MSE são apresentados nas Tabelas 5.9 a 5.10. Com base nesse valores finais, os modelos com maior valor de *lambda* inter apresentam maiores taxas e menor distorção. Além disso, para um mesmo valor de *lambda* inter, uma redução no valor de *lambda* intra leva a um leve aumento na taxa e um aumento controlado no valor da MSE, comportamento em conformidade com o apresentado em 5.2.1.

Nas Tabelas 5.9 a 5.10, observa-se que o modelo cujo ponto de operação possui *lambda* inter igual a 10^{-3} e intra 10^{-2} apresenta uma taxa BPP reduzida mas mantendo uma MSE em um patamar baixo. Seu valor de BPP igual a $0,2704$ equivale a 55% do valor de $0,4914$ e sua MSE de $197,9$ é cerca de 10% maior que $180,5$, comparando-se com o modelo de menor valor de MSE cujo *lambda* inter é igual a $5 \cdot 10^{-3}$ e intra 10^{-2}. Dessa forma, esse ponto apresenta melhor desempenho dentre os modelos treinados pois consegue manter baixa distorção utilizando uma quantidade reduzida de taxa, o que é essencial em um codificador de vídeo.

Com a finalização dos treinamentos dos modelos inter-quadros, as compressões para avaliação de desempenho do codificador foram realizadas conforme indicado em 4.3.2. Depois foram obtidas as métricas de acordo com o apresentado em 4.3.5 cujos resultados serão apresentados na seção 5.3.

Tabela 5.9: BPP estimada obtida nos treinamentos dos modelos inter-quadros após $300K$ e $500K$ iterações utilizando diferentes modelos intra-quadros como entrada.

***Bits* por pixel (BPP)**						
Intra Lambda	**Inter Lambda**					
	$5 \cdot 10^{-3}$		10^{-3}		$5 \cdot 10^{-4}$	
	$300K$	$500K$	$300K$	$500K$	$300K$	$500K$
10^{-2}	$0,5090$	$0,4914$	$0,2842$	$0,2704$	$0,2341$	$0,2247$
10^{-3}	$0,5855$	$0,5859$	$0,3004$	$0,2967$	$0,2403$	$0,2287$

Tabela 5.10: MSE obtida nos treinamentos dos modelos inter-quadros após $300K$ e $500K$ iterações utilizando diferentes modelos intra-quadros como entrada.

***Mean Squared Error* (MSE)**						
Intra Lambda	**Inter Lambda**					
	$5 \cdot 10^{-3}$		10^{-3}		$5 \cdot 10^{-4}$	
	$300K$	$500K$	$300K$	$500K$	$300K$	$500K$
10^{-2}	181	$180,5$	$272,20$	$197,90$	$353,5$	$272,4$
10^{-3}	243	$184,9$	$519,5$	$292,3$	$411,8$	$362,7$

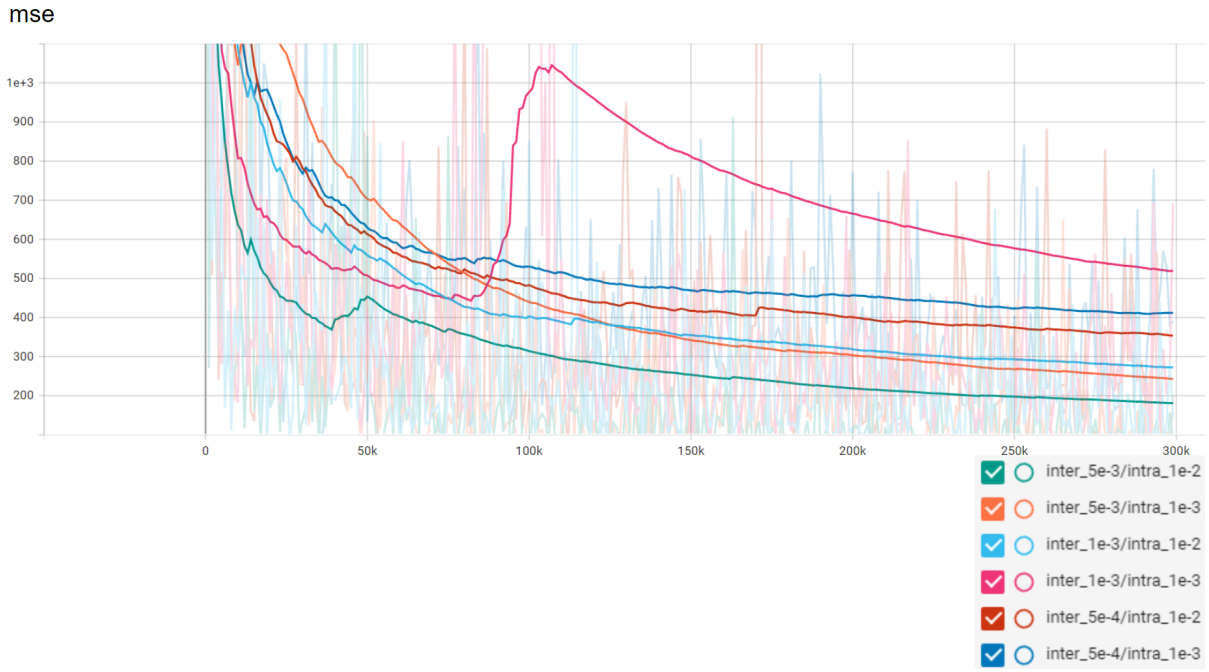

Figura 5.12: Visualização do comportamento da MSE nos treinamentos finais inter-quadros durante primeira etapa de 300 mil iterações, onde ocorre divergência de um dos modelos.

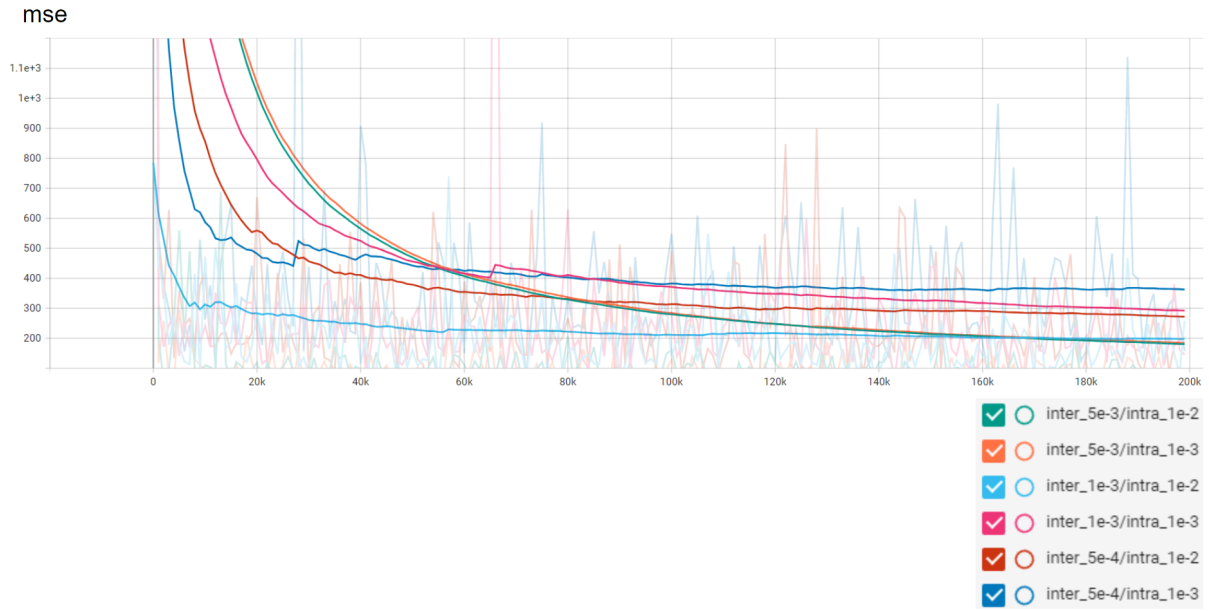

Figura 5.13: Visualização do comportamento da MSE nos treinamentos finais inter-quadros na continuação do treinamento por mais 200 mil iterações.

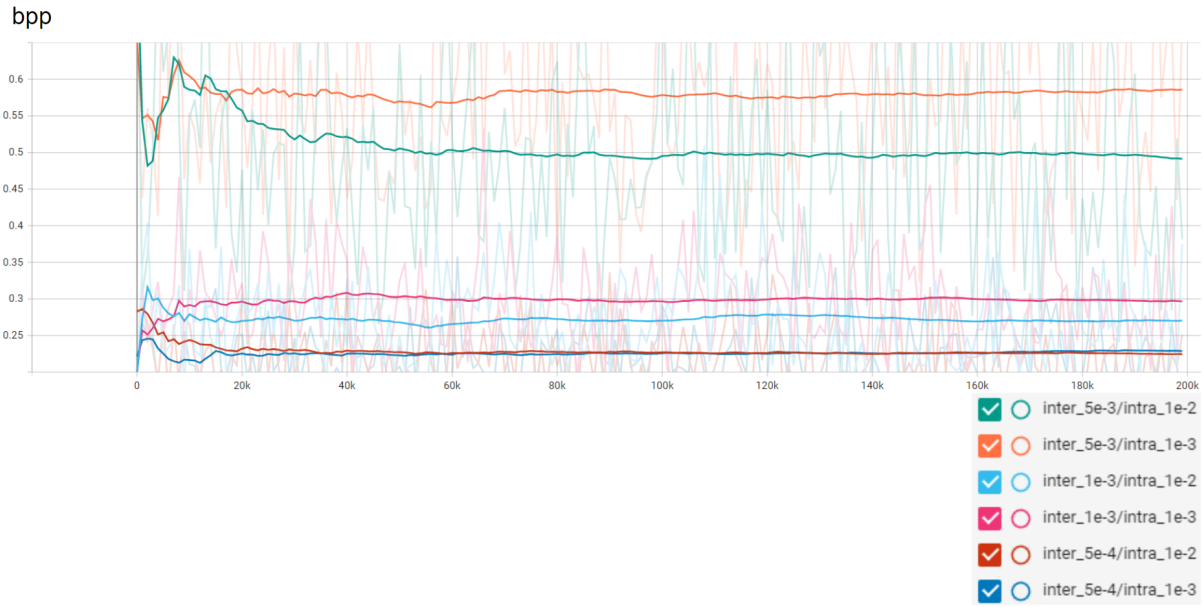

Figura 5.14: Visualização do comportamento da BPP nos treinamentos finais inter-quadros durante a última etapa do treinamento, totalizando 500 mil iterações.

5.3 Resultados das métricas e comparação de desempenho frente aos codificadores tradicionais

Com os modelos inter-quadros totalmente treinados, as sequências de vídeos do JVET foram submetidas à compressão utilizando-se cada um desses modelos. Dessa maneira, realizando-se as compressões sob diferentes taxas foi possível obter as métricas objetivas PSNR, SSIM e MS-SSIM. Essas métricas também foram calculadas para os resultados de compressão obtidos para os codificadores tradicionais HEVC e VVC. Assim, foram realizadas comparações de taxa e distorção apresentadas nas Figuras 5.15 a 5.18.

Nesta seção, os resultados apresentados nos gráficos de taxa e distorção são para as métricas calculadas utilizando-se o canal de luminância Y. De forma simplificada, os pontos mais elevados e localizados mais a esquerda indicam um desempenho superior.

5.3.1 PSNR

Com base nos resultados apresentados para a PSNR - Y na Figura 5.15, é evidente que o modo *Low-delay P* dos codificadores tradicionais HEVC e VVC apresenta desempenho muito superior não somente ao do codificador neural, mas também ao modo *All Intra*

desses codificadores. Essa mesma conclusão é valida para as demais métricas SSIM - Y e MS-SSIM - Y.

Em grande parte das sequências, o desempenho do codificador neural é inferior ao dos codificadores tradicionais. No gráfico (c) da Figura 5.15, é verificado um desempenho similar entre o codificador neural e o HEVC *All Intra*.

Na Figura 5.16, o comportamento da PSNR é apresentado quadro a quadro para a sequência de vídeo *Blowing Bubbles* codificada por cada um dos seis modelos finais inter-quadros. Observa-se que no modelo (b), onde o *lambda* intra-quadros de 10^{-3} é inferior ao *lambda* inter-quadros de $5 \cdot 10^{-3}$, os quadros do tipo I apresentam PSNR inferior aos quadros preditos do tipo P. Nesse modelo, a *PSNR* apresentou o maior valor médio, pois os inter-quadros apresentaram valores maiores e com certa constância. Nos demais modelos, os intra-quadros apresentam picos mais elevados de PSNR frente aos inter-quadros das sequências dos GOPs. Nesses modelos, como os inter-quadros apresentam menor PSNR e são maioria em relação aos intra-quadros, a PSNR média acaba apresentando valores mais baixos. Além disso, pode-se observar que no modelo (d), onde os *lambdas* intra e inter-quadros possuem mesmo valor, a diferença de *PSNR* entre quadros do tipo I e do tipo P dentro de um mesmo GOP é de menor intensidade comparada a dos demais modelos, e alguns inter-quadros apresentam PSNR superior ao quadro intra pertencente ao mesmo GOP. De forma geral, a PSNR média de um determinado modelo tende a seguir a média dos inter-quadros, os quais são maioria e apresentam valores com significativa diferença em relação ao intra-quadro que inicia o GOP ao qual pertencem.

5.3.2 SSIM

Na Figura 5.17, observa-se que, para as sequências (b) e (c), o codificador neural teve um desempenho mais comparável ao modo *All Intra* dos codificadores tradicionais. Em (b) o desempenho foi comparável ao do HEVC e em (c) foi superior ao do HEVC e próximo ao do VVC.

A métrica SSIM envolve uma abordagem voltada à análise de características estruturais, diferentemente da métrica PSNR. Com isso, a melhora de desempenho apresentada pelo codificador neural analisando-se a métrica SSIM - Y indica que os resultados obtidos pelo codificador neural apresentam uma certa qualidade perceptual que a métrica PSNR não foi capaz de demonstrar nos gráficos da Figura 5.15.

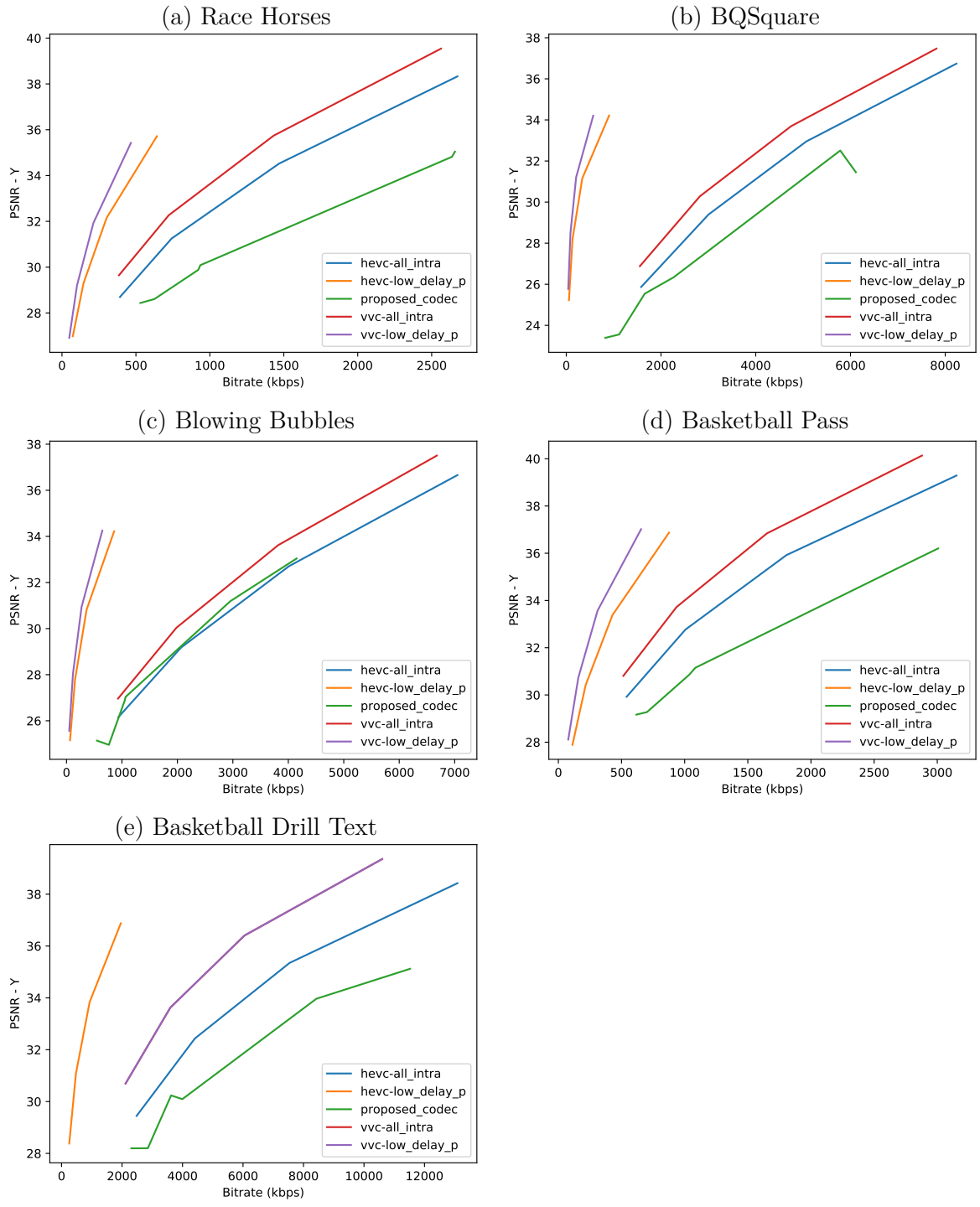

Figura 5.15: Gráficos com a métrica PSNR calculada para o canal de luminância Y de sequências de vídeo do JVET que foram comprimidas. São apresentados os resultados para o codificador implementado pela equipe *Deep Codec* e para os modos *All Intra* e *Low-delay P* dos codificadores tradicionais HEVC e VVC.

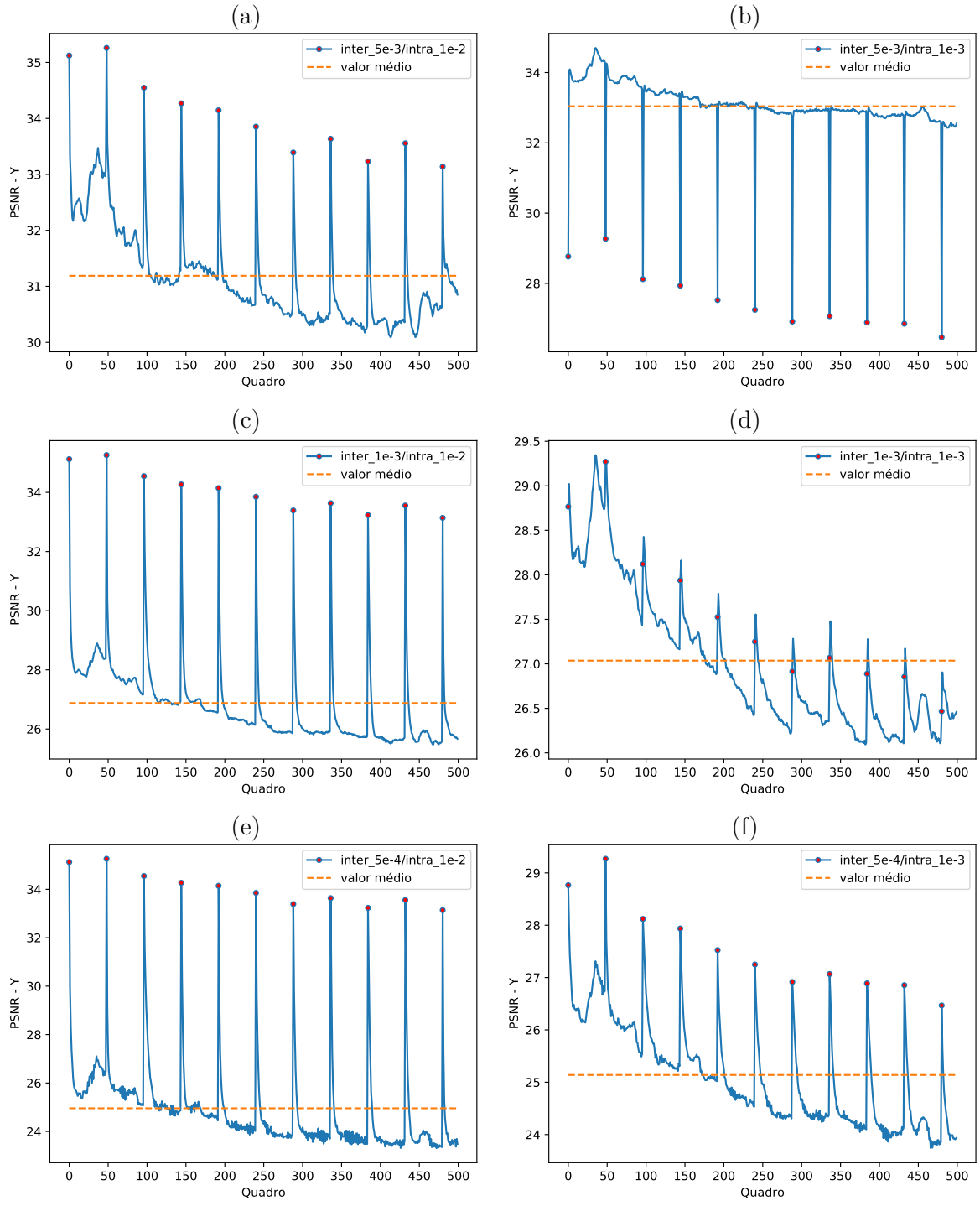

Figura 5.16: Gráficos com a métrica PSNR quadro a quadro para o canal de luminância Y da sequência de vídeo *Blowing Bubbles*. Os pontos em vermelho indicam os intra-quadros que iniciam a sequência de cada GOP, cujo tamanho é de 48 para esse vídeo. As figuras de (a) a (f) apresentam cada um dos seis modelos finais inter-quadros.

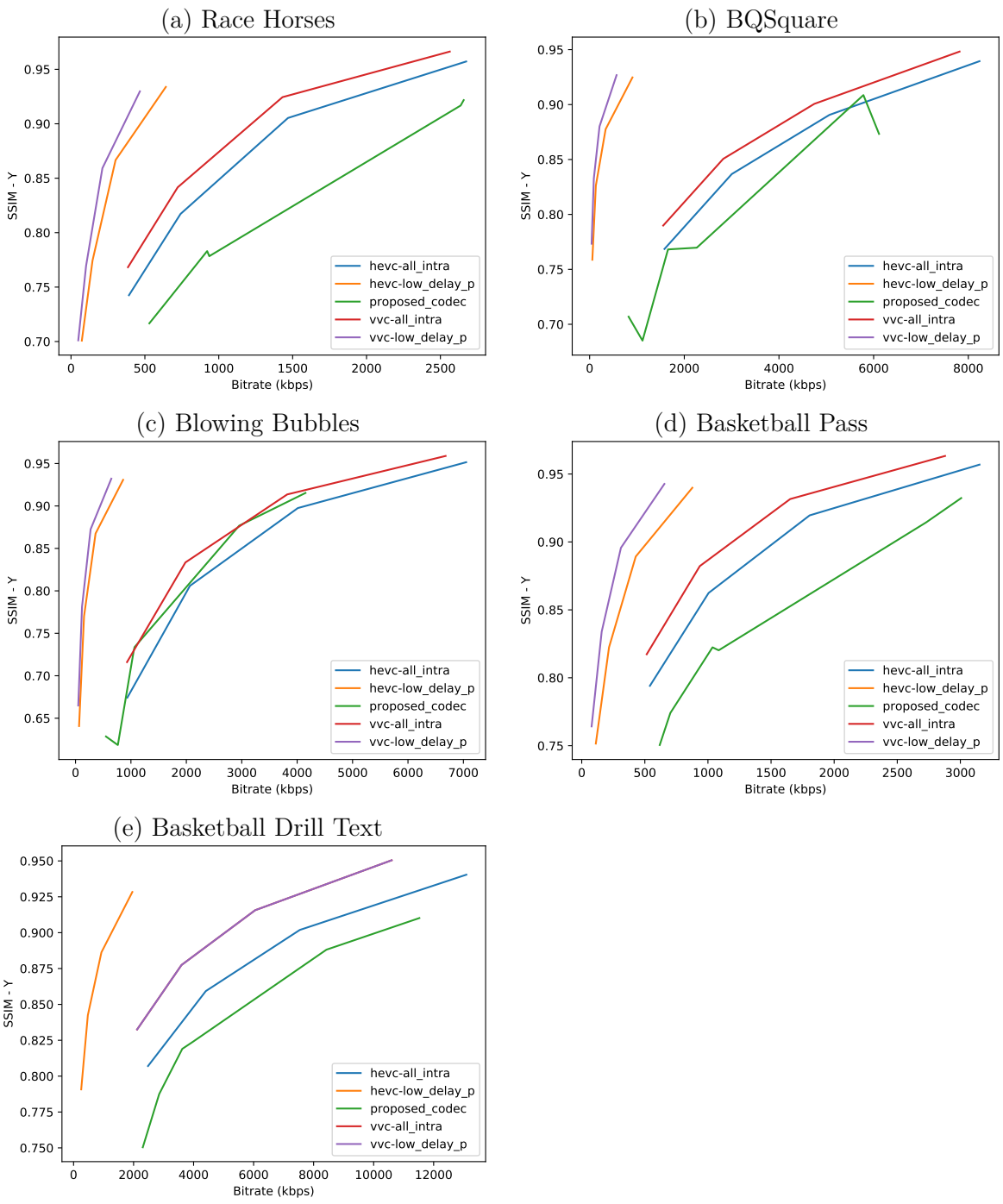

Figura 5.17: Gráficos com a métrica SSIM calculada para o canal de luminância Y de sequências de vídeo do JVET que foram comprimidas. São apresentados os resultados para o codificador implementado pela equipe *Deep Codec* e para os modos *All Intra* e *Low-delay P* dos codificadores tradicionais HEVC e VVC.

5.3.3 MS-SSIM

A métrica MS-SSIM é uma extensão da SSIM envolvendo uma abordagem multiescalar, sendo então uma métrica que também leva em consideração a qualidade perceptual do quadro de vídeo, extraindo informações estruturais.

Com base na Figura 5.18, observa-se que o codificador de vídeo apresentou um desempenho comparável ao modo *All Intra* do HEVC para a sequência (e). Em (c) o desempenho apresentado foi superior ao HEVC *All Intra* na maioria dos pontos e em alguns deles foi superior até em relação ao VVC *All Intra*. Em (b), o desempenho apresentou tendência comparável ao HEVC *All Intra* e em dois pontos atingiu desempenho sobre a curva do VVC *All Intra*.

Portanto, o codificador neural apresentou um desempenho mais competitivo em relação ao modo *All Intra* do HEVC e VVC ao se considerar a métrica MS-SSIM, indicando uma qualidade perceptual dos vídeos codificados comparável a esse modo de execução dos codificadores tradicionais.

Com o intuito de verificar visualmente a qualidade das compressões, a Figura 5.19 exibe um dado quadro selecionado na sequência de vídeo *Blowing Bubbles* o qual foi codificado pelos codificadores tradicionais e pelo codificador neural. Observa-se que o codificador neural analisado apresenta uma boa representação para o quadro em questão, o qual foi codificado como quadro do tipo P. Detalhes nos objetos, na parede em plano de fundo e reflexos nas bolhas de sabão são percebidos satisfatoriamente em relação ao quadro original e contribuem para a percepção geral do quadro reconstruído. Dessa forma, o codificador analisado apresenta um boa qualidade perceptual nas reconstruções dos vídeos codificados.

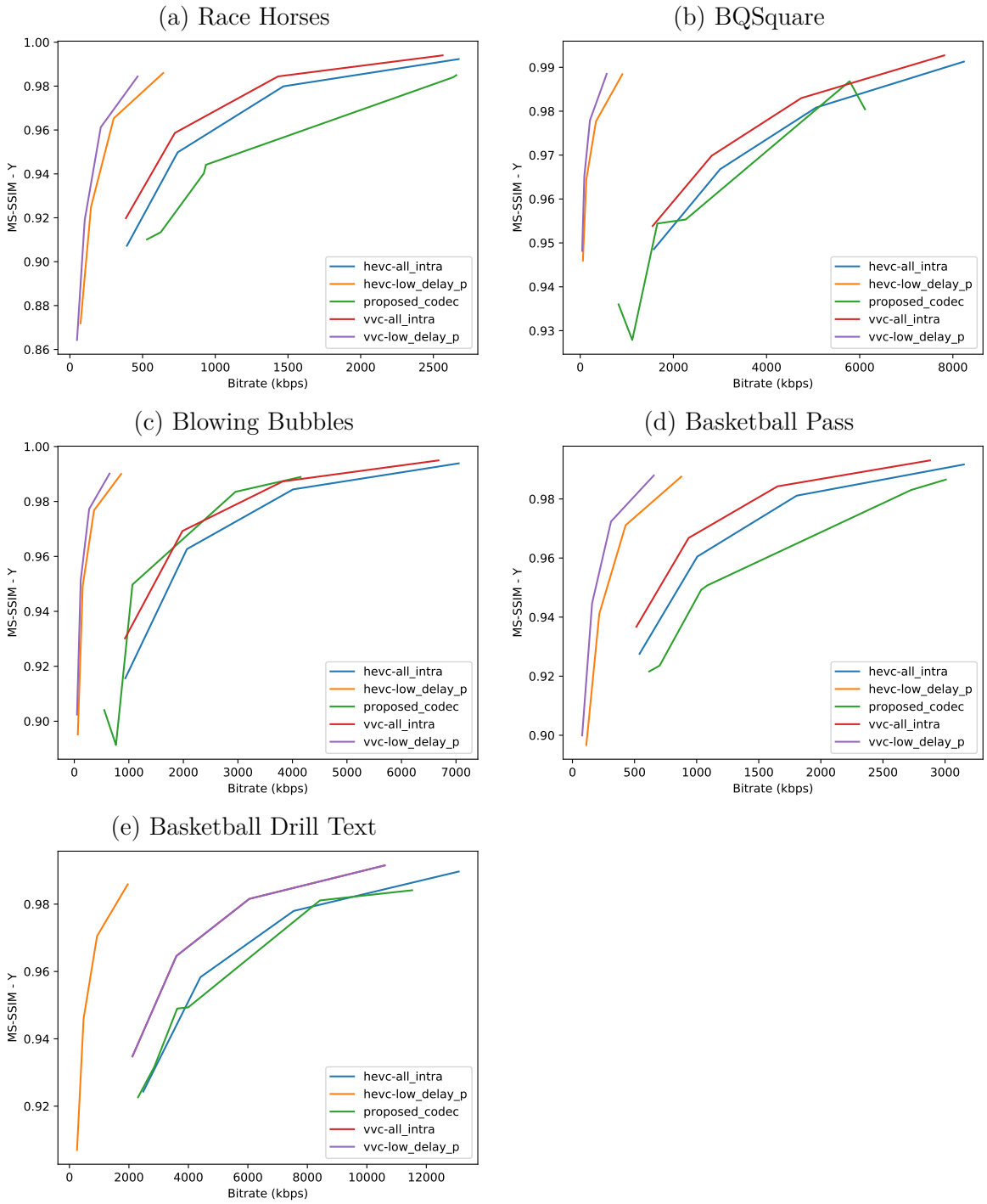

Figura 5.18: Gráficos com a métrica MS-SSIM calculada para o canal de luminância Y de sequências de vídeo do JVET que foram comprimidas. São apresentados os resultados para o codificador implementado pela equipe *Deep Codec* e para os modos *All Intra* e *Low-delay P* dos codificadores tradicionais HEVC e VVC.

(a) Original

(b) Codificador neural
PSNR: 34,243; SSIM: 0,918; MS-SSIM: 0,988

(c) HEVC *All Intra*
PSNR: 37,192; SSIM: 0,946; MS-SSIM: 0,993

(d) HEVC *Low-delay P*
PSNR: 36,295; SSIM: 0,938; MS-SSIM: 0,991

(e) VVC *All Intra*
PSNR: 38,032 , SSIM: 0,954 , MS-SSIM: 0,994

(f) VVC *Low-delay P*
PSNR: 34,669; SSIM: 0,925; MS-SSIM: 0,988

Figura 5.19: Quadro selecionado do vídeo *Blowing Bubbles* para uma comparação de qualidade visual entre os codificadores. Foram selecionados os modelos que geraram maiores valores para as métricas.

5.3.4 Resultados com modo de operação totalmente intra-quadros (*All Intra*)

De forma complementar aos resultados apresentados neste estudo, é interessante observar o comportamento do codificador neural com modo de configuração totalmente intra-quadros (do inglês *All Intra*). Esse modo de operação codifica todos os quadros utilizando apenas a abordagem de compressão intra-quadros, explorando somente redundância espacial.

Os resultados para o modo de operação *All Intra* foram obtidos no estudo [de Oliveira et al., 2021], no qual o presente autor fez parte juntamente com de Oliveira M., Jung H., Júnior N., Silva R., Peixoto E., Macchiavello B., Hung E., Testoni V. e Freitas P., integrantes da equipe *Deep Codec*. Os resultados obtidos neste presente estudo também foram apresentados em [de Oliveira et al., 2021].

O modo *All Intra* do codificador neural gerou resultados a partir de modelos da rede intra-quadros treinados com quatro valores de *lambda*: 10^{-2}, $5 \cdot 10^{-3}$, 10^{-3} e 10^{-4}. O treinamento desses modelos seguiu abordagem semelhante à deste estudo, a qual foi apresentada em 4.2.1, exceto pelos valores dos parâmetros *lambda* adicionais.

Nas Figuras 5.20 a 5.22[1], são apresentados os resultados para as métricas PSNR, SSIM e MS-SSIM, incluindo o codificador neural em modo *All Intra*.

Na Figura 5.20, observa-se que o codificador neural em modo *All Intra* é superado pelo modo que considera inter-quadros na maioria das sequências de vídeo analisadas. Nas sequências (a) e (d), o modo *All Intra* do codificador neural apresenta comportamento semelhante ao do modo inter-quadros, mas se distancia e apresenta desempenho inferior considerando a métrica PSNR nas demais sequências.

Para a métrica SSIM, observa-se na Figura 5.21 que o modo *All Intra* do codificador neural superou o codificador em modo inter-quadros na sequência (a). Na sequência (d) o comportamento foi semelhante entre os diferentes modos. Nas demais sequências, o desempenho do codificador neural com inter-quadros foi superior ao seu modo *All Intra*.

Na Figura 5.22, verifica-se que o modo *All Intra* é semelhante ao modo inter-quadros do codificador neural nas sequências (a) e (d). Para as demais sequências, o codificador neural em modo *All Intra* apresenta desempenho inferior.

Portanto, com base nas comparações apresentadas nas Figuras 5.20 a 5.22, o codificador neural considerando inter-quadros possui desempenho superior ao seu modo *All Intra* em grande parte das vezes. Em particular, as sequências *Race Horses* e *Basketball Pass* apresentaram métricas que aproximaram o desempenho de ambos modos do codi-

[1]Para a sequência de vídeo *Basketball Drill Text*, foi adicionado resultado referente a um dos modelos intra-quadros que foi omitido em [de Oliveira et al., 2021].

ficador, indicando que determinadas sequências de vídeo podem reduzir a vantagem do modo inter-quadros em relação ao modo *All Intra* do codificador neural analisado.

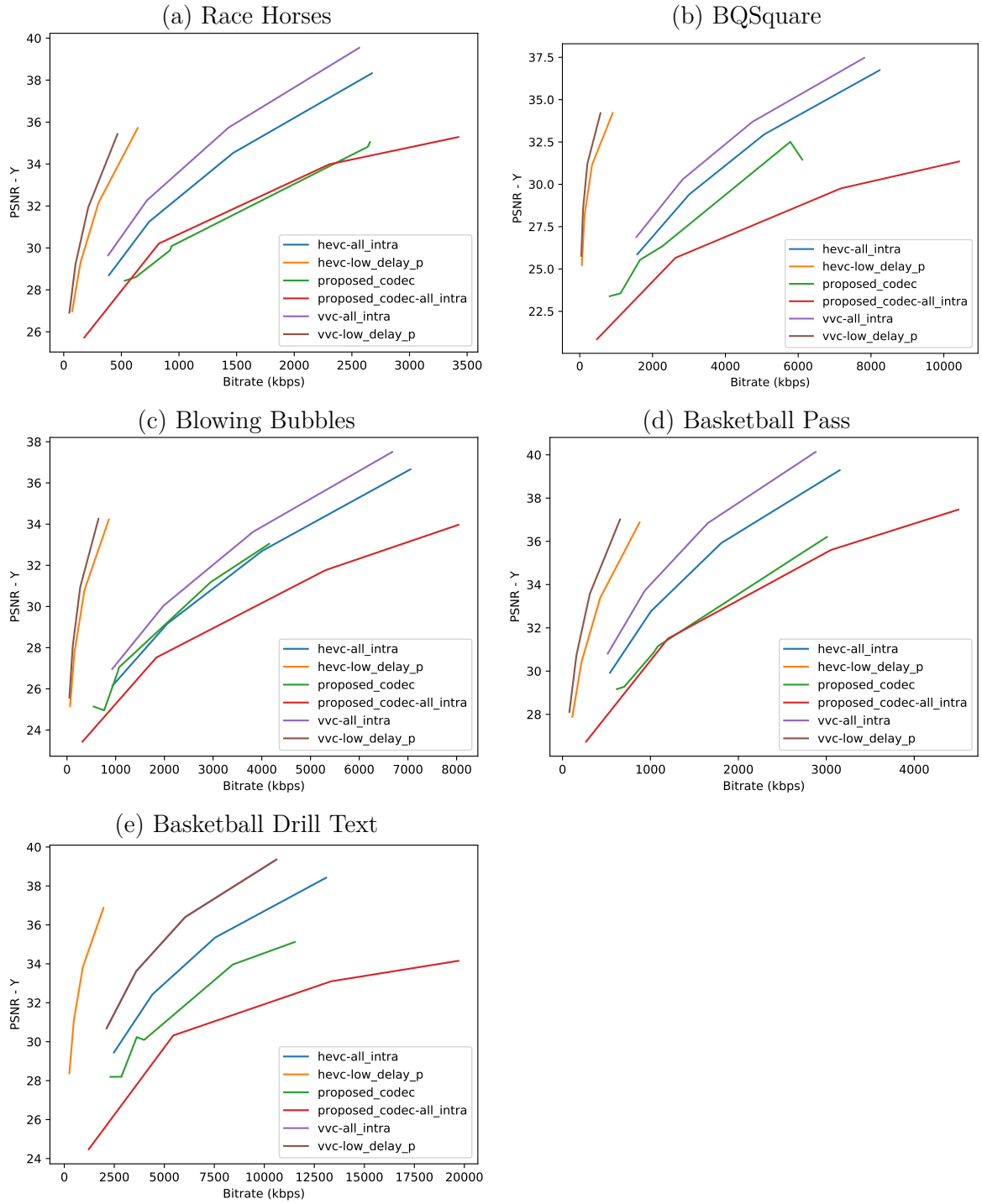

Figura 5.20: Gráficos com a métrica PSNR calculada para o canal de luminância Y. O modo *All Intra* do codificador neural está incluso (Fonte: adaptado de [de Oliveira et al., 2021]).

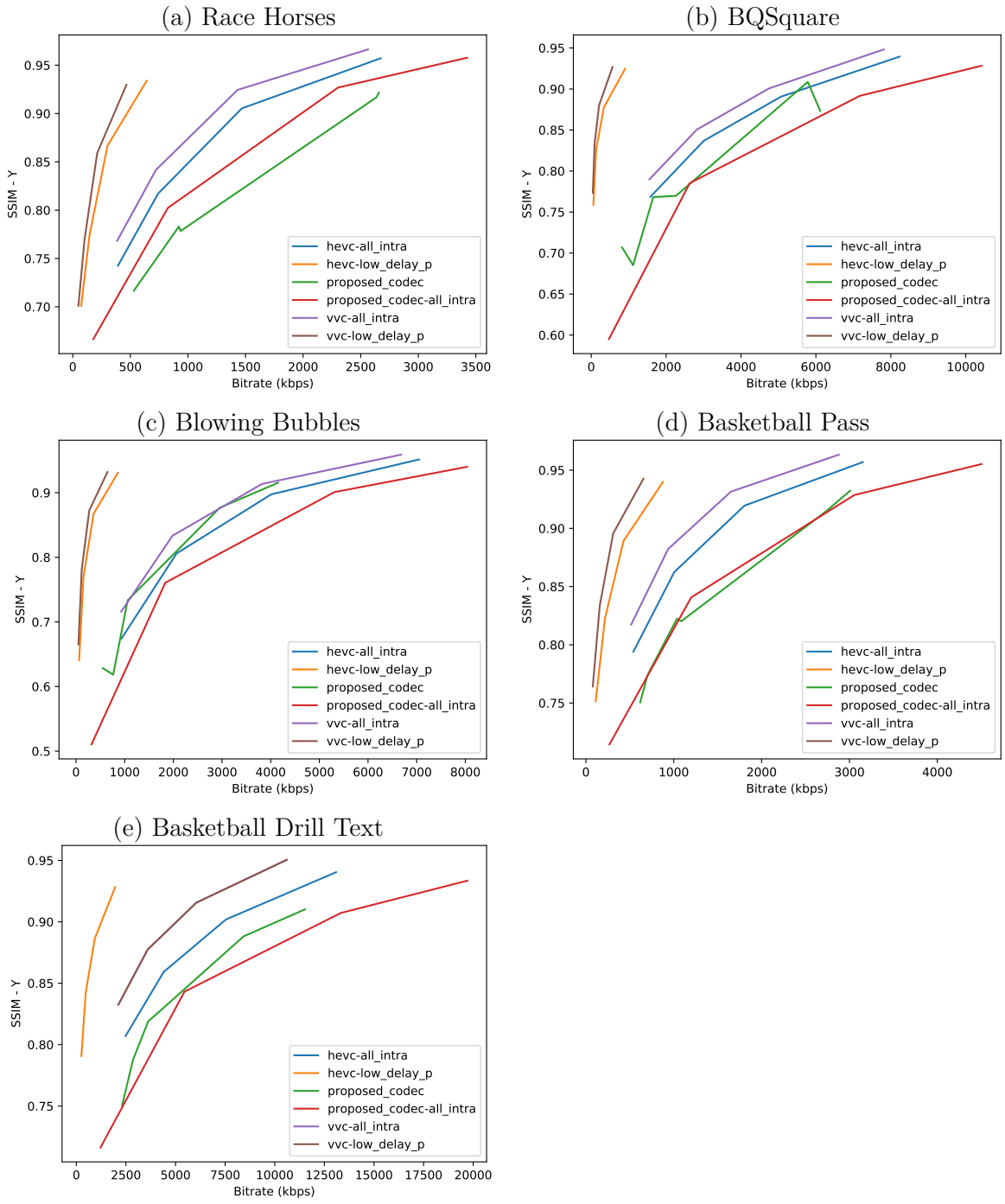

Figura 5.21: Gráficos com a métrica SSIM calculada para o canal de luminância Y. O modo *All Intra* do codificador neural está incluso (Fonte: adaptado de [de Oliveira et al., 2021]).

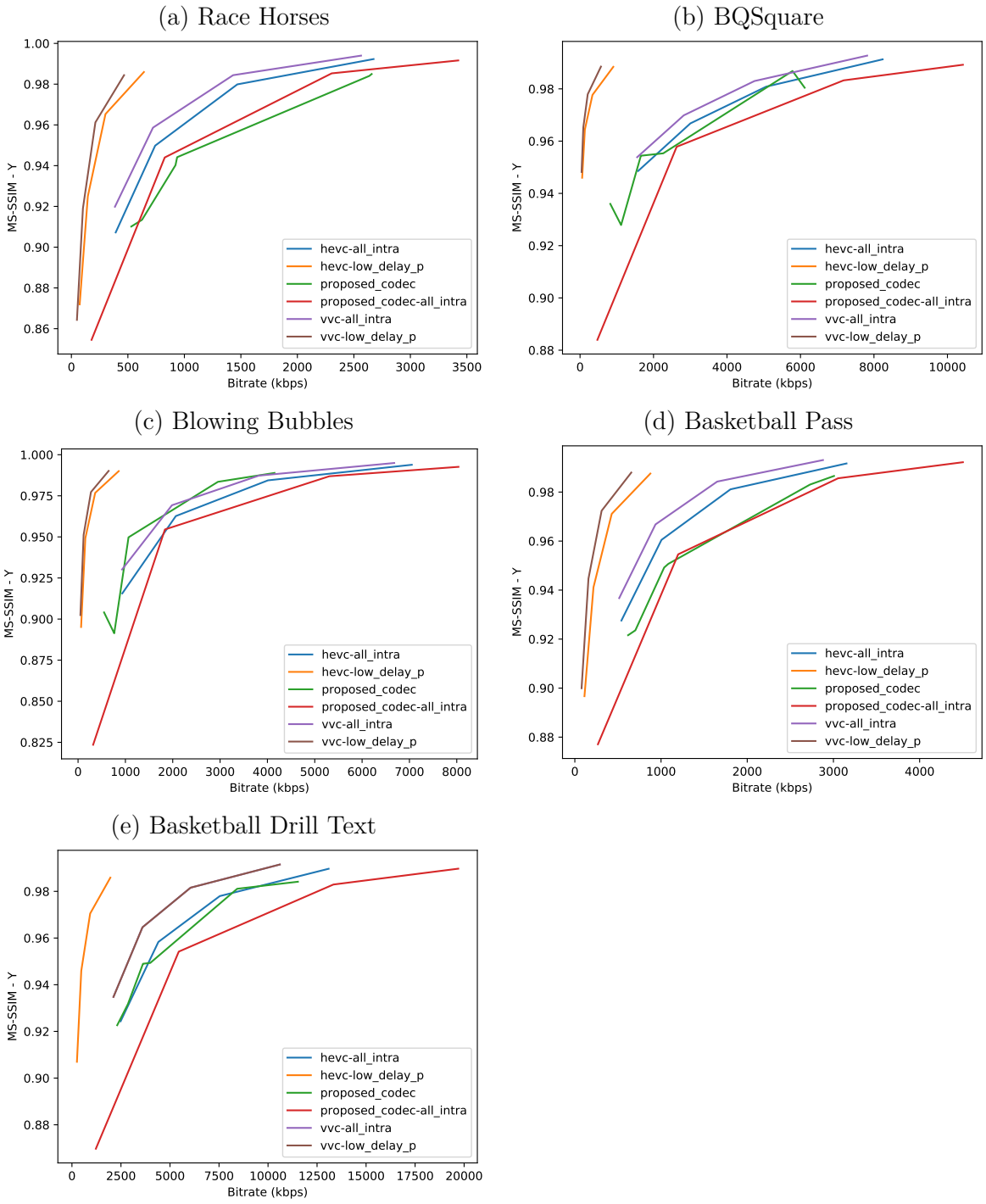

Figura 5.22: Gráficos com a métrica MS-SSIM calculada para o canal de luminância Y. O modo *All Intra* do codificador neural está incluso (Fonte: adaptado de [de Oliveira et al., 2021]).

Capítulo 6

Conclusão

Neste estudo foi realizada análise de desempenho para o codificador de vídeo baseado em *autoencoder* variacional implementado pela equipe do projeto SAMSUNG/FUB/FT - Codificação de imagens e vídeos baseados em inteligência artificial para padrões futuros, referenciado também como *Deep Codec project*. Para a avaliação de desempenho desse codificador, foi desempenhado um estudo inicial acerca do funcionamento geral e identificação dos módulos responsáveis pela realização dos treinamentos. Com isso, uma melhor compreensão acerca do código foi essencial para a realização das demais tarefas necessárias.

Para a obtenção dos modelos finais do codificador, inicialmente foram realizados treinamentos de suas redes componentes intra e inter-quadros. A realização de treinamentos em redes neurais implica a realização de testes para se verificar uma tendência de comportamento da rede, e posterior ajuste dos parâmetros de maneira mais adequada.

Nos treinamentos da rede intra-quadros, foram realizados testes de modelos com número de filtros iguais a 192 e 256 filtros. Os modelos de 256 apresentaram um desempenho levemente superior, sendo mais significativo para altas taxas, com maior valor do parâmetro *lambda*. A decisão neste estudo foi pela utilização somente de 256 filtros, mas uma possibilidade também seria utilizar menor quantidade de filtros para modelos de menores taxas e uma maior quantidade para modelos de maiores taxas. Assim, os treinamentos poderiam se tornar mais ágeis devido a redução de complexidade que o uso de uma menor quantidade de filtros traz para as camadas da rede neural.

Nos demais treinamentos da rede intra, foram analisados os comportamentos para os valores de *lambda* o que possibilitou a determinação de valores mais adequados ao propósito de um codificador de vídeo, evitando-se valores que trouxessem uma taxa excessiva, que consome maior banda ao se pensar na transmissão da informação, ou uma taxa muito baixa que geraria muita distorção. O treinamento da rede inter seguiu o mesmo propósito de se estabelecer valores mais adequados ao parâmetro *lambda*, sendo que os modelos

inter-quadros tem o adicional de utilizarem também o modelo intra em seu treinamento.

Nos treinamentos finais, tanto dos modelos intra quanto inter-quadros, foi empregado maior número de iterações para que os modelos apresentassem melhor convergência. Isso se mostrou importante principalmente na realização do treinamento inter onde, em uma verificação após 300 mil iterações, o modelo de *lambda* inter igual a 10^{-3} utilizando um modelo intra com *lambda* de mesmo valor apresentou uma divergência significativa na MSE. Após a continuação do treinamento desse modelo por mais 200 mil iterações, a rede apresentou uma melhor convergência. Dessa maneira, os treinamentos finais realizados cumpriram com o objetivo de otimização dos modelos definitivos do codificador neural.

Para os modelos finais do codificador de vídeo neural obtidos, observou-se que o ponto de configuração do modelo treinado com *lambda* inter de 10^{-3} e intra de 10^{-2} apresentou comportamento que vai ao encontro da proposta de um codificador de vídeo: economia de taxa mantendo uma distorção aceitável. Com isso, esse modelo foi considerado como o mais adequado levando-se em conta as necessidades de um codificador de vídeo.

Para uma análise mais objetiva do desempenho geral do codificador, foram calculadas métricas sobre os resultados de compressões realizadas. O codificador de vídeo neural apresentou desempenho mais comparável ao modo de configuração *All Intra* dos codificadores clássicos HEVC e VVC. O desempenho se mostrou mais competitivo principalmente ao se considerar a métrica MS-SSIM indicando uma certa qualidade perceptual nos quadros dos vídeos codificados utilizando os modelos treinados.

Portanto, o objetivo de realizar a comparação entre o desempenho do codificador de vídeo neural e os codificadores HEVC e VVC foi atingido. O codificador de vídeo neural analisado neste estudo mostrou que é viável obter resultados com qualidade a partir de uma solução criada com utilização de redes neurais profundas.

Trabalhos futuros incluem a implementação de uma abordagem de predição inter que também envolva a utilização de quadros do tipo B (*B-frame*), a implementação de um treinamento conjunto dos módulos intra e inter-quadros visando melhor otimização e, também, a realização de análises referentes aos tempos de compressão e descompressão do codificador de vídeo neural.

Referências

[Academy, 2021a] Academy, D. S. (2021a). Capítulo 60 – variational autoencoders (VAEs) – definição, redução de dimensionalidade, espaço latente e regularização. Online. `https://www.deeplearningbook.com.br/variational-autoencoders-vaes-definicao-reducao-de-dimensionalidade-espaco-latente-e-regularizacao/`. Acesso em 23/10/2021. 12

[Academy, 2021b] Academy, D. S. (2021b). Capítulo 8 – função de ativação. Online. `https://www.deeplearningbook.com.br/funcao-de-ativacao/`. Acesso em 23/10/2021. 11

[Agustsson and Timofte, 2017] Agustsson, E. and Timofte, R. (2017). Ntire 2017 challenge on single image super-resolution: Dataset and study. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 20

[Ballé et al., 2016a] Ballé, J., Laparra, V., and Simoncelli, E. P. (2016a). End-to-end optimization of nonlinear transform codes for perceptual quality. In *2016 Picture Coding Symposium (PCS)*, pages 1–5. 12

[Ballé et al., 2016b] Ballé, J., Laparra, V., and Simoncelli, E. P. (2016b). End-to-end Optimized Image Compression. *ICLR*. 15

[Bouaafia et al., 2021a] Bouaafia, S., Khemiri, R., Maraoui, A., and Sayadi, F. (2021a). Cnn-lstm learning approach-based complexity reduction for high-efficiency video coding standard. *Scientific Programming*, page 10. 8

[Bouaafia et al., 2021b] Bouaafia, S., Khemiri, R., Messaoud, S., Ben Ahmed, O., and Sayadi, F. E. (2021b). Deep learning-based video quality enhancement for the new versatile video coding. *Neural Computing and Applications*. 8

[Boyce et al., 2018] Boyce, J., Suehring, K., Li, X., and Seregin, V. (2018). JVET-J1010: JVET common test conditions and software reference configurations. Technical report, JVET. 25

[Bross et al., 2021] Bross, B., Wang, Y.-K., Ye, Y., Liu, S., Chen, J., Sullivan, G. J., and Ohm, J.-R. (2021). Overview of the versatile video coding (vvc) standard and its applications. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(10):3736–3764. 1, 2, 8

[Chen et al., 2020] Chen, Z., He, T., Jin, X., and Wu, F. (2020). Learning for video compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(2):566–576. 1

[CLIC, 2020] CLIC (2020). Tasks for workshop and challenge on learned image compression (CLIC) 2020. Online. `http://challenge.compression.cc/tasks/`. Acesso em 01/10/2021. 20

[de Oliveira et al., 2021] de Oliveira, M. C., Martins, L. G., Jung, H. C., Guerin Jr, N. D., da Silva, R. C., Peixoto, E., Macchiavello, B., Hung, E. M., Testoni, V., and Freitas, P. G. (2021). Learning-based end-to-end video compression using predictive coding. xiii, 1, 2, 14, 15, 16, 20, 21, 25, 52, 53, 54, 55

[Ding et al., 2021] Ding, D., Ma, Z., Chen, D., Chen, Q., Liu, Z., and Zhu, F. (2021). Advances in video compression system using deep neural network: A review and case studies. 2, 3

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`. 12

[Jin et al., 2016] Jin, L., Lin, J. Y., Hu, S., Wang, H., Wang, P., Katsavounidis, I., Aaron, A., and Kuo, C.-C. J. (2016). Statistical study on perceived jpeg image quality via mcljci dataset construction and analysis. *Electronic Imaging*, 2016(13):1–9. 21

[JUNG, 2021] JUNG, H. C. (2021). Predictive image compression using autoencoders. Dissertação (mestrado em informática), Universidade de Brasília, Brasília. 6, 7, 11, 12

[Lim et al., 2017] Lim, B., Son, S., Kim, H., Nah, S., and Lee, K. M. (2017). Enhanced deep residual networks for single image super-resolution. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 21

[Nemoto et al., 2014] Nemoto, H., Hanhart, P., Korshunov, P., and Ebrahimi, T. (2014). Ultra-eye: Uhd and hd images eye tracking dataset. In *2014 Sixth International Workshop on Quality of Multimedia Experience (QoMEX)*, pages 39–40. IEEE. 20

[Richardson, 2010] Richardson, I. (2010). The h.264 advanced video compression standard: Second edition. page 316. 9

[Sayood, 2012] Sayood, K. (2012). *Introduction to Data Compression, Fourth Edition*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 4th edition. 4

[Seidel, 2020] Seidel, I. (2020). *Exploiting SATD properties to reduce energy in video coding*. Tese (doutorado), Universidade Federal de Santa Catarina, Florianópolis. 1

[Sullivan et al., 2012] 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. 1, 2, 7, 8

[Tensorflow, 2021] Tensorflow (2021). Tensorboard: kit de ferramentas de visualização do tensorflow. Online. `https://www.tensorflow.org/tensorboard?hl=pt-br`. Acesso em 01/10/2021. 23, 30

[Wang et al., 2004] Wang, Z., Bovik, A. C., Sheikh, H. R., Simoncelli, E. P., et al. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612. 10

[Wang et al., 2003] Wang, Z., Simoncelli, E., and Bovik, A. (2003). Multiscale structural similarity for image quality assessment. volume 2, pages 1398 – 1402 Vol.2. 10

[Weston, 2019] Weston, B. (2019). Bit depth and color subsampling. Online. `https://renewedvision.com/blog/bit-depth-and-color-subsampling/`. Acesso em 12/10/2021. 5

[Wiegand et al., 2003] Wiegand, T., Sullivan, G., 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. 1, 8

[Youtube, 2020] Youtube (2020). Youtube dataset for video compression research. Online. `https://media.withyoutube.com/`. Acesso em 12/09/2021. 21

Apêndice A

Código implementado para conversão de arquivos em formato *YUV* de 10 bits para 8 bits

```
import numpy as np
import glob
from pathlib import Path
import sys

if __name__ == "__main__":
    input_glob, output_folder = sys.argv[1:]
    files = glob.glob(input_glob, recursive=True)

    for f in files:
        output_file = (Path(output_folder) / Path(*Path(f).parts
            [2:])).with_suffix(".yuv")
        parents = output_file.parent

        if not parents.exists():
            parents.mkdir(parents=True)

        file_array = np.fromfile(f, dtype=np.uint16)
        file_array_8bits = (file_array / 4).astype(np.uint8)

        with open(output_file, "wb") as f_out:
            f_out.write(file_array_8bits.ravel().tobytes())
```

Apêndice B

Código implementado para geração de gráficos com métricas para os codificadores em análise

```
import pandas as pd
import sys
import matplotlib.pyplot as plt
import glob
from pathlib import Path


if __name__ == "__main__":
    input_folder, output_folder = sys.argv[1:]
    if not Path(output_folder).exists():
        Path(output_folder).mkdir()

    subfolders_full_path = glob.glob(input_folder + "/*")
    subfolders = list(map(lambda f: Path(f).name,
       subfolders_full_path))
    all_filenames = glob.glob(subfolders_full_path[0] + "/*.csv"
       )
    all_filenames = list(map(lambda f: Path(f).name,
       all_filenames))

    all_metrics = ['PSNR - Y', 'MS-SSIM - Y', 'SSIM - Y']
```

```
for metric in all_metrics:
    aux_metric = metric.replace(" ", "")
    for f in all_filenames:
        for subf in subfolders_full_path:
            df = pd.read_csv(subf + "/" + f)
            df.sort_values("Bitrate (kbps)", inplace=True)
            bitrate = df["Bitrate (kbps)"].values
            metric_value = df[metric].values

            plt.plot(bitrate, metric_value)

        output_file = str(Path(output_folder) / Path(f).stem
            ) + "_" + aux_metric + ".pdf"
        plt.legend(subfolders)
        plt.ylabel(metric)
        plt.xlabel("Bitrate (kbps)")
        plt.savefig(output_file)
        plt.close()
```