UNIVERSIDADE FEDERAL DE PELOTAS
Centro de Desenvolvimento Tecnológico
Programa de Pós-Graduação em Computação

Dissertação

Melhoria de Qualidade de Vídeo Comprimido: Soluções com Redes Neurais Profundas para Múltiplos Codecs

Gilberto Kreisler Franco Neto

Pelotas, 2024

Gilberto Kreisler Franco Neto

Melhoria de Qualidade de Vídeo Comprimido: Soluções com Redes Neurais Profundas para Múltiplos Codecs

Dissertação apresentada ao Programa de Pós-Graduação em Computação do Centro de Desenvolvimento Tecnológico da Universidade Federal de Pelotas, como requisito parcial à obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Guilherme Ribeiro Corrêa
Coorientadores: Prof. Dr. Bruno Zatt
Prof. Dr. Daniel Palomino

Pelotas, 2024

Universidade Federal de Pelotas / Sistema de Bibliotecas
Catalogação da Publicação

F826m Franco Neto, Gilberto Kreisler

Melhoria de qualidade de vídeo comprimido [recurso eletrônico] : soluções com redes neurais profundas para múltiplos Codecs / Gilberto Kreisler Franco Neto ; Guilherme Ribeiro Corrêa, orientador ; Bruno Zatt, Daniel Munari Vilchez Palomino, coorientadores. — Pelotas, 2024.
76 f. : il.

Dissertação (Mestrado) — Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, 2024.

1. Qualidade de vídeo. 2. Rede neural convolucional. 3. Rede neural profunda. 4. Codificação de vídeo. I. Corrêa, Guilherme Ribeiro, orient. II. Zatt, Bruno, coorient. III. Palomino, Daniel Munari Vilchez, coorient. IV. Título.

CDD 005

Elaborada por Simone Godinho Maisonave CRB: 10/1733

Gilberto Kreisler Franco Neto

Melhoria de Qualidade de Vídeo Comprimido: Soluções com Redes Neurais Profundas para Múltiplos Codecs

Dissertação aprovada, como requisito parcial, para obtenção do grau de Mestre em Ciência da Computação, Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas.

Data da Defesa: 27 de junho de 2024

Banca Examinadora:

Prof. Dr. Guilherme Ribeiro Corrêa (orientador)
Doutor em Engenharia Electrotécnica e de Computadores pela Universidade de Coimbra.

Prof. Dr. Daniel Munari Vilchez Palomino (co-orientador)
Doutor em Computação pela Universidade Federal do Rio Grande do Sul.

Prof. Dr. Bruno Zatt (co-orientador)
Doutor em Microeletrônica pela Universidade Federal do Rio Grande do Sul.

Prof. Dr. Luciano Volcan Agostini
Doutor em Computação pela Universidade Federal do Rio Grande do Sul.

Prof. Dr. Mateus Grellert da Silva
Doutor em Computação pela Universidade Federal do Rio Grande do Sul.

Prof. Dr. Carlos Alexandre Silva dos Santos
Doutor em Computação pela Universidade Federal de Pelotas.

RESUMO

FRANCO NETO, Gilberto Kreisler. **Melhoria de Qualidade de Vídeo Comprimido: Soluções com Redes Neurais Profundas para Múltiplos Codecs**. Orientador: Guilherme Ribeiro Corrêa. 2024. 76 f. Dissertação (Mestrado em Ciência da Computação) – Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2024.

O processo de compressão necessário para viabilizar transmissão e armazenamento de vídeos gera artefatos que reduzem a qualidade da imagem, o que se reflete em uma pior qualidade de experiência do usuário. Para atenuar esses artefatos, são tipicamente aplicadas estratégias de filtragem no quadro descomprimido, as quais são classificadas como estratégias *in-loop* e de pós-processamento. Esta última funciona como uma segunda camada de filtragem, já que alguns artefatos persistem após a filtragem *in-loop*. Por não estarem atrelados a nenhum codec em específico, esses filtros podem ser usados como pós-processamento de qualquer codec. As técnicas mais atuais de filtragem de pós-processamento são baseadas em Redes Neurais Profundas, mais especificamente nas Redes Neurais Convolucionais (CNN), como é o caso da *Spatio-Temporal Deformable Fusion* (STDF), que é a técnica estado-da-arte para melhoria de qualidade de vídeos. Entretanto, conforme apresentam os resultados obtidos nesta pesquisa, a arquitetura STDF não apresenta bons resultados quando testada com outros padrões de codificação e níveis de quantização, causando, inclusive, perdas de qualidade em determinados cenários. Por isso, este trabalho propõe a exploração de técnicas de treinamento da arquitetura STDF, com o objetivo de melhorar sua capacidade de generalização em diversos codecs de vídeo. A primeira e a segunda solução são semelhantes quanto à metodologia, ao propor um treinamento baseado em um *dataset* misto formado por vídeos codificados pelo *AOMedia Video 1* (AV1) e *Versatile Video Coding* (VVC). Ao contrário da primeira solução, que é treinada desde o início sem utilizar modelo pré-treinado, a segunda solução emprega a estratégia de *fine tunning*, partindo do modelo STDF original. A terceira solução se baseia no paradigma de treinamento multi-domínio, onde cada domínio corresponde a um codec de vídeo. Os resultados experimentais mostram que a terceira solução atingiu melhorias de qualidade objetiva de até 1,437 dB. Na média, a melhoria de qualidade atingida foi de 0,569 dB e a solução mostrou-se capaz de melhorar a qualidade visual para todos os codificadores e todos os vídeos testados, sendo genérica o suficiente para ser usada como filtro único de pós-processamento para múltiplos padrões/formatos de codificação.

Palavras-chave: Melhoria de Qualidade de Vídeo. Rede Neural Convolucional. Rede Neural Profunda. Codificação de Vídeo.

ABSTRACT

FRANCO NETO, Gilberto Kreisler. **Compressed Video Quality Enhancement: Solutions with Deep Neural Networks for Multiple Codecs**. Advisor: Guilherme Ribeiro Corrêa. 2024. 76 f. Dissertation (Masters in Computer Science) – Technology Development Center, Federal University of Pelotas, Pelotas, 2024.

The compression process necessary to enable video transmission and storage generates artifacts that reduce image quality, thereby resulting in a poorer user experience. To mitigate these artifacts, filtering strategies are typically applied to the decompressed frames, which are classified as in-loop and post-processing strategies. The later functions as a second layer of filtering, as some artifacts persist after in-loop filtering. Since they are not tied to any specific codec, these filters can be used as post-processing for any codec. The most recent post-processing filtering techniques are based on Deep Neural Networks, specifically Convolutional Neural Networks (CNNs), such as Spatio-Temporal Deformable Fusion (STDF), which is the state-of-the-art technique for video quality enhancement. However, as the results obtained in this research show, the STDF architecture does not yield good results when tested with other coding standards and quantization levels, causing quality losses in certain scenarios. Therefore, this work proposes the exploration of training techniques for the STDF architecture with the aim of improving its generalization capability across various video codecs. The first and second solutions are similar in methodology, proposing training based on a mixed dataset consisting of videos encoded by AOMedia Video 1 (AV1) and Versatile Video Coding (VVC). Unlike the first solution, which is trained from scratch without using a pre-trained model, the second solution employs fine-tuning strategy, starting from the original STDF model. The third solution is based on the multi-domain training paradigm, where each domain corresponds to a video codec. Experimental results show that the third solution achieved objective quality improvements of up to 1.234 dB. On average, the quality improvement achieved was 0.544 dB, and the solution proved capable of enhancing visual quality for all tested encoders and videos, being generic enough to be used as a single post-processing filter for multiple encoding standards/formats.

Keywords: Video Quality Enhancement. Convolutional Neural Network. Deep Neural Network. Video Coding.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AV1	AOMidia Video 1
BRNN	Bidirectional Recurrent Neural Network
CNN	Convolutional Neural Network
CQ	Contraint Quality
DNN	Deep Neural Network
HEVC	High Efficiency Video Coding
LSTM	Long Short Term Memory
MFQE	Multi-Frame Quality Enhancement
QoE	Quality of Experience
QP	Quality Parameter
RNN	Recurrent Neural Network
STDF	Spatio-Temporal Deformable Fusion
SVH	Sistema Visual Humano
VQE	Video Quality Enhancement
VVC	Versatile Video Coding

SUMÁRIO

1 INTRODUÇÃO

A evolução da internet abriu caminho para que a maioria do tráfego de dados fosse proveniente de aplicações de vídeo. O advento da pandemia de COVID-19 acentuou ainda mais esse cenário. Só no primeiro mês, o volume de dados proveniente da transmissão de vídeos pela internet aumentou 32,6% (STATISTA, 2022). O serviço de *streaming* Netflix teve que reduzir a qualidade dos seus vídeos para poder suportar o aumento da demanda pelo serviço. Além disso, o consumo de vídeos de alta resolução como 4K (4098 x 2160 píxeis) e 8K (7680 x 4320 píxeis) estão cada vez mais comuns. De acordo com a Cisco (2020), até o fim do ano de 2023 vídeos em 4K representaram 66% do consumo de internet por aparelhos de televisão, sendo este percentual o dobro do previsto em 2018 (33%) (CISCO, 2018). Porém, para que seja possível transmitir e armazenar esses vídeos, são necessárias técnicas que reduzam a quantidade de bits para representar a informação. Isto é necessário pois os recursos como largura de banda e armazenamento são limitados.

O processo de compressão é imprescindível para que se possa armazenar e transmitir um vídeo pela internet, pois esses recursos são limitados. Um vídeo sem compressão ultra-high definition 4K de 4098x2160 píxeis, com três bytes por píxel e com 60 quadros por segundo, com dez minutos de duração seria necessário 955,98 GB para ser armazenado e para transmiti-lo em tempo real seriam necessários 12,75 Gigabits por segundo. A recomendação da Netflix é de 25 Mbps de largura de banda para assistir vídeos de alta resolução. Assim, a largura de banda necessária para transmitir o vídeo não comprimido é 511 vezes maior que o recomendado pela Netflix (PALAU, 2022).

O principal objetivo do processo de compressão é reduzir a quantidade de bits necessária para representar um vídeo, impactando de forma mínima na qualidade visual. Para isso, explora-se o fato de que existem dados redundantes presentes em um vídeo. Essas redundâncias podem ser espaciais, temporais ou entrópicas (RICHARDSON, 2010).

As ferramentas de compressão são caracterizadas como métodos com perda ou sem perda. Nos métodos sem perda, é possível recuperar o dado original após a

descompressão exatamente como ele era antes do processo de compressão. Já no processo de compressão com perda não é possível recuperar o dado original após a descompressão, pois alguma perda de informação ocorre durante o processo. Esta perda de informação é refletida como perda de qualidade, apesar de nem sempre ser perceptível ao sistema visual humano. Este método é o que apresenta um melhor resultado quanto à redução de quantidade de bits para representar um vídeo. Portanto, quando a perda é aceitável, os métodos com perda são os mais indicados.

O processo de perda ocorre devido à etapa de quantização e subamostragem, presente na maioria dos codificadores híbridos como o *High Efficiency Video Coding* (HEVC) (SULLIVAN et al., 2012) e *Versatile Video Coding* (VVC) (BROSS et al., 2021). Nesta etapa, o resíduo transformado para o domínio das frequências passa por um processo de divisão inteira que poda os valores, quase sempre zerando os coeficientes de alta frequência. Consequentemente, este processo acaba por introduzir efeitos visuais indesejáveis, como borramento e efeito de bloco. Estes efeitos são conhecidos como artefatos de compressão, que degradam a qualidade do vídeo reconstruído no decodificador e afetam a qualidade de experiência do usuário final. Além disso, estes artefatos introduzidos podem afetar o desempenho de tarefas de visão computacional, como tarefas de classificação e reconhecimento de objetos/pessoas em um vídeo.

Para atenuar os artefatos de compressão, são aplicadas estratégias de filtragem do quadro descomprimido, que podem ser classificadas como estratégias de filtragem *in-loop* ou de pós-processamento. Os filtros *in-loop* são implementados como parte do processo de codificação/decodificação, já que a filtragem aplicada no momento da codificação tem que ser a mesma aplicada na decodificação. Dessa forma, os filtros *in-loop* atuam melhorando o quadro que vai servir de referência para a codificação do próximo quadro. Porém, apesar do uso dos filtros *in-loop*, ainda é possível observar diversos artefatos presentes no vídeo, pois o processo de filtragem não é capaz de restaurar a imagem ao ponto de equipará-la à imagem original. Além disso, o próprio processo de filtragem *in-loop* também pode acabar por inserir outras artefatos de compressão. Assim, para aplicar mais uma camada de filtragem e tentar aproximar o vídeo ainda mais da sua versão original, podem ser aplicados filtros de pós-processamento. Este tipo de filtro está fora do laço de codificação e é aplicado apenas ao final do processo de decodificação, antes da visualização do vídeo pelo usuário final. Tipicamente, os filtros de pós-processamento não são utilizados no lado codificador.

As técnicas tradicionais de filtragem, tanto *in-loop* quanto de pós-processamento, são baseadas em técnicas clássicas de processamento digital de imagens. As técnicas de deblocagem, por exemplo, requerem a identificação precisa de onde os blocos ocorrem, e a definição de filtros que ao aplicados suavizem as áreas afetadas, preservando ao máximo os detalhes importantes da imagem. No entanto, o desenvolvimento dessas técnicas não é trivial. Algumas correlações podem ser menos óbvias e difíceis

de modelar com heurísticas simples. Por exemplo, a relação entre diferentes regiões de um quadro pode não ser linear ou direta, tornando a filtragem mais complexa. Além disso, artefatos não são uniformes, variando conforme o conteúdo da imagem, intensidade da compressão e outros fatores. Os filtros ou máscaras utilizados no processo de deblocagem, por exemplo, são matrizes de tamanho quadrado, na maioria das vezes de tamanho ímpar, onde cada posição tem um peso. A máscara, então, percorre a imagem pixel a pixel, aplicando uma função representada pelos pesos. Vale ressaltar que estas técnicas geralmente são desenvolvidas para um ou outro tipo de artefato, não servindo com o propósito de restauração do vídeo degradado por diversos tipos de artefatos.

Com o sucesso das redes neurais profundas, mais especificamente as redes neurais convolucionais (*Convolutional Neural Networks* – CNN) em problemas de visão computacional, pesquisadores de melhoria de qualidade de vídeo começaram a voltar sua atenção para este tipo de técnica. Esta também é baseada em máscaras; entretanto, os pesos são encontrados automaticamente por aprendizado de máquina durante a etapa de treinamento. Ao final do treinamento, o modelo pode ser entendido como o estado final do conjunto de pesos/parâmetros após atualizações sucessivas durante o processo de treinamento. Diferente das técnicas tradicionais, as técnicas baseadas em CNN não focam em restaurar o vídeo afetado por um artefato ou outro, mas focam em restaurar a imagem como um todo, na tentativa de aproximar a versão decodificada à versão original.

1.1 Motivação para a Pesquisa

O *Spatio-Temporal Deformable Fusion* (DENG et al., 2020) é o modelo estado-da-arte baseado em CNN para melhoria de qualidade de vídeo (*Video Quality Enhancement* - VQE), porém seu processo de treino utiliza apenas vídeos codificados pelo HEVC e utilizando apenas um nível de quantização. Dessa forma não há garantias que o modelo funcione para diversos codecs.

1.2 Hipótese

Com base na motivação para a pesquisa apresentada, a hipótese que rege este trabalho é a seguinte: É possível desenvolver um modelo genérico de rede neural profunda (*Deep Neural Network* - DNN) que seja capaz de melhorar a qualidade visual, através do pós-processamento, de vídeos comprimidos por mais de um padrão/formato de codificação sob diferentes níveis de quantização.

2 CONCEITOS BÁSICOS DE VÍDEO DIGITAL

Um vídeo digital é uma sequência de imagens estáticas digitais, conhecidas como quadros, que são apresentadas sequencialmente dando a sensação de movimento ao visualizador. A taxa de reprodução das imagens representa um aspecto importante para que o espectador final possa obter uma visualização livre de efeitos de transição, sendo necessário, no mínimo, 24 quadros por segundo (*frames per second – fps*), pois esta é considerada por muitos especialistas e fabricantes a taxa mínima na qual o sistema visual humano percebe a transição como um movimento suave. A taxa de fps representa a resolução temporal do vídeo.

Cada quadro é uma matriz, ou seja, possui uma dimensão horizontal e uma vertical medida em pixeis, representando a resolução espacial do vídeo. Cada pixel contém todas as informações de cor e luminosidade e cada informação separada é conhecida como amostra. Cada amostra, por sua vez, tem seu valor definido de acordo com o nível de precisão da amostra. Dessa forma, uma amostra com uma precisão de 8 bits significa que cada amostra possui 8 bits.

Existem diversas formas de representar as amostras de um quadro. No espaço de cores *red, green, blue* (RGB), um pixel possui uma amostra vermelha, uma verde e uma azul. Neste espaço de cores, as informações de crominância e luminosidade estão misturadas. Outro espaço de cores é o YCbCr, que separa as informações de crominância e luminosidade. Portanto, cada pixel possui uma amostra de luminância (Y), uma de crominância azul (Cb) e uma de crominância vermelha (Cr). Este espaço de cores foi desenvolvido para tirar vantagem da característica do sistema visual humano (SVH) ser muito mais sensível à luminância do que a informações de crominância. Dessa forma, é possível sub-amostrar as informações de cor, reduzindo a quantidade de bits necessária para representar um quadro. As formas de sub-amostragem mais comuns são a 4:2:2, onde para cada quatro amostras de luminância são utilizadas duas de crominância azul e duas de crominância vermelha, e a 4:2:0, onde para cada quatro amostras de luminância são utilizadas uma de cada crominância. Dessa forma, este espaço de cores é o mais usado no processo de compressão, pois ele permite esse processo de sub-amostragem, reduzindo a quantidade de bits necessária para

representar uma imagem.

2.1 Compressão de Vídeo

A compressão de vídeo é um processo que tem como objetivo reduzir a quantidade de bits necessária para representar um vídeo. Este processo pode ser considerado sem perdas ou com perdas. Os algoritmos de compressão sem perda têm o objetivo de reduzir a redundância nos dados e assim representá-lo com uma menor quantidade de bits sem causar nenhuma perda de informação. Neste processo, o vídeo comprimido pode ser completamente recuperado depois do processo de descompressão. Dessa forma, o vídeo original e o reconstruído são os mesmos (CORREA, 2014).

Já os processos de compressão com perda implicam em alguma perda de informação e o dado comprimido não pode ser completamente restaurado durante o processo de descompressão (CORREA, 2014). Estes algoritmos tiram vantagem das características do SVH e removem porções de dados que não são relevantes de forma que a imagem final decodificada parece ter pouca ou nenhuma diferença da original para a maioria dos observadores (CORREA, 2014).

No processo de codificação, inicialmente um quadro é divido em blocos, sendo este a unidade básica do processo de codificação, e então cada bloco é processado individualmente. No processo com perda, cada bloco é codificado através das etapas de predição, transformada, quantização e codificação de entropia, caracterizando um processo de codificação híbrida, que está exemplificado na Figura 1 através de uma representação genérica. A etapa de predição é dividida em predição intraquadro e interquadros. A predição intraquadro tem o objetivo de reduzir a redundância espacial e a interquadros de reduzir a redundância temporal. Nesse processo de predição são utilizadas amostras espacialmente ou temporalmente vizinhas como referência para prever as amostras do bloco sendo processado (VERSATILE VIDEO CODING , VVC).

Na etapa de predição intraquadro são utilizadas apenas informações de dentro do próprio quadro, ou seja, amostras espacialmente vizinhas ao bloco sendo processado. Para prever o valor de uma amostra, o codificador escolhe um modo de predição que vai definir a partir de quais amostras vizinhas já codificadas serão preditas as amostras do bloco atual. Já na etapa de predição interquadros são utilizadas informações de quadros temporalmente vizinhos. Dessa forma, para predizer as amostras do bloco atual são utilizadas informações de quadros passados ou futuros já previamente codificados.

A predição interquadros é composta pelas sub-etapas de estimação de movimento e compensação de movimento. Na etapa de estimação de movimento, é procurado no quadro temporalmente vizinho o bloco que mais se assemelha ao bloco sendo codificado e então é gerado um vetor de movimento que indica a posição do bloco escolhido

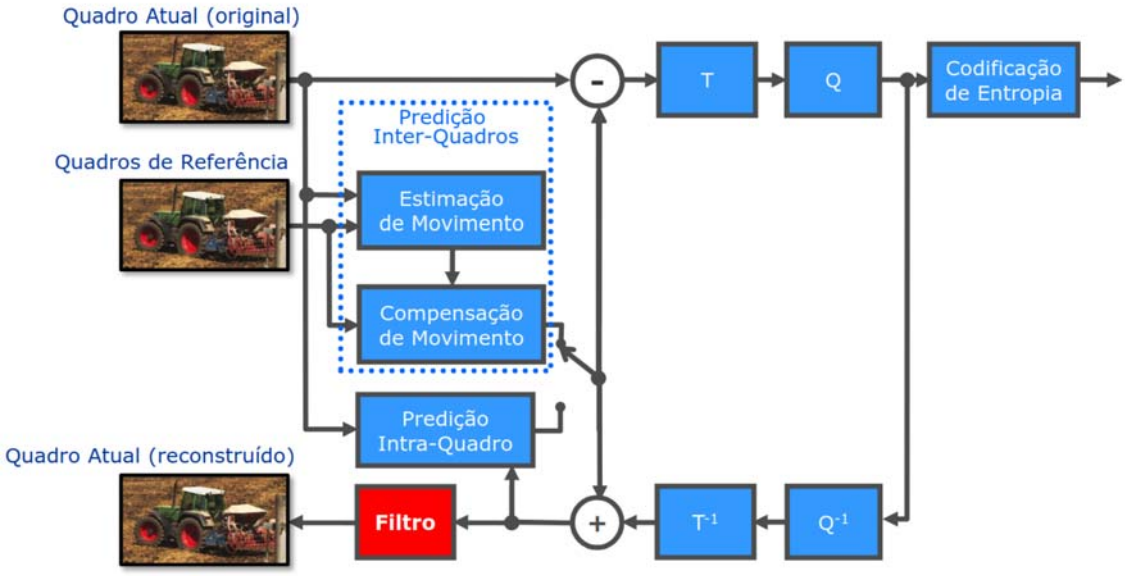

Figura 1 – Modelo genérico de um codificador híbrido (AGOSTINI, 2007)

no quadro de referência (PALAU, 2022). A etapa de compensação de movimento utiliza o vetor de movimento gerado na etapa anterior para mover o bloco de referência escolhido para a posição do bloco sendo codificado.

O bloco predito, então, passa por uma subtração do bloco original, gerando um bloco que é chamado de resíduo. O resíduo passa pelo módulo de transformada, alterando a representação espacial para o domínio das frequências. Dessa forma, a etapa de quantização pode ser aplicada sobre os valores transformados e explorar as características do SVH, descartando as frequências que são menos relevantes à percepção humana (SALDANHA, 2021). Nesta etapa, os valores do resíduo transformado passam por uma divisão inteira por um valor base definido através de um parâmetro de quantização (*Quantization Parameter* – QP), em outros codificadores este parâmetro é chamado de limitação de qualidade (*Contraint Quality* - CQ). Nesta dissertação, na maioria das vezes, será utilizado o termo QP para se referir ao valor base da divisão inteira no processo de quantização. O valor de QP/CQ é diretamente proporcional à taxa de compressão e inversamente proporcional à qualidade da imagem resultante. Assim se caracteriza o processo com perda, pois, como não é armazenado o resto da divisão, não é possível recuperar esta informação na decodificação. Ao final da quantização, o bloco está pronto para a codificação de entropia, que explora a probabilidade de ocorrência dos símbolos codificando aqueles com maior ocorrência com códigos menores e os com menor ocorrência com códigos maiores.

O codificador precisa ter as mesmas referências que o decodificador, então o codificador possui também uma etapa de decodificação. Como a quantização é um processo com perdas e as etapas de predição utilizam blocos já codificados como referência, não se pode utilizar um bloco de um quadro original, já que o decodificador

não terá acesso a ele. Assim, o resíduo, que foi transformado e quantizado, passa por uma etapa de quantização e transformada inversa, e ao ser somado com o bloco predito pode ficar disponível como referência para o processamento do próximo bloco.

2.2 Artefatos de Compressão

Os artefatos de compressão, também conhecidos como efeitos de compressão, são distorções espaciais e temporais introduzidas em quadros de um vídeos após o processo de descompressão. Esses artefatos degradam a qualidade objetiva e subjetiva de um vídeo, ocasionando uma baixa qualidade na experiência do usuário final. As distorções espaciais típicas incluem efeitos de bloco, borda e borramento; já as distorções temporais incluem os artefatos ruído de mosquito e *flickering* (NADERNEJAD et al., 2013).

2.2.1 Artefato de Borramento

O processamento de um bloco pela etapa de transformada seguida pela etapa de quantização frequentemente remove pequenas amplitudes dos coeficientes transformados. Como a energia de sinais visuais naturais se concentra em baixas frequências, a quantização reduz energias de alta frequência, resultando em um efeito de borrão no sinal reconstruído, causando efeitos similares a um filtro passa-baixa. O borramento se manifesta como uma perda de detalhes espaciais, perda de nitidez em bordas ou perda de textura em regiões da imagem. Devido à natureza de processamento independente de cada bloco nos codificadores de vídeo híbridos atuais, usualmente o borramento se manifesta dentro de um bloco (ZENG et al., 2014).

Outro processo que pode causar o efeito de borramento são os filtros *in-loop* de deblocagem, que são aplicados para reduzir o efeito de bloco, explicado na seção a seguir. Estes filtros são essencialmente filtros passa-baixa espacialmente adaptativos que suavizam as regiões de bordas dos blocos, produzindo o efeito de borramento nas suas zonas de limite (ZENG et al., 2014). A Figura 2 mostra um exemplo de efeito de borramento.

Figura 2 – Exemplo de artefatos de borramento. (a) Quadro referência. (b) Quadro após descompressão com artefato de borramento. Fonte: (LIN et al., 2020).

2.2.2 Artefato de Bloco

Os artefatos ou efeitos de bloco ocorrem na direção vertical e horizontal de um quadro e são causados devido à natureza do processo de codificação, onde um quadro é dividido em blocos e cada bloco é processado independentemente (SHEN; KUO, 1998) (NADERNEJAD et al., 2013). Dessa forma, este efeito aparece como uma falsa descontinuidade através das bordas dos blocos. A Figura 3 mostra exemplos de efeitos de bloco. Apesar de os efeitos de bloco serem gerados por razões similares, sua aparência visual pode ser diferente dependendo da região onde o bloco ocorre, podendo ser subdividido em três categorias: efeito mosaico, efeito escadaria e falsa borda (ZENG et al., 2014).

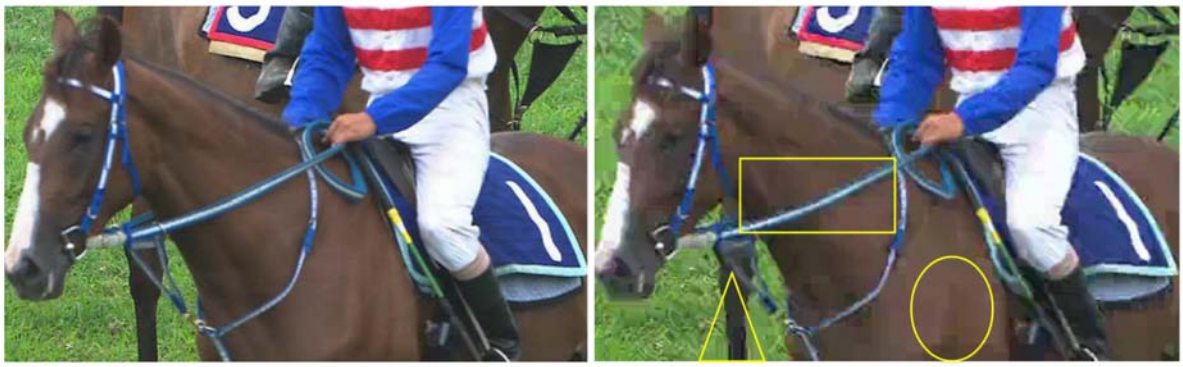

Figura 3 – Exemplo de artefatos de bloco. (a) Quadro referência (b) Quadro comprimido apresentando artefatos de bloco: efeito mosaico (elipse), efeito escadaria (retângulo) e falsa borda (triângulo). Fonte: (ZENG et al., 2014).

O efeito mosaico ocorre quando existem transições de luminância em regiões grandes de baixa energia (por exemplo, paredes e superfície de mesas). Devido à quantização dentro de cada bloco, quase todos os coeficientes AC são quantizados para zero e assim cada bloco é reconstruído como um bloco DC constante, onde os valores DC variam de bloco para bloco. Quando todos os blocos são unificados o efeito mosaico se manifesta como mudanças abruptas de luminosidade de um bloco para outro através da resolução espacial (ZENG et al., 2014).

O efeito escadaria tipicamente ocorre ao longo de uma linha diagonal ou curva, quando misturado com bordas falsas horizontais e verticais na região das bordas do bloco, criando uma falsa estrutura de escadaria. Já a falsa borda é um efeito que aparece próximo de bordas verdadeiras, sendo frequentemente criado pela combinação do processo de estimação/compensação de movimento da predição interquadros e efeitos de bloco em quadros prévios usados como referência, onde o efeito de bloco em quadros prévios é transposto para o quadro atual via compensação de movimento como uma borda artificial (ZENG et al., 2014).

2.2.3 Artefatos de *Ringing* e *Flickering*

Os efeitos de *ringing* ocorrem quando os coeficientes transformados de alta frequência são quantizados ou truncados, perdendo os valores originais, e assim causam oscilações na volta das bordas do bloco (SHEN; KUO, 1998) (NADERNEJAD et al., 2013). A Figura 4 mostra um exemplo de artefato de *ringing*.

(a) (b)

Figura 4 – Exemplo de artefatos de *ringing*. (a) Quadro referência. (b) Quadro após descompressão com artefato de *ringing*. Fonte: (UMNOV et al., 2014).

Já o artefato mosquito é o efeito de *ringing* ao longo do eixo temporal. Portanto, ele ocorre quando o efeito de *ringing* alterna de quadro para quadro enquanto a sequência de vídeo é mostrada. O efeito de *flickering* ocorre devido a inconsistências de qualidade de quadro para quadro na mesma região espacial, aparentando que a região afetada está piscando (NADERNEJAD et al., 2013).

2.3 Filtragem de Vídeos

A filtragem de vídeo é um processo que tem por objetivo reduzir ou suavizar os artefatos provenientes do processo de compressão. O processo de filtragem é aplicado a cada quadro do vídeo. A filtragem pode ser executada dentro do laço de codificação, sendo chamados de filtros *in-loop*, ou após todo o processamento, sendo chamados de filtros de pós-processamento. Os filtros *in-loop* processam o quadro dentro do laço de decodificação e o quadro resultante pode ser utilizado como referência para a codificação dos próximos quadros (ZHANG et al., 2015). Neste caso, as informações da aplicação do filtro precisam ser codificadas junto com as informações do vídeo, pois no decodificador o processo de filtragem tem que ser o mesmo aplicado durante a codificação para que os quadros usados como referência sejam idênticos.

Um exemplo de filtro *in-loop* presente na maioria dos codecs é o filtro de deblocagem, ou *deblocking filter*. O filtro de deblocagem é um filtro *in-loop* presente em codificadores como o VVC, HEVC e o AV1. Seu propósito é amenizar visualmente os artefatos de blocagem que são causados devido ao processamento em blocos de um quadro. Ele atua adaptativamente nas bordas entre os blocos, suavizando a transição

aparente de um bloco para o outro, ao mesmo tempo que busca manter as bordas naturais das imagens.

Já para remover artefatos de *ringing*, filtros como o *Sample Adaptive Offset* (SAO), presente no VVC (KARCZEWICZ et al., 2021), e o Filtro de Aprimoramento Direcional Restrito, presente no AV1 (PALAU, 2022), são utilizados. Diferente do filtro de deblocagem, o filtro de *deringing* atua nas bordas naturais da imagem. Nos últimos anos, técnicas baseadas em aprendizado de máquina vêm surgindo para a filtragem *in-loop*, como, por exemplo, a solução proposta em Jia et al. (2017) que busca amenizar diversos tipos de artefatos através de uma rede totalmente convolucional treinada para aprender um mapa residual que melhora o bloco sendo processado.

Os filtros de pós-processamento estão fora do codificador, sendo aplicados após toda a etapa de decodificação com o objetivo de melhorar ainda mais a qualidade do vídeo. Eles podem ser utilizados ao final do processo de descompressão, imediatamente antes da exibição do vídeo para o usuário final. Como este tipo de filtragem não faz parte do decodificador, não implica na necessidade de o codificador incluir informações extras no *bitstream* para permitir a decodificação do vídeo.

As técnicas de pós-processamento baseadas em heurísticas podem ser separadas em quatro categorias (TANG; LI, 2016): métodos baseados em modelos estatísticos, técnicas de filtragem espacial, métodos baseados em transformadas e métodos baseados em *Projection Onto Convex Sets* (POCS). Um exemplo de filtro espacial é o proposto por Iii; Lim (1984), que aplica um filtro Gaussiano nos píxeis em volta do limite dos blocos para suavizar o efeito de bloco. Já os autores em Zhang et al. (2015) propõem um filtro baseado em transformadas, também se utilizando de modelos estatísticos para estimar os coeficientes *Discrete Cosine Transform* (DCT) originais do bloco transformado e assim suavizar diversos artefatos presentes no quadro.

Técnicas mais recentes têm se baseado em algoritmos de aprendizado de máquina, mais especificamente redes neurais convolucionais, ficando a cargo do modelo aprender as correlações necessárias que aproximam o quadro descomprimido do quadro original. Estes métodos serão abordados em mais detalhes no Capítulo 4 desta dissertação.

2.4 Métricas de Avaliação

Para verificar se um método específico de aprimoramento de qualidade de vídeo realmente funciona, é essencial empregar métodos que avaliem a presença e a magnitude dessas melhorias. Segundo (CHIKKERUR et al., 2011), uma avaliação confiável tem um importante papel em encontrar a qualidade de serviço (*Quality of Service – QoS*) prometida e assim melhorar a qualidade de experiência (*Quality of Experience – QoE*) do usuário final. Para isso, existem dois tipos principais de métricas de avalia-

ção: métricas subjetivas e métricas objetivas.

A avaliação subjetiva se baseia na percepção de observadores humanos. Segundo (MOHAMMADI; EBRAHIMI-MOGHADAM; SHIRANI, 2014), este tipo de avaliação é a mais confiável para verificar a qualidade de um vídeo, já que humanos são os usuários finais da maioria das aplicações. A métrica Pontuação de Opinião Média (*Mean Opinion Score – MOS*), obtida a partir de uma certa quantidade de observadores humanos, foi tida por muitos anos como a forma mais confiável de medida de qualidade objetiva (WANG et al., 2003). Entretanto este tipo de métrica requer muito tempo para ser obtida. Ainda, experimentos subjetivos são complicados, pois dependem de fatores como a distância de visualização, o dispositivo de exibição, condições de iluminação, dentre outros (CHIKKERUR et al., 2011). Dessa forma, alguns modelos matemáticos capazes de predizer a avaliação de qualidade de um observador humano médio vêm sendo desenvolvidos mais recentemente, como por exemplo a métrica *Structural Similarity Index Measure* (SSIM) (ZHOU et al., 2004) explicada mais adiante nesta seção.

Por outro lado, as métricas objetivas se baseiam em formulações matemáticas para avaliar a qualidade de um vídeo. Elas podem ser classificadas em três grupos quanto à disponibilidade da versão original que pode ser usada como referência para comparar um vídeo degradado: *Full Reference* (FR), *Reduced Reference* (RR) e *No Reference* (NR). Nas métricas FR, os quadros originais, não degradados e em perfeita qualidade, estão disponíveis. Nas métricas RR, um vídeo de referência não está totalmente acessível; ao invés disso, parâmetros são extraídos do vídeo de referência e do vídeo processado. A partir desses parâmetros, a métrica objetiva pode ser calculada. Nas NR, nenhuma referência está disponível.

As métricas FR mais utilizadas para avaliação de qualidade objetiva em vídeos são a *Mean Squared Error* (MSE) e *Peak Signal-to-Noise Ratio* (PSNR) (WANG et al., 2003). O $PSNR$ é medido em uma escala logarítmica, utilizando como base o critério de similaridade MSE entre o quadro original e o quadro reconstruído/filtrado.

Na eq. (1), para o PSNR, o valor de MAX se refere ao valor máximo que uma amostra pode alcançar. Portanto, no caso de uma imagem de apenas um canal de cor com 8 bits de profundidade, o valor de MAX será de 255. O MSE está definido na eq. (7), onde m e n são as dimensões da imagem e O e R são os quadros original e reconstruído/filtrado, respectivamente. Quanto maior o valor do PSNR, melhor é a qualidade da imagem sendo comparada. No caso dessas métricas serem aplicadas para vídeos, o cálculo é feito quadro a quadro e no fim um valor médio proporcional à quantidade de quadros do vídeo é computado.

$$PSNR = 20\log(\frac{MAX}{\sqrt{MSE}}) \tag{1}$$

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} (R_{i,j} - O_{i,j})^2 \tag{2}$$

Apesar de muito utilizadas, as métricas anteriormente citadas não são muito apropriadas para perceber a qualidade visual da imagem, pois não consideram informações perceptuais como luminância e contraste. Assim, foi criada a métrica SSIM (ZHOU et al., 2004), que considera as mudanças de percepção na informação estrutural. Esta métrica se baseia no fato que pixeis de uma imagem natural demonstram forte dependência e essas dependências carregam informações úteis sobre a estrutura de uma cena.

Na eq. (3), para o SSIM, μ_x e μ_y denotam as intensidades médias de luminância dos sinais das imagens comparadas **x** e **y**, σ_x e σ_y denotam o desvio padrão das amostras de luminância, definindo uma comparação de contrastes, σ_{xy} denota a covariância das amostras de luminância, definindo uma comparação estrutural. Os termos C_1 e C_2 são constantes estabilizadoras.

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{3}$$

3 APRENDIZADO DE MÁQUINA

O aprendizado de máquina (AM) é uma área da inteligência artificial (IA) que tem o objetivo de dar a computadores a habilidade de resolver problemas sem a necessidade de programar instruções diretas ou heurísticas. Dessa forma esta área se preocupa em desenvolver algoritmos que sejam capazes de melhorar seu desempenho em alguma tarefa por meio da execução repetitiva da mesma e assim aprender com a experiência. Apesar da IA se preocupar com o raciocínio indutivo e dedutivo, o AM se preocupa apenas com o raciocínio indutivo, já que no indutivo parte-se de premissas particulares e se alcança um conclusão geral, enquanto no dedutivo parte-se de uma premissa geral e se alcança uma conclusão particular (DUARTE, 2021). Portanto, o AM tem o intuito de desenvolver algoritmos que a partir de um treinamento sobre um conjunto de dados tenha a habilidade de generalizar o conhecimento para um outro conjunto não visto durante seu treinamento.

3.1 Tipos de Aprendizado de Máquina

Os algoritmos de aprendizado de máquina podem ser classificados em aprendizado supervisionado, não supervisionado e por reforço. O uso de cada tipo vai depender das características dos dados.

No aprendizado supervisionado são apresentados ao computador dados rotulados, ou seja, dados que o resultado de saída desejado já é conhecido. Dessa forma, o algoritmo usa o erro gerado pela comparação com o resultado já conhecido para se auto-ajustar e com isso aprender a fazer inferência de rótulos em novos dados. Como exemplo de aprendizado supervisionado, pode-se destacar o problema de classificação, onde se tem um conjunto de entradas e sua classificação é conhecida e se deseja classificar um outro conjunto que tem a classificação desconhecida. No caso do processo de restauração de imagens com redes neurais convolucionais, a entrada é uma imagem com qualidade degradada e ao final obtém-se uma imagem melhorada. Neste caso, não há intenção de classificar o dado, pois a entrada e a saída da rede são imagens. A imagem original funciona como valor base para a etapa de treinamento, de

forma que se deseja aproximar a imagem de saída da imagem original.

No aprendizado não supervisionado, os dados apresentados ao computador não possuem um rótulo, porém o objetivo continua o mesmo: encontrar padrões. Nesse método, o algoritmo busca por padrões entre os dados para agrupá-los em classes. Este procedimento é usado em casos que se deseja encontrar estruturas ou padrões que não estão nítidos para um ser humano.

No aprendizado por reforço, o computador tem o objetivo de melhorar seu desempenho com base em interações com o ambiente. Portanto, tenta encontrar ações que maximizam sua recompensa. Neste caso, não são dados como exemplo para o treinamento as saídas ótimas, em contraste com o aprendizado supervisionado onde o resultado desejado é dado como entrada para o treinamento. Mas a rede tenta descobri-las através de um processo de tentativa e erro.

3.2 Redes Neurais

As redes neurais (RN) são uma classe de algoritmos de AM que tem sua estrutura de funcionamento inspirada nas redes neurais biológicas do cérebro. Dessa forma, uma RN corresponde a um conjunto de neurônios interconectados entre si formando uma rede. Cada neurônio é conectado a outro através de pesos, também chamados de força sináptica (GOODFELLOW; BENGIO; COURVILLE, 2016). Assim, os neurônios trabalham entrelaçados de forma distribuída para aprender coletivamente a partir das entradas e otimizar a saída final.

Em geral, as redes neurais podem ser classificadas de acordo com sua arquitetura em três grupos (HAYKIN, 2009): redes *feedforward* de camada única, redes *feedforward* com múltiplas camadas e redes recorrentes. Essas redes são ditas de camadas pois é dessa forma que os neurônios estão estruturados na rede, em camadas. Dessa forma, neurônios de uma mesma camada não possuem conexão entre si. Na rede do tipo *feedforward* o fluxo dos dados na rede fluem da camada de entrada para a camada de saída e não ao contrário. Uma RN *feedforward* de camada única, é composta por uma camada de entrada, através das quais os dados serão introduzidos no sistema, e uma camada de saída. A camada de entrada está diretamente ligada à camada de saída, sem possuir nenhuma outra camada no meio.

As redes com múltiplas camadas são mais complexas, podendo ter um número muito maior de camadas entre a camada de entrada e a camada de saída, conhecidas como camadas ocultas. Essa camada é chamada de oculta devido a não ser diretamente vista a partir da entrada ou da saída da rede. A inclusão de camadas ocultas permite à rede extrair das entradas estatísticas de ordens mais altas (GOODFELLOW; BENGIO; COURVILLE, 2016). A rede recorrente se distingue da rede *feedforward* por ter pelo menos um laço de retroalimentação (HAYKIN, 2009). A Figura 5 exemplifica a

diferença entre as três arquiteturas.

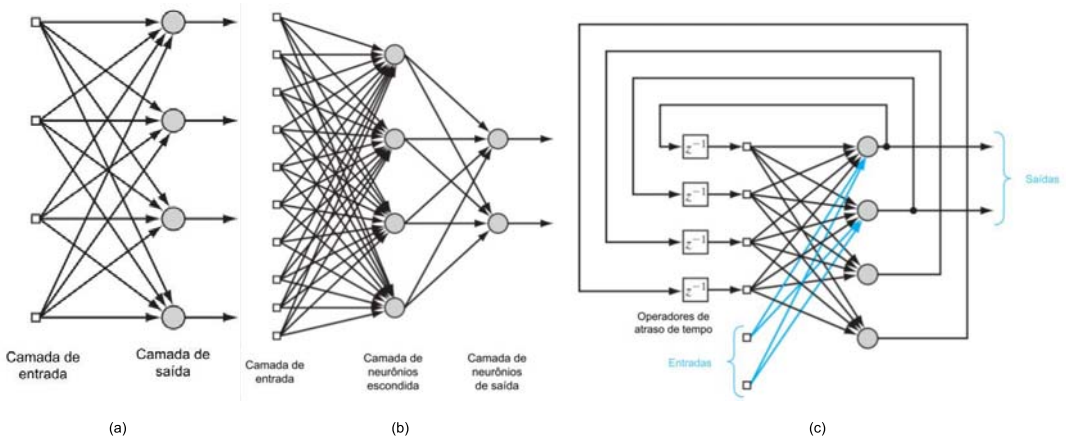

Figura 5 – Arquiteturas básicas das redes neurais: (a) Rede *feedforward* de camada única, (b) Rede *feedforward* com múltiplas camadas, (c) Rede recorrente. Fonte: (HAYKIN, 2009).

3.3 Funções de Ativação

As funções de ativação são um ponto crítico no projeto de uma rede neural, pois é a partir delas que é dada a característica de não linearidade ao modelo, permitindo que a rede aprenda e represente relações complexas nos dados. Introduzidas em cada neurônio da rede neural, as funções de ativação determinam se um neurônio deve ser ativado, ou seja, se vai passar informação adiante, ou não, com base no valor de saída da função. Sem função de ativação, uma rede neural seria equivalente a uma combinação linear de transformações lineares, o que limita sua capacidade de aprender padrões complexos nos dados.

Uma característica importante das funções de ativação é que devem ser deriváveis para o cálculo do erro em relação aos pesos atuais em cada camada durante o algoritmo de retro-propagação (*backpropagation*), possibilitando que a rede neural aprenda. Também é importante que a derivada da função de ativação não desloque o gradiente para zero, acarretando no problema de desaparecimento de gradiente, o que impede que os pesos atualizem seu valor. Da mesma forma, a derivada da função não pode gerar valores muito grandes, acarretando no problema de explosão de gradiente. Outra característica importante é que a função de ativação deve ser centrada no valor zero, de forma a evitar que os gradientes mudem para uma direção específica (FERRUGEM, 2022).

Algumas das funções de ativação mais utilizadas são: sigmóide, tangente hiperbólica e a função de ativação retificada linear (*Rectified Linear Unit* – ReLU), apresentadas brevemente a seguir.

A função sigmóide é uma das formas mais comuns de funções de ativação. Ela traduz uma faixa de entrada de $(-\infty; +\infty)$ para uma faixa de saída em $[0; 1]$ e é repre-

sentada pela seguinte equação:

$$g(x) = \frac{1}{1 + e^{-x}} \tag{4}$$

A Figura 6 mostra o gráfico da função sigmóide e sua derivada. Como é possível observar, $g(x)$ tem a propriedade desejada de ser derivável em todos os pontos, o que ajuda no algoritmo de retro-propagação, já que é possível encontrar o gradiente em qualquer ponto. Porém, ela não é centrada em zero e em sua derivada, $g'(x)$, apenas em uma pequena faixa os valores são significativos. Conforme se encontram valores maiores ou menores de $g(x)$ a função derivada se aproxima de zero, o que pode acarretar no problema de desaparecimento de gradiente, pois o algoritmo de retro-propagação utiliza a regra da cadeia para o processo de derivação e, conforme se avança para as camadas de entrada, muitos termos próximos de zero vão sendo multiplicados. Por este motivo, essa função de ativação não é utilizada em redes mais profundas do que cinco camadas (DING; QIAN; ZHOU, 2004) ou é utilizada apenas na camada de saída devido à distribuição de seus valores.

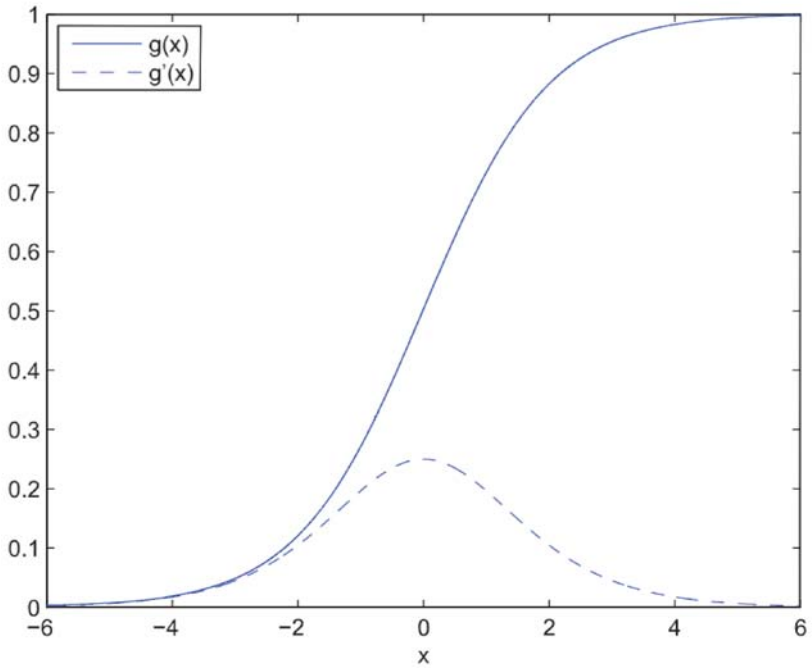

Figura 6 – Função de ativação sigmóide e sua derivada (DING; QIAN; ZHOU, 2004)

A tangente hiperbólica tem seu formato parecido com a função sigmóide, porém sua saída é na faixa de $[-1; 1]$. A equação a seguir descreve a função tangente hiperbólica:

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{5}$$

A Figura 7 mostra o gráfico da função tangente hiperbólica e sua derivada. A função tangente hiperbólica possui as mesmas vantagens que a sigmóide, porém ela

é simétrica em relação à origem, o que permite mapear as entradas para valores negativos. Redes neurais com tangente hiperbólica convergem mais rápido que as redes com sigmóide (DING; QIAN; ZHOU, 2004). Entretanto, a função apresenta o mesmo problema de desaparecimento de gradiente.

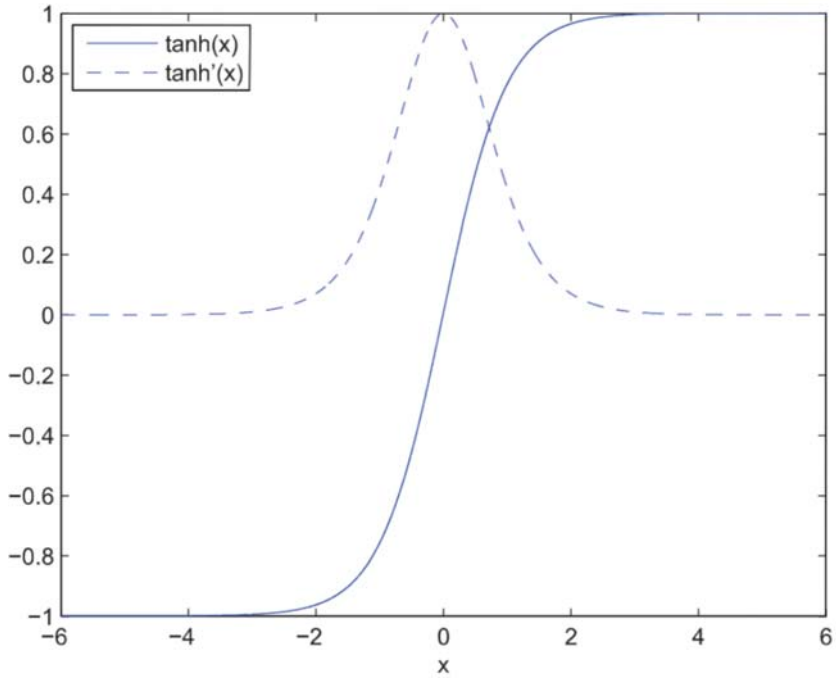

Figura 7 – Função de ativação tangente hiperbólica e sua derivada (DING; QIAN; ZHOU, 2004)

A função ReLU é largamente utilizada em camadas escondidas de redes neurais profundas, por ser computacionalmente menos complexa que as funções sigmóide e tangente hiperbólica (FERRUGEM, 2022). A equação a seguir descreve a função ReLU:

$$g(x) = argmax(0, x) \tag{6}$$

A Figura 8 apresenta o gráfico da função ReLU e sua derivada. Como é possível observar, a ReLU é linear para todos os valores positivos de entrada e zero para todos os valores negativos de entrada. Sua derivada é constante para todos os valores maiores que zero, o que pode evitar que o algoritmo fique preso em um ótimo local, aliviando o problema do desaparecimento de gradiente. Porém, quando $x < 0$ a derivada é $g'(x) = 0$, o que pode ocasionar no problema da ReLU Moribunda (FERRUGEM, 2022), eventualmente causando a morte de alguns neurônios, ou seja, nunca os ativando.

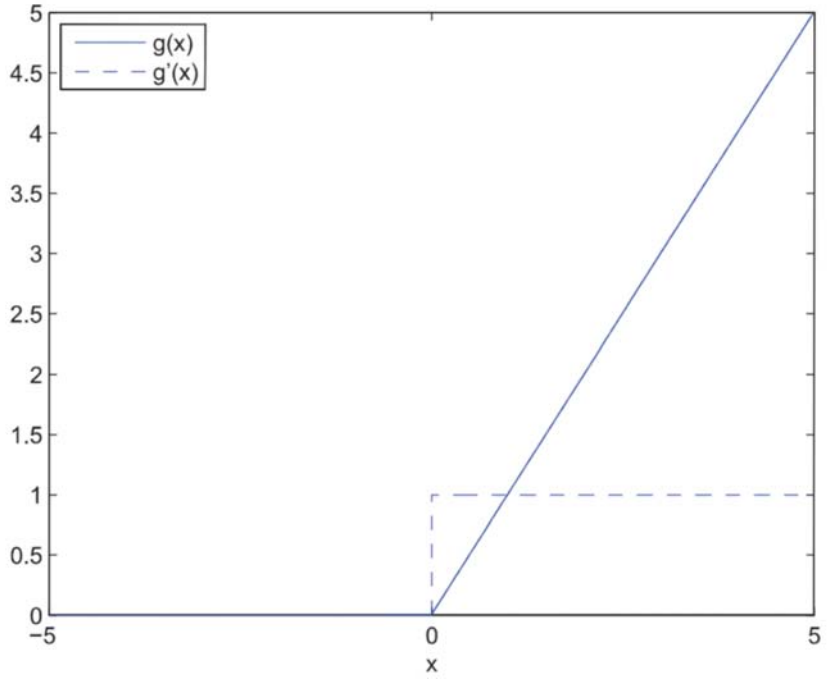

Figura 8 – Função de ativação ReLU e sua derivada (DING; QIAN; ZHOU, 2004)

3.4 Funções de Perda

As funções de perda são componentes fundamentais no treinamento de um modelo de aprendizagem de máquina. Elas medem a diferença entre as previsões do modelo e os valores reais, orientando o processo de otimização para minimizar essa diferença. A escolha da função de perda adequada depende da natureza do problema (classificação, regressão, etc) e das características dos dados (WANG et al., 2022).

Usada principalmente em problemas de regressão a *Mean Squered Error* (MSE) é uma função de perda que calcula a média dos quadrados das diferenças entre os valores previstos e os valores reais. Esta função penaliza erros grandes mais severamente devido ao termo quadrático (HASTIE; TIBSHIRANI; FRIEDMAN, 2009). A seguir é apresentada a equação que representa a MSE, onde n é o número total de amostras, y_i é o valor real da amostra e $\hat{y}_i$ é o valor predito da amostra.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{7}$$

A *Cross-Entropy Loss*, também conhecida como *Log Loss* ou *Logarithmic Loss*, é uma função de perda utilizada em problemas de classificação. Esta função mede a diferença entre duas distribuições de probabilidade: a distribuição real dos rótulos da classe e a distribuição prevista pelo modelo. Esta função penaliza previsões de baixa confiança que estão incorretas, ou seja, quanto mais distante a previsão estiver do valor verdadeiro maior será a penalização. Esta propriedade torna a *Cross-Entropy Loss* particularmente eficaz em ajustes de modelos de classificação, pois incentiva o mo-

delo a fazer previsões confiantes e corretas (GOODFELLOW; BENGIO; COURVILLE, 2016). A seguir é apresentada a equação que representa a *Cross-Entropy Loss* para a classificação multiclasse, onde n é o número de amostras, y_i é o rótulo verdadeiro da amostra, p_i é a probabilidade precista pelo modelo para a classe positiva na amostra, C é o número total de classes, $y_{i,c}$ é um indicador binário que vale 1 se a amostra pertence à classe c e 0 caso contrário, e $p_{i,c}$ que é a probabilidade prevista pelo modelo para a classe c na amostra.

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{c=1}^{C} y_{i,c}\log(p_{i,c}) \tag{8}$$

A *Charbonnier Loss* (WU et al., 2017) é uma função de perda robusta que é utilizada para minimizar os efeitos de outliers em problemas de regressão, mas também é muito utilizada em tarefas de visão computacional, como a reconstrução de imagens. Esta função pode ser expressa matematicamente pela seguinte equação:

$$l_y(y, \hat{y}) = E_{z,y\ p_{data}(z,y)}(p(y - G(z))) \tag{9}$$

Na equação, y denota o quadro original de alta qualidade, $G(z)$ o quadro decodificado filtrado e $p(x) = \sqrt{x^2 + \epsilon^2}$ é a penalidade da função *Charbonnier*.

3.5 Redes *Feedforward* com Camada Única: O Perceptron

O Perceptron foi um modelo desenvolvido pelo cientista Frank Rosenblatt em 1958 (ROSENBLATT, 1958), inspirada em trabalhos anteriores de Warren McCulloch e Walter Pitts. O neurônio do Perceptron atua através de uma combinação linear das entradas com os pesos. Ao cálculo da combinação linear, é adicionado um valor de *bias* ou viés, que é um termo constante independente que desloca a função de saída para que a mesma não passe pelo ponto de origem dos eixos, ajustando a saída pelo aumento ou diminuição da entrada da função de ativação (HAYKIN, 2009). A função de ativação é uma função limitadora que atua como um classificador linear, ou seja, um classificador binário (HAYKIN, 2009). Um classificador binário, por sua vez, é uma função que decide se uma entrada pertence ou não a uma dentre duas classes. Assim, o Perceptron é usado para o treinamento supervisionado.

Como se pode observar pela Figura 9 a entrada do Perceptron é constituída por um vetor $X = [x_1, x_2, ..., x_n]$, mais o bias b e um vetor de pesos $W = [w_0, w_1, ..., w_n]$. Dessa forma, cada uma das entradas é multiplicada por um peso, que são os parâmetros a serem aprendidos pelo modelo. O resultado das multiplicações passam por um somatório e esse resultado, então, passa pela função de ativação, comprimindo o valor de saída entre dois valores possíveis. Matematicamente, este processo pode ser representado pelas equações (10), (11) e (12):

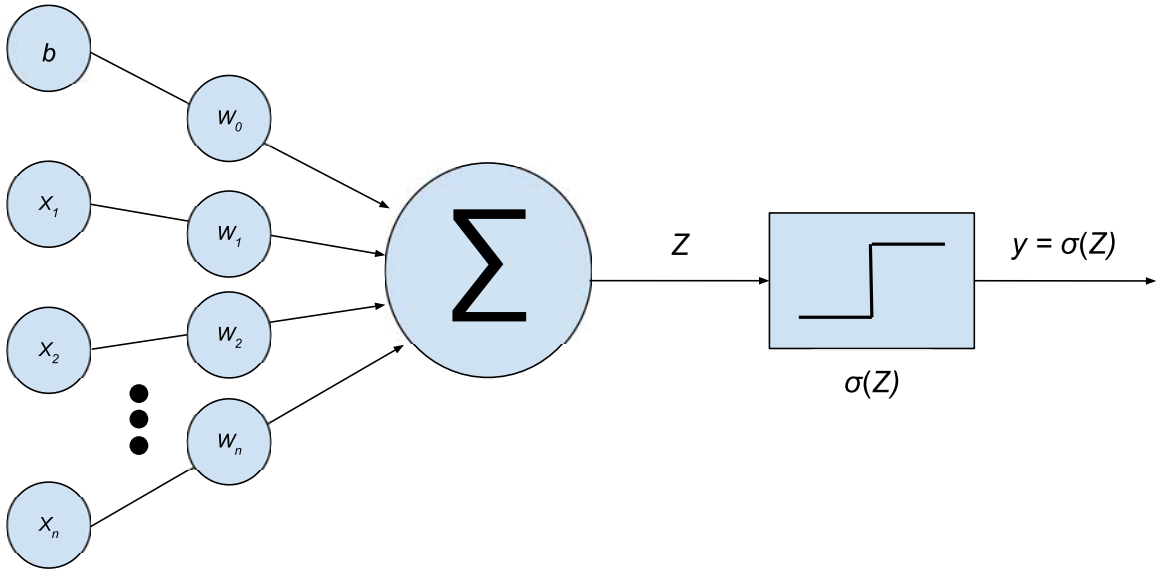

Figura 9 – Exemplo de arquitetura do Perceptron

$$b = 1.w_0 \tag{10}$$

$$Z = \sum_{j=1}^{n} w_j x_j + b = X.W^T + b \tag{11}$$

$$y = \sigma(Z) \tag{12}$$

O problema com o Perceptron é que ele é muito simples e só aprende funções referente a problemas linearmente separáveis, ou seja, onde apenas uma reta não é suficiente para separar as classes do problema. Assim, funções como XOR não são possíveis de serem separadas pelo Perceptron, como é possível observar na Figura 10.

3.6 Redes *feedforward* com múltiplas camadas: *Multilayer Perceptron*

A impossibilidade de treinar redes neurais com mais de uma camada e a inseparabilidade de problemas não linearmente separáveis foram resolvidas com o algoritmo de retro-propagação apresentado por Rumelhart, Hinton e Williams, em 1986. Este algoritmo permitiu que fossem construídas redes com múltiplos perceptrons. Apesar da retro-propagação poder ser aplicada a redes com qualquer número de camadas, foi mostrado que apenas uma camada oculta é suficiente para aproximar qualquer função com um número finito de descontinuidades para precisão arbitrária, desde que

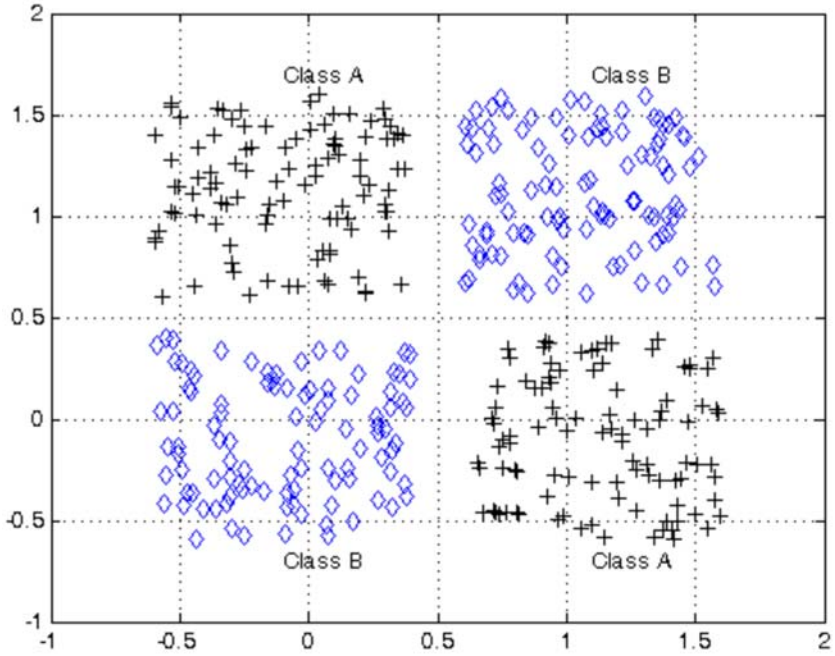

Figura 10 – Gráfico representando um problema XOR (POTOCNIK, 2012).

as funções de ativação das camadas ocultas sejam não lineares. Este é conhecido como o Teorema da Aproximação Universal para Redes Neurais (CYBENKO, 1989) (HORNIK; STINCHCOMBE; WHITE, 1989).

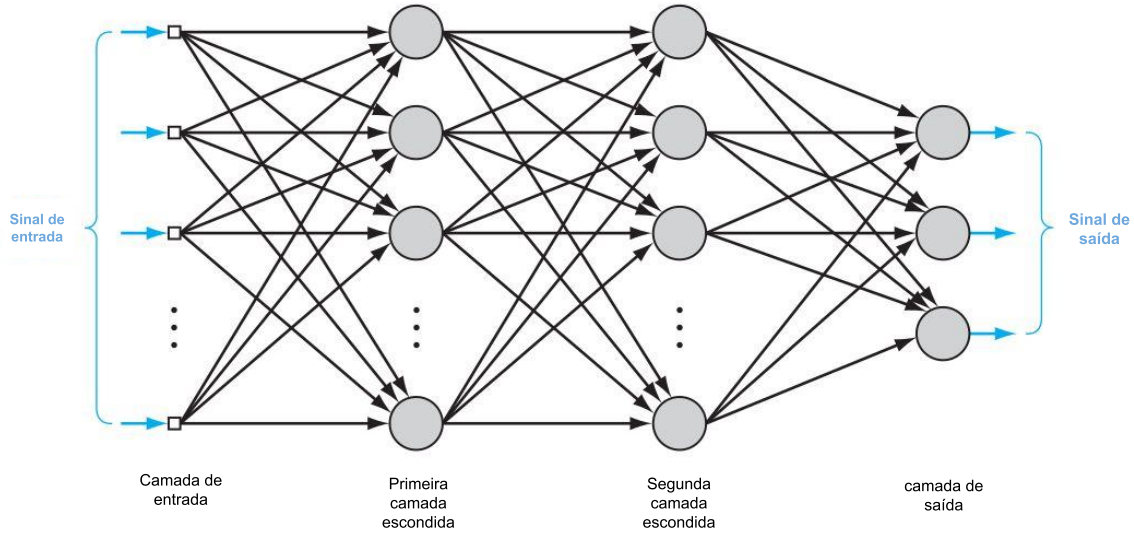

Figura 11 – Representação da arquitetura de um Multilayer Perceptron. Adaptada de Haykin (2009).

Na Figura 11, está representada uma arquitetura de um *Multilayer Perceptron* (MLP) genérico. Ela é composta por uma camada de entrada, onde não há nenhuma computação, duas camadas ocultas, onde de fato há a computação, e uma camada de saída. Como é possível observar, os neurônios de uma mesma camada não se comunicam entre si, apenas com os neurônio das camadas adjacentes. Na MLP, o

treinamento é feito de forma supervisionada, ou seja, é apresentado um vetor de entradas, que será usado para o processo de aprendizagem, e um vetor de saída, que é o resultado esperado e será usado para comparar com a saída dada pelo modelo. A partir da comparação, um erro é gerado, que é então retro-propagado pela rede de forma a ajustar os pesos para que a cada iteração do treinamento o valor de saída dado pela rede seja o mais próximo possível do vetor de saída com o resultado esperado, aproximando o erro cada vez mais de zero.

3.7 Convolução Espacial

A operação de convolução é aplicada a nível de píxel. Ela é constituída por um filtro de tamanho *N*×*N* ímpar, pois dessa forma há um pixel central facilitando a operação. Assim, o novo pixel criado tem as mesmas coordenadas que o pixel central do filtro, que percorre cada pixel de uma imagem de entrada. Cada posição do filtro possui um peso. Esse conjunto de pesos representa uma função que será aplicada sobre a imagem com o objetivo de extrair alguma característica, como as bordas da imagem, por exemplo. A Figura 12 exemplifica a operação de convolução. Onde w representa os coeficientes do filtro a ser aplicado, ou seja, a máscara; f representa os píxels da seção da imagem sob a máscara, ou seja, o pedaço da imagem que será aplicada a função de filtragem.

A operação de convolução pode ser descrita matematicamente da seguinte forma (DAI et al., 2017): dado um filtro 3×3 com dilatação 1 $R = (-1,-1),(-1,0),...,(0,1),(1,1)$. Para cada localização p_0 no mapa de saída y, onde p_n enumera as posições em R:

$$y(p_0) = \Sigma_{p_n \in R} w(p_n).x(p_0 + p_n) \tag{13}$$

3.8 Convolução Deformável

A convolução deformável foi desenvolvida para suprir a dificuldade das convoluções padrões de modelar transformações e variações geométricas devida à sua estrutura fixa. Esta técnica tem o mesmo princípio da convolução padrão, na qual uma matriz de filtro percorre toda a imagem de entrada, porém é adicionado à posição de amostragem original do filtro um *offset* que desloca a posição original. Dessa forma, a grade regular R (representada na subseção anterior) é aumentada com *offsets* $\Delta p_n | n = 1,...,N$, onde $N = |R|$. Assim, a equação 13 é adaptada para a seguinte forma (DAI et al., 2017):

$$y(p_0) = \Sigma_{p_n \in R} w(p_n).x(p_0 + p_n + \Delta p_n) \tag{14}$$

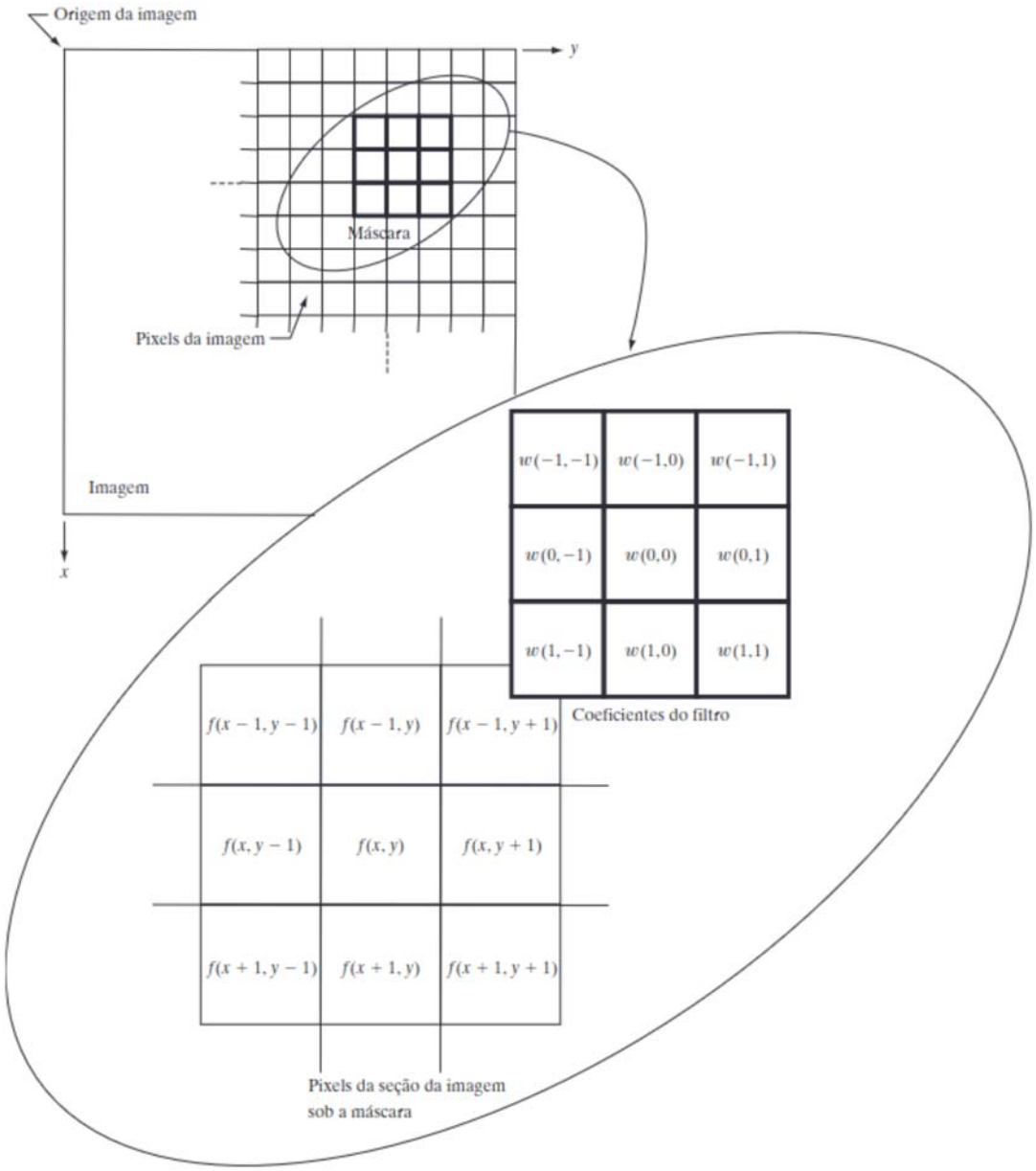

Figura 12 – Funcionamento da operação de convolução utilizando uma máscara 3×3 (GONZALES; WOODS, 2010)

A Figura 13 exemplifica o processo de convolução deformável. O problema é que os *offsets* gerados são fracionários, sendo necessária a aplicação de uma interpolação bilinear para obter a posição exata do pixel deslocado.

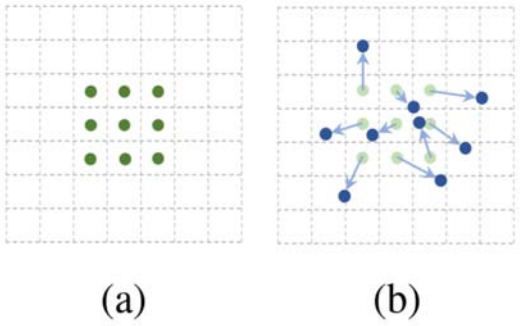

Figura 13 – Ilustração das localizações de amostragem em uma convolução 3×3 padrão e deformável: (a) grade regular da posição de amostragem de uma convolução padrão; (b) localização das posições de amostragem deformadas (DAI et al., 2017).

3.9 Convolução Transposta

A convolução transposta, também chamada de convolução fracionada ou ainda deconvolução, é uma operação utilizada em CNNs durante o processo de *upsampling*. Ela pode ser vista como uma operação inversa à convolução quanto à dimensionalidade da entrada e da saída. Dessa forma, a entrada da convolução transposta tem altura e largura menor que a saída. A ideia é reconstruir a dimensionalidade original existente antes da operação de convolução ter sido aplicada. Porém, isso não quer dizer que os valores originais são reconstruídos. Este tipo de operação é frequentemente utilizada em tarefas como a reconstrução de imagens ou na parte de decodificação de arquiteturas, como as encontradas em redes do tipo U-Net.

A convolução transposta possui parâmetros como o tamanho da máscara, ou *kernel* (k), o tamanho do *stride* (s) e o *zero-padding* (p). Porém, estes são usados para definir outros dois parâmetros, um que regula a quantidade de zeros adicionados entre cada linha e cada coluna da entrada z e um outro que determina quantas bordas de zeros, de fato, serão adicionados na imagem de entrada p'. Com essas modificações na imagem de entrada, o *stride* se torna unitário. Onde:

$$z = s - 1 \tag{15}$$

e

$$p' = k - p - 1 \tag{16}$$

Já o tamanho do mapa de características de saída o pode ser calculado de acordo com a seguinte fórmula, dado o tamanho da entrada i, tamanho do *kernel* k, tamanho

do *zero-padding* p e o *stride* s:

$$o = (i - 1) \times s + k - 2p \tag{17}$$

A convolução transposta apresentada aqui é representada como uma convolução direta sobre uma entrada adicionada de zeros em suas linhas e suas colunas. Entre as formas de se implementar uma convolução transposta, a utilizada para explicação é a forma menos eficiente (DUMOULIN; VISIN, 2018). Porém, através dela é possível englobar todos os casos específicos de convolução transposta.

3.10 Redes Neurais Convolucionais

As Redes Neurais Convolucionais (*Convolutional Neural Network* – CNN) são um tipo especial de rede neural profunda onde o objeto de trabalho são vetores com características espaciais, ou seja, possuem uma dimensão em altura e uma em largura. São comumente utilizadas para tarefas de segmentação, classificação de imagens, agrupamento por similaridade, reconhecimento de objetos/pessoas e melhoria de qualidade de imagens/vídeos. Essas redes foram inspiradas pelo mecanismo de percepção visual natural de criaturas vivas (LI et al., 2022).

Uma CNN voltada para a tarefa de classificação é basicamente composta por três tipos de camadas: camada convolucional, camada de subamostragem ou *pooling* e camadas totalmente conectadas. A Figura 14 mostra um exemplo de arquitetura de rede neural convolucional.

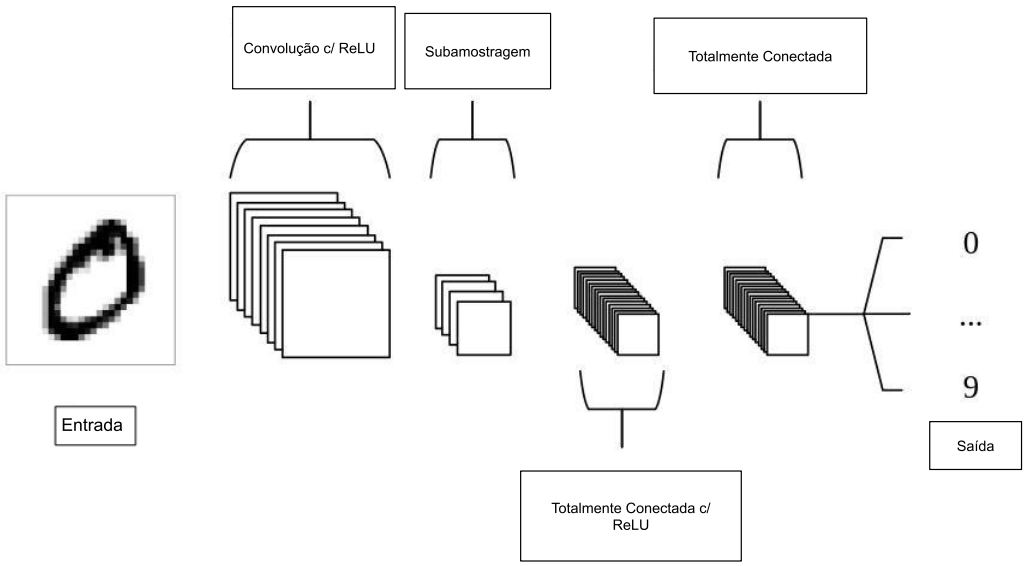

Figura 14 – Exemplo de arquitetura de uma rede neural convolucional (O'SHEA; NASH, 2015).

As camadas convolucionais aplicam filtros para a extração de características da imagem. Diferente das técnicas tradicionais, onde esses pesos são definidos manualmente, nas camadas convolucionais da rede esses pesos são encontrados auto-

maticamente conforme a rede aprende. Nas primeiras camadas de uma CNN são extraídas características mais simples, como linhas, pontos e bordas. Conforme se avança pela rede, características mais complexas vão sendo extraídas, como círculos e outros formatos.

Uma camada de CNN pode ser composta por mais de um filtro ao mesmo tempo e cada filtro terá o seu mapa de ativação, ou mapa de características correspondente, que será empilhado ao longo da dimensão de profundidade para formar o volume de saída completo de uma camada da rede (O'SHEA; NASH, 2015). Cada neurônio é conectado a uma pequena região cuja dimensionalidade é referida como campo receptivo do neurônio.

A camada de convolução tem três hiperparâmetros que permitem a otimização do processo. São eles: **profundidade**, ***stride*** e ***zero-padding***. A **profundidade** se refere ao número de filtros convolucionais aplicados. Uma menor quantidade de filtros pode otimizar o processamento da rede ao custo de uma menor capacidade de reconhecimento de padrões nas imagens. O ***stride*** se refere ao passo que o filtro dará sobre a imagem; por exemplo, um *stride* de valor 1 fará com que o filtro se mova uma unidade, ou um pixel, por vez. Isso afeta diretamente no campo receptivo, pois quanto menor o *stride* mais sobrepostos são os campos receptivos produzindo ativações extremamente largas (KRIZHEVSKY; SUTSKEVER; HINTON, 2012). O ***zero-padding*** é o ato de preencher as bordas da entrada com zero, auxiliando no controle da dimensionalidade do volume de saída.

Os hiperparâmetros citados influenciam na dimensionalidade espacial da saída das camadas convolucionais. Para calcular a nova dimensão espacial, a equação 18 pode ser usada:

$$O = \frac{(V - R) + 2Z}{S + 1} \tag{18}$$

Onde O representa a nova dimensão da saída, V representa o tamanho do volume de entrada (altura $\times$ largura $\times$ profundidade), *R* representa o tamanho do campo receptivo, *Z* é a quantidade de zeros usados no preenchimento e *S* é o *stride* (O'SHEA; NASH, 2015).

As camadas de subamostragem têm por objetivo reduzir a dimensionalidade do mapa de característica selecionando características específicas de uma região. Por exemplo, a subamostragem por máximo (*maxpooling*), tem por objetivo selecionar as características que mais se destacam, ou seja, apresentam o maior valor na região. Esse processo reduz a carga computacional para as próximas camadas. As camadas totalmente conectadas são responsáveis pelo processo de classificação da imagem a partir das características extraídas nas camadas anteriores.

3.11 Redes Adversárias Generativas

As Redes Adversárias Generativas (*Generative Adversarial Networks* – GAN) são um tipo de algoritmo de inteligência artificial desenvolvido para solucionar o problema de modelagem generativa (GOODFELLOW et al., 2020), onde exemplos de treinamento x são desenhados a partir de uma distribuição desconhecida $p_{data}(x)$. Assim, o objetivo de um algoritmo de modelagem generativa é aprender uma distribuição $p_{model}(x)$ que aproxima $p_{data}(x)$ o máximo possível.

O funcionamento das GANs são baseadas em um jogo, no sentido de teoria dos jogos, entre dois modelos de AM (GOODFELLOW et al., 2020), onde as duas redes competem entre si. A rede geradora é responsável por gerar uma imagem a partir de um ruído aleatório com o objetivo de enganar a rede discriminadora. Esta é responsável por identificar se a imagem gerada faz parte da distribuição $p_{data}(x)$, indicando se é uma imagem verdadeira, quando faz parte, ou se é falsa.

Quando a imagem gerada é indicada como falsa o erro é retro-propagado pela rede geradora, fazendo com que a rede aprenda a gerar imagens cada vez mais próximas das presentes no conjunto de dados, ou seja, aprenda uma distribuição $p_{model}(x)$ que se aproxima da representação da distribuição do conjunto de dados $p_{data}(x)$. Consequentemente, a rede discriminadora também aprende cada vez mais a diferenciar uma imagem real de uma falsa.

3.12 U-Net

A U-Net é uma arquitetura de rede neural que foi originalmente proposta para a segmentação semântica de imagens médicas (RONNEBERGER; FISCHER; BROX, 2015). Sua arquitetura pode ser vista como uma rede *encoder* seguida de uma rede *decoder*. Dessa forma, ela é composta por uma etapa de contração e uma etapa de expansão, que conferem a ela o formato de U, como pode ser observado pela Figura 15.

A etapa de contração segue a estrutura típica de uma rede neural convolucional, ou seja, é composta por duas convoluções de *kernel* tamanho 3×3 seguidas da aplicação da função de ativação ReLU e da função de *maxpooling* de tamanho 2×2 com *stride* 2 para *downsampling*. A cada passo de *downsampling* o número de canais de características é dobrado. Nesta etapa, a rede atua como um extrator de características e aprende uma representação abstrata da imagem de entrada.

Na etapa de expansão todo passo consiste em um *upsampling* dos mapas de características, seguido por uma convolução transposta de *kernel* 2×2, que corta pela metade o número de canais de características. Na sequência é aplicada uma concatenação com o mapa de característica da etapa de contração correspondente, chamada de *skip connection*, o que ajuda a evitar o problema de desaparecimento de gradiente.

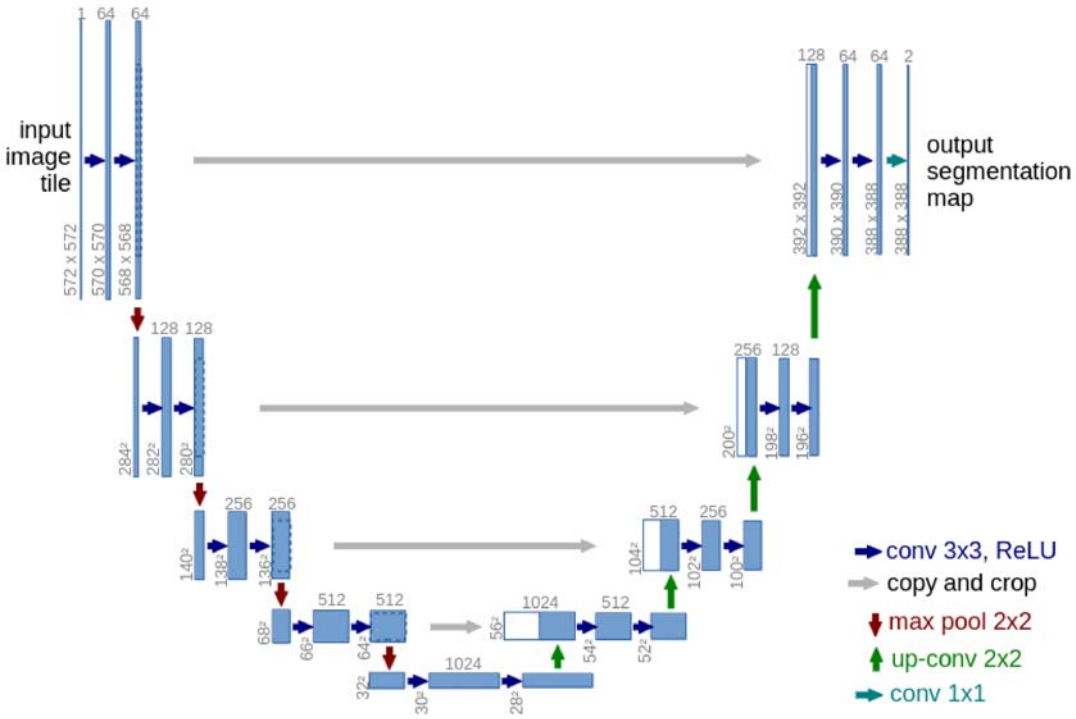

Figura 15 – Arquitetura da U-Net (RONNEBERGER; FISCHER; BROX, 2015).

E finalmente duas convoluções de *kernel* 3×3, cada uma seguida por uma ReLU. A parte de *upsampling* tem um grande número de canais de características, o que permite à rede propagar informações de contexto a camadas de resoluções mais altas.

3.13 Aprendizagem Multi-Domínio

O avanço de algoritmos de aprendizado de máquina foi possível devido à grande quantidade de dados disponíveis. O problema é que os paradigmas de treinamento atuais são restritos na variedade de dados que podem lidar. A maioria dos métodos trabalham com dados advindos de um domínio específico e dessa forma o modelo acaba por aprender o viés inerente ao conjunto de dados. O domínio pode ser contextualizado a partir do seguinte exemplo: Em uma tarefa de classificação de objetos em imagens, se deseja classificar um violão. Porém as imagens que contém um violão tem características diferentes. Umas tem características de *cartoon*, outras são fotos de violões, outras são pinturas de violões, etc. Cada grupo de características define um domínio diferente. Dessa forma, um classificador treinado sobre imagens de violões em pinturas, dificilmente vai classificar corretamente os violões dos outros domínios.

Assim, os modelos apresentam bom desempenho nas tarefas específicas para os domínios nos quais foram treinados, o que acaba por limitar severamente a sua habilidade de generalização quando processam dados de domínios novos e previamente não vistos (YOGATAMA et al., 2019).

Os desafios previamente mencionados tornam-se mais acentuados ao lidar com dados provenientes de domínios altamente variáveis, especialmente quando é necessário desenvolver um único modelo capaz de lidar com múltiplos conjuntos de dados distintos. Treinar um único modelo para abranger essa diversidade de domínios impossibilita a captura de nuances específicas de cada um. A abordagem comum para enfrentar essa questão envolve a recriação de um modelo para cada domínio e a aplicação de cada modelo aos dados correspondentes. Contudo, essa metodologia revela-se altamente ineficiente.

Para resolver este problema surgiu o paradigma de aprendizagem multi-domínio, que permite um único modelo aprender nuances específicas sobre uma variedade de domínios. Assim, surgiu a rede *Multi-Domain Network* (MDNet) (NAM; HAN, 2016) que propõe a separação de vídeos em domínios anotados, de forma que cada domínio siga um caminho diferente nas camadas ramificadas. Seguindo a mesma proposta, foi desenvolvida a arquitetura *Branch-Activated Multi-Domain Convolutional Neural Network for Visual Tracking* (BAMDCNN) (CHEN et al., 2018), onde o conceito multi-domínio é utilizado para o rastreamento de objetos em um vídeo.

Nestas técnicas, é explorada uma arquitetura híbrida onde parte da rede é agnóstica ao domínio, sendo compartilhada entre todos os domínios, e a outra parte é ramificada, onde cada ramo se especializa em capturar nuances especificas ao domínio. A Figura 16 mostra um exemplo da estrutura das arquiteturas multi-domínio. Para que um ramo seja ativado, é necessário uma forma de agrupar os dados de entrada para o treinamento, de forma a possibilitar identificar a qual domínio o lote de treinamento pertence.

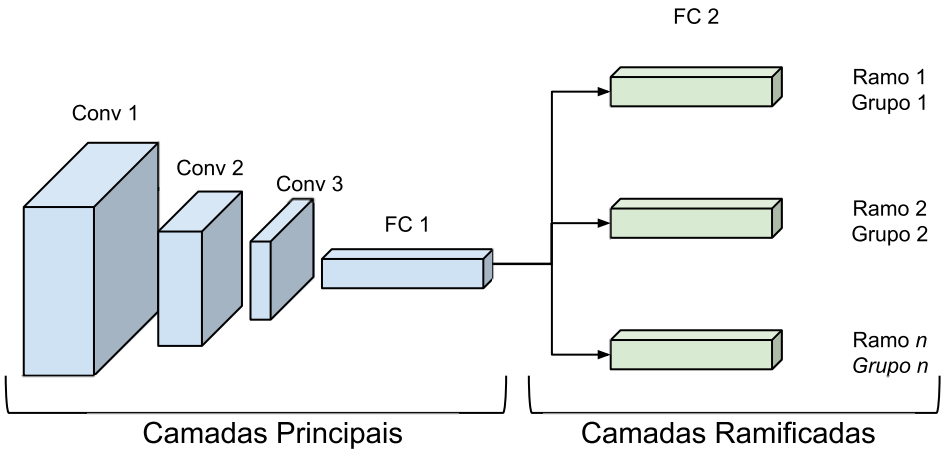

Figura 16 – Exemplo da arquitetura multi-domínio.

4 REVISÃO DO ESTADO-DA-ARTE

Neste capítulo é apresentada uma revisão dos trabalhos relacionados para a melhoria de qualidade de vídeos comprimidos. Na primeira seção, é discutido o panorama geral das soluções baseadas em redes neurais, sendo divido em soluções baseadas em um único quadro e soluções baseadas em múltiplos quadros. Na segunda seção, é apresentada com maior detalhamento a solução estado-da-arte.

4.1 Soluções baseadas em Redes Neurais

4.1.1 Soluções baseadas em um único quadro

As primeiras arquiteturas para remoção de artefatos em vídeos são baseadas em arquiteturas de remoção de artefatos de imagens. Dessa forma, as arquiteturas para vídeos se utilizavam das mesmas técnicas, ou seja, utilizavam técnicas de melhoria baseadas em um único quadro, sejam elas filtragens espaciais ou qualquer outra técnica que se baseia unicamente nas informações presentes no próprio quadro que está sendo melhorado, sem tirar vantagem de informações presentes em quadros adjacentes.

A primeira arquitetura voltada para o problema de redução de artefatos de compressão em imagens baseada em CNN foi a *Artifact Reduction Convolutional Neural Network* (ARCNN) (DONG et al., 2015). Esta rede foi criada a partir da inclusão de uma camada de mapeamento na rede *Super-Resolution CNN* (SRCNN). A camada de mapeamento é responsável por mapear os mapas de características ruidosos para um espaço de característica relativamente mais "limpo", o que é equivalente a um processo de remoção de ruídos dos mapas de características (DONG et al., 2015). A SRCNN é uma rede voltada para o problema de aumento de resolução de imagens. A ARCNN, portanto, consiste em quatro camadas convolucionais, cada uma responsável por uma operação específica. A primeira camada é responsável pela extração de características; a segunda, por aprimorar as características; a terceira, por fazer um mapeamento; e a última, pela reconstrução da imagem.

Alguns estudos posteriores utilizaram a arquitetura ARCNN como base, como o

trabalho de Dai; Liu; Wu (2017), um dos primeiros estudos a utilizar CNN como abordagem ao problema de melhoria de qualidade de vídeo (*Video Quality Enhancement* – VQE), em que foi desenvolvida a arquitetura *Variable-Filter-Size Residue-Learning CNN* (VRCNN), composta de um modelo de CNN de quatro camadas. Na proposta, o modelo é utilizado como um filtro *in-loop* e substitui os filtros SAO e de deblocagem do padrão HEVC.

Tomando como base a ARCNN, os autores em (KUANAR; CONLY; RAO, 2018) desenvolveram, como um filtro de pós-processamento, a *Multiview Deep Convolutional Neural Network* (MDCNN), com camadas de convolução e deconvolução simétricas, totalizando uma arquitetura de 9 camadas. As camadas de convolução fazem o *downsampling* da entrada sucessivamente em abstrações de menor tamanho. Já as camadas de deconvolução fazem o *upsampling* das abstrações para a resolução original.

Yang; Xu; Wang (2017) desenvolveram um filtro de pós-processamento para vídeos codificados no padrão HEVC, também baseado na ARCNN. A rede *Decoder-side Scalable Convolutional Neural Network* (DSCNN) foi desenvolvida levando em consideração o modo de predição usado para codificar o quadro. Dessa forma, a rede possui duas sub-redes, uma voltada para processar quadros codificados com a predição intraquadro (I), chamada DS-CNN-I, e a outra voltada para melhorar quadros codificados com a predição interquadros (P) e com a predição interquadros bi-direcional (B), chamada DS-CNN-B. As duas sub-redes possuem cinco camadas convolucionais, uma a mais que a ARCNN, voltada para a remoção de ruídos das características extraídas das camadas anteriores. Ao mapa de característica gerado na sub-rede DS-CNN-B é concatenado o mapa de característica gerado pela camada de mesmo número da sub-rede DS-CNN-I para o processamento pela camada seguinte.

Wang et al. (2018) propôs um filtro *in-loop* para o padrão HEVC que utiliza uma arquitetura baseada em redes residuais denominada *Dense Residual Convolutional Neural Network* (DRN), que utiliza um novo tipo de bloco residual denso composto por uma camada convolucional 1×1, seguida por duas camadas convolucionais 3×3 com uma camada de ativação entre elas. A saída da convolução 1×1 é adicionada à última camada convolucional 3×3 e então concatenada com a entrada original para gerar a saída final da unidade. Dessa forma, a DRN é composta por blocos de camadas convolucionais 3×3 e uma série de blocos residuais densos.

Zhang et al. (2018) também desenvolveu uma arquitetura que utiliza unidades residuais denominadas *Residual Highway Convolutional Neural Network* (RHCNN). Em sua arquitetura foram utilizadas unidades chamadas de *highway units* que são compostas por camadas de convoluções 3$\times$3 e um atalho composto por uma camada identidade, ou seja, os valores de saída são os mesmos da entrada. O atalho previne o problema de desaparecimento de gradiente e permite que a rede passe detalhes do

início para o fim da rede, o que pode ser benéfico para recuperar imagens. Os atalhos identidade são usados não somente nas *highway units*, mas também é adicionada na rede como um todo, do início ao fim.

Kang; Kim; Lee (2017) propõem um filtro *in-loop* que utiliza informações do laço de codificação no processo de melhoria de qualidade. Para isso, desenvolveu a *multi-modal/multi-scale convolution neural network* (MMS-Net) que utiliza informações de cada *Coding Tree Unit* (CTU) para dar suporte ao processo de aprendizado. Essa informação é inserida na forma de uma imagem onde as bordas das CTUs têm o valor '2' e o interior tem o valor '1'. A partir dessa informação, o modelo foi capaz de se concentrar nas bordas das CTUs eliminando o artefato de bloco. A arquitetura proposta consiste em duas sub-redes de diferentes escalas. A sub-rede de escala grosseira recupera a imagem degradada a partir da imagem de entrada que foi reduzida pela metade. A sub-rede de escala fina recebe como entrada tanto a imagem reconstruída da escala grosseira quanto a imagem original degradada e retorna como saída a imagem restaurada.

Jin et al. (2018) propõe a *Multi-level Progressive Refinement Networks through an adversarial training approach* (MPRGAN), que funciona como um filtro *in-loop* para a codificação intraquadro, baseado na arquitetura GAN, explicada na seção 3.11. Dessa forma, a arquitetura é composta pela rede de melhoria, que atua como a rede geradora, e a rede discriminadora. A rede de melhoria tem o papel de predizer o resíduo entre o quadro original (*ground truth*) e o quadro de entrada (reconstruído pelo decodificador). O resíduo predito é somado de volta ao quadro de entrada, gerando o quadro melhorado.

4.1.2 Soluções baseadas em múltiplos quadros

Arquiteturas visando aprimorar a qualidade visual por meio de uma única imagem têm adotado a incorporação de camadas de CNN progressivamente mais profundas, visando alcançar resultados superiores. Entretanto, essa abordagem acaba por aumentar a quantidade de parâmetros envolvidos durante o treinamento, resultando em um aumento no custo computacional (DENG et al., 2020). Com isso, alguns autores optaram por investigar a inserção de informações temporais para o treinamento de modelos de melhoria de qualidade de vídeo.

As primeiras soluções baseadas em CNN que foram propostas para VQE realizavam o processamento de cada quadro do vídeo de forma individual. Como esses modelos evoluíram de algoritmos utilizados no processamento de imagem, somente eram observadas características espaciais durante o treinamento, não analisando a correlação temporal existente entre quadros. Estudos mais recentes têm como proposta o processamento de múltiplos quadros, utilizando informações de quadros vizinhos para melhorar a qualidade de um quadro central, desta forma englobando informa-

ções espaço-temporais.

Em geral, tais modelos tiram vantagem de quadros na vizinhança temporal para melhorar um quadro alvo, pois assume-se que os quadros vizinhos possuam informações úteis que possam ser usadas no processo de restauração do quadro alvo. Na maioria dos trabalhos, três módulos são identificados na estrutura de um modelo para remoção de artefatos de compressão de vídeos: módulo de alinhamento, módulo de fusão e módulo de reconstrução (ROTA et al., 2022).

O módulo de alinhamento é usado para alinhar os quadros de entrada com o quadro alvo, de forma que a representação obtida esteja espacialmente alinhada. Essa etapa considera o fato que diferentes observações do mesmo objeto ou cena provavelmente estão disponíveis nos quadros temporalmente adjacentes em um vídeo, mas com um pequeno deslocamento (MENG et al., 2019). Nesta etapa, os quadros adjacentes passam por um processo de estimação de movimento em relação ao quadro que vai ser melhorado, gerando vetores de deslocamento. Em posse dos vetores de movimento, os quadros adjacentes passam pela etapa de compensação de movimento, alinhando as informações espaciais com as do quadro alvo. Devido à movimentação dos objetos de um quadro para o outro, a não aplicação da etapa de estimação e compensação de movimento pode introduzir interferências na rede (MENG et al., 2019). Para a etapa de estimação de movimento, Meng et al. (2019) utiliza uma rede que estima o fluxo óptico do quadro ajacente em relação ao quadro alvo. Como os valores gerados não são inteiros, de forma que encontre uma posição exata em pixeis para a compensação de movimento, uma interpolação bilinear é aplicada para corrigir a posição que ocorrerá a compensação do movimento.

O módulo de fusão agrega as representações alinhadas, fundindo as informações presentes nos múltiplos quadros. As técnicas desenvolvidas para o propósito de fusão das informações dos quadros podem ser principalmente classificadas em *early fusion*, *slow fusion* e *3D convolution* (TONG et al., 2019). Segundo Caballero et al. (2017), a *early fusion* é o método mais direto para uma CNN processar vídeos, onde toda a informação temporal é colapsada na primeira camada da rede. A *slow fusion* funde parcialmente as informações temporais em uma estrutura hierárquica e é lentamente fundida à medida que as informações avançam pela rede. Já a *3D convolution* é uma variação da *slow fusion* que força os pesos das camadas serem compartilhadas através da dimensão temporal. A Figura 17 mostra um exemplo dos três tipos de fusão. Por fim, o módulo de reconstrução usa as representações fundidas para reduzir artefatos e produzir o quadro restaurado.

O primeiro estudo que utiliza informações espaço-temporais para a melhoria de qualidade visual de um único quadro foi o de Yang et al. (2018), que propõe a arquitetura *Multi-Frame Quality Enhancement* (MFQE) tomando como base estudos semelhantes para tarefas de super-resolução. O MFQE utiliza uma abordagem de detecção

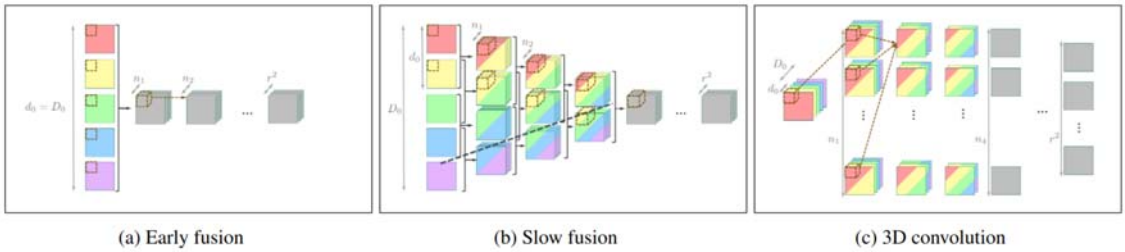

Figura 17 – Métodos espaço-temporais de fusão de quadros (CABALLERO et al., 2017).

de quadros vizinhos com alta qualidade (*Peak Quality Frames – PQF*) para serem utilizados como referência na melhoria de qualidade visual de um único quadro, sendo utilizado como um filtro de pós-processamento. A arquitetura MFQE possui três sub-redes, que fazem a detecção de PQFs, compensação de movimento dos quadros vizinhos e melhoria de qualidade visual de todos os quadros.

A detecção de PQFs é feita de forma a tirar proveito de uma estrutura chamada grupo de imagens (*Group of Pictures* – GOP) que comprime com qualidades diferentes os quadros de uma sequência com o intuito de manter a taxa de compressão constante afetando minimamente a QoE do usuário. A Figura 18 ilustra a estrutura de um GOP. Nela, os retângulos vermelhos representam os quadros I, os laranjas representam os quadros P e os amarelos representam os quadros B. O traçado em azul representa o nível de qualidade visual, medido em PSNR, e as barras rosas representam a taxa de bits de cada um dos quadros.

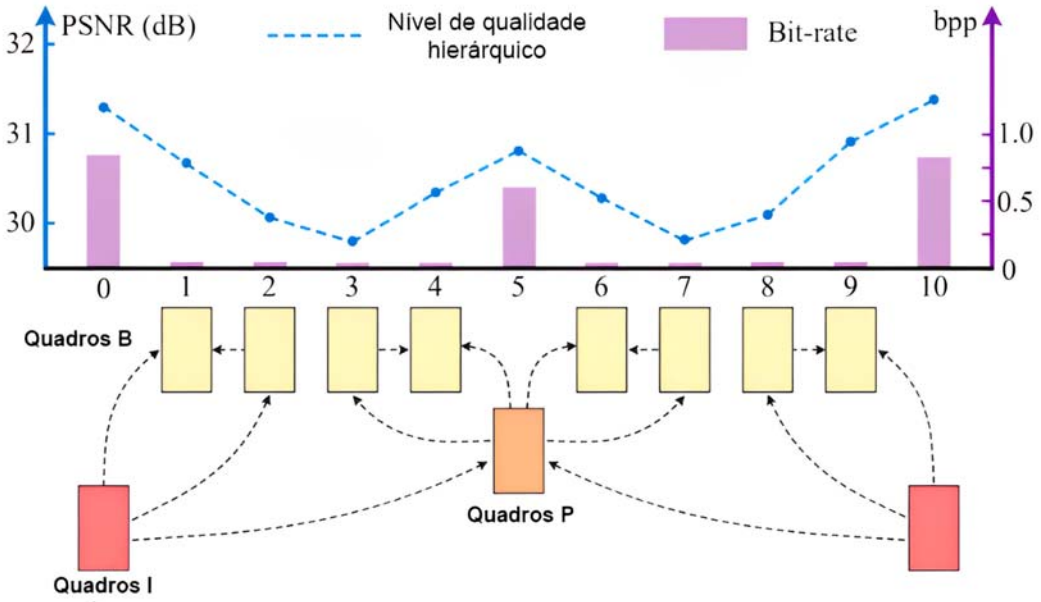

Figura 18 – Exemplo de GOP. Adaptado de Yang et al. (2020).

Analisando a Figura 18, é possível observar que para os quadros I e P é necessária uma alta taxa de bits para sua transmissão. Entretanto, esses quadros possuem a melhor qualidade tendo uma incidência de artefatos de compressão menor que os

quadros B. Dessa forma, os quadros I e P são utilizados como PQF para melhorar a qualidade dos quadros B. A Figura 19 exemplifica a flutuação de qualidade em um GOP.

A arquitetura MFQE aplica os conceitos de alinhamento e fusão no processamento da imagem. Neste tipo de arquitetura, são utilizados múltiplos quadros como entrada, os quais são alinhados através da abordagem de *optical flow* em nível de *pixel* para compensar os movimentos naturais em um vídeo. Esta abordagem provê o deslocamento de objetos entre dois quadros consecutivos, gerando vetores de movimento que apontam a coordenada de um pixel no primeiro quadro para a coordenada do mesmo pixel no segundo quadro, indicando a direção do movimento.

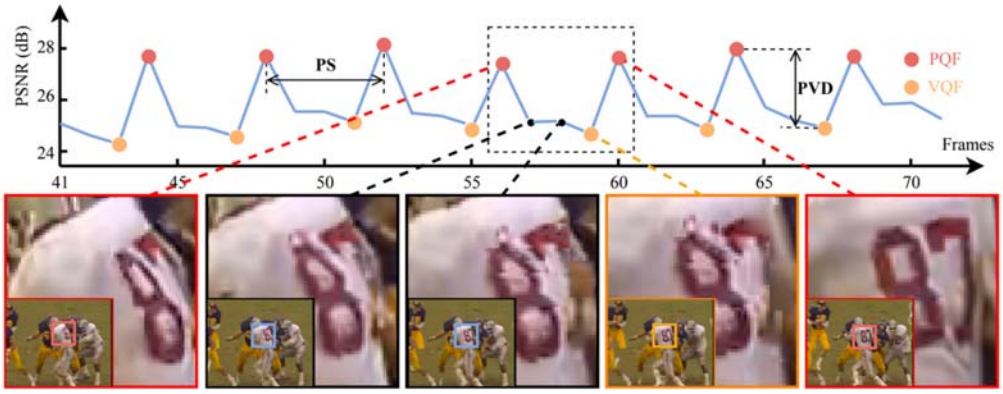

Figura 19 – Variação de qualidade dos quadros de um vídeo (XING et al., 2020)

Segundo Xue et al. (2019), a maioria dos algoritmos centrados em movimento segue um processo de duas etapas. Inicialmente, ocorre a estimativa do movimento entre os quadros, frequentemente realizada por meio de abordagens como o *optical flow*. Posteriormente, realiza-se a fusão desses quadros para gerar um novo quadro como resultado. No contexto do estudo de Xue et al. (2019), destaca-se a introdução da arquitetura *Task-Oriented Flow* (TOFlow). Diferenciando-se dos algoritmos convencionais na época do seu lançamento, a TOFlow emprega uma rede fundamentada em CNN. Essa abordagem não apenas facilita o processo de *optical flow*, mas também contribui para a redução de ruídos e artefatos de compressão nas bordas dos objetos. Apesar de muitos estudos utilizarem abordagens baseadas em *optical flow*, este método se demonstra sub-ótimo para o problema de VQE (DENG et al., 2020). Segundo Rota et al. (2022), métodos baseados em *optical flow* sofrem severamente quando há mudanças de luminosidade, rápida movimentação e oclusão de objetos de um quadro para o outro em um vídeo.

4.2 *Spatio-Temporal Deformable Fusion*

A *Spatio-Temporal Deformable Fusion* (STDF) (DENG et al., 2020) é a arquitetura estado-da-arte para VQE. Diferente dos métodos anteriores, que aplicam o *optical flow*

para o processo de estimação e compensação de movimento e que limitam o processamento a dois quadros por vez, a STDF aplica a convolução deformável, explicada na Seção 3.8. Com esta técnica, o processamento não é limitado a apenas dois quadros, dando possibilidade de serem extraídas informações úteis de quadros mais distantes para a melhoria de um quadro central.

A utilização da convolução deformável permitiu uma modelagem mais precisa da movimentação dos objetos entre as cenas. Como é possível observar na Figura 20, os pixels destacados em vermelho na convolução deformável (parte inferior da figura) acompanham a movimentação dos objetos, enquanto a estrutura rígida da convolução regular (parte central da figura) não permite o acompanhamento da movimentação dos objetos. Para esse processo, o STDF utiliza uma sub-rede dividida em duas etapas.

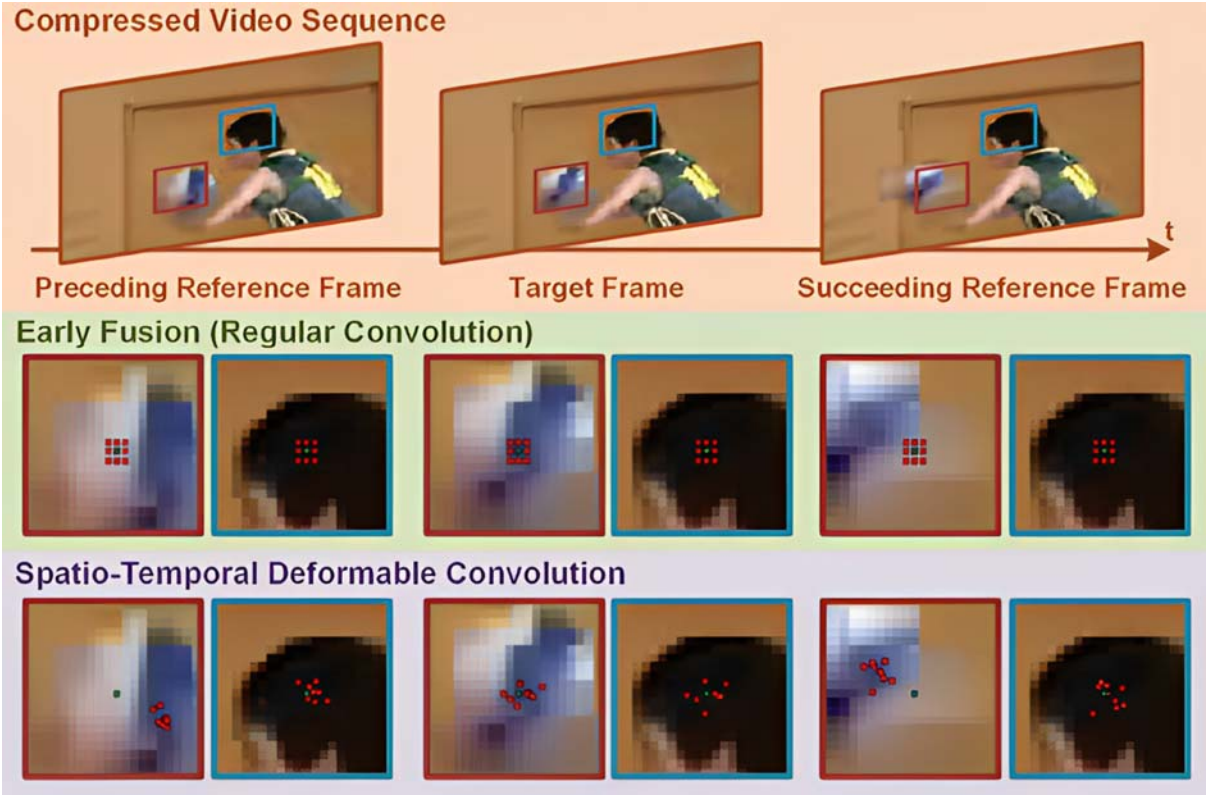

Figura 20 – Exemplo de comparação da convolução deformável com a convolução regular (DENG et al., 2020).

A primeira etapa consiste na rede de predição dos *offsets* que serão utilizados na aplicação da convolução deformável, onde é utilizada uma U-Net, explicada na seção 3.12. Na U-Net usada, é adotado o uso de camadas convolucionais e deconvolucionais com *stride* 2. Para as camadas convolucionais com *stride* 1 é utilizado *zero-padding*. Em todas as camadas, é adotado o uso da função de ativação ReLU, exceto pela última camada, que utiliza uma ativação linear. Esta etapa tem como entrada o quadro central a ser melhorado e R quadros passados e futuros de referência, onde R é um hiperparâmetro da rede. Dessa forma, a entrada da rede consiste em $2R + 1$ quadros concatenados. Segundo Deng et al. (2020), desde que os quadros consecu-

tivos sejam altamente correlacionados, a predição do *offset* para um quadro pode se beneficiar a partir dos outros quadros, levando a um uso mais efetivo da informação temporal. A segunda etapa consiste na aplicação da convolução deformável sobre os $2R + 1$ quadros de entrada, utilizando os *offsets* preditos pela etapa anterior. Neste momento, o processo de compensação de movimento e fusão das informações dos quadros é feita de forma implícita.

A segunda sub-rede do STDF é a rede de melhoria de qualidade, na qual, a partir do mapa de características gerado pela sub-rede STDF, vão ser aplicadas L camadas convolucionais com *stride* 1, refinando as informações fundidas. A saída desta etapa é um mapa residual que é somado ao quadro a ser melhorado, aproximando-o do quadro original. Segundo Kim; Lee; Lee (2016), as redes residuais convergem muito mais rápido do que as redes não residuais e durante a convergência apresentam performance superior. A Figura 21 mostra a arquitetura do STDF apresentada em (DENG et al., 2020). A rede possui um total de 305.000 parâmetros treináveis.

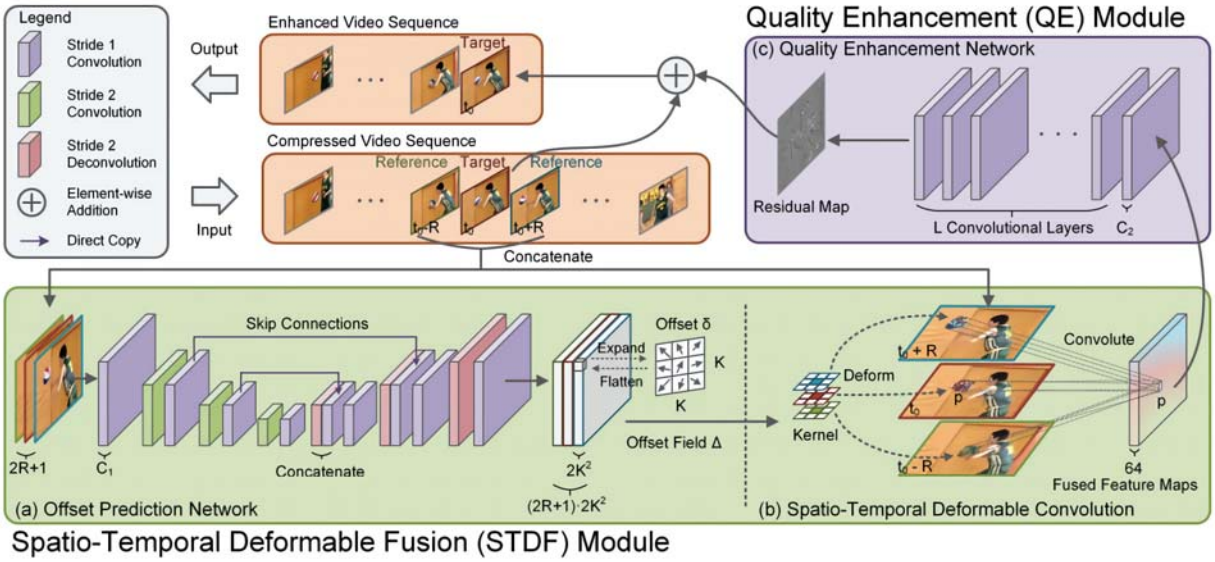

Figura 21 – Arquitetura do STDF (DENG et al., 2020).

A função de perda utilizada é a Soma dos Erros Quadrados (*Sum of Squared Error* – SSE) entre o quadro alvo melhorado $\hat{I}^{HQ}$ e o quadro original I^{HQ}, que pode ser expressa como mostra a equação a seguir:

$$L = \Sigma_{i=1}^{n}(\hat{I}_i^{HQ} - I_i^{HQ})^2 \tag{19}$$

5 ANÁLISES E SOLUÇÕES PROPOSTAS

Este capítulo inicialmente apresenta uma análise sobre o STDF que evidencia o problema a ser tratado. A seguir, são apresentadas as duas soluções propostas para o problema de melhoria de qualidade de vídeos comprimidos por diferentes codecs e configurações de QP/CQ.

Para o treinamento dos modelos apresentados, foi utilizado um computador com a seguinte configuração: processador AMD Ryzen 7 5700X, 32 GB de memória RAM, GPU NVIDIA Geforce RTX 3070 com 8 GB de VRAM. Em todos os casos, o treinamento foi conduzido com o otimizador Adam com $\beta_1 = 0.9$, $\beta_2 = 0.999$ e $\epsilon = 10^{-8}$ e taxa de aprendizagem de 0,0001.

5.1 Análise do Modelo STDF

O estudo apresentado nesta seção foi publicado no artigo de Kreisler et al. (2024), no qual a eficiência da arquitetura STDF (DENG et al., 2020) foi analisada em termos de melhoria de qualidade em diferentes cenários de teste com vídeos comprimidos por diferentes codecs e configurações de QP/CQ. A avaliação centrou-se na observação dos resultados alcançados pelo modelo STDF em termos de melhoria objetiva da qualidade de vídeos. A sessão de testes foi realizada usando o modelo treinado por (DENG et al., 2020), que está disponível publicamente[1]. O conjunto de dados de vídeo utilizado na análise é o mesmo empregado em (YANG et al., 2018), que compreende 126 vídeos sem compressão (RAW) de diferentes resoluções, dos quais 108 foram utilizados para treinamento do modelo e 18 para teste.

O objetivo do estudo foi analisar a capacidade de generalização do modelo disponibilizado por Deng et al. (2020) sob vídeos comprimidos por mais de um padrão com diferentes configurações de QP/CQ. Para isso, o *dataset* de teste, composto por 18 vídeos, foi totalmente comprimido usando os codecs HEVC, VVC, VP9 e AV1. Para a codificação HEVC, foi utilizado o software de referência HEVC Model (HM), versão 16.5, com a configuração *Low Delay P*. Para a codificação VVC, foi usado o software

[1] https://github.com/ryanxingql/mfqev2.0/wiki/MFQEv2-Dataset

de referência VVC Test Model (VTM), versão 13.0, também com a configuração *Low Delay P*. Para a codificação VP9, foi usado o software de referência *libvpx*, *hashcode* 1.12.0. Para a codificação AV1, foi usado o software de referência *libaom*, *hashcode* 3.3. Para HEVC e VVC, o QP foi definido para os valores 32 e 37, enquanto que para VP9 e AV1 o CQ foi definido como 43 e 55, sendo priorizado as configurações que geram mais degradação de acordo com as Condições Comuns de Teste (*Common Test Conditions* - CTC) (BOYCE; SUEHRING; LI, 2018). Assim, foram geradas 8 versões dos 18 vídeos de teste, totalizando 144 vídeos.

Todos os 144 vídeos foram melhorados usando o modelo STDF treinado e disponibilizado em (YANG et al., 2018) e o valor de $\triangle$PSNR foi calculado tomando como referência de cálculo o PSNR das sequências decodificadas não filtradas. Dessa forma, o cálculo do PSNR referência utiliza a sequência original (RAW) não compactada e a sequência decodificada que não passou por nenhum processo de melhoria. O segundo PSNR, para o cálculo do $\triangle$PSNR, utiliza a sequência RAW e a sequência decodificada e melhorada. O cálculo do PSNR é demonstrado pela Equação 1. Com esses dois valores de PSNR é possível calcular o $\triangle$PSNR pela subtração do PSNR que utiliza as informações das sequências decodificadas e não melhoradas com o PSNR que utiliza as informações das sequências decodificadas e que passaram pelo processo de melhoria.

A Tabela 1 mostra os resultados médios obtidos para todas as sequências de teste. A coluna *Decodificado* apresenta o PSNR médio dos vídeos decodificados (ou seja, antes de ser processado pelo modelo STDF). A coluna *Melhorado* apresenta o PSNR médio dos vídeos decodificados após filtragem pelo STDF. A coluna $\triangle PSNR$ apresenta a diferença de PSNR entre os dois casos. A tabela mostra que o ganho de qualidade é maior para vídeos comprimidos com HEVC do que para vídeos codificados com outros codecs. Além disso, o maior $\triangle$PSNR (0,719 dB) ocorre para vídeos comprimidos com HEVC e QP 37. Isso pode ser explicado porque esta é exatamente a mesma configuração usada para comprimir todos os vídeos usados no treinamento do modelo STDF proposto em (DENG et al., 2020).

Também podem ser observados valores negativos na Tabela 1 para algumas configurações de codificação, o que significa que o STDF acaba por diminuir a qualidade dos vídeos comprimidos em alguns casos. Quando isolamos a variável parâmetro de quantização, alterando apenas o codec, percebe-se uma diferença significativa em $\triangle$PSNR (por exemplo, HEVC e VVC com QP 32). Por outro lado, quando isolamos a variável codec, percebe-se que os resultados são sempre melhores para parâmetros de quantização mais elevados, independentemente do codec utilizado.

Assim, a análise revela que existe uma correlação entre a configuração (codec e parâmetro de quantização) utilizada para gerar as sequências de treinamento do STDF e a melhoria de qualidade alcançada. Isso mostra que, apesar do conjunto de

Tabela 1 – Melhoria de qualidade obtida com STDF (DENG et al., 2020) em diferentes cenários. Todos os valores são apresentados em PSNR (dB).

Codec	Quantização	PSNR (dB)		△PSNR
		Decodificado	Melhorado	
HEVC	QP 32	34,192	34,659	0,466
	QP 37	31,608	32,327	0,718
VVC	QP 32	34,713	34,607	-0,106
	QP 37	32,186	32,457	0,271
VP9	CQ 43	36,271	35,987	-0,230
	CQ 55	33,696	34,148	0,452
AV1	CQ 43	37,701	36,431	-1,270
	CQ 55	35,227	34,867	-0,360

tipos de artefatos serem similares em codificadores de vídeo híbridos, cada combinação de codec e parâmetro de quantização produz artefatos em intensidades e padrões diferentes. Dessa forma, um modelo treinado com vídeos gerados por uma determinada configuração pode não servir como bom método de pós-processamento para outros, sendo possivelmente necessário um modelo para cada configuração.

5.2 Soluções STDF Multi-Codec

Esta seção propõe um novo modelo baseado em STDF com base na hipótese de que um modelo para VQE treinado com vídeos comprimidos por diferentes codecs é capaz de atingir melhorias mais significativas de qualidade em diferentes cenários. Para comprovar a hipótese, o modelo STDF foi treinado com vídeos comprimidos conforme os padrões VVC e AV1, por serem esses os codecs mais recentes de suas respectivas famílias. Por serem pertencentes a famílias diferentes, os codificadores diferem entre si em termos de ferramentas de codificação, parâmetros e, potencialmente, na intensidade dos artefatos de compressão produzidos.

O novo modelo foi proposto em duas versões, seguindo metodologias similares. A primeira, foi treinada do zero, ou seja, os pesos do modelo foram inicializados aleatoriamente e o ajuste do modelo foi feito a partir daí. A segunda versão é uma alternativa que realiza o treinamento por *fine tunning*, a partir do modelo pré-treinado disponibilizado por Deng et al. (2020), ou seja, os pesos do modelo foram inicializados com os pesos de um modelo já treinado. A ideia é que o modelo partisse de características já aprendidas anteriormente, o que possivelmente levaria à uma maior eficiência do novo modelo.

O modelo multi-codec sem *fine tunning* apresentado nesta seção foi publicado no artigo Kreisler et al. (2024).

Para o processo de treinamento foi utilizado o dataset MFQEv2[2], pegando no má-

[2]https://github.com/ryanxingql/mfqev2.0/wiki/MFQEv2-Dataset

ximo 300 quadros de cada vídeo. O treinamento do modelo STDF multi-codec começa com a divisão do *dataset* de treino em duas partes iguais: 54 vídeos comprimidos usando VVC com QP 37 e 54 vídeos comprimidos usando AV1 com CQ 55. Foram escolhidos esses QPs/CQs por causarem um maior nível de degradação nos quadros. O *VVC Test Model* (VTM) (versão 13.0) foi utilizado como software de referência para todas as codificações de VVC, seguindo a configuração temporal *Low Delay P*. Por outro lado, o software de referência libaom (*hashcode* 3.3) foi utilizado para todas as codificações AV1. A divisão dos vídeos foi feita de acordo a resolução, como mostrado na Tabela 2. No entanto, como algumas resoluções possuem um número ímpar de vídeos, um vídeo extra foi admitido para um dos padrões (VVC QP 37) e a diferença foi compensada para o outro padrão (AV1 CQ 55) na próxima resolução que também possui um número ímpar de vídeos.

Tabela 2 – Divisão dos vídeos do *dataset* utilizado.

Resolução	Quantidade	
	VVC QP 37	AV1 CQ 55
2048×1080	3	3
1920×1080	18	18
704×576	3	2
720×480	2	2
640×360	18	19
352×288	9	9
352×240	1	1
Total	54	54

A partir dessa divisão, foi gerado o *dataset* multi-codec, utilizado para treinamento do novo modelo STDF (DENG et al., 2020). Durante o processo de treinamento, a rede foi alimentada com uma sequência de *patches* dos quadros degradados de tamanho 128x128. Após gerada a versão melhorada do *patch* do quadro, a versão original foi utilizada como entrada para o cálculo da função de perda *Charbonnier Loss*, apresentada na seção 3.4, onde ϵ foi fixado em 1^{-6}. Dessa forma, para o cálculo da função de perda, emprega-se o quadro de referência não comprimido e o quadro comprimido pelo VVC ou AV1 após o processo de filtragem. O objetivo do treinamento é fazer com que o modelo aprenda a mapear os quadros degradados pelo processo de compressão de volta para os quadros originais de alta qualidade.

Tanto para o modelo fine tunned quanto para o modelo treinado do zero o processo foi repetido por 10 épocas, ajustando os pesos do modelo para reduzir a diferença entre os quadros gerados após a melhoria de qualidade e os quadros originais de alta qualidade. Foi utilizado um lote de tamanho 32, o otimizador Adam com $\beta_1 = 0.9$, $\beta_2 = 0.999$ e $\epsilon = 10^{-8}$ e taxa de aprendizagem de 0,0001.

Os dois modelos treinados (sem e com *fine tunning*) foram ambos avaliados usando o conjunto de dados de teste, que foi comprimido utilizando os seguintes pa-

drões/formatos e configuração de quantização: HEVC com QP 32 e 37; VVC com QP 32 e 37; VP9 com CQ 43 e 55; AV1 com CQ 43 e 55. Dessa forma o conjunto de teste foi aumentado oito vezes, uma vez para cada par padrão-quantização, formando um total de 144 vídeos de teste.

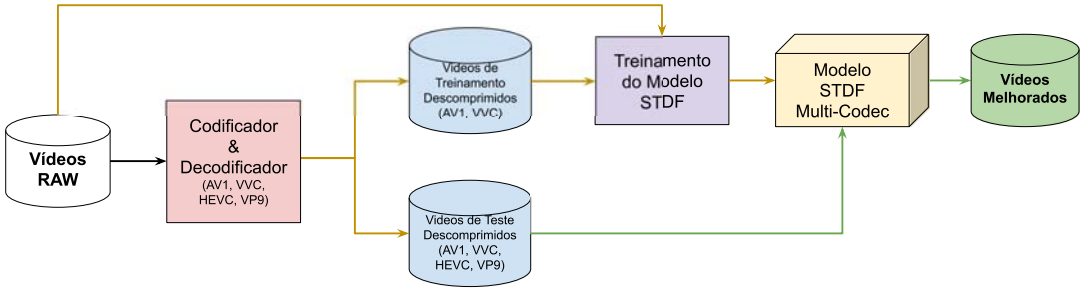

Figura 22 – Metodologia de treinamento e teste do modelo STDF multi-codec.

Na Figura 22 o processo de treinamento é representado pelas setas amarelas, enquanto a etapa de teste é representada pelo caminho com setas verdes. Dessa forma, os vídeos RAW são codificados e depois decodificados, gerando as sequências degradadas, ou seja, que possuem artefatos de compressão. Os vídeos de treinamento, então, passam pela rede e ao final a sequência RAW é dada como entrada para o cálculo da função de perda e respectiva retro-propagação do erro para o ajuste dos parâmetros da rede. Em posse do modelo treinado é iniciada a etapa de teste, gerando as sequências melhoradas. O treinamento dos dois modelos multi-codec foi realizado usando a implementação de referência do STDF (XING; DENG, 2020), utilizando os mesmos hiper-parâmetros e um raio igual a três. Vale destacar que, como os dois módulos do STDF são baseados em CNN, a arquitetura é unificada e é treinada de ponta a ponta.

Os modelos treinados utilizam a mesma arquitetura apresentada em 4.2, só diferenciando os dados de entrada para o treinamento.

5.3 Solução STDF Multi-Domínio

Esta seção propõe uma arquitetura STDF alternativa, de forma a capacitar o modelo treinado a perceber as diversas nuances de artefatos de compressão produzidos por cada um dos diferentes codificadores usados nesta pesquisa. Para isso, foi utilizada a abordagem de aprendizado multi-domínio, apresentada na seção 3.13. Dessa forma, a rede de QE foi quadruplicada de forma a possuir uma sub-rede para cada par padrão-quantização, potencialmente capturando as características necessárias para suavizar os artefatos produzidos por cada codec. Devido a limitações de *hardware* foram construídos dois modelos. Um modelo para o QP 32 e CQ 43; e outro modelo para o QP 37 e CQ 55.

5.3.1 Construção do Dataset

O *dataset* utilizado na solução multi-domínio é composto pelos vídeos do *dataset* MFQEv2 comprimidos nos padrões/formatos HEVC, VVC, VP9 e AV1. Foi utilizado os QPs 32 e 37 para vídeos codificados com os padrões HEVC e VVC e os CQs 43 e 55 para vídeos codificados com os formatos VP9 e AV1. Dessa forma, o *dataset* é composto por 864 vídeos divididos igualmente para cada par padrão-configuração de quantização. Para o processo de compressão dos vídeos, foram utilizados os softwares de referência de cada padrão/formato. Para o HEVC foi utilizado o HM, na versão 16.5, com a configuração *Low Delay P*. Para o VVC foi utilizado o VTM na versão 13.0 com a configuração *Low Delay P*. Para o VP9 foi utilizado o libvpx, hashcode 1.12.0 e para o AV1 foi utilizado o libaom, hashcode 3.3.

5.3.2 Arquitetura Multi-Domínio

A arquitetura original do STDF foi modificada de forma a incluir uma sub-rede de QE para cada par padrão-quantização, onde cada par é considerado um domínio específico. Dessa forma cada sub-rede de QE é responsável por capturar as nuances específicas de cada domínio. A sub-rede de alinhamento e fusão foi compartilhada entre todos os domínios já que se espera que o codec e o nível de quantização utilizados impactem menos nas funções do módulo de alinhamento e fusão do que no módulo de QE. A Figura 23 exemplifica a arquitetura adotada em uma visão de alto nível.

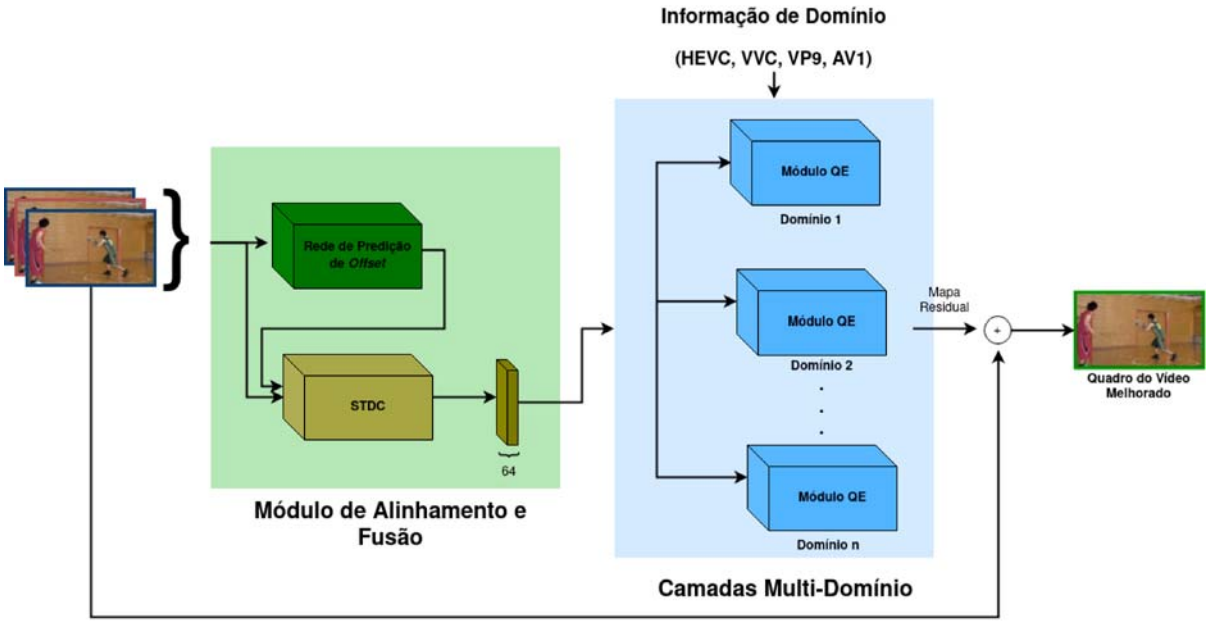

Figura 23 – Arquitetura STDF multi-domínio proposta.

5.3.3 Treinamento

O treinamento foi configurado para ocorrer por 10 épocas e para isso o tamanho do lote de treinamento foi configurado para 32 e a quantidade de iterações foi fixado em 1.200.000. Cada iteração do treinamento é executada sobre um domínio, ou seja, cada lote pertence a um único domínio, ativando apenas um ramo da rede. Dessa forma, a retro-propagação flui por apenas um ramo, sem afetar os outros e passa pelo módulo de alinhamento e fusão que é ajustado em todas as iterações. A forma de reconhecimento sobre qual domínio o lote pertence foi feito através de um *label* externo, como ilustra a Figura 23. Como função de perda, foi mantida a Charbonnier utilizada no treinamento do modelo multi-codec.

A cada iteração, o modelo é atualizado baseado apenas no lote que está sendo processado. Na iteração seguinte, é utilizado um lote de outro domínio, de forma que o lote de um mesmo domínio não seja processado duas vezes na sequência. Este processo é repetido até que o número pré-definido de iterações seja alcançado.

O modelo multi-domínio treinado possui um total de 823.009 parâmetros treináveis, uma diferença de 458.009 parâmetros em relação ao modelo de um único domínio, sendo esse valor referente ao acréscimo de mais três domínios.

6 RESULTADOS EXPERIMENTAIS

Neste capítulo serão apresentados e discutidos os resultados obtidos para cada uma das soluções apresentadas no capítulo anterior. A seção 6.1 apresenta os resultados para as soluções multi-codec e a seção 6.2 apresenta os resultados para a solução multi-domain.

6.1 Qualidade Objetiva para as Soluções STDF Multi-Codec

As Tabelas 3 e 5 apresentam os resultados de VQE em $\triangle$PSNR para os modelos propostos STDF multi-codec e multi-codec *fine tunned*, considerando as 18 sequências de vídeos de teste. As tabelas mostram os resultados obtidos para as 8 versões de vídeos de teste, ou seja, considerando cada par padrão-quantização mencionado no capítulo anterior. Os vídeos de teste são agrupados pela dimensão de acordo com suas classes (BOYCE; SUEHRING; LI, 2018): Classe A (2560x1600), Classe B (1920x1080), Classe C (832x480), Classe D (416x240), Classe E (1280x720). Os resultados são apresentados em PSNR, já que é a métrica mais comum para cálculo de qualidade objetiva utilizada nos trabalhos relacionados.

Como é possível observar na Tabela 3, que apresenta os resultados para o modelo multi-codec treinado do zero, a grande maioria dos resultados por sequência de vídeo é positiva, apenas alguns casos são negativos. Desses, a maioria é próximo de zero, ou seja, praticamente não representa mudança na qualidade objetiva do vídeo. O pior dos casos é de -0,127 dB para a sequência *BQTerrace* codificada com AV1 e utilizando CQ 43 e o melhor caso é de 0,803 dB para a sequência *BQSquare* codificada com o HEVC e utilizando QP 32 . A última linha da tabela apresenta valores médios calculados a partir do $\triangle$PSNR de cada sequência. É possível observar que o modelo multi-codec treinado alcança resultados médios positivos em todos os cenários, diferente dos testes apresentados anteriormente na Tabela 1 com o modelo STDF original, que apresentou uma degradação de qualidade (valores negativos) na metade dos casos. Na média, o modelo multi-codec proposto foi capaz de aumentar a qualidade objetiva dos vídeos entre 0,091 dB (AV1 CQ 43) e 0,382 dB (HEVC QP

32). Esses resultados médios mostram que o modelo possui uma boa capacidade de generalização para a melhoria de qualidade de vídeos codificados por diversos codecs.

Com o objetivo de comparar os resultados obtidos com o modelo STDF multi-codec, três modelos STDF single-codec também foram treinados. O primeiro foi treinado com um *dataset* composto por vídeos comprimidos apenas pelo HEVC; o segundo, com um *dataset* comprimido apenas com VVC; o terceiro, com um *dataset* comprimido apenas com AV1. Os softwares e configurações são as mesmas mencionadas na seção 5.3.1. O processo de treinamento foi o mesmo utilizado em Deng et al. (2020). Porém, ao invés de gerar um modelo apenas com vídeos codificados pelo HEVC utilizando QP 37, foram gerados modelos com vídeos comprimidos pelo VCC utilizando QP 37 e AV1 utilizando CQ 55.

Tabela 3 – Resultados do modelo STDF Multi-codec para vídeos comprimidos por diferentes codecs.

Dataset de Teste		ΔPSNR (dB)							
		HEVC		VVC		VP9		AV1	
		QP 32	QP 37	QP 32	QP 37	CQ 43	CQ 55	CQ 43	CQ 55
Classe A	*Traffic*	0,314	0,291	0,193	0,190	0,277	0,358	-0,012	0,166
	People on Street	0,370	0,368	0,130	0,149	0,340	0,391	0,055	0,190
Classe B	*Kimono*	0,249	0,193	0,114	0,112	0,217	0,195	0,093	0,120
	ParkScene	0,150	0,105	0,115	0,078	0,183	0,164	0,123	0,155
	Cactus	0,244	0,239	0,123	0,163	0,199	0,230	0,011	0,095
	BQTerrace	0,140	0,198	-0,009	0,052	0,074	0,192	-0,127	0,032
	BasketballDrive	0,313	0,316	0,088	0,152	0,261	0,317	-0,001	0,160
Classe C	*RaceHorses*	0,254	0,246	0,084	0,113	0,244	0,283	0,660	0,176
	BQMall	0,388	0,342	0,211	0,248	0,314	0,375	0,013	0,221
	PartyScene	0,492	0,372	0,265	0,258	0,436	0,428	0,203	0,279
	BasketballDrill	0,447	0,443	0,008	0,149	0,424	0,442	0,031	0,234
Classe D	*RaceHorses*	0,348	0,298	0,177	0,175	0,352	0,328	0,220	0,261
	BQSquare	0,803	0,643	0,431	0,375	0,752	0,775	0,571	0,689
	BlowingBubbles	0,460	0,352	0,328	0,316	0,403	0,377	0,246	0,311
	BasketballPass	0,573	0,463	0,380	0,392	0,549	0,515	0,257	0,430
Classe E	*FourPeople*	0,514	0,431	0,316	0,342	0,450	0,514	-0,064	0,209
	Johnny	0,398	0,349	0,221	0,256	0,352	0,410	0,032	0,210
	KristenAndSara	0,428	0,384	0,226	0,260	0,345	0,461	-0,075	0,153
Média		**0,382**	**0,335**	**0,189**	**0,210**	**0,343**	**0,375**	**0,091**	**0,229**

A Tabela 4 apresenta nas primeiras três linhas os resultados médios obtidos com os modelos STDF single-codec. Na última linha, os resultados médios apresentados na Tabela 3 são apresentados novamente para comparação. Na maioria dos modelos single-codec é possível notar resultados negativos, indicando que o modelo não é efetivo para todas as oito combinações de padrão-quantização testadas. O único modelo single-codec que apresentou resultados positivos em todos os casos de teste foi aquele treinado com vídeos codificados pelo AV1. Porém, na média, o modelo multi-codec proposto também apresenta resultados superiores ao modelo single-codec AV1.

A Tabela 5 apresenta os resultados para o modelo multi-codec *fine tunned*. Nela, é possível observar que o melhor resultado médio em ΔPSNR foi de 0,660 dB, para

Tabela 4 – Comparação de resultados de VQE entre os modelos STDF multi- e single-codec, incluindo (DENG et al., 2020).

Modelo STDF	ΔPSNR (dB)							
	HEVC		VVC		VP9		AV1	
	QP 32	QP 37	QP 32	QP 37	CQ 43	CQ 55	CQ 43	CQ 55
HEVC QP 37 (DENG et al., 2020)	0,362	0,755	-0,217	0,250	-0,465	0,357	-1,479	-0,506
VVC QP 37	0,446	0,529	0,216	0,371	0,050	0,385	-0,530	-0,016
AV1 CQ 55	0,346	0,285	0,137	0,144	0,368	0,389	0,109	0,286
Multi-Codec	**0,382**	**0,335**	**0,189**	**0,210**	**0,343**	**0,375**	**0,091**	**0,229**

HEVC QP 37. Já o pior resultado médio em ΔPSNR foi de -0,254 dB, para AV1 CQ 43. Por sequência de vídeo, o melhor resultado foi de 1,089 dB para a sequência *BQSquare* codificada com HEVC QP 32; já o pior resultado foi de -0,459 dB para a sequência *People on Street* codificada com AV1 CQ 43.

Tabela 5 – Resultados em ΔPSNR do modelo STDF Multi-Codec *Fine Tunned* para vídeos comprimidos por vários codecs

Dataset de Teste		ΔPSNR (dB)							
		HEVC		VVC		VP9		AV1	
		QP 32	QP 37	QP 32	QP 37	CQ 43	CQ 55	CQ 43	CQ 55
Classe A	*Traffic*	0,607	0,588	0,350	0,398	0,406	0,651	-0,254	0,148
	People on Street	0,699	1,007	0,227	0,574	0,295	0,895	-0,459	0,263
Classe B	*Kimono*	0,652	0,658	0,391	0,512	0,246	0,443	-0,153	0,139
	ParkScene	0,562	0,483	0,416	0,397	0,352	0,432	-0,118	0,151
	Cactus	0,551	0,583	0,286	0,394	0,409	0,554	-0,135	0,142
	BQTerrace	0,390	0,477	0,106	0,218	0,100	0,335	-0,268	0,002
	BasketballDrive	0,498	0,581	0,142	0,297	0,328	0,526	-0,125	0,185
Classe C	*RaceHorses*	0,327	0,447	0,058	0,201	0,167	0,434	-0,329	0,089
	BQMall	0,700	0,669	0,398	0,492	0,454	0,698	-0,254	0,238
	PartyScene	0,688	0,588	0,373	0,397	0,334	0,592	-0,352	0,186
	BasketballDrill	0,548	0,638	0,063	0,324	0,403	0,672	-0,504	0,135
Classe D	*RaceHorses*	0,577	0,656	0,312	0,399	0,431	0,623	-0,230	0,252
	BQSquare	1,089	0,923	0,608	0,626	0,317	0,789	-0,438	0,278
	BlowingBubbles	0,702	0,552	0,460	0,434	0,445	0,572	-0,184	0,244
	BasketballPass	0,872	0,836	0,564	0,668	0,557	0,793	-0,153	0,446
Classe E	*FourPeople*	0,852	0,767	0,474	0,535	0,663	0,902	-0,208	0,251
	Johnny	0,662	0,652	0,278	0,396	0,517	0,716	-0,178	0,147
	KristenAndSara	0,816	0,783	0,410	0,498	0,553	0,829	-0,207	0,191
Média		**0,655**	**0,660**	**0,329**	**0,431**	**0,388**	**0,636**	**-0,253**	**0,194**

A Tabela 6 apresenta a comparação dos resultados médios de todos os modelos apresentados até aqui. Na média das médias, o modelo multi-codec *fine tunned* se sai melhor que o modelo multi-codec, mesmo apresentando um caso de média negativa (AV1 CQ 43). Na média das médias, o modelo multi-codec *fine tunned* apresenta um acréscimo de PSNR de 0,380 dB, contra 0,270 dB do modelo multi-codec. Porém, o modelo *fine tunned* apresenta uma capacidade de generalização pior que o modelo multi-codec treinado do zero, visto que não é capaz de atingir resultados positivos em todos os casos.

Tabela 6 – Comparação dos resultados de VQE entre os modelos STDF single-codec, multi-codec e multi-codec *fine tunned*

Modelo STDF	ΔPSNR (dB)							
	HEVC		VVC		VP9		AV1	
	QP 32	QP 37	QP 32	QP 37	CQ 43	CQ 55	CQ 43	CQ 55
HEVC QP 37 (DENG et al., 2020)	0,362	0,755	-0,217	0,250	-0,465	0,357	-1,479	-0,506
VVC QP 37	0,446	0,529	0,216	0,371	0,050	0,385	-0,530	-0,016
AV1 CQ 55	0,346	0,285	0,137	0,144	0,368	0,389	0,109	0,286
Multi-Codec	0,382	0,335	0,189	0,210	0,343	0,375	0,091	0,229
Multi-Codec Fine Tunned	**0,655**	**0,660**	**0,329**	**0,431**	**0,388**	**0,636**	**-0,253**	**0,194**

6.2 Qualidade Objetiva para a Solução STDF Multi-Domínio

As Tabelas 7 e 8 apresentam os resultados de variação de qualidade objetiva para cada sequência de vídeo. Onde a Tabela 7 apresenta os resultados em termos de ΔPSNR e a Tabela 8 apresenta os resultados em termos de ΔSSIM para cada sequência de vídeo e a última linha de cada tabela apresenta a média dos resultados. Os vídeos estão agrupados de acordo com suas classes (BOYCE; SUEHRING; LI, 2018): Classe A (2560x1600), Classe B (1920x10809), Classe C (832x480), Classe D (416x240), Classe E (1280x720). Devido aos bons resultados apresentados em termo de ΔPSNR, decidiu-se adicionar a métrica SSIM para complementar a análise dos resultados.

Como é possível observar nas Tabelas, a maioria dos resultados são positivos, onde apenas um caso específico apresentou um valor de ΔPSNR negativo, mas em termos de ΔSSIM o resultado permaneceu positivo. Na média é possível observar que todos os resultados foram positivos. O pior resultado atingido, em ΔPSNR, foi de -0,024 dB para AV1 CQ 55 na sequência *BQTerrace*, porém em termos de ΔSSIM o vídeo apresentou uma pequena melhora. Este também é o pior resultado em termos de ΔSSIM. O melhor resultado em ΔPSNR foi de 1,437 dB para HEVC Qp 32 na sequência *BQSquare*; em ΔSSIM, foi de 0,020 para HEVC QP 37 na sequência *People on Street* e *BlowingBubbles*. Em resultados médios de ΔPSNR, a pior melhoria atingida foi de 0,228 dB para os vídeos codificados com AV1 CQ 55 e a melhor melhoria atingida foi de 0,787 dB para os vídeos codificados com HEVC QP 32. Em termos de ΔSSIM médio, a pior caso foi de 0,004 para vídeos codificados com AV1, tanto para o CQ 43 quanto para o CQ 55, e o melhor caso foi de 0,014 para vídeos codificados com HEVC QP 37. Alguns casos específicos, além do melhor caso, apresentaram resultados em termos de ΔPSNR acima de 1 dB, como é o caso da sequência *BasketballPass*d codificado pelo HEVC com QP 32, que apresentou um resultado de 1,013 dB; a sequência *People on Street* codificado pelo HEVC com QP 37, que apresentou um resultado de 1,126 dB; a sequência *BQSquare* codificado pelo HEVC com QP 37 apresentou um resultado de 1,020 dB; a sequência *BQSquare* codificado pelo VP9 CQ 43 apresentou um resultado de 1,379 dB; a sequência *BQSquare* codificado pelo VP9 CQ 55 apresentando um resultado de 1,234 dB; a sequência *FourPeople* codifi-

cado pelo VP9 com CQ 55 apresentando um resultado de 1,046 dB; e a sequência *BQSquare* codificado com o AV1 CQ 43 apresentando um resultado de 1,147 dB.

Tabela 7 – Resultados de VQE para o modelo multi-domínio em termos de ΔPSNR.

Dataset de Teste		ΔPSNR (dB)							
		HEVC		VVC		VP9		AV1	
		QP 32	QP 37	QP 32	QP 37	CQ 43	CQ 55	CQ 43	CQ 55
Classe A	*Traffic*	0,730	0,662	0,480	0,420	0,697	0,727	0,290	0,120
	People on Street	0,919	1,126	0,465	0,582	0,741	0,911	0,298	0,200
Classe B	*Kimono*	0,777	0,831	0,499	0,547	0,464	0,462	0,237	0,184
	ParkScene	0,628	0,551	0,509	0,426	0,530	0,487	0,255	0,152
	Cactus	0,642	0,698	0,391	0,404	0,554	0,641	0,168	0,130
	BQTerrace	0,498	0,543	0,244	0,217	0,333	0,402	0,102	-0,024
	BasketballDrive	0,586	0,686	0,266	0,292	0,488	0,552	0,206	0,156
Classe C	*RaceHorses*	0,456	0,447	0,263	0,220	0,486	0,473	0,228	0,153
	BQMall	0,880	0,838	0,587	0,529	0,807	0,828	0,369	0,231
	PartyScene	0,854	0,640	0,579	0,380	0,857	0,731	0,557	0,197
	BasketballDrill	0,622	0,718	0,241	0,290	0,753	0,733	0,423	0,197
Classe D	*RaceHorses*	0,698	0,691	0,491	0,436	0,731	0,661	0,420	0,344
	BQSquare	1,437	1,020	0,966	0,667	1,379	1,234	1,147	0,488
	BlowingBubbles	0,834	0,635	0,622	0,453	0,833	0,702	0,475	0,328
	BasketballPass	1,013	0,971	0,763	0,716	0,947	0,860	0,585	0,496
Classe E	*FourPeople*	0,936	0,913	0,590	0,548	0,967	1,046	0,356	0,325
	Johnny	0,757	0,804	0,406	0,414	0,757	0,843	0,222	0,164
	KristenAndSara	0,905	0,969	0,508	0,515	0,829	0,954	0,299	0,270
Média		**0,787**	**0,764**	**0,493**	**0,448**	**0,731**	**0,736**	**0,369**	**0,228**

Tabela 8 – Resultados de VQE para o modelo multi-domínio em termos de ΔSSIM.

Dataset de Teste		ΔSSIM							
		HEVC		VVC		VP9		AV1	
		QP 32	QP 37	QP 32	QP 37	CQ 43	CQ 55	CQ 43	CQ 55
Classe A	*Traffic*	0,008	0,011	0,004	0,006	0,007	0,009	0,002	0,003
	People on Street	0,011	0,020	0,006	0,009	0,009	0,014	0,003	0,004
Classe B	*Kimono*	0,012	0,016	0,006	0,008	0,005	0,007	0,002	0,004
	ParkScene	0,012	0,013	0,009	0,011	0,007	0,008	0,003	0,005
	Cactus	0,009	0,013	0,006	0,007	0,007	0,010	0,003	0,003
	BQTerrace	0,007	0,009	0,003	0,005	0,005	0,006	0,001	0,001
	BasketballDrive	0,008	0,012	0,004	0,005	0,006	0,008	0,003	0,004
Classe C	*RaceHorses*	0,009	0,011	0,005	0,005	0,008	0,011	0,003	0,003
	BQMall	0,012	0,017	0,007	0,009	0,009	0,012	0,003	0,004
	PartyScene	0,015	0,019	0,010	0,011	0,009	0,014	0,005	0,005
	BasketballDrill	0,007	0,015	0,003	0,005	0,012	0,014	0,005	0,004
Classe D	*RaceHorses*	0,013	0,018	0,011	0,011	0,012	0,015	0,005	0,008
	BQSquare	0,012	0,015	0,008	0,009	0,009	0,010	0,008	0,004
	BlowingBubbles	0,017	0,020	0,013	0,015	0,012	0,015	0,006	0,008
	BasketballPass	0,015	0,019	0,011	0,015	0,011	0,015	0,006	0,008
Classe E	*FourPeople*	0,007	0,012	0,004	0,006	0,006	0,008	0,002	0,002
	Johnny	0,006	0,008	0,003	0,003	0,004	0,006	0,001	0,001
	KristenAndSara	0,006	0,009	0,003	0,004	0,005	0,006	0,002	0,002
Média		**0,010**	**0,014**	**0,006**	**0,008**	**0,008**	**0,010**	**0,004**	**0,004**

A Tabela 9 replica a Tabela 6, incluindo o resultado médio em ΔPSNR apresentado na Tabela 7, por conveniência para comparação. Como é possível observar, o modelo multi-domínio apresenta resultado superior ao modelo multi-codec em quase todos os testes, ultrapassando inclusive os resultados dos modelos single-codec testados com

vídeos codificados pelo mesmo padrão usado para gerar os vídeos de treinamento. Este é o caso do HEVC QP 37, que obteve um △PSNR de 0,755 dB para os vídeos de teste codificados com o mesmo padrão, contra 0,764 dB do multi-domínio. O mesmo comportamento se repete para os *datasets* do VVC, VP9 e AV1 CQ 43, este comportamento apenas não ocorre no caso do AV1 CQ 55. Isso mostra que o modelo multi-domínio é capaz de generalizar tão bem quanto o modelo multi-codec, mas sem perder a especificidade e o bom desempenho dos modelos single-codec.

Tabela 9 – Comparação entre todas as abordagens apresentadas.

Modelo STDF	△PSNR (dB)							
	HEVC		VVC		VP9		AV1	
	QP 32	QP 37	QP 32	QP 37	CQ 43	CQ 55	CQ 43	CQ 55
HEVC QP 37 (DENG et al., 2020)	0,362	0,755	-0,217	0,250	-0,465	0,357	-1,479	-0,506
VVC QP 37	0,446	0,529	0,216	0,371	0,050	0,385	-0,530	-0,016
AV1 CQ 55	0,346	0,285	0,137	0,144	0,368	0,389	0,109	0,286
Multi-Codec (KREISLER et al., 2023)	0,382	0,335	0,189	0,210	0,343	0,375	0,091	0,229
Multi-Codec Fine Tunned	0,655	0,660	0,329	0,431	0,388	0,636	-0,253	0,194
Multi-Domínio	**0,787**	**0,764**	**0,493**	**0,448**	**0,731**	**0,736**	**0,369**	**0,228**

6.3 Percepção de Qualidade Visual da Solução STDF Multi-Domínio

Esta seção apresenta uma análise da percepção de melhoria de qualidade visual da solução STDF multi-domínio. São apresentados três quadros de sequências diferentes. Foram selecionados aqueles quadros em que foi percebida a maior diferença em nível de qualidade visual após uma série de análises visuais.

Nas três figuras a seguir, uma para cada quadro, no topo está apresentado o quadro original (RAW). A primeira coluna apresenta um recorte do quadro original demarcado na imagem. A segunda coluna é a versão comprimida da mesma região demarcada e a terceira coluna é a versão melhorada pela solução STDF multi-domínio. Cada linha foi definida para cada codificador. Assim, a primeira linha apresenta a versão comprimida pelo HEVC, a segunda para o VVC, a terceira para o VP9 e a última para o AV1.

A Figura 24 apresenta uma composição a partir do quadro de número 14 da sequência *BasketballDrill*. Observando as imagens da segunda coluna, ou seja, as imagens que passaram pelo processo de compressão, a maioria apresenta uma grande quantidade de artefatos, sendo as mais danificadas as imagens (c) e (i), que correspondem, respectivamente, aos codecs HEVC e VP9. Na imagem (l), que corresponde ao codec AV1, ainda são perceptíveis alguns artefatos, porém em baixa quantidade. Já a imagem (f), correspondente ao codec VVC, é a que menos apresenta artefatos.

Em todas as imagens comprimidas é perceptível o efeito de borramento, principal-

mente no plano de fundo. Nas imagens (c), (i) e (l) é possível observar o efeito de bloco. Na imagem (c), este artefato forma um efeito escadaria nas bordas da bola. Na imagem (i), forma um efeito mosaico. Na imagem (l), também é perceptível o efeito escadaria na borda superior direita da bola.

Comparando as versões de baixa qualidade, da segunda coluna, com suas respectivas versões melhoradas, na terceira coluna, é possível observar que nas imagens (d) e (j) houve um grande impacto na melhoria da qualidade visual. A imagem (m) também apresenta uma suavização dos artefatos. Já a imagem (g) é a que mais se assemelha à sua versão comprimida antes da aplicação do filtro de melhoria, já que a imagem (f) apresenta uma baixíssima incidência de artefatos de compressão.

A Figura 25 apresenta uma composição a partir do quadro de número 31 da sequência Cactus. Em todas as imagens comprimidas é possível observar algum nível de artefatos de compressão. A princípio, é perceptível que todas as imagens são afetadas por um nível de borramento. Na imagem (c) é visível o efeito de bloco, ocasionando os efeitos escadaria e falsa borda. As imagens (f), (i) e (l) apresentam uma menor incidência de artefatos, porém nas bordas da sombra é levemente perceptível um efeito escadaria, também causado pelo artefato de bloco.

Comparando as figuras da segunda coluna com suas versões melhoradas, na terceira coluna, é possível perceber que houve uma melhora da qualidade visual das imagens comprimidas, atenuando os artefatos de bloco. A melhoria mais nítida foi da imagem (c) para a imagem (d), ou seja, para a versão comprimida pelo HEVC. Nas imagens (f), (i) e (l) também é perceptível a atenuação do efeito escadaria nas bordas da sombra. Porém, em todas as imagens melhoradas, ainda é visível um nível de borramento ocasionando uma perda de textura.

A Figura 26 apresenta uma composição a partir do quadro de número 5 da sequência *RaceHorses* na resolução 832×480. Analisando as imagens que passaram pelo processo de compressão, todas possuem algum nível de artefatos, porém as mais afetadas aparentam ser as imagens (c) e (f). Caraterizando os artefatos, é possível perceber um efeito de falsa borda nas imagens (c) e (f). A imagem (i) apresenta um leve efeito escadaria nas bordas da perna do cavalo. A imagem (l) é que menos aparenta ter artefatos de compressão, porém nela, como em todas as outras, apresenta um efeito de borramento.

Comparando as imagens que passaram pelo processo de compressão, segunda coluna, com suas versões melhoradas, terceira coluna, é possível observar que a maioria teve um impacto na melhoria da qualidade visual. Nas imagens (d) e (g) é notável a atenuação do efeito de falsa borda, apesar de algum efeito ainda ter sido mantido. Na imagem (d) também é possível perceber a remoção do efeito escadaria nas bordas do cavalo. Ainda assim, o efeito de borramento persistiu nessas imagens. Analisando as imagens (j) e (m), estas são as que mais se aproximam da versão

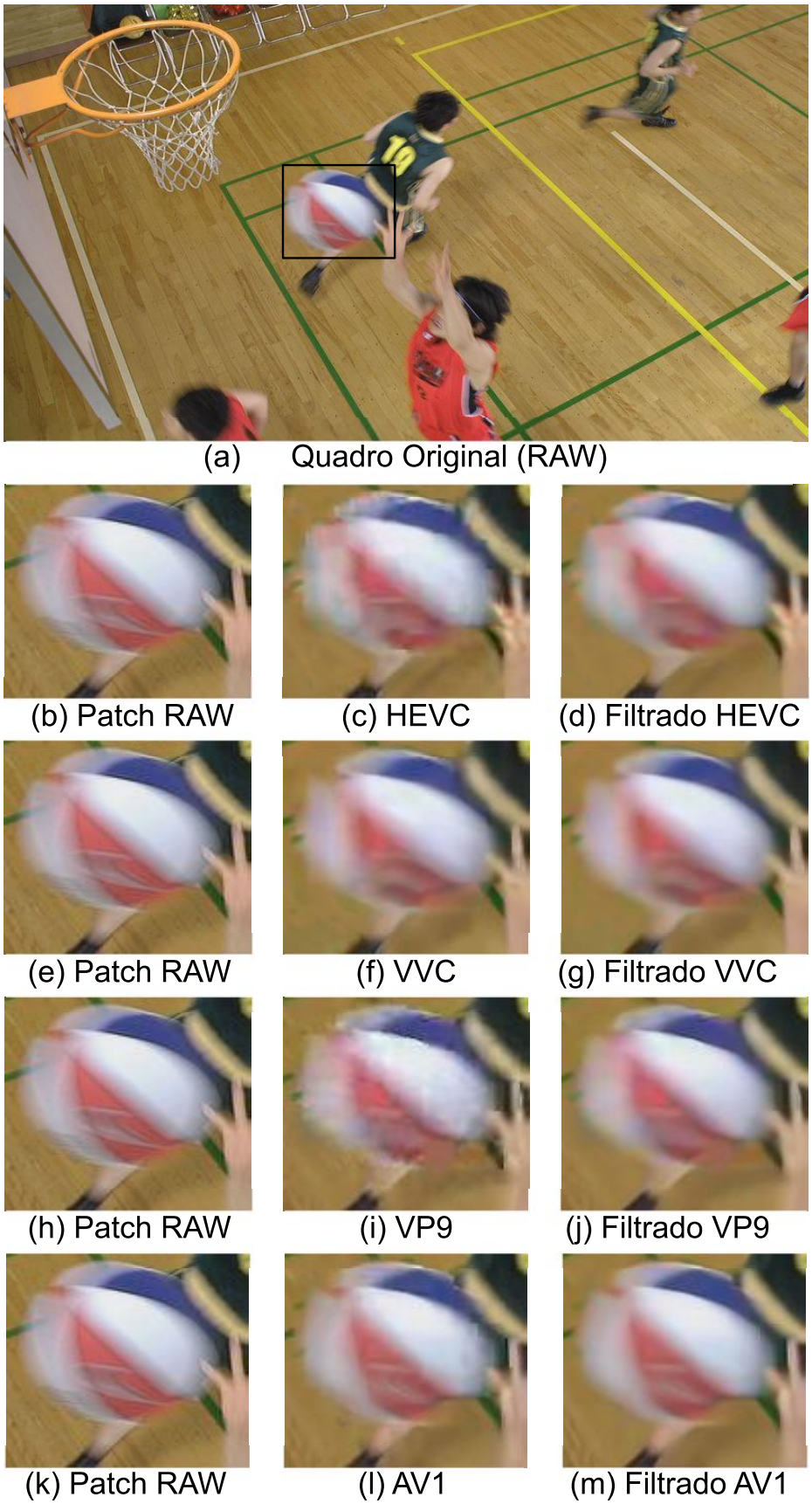

Figura 24 – Quadro de número 14 da sequência BasketballDrill: (a) Quadro original (RAW); (b)(e)(h)(k) Recorte do quadro original; (c) Versão comprimida pelo HEVC; (d) Versão do HEVC melhorada; (f) Versão comprimida pelo VVC; (g) Versão do VVC melhorada; (i) Versão comprimida pelo VP9; (j) Versão do VP9 melhorada; (l) Versão comprimida pelo AV1; (m) Versão do AV1 melhorada.

(a) Quadro Original (RAW)

(b) Patch RAW

(c) HEVC

(d) Filtrado HEVC

(e) Patch RAW

(f) VVC

(g) Filtrado VVC

(h) Patch RAW

(i) VP9

(j) Filtrado VP9

(k) Patch RAW

(l) AV1

(m) Filtrado AV1

Figura 25 – Quadro de número 31 da sequência Cactus: (a) Quadro original (RAW); (b)(e)(h)(k) Recorte do quadro original; (c) Versão comprimida pelo HEVC; (d) Versão do HEVC melhorada; (f) Versão comprimida pelo VVC; (g) Versão do VVC melhorada; (i) Versão comprimida pelo VP9; (j) Versão do VP9 melhorada; (l) Versão comprimida pelo AV1; (m) Versão do AV1 melhorada.

original, devido a conservação de parte dos detalhes da grama. Ainda assim, na imagem (m) ainda é notável um padrão de efeito de bloco. Portando a imagem (j) é a que aparenta um melhor resultado em relação a original.

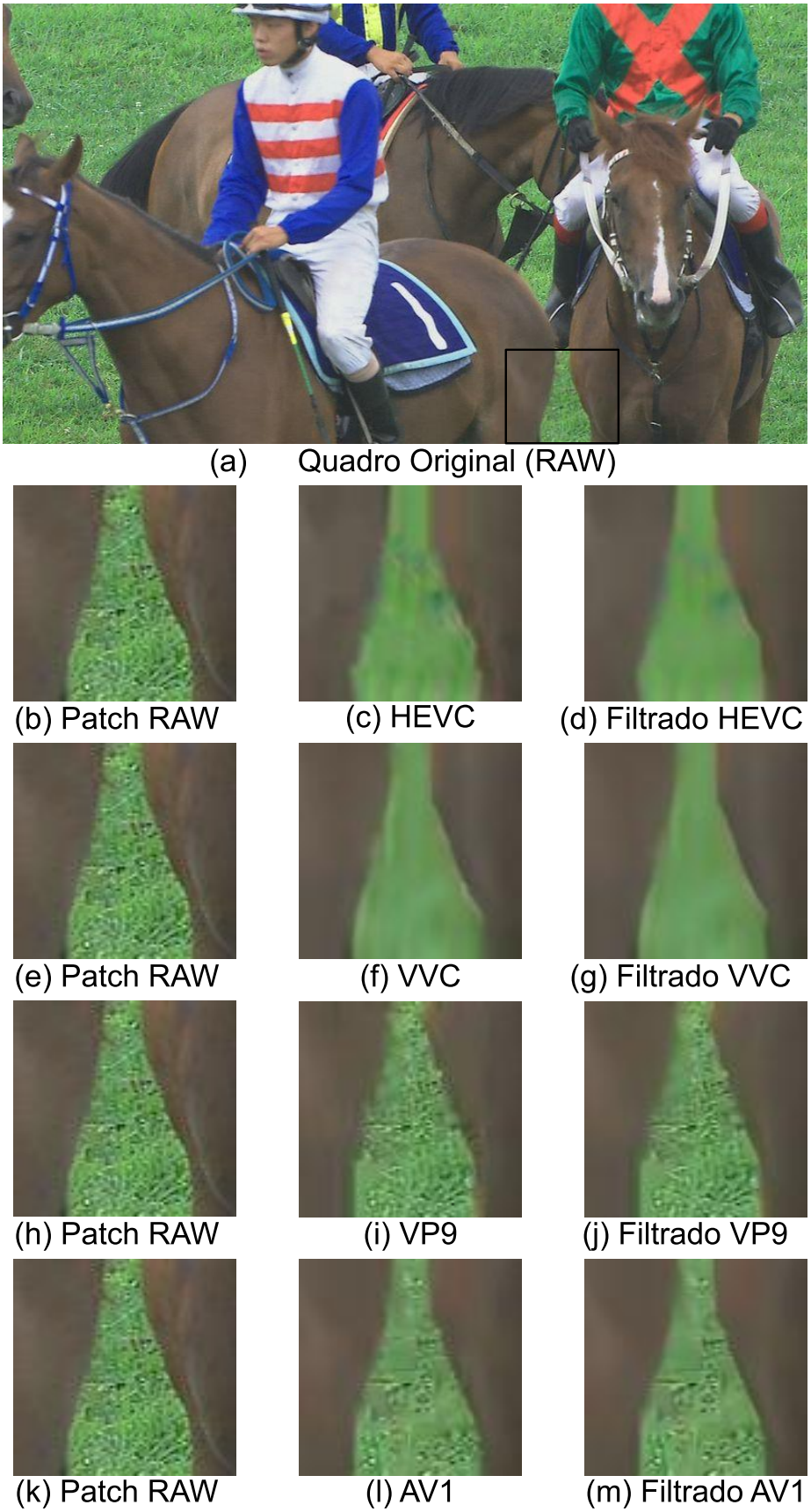

Figura 26 – Quadro de número 5 da sequência RaceHorses na resolução 832×480: (a) Quadro original (RAW); (b)(e)(h)(k) Recorte do quadro original; (c) Versão comprimida pelo HEVC; (d) Versão do HEVC melhorada; (f) Versão comprimida pelo VVC; (g) Versão do VVC melhorada; (i) Versão comprimida pelo VP9; (j) Versão do VP9 melhorada; (l) Versão comprimida pelo AV1; (m) Versão do AV1 melhorada.

7 CONCLUSÃO

O aumento da circulação de vídeos pela internet acentuou a necessidade por tecnologias de compressão de vídeos. Inevitavelmente essas tecnologias introduzem artefatos de compressão que degradam a qualidade do vídeo e reduzem a qualidade de experiência do usuário final. Para reduzir os artefatos de múltiplos codecs, os métodos de filtragem de pós-processamento são os mais indicados, pois não estão atrelados ao codec.

Com o advento das redes neurais e seu sucesso para tarefas de visão computacional, pesquisadores de melhoria de qualidade de imagem começaram a aplicar estas técnicas tirando proveito do contexto espacial. O sucesso destas fez com que pesquisadores de melhoria de qualidade de vídeo (*Video Quality Enhancement* – VQE) estendessem estas técnicas para o problema de VQE, adicionando o contexto temporal ao processo, com o intuito de explorar as flutuações de qualidade que ocorrem durante a compressão.

Em um primeiro momento as arquiteturas estado-da-arte se baseavam em *optical-flow*, porém esta técnica se mostrou sub-ótima já que ela causava inconsistências no quadro de saída. Para resolver este problema técnicas mais recentes aplicaram em suas arquiteturas as convoluções deformáveis, que obtiveram resultados superiores as que usam *optical-flow*, sendo o *Spatio-Temporal Deformable Fusion* a arquitetura estado-da-arte.

Na técnica estado-da-arte, apesar de ser um filtro de pós-processamento, os processos de treino e teste são executados apenas com o padrão HEVC. Dessa forma, sua eficácia não foi verificada com outros padrões/formatos, como o VVC, AV1 e VP9. A pesquisa apresentada nesta dissertação evidenciou que o modelo estado-da-arte não possui boa capacidade de generalização contra outros padrões/formatos além daquele para o qual foi treinado. Inclusive, modelos single-codec retreinados do zero utilizando outro padrão/formato não foram capazes de alcançar os mesmos níveis de melhoria de qualidade que o modelo treinado com vídeos comprimidos utilizando o padrão HEVC, o que torna necessária a proposta de um modelo que seja de fato genérico.

Para solucionar este problema, a exploração apresentada nesta dissertação propõe três soluções. Uma solução baseada em um *dataset* misto, construído com vídeos codificados pelo AV1 e VVC, por serem os mais recentes de suas famílias. Esta solução, apesar de não conseguir superar os modelos single-codec nos casos de teste que o codificador é o mesmo usado no treino, obteve uma capacidade de generalização constante, alcançando resultados de $\triangle$PSNR entre 0,091 dB e 0,382 dB.

A segunda solução seguiu a mesma estratégia da primeira abordagem, porém esta solução parte do modelo pré-treinado pelos autores do STDF. Era esperado que o processo de *fine tunning* gerasse um modelo melhor que o multi-codec, o que acabou acontecendo na grande maioria dos casos. Na média das médias, o modelo *fine tunned* obteve um resultado superior ao modelo treinado do zero. Porém, no caso AV1 CQ 43 o modelo obteve um resultado negativo, invalidando sua capacidade de generalização.

A terceira solução, chamada de multi-domínio, é baseada no paradigma de treinamento de mesmo nome, sendo utilizado como domínio os padrões/formatos HEVC, VVC, VP9 e AV1. Esta solução obteve resultados muito superiores à solução multi-codec, superando inclusive, na maioria dos casos, os modelos single-codec nos casos de teste em que o codec é o mesmo usado no treinamento. Isso pode ser explicado pela quantidade de amostras de treino ter sido maior, o que pode ter levado a um maior refinamento da rede de alinhamento e fusão, que é a rede compartilhada entre todos os domínios. Os resultados alcançados em $\triangle$PSNR estão entre 0,228 dB e 0,787 dB e em $\triangle$SSIM estão entre 0,004 e 0,014.

Os bons resultados da análise objetiva do modelo multi-domínio também se refletem na melhoria de qualidade perceptual. Como mostrado na análise de qualidade visual, o modelo multi-domínio foi capaz de remover satisfatoriamente os artefatos presentes nos quadros, suavizando as imagens. Alguns detalhes não podem ser restaurados devido à natureza do processo de compressão com perda, porém, no geral, é perceptível a melhoria de qualidade nos quadros.

Como trabalho futuro, pretende-se utilizar QPs/CQs extremos nos experimentos e analisar o desempenho do modelo atual, assim descobrindo até que ponto compensa comprimir um vídeo de forma que seja possível restaurá-lo a um nível de qualidade agradável ao observador humano. Também pretende-se trabalhar em redução de complexidade da rede neural voltada para VQE, já que atualmente para que estas redes neurais funcionem em tempo real é necessário o emprego de uma GPU.

REFERÊNCIAS

AGOSTINI, L. **Desenvolvimento de Arquiteturas de Alto Desempenho Dedicadas à Compressão de Vídeo Segundo o Padrão H.264/AVC**. 2007. 173p. Tese de Doutorado (Ciência da Computação) — Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

BOYCE, J.; SUEHRING, K.; LI, X. JVET-J1010: JVET common test conditions and software reference configurations. **JVET-J1010**, [S.l.], 2018.

BROSS, B. et al. Overview of the Versatile Video Coding (VVC) Standard and its Applications. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.31, n.10, p.3736–3764, 2021.

CABALLERO, J. et al. Real-Time Video Super-Resolution With Spatio-Temporal Networks and Motion Compensation. In: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR), 2017. **Proceedings...** [S.l.: s.n.], 2017.

CHEN, Y.; LU, R.; ZOU, Y.; ZHANG, Y. Branch-Activated Multi-Domain Convolutional Neural Network for Visual Tracking. **Journal of Shanghai Jiaotong University (Science)**, [S.l.], v.23, p.360–367, 2018.

CHIKKERUR, S.; SUNDARAM, V.; REISSLEIN, M.; KARAM, L. J. Objective video quality assessment methods: A classification, review, and performance comparison. **IEEE transactions on broadcasting**, [S.l.], v.57, n.2, p.165–182, 2011.

CISCO. **Cisco Annual Internet Report (2018–2023) White Paper**. Acessado em 15/02/2023, `https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html`.

CISCO, V. Cisco visual networking index: Forecast and trends, 2017–2022. **White Paper**, [S.l.], v.1, n.1, 2018.

CORREA, G. **Computational Complexity Reduction and Scaling for High Efficiency Video Encoders**. 2014. 286p. Tese de Doutorado (Faculdade de Ciências e Tecnologia) — Departamento de Engenharia Eletrotécnica e de Computadores, Universidade de Coimbra, Coimbra.

CYBENKO, G. Approximation by superpositions of a sigmoidal function. **Mathematics of Control, Signals, and Systems**, [S.l.], v.2, n.4, p.303–314, 1989.

DAI, J. et al. Deformable Convolutional Networks. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION (ICCV), 2017. **Proceedings...** [S.l.: s.n.], 2017.

DAI, Y.; LIU, D.; WU, F. A convolutional neural network approach for post-processing in HEVC intra coding. In: MULTIMEDIA MODELING: 23RD INTERNATIONAL CONFERENCE, MMM 2017, REYKJAVIK, ICELAND, JANUARY 4-6, 2017, PROCEEDINGS, PART I 23, 2017. **Anais...** [S.l.: s.n.], 2017. p.28–39.

DENG, J.; WANG, L.; PU, S.; ZHUO, C. Spatio-temporal deformable convolution for compressed video quality enhancement. In: AAAI CONFERENCE ON ARTIFICIAL INTELLIGENCE, 2020. **Proceedings...** [S.l.: s.n.], 2020. v.34, p.10696–10703.

DING, B.; QIAN, H.; ZHOU, J. Activation Functions and Their Characteristics in Deep Neural Networks. **IEEE transactions on image processing**, [S.l.], v.13, n.4, p.600–612, 2004.

DONG, C.; DENG, Y.; LOY, C. C.; TANG, X. Compression artifacts reduction by a deep convolutional network. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION, 2015. **Proceedings...** [S.l.: s.n.], 2015. p.576–584.

DUARTE, A. **Redução de Complexidade do Processo de Decisão de Modo da Predição Intra-Quadro do Codificador de Vídeo VVC utilizando Aprendizado de Máquina**. 2021. 107p. Dissertação de Mestrado (Ciência da Computação) — Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas.

DUMOULIN, V.; VISIN, F. **A guide to convolution arithmetic for deep learning**.

FERRUGEM, A. **Extração de Mapas de Profundidades de Dense Light Fields usando DeepLearning**. 2022. 143p. Tese de Doutorado (Ciência da Computação) — Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas.

GONZALES, R. C.; WOODS, R. E. **Processamento Digital de Imagens**. Av. Ermano Marchetti, 1435, SP, BR: Pearson, 2010. 644p.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep learning**. [S.l.]: MIT press, 2016.

GOODFELLOW, I. et al. Generative adversarial networks. **Commun. ACM**, New York, NY, USA, v.63, n.11, p.139–144, oct 2020.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. **The Elements of Statistical Learning**: Data Mining, Inference, and Prediction. 2nd.ed. New York: Springer, 2009.

HAYKIN, S. **Neural networks and learning machines**. 3.ed. [S.l.]: Pearson Education, 2009.

HORNIK, K.; STINCHCOMBE, M.; WHITE, H. Multilayer feedforward networks are universal approximators. **Neural Networks**, [S.l.], v.2, n.5, p.359–366, 1989.

III, H. C. R.; LIM, J. S. Reduction Of Blocking Effects In Image Coding. **Optical Engineering**, [S.l.], v.23, n.1, p.230134, 1984.

JIA, C. et al. Spatial-temporal residue network based in-loop filter for video coding. In: IEEE VISUAL COMMUNICATIONS AND IMAGE PROCESSING (VCIP), 2017., 2017. **Anais...** [S.l.: s.n.], 2017. p.1–4.

JIN, Z.; AN, P.; YANG, C.; SHEN, L. Quality enhancement for intra frame coding via cnns: An adversarial approach. In: IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING (ICASSP), 2018., 2018. **Anais...** [S.l.: s.n.], 2018. p.1368–1372.

KANG, J.; KIM, S.; LEE, K. M. Multi-modal/multi-scale convolutional neural network based in-loop filter design for next generation video codec. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 2017., 2017. **Anais...** [S.l.: s.n.], 2017. p.26–30.

KARCZEWICZ, M. et al. VVC In-Loop Filters. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.31, n.10, p.3907–3925, 2021.

KIM, J.; LEE, J. K.; LEE, K. M. Accurate image super-resolution using very deep convolutional networks. In: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 2016. **Proceedings...** [S.l.: s.n.], 2016. p.1646–1654.

KREISLER, G. et al. Modelo Multi-Codec Baseado em Spatio-Temporal Deformable Fusion para Melhoria de Qualidade de Vídeos Comprimidos. In: L SEMINÁRIO INTEGRADO DE SOFTWARE E HARDWARE, 2023. **Anais...** [S.l.: s.n.], 2023. p.143–154.

KREISLER, G. et al. Multi-Codec Video Quality Enhancement Model Based on Spatio-Temporal Deformable Fusion. In: IEEE 15TH LATIN AMERICA SYMPOSIUM ON CIRCUITS AND SYSTEMS (LASCAS), 2024., 2024. **Anais...** [S.l.: s.n.], 2024.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. ImageNet Classification with Deep Convolutional Neural Networks. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, 2012. **Anais...** Curran Associates: Inc., 2012. v.25.

KUANAR, S.; CONLY, C.; RAO, K. Deep learning based HEVC in-loop filtering for decoder quality enhancement. In: PICTURE CODING SYMPOSIUM (PCS), 2018., 2018. **Anais...** [S.l.: s.n.], 2018. p.164–168.

LI, Z. et al. A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. **IEEE Transactions on Neural Networks and Learning Systems**, [S.l.], v.33, n.12, p.6999–7019, 2022.

LIN, L.; YU, S.; ZHAO, T.; WANG, Z. PEA265: Perceptual Assessment of Video Compression Artifacts. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], p.3898 – 3910, Mar. 2020.

MENG, X.; DENG, X.; ZHU, S.; ZENG, B. Enhancing Quality for VVC Compressed Videos by Jointly Exploiting Spatial Details and Temporal Structure. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 2019., 2019. **Anais...** [S.l.: s.n.], 2019. p.1193–1197.

MOHAMMADI, P.; EBRAHIMI-MOGHADAM, A.; SHIRANI, S. Subjective and Objective Quality Assessment of Image: A Survey. **CoRR**, [S.l.], v.abs/1406.7799, 2014.

NADERNEJAD, E.; KORHONEN, J.; FORCHHAMMER, S.; BURINI, N. Enhancing Perceived quality of compressed images and video with ansiotropic diffusion and fuzzy filtering. **Elsevier Singnal Processing: Image Communication**, [S.l.], Jan. 2013.

NAM, H.; HAN, B. Learning multi-domain convolutional neural networks for visual tracking. In: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 2016. **Proceedings...** [S.l.: s.n.], 2016. p.4293–4302.

O'SHEA, K.; NASH, R. An Introduction to Convolutional Neural Networks. **CoRR**, [S.l.], v.abs/1511.08458, 2015.

PALAU, R. **Investigação de Soluções em Hardware para os Filtros de Laço do Decodificador AV1**. 2022. 103p. Tese de Doutorado (Ciência da Computação) — Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas.

POTOCNIK, P. **Solving XOR problem with a multilayer perceptron**. Disponível em: <https://web.fs.uni-lj.si/lasin/wp-content/include-me/neural/nn04$_m lp_x or/ > .Acessoem : 2023 - 08 - 31.$

RICHARDSON, I. E. G. **The H.264 Advanced Video Compression Standard**. 2nd.ed. [S.l.]: Wiley, 2010.

RONNEBERGER, O.; FISCHER, P.; BROX, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In: MEDICAL IMAGE COMPUTING AND COMPUTER-ASSISTED INTERVENTION – MICCAI 2015, 2015, Cham. **Anais...** Springer International Publishing, 2015. p.234–241.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. **Psychological review**, [S.l.], v.65, n.6, p.386, 1958.

ROTA, C.; BUZZELLI, M.; BIANCO, S.; SCHETTINI, R. Video restoration based on deep learning: a comprehensive survey. **Artificial Intelligence Review**, [S.l.], June 2022.

SALDANHA, M. **Exploration of Encoding Time Reduction Solutions for Intra-Frame Prediction of VVC Encoders**. 2021. 134p. Tese de Doutorado (Ciência da Computação) — Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas.

SHEN, M.-Y.; KUO, C. J. Review of Postprocessing Techniques from Compression Artifact Removal. **Journal of Visual Communication and Image Representation**, [S.l.], Mar. 1998.

STATISTA. **Semiconductor market size worldwide from 1987 to 2020**. Disponível em: <https://www.statista.com/statistics/266973/global-semiconductor-sales-since-1988/>. Acesso em: 2022-10-01.

SULLIVAN, G. J.; OHM, J.-R.; HAN, W.-J.; WIEGAND, T. Overview of the High Efficiency Video Coding (HEVC) Standard. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.22, n.12, p.1649–1668, 2012.

TANG, T.; LI, L. Adaptive deblocking method for low bitrate coded HEVC video. **Elsevier Journal of Visual Communication and Image Representation**, [S.l.], Apr. 2016.

TONG, J. et al. Learning-Based Multi-Frame Video Quality Enhancement. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 2019., 2019. **Anais...** [S.l.: s.n.], 2019. p.929–933.

UMNOV, A. V.; NASONOV, A.; KRYLOV, A.; YONG, D. Sparse Method for Ringing Artifact Detection. In: INTERNATIONAL CONFERENCE ON SIGNAL PROCESSING (ICSP), 12., 2014, Hangzhou, China. **Proceedings...** [S.l.: s.n.], 2014. p.662–667.

SALDANHA, M.; SANCHEZ, G.; MARCON, C.; AGOSTINI, L. **Versatile Video Coding (VVC)**. Cham: Springer International Publishing, 2022. p.7–22.

WANG, Q.; MA, Y.; ZHAO, K.; AL. et. A Comprehensive Survey of Loss Functions in Machine Learning. **Annals of Data Science**, [S.l.], v.9, p.187–212, 2022.

WANG, Y. et al. Dense residual convolutional neural network based in-loop filter for HEVC. In: IEEE VISUAL COMMUNICATIONS AND IMAGE PROCESSING (VCIP), 2018., 2018. **Anais...** [S.l.: s.n.], 2018. p.1–4.

WANG, Z.; SHEIKH, H. R.; BOVIK, A. C.; FURHT, B. The handbook of video databases: design and applications. **Objective video quality assessment**, [S.l.], p.1041–1078, 2003.

WU, B.; DUAN, H.; LIU, Z.; SUN, G. SRPGAN: perceptual generative adversarial network for single image super resolution. **arXiv preprint arXiv:1712.05927**, [S.l.], 2017.

XING, Q.; DENG, J. **PyTorch implementation of STDF**. https://github.com/ryanxingql/stdf-pytorch, version 1.0.0, 2020.

XING, Q. et al. MFQE 2.0: A New Approach for Multi-frame Quality Enhancement on Compressed Video. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, [S.l.], Oct. 2020.

XUE, T. et al. Video enhancement with task-oriented flow. **International Journal of Computer Vision**, [S.l.], v.127, n.8, p.1106–1125, 2019.

YANG, R. et al. Enhancing quality for HEVC compressed videos. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.29, n.7, p.2039–2054, 2018.

YANG, R.; MENTZER, F.; GOOL, L. V.; TIMOFTE, R. Learning for video compression with hierarchical quality and recurrent enhancement. In: IEEE/CVF CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 2020. **Proceedings...** [S.l.: s.n.], 2020. p.6628–6637.

YANG, R.; XU, M.; WANG, Z. Decoder-side HEVC quality enhancement with scalable convolutional neural network. In: IEEE INTERNATIONAL CONFERENCE ON MULTIMEDIA AND EXPO (ICME), 2017., 2017. **Anais...** [S.l.: s.n.], 2017. p.817–822.

YANG, R.; XU, M.; WANG, Z.; LI, T. Multi-frame quality enhancement for compressed video. In: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 2018. **Proceedings...** [S.l.: s.n.], 2018. p.6664–6673.

YOGATAMA, D.; D'ARBELOFF, T.; CLARK, C.; DYER, C. Learning and Evaluating General Purpose Representations. In: INTERNATIONAL CONFERENCE ON LEARNING REPRESENTATIONS (ICLR), 7., 2019. **Proceedings...** [S.l.: s.n.], 2019.

ZENG, K.; ZHAO, T.; REHMAN, A.; WANG, Z. Characterizing Perceptual Artifacts in Compressed Video Streams. **Human Vision and Electronic Imaging XIX**, [S.l.], Mar. 2014.

ZHANG, X. et al. Video Compression Artifact Reduction via Spatio-Temporal Multi-Hypothesis Prediction. **IEEE Transactions on Image Processing**, [S.l.], Oct. 2015.

ZHANG, Y. et al. Residual highway convolutional neural networks for in-loop filtering in HEVC. **IEEE Transactions on image processing**, [S.l.], v.27, n.8, p.3827–3841, 2018.

ZHOU, W.; BOVIK, A. C.; SHEIKH, H. R.; SIMONCELLI, E. P. Image quality assessment: from error visibility to structural similarity. **IEEE Transactions on Image Processing**, [S.l.], v.13, n.4, p.600–612, 2004.