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

Tese

Predição de *Light Fields* Utilizando Técnicas de Aprendizado Profundo

Ítalo Dombrowski Machado

Pelotas, 2025

Ítalo Dombrowski Machado

Predição de *Light Fields* Utilizando Técnicas de Aprendizado Profundo

Tese 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 Doutor em Ciência da Computação.

Orientador: Prof. Dr. Bruno Zatt
Coorientadores: Prof. Dr. Marcelo Porto
Prof. Dr. Daniel Palomino

Pelotas, 2025

Universidade Federal de Pelotas / Sistema de Bibliotecas
Catalogação da Publicação

M149p Machado, Ítalo Dombrowski

Predição de *Light Fields* utilizando técnicas de aprendizado profundo [recurso eletrônico] / Ítalo Dombrowski Machado ; Bruno Zatt, orientador ; Marcelo Porto, Daniel Palomino, coorientadores. — Pelotas, 2025.
148 f. : il.

Tese (Doutorado) — Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, 2025.

1. Light Fields. 2. Predição. 3. Aprendizado de máquina. 4. Codificação. I. Zatt, Bruno, orient. II. Porto, Marcelo, coorient. III. Palomino, Daniel, coorient. IV. Título.

CDD 005

Elaborada por Maria Inez Figueiredo Figas Machado CRB: 10/1612

Ítalo Dombrowski Machado

Predição de *Light Fields* Utilizando Técnicas de Aprendizado Profundo

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

Data da Defesa: 21 de fevereiro de 2025

Banca Examinadora:

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

Prof. Dr. Guilherme Corrêa.
Doutor em Engenharia Elétrica e de Computação pela Universidade de Coimbra.

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

Prof. Dr. Ismael Seidel
Doutor em Ciência da Computação pela Universidade Federal de Santa Catarina.

A melhor forma de prever o futuro é inventá-lo.
— ALAN KAY

RESUMO

MACHADO, Ítalo Dombrowski. **Predição de *Light Fields* Utilizando Técnicas de Aprendizado Profundo**. Orientador: Bruno Zatt. 2025. 148 f. Tese (Doutorado em Ciência da Computação) – Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2025.

O crescente uso de múltiplas câmeras tem levado pesquisadores a revisitar a teoria de Light Fields, que captura informações espaciais e angulares, aumentando a quantidade de dados armazenados e exigindo técnicas avançadas de compressão. Algumas abordagens comprimem Light Fields reorganizando-os em sequências pseudo-temporais ou utilizando JPEG-Pleno, mas a falta de predição em blocos ou a conversão para 3D pode diminuir a eficiência da compressão, criando oportunidades para explorar predições ao nível de blocos que aproveitem melhor a estrutura 4D. Além disto, pesquisas recentes têm obtido resultados interessantes ao utilizar algoritmos de aprendizado de máquina como redes neurais convolucionais para realizar predição em codificadores tanto de imagens como de vídeos. Contudo, existem inúmeras técnicas de treinamento e arquiteturas de redes neurais convolucionais, e seus desempenhos variam drasticamente com o tipo de tarefa e dado. Desta maneira, este trabalho propõe um método de treinar redes neurais convolucionais capazes de adaptar a predição intra de codificadores de vídeo para explorarem as redundâncias angulares e espaciais presentes nos Light Fields. Todas as etapas de avaliação e desenvolvimento durante o fluxo de trabalho foram minuciosamente analisadas, com uma explicação detalhada dos objetivos de cada técnica, bem como suas falhas e sucessos. O software de referência do EVC foi utilizado para avaliar diversas arquiteturas como autoencoders, Highway, Residuais, sob diferentes configurações de kernel e hiperparâmetros como data augmentation e métodos de decaimento do learning rate. Outro experimento realizado foi a comparação da métrica de SATD com a MSE e a SAD como funções de perda no treinamento. Ainda, técnicas de poda estruturada e não estruturada foram avaliadas para aperfeiçoar a eficiência dos modelos treinados. Ao final dos experimentos, os preditores resultantes são constituídos por aproximadamente 1,3M de parâmetros e, quando validados, atingiram um BD-Rate de -40,95% para o codificador HM e -46,89% para o codificador VTM. Quando validados realizando a predição da codificação de um segundo dataset, atingiram uma melhora de eficiência no codificador VTM de -30,09%. Ainda, os preditores se mostraram competitivos com o estado-da-arte de compressão de Light Fields e superaram os trabalhos relacionados em torno de -20%.

Palavras-chave: Light Fields. Predição. Aprendizado de máquina. Codificação.

ABSTRACT

MACHADO, Ítalo Dombrowski. **Prediction of Light Fields Using Deep Learning Techniques**. Advisor: Bruno Zatt. 2025. 148 f. Thesis (Doctorate in Computer Science) – Technology Development Center, Federal University of Pelotas, Pelotas, 2025.

The increasing use of multiple cameras has led researchers to revisit the theory of Light Fields, which captures spatial and angular information, increasing the amount of stored data and requiring advanced compression techniques. Some approaches compress Light Fields by reorganizing them into pseudo-temporal sequences or using JPEG-Pleno, but the lack of block-level prediction or conversion to 3D may reduce compression efficiency, creating opportunities to explore block-level predictions that better leverage the 4D structure. Furthermore, recent research has achieved promising results by employing machine learning algorithms, such as convolutional neural networks, to perform prediction in both image and video encoders. However, there are numerous training techniques and convolutional neural network architectures, and their performance varies significantly depending on the task and data type. Thus, this work proposes a method for training convolutional neural networks capable of adapting intra prediction in video encoders to exploit the angular and spatial redundancies present in Light Fields. All evaluation and development steps throughout the workflow were meticulously analyzed, providing a detailed explanation of the objectives of each technique, as well as their failures and successes. The EVC reference software was used to evaluate various architectures, such as autoencoders, Highway networks, and Residual networks, under different kernel configurations and hyperparameters, including data augmentation and learning rate decay methods. Another experiment conducted was the comparison of the SATD metric with MSE and SAD as loss functions during training. Additionally, structured and unstructured pruning techniques were evaluated to improve the efficiency of the trained models. At the end of the experiments, the resulting predictors consisted of approximately 1.3M parameters and, when validated, achieved a BD-Rate of -40.95% for the HM encoder and -46.89% for the VTM encoder. When validated by predicting the encoding of a second dataset, they achieved an efficiency improvement of -30.09% in the VTM encoder. Furthermore, the predictors proved to be competitive with the state of the art in Light Field compression, outperforming related works by approximately -20%.

Keywords: Light Fields. Prediction. Machine Learning. Encoding.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Os avanços em fotografia, vídeos e dispositivos móveis tornaram as câmeras mais poderosas e populares. A maioria dos celulares agora pode tirar fotos e gravar vídeos com resolução de 4K (4096 $\times$ 2160), enquanto aplicam algoritmos para melhorar a qualidade da imagem. Por exemplo, usar interpolações para criar uma imagem mais estável e com foco ajustado através das informações das fotos tiradas de diferentes ângulos, exposições ou instantes de tempo.

Recentemente, para ampliar a performance de técnicas deste tipo, os celulares vêm adicionando um maior número de câmeras traseiras que lhes permitem capturar informações espaciais e de profundidade. Essas técnicas e informações são também largamente utilizadas em ambientes de realidade aumentada e mista. Pois, com as informações adicionais, é possível realizar tratamentos e operações mais avançadas nas fotos. Algumas dessas técnicas têm como base explorar o efeito de paralaxe, ou seja, o deslocamento aparente de um objeto devido a uma mudança de posição no ponto de vista do observador. Desta forma, tendo-se ao menos duas câmeras em um dispositivo capturando de posições diferentes uma mesma cena, é possível utilizar o efeito de paralaxe para estimar a distância entre os objetos capturados e as câmeras. Essa informação sobre a distância pode ser utilizada para se obter mapas de profundidade da cena, o que é muito útil para operações de autofoco, refoco, ajuste de nitidez, estabilização de vídeos, classificação de materiais e até mesmo algoritmos de reconhecimento facial (CHEN; LU; SU, 2010; WU et al., 2017).

Os mesmos princípios podem ser utilizados não somente na gravação de imagens, mas também de vídeos e mídias imersivas. Seguindo naturalmente essa evolução tecnológica, novas formas de adquirir informações adicionais na captura de fotos e vídeos começaram a ser pesquisadas. O *Google Pixel*, por exemplo, ao invés de utilizar múltiplas câmeras, propõe uma técnica de captura estéreo, onde cada pixel é composto por duas amostras, cada uma obtida por uma das câmeras. Outra técnica que vem sendo estudada é uma teoria concebida há mais de sete décadas por Gershun (1939) denominada de *Light Fields* (LFs). Essa teoria tem se tornado um tema de pesquisa crescente e é a aposta tecnológica de diversos laboratórios e empresas para o futuro

das câmeras, desde dispositivos móveis e realidade aumentada até produção cinematográfica.

Light Fields digitais são uma simplificação da função plenóptica que descreve todos os raios de luz no espaço através de sete variáveis usando a função $P(x, y, z, \theta, \phi, t, \lambda)$ (ADELSON; WANG, 1992). Em outras palavras, um Light Field digital é a captura de uma cena a partir de diversos ângulos fixos distintos, tornando assim possível descrever os raios de luz incidentes na cena provenientes de múltiplas direções.

Um light field pode ser obtido de múltiplas maneiras, sendo uma alternativa prática a utilização de uma câmera de *Light Field*. Tais câmeras possuem uma matriz 2D de micro-lentes além da lente principal, onde cada micro-lente captura informação da cena de um ângulo diferente. Desta maneira, essas câmeras capturam uma matriz 2D de amostras para cada pixel. Ao organizar os micro-píxeis de uma imagem obtida com uma câmera Lytro com resolução de $625 \times 434 \times 15 \times 15$, obtém-se uma matriz 15×15 de vistas (SubAperture Images SAIs), cada uma representando a cena sob um ângulo ligeiramente diferente. Desta maneira, as diferentes vistas de um LF com 15×15 vistas podem ser representadas em *Multi-View* conforme a Figura 1.

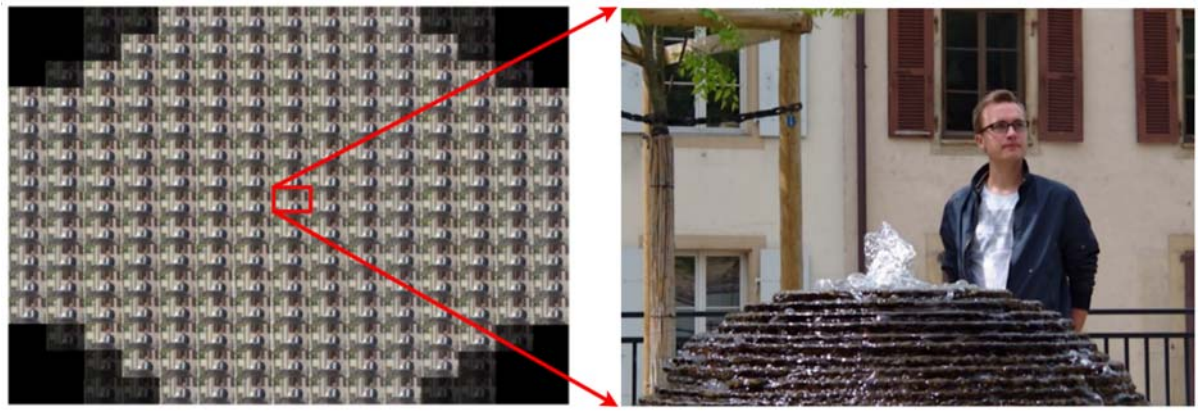


Figura 1 – Todas Vistas em *Light Field Fountain & Vincent 2* em Multi-view e sua Vista Central
Fonte: Desenvolvido pelo autor.

Apesar de LFs fornecerem muito mais informações e possibilitarem tratamentos e operações de imagens mais eficientes, eles demandam um volume de dados significativamente superior ao de fotografias e vídeos comuns para serem processados e armazenados (WU et al., 2017). Dadas as resoluções mencionadas, uma fotografia *lenslet* típica acaba necessitando armazenar 225 fotos com resolução 625×434, ou seja, uma foto para cada vista da matriz de 15×15 micro-píxeis. Ao calcular esses dados, conclui-se que uma fotografia *lenslet* necessita armazenar aproximadamente 61 milhões de píxeis, enquanto uma câmera 8K (8192 $\times$ 4320) precisa armazenar pouco mais de 33 milhões de píxeis. Números que representam um aumento de 96% na quantidade de informações a serem armazenadas, seja em um computador, servidor ou dispositivo móvel.

Para possibilitar o armazenamento deste tipo de dado, foram propostos codificadores para *Light Fields* que foram desenvolvidos visando explorar suas características específicas. Carvalho et al. (2018) propôs o *Multidimensional Light Field Encoder Using 4D Transforms and Hexadeca-trees* (MuLE). Para explorar a estrutura de quatro dimensões dos LFs, o MuLE separa o LF de entrada em blocos 4D com dimensões entre $125 \times 125 \times 25 \times 25$ até $13 \times 13 \times 4 \times 4$ amostras. Então, o encoder aplica DCTs-4D nestas amostras e utiliza uma *Hexadeca-tree Bitplane* (HT-B) para quantizar os dados. Devido a esta melhor exploração da estrutura dos dados de um LF, o MuLE obtém eficiências de codificação superiores ao *software de referência* do HEVC em torno de -38,12% (CARVALHO et al., 2018). Contudo, o MuLE não utiliza nenhuma forma de predição em sua codificação, abrindo a possibilidade de unir as técnicas de predição de codificadores de vídeo com as estruturas 4D dos codificadores de LF para obter eficiências de compressão ainda maiores. Posteriormente, este codificador ainda foi aperfeiçoado com modos inclinados (*Slanted Modes*), melhorando sua eficiência de codificação em -31.03% (CARVALHO et al., 2023).

Além do MuLE, o codificador WaSP (*Hierarchical Warping, Merging, and Sparse Prediction*) (ASTOLA; TABUS, 2018a) também foi desenvolvido com foco para LFs. O WaSP divide as vistas em hierarquias e codifica as hierarquias mais baixas utilizando o codificador JPEG 2000 (SKODRAS; CHRISTOPOULOS; EBRAHIMI, 2001). Estas vistas codificadas são utilizadas como vistas de referência em conjunto com mapas de profundidade para sintetizar as vistas intermediárias de hierarquias posteriores. Então, o resíduo entre a vista sintetizada e a de referência é gerado para ser codificado.

Tanto o codificador MuLE quanto o WaSP foram integrados ao codificador JPEG-Pleno (SCHELKENS et al., 2019) para proporcionar soluções em codificação de LFs. Enquanto o MuLE foi denominado modo de transformadas, o WaSP foi denominado modo de predição. Contudo, este modo não realiza predição em blocos como codificadores de vídeo, mas sim sintetiza vistas inteiras com base em vistas de referência, sendo assim técnicas com desafios distintos. Ademais, ambos os modos presentes no JPEG Pleno ainda não são amplamente suportados por dispositivos comerciais. Logo, diferentemente de determinados codificadores de vídeo, para utilizar em dispositivos móveis estes codificadores especializados em LFs, é necessário um conjunto de adaptações e implementações em *hardware*, o que gera custos indesejados.

Outra alternativa atualmente sendo adotada é utilizar compressores de vídeo para codificar os diferentes quadros do LF como se fossem a sequência temporal de um vídeo (pseudo-vídeo). Realizar a compressão eficiente de vídeos de alta resolução já é um desafio considerável de pesquisa, obtendo investimentos e dedicação de várias empresas de tecnologia. Entre elas estão a *Google, Intel, Cisco, Mozilla, Microsoft, Netflix, Amazon* e muitas outras que se uniram e formaram a *Alliance for Open Media* (AOM) (AOMEDIA, 2019). Essa aliança tem como objetivo desenvolver codificadores

de vídeo de alta resolução e livre de royalties como o *AOMedia Video* 1 (AV1) (AOMEDIA, 2019) e o *AOMedia Video* 2 (AV2), este ainda em desenvolvimento. Além dos codecs da AOM, existem os padrões da família H.26x como, por exemplo, o H.266, ou VVC *Versatile Video Coding* (BROSS et al., 2021), lançado em 2020, que atualmente é considerado o estado da arte em codificação de vídeo.

Para se codificar um LF como um pseudo-vídeo, cada uma das suas SAIs é interpretada como um quadro temporalmente sequencial, o que não é coerente com a realidade. Todas as SAIs de um LF pertencem a um mesmo instante de tempo, porém com um ângulo distinto. Por outro lado, cada quadro de um vídeo está em um instante de tempo diferente, porém geralmente sob o mesmo ângulo. Desta maneira, é natural que o comportamento entre os quadros e o deslocamento nas suas respectivas amostras sejam distintos. Ainda, a maneira de se representar ambos os tipos de dados pode diferir consideravelmente. Por exemplo, um LF pode ser representado em uma estrutura de dados 4D constituída por uma matriz 2D de SAIs (que já são imagens 2D), constituída pela fórmula $L(r, s, u, v)$ descrita anteriormente. Este formato de representação de LFs é denominado *Lenslet* e, para melhor representação visual, pode ser comprimido em uma única imagem 2D como representado pela Figura 2. Neste formato de representação, os píxeis de todas as SAIs são reorganizados conforme o seu eixo angular. Desta maneira, cada pixel do LF em formato *lenslet* é constituído pelos seus 15×15 micro-píxeis correspondentes de todas as 15×15 SAIs. Os conjuntos de 15×15 micro-píxeis podem ser identificados como os círculos com dimensões 15×15 na ampliação à direita da figura, onde denominamos cada círculo de Micro-Imagem.

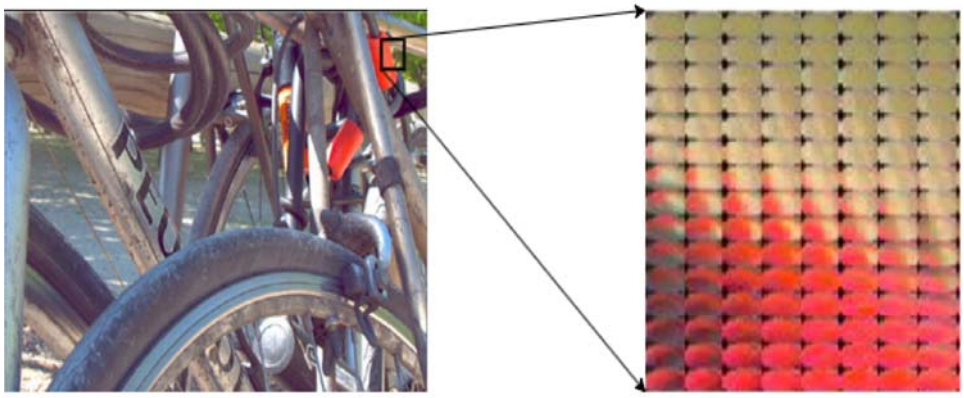


Figura 2 – Exemplo de um Light Field no formato lenslet com ampliação de suas MIs em uma região.
Fonte: Desenvolvido pelo autor.

Já para um vídeo, uma representação 3D (1D+2D), ou seja, uma dimensão temporal linear de quadros 2D, se torna muito mais fiel à estrutura dos dados. A comparação da estrutura de vistas angulares *multi-view* de um *Light Field* e de quadros temporalmente vizinhos de um vídeo pode ser observada na Figura 3. Desta maneira, a estrutura dos dados de um LF difere da de um vídeo. Enquanto um vídeo possui alta

redundância devido à proximidade temporal dos quadros, um LF possui redundâncias angulares e espaciais, derivadas das múltiplas vistas das cenas. Esta diferença na natureza dos dados faz com que a predição inter-quadros, baseada em compensação de movimento, se comporte de maneira distinta entre vídeos e LFs. Da mesma forma, as interpolações realizadas pelos preditores intra-quadro dos codificadores de vídeo não cobrem os comportamentos presentes entre as vistas dos *Light Fields* evidenciadas pelas MIs. Logo, podemos afirmar que, apesar de atingirem eficiências aceitáveis na compressão de LFs, codificadores de vídeo não exploram algumas características específicas dos LFs, o que abre espaço para o desenvolvimento de técnicas e metodologias que possam obter eficiências ainda maiores ao explorar estas características (X. JIN, 2023).

Figura 3 – Comparação entre (a) Estrutura Multi-view de um LF e (b) Quadros Sequenciais de um Video
Fonte: Desenvolvido pelo autor.

Com os recentes avanços de aprendizado profundo, as redes neurais têm demonstrado notável eficiência em diversas tarefas de processamento de sinais, muitas vezes superando métodos tradicionais baseados em ferramentas puramente matemáticas ou estatísticas. No contexto de compressão de imagens e vídeos, abordagens baseadas em redes neurais convolucionais (RNCs) têm alcançado desempenho superior a algoritmos clássicos, explorando de maneira mais eficaz as correlações espaciais e temporais nos dados (BALLÉ; LAPARRA; SIMONCELLI, 2018; LU et al., 2019). Modelos de aprendizado profundo conseguem identificar padrões nos dados e até mesmo gerar representações compactas dos dados (THEIS et al., 2017). Além disso, arquiteturas baseadas em *autoencoders* variacionais e *transformers* vêm sendo exploradas para otimizar a eficiência da codificação em padrões emergentes, como o H.266/VVC e JPEG AI (CHENG et al., 2020; MENTZER et al., 2022). Estes estudos demonstram que a aplicação de redes neurais no processo de compressão de *light fields* vem a ser uma abordagem promissora, permitindo explorar os padrões e redundâncias espacial-angulares presentes neste tipo de dados.

1.1 Objetivos da Tese

Esta tese visa abordar os desafios envolvidos no desenvolvimento de predições intra-quadro que explorem as redundâncias espaciais e angulares na estrutura 4D dos *Light Fields*. Para isso, será proposto um método de avaliação e desenvolvimento de RNCs que realizem predição intra-quadro para codificadores de vídeo.

Com base nos avanços das técnicas de aprendizado profundo, defendemos que redes neurais convolucionais possuem a capacidade necessária para realizar predição intra-quadro de maneira eficiente. Para isso, esta tese avalia todos os aspectos, desde o tratamento do *dataset* até o treinamento de diferentes topologias de RNCs sobre os LFs para desenvolver preditores intra-quadro que melhor explorem as distintas características de um *light field*. E, quando inseridos como preditores em codificadores de vídeo, os tornem capazes de comprimir LFs no formato *Lenslet*. Para explorar as redundâncias espaciais em conjunto com as angulares, utilizaremos os LFs no seu formato lenslet, que concentra as informações angulares.

A Figura 4 demonstra a estratégia de predição proposta onde uma RNC receberá blocos de referência (demonstrados em A, B e C) para, através das similaridades espaciais e angulares presentes nestes blocos, predizer o conteúdo do bloco sendo atualmente codificado (bloco em cinza). Como nas predições anteriores, o preditor RNC tem como objetivo predizer o bloco o mais próximo possível do original para, dessa forma, gerar resíduos o mais próximo de zero possível. Os preditores desenvolvidos nesta tese devem ser genéricos e aplicáveis em qualquer codificador de vídeo que possua uma predição intra-quadro baseada em blocos. Portanto, os preditores desenvolvidos serão experimentados nos codificadores EVC, HEVC e validados sem modificação no codificador VVC. Ressaltamos que os codificadores escolhidos possuem eficiências, complexidades e técnicas de predição intra consideravelmente diferentes, fortalecendo a avaliação de generalidade dos preditores desenvolvidos. Como as dimensões angulares de um LF costumam ser muito menores que as espaciais (15×15 se obtido com uma câmera Lytro), e dos tamanhos de bloco utilizados nestes codificadores, a rede poderá explorar simultaneamente informações espaciais e angulares

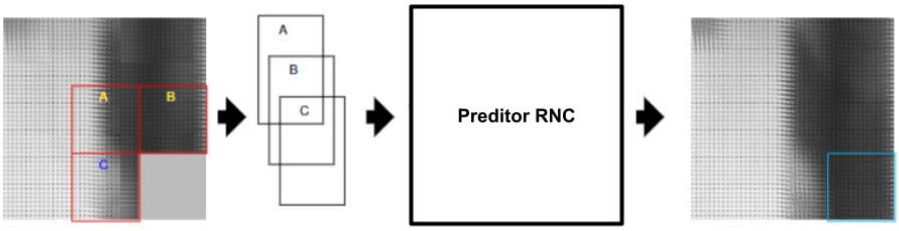


Figura 4 – Estrutura de Predição Intra-quadro Utilizando RNCs
Fonte: Desenvolvido pelo autor.

destes blocos tendo em vista que os LFs são utilizados na representação lenslet.

Ademais, cabe também investigar a eficiência de diferentes topologias RNCs como, por exemplo, as topologias de *autoencoder, Resnets, HighwayNets* e *UNets*. Ainda, qualquer arquitetura de aprendizado profundo tem sua eficiência sujeita a como é realizado o seu treinamento. E, por sua vez, o treinamento deve ser ajustado para o problema sendo resolvido por meio de hiperparâmetros como *learning rate* e *loss function* e técnicas como *data augmentation*. Desta maneira, para melhor avaliar o desempenho das diferentes arquiteturas, esta tese também apresenta um método para realizar avaliações sobre o impacto de cada um destes hiperparâmetros e técnicas que permite identificar quais as mais eficientes a serem utilizadas no codificador desejado. Uma avaliação extensiva de múltiplas técnicas e arquiteturas para cada passo do treinamento de um preditor RNC será apresentada. Ainda, a motivação e o desempenho de cada experimento realizado na avaliação são discutidos em profundidade.

Como subproduto do método de avaliação proposto, esta tese disponibiliza um programa que permite treinar qualquer uma das arquiteturas sob qualquer uma das técnicas experimentadas para qualquer codificador de vídeo, independente das dimensões da sua estrutura de partição de blocos. Ainda, o programa que pode ser encontrado em (MACHADO, 2025) permite a adição de arquiteturas adicionais de redes neurais convolucionais sem a necessidade de modificações internas.

Por fim, codificadores são comumente implementados em hardware para otimizar sua execução. Logo, tem-se uma preocupação com o tempo de execução e complexidade de implementação em hardware de suas ferramentas. Para aperfeiçoar o preditor proposto neste aspecto, técnicas de *pruning* não estruturado e estruturado também são avaliadas.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo discutiremos os principais conceitos necessários para a compreensão desta tese. Partindo da base de aplicação, iniciaremos discutindo na Seção 2.1 o que são *Light Fields*, como surgiram e como são capturados. Posteriormente, para melhor compreender os desafios de comprimir um LF e os objetivos desta tese, discutiremos na Seção 2.2.1 os princípios da codificação de vídeo. Posteriormente, na Seção 2.3, apresentaremos o funcionamento dos codificadores desenvolvidos especificamente para comprimir LFs propostos no JPEG Pleno.

A partir da seção 2.4 mudamos o foco deste capítulo para conceitos de redes neurais. Como esta tese explora múltiplas arquiteturas de redes neurais convolucionais, a seção 2.5 estende a discussão para redes neurais profundas. Por fim, na Seção 2.6 são discutidas técnicas de *pruning* estruturado e não estruturado que são explicadas.

2.1 Light Fields e a Função Plenóptica

Um *Light Field* é baseado na função plenóptica, a qual é uma função matemática que modela a descrição de toda informação visual em qualquer ponto do espaço para qualquer instante de tempo proposto por Faraday (1846). Essa primeira proposta considerava que cada ponto do espaço carrega informações sobre a intensidade e direção da radiação luminosa, de maneira similar às linhas de campo elétrico e magnético. No entanto, com o avanço dos estudos na área de óptica e computação gráfica, a representação dos Light Fields passou a ser reformulada sob uma perspectiva geométrica, focada na parametrização dos raios de luz ao longo do espaço, afastando-se da abordagem física original. Nesta abordagem geométrica, Gershun (1939) definiu com maior precisão conceitos como *luminosidade*, *iluminação*, *intensidade* e *brilho*. Por fim, Adelson; Wang (1992) definiram a função plenóptica conforme a 1.

$$P(\theta, \phi, \lambda, t, x, y, z) \tag{1}$$

Para descrever todas as informações visuais em qualquer ponto do espaço, a função plenóptica necessita das 7 dimensões definidas em 1. Entre estas dimensões,

x, y, z representam a posição do ponto de vista na cena 3D retratada em um determinado instante de tempo t. Já o λ representa o comprimento da onda eletromagnética (coloração da luz) naquele ponto, enquanto θ e ϕ representam o ângulo de incidência da luz.

A representação de todos os raios de luz em qualquer lugar de um espaço 3D em qualquer instante de tempo é uma tarefa custosa, por exigir uma quantidade de informações e processamento gigantesca. A forma de seu funcionamento e de ser calculada pode ser comparada à de um *ray tracing* devido à grande dimensionalidade e falta de discretização de certas dimensões, o que leva a trabalhos a até mesmo utilizarem está técnica para renderizarem LFs (LEVOY; HANRAHAN, 1996). Contudo, ao utilizarmos *Light Fields* em um espaço discreto, permite-se a restrição e simplificação de determinadas dimensões.

2.1.1 Light Fields Digitais

Em Mcmillan; Bishop (1995) *Light Fields* são definidos como fotos panorâmicas em diferentes pontos em um espaço tridimensional, onde cada ponto é denominado como uma "vista" diferente da cena. Ao considerarmos um LF como uma foto temporalmente estática, é possível desconsiderarmos a dimensão t de tempo, podendo assim reduzir a função plenóptica para 6 dimensões e, consequentemente, diminuir também a quantidade de informações para representar o *Light Field*. Observe, contudo, que *Light Fields* dinâmicos já são explorados na literatura (WILBURN et al., 2005) e serão utilizados como exemplos para possíveis estudos.

Ainda, ao utilizarmos canais de cores como RGB ou YCbCr para representar a cor da luz em um sistema computacional, podemos considerar estas informações como um objeto à parte e, desta forma, discretizar a dimensão λ. De acordo com Levoy; Hanrahan (1996), também pode-se assumir que a luz se propaga linearmente quando capturada por um dispositivo, ou seja, não há necessidade de considerar obstáculos entre o raio de luz e o ponto de captura. Logo, as informações sobre a profundidade da captura z também não possuem relevância e podem ser desconsideradas. Com estas simplificações, é possível representarmos a função plenóptica através de 4 dimensões, conforme 2.

$$P(\theta, \phi, x, y) \tag{2}$$

Por fim, já que os pontos de vista serão finitos por estarem sendo representados como uma mídia digital, as dimensões que representam o ângulo de incidência da luz, θ e ϕ, podem ser discretizadas, permitindo que a função plenóptica seja reescrita para utilizar apenas as 4 dimensões discretizadas conforme a 3 (LEVOY; HANRAHAN, 1996).

$$P(u, v, x, y) \tag{3}$$

Vale notar que, apesar de destas simplificações e discretizações, ainda é possível realizar diversas operações nos *Light Fields* que não são triviais em mídias 2D e 3D como, por exemplo, mudança de perspectiva e refoco (IHRKE; RESTREPO; MIGNARD-DEBISE, 2016).

2.1.2 Captura de Light Fields

As tecnologias de captura de *Light Fields* emergiram recentemente, isto faz com que, no cenário atual, existam diferentes formas de se capturar um *Light Field* sem haver uma padronização única definida como estado da arte. Desta maneira, pode-se dizer que existem quatro principais maneiras de se capturar uma cena em um *Light Field* digital.

A primeira maneira, e possivelmente a mais custosa monetariamente, é capturar o *Light Field* através de um *array* de câmeras. Neste modelo de captura, diversas câmeras são dispostas em formato de uma matriz 2D onde cada câmera capturará uma das vistas do *Light Field*. Para o LF ser capturado e processado corretamente, o espaçamento entre as câmeras deve ser precisamente regulado, pois ao se alterar o ângulo entre as câmeras, as distâncias de paralaxe entre as vistas também serão alteradas, ocasionando um erro no cálculo de operações como refoco e incoerências visuais entre as vistas. A Figura 5 apresenta um exemplo de matriz de câmeras para obtenção de *Light Fields* e deixa clara a relação desta matriz de câmeras com a matriz de vistas apresentada na Figura 1 (a).

Figura 5 – Array de Câmeras de Stanford para Obtenção de Light Fields
Fonte: Adaptado de (ELECTROSOME, 2024).

Outra técnica para adquirir *light fields* consiste na captura de fotos de diferentes ângulos da cena com uma câmera de imagens naturais através da sua movimentação

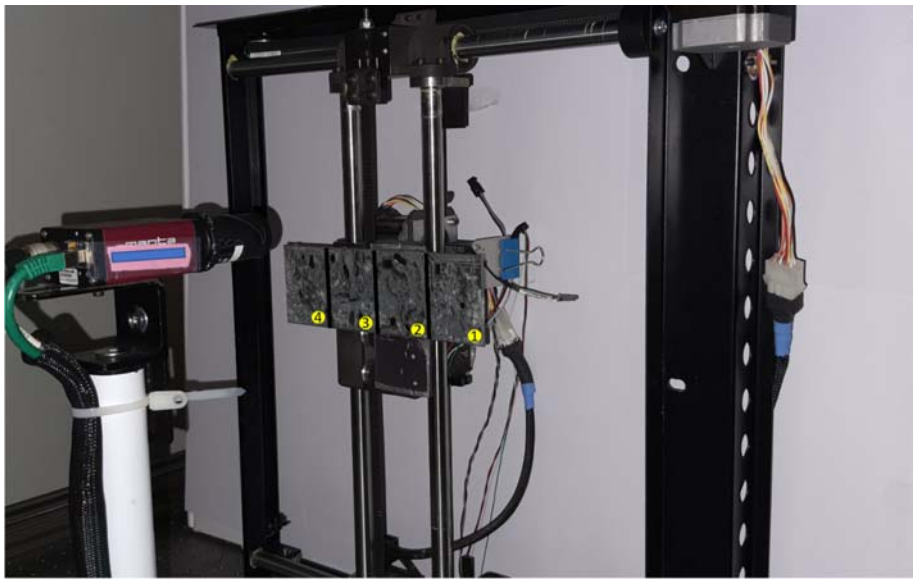


Figura 6 – Sistema de captura de light fields através de um braço mecânico.
Fonte: (LUXMAN et al., 2022).

para capturar a cena de diferentes ângulos. Contudo, este método de captura necessita que a movimentação da câmera entre as vistas esteja precisamente calibrada para manter coerentes as distâncias de captura entre as vistas. Devido a esta necessidade, muitos trabalhos fazem uso de braços mecânicos para realizar a captura dos LFs. Um exemplo de sistema mecânico para obtenção de *light fields* pode ser visto na figura 6 proposto por Luxman et al. (2022). Porém, o uso destes sistemas torna-se imprático devido à plataforma estática e espaçosa que necessita estar sempre devidamente calibrada, o que leva estas plataformas a serem utilizadas para obtenção de cenas isoladas. Por exemplo, o sistema proposto em Luxman et al. (2022) tem como propósito capturar superfícies de objetos de patrimônio cultural de médio e grande porte para estudos. Ainda, podem ser obtidos através de câmeras de imagens naturais, e *softwares* de CAD 3D também podem ser utilizados para gerar um cenário de onde serão obtidos os *Light Fields*. Um LF gerado artificialmente por meio de um *software* é denominado de *Light Field* Sintético.

Por fim, foram desenvolvidas câmeras como a *Lytro Illum* (RAYTRIX, 2022) e a Raytrix (RAYTRIX, 2022) capazes de capturar *Light Fields* sem a necessidade de múltiplas câmeras, sincronizações ou calibrações. Os LFs capturados por essas câmeras são estruturados no formato *Lenslet* conforme a Figura 2 e são considerados densos, ou seja, uma proximidade maior entre as vistas do que os LFs obtidos pelos *arrays de câmeras* (LFs esparsos).

Uma câmera para capturar *Light Fields* possui tecnologia de microlentes que não está presente em câmeras convencionais. Conforme pode ser observado na Figura 8 além da lente principal, a câmera possui um *array* de micro lentes responsáveis por dividir os raios de luz que representam a cena capturada em diferentes ângulos.

Figura 7 – Foto de uma camera Lytro Illum.
Fonte: Adaptado de Zhang et al. (2020).

Perceba também que, para capturar estes raios de luz nesta granularidade mais fina, é necessário também um maior número de fotosensores. Por exemplo, uma câmera Lenslet *Lytro Illum* ilustrada pela Figura 7, possui resolução de $625 \times 434 \times 15 \times 15$, sendo 625×434 sua resolução espacial definida pela sua lente principal e 15×15 sua resolução angular, definida pelas suas micro-lentes. Isto quer dizer que, os *light fields* obtidos por esta câmera capturam 15×15 amostras de micro-píxeis para cada um dos seus 625×434 píxeis, onde as posições de cada pixel representam os valores de r e s, enquanto as posições dos micro-píxeis são representadas pelas variáveis u e v da função L. Vale salientar que podem existir câmeras de *Light Fields* com diferentes resoluções e disposições de píxeis.

Considerando uma câmera *Lytro Illum*, para cada um de seus 625×434 píxeis, tem-se uma matriz de 15×15 sub-píxeis. Ao considerarmos 3 canais de cores (RGB ou YCbCr) com 10 *bits* por canal, um LF obtido por esta câmera necessitaria de 218,26MB para ser armazenado, e se compararmos aos 7,7MB necessários para armazenar uma foto *Full HD* (1920×1080), fica clara a necessidade de métodos eficientes para a compressão de *Light Fields* (ZHAO et al., 2016; CONCEICAO et al., 2018; CARVALHO et al., 2018).

2.2 Codificação de Sinais Visuais

Este projeto visa desenvolver algoritmos de predição eficientes para *Light Fields* com o objetivo de melhorar a eficiência de codificação de codificadores de LFs. Para isso, é necessário o amplo entendimento do funcionamento interno de um codificador e de suas ferramentas, especialmente sobre a etapa de predição. Não obstante, é de suma importância também explicitar as diferenças entre um codificador de vídeo e um codificador de LFs.

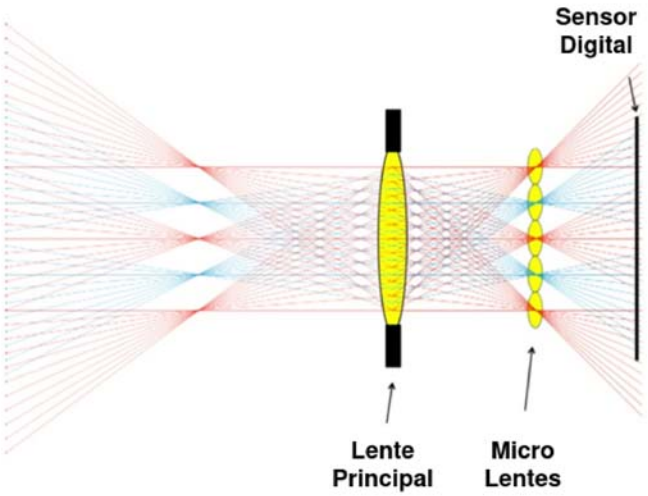


Figura 8 – Estrutura de Lentes de uma câmera Lytro Illum
Fonte: Adaptado de Schelkens et al. (2019).

2.2.1 Codificação de vídeo

Vídeos digitais são compostos por uma sequência de imagens (fotos) denominadas quadros ou *frames*. Estes são capturados e reproduzidos em uma taxa de amostragem fixa chamada de *frame rate* (FPS). Todos os quadros de um determinado vídeo possuem uma resolução que indica quantos píxeis formam a imagem horizontal e verticalmente. Cada pixel é representado conforme um sistema de representação de cores. Em codificadores de vídeo, de maneira geral, o sistema de cores predominante é o YCbCr, onde são armazenadas informações sobre luminância (Y) e cor (crominância azul - (Cb) e crominância vermelha - (Cr)) com valores reais entre 0 e 255 (RICHARDSON, 2002). O sistema YCbCr é largamente utilizado por separar as informações de luminância, as quais são muito perceptíveis para o sistema visual humano, das informações de crominância (não tão perceptíveis). Desta forma, este sistema permite que se diminua a precisão/resolução dos dados de crominância e, consequentemente, a quantidade de informações a se armazenar, em troca de perdas visuais pouco perceptíveis ou até mesmo imperceptíveis ao olho humano.

O objetivo do codificador é reduzir as informações redundantes sobre os píxeis de um vídeo com o mínimo de perda possível na qualidade visual do mesmo. Deste modo, para realizar reduções de informação no canal Y, é necessário realizar processos mais complexos e eficientes. De acordo com Ghanbari (2003), para atingir este objetivo, os codificadores buscam explorar eficientemente três tipos de redundância presentes em vídeos digitais:

- Redundância Espacial: Diz respeito a similaridade entre os píxeis próximos em um determinado quadro. Por exemplo, uma cena com uma parede branca terá diversos píxeis brancos similares e vizinhos.

- Redundância Temporal: Como mencionado anteriormente, um vídeo costuma ter pelo menos 30 quadros por segundo de vídeo. Logo, quadros temporalmente vizinhos tendem a ser muito similares devido ao baixo deslocamento dos objetos da cena em um espaço de 0,03s. Esta similaridade entre quadros vizinhos representa uma redundância temporal.
- Redundância Entrópica: É a similaridade nos símbolos utilizados para representar o sinal.

Para explorar estas três redundâncias de maneira eficiente, os codificadores de vídeo seguem o fluxo apresentado pelo diagrama de blocos presente na Figura 9.

Devido ao fato de regiões próximas em um quadro tenderem a ser similares (Redundância Espacial), na primeira etapa da codificação, o quadro sendo comprimido tem suas regiões divididas em blocos. Os tamanhos destes blocos podem variar e tendem a se adequar à homogeneidade da região. Uma região com muitos detalhes tende a ser dividida em blocos menores que chegam até a 4×4 píxeis, possibilitando assim que o codificador encontre similaridades em regiões específicas. Já uma região com texturas simples e cores mais homogêneas tende a ser particionada em blocos maiores com tamanhos que podem exceder 64×64 píxeis, dependendo do codificador adotado.

Após divididos, os blocos podem ser preditos utilizando a predição intra-quadro ou inter-quadros. A predição intra-quadro é a principal técnica responsável por explorar redundâncias espaciais, enquanto a predição inter-quadros é responsável por identificar e explorar redundâncias temporais entre múltiplos quadros. Ambas as predições serão abordadas na próxima seção. Contudo, independente da predição selecionada para o bloco atual, o bloco predito pela ferramenta será subtraído do bloco original, gerando como resultado um resíduo (diferença) entre o bloco predito e o bloco sendo codificado.

Ao se utilizar uma predição eficiente, o bloco predito será similar ao bloco codificado, fazendo assim com que o resíduo resultante seja constituído de valores baixos e próximos de zero. Um maior número de zeros e valores baixos significa um aumento na redundância entrópica (redução na entropia do sinal) que será explorada posteriormente pelo codificador. Para reduzir mais os resíduos, eles passam pela etapa de Transformada e Quantização. Na etapa das transformadas, um algoritmo de DCT (*Discrete Cosine Transform*) ou DST (*Discrete Sine Transform*) é tipicamente utilizado para converter os dados do domínio espacial para o domínio das frequências. Neste domínio, é possível separar informações de acordo com suas frequências, sendo assim possível separar as informações com baixas frequências das com altas frequências, concentrando-as em uma região do bloco.

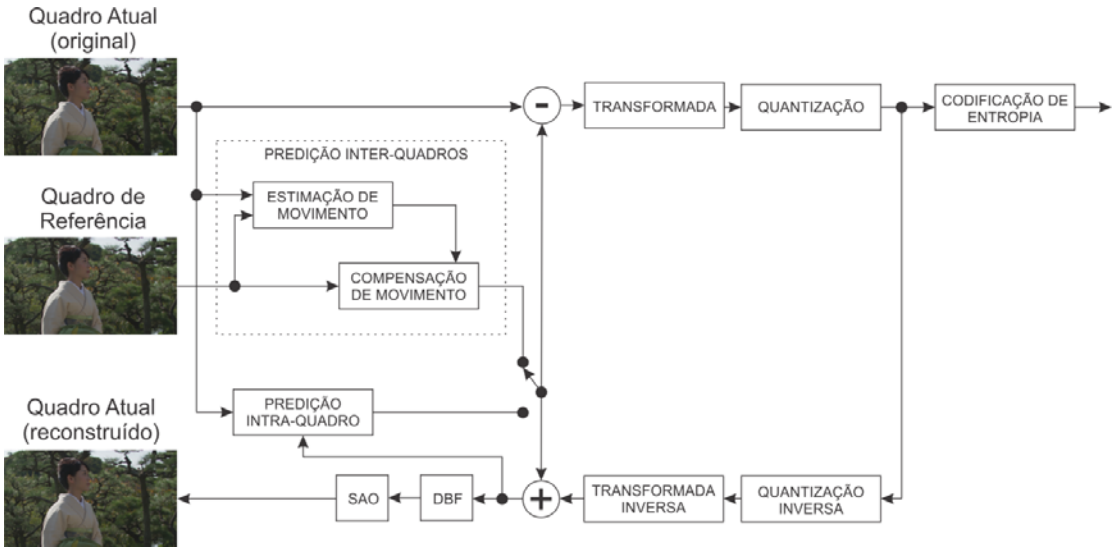


Figura 9 – Diagrama de Blocos de um Codificador de Vídeo Genérico
Fonte: Adaptado de (AFONSO, 2013).

Essa concentração de baixas frequências é interessante, pois, além de serem representadas por coeficientes de maior magnitude, também são as informações mais sensíveis ao olho humano. Estas características nos permitem, assim como feito nos canais de crominância, atenuar especificamente informações não tão perceptíveis (altas frequências) para reduzir a entropia do sinal. Para tanto, os coeficientes transformados passam pela etapa de Quantização, onde são divididos (divisão inteira) por um valor controlado pelo parâmetro denominado QP (*Quantization Parameter*).

Posteriormente, após quantizados, os dados são enviados para o codificador de entropia que explora as redundâncias entrópicas do sinal quantizado. Para isso, o codificador de entropia minimiza os *bits* necessários para representar os símbolos mais frequentes. Por fim, a saída da etapa de entropia é o *bitstream* final da codificação, que poderá ser armazenado ou transmitido.

Contudo, como os quadros já codificados são necessários como quadros de referência para a etapa de predição, o *bitstream* também passa pelas etapas de quantização e transformadas inversas que resultam no resíduo reconstruído. Após ser somado com os valores do bloco predito, tem-se o quadro reconstruído e pronto para ser utilizado pela predição.

2.2.2 Predição em Vídeos

Conforme mencionado, codificadores de vídeo como o HEVC, VVC e o AV1 possuem dois tipos de predição, a predição intra-quadro e a predição inter-quadros. Ambas as ferramentas provêm ganhos significativos de eficiência de codificação para os codificadores e possuem funcionamentos distintos.

2.2.2.1 Predição Intra-Quadro

Este modo de predição visa comprimir informações utilizando redundâncias presentes em um único quadro, como uma área homogênea ou padrões complexos que se repetem no mesmo quadro. A proposta desta forma de predição é utilizar as bordas de um bloco (bordas externas, ou seja, pixels pertencentes a blocos vizinhos) para prever os píxeis presentes dentro do bloco. Por ser útil para compressão de informações de uma única imagem, a predição intra-quadro é utilizada em padrões de compressão de imagem como no modo *lossless JPEG* do (*Joint Photographic Experts Group*) (PENNEBAKER; MITCHELL, 1992). Devido ao fato de não utilizar quadros de referência, este modo é sempre utilizado para o primeiro quadro codificado em um vídeo.

A predição intra-quadro pode ser realizada de maneira direcional ou não direcional. Ainda, o número de modos pode variar entre os padrões. Por exemplo, o HEVC propõe 35 formas de realizar a predição intra-quadro, onde 33 deles são modos direcionais e outros 2 não direcionais; já o codificador AV1 define 61 modos direcionais e 5 modos não direcionais. Sendo os modos direcionais eficientes para compressão de padrões que se repetem, como linhas, colunas, etc. Já os modos não direcionais tendem a ser focados para regiões com características específicas, como troca gradiente de cor ou regiões homogêneas.

Para interpolar o bloco sendo predito, a predição intra-quadro utiliza as amostras das bordas adjacentes de blocos vizinhos previamente codificados e reconstruídos. Observando a Figura 10, pode-se observar as amostras de referência do bloco vizinho, em cinza, sendo utilizadas para predizer o bloco atual. Já as flechas na figura indicam o ângulo que a interpolação está direcionando as amostras vizinhas.

Diferentes padrões de codificação definem modos de predição intra próprios que seguem a lógica descrita. Por exemplo, o padrão HEVC define a implementação dos modos Planar e DC juntamente de outros 33 modos angulares conforme descrito na Figura 11. Enquanto os modos angulares interpolam os pixels das bordas superior e esquerda vizinhas em determinados ângulos, o modo Planar realiza uma interpolação suave desses pixels para gerar uma transição homogênea. Já o modo DC calcula a média dos pixels de referência e preenche toda a região prevista com esse valor uniforme.

Diferentemente do HEVC, o codificador EVC (Essential Video Coding) (BROSS et al., 2021) determina a utilização de apenas quatro ângulos de interpolação além dos modos Planar e DC. Ainda, dependendo dos modos de predição dos vizinhos superiores e esquerdos, um código é gerado de forma adaptativa e atribuído ao modo de predição do bloco atual. Seguindo uma implementação de predição intra mais elaborada, o codificador VVC (Versatile Video Coding) (CHOI et al., 2020) aumenta para 65 o número de modos intra angulares conforme a Figura 12. Estes modos são divi-

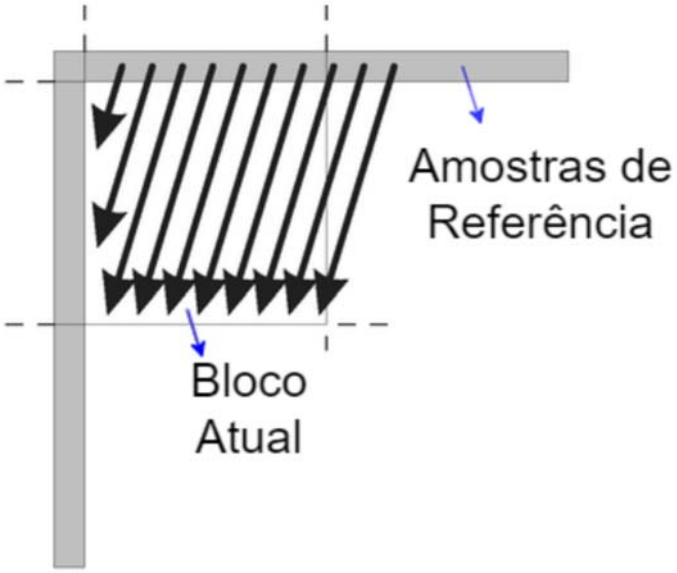


Figura 10 – Modos Angulares da Predição Intra
Fonte: Adaptado de Sulllivan et al. (2012).

didos em modos de angulação inteira e fracionária. Enquanto o filtro de suavização das amostras de referência é aplicado aos modos de inclinação inteira em blocos de luminância, o filtro de interpolação é aplicado aos modos de inclinação fracionária. Nos blocos de formato quadrado, é designado o mesmo número de modos angulares para os lados superior e esquerdo. Já nos blocos intra de formato retangular, que não existem no HEVC, mas são um elemento central do esquema de particionamento do VVC, um maior número de direções de predição intra é alocado no lado mais longo do bloco. Esses modos adicionais, chamados de Modos de Predição Intra de Ângulo Amplo (WAIP - *Wide Angle Adaptative Intra Predictors*), representam direções de predição com ângulos maiores que 45ř em relação aos modos horizontal ou vertical.

Por fim, para decidir qual modo ou direção utilizar, é implementada uma busca de modo de decisão baseada em *Rate Distortion Optmization* (RDO), responsável por escolher o modo que gera o melhor *tradeoff* entre distorção (perda de qualidade) e compressão (redução de *bitrate*) - i.e., minimiza o custo taxa-distorção (*RD Cost*). O custo RD do melhor modo intra é também comparado com o RD da predição inter-quadros para, desta maneira, o codificador poder escolher a melhor predição para o bloco atual entre todas as predições disponíveis.

Ao melhorar a performance da predição intra de um codificador, espera-se, por exemplo, que os resíduos gerados pela predição diminuam e, assim, o codificador de entropia precise de menos *bits* para representar os resíduos. Ainda, com uma predição eficiente, o codificador acaba empregando tamanhos de blocos maiores na codificação, precisando de menos blocos para comprimir a imagem ou vídeo e, consequentemente, precisando de menos *bits* de sinalização (para indicar os modos de

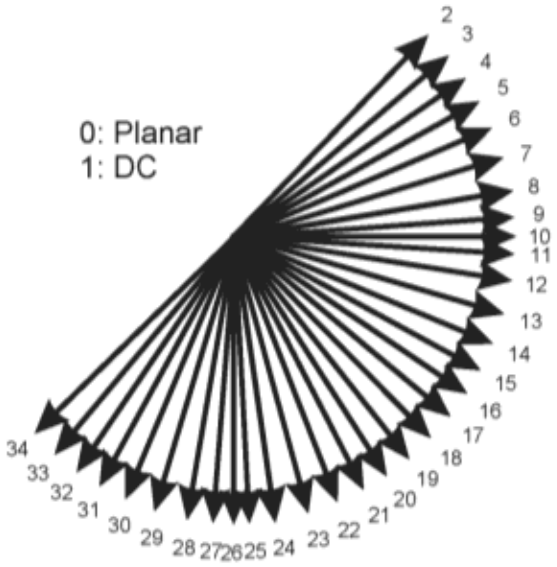


Figura 11 – Modos de Predição Intra do Codificador HEVC.
Fonte: Adaptado de (SULLLIVAN et al., 2012).

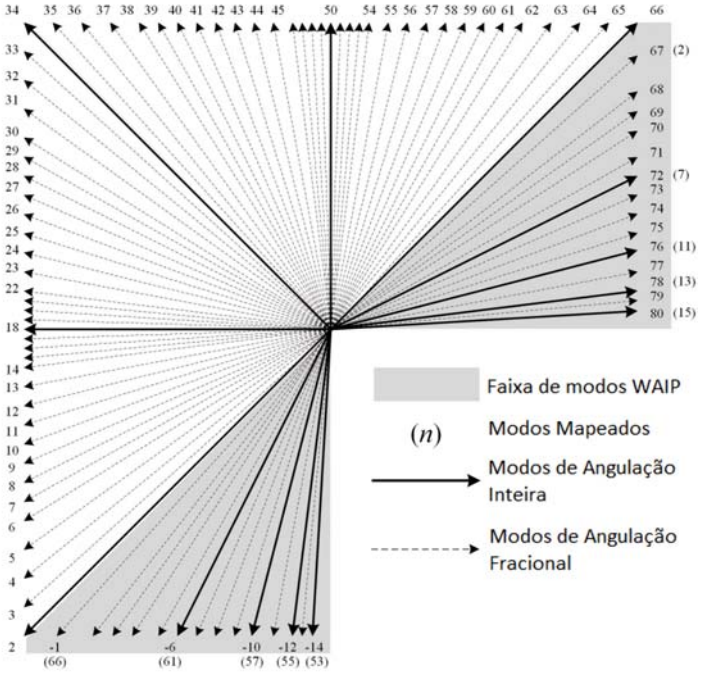


Figura 12 – Modos de Predição Intra do Codificador VVC
Fonte: Adaptado de Pfaff et al. (2021).

predição utilizados).

2.2.2.2 Predição Inter-Quadros

A predição inter-quadros busca explorar informações redundantes em dois ou mais quadros de um vídeo, podendo assim identificar a movimentação de uma amostra ao longo do vídeo. Segundo (GRELLERT, 2014), as ferramentas utilizadas nesta etapa possuem o maior custo computacional de toda a codificação.

Na predição inter-quadros, as informações de movimento do bloco são obtidas pelo processo de estimação de movimento (ME-*Motion Estimation*). A ME busca em um quadro já codificado (quadro de referência) o bloco mais similar ao bloco que está sendo codificado, pois, devido à alta similaridade, os blocos tendem a apresentar mais informações repetidas que podem então ser consideradas redundantes. A Figura 13 apresenta o bloco atual sendo predito através de blocos de referência em quadros previamente codificados com distâncias temporais de 2 e 4 quadros ($\triangle_t$). Após os blocos mais similares serem encontrados, um vetor de movimento $\triangle_x, \triangle_y$ indica o deslocamento ocorrido entre o bloco de referência e o bloco predito.

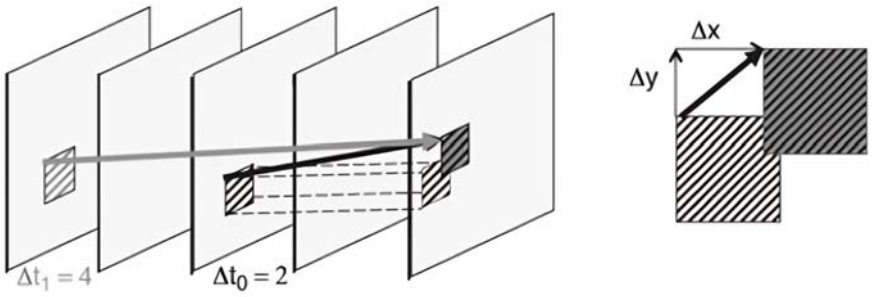


Figura 13 – ME Encontrando Blocos Similares Entre Quadros Temporalmente Vizinhos
Fonte: Adaptado de Sulllivan et al. (2012).

No padrão HEVC, o custo computacional da ME chega a 71% do custo computacional necessário para todo o processo de codificação (VIITANEN et al., 2012). Apesar do alto custo, ao utilizar as ferramentas de predição intra-quadro e inter-quadros conjuntamente, os codificadores conseguem atingir uma redução de *bitrate* de 10x a 100x (LAUDE et al., 2019).

Contudo, este ganho de eficiência de codificação se refere apenas a vídeos de testes baseados nas CTCs (*Common Test Conditions*) (BOSSEN, 2013), que define um conjunto de regras e configurações para padronizar testes, neste caso com vídeos, sobre o codificador. Como não existem codificadores que apliquem predições desenvolvidas com foco em LFs, estas estatísticas podem variar quando aplicadas a este novo formato de dado. Cabem, assim, estudos sobre codificação e predição especializados em LFs.

2.3 Codificação de Light Fields

Ao se tratar de *Light Fields*, as estruturas e características dos dados mudam e, consequentemente, as suas redundâncias e maneiras de explorá-las também. Analisando *Light Fields* estáticos, tem-se dois tipos de redundâncias: espaciais e angulares.

Apesar das predições intra-quadro dos codificadores de vídeo explorarem redundâncias espaciais, estas se apresentam de maneira diferente nos LFs. Para exemplificar, podemos observar um bloco de MIs (Micro Imagens) do LF *Bikes* na Figura 14 (considerando o formato *lenslet*). Nesta visualização, são exibidas todas as MIs de um bloco deste LF, onde cada MI, e.g., cada círculo, representa um conjunto de todos os 15×15 micro-píxeis capturados para um determinado pixel do LF. Desta maneira, diz-se que os micro-píxeis representam as informações angulares em $\{R_{x,y}, S_{x,y}\}$ de uma MI $\{X, Y\}$ do bloco.

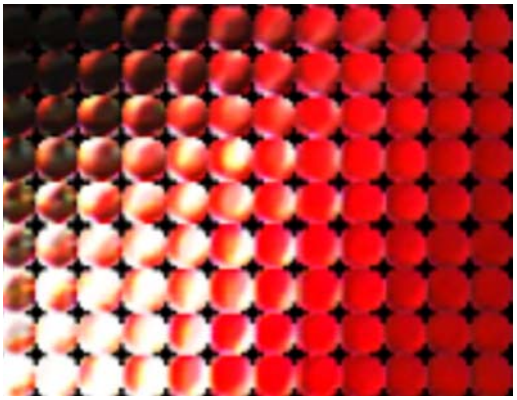


Figura 14 – MIs de uma região amplificada do *Light Field Bikes*
Fonte: Desenvolvido pelo autor.

É possível notar que várias das MIs possuem conteúdo similar, porém, a região também possui MIs discrepantes. Nesse caso, fica evidente a diferença entre explorar a redundância espacial em vídeos digitais e em *Light Fields* no formato *lenslet* onde, devido à presença das MIs, deve-se considerar a alta concentração de informações angulares. Frente a esta diferença, pode-se afirmar que enquanto a predição intra de vídeos digitais explora redundâncias espaciais copiando ou reaproveitando dados de regiões similares dentro de um mesmo quadro, em *Light Fields* essa predição poderia, por exemplo, ser feita copiando ou reaproveitando dados entre MIs similares. Alguns trabalhos da literatura já propõem preditores para *Light Fields* explorando esse tipo de redundância, como em Monteiro et al. (2017). Essas propostas de predições para *Light Fields* serão explicadas detalhadamente no capítulo de 3.

O outro tipo de redundância presente nos *Light Fields* é a redundância angular. Ao selecionarmos um micro-pixel em uma determinada posição $\{R, S\}$ de todas as MIs do LF, a vista capturada pelo determinado ângulo de R, S irá se formar. Conforme pode ser observado na Figura 15, as vistas adjacentes (a) e (b) são muito similares,

possuindo assim muita informação redundante. Vale ressaltar que esta é uma redundância que poderia ser explorada usando as predições inter-quadros dos codificadores de vídeo quando as vistas são interpretadas como quadros temporalmente distintos (essa organização é frequentemente denominada pseudo-vídeo).

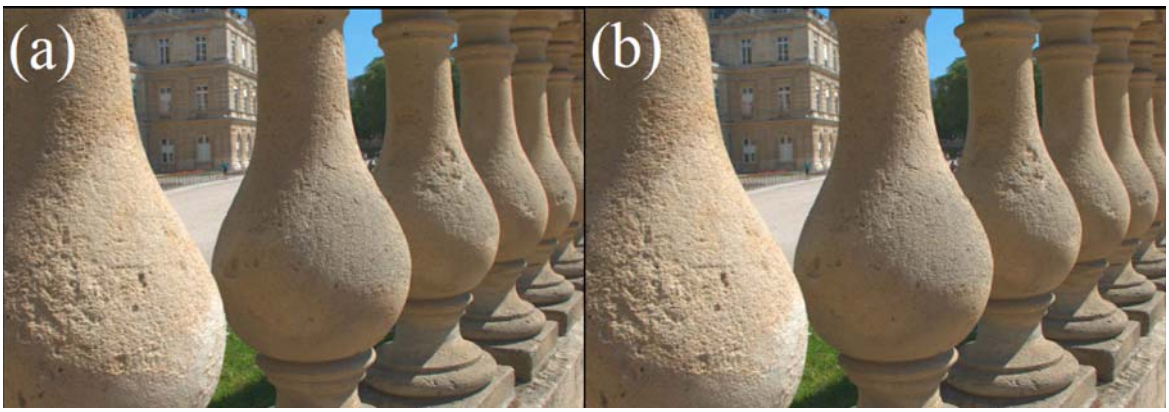


Figura 15 – Vista Central e Adjacente Posterior do LF *Stone Pillars Outside*
Fonte: Desenvolvido pelo autor.

Contudo, ao contrário de vídeos, onde os objetos se movimentam de maneira heterogênea (movimentos diferentes para diferentes objetos e para diferentes vídeos), o deslocamento das vistas de um LF é fixo e decorre da distância entre vistas. Isto faz com que o deslocamento dos objetos na cena entre uma vista e outra dependa somente de sua distância da câmera, tornando-os assim previsíveis. Pois, conforme mencionado anteriormente, seja o LF capturado por um *array* de câmeras ou por uma câmera *lenslet*, a distância entre as vistas capturadas deve estar bem calibrada e ser constante. Além disto, as vistas são dispostas sempre em um formato de matriz, ou seja, têm-se também um padrão definido sobre para que lado e o quanto as amostras se deslocam entre uma vista e outra. Sendo assim, pode-se concluir que os codificadores de vídeo, embora possam encontrar tais padrões e explorá-los, não o fazem da maneira mais eficiente possível, abrindo a possibilidade do desenvolvimento de predições especializadas em *Light Fields*.

Para investigar predições, conversão e codificação de *Light Fields*, *Point Cloud* e Hologramas, o comitê *Joint Photographic Experts Group* (JPEG), ISO/IEC JTC1/SC29/WG1, lançou o projeto JPEG Pleno (SCHELKENS et al., 2019). No JPEG Pleno Parte 2 (*Light Fields Coding*) tem-se 2 modos de codificação de LFs realizados por dois codificadores distintos, o modo de transformadas, que emprega o codificador MuLE, e o modo com predição, realizado por um codificador baseado no WaSP.

No modo de transformadas, especificado pelo diagrama de blocos da Figura 16, um LF é particionado em blocos e cada bloco é transformado individualmente. Esta transformação, assim como nos codificadores de vídeo, permite mudar os blocos do domínio espacial para o domínio das frequências. Os valores do bloco resultantes são chamados de coeficientes transformados. Em seguida, diferentemente dos codi-

ficadores de vídeo, o processo de quantização é realizado por *bitplanes*, no qual os coeficientes são decompostos em planos de bits e os bits menos significativos são zerados para reduzir a heterogeneidade dos valores. Essa redução é controlada pelo parâmetro λ (lambda), que regula a agressividade da redução, influenciando a taxa de compressão e a qualidade final. Por fim, os dados quantizados são codificados por um codificador de entropia. Este modo de transformadas é especialmente eficaz em LFs densos como os obtidos através de uma câmera *Lytro Illum* utilizados nesta tese e, portanto, sua performance é de interesse comparativo para os resultados desta tese.

Figura 16 – Diagrama de Blocos do MuLE
Fonte: Adaptado de Carvalho et al. (2018).

Já o codificador WaSP (*Hierarchical Warping, Merging, and Sparse Prediction*) possui 3 módulos na sua codificação. O primeiro passo realizado pelo WaSP é dividir as vistas de um LF em níveis hierárquicos. Os níveis hierárquicos mais baixos servem como quadros de referência para as hierarquias mais baixas. Desta maneira, os quadros com hierarquia $H_l = 2$ onde H_l representa o nível de hierarquia (*Hierarchic Level*) serão decodificados pelas vistas definidas como $H_l = 1$. Já as vistas definidas com $H_l = 3$ ou superior, podendo chegar até $H_l = 6$, podem ser decodificadas com vistas de até dois níveis inferiores na hierarquia. Em seguida, o primeiro módulo utiliza o codificador de imagens estáticas (SKODRAS; CHRISTOPOULOS; EBRAHIMI, 2001) para codificar as vistas do nível hierárquico mais baixo. Apesar de o WaSP ser definido e desenvolvido utilizando o JPEG 2000 nesta etapa, qualquer outro codificador pode ser utilizado no lugar deste, abrindo possibilidades para experimentos e avaliações. Posteriormente, o segundo módulo utiliza o JPEG 2000 para codificar também os mapas de profundidade inversos de N vistas subsequentes que, por sua vez, vão ser utilizadas de referência para a codificação das próximas hierarquias.

Por fim, o terceiro módulo sintetiza vistas para realizar predições em matrizes residuais esparsas. Primeiramente, vistas intermediárias são reconstruídas através da interpolação de várias vistas de referência vizinhas através de uma compensação de deslocamento denominada como *warping*. Em seguida, versões interpoladas das vistas são fundidas para obter uma versão predita da vista vizinha e, em seguida, os resíduos entre a imagem original da vista e sua versão predita são codificados. Ressaltamos que, ao contrário do modo de transformadas, este modo de "predição"(em vista da síntese predita das vistas) possui maior eficiência para LFs esparsos, como os obtidos por *arrays* de câmeras.

Light Fields também podem ser representados fixando-se os eixos $\{V, X\} e \{Y, U\}$,

gerando um formato de imagem denominado como *Epipolar Plane Image* (EPI), representada na Figura 17. Esta estrutura é muito útil para identificar a profundidade entre objetos no LF. Assumindo que nenhum objeto que aparece na imagem seja oculto em alguma das SAIs, ele deve aparecer ligeiramente deslocado devido ao efeito de *paralaxe*. Isso faz com que EPIs sejam, essencialmente, várias retas, de tal forma que quanto menor a inclinação, mais distantes os objetos estão (HONAUER et al., 2016). Este conceito já foi observado na década de 80 por Bolles; Baker; Marimont (1987), porém para observar os fenômenos de *paralaxe* em fotos estáticas capturadas com ligeiros deslocamentos na posição da câmera. A estrutura de EPI tem mostrado diversas vantagens para a aplicação de redes neurais convolucionais para codificação e reconstrução de matrizes esparsas (similar ao WaSP), geração de mapas de profundidade, entre outras aplicações que serão detalhadas no Capítulo 3.

Figura 17 – Imagem EPI do LF *Foutain & Vincent* 2
Fonte: Desenvolvido pelo autor.

2.4 Redes Neurais

Na década de cinquenta, em uma primeira tentativa de simular o funcionamento de um neurônio, Rosenblatt (1958) propôs o conceito de *Perceptrons*. Estes neurônios artificiais atuam como classificadores lineares (binários) considerados como uma rede neural de camada única.

Como pode ser observado na Figura 18, um perceptron possui um vetor de entradas i_x e gera uma saída O. As entradas inicialmente são multiplicadas por um peso w_x e posteriormente somadas juntamente a um bias b. Os pesos são responsáveis por determinar o grau de importância da entrada para a classificação e, em conjunto com o valor de bias, são responsáveis por definir a localização do hiperplano no espaço das entradas (MICHALSKI; CARBONELL; MITCHELL, 2013). O resultado O pode ser definido alternativamente pela equação 4. Por fim, a soma das entradas resultante é submetida a uma função de ativação que retornará a classificação binária como 0 ou 1.

$$y = f\left(b + \sum_{k=1}^{n} x_k \cdot w_k\right) \tag{4}$$

A função de ativação em geral possui alguma não linearidade, alguns exemplos de funções de ativação são $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, $\sigma(x) = \frac{e^x}{1+e^x}$ e $relu(x) = max(0, x)$ (AGARAP, 2018) cujas curvas não lineares podem ser visualizadas na Figura 19. Existe ainda a `PReLU` (*Parametric ReLU*), definida como $prelu(x) = \max(0, x) + a \cdot \min(0, x)$,

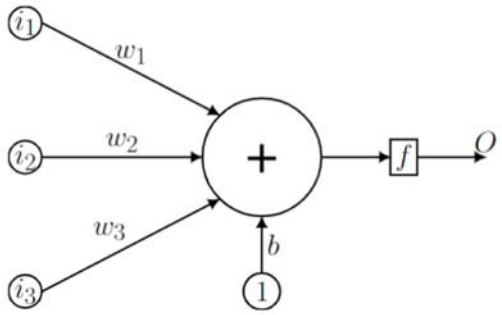


Figura 18 – Diagrama de um Perceptron
Fonte: Adaptado de Taulli (2020).

onde a é um parâmetro a ser aprendido (HE et al., 2015).

Com a saída obtida pela rede neural, neste caso o perceptron, e o resultado esperado, podemos utilizar alguma medida de erro (*loss*) como, por exemplo, a MSE (*Mean Squared Error*) para determinar o quão distante da resposta correta o perceptron classificou a entrada. O MSE é especialmente útil ao considerarmos redes neurais artificiais, ou seja, quando há mais de um perceptron em múltiplas camadas, pois este leva em consideração a média do quadrado dos erros de cada um dos perceptrons da arquitetura.

Tendo uma medida de erro, podemos utilizar técnicas de minimização para otimizar a atualização dos pesos das entradas. A equação (5) descreve o método da descida de gradiente (*Gradient Descent*), também conhecido como subida de encosta e proposto por Lecun et al. (1989). Esta técnica é essencial para a atualização e ajuste fino dos pesos de redes neurais. Vale observar que uma análise similar é realizada por Rumelhart; Hinton; Williams (1985).

$$w'_n = w_n - \eta \cdot \frac{\partial e}{\partial w_n} \tag{5}$$

Na descida de gradiente, o $\eta > 0$ é uma constante referida como taxa de aprendizado (*learning rate*) que, ao multiplicar o peso de saída w_n (antigo valor do peso),

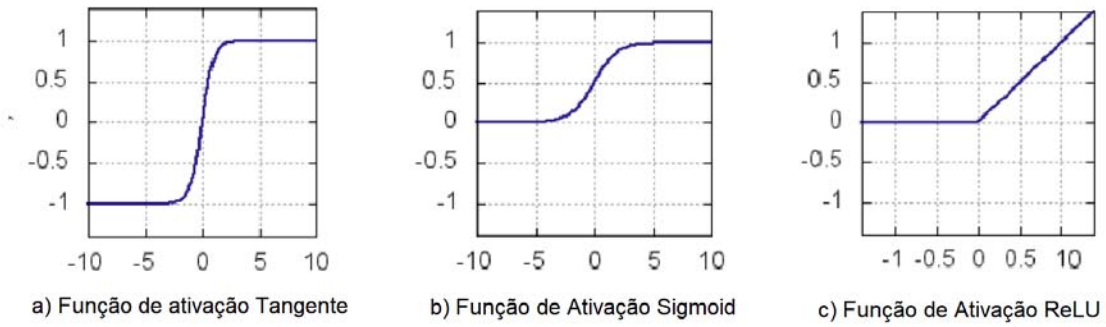


Figura 19 – Gráficos do comportamento de diferentes funções de ativação.
Fonte: Adaptado de Pinheiro (2019).

determinará o peso da sua influência no novo valor do peso, determinado por w'_n. Sendo que ∂e é a derivada do erro do perceptron pelo peso w_n. Por exemplo, se estivermos utilizando a métrica MSE sob a saída de um perceptron, temos o erro como $e = \texttt{MSE}\left(f\left(b + \sum_{k=1}^{n} x_k \cdot w_k\right), y_l\right)$. Em vista de que estamos tratando de saídas binárias, $n = 1$ e $\texttt{MSE}(a, b) = (a - b)^2$, se aplicarmos a derivada parcial em função de w_n sobre o erro, obtemos $\frac{\partial e}{\partial w_n} = 2 \cdot x_k \cdot f'\left(b + \sum_{k=1}^{n} x_k \cdot w_k\right) \cdot e$. Para atualizar o viés b, pode-se considerar que ele é um peso para uma entrada que sempre é um.

Redes neurais artificiais de *perceptrons* podem ser compostas de diversas maneiras, uma delas é a chamada *fully connected* ou *multilayer perceptron*, na qual temos um número determinado de camadas dispostas uma após a outra. Cada camada possui um número de perceptrons arbitrário, na qual as saídas y_k de todos *perceptrons* de uma camada são entradas x_k para todos *perceptrons* da camada imediatamente posterior, tal como está representado na Figura 20 (RUCK; ROGERS; KABRISKY, 1990).

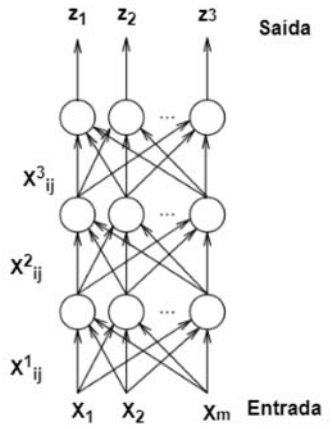


Figura 20 – Rede Neural *fully connected*
Fonte: Adaptado de Ruck; Rogers; Kabrisky (1990).

Nestas redes *fully connected* pode-se avaliar o gradiente pela propagação da *loss* entre as camadas da rede. Pois ao ser transmitida de um perceptron para outro, a *loss* será multiplicada pelos pesos dos perceptrons da camada seguinte.

Apesar destas arquiteturas de redes neurais possuírem a capacidade para resolver diversos problemas de classificação e regressão, quando estes problemas envolvem imagens, é indicado aplicar o conceito de convolução às imagens, sejam elas vídeos aplicados à visão computacional em tempo real ou análise de fotografias estáticas (O'SHEA; NASH, 2015).

2.5 Redes Neurais Convolucionais

Uma operação comum para processamento de sinais é a convolução, cuja é aplicada a dois sinais e/ou sistemas $x[n]$ e $w[m]$ que resulta em um sinal $y[n+m-1]$ tal como está definido na equação (6) (BURRUS; PARKS, 1985). A convolução pode ser vista como a aplicação de um filtro a uma entrada que resulta em uma ativação. A aplicação deste mesmo filtro em uma imagem pode ser repetida com deslocamentos ao longo de toda a imagem, gerando um mapa de ativações denominado *feature map* ou mapa de características. Este mapa indica as localizações e a importância das características detectadas em uma entrada.

$$y(n) = \sum_{m=0}^{M-1} h(m)w(n-m) \tag{6}$$

É possível construir uma rede neural na qual as saídas são $y[k]$, aplicadas à função de ativação, as entradas são $x[k]$ e os pesos são $w[k]$. Esta arquitetura de rede é denominada *Rede Neural Convolucional* (RNC). Podemos estender essas redes para mais de uma dimensão - assim como fazemos com a convolução. Aqui, w representa o núcleo (*kernel*) da convolução, o que dita o comportamento do filtro aplicado.

Uma das características das RNCs é que os pesos $w[k]$ são compartilhados, ou seja, para calcular qualquer $y[n]$, são utilizados os mesmos pesos, porém sobre entradas diferentes. Isso implica que um deslocamento na entrada causará um deslocamento na saída e, desta forma, diz-se que estas são invariantes a deslocamento (*shift invariant*) (LECUN; BENGIO et al., 1995).

A função de uma convolução é gerar mapas de características (também chamados de *feature maps*) que contenham informações que podem ser compreendidas pela RNC. A Figura 21 demonstra o comportamento de uma convolução com *kernel* de tamanho 3×3 representado pelos quadrados mais escuros. Neste exemplo, o mapa de características, representado pelos quadrados verdes, é extraído a partir de multiplicações da matriz original, representada pelos quadrados azuis. Desta maneira, a posição sendo extraída (quadrado verde-escuro) é calculada a partir das nove referências (quadrados azuis-escuros). Ao alterar o tamanho de *kernel*, a convolução considera uma região maior ou menor para determinar o valor de cada posição na matriz sendo extraída (mapa de características). Diz-se portanto que, ao aumentar o *kernel* de uma convolução, se está aumentando o campo receptivo da convolução. Ainda, um fator importante em uma convolução é o seu *stride*, i.g., quantas amostras o *kernel* se deslocará para calcular a próxima saída da convolução. Portanto, assim como no exemplo, caso o *stride* de uma convolução seja 2, o tamanho de sua saída terá a metade de suas dimensões.

Ademais, além do processo de convolução, também utiliza-se a técnica de con-

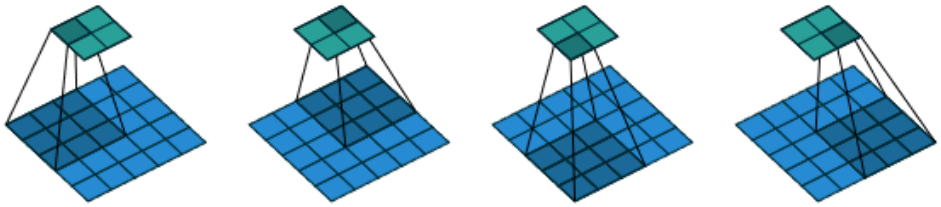


Figura 21 – Comportamento da Convolução com *stride* 1
Fonte: Figura adaptada de Dumoulin; Visin (2016).

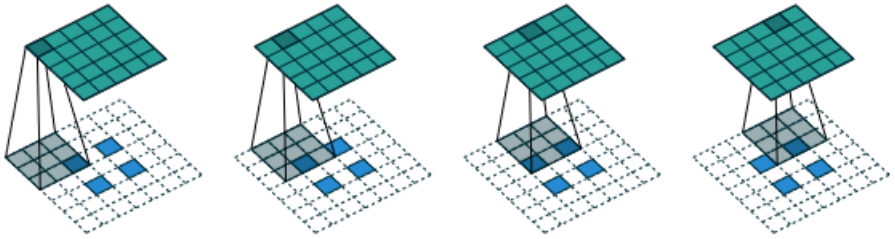


Figura 22 – Comportamento da Convolução Transposta com *stride* 1
Fonte: Figura adaptada de Dumoulin; Visin (2016)

volução transposta, onde as dimensões da entrada são incrementadas ao invés de reduzidas. Esta técnica é interessante em determinados casos pois, em geral, esta não descarta os elementos próximos das bordas das entradas, onde nem todos os pesos do núcleo de convolução são aplicados, o que ocorre usualmente nas camadas convolucionais (SHI et al., 2016). Ainda, inversamente à convolução, caso o *stride* de uma convolução transposta seja 2, as dimensões da saída serão o dobro da entrada. Unindo ambas as técnicas, possibilita-se que as RNCs extraiam informações dos dados de entrada enquanto manipulam as suas dimensões, o que permite também que a saída da RNC tenha as dimensões desejadas. Existe também a técnica de deconvolução, que possui comportamento similar ao da convolução transposta, contudo, seu foco é desfazer a convolução. Logo, a deconvolução não é uma operação em específico, mas pode ser um conjunto de operações que visam apenas reverter com precisão a convolução. Deste modo, enquanto a deconvolução é aplicada para principalmente para estaurar imagens degradadas por algum filtro, a convolução tranposta é utilizada em redes focadas em gerar imagens como em autoencoders.

Uma das arquiteturas de redes neurais convolucionais amplamente aplicada a diversos cenários é a de autoencoder. A Figura 23 apresenta a arquitetura autoencoder mais simples possível. Conforme pode ser visto na figura, similarmente a um codificador de vídeo, um autoencoder também possui dois lados, um lado que codifica a entrada, chamado de encoder, e um lado que decodifica a informação em uma saída, chamado de decoder. O encoder, que recebe inicialmente uma imagem, neste caso de tamanho $H_{in} \times W_{in}$ e RGB (espaço de cores com 3 canais), é tipicamente consti-

tuído por uma ou mais camadas de convolução seguidas de funções de ativação. Por simplicidade, neste exemplo omitimos detalhes sobre função de ativação e *kernels*. As dimensões de altura e largura da camada inicial do encoder dependem das dimensões da entrada, já o número de canais pode assumir qualquer valor acima de zero. Um maior número de canais tende a manter mais informações sobre a imagem ao longo da rede e assim melhorar sua performance ao custo do uso de memória e processamento. Ainda, um número muito alto de canais pode levar a rede a um *overfitting*, ou seja, um ajuste excessivo da rede aos dados de treinamento, o que diminui sua capacidade de generalização. Assim, uma boa prática de para equilibrar o número de canais, performance e custo, é começar com um número baixo como por exemplo 32, e duplicar este número a cada camada do encoder. Ainda, esta estratégia pode e deve ser revisada conforme o número de camadas convolucionais aumenta (BANK; KOENIGSTEIN; GIRYES, 2023). As dimensões de saída de uma camada convolucional são diretamente relacionadas ao *stride* conforme mencionado anteriormente. Ou seja, se o *stride* for 1, as dimensões de saída da camada convolucional serão iguais às da entrada. Caso seja 2, o tamanho da saída também é dividido por 2. Um *stride* muito grande pode reduzir a imagem muito rapidamente, fazendo assim com que o processo de convolução elimine muitas informações. Desta maneira, o valor de *stride* mais comumente utilizado é de dois, que também é o menor possível quando se visa diminuir as dimensões da entrada. Por simplicidade, não entraremos em detalhes, mas note que o tamanho do *kernel* pode gerar perda de 1 ou mais amostras que comumente são preenchidos com *padding*.

Ao fim, o encoder gera um espaço latente, que por sua vez é um conjunto de *feature maps* que apresenta informações sobre a entrada de maneira que a rede consiga reconstruí-la ou até mesmo alterá-la como desejar no lado do decoder. As dimensões de altura e largura dos *feature maps* do espaço são normalmente pequenas devido às diminuições realizadas pelas camadas convolucionais com *strides*. Este espaço latente pode ser interpretado como uma versão comprimida da entrada, assim como um *bitstream* gerado por um codificador de vídeo. Assim como a maioria dos modelos de redes neurais (e um *bistream*), seu estado pode ser armazenado, comprimido (via rar ou qualquer outra ferramenta de compressão entrópica) e descomprimido antes de ser enviado para o decoder.

Para decodificar o espaço latente, o decoder normalmente deve redimensioná-lo até as dimensões desejadas para a saída. Duas estratégias são mais comumente utilizadas para isso. Primeiramente, pode-se utilizar uma convolução transposta com um passo de *2* que, inversamente a uma camada convolucional, irá duplicar o tamanho dos *feature maps* e já extrair informações deles no processo. A outra alternativa é utilizar uma camada de *upsampling* para redimensionar os *feature maps* seguida de uma camada convolucional com *stride* 1 para extrair as informações sem alterar

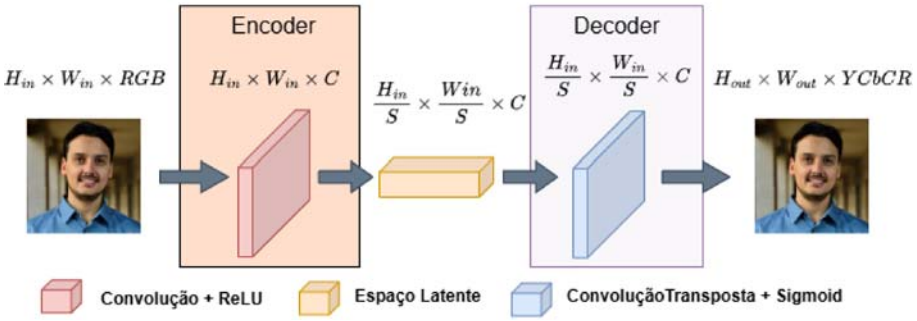


Figura 23 – Exemplo de arquitetura AutoEncoder simples.
Fonte: Desenvolvida pelo autor.

as dimensões. Vale evidenciar que, como neste exemplo a arquitetura possui apenas uma camada convolucional no encoder e uma convolucional transposta no decoder, caso ambas tenham o mesmo *stride*, as dimensões de saída serão iguais às de entrada, fazendo assim com que a rede atue simplesmente como um conversor RGB para YCbCR. Ainda, caso a intenção desta arquitetura seja realizar um *upscale* da imagem, poderia-se utilizar um *stride* de 1 na camada do encoder e de 2 na camada do decoder. Por fim, o número de canais da saída determinará o espaço de cores que o autoencoder gerará. Como a última camada utiliza uma *sigmoid* para gerar sua saída, esta será constituída de números entre 0 e 1 (conforme discutido na Seção anterior). Logo, caso $C_{out} = 1$, a imagem da saída estará em escala de cinza em luma (Y), caso $C_{out} = 3$, teria-se uma saída em YCbCr.

Outro exemplo de arquitetura RNC amplamente empregada é a *LeNet*, proposta em (LECUN et al., 1989), cuja arquitetura pode ser vista na Figura 24. Em sua primeira etapa, uma camada convolucional é aplicada e, em seguida, é realizado o processo de *pooling*, que é uma operação feita para reduzir o tamanho dos dados ao longo das camadas sem a utilização de *strides* nas convoluções. Posteriormente, há mais duas camadas convolucionais que linearizam os dados para serem então inseridos em uma rede neural *fully connected*.

Inicialmente, RNCs foram empregadas para realizar reconhecimento de caracteres (FUKUSHIMA, 1988). Problema este que envolve atividades de classificação onde, a partir de uma imagem estática como entrada, classifica-se ela dentro de um conjunto finito de valores como, por exemplo, caracteres de um alfabeto. Vários problemas apresentam este formato, contudo, alguns requerem ainda que se estime uma medida contínua y. Estes problemas são definidos como problemas de regressão (WARNER; MISRA, 1996) e também podem ser abordados através de RNCs.

É correto afirmar que a predição em uma imagem é um problema de regressão, pois deve-se estimar uma série de valores (RGB dos píxeis) que formam o bloco a ser predito. Um exemplo da aplicação de RNCs em problemas de regressão é a sua aplicação no pré-processamento de imagens médicas, onde utilizam-se RNCs para

traçar os limites das células de uma das camadas da córnea, tarefa que, até então, era feita manualmente conforme relatado em Zhang et al. (1991).

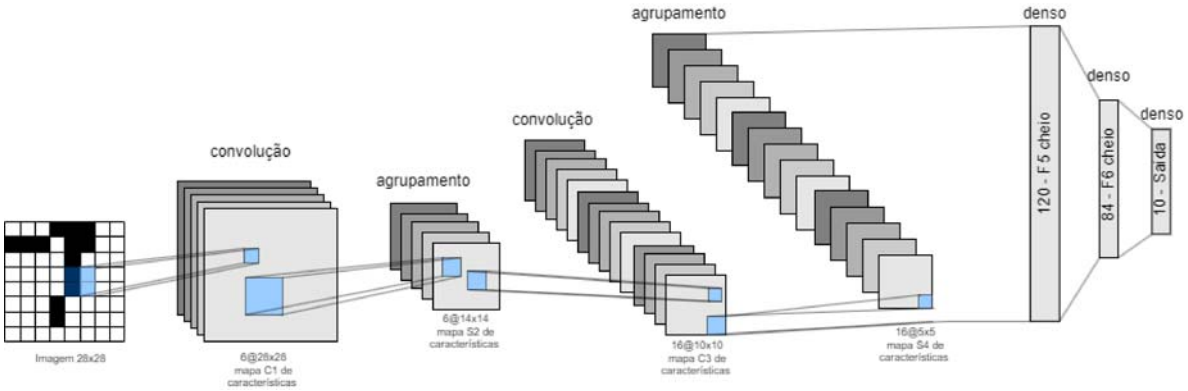


Figura 24 – *LeNet* proposta para o reconhecimento de caracteres por Lecun et al. (1989)
Fonte: Adaptado de Siebert (2022).

Um dos fatores que limitou o desenvolvimento de redes neurais no passado foi a baixa disponibilidade de recursos computacionais, especialmente para realizar o treinamento das redes. Pois esta etapa demanda a execução do mesmo algoritmo diversas vezes sobre um conjunto grande de imagens (*dataset*) que podem, facilmente, passar de dezenas de milhares de imagens. Com o crescimento exponencial no poder de processamento dos computadores nas últimas décadas, tornou-se possível o desenvolvimento de RNCs maiores e também a utilização de *datasets* consideravelmente mais ricos. Outro fator importante para a utilização destas arquiteturas é a utilização de *Graphics Processing Units* (GPUs), pois estas permitem um número elevado de ciclos de treinamento simultâneos, o que tornou o processo várias vezes mais rápido (OH; JUNG, 2004).

Ao se aplicar o conceito de retro-propagação mencionado para ajustar os pesos em redes neurais convolucionais com múltiplas camadas (M-RNC), pode-se ter um aumento no erro dos treinos e testes caso a rede fique muito profunda. Isso ocorre devido ao comportamento denominado *gradient vanish*, onde o valor das funções de ativação tendem a diminuir exponencialmente ao longo das camadas, podendo impedir as funções de serem ativadas e, consequentemente, impedir completamente a rede de aprender após um determinado número de camadas. O comportamento do *gradient vanish* é apresentado na Figura 25. Houveram múltiplas propostas de técnicas para resolver este problema, dentre elas Srivastava; Greff; Schmidhuber (2015) propôs as *Highways Neural Networks*, ou, Redes Neurais em Rodovia.

2.5.1 HighwayNets e ResNets

Para melhor propagar os valores das funções de ativação e evitar, ao menos parcialmente, o problema do *gradient vanish*, Srivastava; Greff; Schmidhuber (2015) propôs pular uma ou mais camadas ao passar os pesos, criando "caminhos" (análogos a ro-

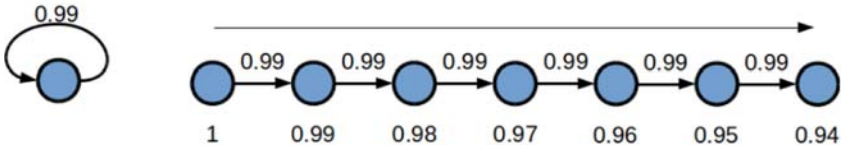


Figura 25 – Fenômeno *Gradient Vanish* ao longo de Camadas
Fonte: Adaptado de Bayer (2017).

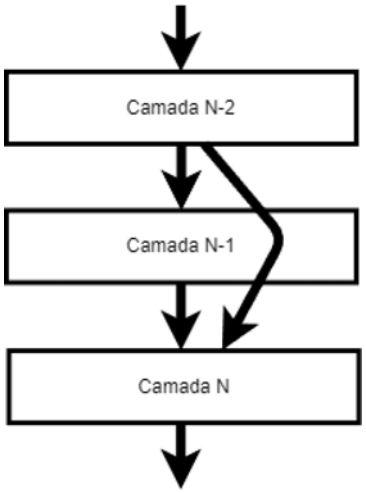


Figura 26 – Exemplo de transmissão de peso através de uma conexão *Highway*
Fonte: Desenvolvida pelo Autor.

dovias) que facilitam com que estes cheguem às camadas mais profundas. A Figura 26 representa uma *Highway* transmitindo o peso de uma camada N-2 diretamente para a camada N, ignorando a camada N-1.

Ao passar estes pesos para camadas mais profundas, a *HighwayNet* possui um atributo parametrizável denominado de "portão" (*gateway*) que determina quanta informação pode ser transmitida nestes caminhos. He et al. (2016) defende que deixar as camadas realizarem o mapeamento destes resíduos transmitidos diretamente é mais eficiente que parametrizá-los. Assim, He et al. (2016) propôs as ResNets, uma variação da *HighwayNet* que não possui portões, mas sim o conceito de *skip-connections* ou *residual-connections* (TARG; ALMEIDA; LYMAN, 2016). Nesta técnica, além de os resíduos de uma camada serem transmitidos para a próxima camada, estes também são concatenados com as saídas de camadas mais profundas, deixando assim a própria rede determinar e refinar quanta informação será transmitida (JASTRZĘBSKI et al., 2017).

Ademais, um subtipo de *Resnets* é a arquitetura de rede neural em U, denominada de *UNet*, que foi desenvolvida por Ronneberger; Fischer; Brox (2015), originalmente, para a segmentação de imagens médicas. A arquitetura consiste de um *autoencoder* com camadas convolucionais ajustadas de modo que elas reduzam as dimensões espaciais, seguidas de camadas convolucionais transpostas (ou deconvolucionais) que fazem o inverso, ou seja, aumentam as dimensões espaciais. Logo, o resultado é um

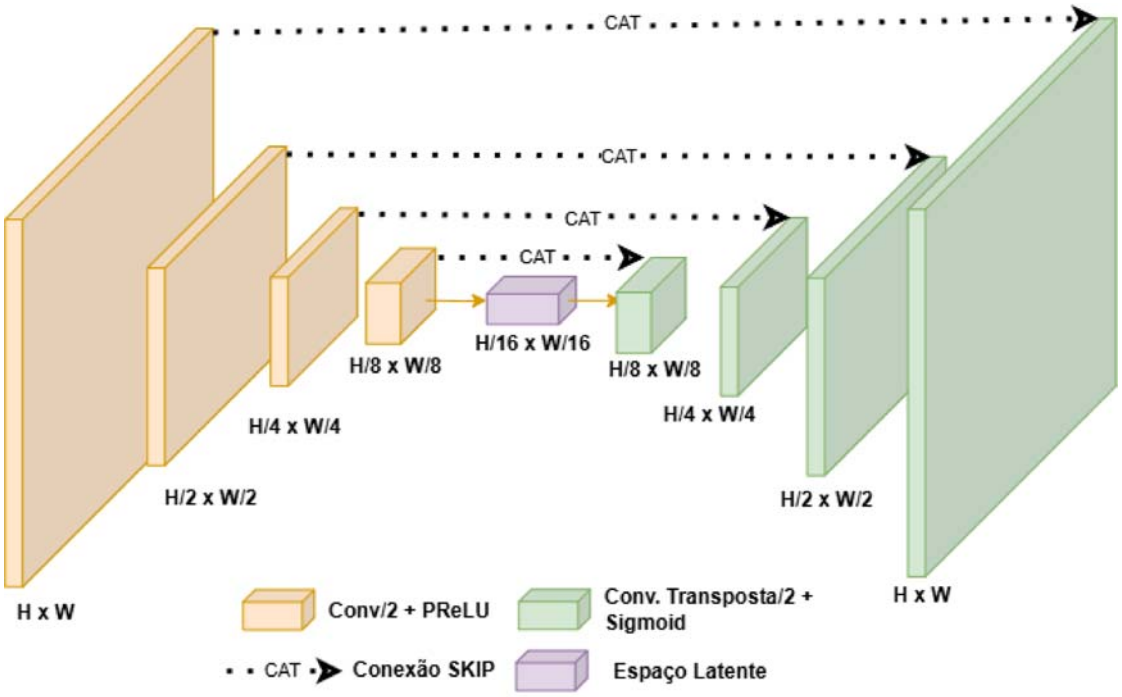


Figura 27 – Arquitetura de uma Rede Neural UNet
Fonte: Desenvolvido pelo autor.

autoencoder simétrico na qual para cada camada do *encoder*, tem-se uma camada no *decoder* com as suas respectivas dimensões, tal como pode ser observada na Figura 27. Esta simetria permite com que haja conexões *skip* entre as camadas de mesmo tamanho entre o *encoder* e o *decoder*, mantendo assim informações estruturais das camadas convolucionais ao longo de toda a rede.

2.6 Técnicas de Pruning em Redes Neurais

Em cenários de predição intra para *light fields*, onde há uma grande quantidade de dados espaciais e angulares a serem processados, o *pruning* é particularmente útil para aumentar a eficiência das redes sem comprometer a qualidade da compressão. Isso se deve ao fato de que redes neurais profundas precisam ser superparametrizadas para aprenderem, mas muitos parâmetros se tornam irrelevantes. Em alguns casos o *pruning* pode levar a uma melhora na capacidade de generalização da rede. Existem duas abordagens principais para a aplicação de *pruning*: o *pruning* estruturado e o não estruturado, cada uma com características específicas que afetam a arquitetura e o desempenho da rede de diferentes maneiras.

Em Liao et al. (2023); Chen et al. (2020) são propostos três níveis de *pruning* possíveis em redes neurais convolucionais. Para melhor entendermos como é realizado cada *pruning*, a Figura 28 demonstra o que é podado em cada um destes níveis em uma rede simplificada, que possui apenas uma camada convolucional com *stride* 1 seguida de outra camada convolucional de mesmo tamanho, porém com o dobro de

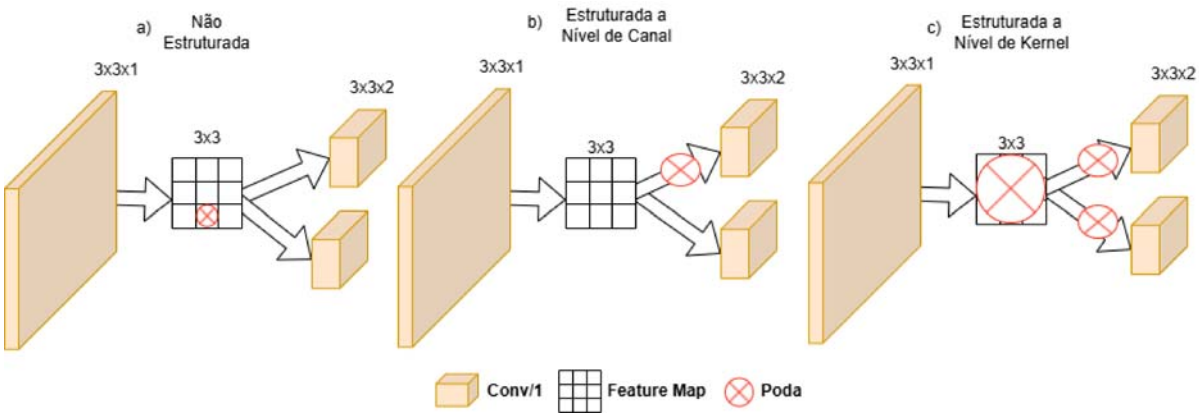


Figura 28 – Comparação entres os três níveis de *pruning* em redes neurais convolucionais.
Fonte: Desenvolvido pelo autor.

canais (comportamento padrão discutido na Seção 2.5). Note que, como a primeira camada possui dimensões $3 \times 3 \times 1$, apenas um *feature map* com tamanho 3×3 será enviado para os dois canais da próxima camada.

O *pruning* não estruturado (Figura 28 a) remove (zera) pesos individuais próximos de 0 nos *features maps* de uma camada. Esse tipo de *pruning* oferece maior flexibilidade na remoção de parâmetros, permitindo que a rede se adapte de forma granular, removendo apenas as conexões menos relevantes, i.e., pesos muito próximos de zero. No caso da predição intra de *light fields*, o *pruning* não estruturado pode ser eficaz para melhorar o desempenho da rede ao longo de treinamentos maiores, enquanto evita o *overfitting*. Entretanto, essa abordagem pode levar a redes com estruturas esparsas, o que nem sempre resulta em melhorias diretas no tempo de inferência ou implementação em hardware, pois as operações ainda precisam lidar com matrizes esparsas que o hardware não consegue otimizar.

Por outro lado, o *pruning* estruturado (ANWAR; HWANG; SUNG, 2017) (Figura 28 b) remove (zera) partes inteiras da arquitetura da rede, como filtros, canais ou camadas, resultando em uma rede mais compacta e eficiente em termos de cálculo. Essa técnica é especialmente adequada para aplicações de *light fields*, onde o volume de dados e a quantidade de cálculos podem ser altos. Ao remover estruturas inteiras, o *pruning* estruturado também facilita a implementação em hardware, uma vez que as operações se tornam mais regulares, permitindo maior paralelismo e otimizações de memória. Além disso, essa técnica mantém a densidade da rede, o que pode levar a melhorias significativas no tempo de inferência em comparação com o *pruning* não estruturado. Contudo, por ser um *pruning* mais agressivo (que remove estruturas maiores), esta técnica tende a não prover ganhos em eficiência.

Ainda, é possível realizar o *pruning* a nível de *kernel*. Neste *pruning* ainda mais agressivo, o *feature map* é zerado por completo, fazendo assim com que não seja enviado para nenhum dos canais da camada posterior. Logo, enquanto o *pruning* a nível de canal é interessante em situações que o *feature map* será relevante para um

canal e não para outro, o *pruning* a nível de *kernel* é interessante quando o *feature map* não é relevante para nenhum dos canais.

Os três níveis de *pruning* apresentam diferentes vantagens e desvantagens para redes neurais. O *pruning* não estruturado pode ser mais eficaz para preservar a precisão em redes altamente complexas, onde a remoção de conexões individuais não compromete a capacidade da rede de capturar as nuances dos dados. Já o *pruning* estruturado a nível de *kernel* oferece uma melhoria direta no tempo de inferência, sendo especialmente útil em sistemas de codificação de vídeo em tempo real e em FPGAs, onde a redução do custo computacional é crucial. Ainda, o *pruning* estruturado a nível de *kernel* pode prover ainda mais reduções em tempo de inferência. Contudo, é uma técnica demasiadamente agressiva que nossos experimentos mostraram perdas impraticáveis na eficiência do preditor.

2.7 Sumário

O presente capítulo apresentou os principais conceitos necessários para uma boa compreensão desta tese. Inicialmente, na Seção 2.1 discutimos o que são LFs e suas formas de captura, pois entender o tipo de dado que estamos comprimindo é primordial para entender o desafio de comprimi-lo. A seção 2.2.1 apresentou os princípios da codificação de vídeo, assim como suas predições intra-quadro e inter-quadros. A compreensão destas ferramentas, sua importância na codificação e onde elas falham ao predizer *light fields* é necessária para entender o funcionamento das arquiteturas que serão discutidas no capítulo 4. Ainda, estas informações também facilitam a compreensão de muitos trabalhos discutidos no próximo capítulo e suas diferenças perante esta tese. Nesta seção também serão abordadas as diferenças entre as predições intra dos codificadores EVC, HEVC e VVC, o que é essencial para entender as diferenças em implementações e resultados apresentados nos capítulos 5 e 6. Posteriormente, na Seção 2.3 discutimos o funcionamento dos codificadores de *light fields* propostos pelo JPEG Pleno. Embora esta tese não abranja tais codificadores, é importante saber suas diferenças com codificadores de vídeo e os implicações de tais diferenças.

Este capítulo ainda abordou os conceitos fundamentais sobre redes neurais na Seção 2.4, descrevendo brevemente o funcionamento desde neurônios, pesos, funções de perda, *learning rate* até uma arquitetura básica. Estes conceitos, embora simples, são de extrema importância para entender a análise exploratória e as técnicas avaliadas descritas no capítulo 4. Já para viabilizar uma melhor compreensão do funcionamento e o que ocorre em cada camada das arquiteturas propostas, a seção 2.5 descreveu os conceitos sobre redes neurais convolucionais e seu funcionamento, assim como a diferença entre convolução e convolução transposta. Ainda, esta se-

ção também apresentou os conceitos de *kernel* e campo receptivo, que também são explorados na nossa análise exploratória desenvolvida no capítulo 5.

No capítulo 5 também serão avaliadas diversas arquiteturas de RNCs considerando diferentes estratégias topológicas. Para compreender as diferenças de eficiência e implementação destas arquiteturas, a seção 2.5 descreveu o conceito de conexões *skip* e esclareceu as principais diferenças entre arquiteturas *highway*, residuais e *Unets*. Por fim, a seção 2.6 definiu o conceito de *pruning* não estruturado, que providencia ganhos de eficiência, mas não de inferência, e de *pruning* estruturado que, apesar de mais agressivo, apresenta, principalmente, ganhos de inferência explorados pelo hardware. Ambos os conceitos serão explorados para aperfeiçoar as arquiteturas propostas e essenciais para entender os resultados discutidos na Seção 5.2.6.

A seguir, discutiremos os trabalhos científicos que aplicam vários dos conceitos discutidos neste capítulo e técnicas que são relevantes para esta área de pesquisa. Ao longo do capítulo, os trabalhos exploram desde os conceitos de *light fields* e codificadores de sinais inicialmente discutidos neste capítulo, até redes neurais convolucionais profundas para predição intra-quadro. Ainda, trabalhos demonstrando a eficiência das técnicas de *pruning* em diferentes contextos também são discutidos, fazendo-se assim importante compreender todos os conceitos apresentados até então para o prosseguimento da leitura.

3 TRABALHOS RELACIONADOS E DESAFIOS DE PESQUISA

Neste capítulo abordaremos os trabalhos científicos relevantes em torno do tópico de pesquisa desta tese com uma abordagem *top-down*. Inicialmente discutiremos na Seção 3.1 trabalhos mais abrangentes, que propõem desde o uso de redes neurais com diferentes propósitos, como detecção de saliências em LFs, até diferentes metodologias de compressão de *light fields*. Já na Seção 3.2 discutiremos trabalhos que propõem síntese de vistas com as técnicas desenvolvidas pelos trabalhos discutidos na Seção 3.1 e preditores intra-quadro com aprendizado profundo tanto para imagens naturais quanto para LFs. A seção 3.3 aborda trabalhos que aplicam técnicas de *pruning*g em diferentes áreas, consolidando que essas técnicas são capazes de prover ganhos, seja de performance ou tempo de inferência, para diferentes tipos de dados e arquiteturas. Por fim, com base nos trabalhos discutidos e a maior compreensão sobre a área de pesquisa desta tese, a seção 3.4 apresenta o desafio de pesquisa presente que abordaremos neste trabalho.

3.1 Redes Neurais e Metodologias de codificação de LFs

Light Fields tem sido foco de pesquisas e projetos no mundo todo e, desta forma, existem trabalhos que propõem não somente diferentes maneiras de explorar as informações espaciais e angulares redundantes em *Light Fields*, mas também codificadores especializados.

Entre os tópicos que exploram redes neurais para propor soluções estão a detecção de saliências estudadas em Liang et al. (2022), Zhang et al. (2020) e Wang et al. (2019). Também a detecção de profundidade como as propostas em Ferrugem; Zatt; Agostini (2022), Jin; Hou (2022), Li et al. (2021) e Shi; Jiang; Guillemot (2019). Contudo, neste capítulo serão discutidos trabalhos sobre métodos de codificação, predições e síntese de *Light Fields*, considerando que estes tópicos são mais relevantes para o presente trabalho.

Na pesquisa de métodos de codificação e predição para LFs existem dois tipos

de ferramentas de predição distintas: a predição de blocos, onde de maneira análoga aos preditores de vídeo, busca-se predizer um bloco de amostras com base em um ou mais blocos de referência, e a predição de vistas, onde se cria uma vista sintética a partir de uma ou mais vistas de referência. Para melhor divisão destes trabalhos, serão denominados de preditores aqueles que predizem blocos internos do LF e sintetizadores os que geram vistas sintéticas completas.

Mukati et al. (2021) propõe um sintetizador de LFs em vista de uma característica importante dos codificadores de vídeo que é possuir uma arquitetura assimétrica, ou seja, sua complexidade reside majoritariamente na etapa de codificação, deixando consideravelmente menos processamento para ser realizado na decodificação. Mukati et al. (2021) defende que o *overhead* de comunicação e processamento das informações entre diferentes vistas torna desejável um melhor balanceamento de complexidade entre as etapas de codificação e decodificação. Para realizar este balanceamento, o autor utiliza um paradigma de codificação distribuída baseado no teorema de *Slepian-Wolf* desenvolvido por Wyner-Ziv, cujo afirma que duas fontes de informações estáticas correlacionadas podem ser codificadas separadamente e decodificadas em conjunto, isto enquanto atinge a mesma compressão caso codificadas conjuntamente (COVER; THOMAS, 2006).

Neste paradigma, primeiramente o codificador separa as vistas do LF em duas categorias: Vistas "chave" e vistas WZ (Wyner-Ziv). As quatro vistas chave são utilizadas como referências para síntese das vistas WZ e são selecionadas de acordo com distribuições específicas. Enquanto as vistas chave são codificadas utilizando métodos de codificação convencionais, neste caso o HEVC, as demais vistas são processadas através de um codificador WZ que possui menor complexidade. Este, por sua vez, possui apenas a finalidade de passar *bits* de paridade para o decodificador. Neste processo, as vistas primeiramente passam pela transformada DCT em blocos 4×4 e 8 matrizes de quantização desenvolvidas especificamente para LFs. Após a quantização, os *bits* são comprimidos por um codificador de entropia de checagem acumulada de paridades com baixa densidade (*Low-Density Parity Check Accumulate* - LDPCA), que garante também a possibilidade de corrigir ruídos provenientes das vistas sintetizadas (VARODAYAN; AARON; GIROD, 2006).

Já a decodificação segue o diagrama de blocos da Figura 29. Inicialmente, as vistas chaves decodificadas pelo HEVC são utilizadas para sintetizar as vistas R e Y, tal que Y é considerada uma vista WZ com ruídos. Estas vistas WZ são sintetizadas através de um método proposto por Navarro; Sabater (2021) que é detalhadamente descrito na Seção 3.2. Então, ambas as classes de vistas são transformadas por uma DCT, onde as vistas R são encaminhadas para o bloco de Modelagem de Ruído para que este modele os resíduos das vistas WZ. Segundo os autores, esta modelagem provê os melhores resultados em tempo de execução quando realizada ao nível de

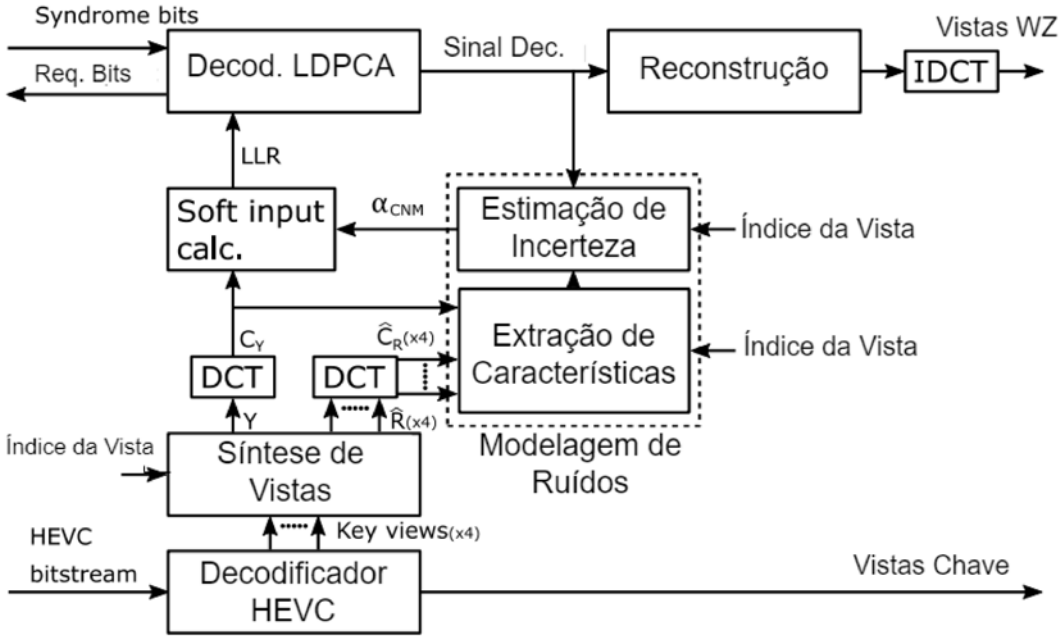


Figura 29 – Decodificador baseado em síntese de vistas proposto por Mukati et al. (2021)
Fonte: Adaptado de Mukati et al. (2021).

pixel e, portanto, eles realizam a modelagem por meio de duas redes neurais. A primeira rede é responsável pela extração de características espaciais e por estimar o sinal dos resíduos. Esta é constituída de dois blocos compostos por *kernels* 3D de profundidade 1 com funções de ativação ELU (Exponential Linear Unit) que compartilham seus pesos. O primeiro bloco é treinado para extrair características comuns entre todas as bandas de frequência dos resíduos, já o segundo bloco é treinado para identificar características específicas de cada banda, dado suas propriedades distintas. A segunda rede neural é responsável pela quantização inversa do sinal e, para melhores resultados, cada camada da rede é treinada para um índice de quantização específico em vista de suas propriedades também distintas. Por fim, todas as camadas das redes sofrem uma normalização de *batch*.

Após reconstruídos, os sinais de resíduos são enviados para o *soft input*, onde as probabilidades dos cortes condicionais dos *bitplanes* são calculadas e enviadas para o decodificador LDPCA realizar a correção de erros nos *bits* e, posteriormente, enviá-los para serem reconstruídos. Finalmente, o sinal reconstruído passa pela DCT inversa para dar origem à vista WZ. Os resultados atingidos por este codificador se mantêm, em média, 3 dB abaixo do MuLE para um *bitrate* com 0,1 bpp inferior. Contudo, atinge ganhos de PSNR e *bitrate* quando comparado com HEVC intra e outros trabalhos como Salmistraro et al. (2014).

Para transmitir LFs pela internet com usabilidade assim como é feito atualmente com vídeos digitais, faz-se necessário considerar os fatores das escalabilidades de janela de exibição, qualidade, resolução, possibilidade de acesso aleatório e uma dis-

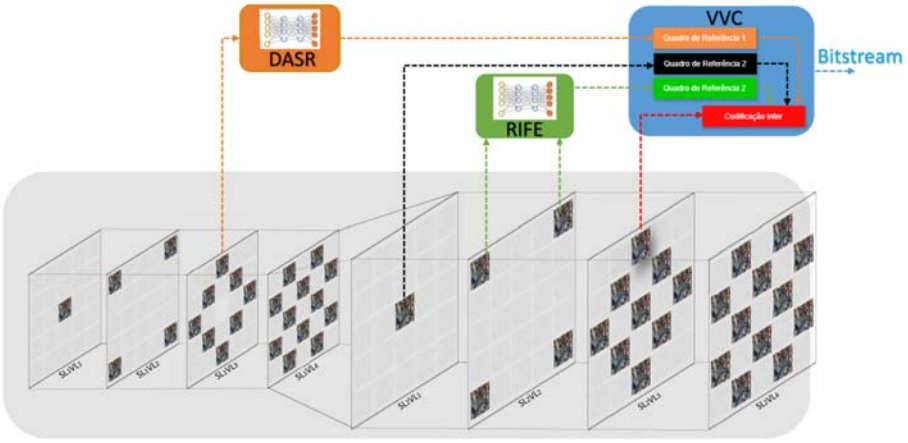


Figura 30 – Estrutura de compressão de light fields proposta por Amirpour et al. (2022)
Fonte: Adaptado de Amirpour et al. (2022).

tribuição uniforme de qualidade entre as vistas. Para atender a estas características, Amirpour et al. (2022) propõe um codificador de LF que proporciona uma alta eficiência de compressão e pode adaptar sua configuração ao tipo de exibição, canal de transmissão, condição da rede, capacidade de processamento e demais necessidades do usuário.

Para prover adaptação a diferentes resoluções, inicialmente são geradas vistas com 50% da resolução original. Já para prover adaptação a cenários com diferentes capacidades de processamento, as vistas de ambas resoluções são divididas em 3 diferentes camadas denominadas de *Viewport Layer* (VL) conforme o fluxo demonstrado na Figura 30, onde cada camada possui o dobro de vistas disponíveis para a transmissão. Devido às diferentes propriedades entre imagens de baixa e alta resolução, estas são codificadas de maneiras distintas.

Para as imagens de baixa resolução, primeiramente a vista central é codificada no modo intra para permitir acesso independente à vista. As vistas dos cantos são codificadas utilizando a vista central como referência. Já as demais vistas são codificadas utilizando também como referência vistas sintetizadas por um sintetizador de quadros de vídeo baseado em interpolação denominado RIFE (bloco verde) e proposto por (HUANG et al., 2020). Portanto, para codificar uma vista no modo inter, são necessárias 4 vistas de referência: (1) a vista central, (2,3) as vistas dos cantos utilizadas para a interpolação e (4) a vista sintética.

Já as imagens de maiores resoluções são codificadas conforme a de *viewport* que pertencem. A primeira camada, constituída apenas pela vista central, é codificada utilizando como referência uma versão *upscale* da vista central de baixa resolução obtida por meio de redes neurais profundas propostas por Niklaus; Mai; Liu (2017). As vistas do segundo *viewport* são codificadas independentemente usando versões *upscale* colocadas na camada espacial de menor resolução juntamente com a vista central. Para realizar o *upscale* das imagens é utilizado o algoritmo DASR (bloco

laranja) proposto em (WANG et al., 2021). Já os demais *viewports* utilizam as mesmas referências da segunda camada em conjunto com uma vista sintetizada.

Após todas as referências estarem disponíveis, é realizada a codificação inter no software de referência do padrão VVC, onde, posteriormente, o resíduo gerado é enviado para a quantização. O codificador provê para o usuário a possibilidade de alocar a concentração de *bits* de maneira flexível. Para isso, a vista central é codificada com um QP padrão e sua qualidade tomada como base para a quantização das demais vistas. Logo, os QPs dos demais *viewports* são definidos empiricamente de maneira a atingir uma qualidade similar à da vista central de acordo com um limite definido pelo usuário.

Os resultados do trabalho apresentam perda de 6dB na compressão do LF de resolução reduzida em relação ao original. Quando comparado a métodos focados em LFs com vistas mais esparsas, este trabalho apresenta ganhos de PSNR ao custo de aumento na taxa de bits em relação aos codificadores MV-HEVC (AHMAD; OLSSON; SJÖSTRÖM, 2017), *Shearlet Transform Based Prediction* (STBP) (AHMAD et al., 2020) e x265 (PEREIRA et al., 2019). Para analisar seus resultados sobre LFs com vistas mais densas (próximas uma a outra), estes foram comparados com o codificador JPEG Pleno *Verification Model* (PEREIRA et al., 2019), que utiliza predições 4D e possui melhor desempenho na codificação de LFs densos. Para esta classe de LFs, o codificador proposto atinge um PSNR em média 1,5dB superior ao JPEG Pleno para taxas maiores de compressão (entre 0,01 bpp e 0,05 bpp) e 1 dB a mais ou a menos para *bitrates* entre 0,1 bpp e 1 bpp. Por fim, vale ressaltar que o método proposto atinge melhores resultados para LFs com formas geométricas mais simples.

Em (CORRÊA, 2021) é desenvolvido um codificador específico para LFs capaz de diminuir o LF em 99,9996%, apresentando uma melhora de 35,22% em relação ao MuLE. Este ganho de eficiência foi obtido aplicando os conceitos de *Optical Flow* (OF) e *Phase Correlation* (PC) para explorar a característica de deslocamentos fixos entre as vistas dos LFs. O OF, proposto por Horn; Brian (1980), representa a distribuição de movimento entre duas imagens. Isto é, comparando duas imagens como, por exemplo, duas vistas de um LF, o OF representa o deslocamento aparente dos objetos de uma vista para a outra. Já o PC representa o movimento de translação entre duas imagens (KUGLIN; HINES, 1975). Diferentemente do OF, o PC representa a disparidade em termos de píxeis de uma imagem para outra com somente um vetor composto pelas componentes x e y. Vale ressaltar que, em vista da sinergia destas técnicas com as características dos LFs e a estrutura de informações retornadas pela sua saída, elas também podem ser utilizadas como entradas no treinamento de RNCs para aprimorar as informações analisadas sobre os LFs. Contudo, este trabalho é proposto apenas para altas taxas de compressão e atinge qualidades de imagem inferiores ao MuLE.

Uma das novas técnicas de codificação sendo estudadas para LFs constitui-se,

inclusive, não no uso de codificadores clássicos com ferramentas de predição e codificações de entropia, mas sim no conceito de representar o LF por via de redes neurais, seja o LF estático ou um vídeo. Feng; Varshney (2021) propõe o SIGNET (*Sinusoidal Gegenbauer Network*), uma metodologia que aplica transformadas de Gegenbauer (PADIERNA et al., 2018) em quatro vistas esparsas dos LFs para posteriormente armazená-los em uma representação neural através de um *Multilayer Perceptron* (MLP). Nesta técnica de representação as funções do MLP mapeiam as posições de cada píxeis aos seus valores RGB, sendo assim possível armazenar apenas as funções resultantes e reconstruir o LF original a partir destas. Através desta representação neural os autores armazenam um LF com em média 80% menos *bits* e os reconstroem com uma qualidade de 1dB acima de outros métodos similares. Vale ressaltar que nenhum destes métodos utiliza a técnica de quantização e, por isso, seus resultados não são comparados com nenhum codificador ou metodologia que a utilize como, por exemplo, o MuLE.

Existem diversos outros trabalhos que utilizam arquiteturas similares para representações neurais de LFs. Sitzmann et al. (2020) propõe uma arquitetura similar denominada SIREN (*Sinusoidal Representation Networks*) que utiliza funções de ativação sinusoidais sem aplicar transformações de *Gegenbauer*. Já Tancik et al. (2020) utiliza propriedades do domínio das transformadas de *fourrier* para representar os LFs em MLPs através de funções de alta frequência.

Ademais, todos estes trabalhos utilizam funções de ativação denominadas periódicas, ou seja, funções baseadas em seno ou cosseno que repetem seus padrões de ativação ao longo do tempo. A família de funções de ativação periódicas foi avaliada em (SITZMANN et al., 2020) e comparadas com funções não periódicas. Este trabalho concluiu que a família de funções periódicas apresenta uma maior precisão para modelar informações de alta frequência e de maiores ordens de derivação como os *Light Fields* e sua estrutura de quatro dimensões. Por fim, outro aspecto importante que estes trabalhos apresentam em comum é a utilização de GPUs para treinamento de suas redes, ressaltando a importância e necessidade desta prática.

Diferentes técnicas de predições podem ser aplicadas às diferentes formas de representação dos LFs. Chen et al. (2020) defende que a representação *multi-view* (Figura 3 (a)) é viável e útil para a predição de vistas inteiras de um LF, o que, na verdade, é também uma síntese de vistas. Para sintetizar as vistas, este trabalho cria mapas de disparidade entre algumas vistas definidas como "essenciais" e excluem as vistas consideradas como "opcionais". A seguir, utilizam o HEVC inter-quadros para codificar as vistas "essenciais" como um pseudo vídeo. Por fim, ao decodificar o LF, as vistas intermediárias "opcionais" são sintetizadas por um algoritmo de aprendizado de máquina proposto por Zhou et al. (2018) e denominado de MPI (*learning-based multiplane images*). O MPI foi desenvolvido para sintetizar diferentes vistas de uma cena

de realidade virtual ou realidade aumentada via duas imagens de referência. Apesar de um sintetizador genérico de vistas poder ser utilizado para sintetizar vistas de um LF, ele não explora a estrutura do *Light Field*, pois uma parte considerável das vistas é perdida e posteriormente reconstruída do zero.

Note que várias predições foram propostas para diferentes representações e formatos. Além destas predições, a literatura ainda apresenta diversos trabalhos usando predições realizadas por RNCs.

3.2 Preditores e Sintetizadores com Aprendizado Profundo

Para realizar a síntese das vistas utilizadas no codificador proposto por Mukati et al. (2021), o autor propõe em Navarro; Sabater (2021) um sintetizador constituído por três redes neurais. A primeira rede extrai características das vistas de referência posicionadas nos cantos do LF. As informações relevantes extraídas são então enviadas para a segunda rede neural, que é responsável por estimar mapas de disparidade entre a vista sintetizada e as vistas de referência. Por fim, a terceira rede transforma as vistas de referência nas vistas sintetizadas através do mapa de disparidade estimado. As vistas sintéticas alcançam um PSNR de 38,12dB em relação às vistas dos cantos, apresentando uma melhora média de 4,5dB em relação a outros trabalhos que empregam apenas uma rede neural.

Predição para compressão não é o único desafio tecnológico presente na utilização de *Light Fields* e, também, não é a única aplicação de preditores e RNCs neste campo de pesquisa. Devido à alta quantidade de informações e processamento necessários, atualmente as resoluções espaciais das câmeras acabam sendo restringidas (WU et al., 2017). Visando aumentar estas resoluções, Wu et al. (2017) propõem a utilização de RNCs para aumentar a resolução através de *up-scalers* e reconstruir todas as vistas dos LFs a partir de apenas uma seleção de vistas esparsas sob a representação EPI (17). Nesta proposta, os autores aplicam um desfoque na representação EPI das vistas esparsas para remover as baixas frequências espaciais presentes. Posteriormente, é realizado um *up-sampling* (técnica para aumento de resolução de imagens (SZELISKI, 2010)). A seguir, uma RNC treinada com EPIs reconstruídas e redimensionadas (*down-sampled*) para sua resolução original é utilizada para restaurar os detalhes angulares na nova EPI. Por fim, um *blur* inverso é aplicado para restaurar os detalhes das informações angulares.

Buscando aprimorar a metodologia proposta por (WU et al., 2017) e avaliar o impacto do resultado no processo de codificação, Zhao et al. (2018) modifica a metodologia de reconstrução de LFs aplicando RNCs nas demais etapas do processo. A primeira modificação foi utilizar duas RNCs para realizar o *up-sampling* nas imagens EPIs ao invés de um algoritmo convencional de *up-sampling*. Desta maneira, enquanto

uma RNC se especializa em aumentar a resolução das linhas de amostras da EPI, a outra se especializa na resolução das colunas da EPI. Cada RNC gera uma nova EPI, a EPI-Linha, que representa as disparidades horizontais do LF, e a EPI-Coluna, que representa as disparidades verticais do LF. Ao se ter duas EPIs com informações especializadas diferentes, pode-se também aplicar uma RNC a cada EPI. Desta maneira, os autores treinaram 2 RNCs separadas, uma para aprender as disparidades verticais e a outra as horizontais, possibilitando assim que cada RNC precise aprender apenas informações unidirecionais.

Outra diferença no treinamento das RNCs neste trabalho é que as EPIs de baixa resolução foram usadas como dados de treinamento e as EPIs de alta resolução como *labels*, ao invés de utilizar uma versão *down-sampled* também como dados de treinamento. Esta modificação no treinamento permitiu que as RNCs se tornassem capazes de reparar não somente perdas de informação causadas pela seleção de vistas esparsas, mas também perdas causadas pelo processo de codificação. Desta maneira, os autores possibilitam reconstruir todas as vistas de um LF após a codificação e decodificação de apenas metade das vistas do mesmo. Ao comprimir apenas as vistas selecionadas com o HEVC e posteriormente reconstruí-las, essas modificações apresentaram uma redução de BD-*Rate* de 35,85% quando comparado com o HEVC codificando o LF inteiro como um pseudo-vídeo.

Buscando aperfeiçoar a codificação do JPEG Pleno para LFs HDCAs (High Density Camera Arrays), Astola; Tabus (2018b) propoem uma adaptação das predições realizadas neste codificador. LFs capturados por *arrays* de câmeras como demonstrado na Figura 5. O foco em LFs HDCA é necessário porque em *arrays* de câmeras, a distância entre as vistas do LF é maior do que de um LF obtido por meio de uma câmera *lenslet* com micro lentes. Consequentemente, o deslocamento dos píxeis de uma vista para outra será maior, o que é uma característica específica de LFs HDCA até então não explorada especificamente pelo padrão. Esses experimentos focaram em aperfeiçoar a exploração de informações angulares de blocos vizinhos de LFs no formato *lenslet* (Figura 2) mediante predições lineares.

Inicialmente, a arquitetura de codificação apresentada solicita que o usuário escolha um conjunto de vistas esparsas e que não sejam adjacentes, para servirem como referência no processo de predição durante a sua codificação pelo padrão JPEG 2000. Simultaneamente, um conjunto de mapas de profundidade é codificado utilizando quantização. Em vista da maior distância entre as vistas em LFs HDCA, torna-se muito importante a precisão dos métodos de estimação de profundidade para a reconstrução das vistas vizinhas. Portanto, para este fim, os autores utilizaram o algoritmo definido pelo JPEG em Naman et al. (2017). Então, os mapas de profundidade estimados e até cinco vistas de referência são utilizados por um preditor esparso definido em Helin et al. (2016) para predizer a próxima vista. Nesta predição, separadamente

para cada componente de cor, o algoritmo aplica convoluções na vista com *stride* de 1 pixel buscando a referência que gere o menor resíduo. Esta busca é realizada em pequenas janelas de busca não definidas no texto, isso por ser focada para predizer pequenas disparidades nas vistas, pois, segundo os autores, as grandes disparidades são identificadas eficientemente através dos mapas de profundidade recíprocos e, portanto, não necessitam da busca. A inclusão desta predição linear aprimorou o PSNR da compressão do LF pelo JPEG 2000 em 8dBs e 9dBs para os *bitrates* de 0,03 e 0,05 *bits* por pixel respectivamente. Contudo, o desempenho deste trabalho cai para uma melhora de 0,2dB para um *bitrate* de 0,8. Comparações com o JPEG Pleno não foram realizadas.

Outra técnica aplicada em compressão de imagens são as transformadas baseadas em grafos seguidas de predições baseadas em grafos. Apesar da sua utilização ser eficiente na compressão de imagens, o custo computacional das funções matemáticas destas transformadas cresce a níveis intratáveis quando aplicada a conjuntos maiores de dados como *Light Fields*. Desta forma, para realizar predições espaciais-angulares baseadas em grafos para LFs, Rizkallah; Maugey; Guillemot (2019) utiliza a predição intra-quadro do HEVC em modo sem perdas (*lossless*) para gerar um quadro de referência inicial e, para os quadros posteriores, serão aplicadas a transformadas de grafos independentes em blocos menores das vistas, com o intuito de diminuir o custo computacional e tornar a aplicação possível. Contudo, ao realizar transformadas de grafos localizadas e independentes, perde-se as dependências com outras localidades do sinal, como outros blocos ou vistas. Essa perda pode acarretar uma perda de eficiência indesejável para as predições baseadas em grafos. Por fim, os dados resultantes da predição foram codificados por um codificador de entropia simples. Os experimentos utilizando esta metodologia mostraram uma melhora de cerca de 30% em compressão quando comparado com o HEVC inter-quadros. Porém, este resultado é atingido apenas em compressões *quasi-lossless* (com perdas mínimas de qualidade de 50dB) para LFs de alta qualidade. Além disso, para atingir esta eficiência, a técnica necessita de um codificador intra-quadro em conjunto para gerar os quadros de referência da predição.

Inspirado pela predição intra do codificador HEVC, Li; Jin; Zhong (2020) desenvolveu uma predição intra-quadro focada em vídeos de *Light Fields* 2,0 que, por possuírem resoluções espaciais maiores para cada vista, apresentam volumes de dados mais significativos. Nesta abordagem, os autores dividem as vistas do LF em blocos separados e, similarmente à predição intra-quadro do HEVC, utilizam amostras de blocos vizinhos adjacentes ao bloco atual para o predizer, explorando assim a alta correlação entre estes. A predição intra-quadro deste trabalho é realizada através da interpolação de amostras vizinhas sob pesos específicos que ditam a relevância de cada amostra vizinha para cada posição das amostras do bloco atual. Comparando

com o HEVC inter-quadros, esta predição intra-quadro especializada aumentou a taxa de compressão em 9,63% e 4,73% à em comparação ‘s configurações *All Intra* e *Random Access* do codificador HEVC.

Spadaro et al. (2023) apresenta uma abordagem inovadora para o aprimoramento do processo de predição intra no codec MPEG-5 Essential Video Coding (EVC)(CHOI et al., 2020) utilizando redes neurais convolucionais. Os autores reconfiguram o problema da predição de blocos como uma tarefa de preenchimento de lacunas (*inpainting*). Onde utilizam *masked convolutions*, técnica clássica de *inpaint* onde as convoluções aplicam filtros apenas a píxeis relevantes e não-mascarados (ignorando regiões ausentes), aplicando assim redes convolucionais para melhorar a eficiência da compressão. Seus experimentos demonstram uma redução média de mais de 6% na BD-Rate em relação ao perfil básico do EVC. A similaridade da metodologia empregada, especialmente o uso de redes neurais convolucionais para predição intra, estabelece uma conexão direta com o trabalho desenvolvido nesta tese, que também busca otimizar processos de codificação de vídeo utilizando redes neurais, porém focando na estrutura de dados específica dos Light Fields.

3.3 Técnicas de *Pruning* para Aprendizado Profundo

Técnicas de *pruning* vem sido largamente aplicadas para acelerar redes neurais profundas e também melhorar seus desempenhos. Anwar; Hwang; Sung (2017) apresenta uma abordagem de poda estruturada em redes neurais convolucionais profundas para reduzir a complexidade computacional e o consumo de memória, focando em aplicações em tempo real. Para selecionar conexões a serem podadas, utilizam um filtro de partículas evolutivo (*Evolutionary Particle Filter* - EPF), que avalia combinações de padrões de conectividade com base no impacto na taxa de erro de classificação. A técnica proposta ainda introduz esparsidade estruturada em três granularidades: no nível de *feature maps*, de *Kernels* e de esparsidade intra-kernel com passos definidos (*Intra-Kernel Stride Sparsity* - IKSS). Na poda de *feature maps* todos os pesos associados a conexões de entrada e saída dos mapas são zerados. A remoção reduz diretamente a largura da camada convolucional, diminuindo a quantidade de operações e parâmetros da rede. Já na poda de *Kernels* das convoluções, cada *kernel* representa um filtro aplicado ao mapa de características de entrada, e os determinados como irrelevantes pelo EPF são removidos inteiramente. Assim, remover um núcleo equivale a eliminar um filtro específico de uma camada. Por fim, na poda IKSS, os pesos dentro de um núcleo são zerados de forma estruturada. Isso cria padrões regulares de esparsidade com base nos parâmetros de *offset* e *stride*.

Além disso, os autores comparam os resultados com métodos de poda não estruturada, evidenciando a eficiência computacional e a viabilidade prática das técnicas

estruturadas. Os experimentos demonstram que, embora a poda não estruturada permitisse ganhos de eficiência maiores (acima de 70%), a poda estruturada conseguiu remover até 72% dos parâmetros com menos de 1% de perda de acurácia no *dataset* CIFAR-10. Ainda, essa redução no tamanho da rede reduziu em até 50% o uso de memória em comparação com a poda não estruturada. Por fim, convolução foi implementada como multiplicação de matrizes usando a biblioteca BLAS (CHELLAPILLA; PURI; SIMARD, 2006), o que acelerou a execução de redes podadas estruturalmente em até 2,67 vezes.

Gu; Meng; Sun (2024) propõem uma abordagem de poda para o modelo YOLOv7, visando realizar detecção rápida e precisa de variações de sementes de Colza (*Brassica Napus*) de forma eficiente em dispositivos embarcados com recursos limitados. O trabalho emprega uma estratégia de poda em duas dimensões (espacial e de canais) conforme Anwar; Hwang; Sung (2017) para reduzir a complexidade computacional e o número de parâmetros do modelo, sem comprometer a precisão.

No nível espacial, ramos redundantes em módulos sobre-parametrizados são removidos com base em sua contribuição ao desempenho do modelo. No nível de canais, é aplicada uma estratégia baseada na normalização em lotes (*Batch Normalization*), que identifica canais menos importantes pela análise de seus fatores de escala. A abordagem customizada de poda camada a camada demonstrou ser a mais eficaz, reduzindo os parâmetros do modelo de 36,5M para 9,19M e o tempo de inferência por imagem em um Raspberry Pi 4B de 4,48s para 1,18s, enquanto aumentava a precisão média (mAP) de 96,68% para 96,89%. Este trabalho é relevante no contexto desta tese por demonstrar a viabilidade da poda estruturada para acelerar modelos de redes neurais com baixa ou nenhuma perda de desempenho em cenários de recursos limitados.

Já He; Zhang; Sun (2017) apresenta um novo método de *pruning* de canais (*channel-wise prunning*) que visa acelerar CNNs profundas, enquanto mantém a eficiência da rede. Ele propõe um algoritmo iterativo de duas etapas que utiliza regressão LASSO para selecionar os canais a serem podados e mínimos quadrados para reconstruir os canais mantidos. O método reduz efetivamente o erro acumulado e é compatível com diversas arquiteturas, alcançando uma aceleração de 5x com um aumento de 0,3% no erro. Ele também demonstra desempenho significativo em redes modernas como ResNet e Xception, com uma perda irrisória de precisão, atingindo ganhos de aceleração em *hardware* de 2x.

3.4 Desafios de Pesquisa

Ao estudar os trabalhos relacionados, fica claro que muito ainda precisa ser definido e avaliado em torno da representação, compressão e ferramentas de predição

em volta de *Light Fields*. Em muitos trabalhos o HEVC é utilizado para codificar os LFs como pseudo-vídeos, outros o JPEG Pleno no modo de transformadas (MuLE) e até mesmo o JPEG 2000, codificador de imagens estáticas proposto no ano de 2000.

O amplo espaço de exploração é típico de tópicos de pesquisa emergentes como *Light Fields* e, consequentemente, abre espaço para avaliações e propostas em diversos aspectos do tema. Por exemplo, ao se propor uma arquitetura de RN para predição intra-quadro, representação ou reconstrução de LFs, esta pode apresentar resultados melhores ao se utilizar o codificador HEVC para compressão posterior dos resíduos como em Li; Jin; Zhong (2020) ou ao se utilizar o JPEG Pleno. De maneira análoga, cabe avaliações sobre quais tipos de arquiteturas e tecnologias se adaptam melhor quando aplicadas a codificadores diferentes.

O desenvolvimento de técnicas de predição e síntese de *Light Fields* não é trivial, especialmente considerando que LFs apresentam estruturas diferentes das imagens estáticas utilizadas até então. Conforme mencionado, a robusta e complexa estrutura 4D dos LFs permite a utilização de múltiplas formas de representação. Analisando os trabalhos relacionados discutidos, também fica evidente que não há definição de uma estrutura de representação ótima. Estruturas EPI apresentam boas características estruturais sobre as profundidades dos objetos na cena e têm sido utilizadas em abrangência para operações de *up-sampling*, geração de mapas de profundidade e reconstrução de vistas. Ainda assim, existem trabalhos como Feng; Varshney (2021) que realizam reconstruções de vistas eficientes utilizando vistas esparsas separadas.

Muito se tem a pesquisar, propor e avaliar sobre LFs, porém uma abordagem pouco explorada até o momento é a utilização de Redes Neurais para realizar a tarefa de predição. A tarefa de adaptar a predição intra-quadro e suas interpolações definidas pelo HEVC para o contexto 4D dos LFs não é trivial. Criar novas técnicas de predição que explorem ao máximo esta estrutura acaba por ser um desafio consideravelmente mais complexo.

Cabe a tal esforço não apenas identificar as redundâncias angulares e espaciais presentes nos LFs, mas também desenvolver uma nova técnica que consiga realizar uma predição eficiente para qualquer cena independente do seu conteúdo. A Figura 31 exemplifica o problema de predizer um bloco através de blocos de referência já codificados. Observe que há grande semelhança entre os blocos de referência (em azul) superior central e esquerdo, assim como entre o superior esquerdo e superior direito. Compete à predição utilizar as informações semelhantes presentes nestes quatro blocos de referência para estimar o conteúdo do bloco atualmente sendo codificado (em amarelo). Quanto mais precisa for esta predição, menor será a diferença entre as amostras geradas pela predição e as originais, consequentemente diminuindo o resíduo gerado, o que aumenta a capacidade de compressão do sinal.

Tendo em vista que a eficiência das predições deste gênero depende da correlação

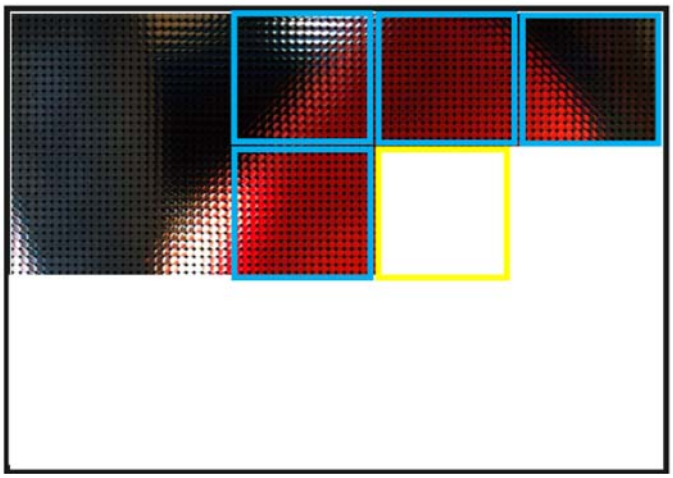


Figura 31 – Contexto de predição constituído por 4 blocos de referência (em azul) e 1 bloco de objetivo (amarelo) de um Light Field.
Fonte: Desenvolvido pelo Autor.

entre os diferentes blocos da imagem, pode-se analisar a diferença no desafio de se predizer uma imagem 2D comum e um LF no formato *lenslet* avaliando a correlação entre os píxeis destes dois formatos de imagem. A Figura 32 mostra a auto correlação dos píxeis em uma imagem 2D (a) e dos píxeis em um *Light Field* (b). Analisando os gráficos, fica evidente que a correlação em uma imagem 2D é muito mais suave e com menos mínimos e máximos locais, e.g., vales e picos. A presença de vales e picos no espaço de busca dificulta a tarefa dos algoritmos de não ficarem presos em mínimos ou máximos locais.

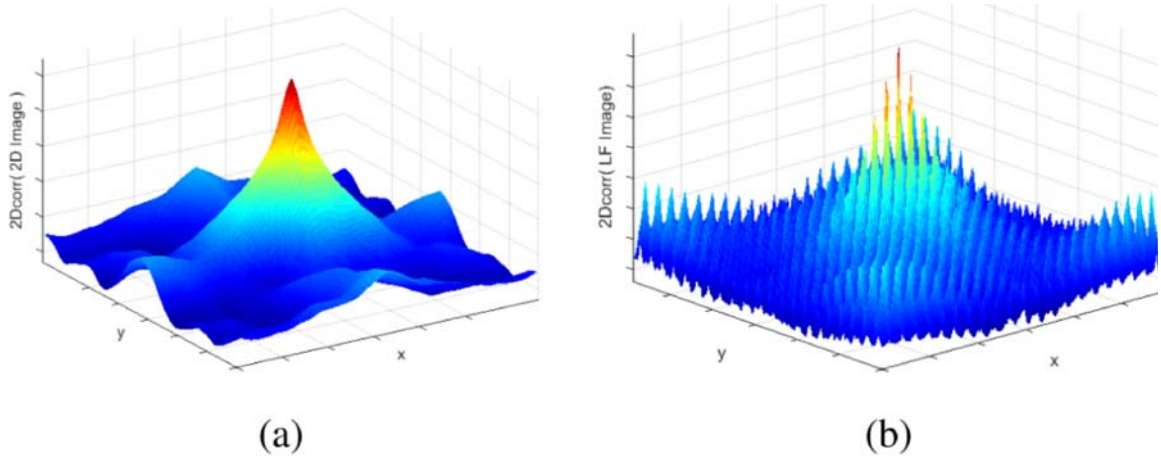


Figura 32 – Auto Correlação nos píxeis de (a) uma Imagem 2D e (b) um *Light Field*
Fonte: Adaptado de Ashok; Neifeld (2010).

Devido a essa maior dificuldade, em espaços amostrais mais complexos como da autocorrelação dos píxeis de um LF, algoritmos de inteligência artificial são propícios para sua navegação. Contudo, algoritmos mais simples como subidas de encostas e têmpera simulada não trabalham ao nível de píxels, o que os torna inaplicáveis neste contexto. Já técnicas como algoritmos genéticos não são ideais por não serem episó-

dicos, ou seja, mesmo após definidos, seus vários indivíduos instanciados precisarão ser processados completamente para cada LF, o que demandaria um tempo inviável.

Por fim, algoritmos de aprendizado profundo têm apresentado excelentes capacidades de aprender a encontrar correlações em espaços amostrais complexos e de várias dimensões (LI et al., 2021). Ademais, esses algoritmos, após treinados, podem ser rapidamente aplicados a qualquer LF ou bloco de maneira independente, possibilitando o seu uso mesmo durante um processo de codificação.

4 SOLUÇÕES PROPOSTAS

A metodologia do processo para utilizar RNCs consiste em três etapas: (i) preparar o conjunto de dados (*dataset*) para o treinamento, (ii) definir a estrutura de treinamento, (iii) definir as topologias a serem utilizadas . Cada uma destas etapas possui objetivos próprios e múltiplas técnicas para realizá-las. Neste trabalho avaliaremos não somente diferentes topologias de RNCs, mas também múltiplas técnicas para cada uma destas etapas. Este capítulo estará focado na explicação deste processo e das diferentes técnicas avaliadas. Todos os procedimentos descritos neste capítulo tem sua implementação disponibilizada em Machado (2025).

4.1 Metodologia de Treino e Avaliação

Como o interesse é otimizar a eficiência da RNC como um preditor intra especializado em LFs para codificadores de vídeo, foi adotado um ciclo de avaliação que tome a eficiência de codificação como métrica final exemplificado pela Figura 33. Para avaliar uma modificação na metodologia, primeiramente treinamos a RNC sob a modificação desejada e, posteriormente, esta RNC é utilizada para codificar os LFs de teste.

Para viabilizar a comunicação dos codificadores programados na linguagem C, com as redes programadas em *python* e instanciadas na GPU, os codificadores avaliados foram alterados para que, durante a predição intra, eles enviem o contexto disponível nesta etapa para o preditor através de um pacote UDP (*User Datagram Protocol* - Protocolo de Datagrama do Usuário), esteja o preditor em GPU ou outro *hardware*. O contexto é formado pelo conjunto dos três blocos vizinhos juntamente com o bloco sendo predito, cujas amostras estão atualmente zeradas. Após enviar o contexto para o servidor, o codificador espera receber, via a mesma sessão UDP, o bloco predito pela rede neural.

O servidor, por sua vez, tem o modelo treinado carregado na memória da GPU e o utiliza para predizer o bloco desejado a partir do contexto de entrada. Por fim, o servidor envia o bloco predito para o codificador que, ao recebê-lo, o coloca em com-

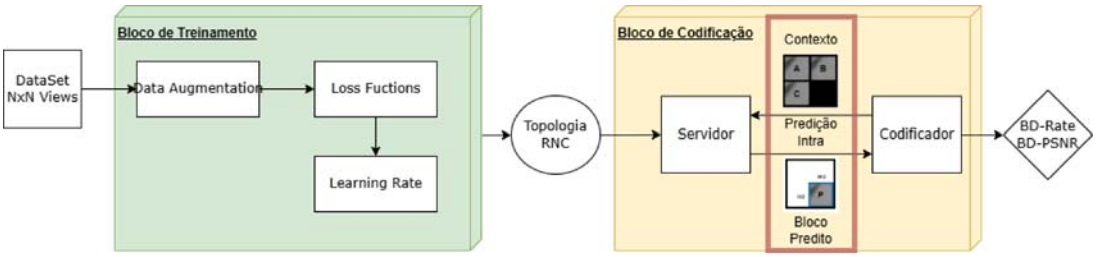


Figura 33 – Fluxo de avaliação das técnicas avaliadas.
Fonte: Desenvolvido pelo autor.

petição com os demais blocos preditos pelos outros modos intra, prosseguindo com a codificação como de costume. Este fluxo demonstrado pelo módulo de codificação da Figura 33 ocorre para cada bloco que passa pela predição intra do codificador sendo avaliado. Após a codificação de todos os LFs para todos os 4 QPs definidos pelas CTCs (Common Test Conditions) Bossen (2013); Bossen et al. (2020) do codificador, as métricas de BD-Rate e BD-PSNR são extraídas para avaliar a eficiência de codificação fornecida pelo uso da rede. Ambas as métricas foram calculadas a partir das informações providas pelos relatórios dos codificadores e utilizaram apenas o canal de luma Y.

BD-Rate (Bjøntegaard Delta Rate) e BD-PSNR (Bjøntegaard Delta PSNR) (BJØNTEGAARD, 2008) são métricas amplamente utilizadas para comparar a eficiência de codificação de codificadores de vídeo. O BD-Rate calcula a variação percentual na taxa de *bits* necessária para alcançar uma mesma qualidade objetiva (tipicamente PSNR), enquanto o BD-PSNR avalia a diferença média na qualidade visual (em dB) para uma mesma taxa de *bits*. Ambas as métricas baseiam-se na interpolação das curvas de desempenho taxa-distância (Rate-Distortion, RD) entre duas codificações. Nos resultados apresentados neste trabalho, calculamos o BD-Rate e BD-PSNR entre as codificações feitas pelos codificadores apenas com seus modos intra originais e após inserirmos o nosso. Ainda, é importante ter em mente que, em vista de que o BD-Rate mede o percentual (%) de variação da taxa de *bits* para uma qualidade fixa, um resultado negativo indica ganhos de redução na taxa de *bits*, enquanto um resultado positivo indica perdas. Logo, quanto menores os resultados de BD-Rate, melhor. Já o BD-PSNR, por medir a qualidade visual para uma mesma taxa de *bits*, deseja-se números positivos.

Destacamos que o *bitstream* gerado por este fluxo de operação permanece decodificável pelo receptor sem qualquer informação adicional, uma vez que o transmissor realiza a predição intra dos blocos utilizando o mesmo contexto disponível no receptor. Para isso, basta o decodificador no receptor utilizar o preditor utilizado na codificação.

Esse fluxo geral de treinamento e codificação deve ser aplicado na sua totalidade a um *dataset*. Embora estimemos que ganhos obtidos em um *dataset* se mantenham também para outros *datasets*, é importante definir um *dataset* inicial para o treino e

como este é obtido, de modo a estabelecer uma base de avaliação para modificar e analisar as demais etapas do treinamento e características da arquitetura.

(a) Ankylosaurus-1

(b) Bikes

(c) Black-Fence

(d) Ceiling-Light

(e) Dange-De-Mort

(f) Friends-1

(g) Houses-&-Lake

(h) Reeds

(i) Rusty-Fence

(j) Slab-&-Lake

(k) Swans-2

(l) Vespa

Figura 34 – Light Fields do dataset EPFL selecionados como caso de teste (RERABEK; EBRAHIMI, 2016).
Fonte: Desenvolvido pelo author.

4.2 Preparação do Conjunto de Dados

O *dataset* de *Light Fields* utilizado neste trabalho foi disponibilizado pela *École Polytechnique Fédérale de Lausanne* (EPFL), sendo este constituído por 115 LFs dis-

tintos que podem ser visualizados na Figura 34 (RERABEK; EBRAHIMI, 2016). Gostaríamos de ressaltar que a denominação original da sequência aqui nominada como "*Ankylosaurus-1*" é "*Ankylosaurus-&-Diplodocus-1*", e que esta alteração na nomenclatura foi efetuada apenas por motivos de compatibilidade com a formatação deste documento e melhor visualização de figuras e tabelas. Destes 115 LFs, 12 foram selecionados para teste conforme proposto por Zhong et al. (2019), não estando assim presentes nas etapas de atualização dos pesos do treinamento.

Os LFs presentes na imagem, assim como qualquer LF capturado por uma câmera Lytro, por consequência da estrutura de micro-píxeis dos *Light Fields*, ocorre um fenômeno nas imagens chamado de *vignetting*, que leva a borda das vistas a possuírem píxeis pretos ou muito escuros. Em vista de que este comportamento pode prejudicar o treinamento das RNCs, as duas linhas e colunas de píxeis mais externas das SAIs foram retiradas e os píxeis imediatamente mais internos tiveram seus valores corrigidos através do processo de *brightening*. Desta maneira, os LFs *lenslet* originalmente compostos de 15×15 SAIs foram reduzidos para 13×13 SAIs. Estes processos foram todos realizados pela ferramenta Raytrix (2022) disponível no *github*. Contudo, durante este processo, alguns píxeis extras são descartados, fazendo com que alguns LFs percam de 1 a 2 píxeis de altura ou largura.

Duas alterações deste *dataset* serão utilizadas, uma com 8×8 vistas e outra com 16×16 vistas. Inicialmente, avaliaremos adaptar os LFs selecionando um conjunto de 8 vistas centrais e descartando as demais vistas, como similarmente proposto em Mukati et al. (2020, 2021). Gerando LFs com resolução $625 \times 434 \times 8 \times 8$, possuindo assim MIs de 8×8 píxeis e alinhadas com os tamanhos de blocos dos codificadores de vídeo. Estes LFs modificados serão então utilizados para a extração de exemplos de blocos aos quais as arquiteturas devem ser treinadas. Em vista de que esta estrutura possui menos informações nas MIs para serem preditas pelas redes em comparação às MIs 16×16 e que isso as torna mais fáceis de serem preditas, esta versão com 8×8 vistas do *dataset* será utilizada para a análise exploratória inicial deste trabalho.

Posteriormente, para realizar uma comparação justa, modificaremos o *dataset* conforme proposto por Zhong et al. (2019), trabalho similar ao nosso, onde se utiliza *padding* para deixar os LFs com 16×16 vistas, ao invés de 13×13 ou 8×8. Em vista de que o trabalho em questão não define como o *padding* foi realizado, para sermos o mais justos possível, definimos um *padding* o mais próximo de novas vistas sem realizarmos a síntese ou gerarmos novos dados. Ao invés de apenas completarmos com píxeis pretos (o que facilitaria a predição da rede e enviesaria os resultados ao nosso favor), nós copiamos as 48 vistas mais externas de cada borda do LF até formarmos 1 nova coluna e linha nos lados superior e esquerdo do LF e 2 colunas e linhas nos lados direito e inferior do LF conforme demonstrado na Figura 35 do LF *Bikes* no seu formato *Multi-view*. Como após esta cópia ainda faltam vistas nas diagonais entre as

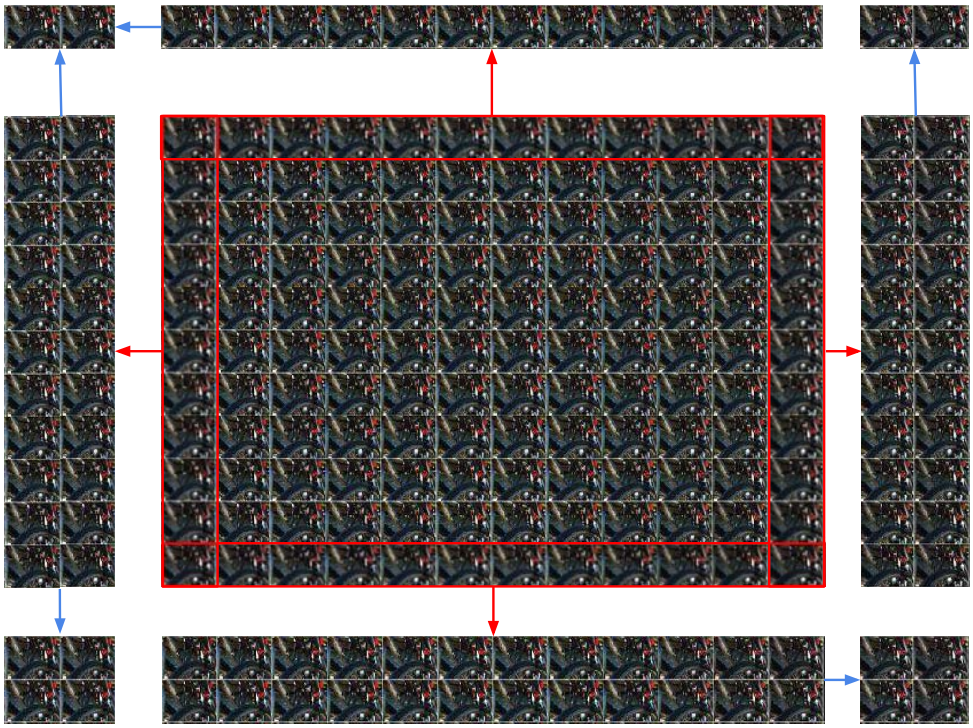


Figura 35 – Método de preenchimento de vistas (padding) utilizado.
Fonte: Desenvolvido pelo autor.

colunas e linhas de vistas criadas, para manter a coerência entre as diferenças entre os ângulos das vistas o melhor possível, preenchemos estas lacunas copiando as vistas mais externas do *padding* em sentido anti-horário. Vale ressaltar que o sentido é puramente opcional e fazê-lo em sentido horário resultaria em diferenças angulares similares.

Ainda, os blocos de referência que serão recebidos pela rede durante a codificação serão amostras codificadas e decodificadas, fazendo com que treinar as redes com LFs também codificados e decodificados, em teoria, aproxime o treinamento da CNN do cenário final e, consequentemente, melhore o treinamento. Contudo, segundo Spadaro et al. (2023), isto pode retirar informações das imagens e gerar artefatos indesejados, fazendo assim com que treinar a rede sobre imagens sem compressão obtenha eficiências melhores.

Ainda, Lecun et al. (2012) recomenda que as entradas de algoritmos de *machine learning* com *backpropagation* como as RNCs, sejam treinadas com entradas normalizadas cujo valor médio seja 0. Ademais, os blocos de referência dos contextos de entrada da rede durante a codificação possuirão apenas seu canal luma. Logo, os LFs foram convertidos para YCbCr e apenas o canal Y foi mantido para aproximar o treinamento do cenário de predição do codificador. Já para realizar as codificações, os LFs utilizados na validação foram convertidos no seu formato PNG em RGB para arquivos YUV em YCbCr através da ferramenta ffmpeg (FFMPEG DEVELOPERS, 2024).

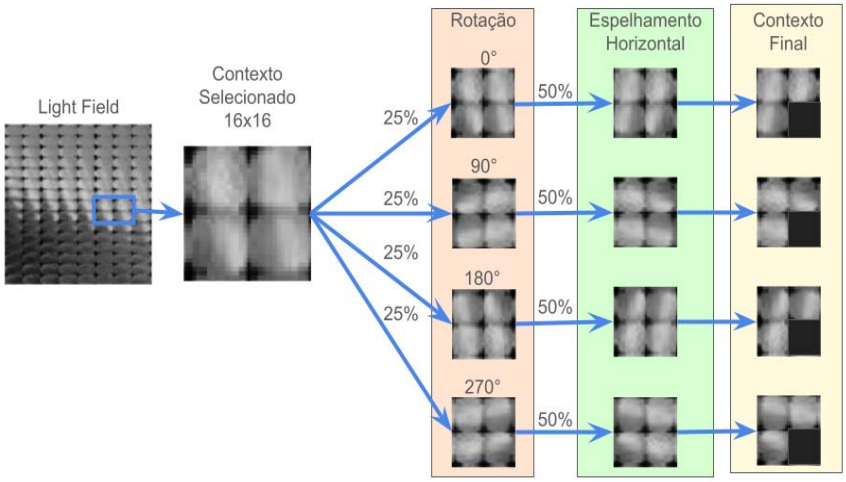


Figura 36 – Operações de data augmentation aplicadas com suas respectivas ordens e probabilidades.
Fonte: Desenvolvido pelo autor.

4.3 Bloco de Treinamento das Redes

Nesta subseção, definiremos em detalhes o funcionamento dos hiperparâmetros de *data augmentation*, *loss function* e *learning rate* do bloco de treinamento, assim como o porquê de estes terem sido selecionados. A definição destes hiperparâmetros é essencial para evitar o comportamento de *overfitting* e providenciar uma boa convergência para a rede.

4.3.1 Data Augmentation

Devido ao tamanho limitado do conjunto de treinamento, recorremos à data augmentation para evitar que o modelo sofra *overfitting*. Aqui cabe definir como os contextos são extraídos dos LFs para servirem de entrada das redes neurais ao longo do treinamento.

A Figura 36 apresenta o diagrama de blocos dos procedimentos de *data augmentation* aplicados considerando blocos 16×16 pela simplicidade visual que estes apresentam por cobrirem uma MI por bloco. O primeiro passo a ser definido é como os contextos dos LFs que servirão de entrada da rede são selecionados. Para isso, duas estratégias são possíveis: selecionar os contextos dos LFs sequencialmente (assim como ocorrerá no codificador de vídeo) ou aleatoriamente.

Conforme pode ser visto no primeiro recorte de um LF na figura, regiões próximas possuem MIs muito similares umas às outras que apresentam padrões como gradientes, o que é uma característica fundamental que esta tese visa explorar. Contudo, se a cada exemplo utilizado para treinar a rede, o algoritmo mandar um contexto com um deslocamento de, por exemplo, 32 ou 16 píxeis, este novo exemplo será muito

semelhante ao anterior. Essa semelhança pode prejudicar o treinamento da rede por não prover novas informações ainda no mesmo *batch*. Podemos dizer que tornaria o aprendizado dos *batchs* muito homogêneos, provavelmente causando *overfitting* e necessitando mais exemplos para o treinamento. Ao se selecionar os exemplos de treino em posições aleatórias do LF, essa heterogeneidade nas informações aumenta, sendo necessário menos contextos para treino enquanto o *overfitting* tende a diminuir. Gostaríamos de ressaltar que não apresentamos este resultado em específico na Seção 6 por ser um processo já consolidado em treinamentos de redes neurais e um recorte muito pequeno das técnicas investigadas. Resumidamente, nossos experimentos mostraram que, ao selecionar os contextos de maneira aleatória, a rede generalizou melhor e foram necessários apenas 25% do total de contextos de cada LF no conjunto de treino para realizar o treino. Desta maneira, selecionamos contextos em posições aleatórias dos LFs, e utilizamos 25% do total de contextos de cada um dos LFs de treino e validação em cada época de treino. Vale ressaltar que os contextos podem se sobrepor.

Ainda, outro aspecto importante nesta seleção aleatória é utilizar passos que sejam múltiplos dos tamanhos das MIs, considerando sempre uma grade de $N \times N$ píxeis, onde N é o tamanho das MIs dos LFs, isto para que estes contextos estejam sempre alinhados com as MIs. O que já ocorrerá naturalmente durante a codificação em vista que este N será um múltiplo de 8 e alinhado com os tamanhos de blocos do codificador devido à preparação dos dados discutida na subseção anterior. Desta maneira, as entradas da rede no treino e na codificação possuem sempre a mesma estrutura.

Para proporcionar ainda maior heterogeneidade no treinamento por meio de aleatoriedade nos exemplos, após um exemplo ser selecionado, este é rotacionado aleatoriamente em $0°$, $90°$, $180°$ ou $270°$. Cada ângulo de rotação possui uma chance igual de ser selecionado, dando assim 25% de chance para cada um ocorrer sobre um exemplo de entrada. Após a rotação, o contexto tem 50% de chance de ser rotacionado horizontalmente antes de ter seu bloco definido como objetivo da rede e escurecido. Note que estas operações proporcionam todas as alterações possíveis perante as MIs, enquanto mantêm a correlação angular das vistas. Ainda, é importante manter ângulos de $90°$ para que o formato quadricular do contexto se mantenha tal qual nos codificadores.

Após estas operações de *data augmentation*, cabe ainda definirmos os hiperparâmetros do treinamento como a *loss function* e o *learning rate*, que também são essenciais para evitar o *overfitting* e garantir a convergência do treino.

4.3.2 Loss Function

No treinamento de uma rede neural, a *loss function*, que pode ser traduzida para função de perda, é uma equação que determina o erro entre a saída da rede e o seu

objetivo ideal (*ground truth*). A distância entre a saída e o objetivo então é utilizada como referência em conjunto com o *learning rate* para atualizar os pesos após uma iteração. Uma das funções de perda amplamente aplicadas em CNNs para *inpating*, predição e síntese de imagens é o MSE (Mean Squared Error - Erro Médio Quadrático) (SPADARO et al., 2023). O MSE entre duas imagens ou entre dois blocos como neste caso, pode ser obtido através da equação 7 onde se eleva ao quadrado a soma dos erros entre cada pixel do bloco predito I_{pred} e do bloco de *ground truth* $I_{\text{gt}}(i,j)$ e divide-se o resultado pelas dimensões do bloco A (altura) e L (largura) para obter o resultado médio dos erros residuais.

$$\text{MSE} = \frac{1}{A \cdot L} \sum_{i=1}^{A} \sum_{j=1}^{L} \left(I_{\text{gt}}(i,j) - I_{\text{pred}}(i,j)\right)^2 \tag{7}$$

Além da métrica MSE, também avaliamos outra função de perda que visa aproximar o treinamento da rede de seu ambiente de execução posterior por serem utilizadas internamente em codificadores de vídeo. A SATD (*Sum of the Transformed Absolute Differences* - Soma das Diferenças Absolutas Transformadas) definida pela equação 8, similarmente ao MSE, calcula os erros residuais pixel a pixel entre o bloco predito e o objetivo. Contudo, os erros são ainda transformados para o domínio das frequências através da transformada de *hadamard* definida pela equação 9. Posteriormente, é então aplicada a operação de absoluto para deixar os resultados positivos ao invés de elevá-los ao quadrado. Após os absolutos, os erros são finalmente somados para gerar o resultado da SATD. Para melhor quantificar o impacto da aplicação das transformadas, também avaliamos apenas a métrica SAD (*Sum of the Absolute Differences*), que segue a mesma equação da SATD porém sem a aplicação das transformadas (H).

$$\text{SATD} = \sum_{i=1}^{A} \sum_{j=1}^{L} \left|H \cdot (I_{\text{gt}} - I_{\text{pred}})\right| \tag{8}$$

$$H_1 = \begin{bmatrix} 1 \end{bmatrix}, \quad H_{2^n} = \begin{bmatrix} H_{2^{n-1}} & H_{2^{n-1}} \\ H_{2^{n-1}} & -H_{2^{n-1}} \end{bmatrix}. \tag{9}$$

4.3.3 Learning Rate

Para o treinamento das Redes Neurais Convolucionais é necessário estabelecer alguns parâmetros e técnicas que determinarão parte de seus comportamentos. Como, por exemplo, o *learning rate* (LR) dos treinamentos que determina o tamanho do passo dado na direção do gradiente durante o processo de atualização dos parâmetros do modelo (por exemplo, pesos e vieses), visando minimizar a função de perda. Neste trabalho, o LR utilizado inicialmente com o valor padrão de $10^{-}4$. Vale ressaltar que avaliamos diversos outros valores de LR entre $10^{-}3$ e $10^{-}5$. Contudo, esses valores

não convergiram, inviabilizando uma obtenção de BD-Rate.

Ao longo do treinamento a rede começa a encontrar os vários mínimos locais no espaço de busca, para forçar a busca nos melhores mínimos, de maneira similar a métodos como têmpera simulada (VAN LAARHOVEN et al., 1987) e algoritmos genéticos (MIRJALILI; MIRJALILI, 2019), é comum diminuirmos os passos dados pelo algoritmo ao longo do treinamento. Isto é possível reduzindo o *learning rate* nas épocas mais avançadas de acordo com uma heurística.

Duas técnicas de decaimento de *learning rate* foram avaliadas, decaimento dinâmico (*step decay*) e exponencial (KONAR; KHANDELWAL; TRIPATHI, 2020). O decaimento dinâmico é o decaimento padrão utilizado pelo otimizador Adam (KINGMA; BA, 2014) , onde a partir de 3 pesos $\beta 1$ (taxa de suavização para o primeiro momento), $\beta 2$ (taxa de suavização para o segundo momento) e ϵ (valor para estabilizar divisões), o otimizador adapta o LR adequadamente dividindo o treinamento em dois momentos: um com passos mais largos e um segundo momento com passos mais conservadores. Os pesos de $\beta 1$, $\beta 2$ e ϵ foram utilizados na sua configuração padrão com valores de $0,9$, $0,999$ e 10^{-8} respectivamente. Em vista dos vários mínimos locais presentes na autocorrelação dos LFs, foi avaliada também a técnica de decaimento exponencial do *learning rate* descrito pela equação 10. Onde o LR da próxima época é igual ao LR inicial multiplicado pelo DR (decay_rate - taxa de decaimento) elevado na época de treinamento atual. Avaliamos dois valores de *decay rate*, 0,2 utilizado por Spadaro et al. (2023) e 0,3, que apresenta um decaimento ligeiramente mais lento e possivelmente benéfico no tópico deste trabalho em vista de que LFs apresentam um espaço de busca mais complexo que imagens naturais.

$$\mathrm{LR}_{\text{época+1}} = \mathrm{LR}_0 \times \mathrm{DR}^{\text{época}} \tag{10}$$

Dos modos de treinamento possíveis, foi utilizado o treinamento em *mini-batch*, ou seja, as entradas do treinamento são agrupadas em pequenos lotes e a atualização dos pesos é realizada apenas ao término do processamento de todo o lote. Desta maneira, pode-se paralelizar os treinamentos na GPU e, quando todos terminarem, faz-se o ajuste dos pesos, permitindo a exploração máxima do paralelismo providenciado pela GPU. Valores maiores de *batch* tendem a gerar treinamentos mais eficientes em vista de considerar mais exemplos para atualizar os pesos da rede, com a capacidade de memória disponível no sistema sendo seu limite superior. Ainda, *batchs* demasiadamente grandes podem acabar implicando no efeito contrário, generalizando demasiadamente as atualizações dos pesos. Desta maneira, visando um equilíbrio nas atualizações de pesos, este trabalho utilizou um valor de *batch* de 64 conforme proposto em (SPADARO et al., 2023).

4.4 Topologias Propostas

Neste trabalho avaliamos diversas topologias, realizando alterações tanto em suas camadas como em seus *kernels*. Iniciamos nossas avaliações topológicas aplicando umar RNC similiar a proposta em (SPADARO et al., 2023) e definida na Figura 37. O autoencoder ilustrado na Figura 2 é composto por dois lados, um de codificação (*encoder*) e um de decodificação (*decoder*) da informação. A rede recebe como entrada pelo lado do *encoder* o contexto de tamanho $H \times W$, que inclui três blocos de referência de tamanho $\frac{H}{2} \times \frac{W}{2}$, mais um bloco (destacado em preto na figura) que corresponde à parte da imagem a ser preenchida (*inpainted*) i.e., o contexto disponível (vizinhança causal) durante a predição tanto no codificador quanto no decodificador de vídeo.

O autoencoder gera como saída um bloco de tamanho $H \times W$ que corresponde ao tamanho do contexto de entrada. Então, a área de interesse na saída da rede, o bloco inferior direito de tamanho $\frac{H}{2} \times \frac{W}{2}$, deve ser recortada e tomada como saída da rede. Este método se mostra mais eficiente do que predizer apenas um bloco $\frac{H}{2} \times \frac{W}{2}$ (SPADARO et al., 2023). No nosso caso de predição intra de *light fields* no formato lenslet, assumimos que cada bloco de $\frac{H}{2} \times \frac{W}{2}$ píxeis corresponde a uma única MI 8×8 ou duas MIs 16×16, ou seja, a partir do contexto de entrada, o autoencoder prediz um total de 4 a 8 MIs conforme o tamanho das mesmas.

Observando as camadas da arquitetura, verifica-se que o codificador (encoder) inclui 4 grupos de pares de camadas, cada um composto por uma camada convolucional com *stride* 1 (Conv/1) e outra com *stride* 2 (Conv/2) para o *downsampling* do mapa de características (*feature map*). Todas as camadas convolucionais utilizam *kernels* 3×3 e são seguidas por uma ativação PReLU. Após estas 4 camadas, tem-se uma camada adicional composta apenas por uma Conv/1 e ativação PReLU. Esta camada possibilita extrair as informações dos *feature maps* de tamanho $\frac{H}{16} \times \frac{W}{16}$ sem os reduzir ainda mais antes de serem aumentados pelo *upsampling* da camada posterior. O *feature map* gerado pela última camada do encoder é chamado de espaço latente. O espaço latente contém todas as informações retiradas da entrada da rede e, a partir deste, o *decoder* da rede é capaz de gerar a saída correspondente. As dimensões do espaço latente são $\frac{H}{16} \times \frac{W}{16} \times 512$, enquanto o decodificador (*decoder*) é composto por 4 camadas de convolução transposta com *stride* de 2px (píxeis) para *upsampling* (T.Conv/2). As três primeiras camadas transpostas são seguidas por uma ativação ReLU, e uma ativação final sigmoide determina as intensidades normalizadas dos píxeis de saída no canal de luminância (luma).

O primeiro experimento para adaptar esta rede para o contexto de predição de LFs foi alterar o tamanho de seus *kernels* 3×3 para 4×4 para aumentar o tamanho do seu campo receptivo (*receptive field*) em vista de que MIs possuem uma forte correlação e tamanhos de 8×8 e 16×16. Também avaliamos o impacto no desempenho da rede

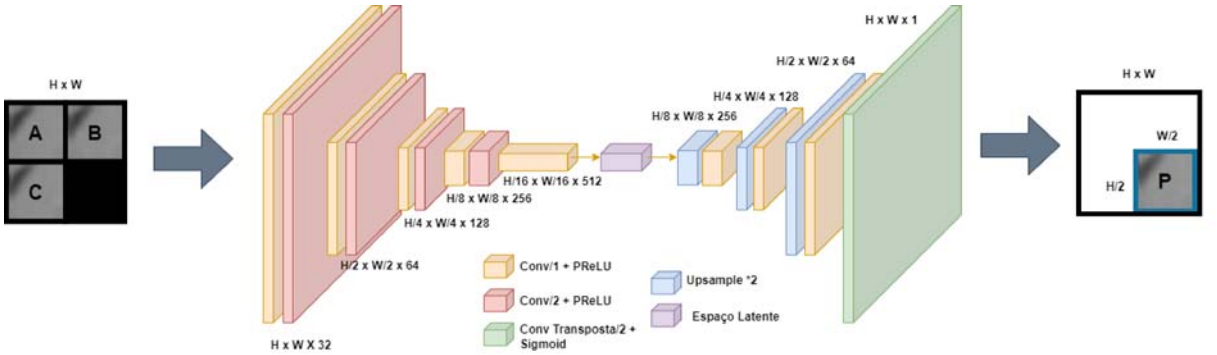


Figura 37 – Arquitetura autoencoder com camadas upsample e convolução convencional no decoder.
Fonte: Desenvolvido pelo Autor.

ao inserirmos conexões *SKIP* para transformá-la em uma Unet com a finalidade de melhor manter as informações estruturais entre o *encoder* e o *decoder*.

Além da arquitetura Unet, também avaliamos o desempenho de arquiteturas com conexões em *highway* e conexões residuais. As conexões em *highway*, conforme arquitetura acima na Figura 38, concatenam os *feature maps* de saída das camadas Conv/2 com a entrada da próxima camada Conv/2, permitindo assim que os *feature maps* sejam mantidos após a camada Conv/1 correspondente. Estas informações adicionais mantidas entre as camadas permitem com que informações estruturais dos LFs sejam mantidas e, consequentemente, possam ser também utilizadas pelas camadas mais profundas para produzir um resultado final superior. Observe que, como a saída de uma camada Conv/2 possui as mesmas dimensões que a entrada da próxima camada Conv/2, não são necessárias operações de redimensionamento no *feature map* sendo concatenado.

Já nas redes residuais as conexões *SKIP* são realizadas através da soma dos *feature maps* de saída de uma camada N com a saída da camada posterior $N+1$ através da operação de soma de matrizes "SUM". O que faz com que ambas as informações ainda estejam presentes, contudo, sem a necessidade do dobro de canais. Contudo, a soma preserva menos a separação das informações das camadas anteriores da atual, o que pode levar a perdas. Observe que em alguns casos os *feature maps* sendo somados não possuem o mesmo tamanho. Nestes casos, eles foram redimensionados por meio de convoluções com *kernel size* de 1 e *stride* de 2. Por fim, note também que em ambas as arquiteturas presentes na Figura 38 possuem conexões *SKIP* apenas na parte do *encoder*. Isso ocorre porque o *encoder* da arquitetura busca representar a entrada no espaço latente, então manter informações estruturais da entrada no espaço latente é interessante. Logo, para melhor manter estas informações, o *encoder* possui camadas com *stride* 1, que mantêm as dimensões da entrada inalteradas para a camada seguinte enquanto extrai características. Além de permitir que a entrada da camada anterior seja transmitida também para a posterior. Contudo, o mesmo não se

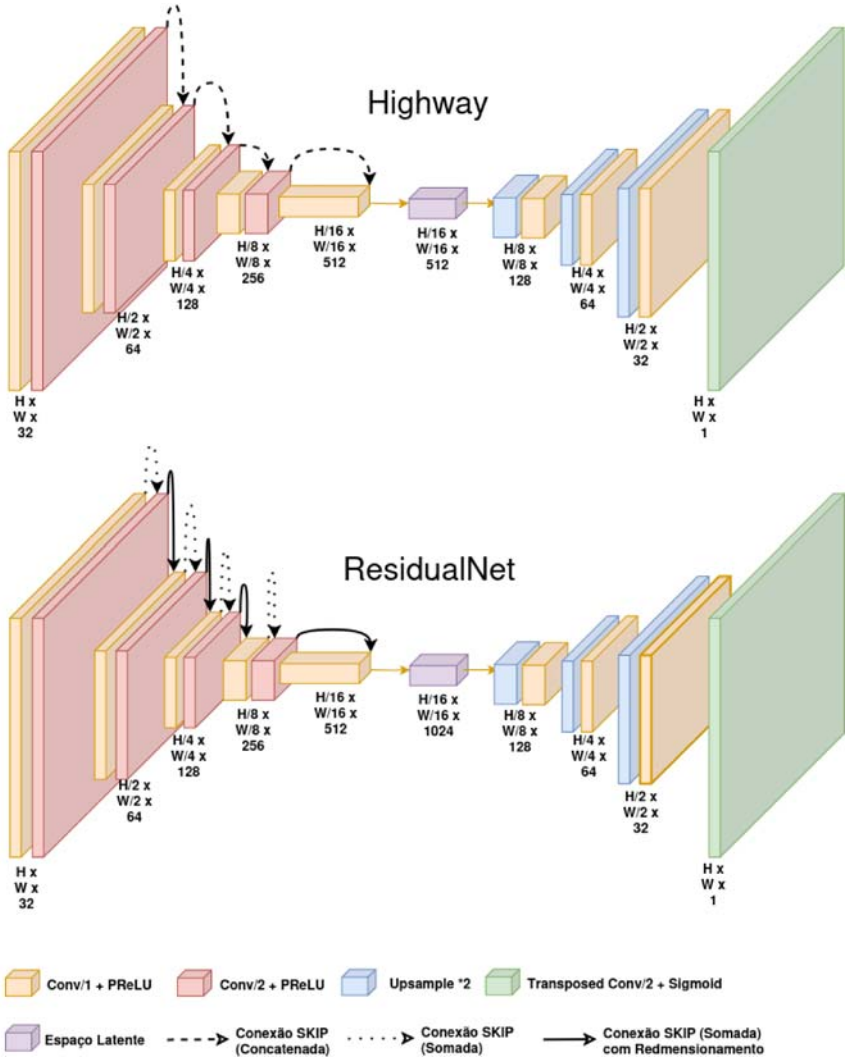


Figura 38 – Arquiteturas com conexões Highway e Residuais
Fonte: Desenvolvido pelo autor.

aplica ao *decoder*. Pois este é responsável por "decodificar" a informação de um espaço $\frac{H}{16} \times \frac{W}{16} \times 512$ em um contexto de saída em luma de tamanho $HxWx1$ e, portanto, manter informações estruturais de uma camada anterior mais próxima do formato de informação do espaço latente e mais distante da final não é interessante.

Em vista que esta rede possui cerca de 3,4M de parâmetros, reduzir seu tamanho também é um objetivo interessante para melhorar seu desempenho em hardware. Desta maneira, também avaliamos o impacto de remover a última camada do encoder que, por possuir 256 canais de entrada, 512 de saída e *kernels* 3×3, é responsável por mais de 1M de parâmetros, correspondendo a aproximadamente 30% do total de parâmetros da rede. Contudo, ressaltamos que ao remover esta camada, seu número de canais de saída deve ser repassado para a camada anterior, pois o número de canais passados para o espaço latente que é denominado de *bottleneck* é muito importante para a eficiência da rede. Nossos experimentos mostraram que diminuir o *bottleneck* pela metade cria perdas de eficiência impraticáveis. Logo, a remoção desta

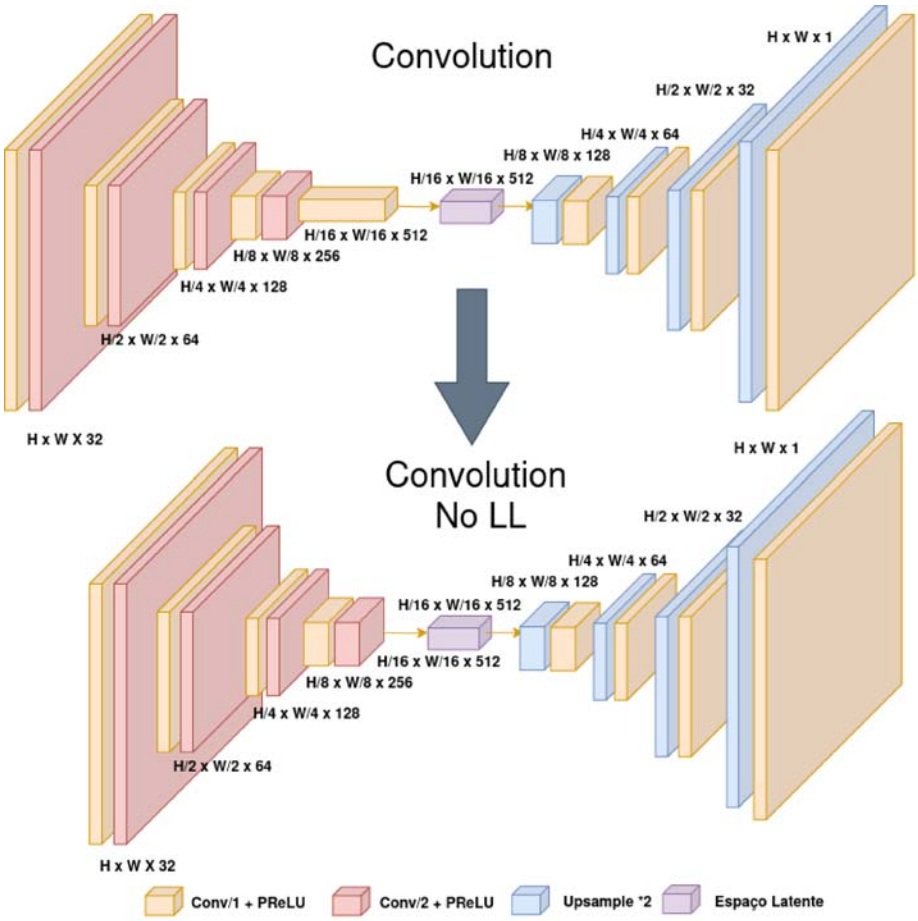


Figura 39 – Arquitetura autoencoder com apenas convoluções no decodificador com e sem última camada do encoder.
Fonte: Desenvolvido pelo autor.

camada implica na redução de aproximadamente 500 mil parâmetros ao invés de 1M. Para facilitar a compreensão, a arquitetura "Default" (Padrão) com a última camada removida será nomeada de "*Convolution No Last Layer*" (Convoluções Sem a Última Camada).

Ainda avaliamos quatro outras modificações topológicas alterando entre convoluções convencionais e compostas. No intuito de padronizarmos o decoder com apenas um tipo de convolução para facilitar a implementação da rede em hardware, primeiramente substituímos a convolução transposta do *decoder* por uma convolução convencional (Figura 39). Gerando assim a arquitetura denominada de "*Convolutions*" (Convoluções). Posteriormente, modificamos a rede retirando a última camada do *encoder* pelas mesmas razões mencionadas no parágrafo anterior. Após estas alterações, avaliamos também o impacto de trocar todos os conjuntos de *upsample* e convolução por uma camada de convolução transposta com *stride* 2 (Figura 40). Pois ao aplicar uma convolução transposta com *stride* 2, esta aumenta as dimensões dos *feature maps* similarmente a um *upsample* (vide Figura 22. Por fim, também avaliamos a remoção da última camada do *encoder*. A remoção desta camada se torna especi-

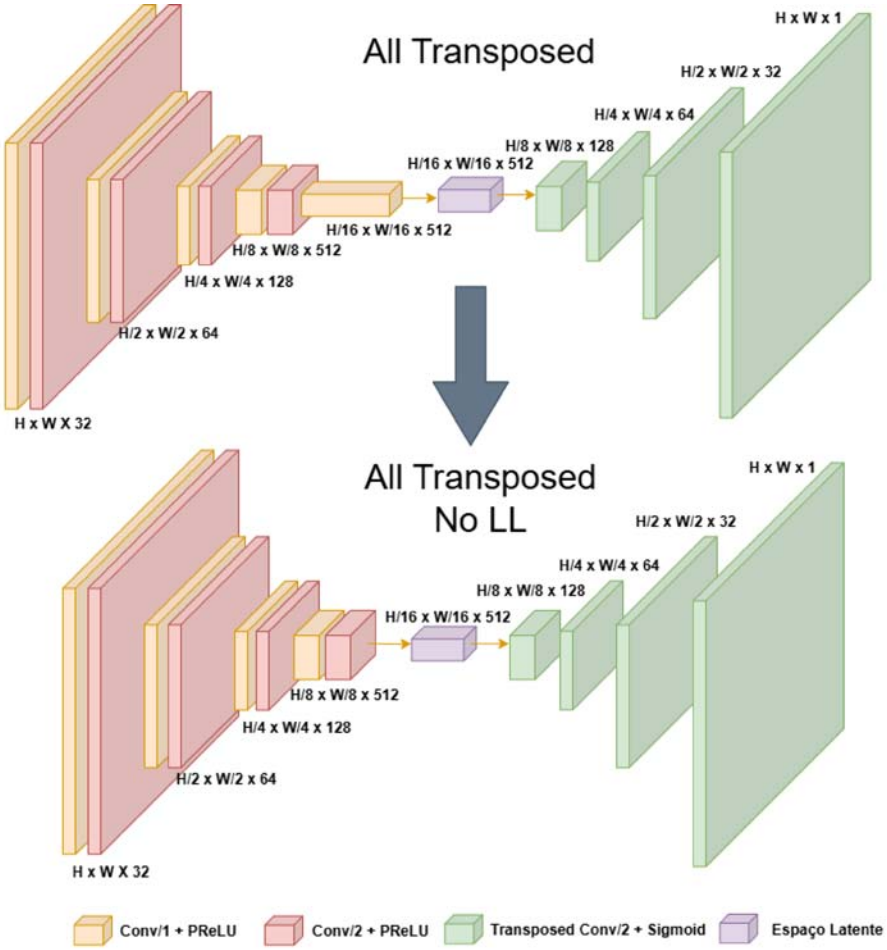


Figura 40 – Arquitetura autoencoder com apenas convoluções transpostas no decodificador com e sem última camada do encoder.
Fonte: Desenvolvido pelo autor.

almente interessante nesta arquitetura com convoluções transpostas em vista de que a primeira camada de deconvolução do *decoder* é capaz de avaliar o espaço latente com tamanho $\frac{H}{16} \times \frac{W}{16} \times 512$ sem que este tenha suas dimensões alteradas pela camada de *upsample*. Seguindo o padrão de nomenclaturas definido, nomeamos a arquitetura que utiliza apenas camadas convolucionais transpostas de "*Transposed Convolutions*" (Convoluções Transpostas) e "*Transposed Convolutions No Last Layer*" (Convoluções Transpostas Sem Último Layer).

Tabela 1 – Resumo das arquiteturas investigadas e seus números totais de parâmetros.

Topologia	Descrição	Camadas Encoder	Camadas Decoder	#Parâmetros
Padrão	Arq. com convoluções, e upsampling	4 (Conv/1+Conv2) + 1 (Conv/1)	3 (upsample/2 + Conv/1) + 1 (Conv.Trans./2)	3M
Highway	C/ conexões em Highway	5 (Conv/1+Conv2) + 1 (Conv/1) + Con. Highway	4 (upsample/2 + Conv/1) + 1 (Conv.Trans./2)	3.1M
Residual	C/ conexões Residuais	5 (Conv/1+Conv2) + 1 (Conv/1) + Con. Residuais	4 (upsample/2 + Conv/1) + 1 (Conv.Trans./2)	3M
Convolution	S/ a camada Conv.Transp./2 no dec.	5 (Conv/1+Conv2) + 1 (Conv/1)	5 (upsample/2 + Conv/1)	3M
ConvolutionNoLL	"Convolution" S/ camada final do enc.	5 (Conv/1+Conv2)	5 (upsample/2 + Conv/1)	2.4M
AllTransposed	Somente Conv.Transp./2 no decoder	5 (Conv/1+Conv2) + 1 (Conv/1)	4 (Conv/2)	3.5M
AllTransposedNoLL	"AllTransposed" S/ camada final do enc.	5 (Conv/1+Conv2)	4 (Conv/2)	3M

Neste capítulo, descrevemos os fluxos de treinamento e avaliação das redes neurais convolucionais propostas. Na etapa de treinamento das redes discutimos o *dataset* utilizado e os passos tomados para utilizá-lo para o treino e teste das redes. Tam-

bém foram discutidas as investigações referentes aos hiperparâmetros pertencentes à etapa de treinamento. Por fim, todas as arquiteturas desenvolvidas e investigadas foram detalhadamente abordadas e suas diferenças ressaltadas. Para facilitar consultas futuras, a Tabela 1 apresenta um resumo das arquiteturas estudadas e suas características. Os desempenhos de todas estas arquiteturas foram avalidas conforme o fluxo proposto no início deste capítulo e serão discutidos detalhadamente no capítulo a seguir.

5 RESULTADOS PRELIMINARES

Nesta seção serão discutidos os experimentos realizados para chegarmos nas melhores especificações encontradas para os treinamentos e arquiteturas. Inicialmente, executamos experimentos utilizando o codificador EVC como ferramenta de avaliação de suas eficiências, por este ser um codificador mais leve e rápido, o que nos permitiu avaliar um maior número de modificações em menos tempo. Nosso estudo exploratório começou avaliando diferentes tamanhos de *kernels* e conexões nas arquiteturas, assim como diferentes *loss functions* e aprimoramentos dos treinos. Após, movemos o estudo para o codificador HEVC, que é mais robusto e aplica ferramentas mais avançadas na sua codificação. Neste codificador, avaliamos não somente a permanência da eficiência obtida sobre o EVC, mas também outros aspectos, como mudanças no *dataset*, modo intra substituído, e técnicas de *pruning* sobre as redes.

Os LFs utilizados para treino e para avaliar os experimentos são os mesmos 12 definidos por (ZHONG et al., 2019) e utilizados para testes. Os treinos foram realizados sobre 100 épocas para cada etapa de *pruning*, sob o otimizador Adam (Adaptive Moment Estimation) (KINGMA; BA, 2014) e learning rate de $10^{-}4$ com decaimento exponencial sob lambda de 0,3. Como métrica de avaliação, utilizamos a métrica de BD-Rate, que determina a porcentagem de redução de *bitrate* dada uma qualidade fixa. Ou seja, quanto menor o valor, melhor o resultado para esta métrica. Também utilizamos como um fator de avaliação secundário o BD-PSNR, que dita a diferença de PSNR em dBs entre uma codificação proposta e uma original dado um *bitrate* fixo. Ou seja, valores negativos indicam perda de qualidade visual e valores positivos indicam melhora de qualidade visual. Resumindo, para BD-Rate desejam-se valores negativos e para BD-PSNR valores positivos.

5.1 Predição intra no codificador EVC: Avaliação Exploratória

Inicialmente realizamos experimentos sob a arquitetura *autoencoder* proposta por (SPADARO et al., 2023). Em um primeiro estudo avaliamos o impacto da alteração dos tamanhos de kernel de 3×3 por 4×4 que também são usados na literatura.

Ainda, avaliamos a aplicação de conexões *SKIP*, transformando o *autoencoder* em uma arquitetura Unet que, por sua vez, deve manter um maior grau de informações estruturais entre o encoder e o decoder. Como informação complementar, considere que a arquitetura com *kernels* 4×4 tem 3,5M de parâmetros e a com *kernels* 3×3 possui 3,1M de parâmetros, enquanto suas versões Unet possuem 200mil e 100mil parâmetros a mais devido às conexões *SKIP* adicionais.

5.1.1 Tamanhos de Kernel e Conexões em U

A Tabela 2 apresenta os resultados obtidos para ambos tamanhos de kernel e arquiteturas em relação a eficiência de compressão atingida pelo EVC sem modificações. Avaliando inicialmente os resultados de BD-Rate, observa-se que para a arquitetura de *autoencoder* utilizando *kernels* 4×4 obtém-se um BD-Rate de -12,14%, ou seja, para uma mesma qualidade de imagem, o *bit-rate* (quantidade de *bits* necessários para armazenar a imagem) é reduzido em 12,14%. Ao transformar a arquitetura em uma Unet aplicando conexões *SKIP* simetricamente entre o encoder e o decoder, o desempenho da rede é degradado em 1,11%. Já os *kernels* com tamanho 3×3 apresentam uma eficiência de compressão de -15,59% para o *autoencoder* e de -12,59% para a Unet. Observe que este tamanho de kernel excede os ganhos do 4×4 em 3,45% e 1,56% para o *autoencoder* e a Unet respectivamente. Por fim, note que para este tamanho de kernel a Unet se mantém menos eficiente do que a arquitetura menos complexa de *autoencoder* em 3%. Desta maneira, é possível concluir que para o nosso objeto de estudo, *kernels* 3×3 são mais eficientes que 4×4 e que as conexões *SKIP*, embora mais custosas, não apresentam benefícios.

Ao analisarmos os LFs individualmente, podemos observar que *Ceiling-Light*, *Black-Fence* e *Reeds* são os que apresentam os menores ganhos de -1%, -2,75% e -4,89% respectivamente. Já os LFs que apresentam maiores reduções são o *Swans-2*, *Slab-Lake* e *Bikes* com BD-Rates de -24,08%, -22,81% e -18,59% respectivamente. O comportamento destes 6 LFs, sendo 3 deles os maiores ganhos e 3 os menores ganhos, se manterá verdadeiro para a maioria dos experimentos, e evidenciaremos quando houver divergências.

Esses comportamentos se justificam especialmente pelas suas características angulares e de textura. Conforme pode ser observado na Figura 34, enquanto os LFs *Vespa, Swans-2* e *Slab-Lake* possuem áreas homogêneas e com mudanças graduais de profundidade como um lago indo ao horizonte, *Ceiling-Light* possui um lustre com diversos cristais pequenos com diferentes profundidades entre eles. Similarmente, *Reeds* possui finos matos próximos à câmera com um distante mar ao fundo e *Black-Fence* possui uma cerca com diversas aberturas para prédios distantes.

Ao direcionarmos a análise para a métrica de BD-PSNR podemos notar que a arquitetura de *autoencoder* com *kernels* 3×3 apresentando um BD-PSNR de 0,64dB,

Tabela 2 – Avaliação dos tamanhos de kernel e Conexões SKIP (BD-Rate e BD-PSNR em relação ao EVC)

Predictor / Sequence	Kernels 4x4		Kernels 4x4 Unet		Kernels 3x3		Kernels 3x3 Unet	
	BD-Rate	**BD-PSNR**	**BD-Rate**	**BD-PSNR**	**BD-Rate**	**BD-PSNR**	**BD-Rate**	**BD-PSNR**
Ankylosaurus-Diplodocus	-13.86	0.37	-12.3	0.33	-18.23	0.5	-14.79	0.39
Bikes	-18.59	0.9	-16.66	0.8	-23.12	1.15	-19.44	0.93
Black-Fence	-2.75	0.08	-2.67	0.08	-4.34	0.13	-2.93	0.09
Ceiling-Light	-1	0.03	-0.37	0.01	-1.68	0.04	-0.89	0.02
Danger-de-Mort	-9.4	0.48	-10.16	0.52	-11.81	0.6	-10.38	0.52
Friends-1	-6.43	0.22	-3.99	0.14	-10.82	0.39	-7.46	0.26
Houses-Lake	-9.27	0.24	-7.96	0.21	-14	0.38	-11.23	0.3
Reeds	-4.89	0.14	-3.09	0.09	-7.89	0.23	-6.95	0.2
Rusty-Fence	-11.33	0.53	-8.68	0.41	-14.54	0.7	-11.03	0.51
Slab-Lake	-22.81	1.16	-21.7	1.09	-25.94	1.34	-22.5	1.13
Swans-2	-24.08	0.93	-24.61	0.96	-29.61	1.2	-21.79	0.83
Vespa	-21.28	0.89	-20.15	0.84	-25.08	1.07	-21.64	0.89
AVG	-12.14	0.5	-11.03	0.46	**-15.59**	**0.64**	-12.59	0.51

ou seja, para *bitrates* fixos, a qualidade da imagem melhorou em 0,64dB. Já a arquitetura de *autoencoder* com *kernels* 4×4 obteve um BD-PSNR de 0,5dB. Ainda, as arquiteturas Unet apresentaram degradação de 0,4dB a 0,7dB em relação aos *autoencoders*. Desta forma, podemos afirmar que os comportamentos médios se mantêm os mesmos para esta métrica, com *kernels* 3×3 a arquitetura de *autoencoder* se mostrando mais eficientes. Gostaríamos de ressaltar que avaliamos também *kernels* 2×2, contudo os resultados não foram positivos.

Uma vez que a arquitetura de *autoencoder* com *kernels* 3×3 se mostrou mais eficiente, esta será a arquitetura padrão utilizada e denominada de "*Default*" em todos os resultados discutidos a seguir.

5.1.2 Avaliações do Bloco de Treinamento

Após as conclusões mencionadas, avançamos para aperfeiçoar o treinamento da rede *autoencoder* com *kernels* 3×3. Para isto, aplicamos dois conceitos de *data augmentation*. Primeiramente, começamos a selecionar os blocos de entrada da rede com posições aleatórias no LF, o que nos provê uma maior aleatoriedade no treino e, consequentemente, melhora a generalização e o *overfitting* da rede. Nossos experimentos mostraram também que selecionar 25% dos blocos presentes em um LF não só diminui o tempo de treinamento em aproximadamente 75%, mas também melhora o desempenho do treinamento, em vista de que o espaço amostral dos exemplos entre as épocas do treinamento possui uma maior distância entre si. A segunda técnica de *data augmentation* aplicada foi a de aplicar transformadas de rotação nos blocos selecionados com ângulos de 90ř, 180ř e 270ř, e espelhar horizontalmente o bloco, assim obtendo-se todas as diferentes representações de um bloco quadrado rotacionado e espelhado. Para maior aleatoriedade no treinamento, cada uma destas duas operações de transformação tem uma chance de 50% de ocorrer.

Em vista de que a técnica de *data augmentation* tem foco em melhorar o comportamento e o *overfitting* do treinamento, é importante avaliar não somente o BD-Rate

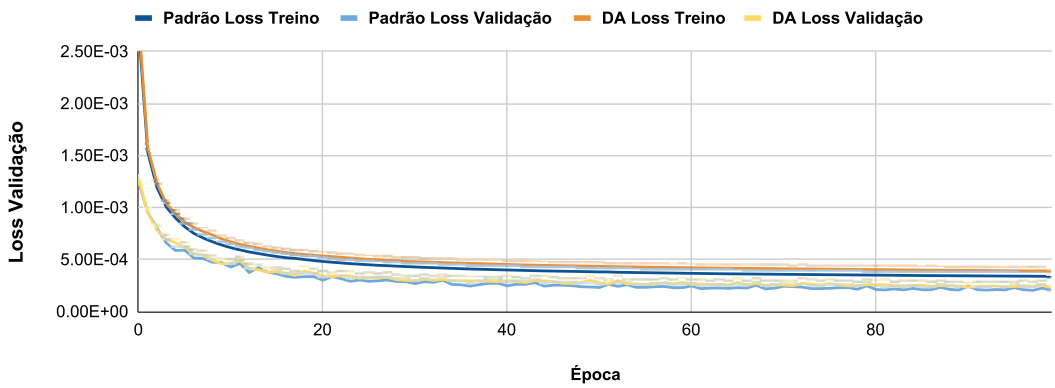


Figura 41 – Comparação das curvas de Loss para a o treinamento padrão e com data augmentation.
Fonte: Desenvolvido pelo autor.

obtido pelo processo, mas também o comportamento do treinamento. A Figura 41 mostra os valores de *loss* da arquitetura treinada com a aplicação de *data augmentation* (curvas azuis) e com o treinamento padrão até aqui (curvas laranjas). Para melhor observarmos o overfit, as curvas são apresentadas tanto para cada época do treinamento (cores mais escuras) quanto para cada época da validação (cores mais claras).

À primeira vista o gráfico pode passar a impressão de que aplicar a técnica está piorando o treinamento por causar uma *loss* superior tanto para o treino quanto para a validação. Contudo, ao treinar uma rede, o comportamento da *loss* além do seu valor absoluto também importa. Um indício de que as redes não estão tendo *overfitting* é que as curvas de validação estão abaixo das curvas de treino. Apenas observando o gráfico torna-se difícil visualizar qual curva de validação está mais distante da respectiva curva de treino. Contudo, ao calcularmos a média das distâncias entre as curvas de treino e validação para cada época em ambos os casos, obtém-se uma distância de $1,66 * 10^{-4}$ para o treino padrão e de $1,91 * 10^{-4}$ para o treinamento modificado. O que indica uma melhor generalização da rede treinada com as técnicas de *data augmentation* em vista desta aumentar heterogeneidade das informações presentes no treino.

Ainda, embora seja possível visualizar que a curva de validação com *data augmentation* é mais suave do que a do treinamento padrão, para avaliar este comportamento do aprendizado de maneira objetiva é importante avaliarmos também o desvio padrão de ambas as curvas. Embora a curva utilizando *data augmentation* possua valores maiores, seu desvio padrão é de $1,54 * 10^{-4}$ enquanto o do treinamento padrão é de $1,57 * 10^{-4}$. De maneira geral, um desvio padrão menor é desejável por demonstrar um treino mais estável.

Como os resultados obtidos ainda são inconclusivos em relação à técnica sendo avaliada, analisamos também a entropia de *Shannon* calculada pela biblioteca *python*

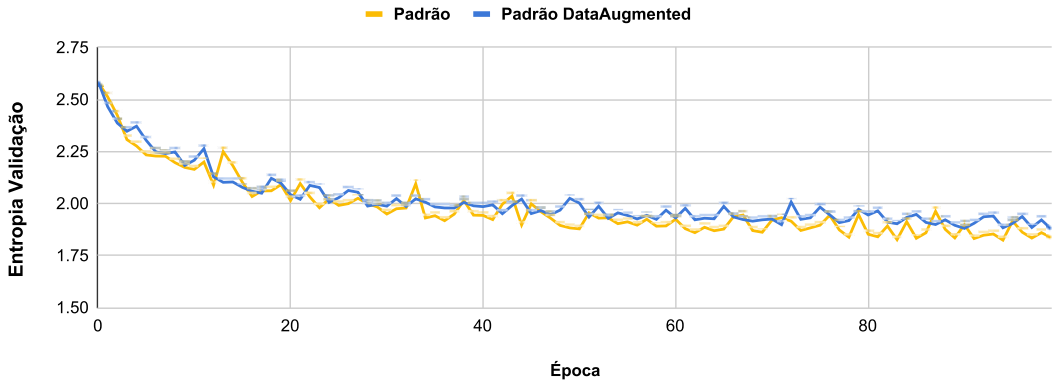


Figura 42 – Comparação das curvas de Loss na validação para a o treinamento padrão e com data augmentation.
Fonte: Desenvolvido pelo autor.

Skimage sobre os resíduos gerados pelo bloco predito. Teorizamos que também é interessante avaliar esta métrica ao longo do treinamento em vista de que o codificador explore a baixa entropia dos resíduos gerados pelas suas predições.

Analisando a Figura 42 podemos observar que a entropia do treinamento também é levemente superior, porém mais suave. Os desvios-padrões da entropia de ambas as curvas são de $1,48 * 10^{-1}$ para o treinamento padrão e de $1,33 * 10^{-1}$ para o treinamento modificado. Embora o desvio padrão da entropia tenha sido melhorado em ~10%, é necessário analisar a eficiência de codificação obtida por cada treinamento para concluirmos se a melhora no desvio padrão é mais interessante ou resultados absolutos mais baixos.

Para possibilitar esta conclusão, codificamos os LFs de teste sob a predição de ambas as redes e extraímos também seus BD-Rate e BD-PSNR que se encontram na Tabela 3. Podemos notar que a utilização destas técnicas melhorou o BD-Rate e BD-PSNR médios em 0,74% e 0,04dB respectivamente. Embora os ganhos médios tenham sido positivos, vale ressaltar que os 3 piores casos apresentaram perdas irrisórias: *Ceiling-Light* de 0,1%, *Black-Fence* de 0,58% e *Reeds* de 0,22%. Este ganho de 0,74%, embora pequeno, pode ser considerado suficiente para concluir que a técnica de *data augmentation* providenciou ganhos no treinamento da rede, e que o desvio padrão das curvas de *loss* e entropia ao longo do treinamento pode ser uma métrica para análise mais eficiente do que apenas os valores absolutos destas curvas. Ademais, um treinamento homogêneo faz com que o modelo da última época de treinamento tenha uma maior tendência de ser a instância mais eficiente ou, ao menos, esteja mais próximo da instância mais eficiente. Este comportamento reduz a necessidade da custosa avaliação da eficiência de codificação (BD-Rate e BD-PSNR) de modelos intermediários do treinamento, em vista de que uma *loss* menor não necessariamente indica um modelo mais eficiente. Embora não esteja no escopo desta tese realizar uma análise estatística sobre a homogeneidade da eficiência das redes ao longo das

Tabela 3 – Avaliação do uso de Data Augmentation para a rede "Padrão"(BD-Rate e BD-PSNR em relação ao EVC).

Preditor / Sequência	Padrão		Padrão DataAugmention	
	BD-Rate	**BD-PSNR**	**BD-Rate**	**BD-PSNR**
Ankylosaurus	-18,23	0,5	-18,31	0,5
Bikes	-23,12	1,15	-23,81	1,17
Black-Fence	-4,34	0,13	-3,76	0,12
Ceiling-Light	-1,68	0,04	-1,58	0,04
Danger-de-Mort	-11,81	0,6	-15,52	0,8
Friends-1	-10,82	0,39	-11,06	0,39
Houses-Lake	-14	0,38	-12,87	0,34
Reeds	-7,89	0,23	-7,67	0,23
Rusty-Fence	-14,54	0,7	-16,04	0,76
Slab-Lake	-25,94	1,34	-27,22	1,4
Swans-2	-29,61	1,2	-31,18	1,25
Vespa	-25,08	1,07	-26,93	1,15
Média	-15,59	0,64	-16,33	0,68

épocas do treinamento perante as técnicas sendo aplicadas, nossos experimentos também demonstraram que aplicar *data augmentation* implicou em um aprendizado mais linear das redes, o que levou a última época do treino a conter regularmente a instância com melhor eficiência.

Após aperfeiçoarmos como as amostras dos LF são selecionadas no treinamento, experimentamos diferentes funções de perda, por esta ser a próxima camada do treino responsável por determinar a eficiência da rede sobre as amostras tanto no treino quanto na validação. Avaliamos inicialmente o impacto de calcularmos a *loss* em volta de todo o contexto ao invés de apenas o bloco de interesse (coluna MSE Contexto) assim como outras métricas, a métrica SAD no domínio dos píxeis e SATD no domínio das frequências (eq. 8).

Ao avaliarmos os ganhos médios da métrica MSE calculada somente no bloco de interesse e sobre todo o contexto predito, podemos concluir que a técnica inicial apresenta um BD-Rate 0,89% superior. Embora esta estratégia apresente melhores ganhos para 2 dos casos mais difíceis (*Black Fence* e *Ceiling-Light*), assim como para *Friends-1* e *Swans-2*, estes ganhos são pequenos comparados às demais perdas. Desta forma, não avaliamos esta estratégia para as demais métricas. Observa-se também que a métrica SAD supera a métrica MSE em 0,44%. Contudo, o maior ganho se apresenta ao utilizar a métrica SATD, atingindo um BD-Rate de -17,6%, excedendo a métrica MSE em -1,3%. Para BD-PSNR, todos os comportamentos se mantêm similares, com a métrica SATD superando a MSE em 0,2dBs, atingindo assim um total de 0,86dBs.

Determinar qual *loss* function possui melhor eficiência de codificação é de grande importância para poder-se analisar as técnicas de decaimento do *learning rate* ao longo do treinamento. Isto porque a *loss function* apresenta impacto direto na taxa

Tabela 4 – Avaliação de diferentes Loss Functions para a rede "Padrão" (BD-Rate e BD-PSNR em relação ao EVC).

Sequências	MSE		MSEContext		SAD		SATD	
	BD-Rate	**BD-PSNR**	**BD-Rate**	**BD-PSNR**	**BD-Rate**	**BD-PSNR**	**BD-Rate**	**BD-PSNR**
Ankylosaurus	-18,31	0,5	-18,43	0,51	-18,76	0,51	-19,45	0,55
Bikes	-23,81	1,17	-23,03	1,13	-23,89	1,18	-25,01	1,26
Black-Fence	-3,76	0,12	-3,9	0,12	-3,97	0,12	-5,16	0,16
Ceiling-Light	-1,58	0,04	-1,76	0,05	-1,91	0,05	-1,89	0,05
Danger-de-Mort	-15,52	0,8	-13,2	0,68	-16,02	0,83	-17	0,9
Friends-1	-11,06	0,39	-11,74	0,42	-11,34	0,4	-12,07	0,43
Houses-Lake	-12,87	0,34	-10,1	0,27	-13,7	0,37	-15,92	0,44
Reeds	-7,67	0,23	-4,79	0,14	-8,09	0,24	-9,09	0,27
Rusty-Fence	-16,04	0,76	-14,46	0,69	-16,24	0,78	-16,73	0,82
Slab-Lake	-27,22	1,4	-25,51	1,31	-28,71	1,48	-29,34	1,56
Swans-2	-31,18	1,25	-33,71	1,4	-31,12	1,27	-32,43	1,35
Vespa	-26,93	1,15	-24,65	1,04	-27,48	1,18	-27,08	1,18
AVG	-16,33	0,68	-15,44	0,65	-16,77	0,7	-17,6	0,75

de aprendizado da rede e, portanto, diferentes *loss functions* vão responder melhor ou pior a diferentes decaimentos de *learning rate*. Ainda, vale mencionar que, apesar de termos experimentado *learning rates* iniciais maiores e inferiores que 10^{-4} como faixas entre 10^{-3} e 10^{-5}, assim estas faixas junto da utilização do automatizador SGD (*Stocastic Gradient Descent*) ao invés do Adam, contudo, estes experimentos não serão abordados em vista de que seus treinamentos não convergiram.

Até então o treinamento estava sendo realizado com decaimento por passos, onde a cada 30 passos o *learning rate* era multiplicado por 0,1, ou seja, era diminuído em 90%. Para aperfeiçoar a busca da rede no espaço amostral, avaliamos também o uso do decaimento de *learning rate* exponencial (*Exponential Decay*- ED).

Para facilitar a compreensão das camadas de técnicas sendo mantidas na metodologia nos resultados a seguir, manteremos uma sigla para cada uma ao longo das tabelas. Desta maneira, atualmente estamos utilizando a arquitetura *Default* sendo treinada com *Data Augmentation* (DA) e a métrica SATD (S), logo, esta será representada pela sigla DDAS.

A Tabela 5 apresenta os resultados do treino atualmente sendo utilizado com decaimento linear (DDAS) assim como com decaimento exponencial (ED) com os *decay rates* de 0,2 e 0,3. Podemos observar que o decaimento de *learning rate* exponencial provê ganhos significativos ao treino, com o decaimento mais agressivo obtendo uma melhora de -2,43% enquanto o mais conservador obteve -1,63%. Observe também que esta técnica de decaimento beneficiou os dois piores casos (*Ceiling Light* e *Black Fence*) em -0,81% e -1,3% que, apesar de serem ganhos abaixo da média, representam um aumento na redução destes casos de quase 2x. Em relação ao BD-PSNR, também houveram ganhos médios de 0,02dBs e 0,04dBs respectivamente. Desta maneira, assim como esperado, esta técnica com o decaimento mais lento proporcionou os melhores ganhos dentre os avaliados para o caso de estudo deste trabalho.

Tabela 5 – Avaliação de diferentes decaimentos de learning rate para a rede "Padrão" (BD-Rate e BD-PSNR em relação ao EVC).

Predictor / Sequence	Kernels 3x3 DataAug.		LR Decay 0.2		LR Decay 0.3	
	BD-Rate	**BD-PSNR**	**BD-Rate**	**BD-PSNR**	**BD-Rate**	**BD-PSNR**
Ankylosaurus	-19,45	0,55	-21,87	0,62	-22,32	0,63
Bikes	-25,01	1,26	-27,1	1,39	-28,02	1,44
Black-Fence	-5,16	0,16	-5,86	0,18	-6,46	0,2
Ceiling-Light	-1,89	0,05	-2,47	0,06	-2,7	0,07
Danger-de-Mort	-17	0,9	-18,15	0,97	-18,79	1,01
Friends-1	-12,07	0,43	-13,9	0,51	-15,32	0,56
Houses-Lake	-15,92	0,44	-16,28	0,45	-17,54	0,48
Reeds	-9,09	0,27	-9,94	0,3	-9,93	0,3
Rusty-Fence	-16,73	0,82	-19,09	0,94	-19,24	0,95
Slab-Lake	-29,34	1,56	-31,16	1,67	-31,94	1,71
Swans-2	-32,43	1,35	-35,12	1,49	-36,83	1,58
Vespa	-27,08	1,18	-29,77	1,31	-31,21	1,39
AVG	-17,6	0,75	-19,23	0,82	-20,03	0,86

5.1.3 Topologias Alternativas

Após a avaliação dos aspectos relacionados ao treinamento da rede neural convolucional, atingiu-se um ganho de -4,44% de BD-Rate desde o método *default* que atingia -15,59% de redução até o método DDASED (*Default Data Augmented SATD Exponential Decay*) que atinge -20,03% de redução. O que evidencia a importância de analisar e avaliar diferentes estratégias para cada etapa do treinamento de uma rede neural ao se mudar o espaço de busca.

Considerando que a estrutura das MIs é crucial para a predição, avaliamos também o comportamento de conexões em *Highway* e conexões residuais como das Resnets. Contudo, como pode ser observado na Tabela 6, ambas as técnicas causaram perdas em torno de 3% de BD-Rate e 0,2dB de BD-PSNR. Gostaríamos de ressaltar que estas redes foram experimentadas sob outros valores de *learning rate*, porém também sem sucesso. Portanto, em vista de que estas técnicas são comumente aplicadas a imagens inteiras e não contextos de 64×64 píxeis, formulamos a hipótese de que elas não são capazes de se beneficiar da informação estrutural pelo fato de os contextos de entrada serem muito pequenos, mesmo possuindo um maior número de parâmetros.

5.1.4 Tamanhos de Blocos

Após definirmos a rede e a metodologia de treino mais eficientes para o caso de estudo desta tese, torna-se necessário validarmos as conclusões obtidas para os demais casos da codificação. Sendo assim, configuramos o codificador para utilizar nossa rede apenas para blocos 16×16, enquanto os demais tamanhos de bloco têm acesso somente aos preditores intra padrão. Esclarecemos que a diferença nos blocos em relação às MIs é que, ao invés de serem formados por 4 MIs de 16×16 pixels, agora serão formados por 1 MI 16×16. Para uma melhor validação de que o que foi

Tabela 6 – Avaliação das arquiteturas Highway e Residual em comparação com a Padrão (BD-Rate e BD-PSNR em relação ao EVC).

Preditor / Sequência	LR Decay 0,3		Highway Connection		Residual Connection	
	BD-Rate	**BD-PSNR**	**BD-Rate**	**BD-PSNR**	**BD-Rate**	**BD-PSNR**
Ankylosaurus	-22,32	0,63	-20,14	0,55	-19,43	0,54
Bikes	-28,02	1,44	-24,83	1,23	-24,32	1,21
Black-Fence	-6,46	0,2	-3,97	0,12	-4,27	0,13
Ceiling-Light	-2,7	0,07	-1,98	0,05	-1,92	0,05
Danger-de-Mort	-18,79	1,01	-14,15	0,72	-16,17	0,84
Friends-1	-15,32	0,56	-12,52	0,45	-11,97	0,43
Houses-Lake	-17,54	0,48	-15,27	0,41	-13,18	0,35
Reeds	-9,93	0,3	-9,45	0,28	-8,25	0,25
Rusty-Fence	-19,24	0,95	-16,58	0,79	-16,55	0,8
Slab-Lake	-31,94	1,71	-27,27	1,4	-29,22	1,53
Swans-2	-36,83	1,58	-32,18	1,31	-32,59	1,33
Vespa	-31,21	1,39	-25,71	1,09	-27,39	1,18
Média	-20,03	0,86	-17	0,7	-17,11	0,72

experimentado em blocos 32×32 também será verdade para blocos 16×16, treinamos e aplicamos a rede inicial sobre as métricas MSE e SATD, assim como experimentamos separadamente as camadas de Data Augmentation (DA) e *exponential decay* (ED) para este tamanho de bloco sobre a *loss* SATD (S), conforme apresentado na Tabela 7. Ainda, note que aqui acrescentamos um novo experimento de *context crop* (CC) antes da aplicação do *exponential decay* (ED) em vista de que o decaimento de *learning rate* é aplicado posteriormente a todas as transformações. Pois teorizamos que, embora o codificador EVC passe contextos 64×64 mesmo quando codificando blocos 16×16, as amostras de referência mais distantes podem não ser tão relevantes para o bloco predito e, portanto, recortar o contexto 64×64 em um contexto 32×32 formado pelas MIs vizinhas mais próximas pode facilitar o aprendizado da rede.

Analisando os resultados, podemos observar que, para este tamanho de bloco, a métrica SATD segue sendo mais eficiente que a MSE em -1,84% e que a aplicação de data augmentation segue importante, provendo -1,6% de redução. Ainda, recortar apenas as 16×16 amostras constituídas pelas 2 MIs 8×8 mais próximas apresentou ganhos significativos de 1,41%. Ao treinarmos a rede sob estas técnicas juntamente com o decaimento exponencial de *learning rate* obtemos uma redução média de $-27,12\%, 5,2\%$ superior à rede treinada com decaimento linear.

Perceba também que para blocos 16×16 além de todos os melhores casos apresentarem ganhos de até -9% como no caso da sequência *Slab Lake*, os piores casos também atingem ganhos consideráveis de -10,45% para o *Ceiling Light* (uma redução ~6x maior), -6,16% para o *Black Fence* (redução ~2x maior), e -5,31% (redução ~1,5x maior). Estes ganhos de eficiência de codificação demonstram que o baixo desempenho inicial destes casos de teste se deve também à sua complexidade de serem codificados e à necessidade da aplicação de múltiplos tamanhos de bloco para uma

Tabela 7 – Avaliação de diferentes técnicas para blocos 16×16 para a rede "Padrão" (BD-Rate e BD-PSNR em relação ao EVC).

Preditor / Sequência	MSE		SATD		DAS		DASCC 32×32		DASCCLR Decay 0,3	
	BD-Rate	**BD-PSNR**	**BD-Rate**	**BD-PSNR**	**BD-Rate**	**BD-PSNR**	**BD-Rate**	**BD-PSNR**	**BD-Rate**	**BD-PSNR**
Ankylosaurus	-21,51	0,61	-22,36	0,65	-23,45	0,69	-24,43	0,72	-28,17	0,88
Bikes	-23,58	1,17	-26,77	1,38	-28,67	1,51	-31,26	1,69	-36,97	2,15
Black-Fence	-6,08	0,19	-7,22	0,23	-8,44	0,27	-8,94	0,28	-12,62	0,41
Ceiling-Light	-6,6	0,16	-8,69	0,21	-9,29	0,23	-11,03	0,27	-13,15	0,33
Danger-de-Mort	-17,39	0,9	-20,26	1,1	-22,85	1,26	-23,9	1,34	-30,38	1,83
Friends-1	-14,82	0,54	-17,45	0,65	-17,74	0,67	-18,74	0,71	-22,4	0,89
Houses-Lake	-13,18	0,36	-11,91	0,32	-15,63	0,43	-15,89	0,44	-22,3	0,66
Reeds	-9,58	0,29	-9,49	0,29	-11,24	0,34	-11,77	0,36	-14,24	0,45
Rusty-Fence	-20,99	1,03	-22,27	1,13	-25,62	1,34	-26,75	1,42	-30,81	1,72
Slab-Lake	-24,89	1,28	-28,33	1,53	-28,91	1,57	-33,28	1,89	-40,52	2,49
Swans-2	-29,19	1,19	-31,35	1,31	-32,19	1,36	-33,37	1,43	-39,43	1,82
Vespa	-22,33	0,93	-26,06	1,14	-26,91	1,19	-28,49	1,28	-34,46	1,65
Média	-17,51	0,72	-19,35	0,83	-20,91	0,9	-22,32	0,99	-27,12	1,27

melhor predição de suas texturas e geometrias.

Como cada MI é composta por 8×8 píxeis, faz sentido predizer uma MI 8×8 a partir de um contexto 16×16 composto pelas três MIs vizinhas juntamente do bloco vazio sendo predito. Desta maneira, a Tabela 8 apresenta os mesmos experimentos da tabela anterior para blocos 8×8. Além da *loss function* SATD continuar apresentando superioridade em relação a MSE, para este tamanho de bloco os ganhos foram mais expressivos. Provendo uma melhora de -2,73% em relação à MSE, uma diferença ~2x maior que para blocos 16×16, o que evidencia que a SATD captura eficientemente a performance da rede mesmo quando existem apenas 64 píxeis para ser calculada, o que é uma quantidade de amostras consideravelmente pequena para ser avaliada. Já a *data augmentation*, pelo mesmo motivo, apresentou ganhos que excedem a rede inicial em apenas 0,7%, aproximadamente metade do ganho apresentado no contexto 2x maior. Assim, conclui-se que a eficiência da aplicação desta técnica cai quase que linearmente com o tamanho do contexto.

Posteriormente o contexto foi recortado de 64×64 para 16×16 ao invés de 32×32 como na Tabela 7. A utilização das MIs mais próximas do bloco sendo predito forneceu ganhos de -4,71%. Por fim, o *learning rate* com decaimento exponencial apresentou ganhos de -4,39%. Já para os piores casos, perceba que a rede DASCCED para blocos 8×8 atinge um BD-Rate de -19,09% (~10x maior que blocos para 32×32), excedendo a rede para blocos 16×16 em -6,54%. Enquanto para os piores casos *Black Fence* e *Reeds* apresentam perdas de 2,3% e 2,88%. Vale ressaltar que estas perdas ocorrem puramente devido às características específicas destes LFs, e que evidenciam a necessidade de utilizar múltiplos tamanhos de blocos quando predizendo MIs de LFs.

5.1.5 Múltiplos Tamanhos de Bloco em Conjunto

Após avaliarmos os tamanhos de bloco trabalhando separadamente, cabe avaliar os seus impactos quando utilizados em conjunto e competindo entre si.

Ao examinarmos os resultados médios da Tabela 9 verifica-se que ao utilizarmos

Tabela 8 – Avaliação de diferentes técnicas para blocos 8×8 para a rede "Padrão"(BD-Rate e BD-PSNR em relação ao EVC).

Preditor / Sequência	MSE		SATD		DAS		DASCC 16×16		DASCCLR Decay 0,3	
	BD-Rate	BD-PSNR	BD-Rate	BD-PSNR	BD-Rate	BD-PSNR	BD-Rate	BD-PSNR	BD-Rate	BD-PSNR
Ankylosaurus	-13,69	0,39	-18,87	0,52	-15,97	0,47	-20,48	0,62	-24,98	0,8
Bikes	-17,15	0,85	-22,77	1,13	-22,86	1,22	-29,08	1,6	-33,94	1,98
Black-Fence	-3,75	0,11	-3,86	0,12	-5,65	0,18	-7,59	0,24	-10,32	0,34
Ceiling-Light	-11,25	0,28	-2,28	0,06	-12,8	0,33	-13,37	0,35	-19,69	0,54
Danger-de-Mort	-19,24	1,1	-11,64	0,59	-23,33	1,44	-26,5	1,61	-31,86	2,05
Friends-1	-8,21	0,3	-12,42	0,45	-10,31	0,39	-12,2	0,46	-16,62	0,66
Houses-Lake	-5,15	0,14	-11,12	0,3	-9,52	0,26	-13,05	0,36	-15,35	0,44
Reeds	-5,7	0,17	-7,13	0,21	-7,25	0,22	-8,48	0,26	-11,36	0,36
Rusty-Fence	-17,15	0,88	-14,35	0,69	-20,31	1,11	-26,36	1,49	-31,19	1,86
Slab-Lake	-16,88	0,85	-24,28	1,24	-21,56	1,13	-33,19	1,92	-38,78	2,39
Swans-2	-16,56	0,63	-30,25	1,23	-22,86	0,93	-30,51	1,32	-35,01	1,6
Vespa	-15,79	0,65	-24,21	1,02	-18,72	0,81	-26,91	1,24	-31,31	1,52
Média	-12,54	0,53	-15,27	0,63	-15,93	0,71	-20,64	0,96	-25,03	1,21

Tabela 9 – Avaliação do impacto da integração dos preditores DASCCED sob os diferentes tamanhos de bloco (BD-Rate e BD-PSNR em relação ao EVC).

Preditor / Sequência	Pretitor 32×32		Preditor 32×32+16×16		Preditor 32×32+16×16+8×8	
	BD-Rate	BD-PSNR	BD-Rate	BD-PSNR	BD-Rate	BD-PSNR
Ankylosaurus	-22,32	0,63	-34,99	0,78	-32,7	1
Bikes	-28,02	1,44	-39,67	1,71	-41,88	2,44
Black-Fence	-6,46	0,2	-31,58	1,21	-10,42	0,33
Ceiling-Light	-2,7	0,07	-19,45	0,41	-21,89	0,58
Danger-de-Mort	-18,79	1,01	-32,37	1,43	-33,67	2,07
Friends-1	-15,32	0,56	-30,77	0,93	-28,13	1,13
Houses-Lake	-17,54	0,48	-28,11	0,74	-22,15	0,64
Reeds	-9,93	0,3	-18,8	0,49	-14,66	0,46
Rusty-Fence	-19,24	0,95	-36,56	1,6	-36,06	2,05
Slab-Lake	-31,94	1,71	-36,61	1,65	-40,16	2,36
Swans-2	-36,83	1,58	-43,86	1,54	-46,34	2,2
Vespa	-31,21	1,39	-35,5	1,25	-38,33	1,84
Média	-20,03	0,86	-32,36	1,15	-30,53	1,42

os tamanhos de bloco de 32×32 em conjunto com 16×16 obtém-se um ganho de -12,33%. Note que a junção dos preditores para os diferentes tamanhos de bloco não causa uma soma direta dos seus ganhos de eficiência, isto porque estes competem entre si para todos os blocos, fazendo com que blocos que, por exemplo, tenham selecionado o preditor proposto de tamanho 32×32 quando não havia a opção do 16×16, agora selecionem o 16×16, assim como o inverso. Por outro lado, mesmo predizendo apenas uma MI através de outras 3 MIs adjacentes, ao adicionarmos o preditor 8×8 em conjunto com os demais obteve-se uma perda de -1,83%. Contudo, note que em múltiplos casos como *"Ankylosaurus-1*, *"Friends-1"*, *"Reeds"* e *"Rusty-Fence"* há ganhos de BD-PSNR mesmo com perdas significativas de BD-Rate. Este comportamento acarretou que ao utilizar blocos 8×8 em média o codificador atingisse um BD-PSNR de 0,37dB, equivalente a uma melhora de 20% da métrica.

5.1.6 Manipulação dos Light Fields

Assim como para adaptar os tamanhos das MIs dos LFs para 8×8 descartamos parte de suas vistas, similarmente ao que foi proposto por Mukati et al. (2020), também é possível transformar os LFs para possuírem MIs 16×16. Estas vistas extras foram geradas por meio de *padding*, conforme descrito na Seção 4.3.1. Inicialmente, teoriza-

Tabela 10 – Eficiência do preditor DASCCED 32×32 em LFs com *padding* para 16×16 vistas (BD-Rate e BD-PSNR em relação ao EVC).

Sequências	16×16 Vistas	
	BD-Rate	BD-PSNR
Ankylosaurus	-34,53	0,76
Bikes	-39,4	1,69
Black-Fence	-31,48	1,21
Ceiling-Light	-19,23	0,41
Danger-de-Mort	-32,2	1,42
Friends-1	-30,37	0,91
Houses-Lake	-26,72	0,7
Reeds	-18,53	0,48
Rusty-Fence	-36,31	1,59
Slab-Lake	-36,56	1,64
Swans-2	-42,26	1,47
Vespa	-35,21	1,23
Média	-31,9	1,13

mos que as avaliações já feitas seriam similarmente benéficas, porém com menores reduções de BD-Rate. Isto devido à estrutura similar destas MIs às 8×8, porém com maior quantidade de informações para serem preditas. Contudo, analisando a Tabela 10, é possível concluir o contrário, que para LFs com MIs de 16×16 a eficiência de codificação foi melhorada para -31,9%, um acréscimo de -11,87% quando comparado ao método DASCCED 32×32 aplicados a vistas 16×16.

Terminadas as avaliações de todas as técnicas experimentadas para ambas as transformações do *dataset* (8×8 e 16×16), resta avaliarmos estas no HEVC, o qual é um codificador mais eficiente e com ferramentas mais atuais.

5.2 Predição Intra no Codificador HEVC

Ao mudarmos o codificador, novas questões surgem. Por exemplo, com um novo codificador que possui novas ferramentas, os LFs com MIs 16×16 seguem sendo mais eficientes? Ainda, nas alterações realizadas no HEVC, é possível escolher qual dos 35 modos intra iremos substituir pelo preditor, abrindo assim a possibilidade de avaliar qual modo seria melhor substituir.

Ainda, conforme mencionado anteriormente, assumimos que o desempenho das diferentes topologias e técnicas avaliadas no EVC se mantenha verdadeiro quando alterarmos o codificador. Desta maneira, não avaliamos ou ajustamos para o codificador HEVC o que foi proposto até aqui, mas sim analisamos a sua eficiência quando aplicada a este novo ambiente, também como uma forma de provar a flexibilidade do método sendo proposto.

Desta maneira, começamos nossos experimentos aplicando a arquitetura DASCCED 32×32 sob LFs 8×8 substituindo o modo DC do codificador, assim como proposto

por Spadaro et al. (2023), e o modo 18 por ser mais utilizado apenas em casos mais raros, conforme estudado por Chernyak (2014).

5.2.1 Modo Substituído

Ao avaliarmos inicialmente o preditor para o modo DC, notamos que os ganhos em BD-Rate são de -14,29%, um ganho de eficiência menor quando comparado aos -20,03% obtidos no EVC. Este resultado é esperado em vista de que a predição mais avançada do HEVC será mais competitiva com a proposta neste trabalho. Em relação ao modo sendo substituído, cada um apresenta ganhos e perdas para casos específicos. Desta forma, para melhor fundamentar uma escolha, realizamos um estudo estatístico em cima dos LFs onde dois LFs com características distintas foram codificados através do HEVC com e sem o nosso preditor, e a porcentagem de seleção de cada modo foi obtida.

Tabela 11 – Avaliação de modos intra do HEVC para LFs 8×8 para o preditor DASCCLR (BD-Rate e BD-PSNR em relação ao HEVC).

Sequência	DASCCLR Mode 1		DASCCLR Mode 18	
	BD-Rate	BD-PSNR	BD-Rate	BD-PSNR
Ankylosaurus	-14,46	0,39	-14,94	0,4
Bikes	-20,34	0,94	-20,24	0,94
Black-Fence	-4,92	0,15	-4,84	0,15
Ceiling-Light	-2,58	0,08	-2,67	0,08
Danger-de-Mort	-16,79	0,89	-16,65	0,88
Friends-1	-9,77	0,34	-10,05	0,35
Houses-Lake	-7,55	0,18	-7,96	0,2
Reeds	-5,34	0,14	-5,08	0,14
Rusty-Fence	-15,35	0,73	-15,28	0,73
Slab-Lake	-27,4	1,35	-27,51	1,35
Swans-2	-24,54	0,94	-24,57	0,94
Vespa	-22,48	0,89	-22,46	0,89
Média	-14,29	0,59	-14,35	0,59

5.2.2 Taxa de Escolha dos Modos Intra

A figura 43 mostra a taxa de uso dos preditores padrões do HM (Planar, DC e Angulares) e o preditor desenvolvido (Modo NN) para a referência e o HM modificado sob todos 4 QPs avaliados. Para o codificador de referência HM, os modos mais selecionados são Planar (0) e DC (1), escolhidos, respectivamente, em 31% e 25% das vezes (também verificamos que o modo 18 foi raramente utilizado, cerca de 0,08%). Contudo, ao substituir o modo 18 pelo nosso preditor treinável, este alcança uma taxa média de seleção de 77,16%, com 89,2% e 64,59% para cada sequência, respectivamente. Assim, os modos Planar e DC passam a ser escolhidos em apenas 5% dos casos, em média. Esses valores confirmam que, em média, nosso preditor treinável apresenta melhor desempenho RD em comparação aos outros modos, explicando os

ganhos de BD-Rate observados.

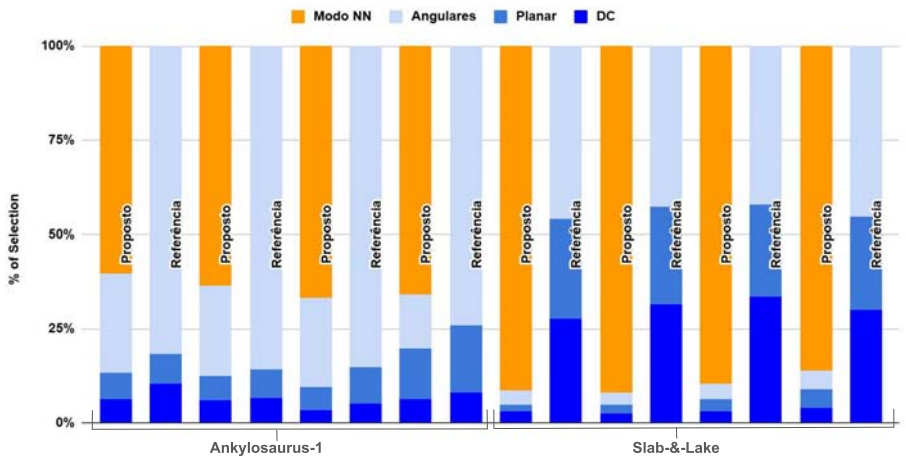


Figura 43 – Comparação da taxa de seleção de cada um dos modos intra do codificador HEVC com e sem o preditor DASCCED.
Fonte: Desenvolvido pelo autor.

Porém, em vista do ganho médio ter sido 0,06% superior e, estatisticamente, o modo 1 ser um dos modos mais utilizados pelo codificador, enquanto o 18 é empregado em menos de 1% dos blocos, os experimentos apresentados referentes ao HEVC a partir daqui serão todos substituindo o modo intra 18 do codificador.

5.2.3 Múltiplos Tamanhos de Bloco para Light Fields 8x8 Vistas

Após definirmos qual modo intra deve ser substituído no HEVC, cabe avaliar a performance dos preditores para os 3 tamanhos de bloco, utilizados separadamente e em conjunto para melhor entendermos os comportamentos apresentados no HEVC.

A Tabela 12 apresenta o BD-Rate e BD-PSNR obtidos pelos preditores para cada um dos três tamanhos de blocos avaliados separadamente para LFs 8×8. Desejamos esclarecer que, para realizar este experimento, o preditor do HEVC foi substituído apenas para um tamanho de bloco por vez, competindo assim com todos os outros modos intra e todos os outros tamanhos de bloco. Logo, o modo substituído (18) para os tamanhos de bloco não substituídos segue a interpolação padrão definida pelo HEVC.

Analisando a tabela, podemos concluir que mesmo as MIs possuindo dimensões de 8×8 píxeis, o tamanho de bloco 8×8 que prediz uma MI a partir das três MIs vizinhas (uma por bloco vizinho) apresentou o pior resultado dos três tamanhos de bloco com um BD-Rate de -13,01%, estando 1,34% abaixo mesmo do tamanho de bloco 32×32. Sendo assim, similarmente aos resultados obtidos no EVC, ao predizer LFs 8×8 (sem realizar o *padding*), o preditor mais efetivo foi o de tamanho 16×16

também no HEVC, atingindo um BD-Rate de -24,82%, aproximadamente 10,5% acima do tamanho de bloco 32×32.

Tabela 12 – Comparação de BD-Rate e BD-PSNR para os três tamanhos de blocos separadamente para LFs 8×8 com o preditor DASCCLR (BD-Rate e BD-PSNR em relação ao HEVC).

Sequência 8×8	32×32		16×16		8×8	
	BD-Rate	BD-PSNR	BD-Rate	BD-PSNR	BD-Rate	BD-PSNR
Ankylosaurus	-14,94	0,4	-26,05	0,76	-13,55	0,38
Bikes	-20,24	0,94	-35,16	1,82	-19,92	0,96
Black-Fence	-4,84	0,15	-13,53	0,43	-4,42	0,13
Ceiling-Light	-2,67	0,08	-15,55	0,48	-11,93	0,38
Danger-de-Mort	-16,65	0,88	-30,96	1,79	-20,74	1,2
Friends-1	-10,05	0,35	-19,86	0,74	-8,04	0,28
Houses-Lake	-7,96	0,2	-10,14	0,26	-2,77	0,07
Reeds	-5,08	0,14	-8,69	0,24	-0,99	0,03
Rusty-Fence	-15,28	0,73	-27,55	1,45	-17,79	0,91
Slab-Lake	-27,51	1,35	-41,58	2,31	-21,99	1,1
Swans-2	-24,57	0,94	-35,76	1,5	-19,76	0,77
Vespa	-22,46	0,89	-32,98	1,43	-14,23	0,56
Média	-14,35	0,59	-24,82	1,1	-13,01	0,56

Analisando os resultados médios da Tabela 13 podemos observar que utilizar o preditor 16×16 em conjunto com o 32×32 apresenta uma redução expressiva de -12,65% em comparação com o 32×32 isolado. Já a adição do preditor 8×8 ao conjunto apresentou uma perda de 0,77%. Diferentemente do codificador EVC, o HEVC não se beneficia de blocos 8×8 na predição de LFs, mesmo com estes possuindo MIs 8×8. Isto se deve à eficiência do preditor 16×16 atingir um BD-Rate consideravelmente superior ao 8×8, estando -11,81% acima deste, conforme mostrado na tabela anterior. Esta baixa eficiência do preditor 8×8 faz com que os poucos casos beneficiados pelos blocos 8×8 acabem não compensando o *overhead* de sinalizá-los.

5.2.4 Múltiplos Tamanhos de Bloco para Light Fields 16x16 Vistas

Após avaliarmos a eficiência dos preditores desenvolvidos para os LFs com suas vistas reduzidas a 8×8, realizamos a mesma análise sobre a eficiência dos preditores utilizando a técnica de preenchimento de vistas até obtermos LFs com MIs 16×16.

Analisando a Tabela 14 que apresenta os resultados dos preditores para cada um dos tamanhos de bloco separadamente, podemos observar que o preenchimento de vistas obteve resultados superiores aos LFs não preenchidos para os tamanhos de bloco 32×32 e 16×16. Assim como esperado, blocos 8×8 acabam partindo as MIs no meio, causando um desalinhamento entre as MIs e os blocos e causando uma performance muito abaixo dos demais tamanhos de bloco. Ainda, similarmente ao comportamento para LFs 8×8, o tamanho de bloco 16×16 possui um BD-Rate superior ao 32×32, porém para LFs 16×16 esta foi de apenas -1,66%, enquanto para

Tabela 13 – Avaliação de adição de modos em LFs 8×8 com o preditor DASCCLR (BD-Rate e BD-PSNR em relação ao HEVC).

Sequência	32×32		32×32,16×16		32×32,16×16, 8×8	
	BD-Rate	**BD-PSNR**	**BD-Rate**	**BD-PSNR**	**BD-Rate**	**BD-PSNR**
Ankylosaurus	-14,94	0,4	-28,4	0,82	-30,05	0,88
Bikes	-20,24	0,94	-37,43	1,93	-39,87	2,1
Black-Fence	-4,84	0,15	-14,9	0,47	-16,49	0,53
Ceiling-Light	-2,67	0,08	-17,72	0,54	-24,79	0,78
Danger-de-Mort	-16,65	0,88	-32,47	1,86	-35,92	2,13
Friends-1	-10,05	0,35	-22,11	0,82	-23,97	0,91
Houses-Lake	-7,96	0,2	-12,51	0,32	-12,51	0,32
Reeds	-5,08	0,14	-11,04	0,31	-11,25	0,32
Rusty-Fence	-15,28	0,73	-30,37	1,58	-34,9	1,89
Slab-Lake	-27,51	1,35	-43,21	2,38	-44,45	2,48
Swans-2	-24,57	0,94	-38,47	1,6	-40,05	1,68
Vespa	-22,46	0,89	-35,36	1,53	-36,6	1,6
Média	-14,35	0,59	-27	1,18	-26,23	1,3

LFs 8×8 foi acima de 10%. Essa maior eficiência dos blocos 32×32, atingindo um BD-Rate de -26,29%, é interessante também pelo fato de que blocos maiores requerem menos metadados e *bits* de sinalização, enquanto também apresentam melhor aproveitamento pelo codificador de entropia (MARTÍNEZ-RACH et al., 2021).

Ainda, um comportamento curioso apresentado pela tabela é que, apesar do *Slab-Lake* ter obtido um BD-Rate -2,33% inferior ao *Swans-2*, este LF obteve um BD-PSNR 0,25dB superior. Embora pareça uma diferença pequena devido às diferentes escalas, isto representa uma diferença de 20% entre os BD-PSNRs destes dois LFs. Embora possa parecer um comportamento normal, por conta de o BD-PSNR ser calculado em função do BD-Rate e vice-versa, um BD-Rate melhor, de maneira geral, indica um BD-PSNR também melhor, conforme discutido no início deste capítulo. Para os demais LFs, os padrões comportamentais se mantêm os mesmos.

A Tabela 15 apresenta a eficiência dos preditores sendo utilizados incrementalmente em conjunto para LFs com MIs de 16×16. Pode-se observar que no codificador HEVC utilizar LFs com *padding* de vistas ao invés do descarte apresenta um ganho de eficiência de -11,94%, com o preditor 32×32 atingindo um BD-Rate de -26,30%. Ao adicionarmos o preditor 16×16 atinge-se um ganho extra de -6,64%. Similarmente ao comportamento exibido na tabela anterior, ao adicionarmos o preditor 8×8 houve uma perda irrisória de 0,01% devido à sua baixa eficiência, além dos motivos discutidos na subseção 5.1.5.

5.2.5 Otimização Topológica

Após validarmos nosso método no codificador HEVC, realizamos algumas variações na estrutura da arquitetura *default* visando otimizá-las em custo de processamento e implementação em hardware. Logo, nesta etapa considera-se um *tradeoff*

Tabela 14 – Comparação de BD-Rate e BD-PSNR para os três tamanhos de blocos separadamente para LFs 16×16 com o preditor DASCCLR (BD-Rate e BD-PSNR em relação ao HEVC).

Sequências	32×32		16×16		8×8	
	BD-Rate	BD-PSNR	BD-Rate	BD-PSNR	BD-Rate	BD-PSNR
Ankylosaurus	-23,86	0,48	-22,13	0,44	-4,8	0,09
Bikes	-32,85	1,32	-36,63	1,52	-4,03	0,15
Black-Fence	-28,08	0,98	-35,92	1,3	-8,45	0,29
Ceiling-Light	-18,83	0,4	-21,74	0,47	-3,77	0,08
Danger-de-Mort	-29,68	1,26	-33,32	1,51	-2,98	0,12
Friends-1	-22,11	0,62	-20,06	0,56	-3,17	0,08
Houses-Lake	-17,11	0,41	-14,28	0,34	-0,2	0
Reeds	-12,22	0,29	-8,01	0,19	-0,09	0
Rusty-Fence	-30	1,25	-34,09	1,5	-5,74	0,22
Slab-Lake	-34,03	1,48	-38,81	1,73	-1,32	0,05
Swans-2	-36,33	1,23	-39,22	1,34	-3,23	0,1
Vespa	-30,37	1	-31,2	1,04	-3,69	0,11
Média	-26,29	0,89	-27,95	0,99	-3,46	0,11

entre eficiência de codificação atingida e o número de parâmetros da rede, ao invés de apenas ganhos de BD-Rate.

A Tabela 16 apresenta a comparação entre a arquitetura padrão, com as arquiteturas utilizando apenas convoluções convencionais no *decoder* com e sem a última camada do *encoder* (conforme Figuras 39) e utilizando apenas convoluções transpostas no *decoder* também com e sem a última camada do *encoder* (Figura 40).

Analisando os resultados médios da tabela, observa-se que substituir a convolução composta por uma convencional na última camada do *encoder* (coluna 2) gerou uma perda de aproximadamente 3% em BD-Rate. Ainda, retirar a maior camada do decoder (a final), apesar de diminuir o tamanho da rede em 600 mil parâmetros, causou uma perda de 6,51%. Em todos os resultados médios é possível notar que há uma relação direta entre o número de parâmetros e a eficiência obtida. Contudo, ao substituirmos todas as camadas do decodificador por convoluções transpostas, foi possível aumentar a eficiência da rede em -8,51%. Desta maneira, foi possível exceder a eficiência da arquitetura padrão em aproximadamente -4,0% e diminuir o seu número de parâmetros em 26% (para 2,97 milhões) ao removermos a camada final do *encoder*. Logo, devido à preocupação da necessidade destes modelos serem executados em FPGAs, e deste custo-benefício entre tamanho da rede e eficiência, a arquitetura padrão adotada nos próximos experimentos será utilizando apenas convoluções transpostas no *decoder* e sem a camada final do encoder conforme a Figura 40.

5.2.6 Pruning para Redução de Complexidade

Com esta nova arquitetura definida, direcionamos o foco do estudo especificamente para a redução de complexidade das redes e sua viabilização em FPGAs. Para

Tabela 15 – Avaliação de adição incremental dos preditores em LFs 16×16 vistas com o preditor DASCCLR (BD-Rate e BD-PSNR em relação ao HEVC).

Sequências	32×32		32×32, 16×16		32×32, 16×16, 8×8	
	BD-Rate	**BD-PSNR**	**BD-Rate**	**BD-PSNR**	**BD-Rate**	**BD-PSNR**
Ankylosaurus	-23,86	0,48	-28,84	0,59	-28,8	0,59
Bikes	-32,85	1,32	-41,84	1,78	-41,79	1,78
Black-Fence	-28,08	0,98	-36,92	1,37	-36,96	1,37
Ceiling-Light	-18,83	0,4	-27,02	0,59	-27,04	0,59
Danger-de-Mort	-29,68	1,26	-37,61	1,72	-37,6	1,72
Friends-1	-22,11	0,62	-26,58	0,77	-26,58	0,77
Houses-Lake	-17,11	0,41	-20,35	0,5	-20,28	0,49
Reeds	-12,22	0,29	-13,75	0,33	-13,77	0,33
Rusty-Fence	-30	1,25	-39,59	1,79	-39,61	1,79
Slab-Lake	-34,03	1,48	-41,98	1,89	-41,97	1,89
Swans-2	-36,33	1,23	-44,28	1,57	-44,27	1,57
Vespa	-30,37	1	-36,36	1,23	-36,34	1,23
Média	-26,29	0,89	-32,93	1,18	-32,92	1,18

Tabela 16 – Comparação de BD-Rate e BD-PSNR da arquitetura padrão com as demais arquiteturas avaliadas a nível de camadas (BD-Rate e BD-PSNR em relação ao HEVC).

Sequências	Padrão		Conv		ConvNoLL		ConvTrans		ConvTransNoLL	
	BD-Rate	**BD-PSNR**	**BD-Rate**	**BD-PSNR**	**BD-Rate**	**BD-PSNR**	**BD-Rate**	**BD-PSNR**	**BD-Rate**	**BD-PSNR**
Ankylosaurus	-23,86	0,48	-21,44	0,42	-15,14	0,29	-33,3	0,72	-28,5	0,6
Bikes	-32,85	1,32	-29,68	1,14	-21,96	0,81	-43,06	1,88	-38,1	1,61
Black-Fence	-28,08	0,98	-22,95	0,77	-18,1	0,61	-41,24	1,61	-37,59	1,44
Ceiling-Light	-18,83	0,4	-14,45	0,3	-9,99	0,2	-23,9	0,53	-16,79	0,36
Danger-de-Mort	-29,68	1,26	-24,77	0,98	-18,21	0,7	-39,18	1,82	-33,85	1,52
Friends-1	-22,11	0,62	-20,09	0,55	-14,01	0,37	-27,61	0,8	-22,34	0,63
Houses-Lake	-17,11	0,41	-17,43	0,42	-11,24	0,26	-22,73	0,57	-21,74	0,54
Reeds	-12,22	0,29	-13,86	0,33	-6,82	0,16	-15,06	0,36	-14,85	0,36
Rusty-Fence	-30	1,25	-25,38	1,01	-18,81	0,72	-40,12	1,82	-34,12	1,49
Slab-Lake	-34,03	1,48	-29,19	1,19	-21,57	0,85	-42,99	2,01	-36,93	1,65
Swans-2	-36,33	1,23	-33,11	1,07	-25,01	0,77	-50,84	1,95	-44,25	1,61
Vespa	-30,37	1	-27,86	0,89	-21,24	0,65	-37,59	1,3	-34,26	1,16
Média	-26,29	0,89	-23,35	0,76	-16,84	0,53	-34,8	1,28	-30,28	1,08
#Parâmetros	3,5MI		3MI		2,4MI		3,5MI		2,97MI	

isso, comparamos a eficiência das técnicas de *pruning* estruturado e não estruturado explicadas na subseção 2.6. A cada passo de *pruning* o número de parâmetros cortados da rede também tem grande impacto na sua eficiência durante o retreinamento. Frente a isso, para cada uma das técnicas de *pruning* avaliamos com duas taxas de corte, uma agressiva e uma branda. Iniciamos estas análises com blocos 16×16, pelo motivo de o menor número de amostras por bloco fazer com que o impacto das podas seja maior, tornando o treino mais instável e, consequentemente, difícil de atingir um resultado positivo. Dito isto, ao chegar neste resultado, ele tem maior probabilidade de ser positivo também para blocos 32×32, enquanto o inverso não necessariamente é verdade.

A Figura 44 apresenta a eficiência do preditor com dimensões 16×16 em BD-Rate para cada um dos passos de poda ao longo do *pruning* começando pelo passo 0 onde o modelo ainda não foi podado. Desta maneira, o *pruning* não estruturado zera os 10% e 25% pesos mais próximos de 0 até que cheguem a 80% de redução ou comecem a

gerar perdas por duas iterações seguidas. Para uma melhor percepção dos tamanhos dos modelos frente às suas eficiências, os pontos referentes a cada passo de *pruning* foram rotulados com o tamanho do modelo na determinada iteração.

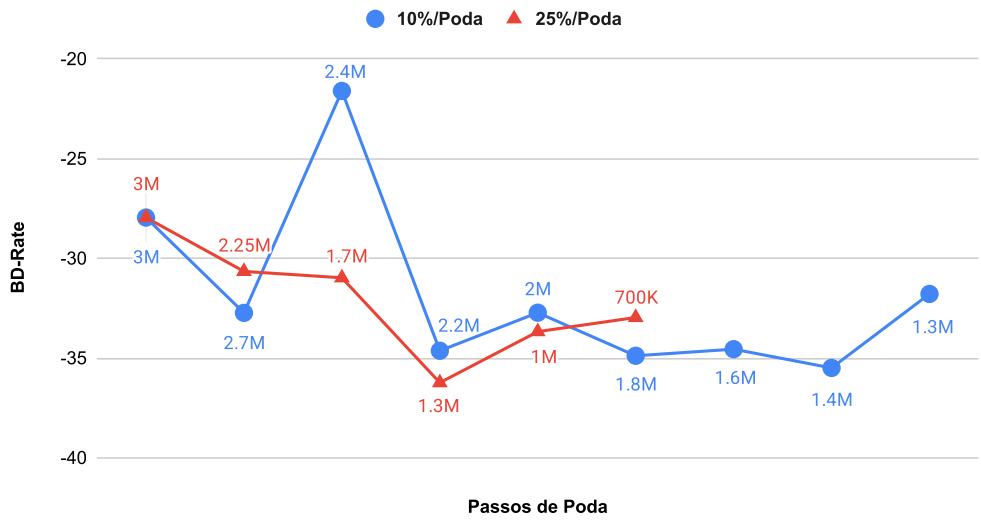


Figura 44 – Eficiência de codificação ao longo do pruning não estruturado no codificador HEVC para blocos 16X16.
Fonte: Desenvolvido pelo autor.

As curvas demonstram que, apesar da poda de 25% ser mais agressiva, esta atinge um resultado melhor de -36,23% de BD-Rate em apenas 3 passos de poda (totalizando 400 épocas de treinamento) enquanto a poda menos agressiva atinge seu menor BD-Rate de -35,49% após 7 passos de poda (totalizando 800 épocas). Ainda, no terceiro passo de *pruning* a 25% dos pesos, a rede possui apenas 1,3 milhões de parâmetros, enquanto a poda menos agressiva tem 1,4 milhões de parâmetros.

Em vista da alta demanda de tempo para treinar e codificar os vários passos de poda, e que o treinamento dos preditores com dimensões 32×32 possui comportamento mais previsível que com dimensões 16×16, seguimos a hipótese de que podas de 25% também serão mais eficientes que as de 10% também para este caso. Desta maneira, com base nesta hipótese, treinamos e avaliamos os preditores com dimensões de 32×32 apenas com a poda de 25%. Gostaríamos de ressaltar que podas acima de 25% se mostraram demasiadamente agressivas para convergir o treinamento de maneira eficiente.

A Figura 45 apresenta os resultados de BD-Rate para cada um dos passos de poda do *pruning* não estruturado do treinamento do preditor com dimensões 32×32. A curva apresenta o formato de "U" desejado, evidenciando também que o treinamento foi interrompido após começar a haver perdas de BD-Rate em dois passos de *pruning* seguidos. De maneira similar ao treinamento do preditor 16×16, o modelo atingiu sua melhor instância no terceiro passo de *pruning* com um BD-Rate de -31,43%. O fato de o preditor ter atingido o seu pico de eficiência no mesmo passo de *pruning* que o

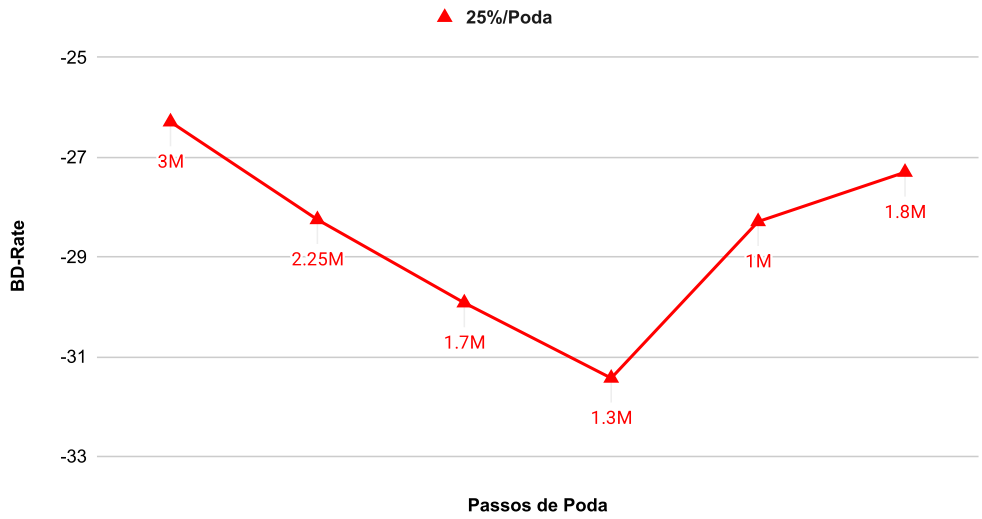


Figura 45 – Eficiência de codificação ao longo do pruning não estruturado no codificador HEVC para blocos 32x32.
Fonte: Desenvolvido pelo autor.

com dimensões de 16×16 fortalece a hipótese levantada de que os comportamentos do *pruning* para blocos menores podem ser traduzidos para os blocos maiores. Ressaltamos que, como as arquiteturas utilizadas para ambas dimensões são as mesmas tal qual as taxas de poda, o tamanho dos modelos nos respectivos passos são os mesmos para ambas dimensões.

Ainda, como pode ser observado na Figura 46, o comportamento das *loss* de treino e validação ao aplicarmos *pruning* se torna mais complexo. Ao longo do primeiro treinamento da época 0 até a época 100, a curva permanece normal por não ter ocorrido nenhuma poda ainda. Após a centésima época, ao sofrer a poda e o treino ser reinicializado, o modelo perde um pouco de sua eficiência e volta a aprender ao longo das próximas 100 épocas. A cada poda realizada, a perda do modelo tende a crescer devido à sua capacidade de aprender estar sendo diminuída. Consequentemente, a eficiência de inferência do modelo tende a diminuir também. Como produzir o BD-Rate para todos os passos de *pruning* é indesejável, considerando o tempo para realizar as codificações, e o treino do modelo por 900 épocas pode levar semanas, torna-se necessária uma maneira de avaliar em qual passo o modelo começará a apresentar perdas significativas para então poder-se encerrar o treinamento.

Para podermos realizar esta análise e tomar futuras decisões, relacionamos os valores de *loss* obtidos pelo modelo a cada passo do *pruning* com a esparsidade do modelo. Para esclarecimento, tem-se por esparsidade a porcentagem de pesos zerados no modelo. Logo, uma instância do modelo com 50% de esparsidade tem metade dos seus pesos zerados e que, em potencial, não precisam ser calculados pela GPU. De maneira geral, o comportamento esperado da curva de *loss* por esparsidade possui um formato de "U", onde o modelo após os primeiros passos de poda e retreino,

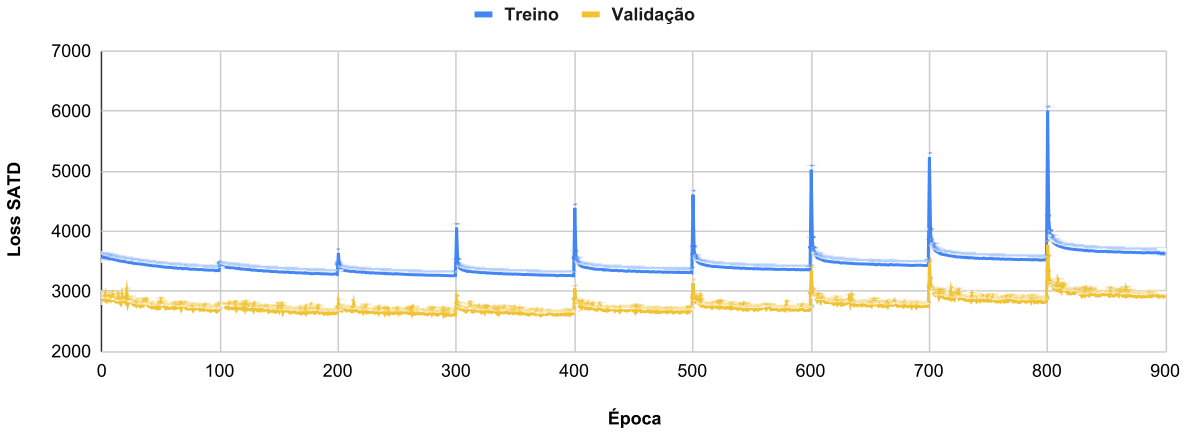


Figura 46 – Curva de Loss para o treino e validação do modelo sob pruning de 25% a cada 100 épocas.
Fonte: Desenvolvido pelo autor.

tem sua performance melhorada até certo ponto onde começa a perder performance sistematicamente.

As Figuras 47 e 48 apresentam a relação da função de *loss* com a esparsidade dos modelos de dimensões 32×32 e 16×16 treinados sob *pruning* de 25% discutidos nesta seção. Atente-se que os pontos na curva representam cada iteração de treino constituído de 100 épocas seguido da poda, começando pelo passo 0, onde nenhuma poda é realizada. Analisando as curvas é possível visualizar que os modelos para ambas dimensões são capazes de seguir melhorando seu aprendizado até o fim do treinamento da sua terceira poda (quarto ponto), onde atingem uma esparsidade de 57%, totalizando 1,3M de parâmetros podados. Se relacionarmos ambas figuras com as Figuras 44 e 45, é possível visualizar que, nos mesmos passos em que os modelos atingem suas melhores *loss* também atingem seus melhores BD-Rate. Logo, podemos afirmar que a métrica de *loss* SATD é precisa o suficiente para determinarmos em qual passo de *pruning* podemos encerrar o treinamento e qual será o modelo mais eficiente, eliminando a necessidade de codificar os LFs para obter os BD-Rates a cada iteração. Ademais, note que os valores de *loss* para os preditores de dimensões 16×16 são menores que para os preditores com dimensões 32×32, fazendo com que esta curva precise ser analisada com maior cuidado. Porém, a estabilidade em ambos treinamentos se mostra muito similar, com o menor valor de *loss* estando aproximadamente 7% abaixo do valor obtido no passo 0.

Outro comportamento a se observar que ocorre em ambas as curvas e que mantém seu formato em "U" é a ausência de picos ou vales após ou antes do melhor resultado, o que também evidencia que, após um determinado passo de *pruning*, o treino tende a não gerar melhores eficiências novamente, facilitando a identificação de quando terminar o treinamento cedo.

Contudo, vale recordar que embora o *pruning* não estruturado diminua o número de

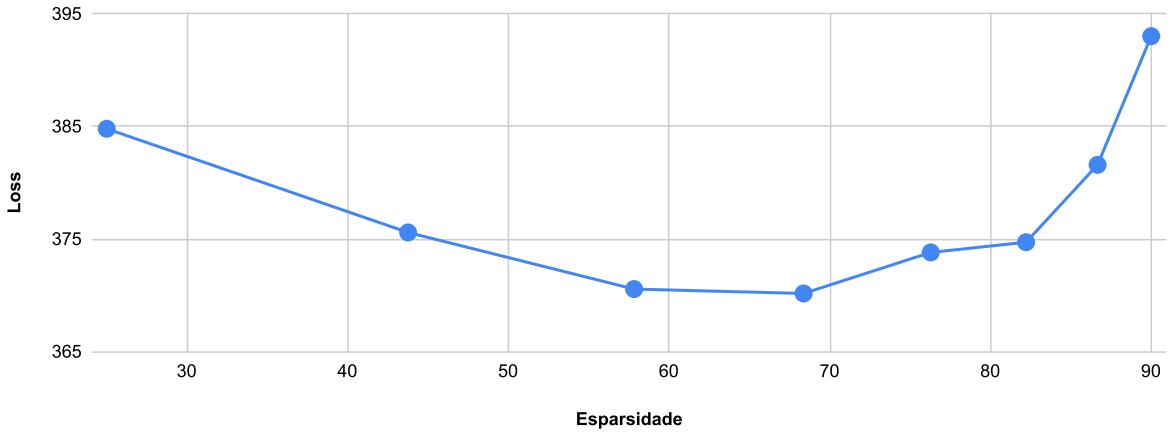


Figura 47 – Loss da validação por esparsidade do modelo treinado sob pruning não estruturado com passos de 25% para blocos 16×16
Fonte: Desenvolvido pelo autor.

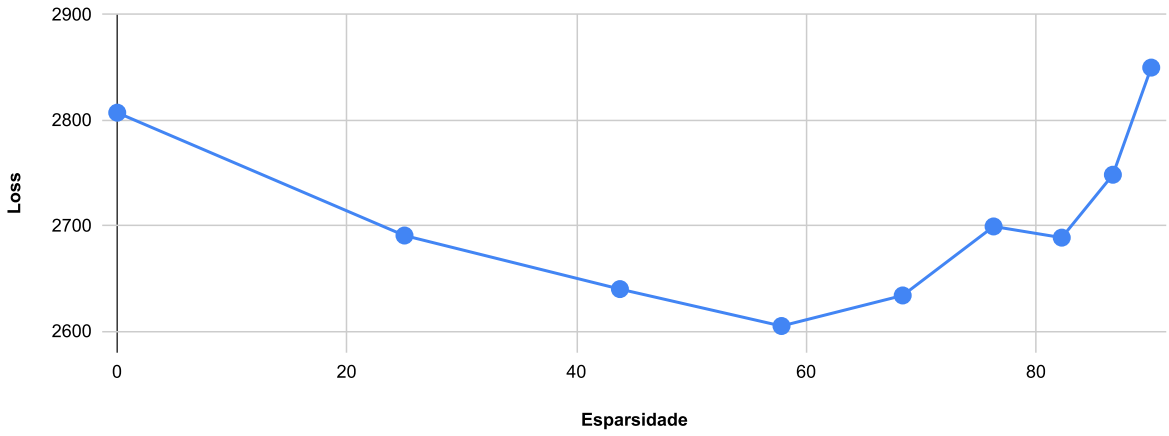


Figura 48 – Loss da validação por esparsidade do modelo treinado sob pruning não estruturado com passos de 25% para blocos 32×32.
Fonte: Desenvolvido pelo autor.

parâmetros, seu ganho de eficiência não é garantido em *hardware*. Como desejamos obter boas performances executando em hardware, avaliamos também a eficiência de *pruning* estruturado. Nesta avaliação aplicamos os mesmos conceitos e hipóteses definidas para o *pruning* não estruturado. Porém, em vista que esta técnica é muito mais agressiva que o *pruning* não estruturado por podar conjuntos inteiros de filtros, realizamos as podas com taxas de corte de 5%. Gostaríamos de ressaltar que este valor foi atingido experimentalmente, onde valores acima disto geravam treinamentos muito instáveis, com o treino parando de convergir totalmente com uma taxa de poda de 10%.

As Figuras 49 e 50 apresentam as curvas de *loss* por esparsidade para os treinos dos modelos sob *pruning* estruturado de 5% para os tamanhos de blocos de 16×16 e 32×32 respectivamente. Analisando ambas figuras é possível notar que ambas apresentam picos e vales mais agressivos que não são presentes nas curvas do *pruning* não estruturado. Em particular, estes picos podem ser observados nos passos 5 e 8 para a rede treinada para blocos menores e no passo 5 para a outra rede. Estes picos mais elevados são previstos, pois o aumento na agressividade das podas nos pesos das redes naturalmente tende a provocar maiores impactos na performance das redes.

Similarmente ao *pruning* não estruturado, ambas as redes atingiram sua melhor performance em passos de *pruning* próximos. A rede treinada para blocos 16×16 atingiu sua melhor performance no sétimo passo, enquanto para blocos 32×32 atingiu no passo 6. Desta maneira, estas redes atingiram uma esparsidade de 30% e 26%, em outras palavras, foram podados aproximadamente 115 mil e 109 mil parâmetros, fazendo com que as redes fossem reduzidas a 2.077.500 e 2.186.843 parâmetros, respectivamente. Ao codificar as instâncias dos modelos nesses passos, obtemos BD-Rates de -37,84% e -34,23%, e BD-PSNRs de 1,48dB e 1,26dB respectivamente. Ao compararmos a rede para blocos 32×32 à sua rede equivalente avaliada inicialmente nesta tese, percebe-se que houve uma redução de aproximadamente 44% no número de parâmetros de (3,3M para 2,2M) enquanto atingimos também uma melhora de 19% de BD-Rate e uma melhora de 0,62dB de BD-PSNR, indicando um ganho de 2x no BD-PSNR inicial com 0,64dB (comparações com "Kernels 3×3", melhor resultado presente na Tabela 2). Ainda, para averiguar o potencial máximo de ambos os preditores, estes devem ser utilizados em conjunto pelo codificador.

5.2.7 Comparação com Trabalhos Relacionados

A Tabela 17 apresenta os resultados de BD-Rate que foram alcançados por ambos os preditores desenvolvidos nesta tese quando inseridos em conjunto no codificador HM. Ainda, nesta tabela estão presentes os resultados obtidos por Zhong et al. (2019), que propõe o uso de 6 redes neurais para predição intra de LFs para o co-

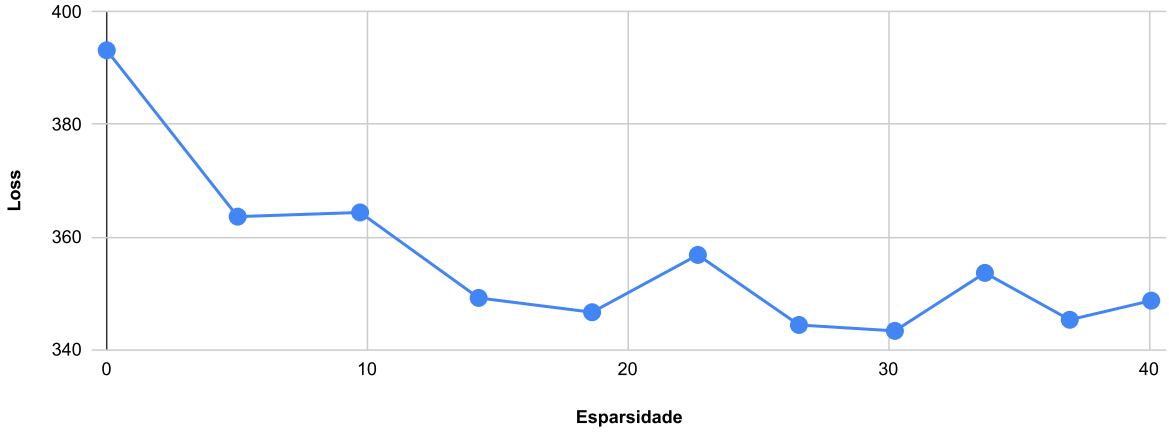


Figura 49 – Loss da validação por esparsidade do modelo treinado sob pruning estruturado com passos de 5% para blocos 16×16.
Fonte: Desenvolvido pelo autor.

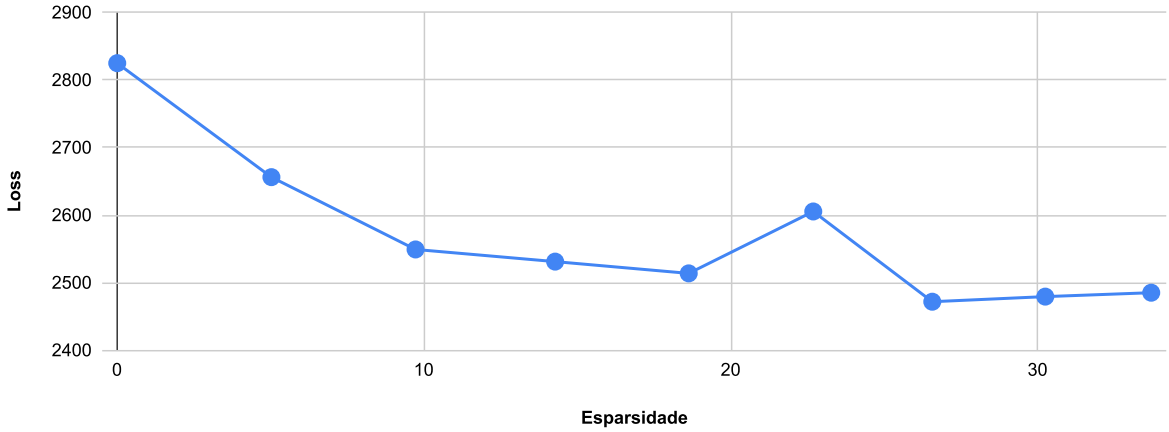


Figura 50 – Loss da validação por esparsidade do modelo treinado sob pruning estruturado com passos de 5% para blocos 32×32.
Fonte: Desenvolvido pelo autor.

Tabela 17 – Resultados de eficiência de codificação do HEVC integrado com os preditores DASCCLR Conv.Trans.NoLL para blocos 32x32 e 16x16 em comparação com Zhong et al. (2019) (BD-Rate e BD-PSNR em relação ao HEVC).

Sequências	Proposto 32+16		Zhong et al. (2019)
	BD-Rate (%)	BD-PSNR (dB)	BD-Rate (%)
Ankylosaurus-&-Diplodocus-1	-35.41	0.78	-24.09
Bikes	-50.8	2.42	-11.38
Black-Fence	-49.62	2.13	-7.74
Ceiling-Light	-31.52	0.72	0.42
Danger-de-Mort	-45.2	2.29	-7.67
Friends-1	-32.23	0.99	-3.23
Houses-Lake	-30.35	0.81	-21.14
Reeds	-18.06	0.45	-1.74
Rusty-Fence	-47.86	2.38	-4.41
Slab-Lake	-50.77	2.52	-6.59
Swans-2	-56.42	2.32	-34.04
Vespa	-43.13	1.57	-15.43
Média	**-40.95**	**1.61**	**-11.42**

Tabela 18 – Comparação dos resultados de eficiência de codificação do HEVC integrado com os preditores e o JPEG Pleno - MuLE para LFs em comum (BD-Rate e BD-PSNR em relação ao HEVC).

Sequências	Proposto 32+16	MuLE
	BD-Rate (%)	BD-Rate (%)
Bikes	**-50.80**	-39.11
Danger-de-Mort	-45.20	**-53.19**
Friends-1	**-32.23**	-34.83
Average	**-42.74**	**-42.38**

dificador HEVC (conforme discutido no capítulo 3). Analisando os resultados obtidos pelos preditores desenvolvidos nesta tese, um ganho médio de -40,95% de BD-Rate e uma melhora de 1,61dB em BD-PSNR podem ser observados. Ao analisar os LFs com menores ganhos, nota-se que a compressão do LF *Reeds* foi melhorada em 5% pela técnica de *pruning*, fazendo com que este atingisse um BD-Rate de -18%. Já a sequência *Houses-Lake* teve seu BD-Rate melhorado em 13,24%, atingindo um total de -30,35%. Ainda, a sequência *Swans-2* segue atingindo a melhor eficiência de compressão, com um BD-Rate de -56,42%, uma melhora de 20,19% como resultado da aplicação da técnica de *pruning* estruturado.

Se compararmos os ganhos médios obtidos, aos ganhos apresentados pelo codificador JPEG Pleno-MuLE sobre o HEVC em Carvalho et al. (2018), os preditores desenvolvidos nesta tese os superam em 0.4%. Infelizmente, Carvalho et al. (2018) apresenta resultados apenas para 5 LFs do *dataset* EPFL, onde 2 estão presentes no conjunto de treinamento definido e sua comparação seria injusta. Desta forma, a Tabela 18 apresenta os resultados de BD-Rate dos 3 LFs em comum entre esta tese

e Carvalho et al. (2018). Analisando estes resultados, os preditores propostos ainda atingiram um resultado competitivo -0,36% superior ao MuLE. Enquanto o codec de LFs densos atingiu uma redução maior para a sequência "*Danger-de-Morte*" , os preditores desenvolvidos superaram o codec para as sequências "*Bikes*" e "*Friends-1*". O fato de a inserção de apenas 2 preditores intra-quadro em um codificador de vídeo superar os ganhos obtidos por um codificador desenvolvido especificamente para a compressão de LFs sob o uso de transformadas evidencia a eficiência dos preditores e sua capacidade de adequar o processo de compressão dos codificadores de vídeo para o contexto de LFs. Ainda, conforme discutido no capítulo 1, estes resultados também possibilitam que LFs sejam comprimidos seguindo um padrão de codificação já definido, o que possibilita o seu uso em dispositivos móveis, evitando a necessidade do desenvolvimento de novas implementações e adaptações de *hardwares*.

Ao comparar os resultados obtidos pelo método proposto com poda com os atingidos por Zhong et al. (2019), é possível concluir que nossa abordagem alcança uma redução média de BD-Rate de -29,53%. Gostaríamos de ressaltar que não apresentamos comparação de BD-PSNR porque os autores não fornecem valores de BD-PSNR.

Note que, embora (ZHONG et al., 2019) obtenha reduções semelhantes para as sequências *Ankylosaurus-&-Diplodocus-1* e *Swans-2*, hipotetizamos que isso ocorre devido à presença de LFs com características semelhantes no conjunto de treinamento, como *Ankylosaurus-&-Diplodocus-2* e *Swans-1*, respectivamente. Este comportamento nos resultados pode ser tido como uma evidência adicional de que nossa rede possui maior capacidade de generalização em comparação com Zhong et al. (2019).

Além disso, a pior sequência do método deles foi Ceiling-Light, com uma perda de BD-Rate de 0,42%, enquanto conseguimos um ganho de -31,52% para essa sequência e -19,14% no nosso pior caso ("*Reeds*").

Neste capítulo discutimos os desempenhos atingidos por todas as técnicas e arquiteturas avaliadas na nossa análise exploratória. Inicialmente investigamos através do EVC as eficiências de características das redes como tamanhos de *kernel* e formato de *Unet*. Em seguida, avaliamos diversos hiperparâmetros do treinamento, como diferentes funções de perda, maneiras de calculá-las e *learning rate*. Posteriormente, ainda avaliamos para o EVC o desempenho dos preditores utilizados separadamente para cada tamanho de bloco e em conjunto, assim como a aplicação de *padding* de vistas nos LFs. Após definirmos estes aspectos do treinamento, começamos estudos sobre os desempenhos dos preditores no HEVC. Avaliando aspectos como o modo de predição a ser substituído e a taxa de seleção dos preditores propostos. Após resultados satisfatórios, avaliamos diferentes arquiteturas que trouxessem um bom *tradeoff* em relação à BD-Rate e número de parâmetros. Para melhorar este *tradeoff* aplicamos também múltiplas técnicas de *pruning* que reduziram o tamanho das redes em

44% enquanto melhoraram seus BD-Rates em 36%. Os preditores resultantes então foram comparados com os resultados obtidos por (ZHONG et al., 2019) mostrando reduções -29,53% superiores.

Em vista dos resultados satisfatórios atingidos pelos métodos de avaliação propostos nesta seção, consideramos as hipóteses levantadas comprovadas suficientemente para justificar a aplicação do modelo selecionado. Desta maneira, o capítulo seguinte busca validar o que foi proposto, inserindo em um novo codificador de vídeo os preditores já treinados sem modificação alguma, para assim verificarmos a eficiência e generalidade dos preditores desenvolvidos.

6 RESULTADOS FINAIS

Após definirmos a versão final dos preditores visando o melhor *tradeoff* entre número de parâmetros e eficiência de codificação, é necessário avaliarmos sua generalidade e eficiência tanto em relação à tecnologia do codificador quanto ao *dataset*. Desta maneira, inserimos os preditores de tamanho 32×32 e 16×16 avaliados na Seção 5.2.6 sem modificação alguma em um novo codificador estado-da-arte.

O codificador escolhido foi o *software* de referência VTM v23.1 do padrão de codificação VVC e modificado seguindo a mesma metodologia discutida na Seção 4. Conforme discutido na Seção 2, este codificador possui um maior número de modos intra e ferramentas de codificação muito mais complexas do que o HEVC, o que o torna um caso de teste desafiador para a técnica proposta. Ainda, para realizarmos uma análise mais profunda dos resultados obtidos e provarmos que a técnica proposta gera *bitstreams* decodificáveis, o decodificador do VTM também foi modificado sob a mesma metodologia para utilizar os preditores desenvolvidos via sessão UDP.

O codificador modificado foi primeiramente utilizado para codificar o *dataset* de teste utilizando inicialmente o preditor 32×32 e 16×16 para observarmos o impacto BD-Rate e BD-PSNR de cada preditor. Analisando a Tabela 19 podemos observar que, mesmo o VTM sendo um codificador com ferramentas mais avançadas que o EVC e o HEVC, este obteve ganhos ainda superiores aos seus antecessores. Ao utilizar o preditor 32×32 a eficiência de codificação do VTM foi melhorada em -39,15%, com o melhor caso "*Swans-2*" atingindo uma taxa de redução de 52%. Para os blocos 16×16 o BD-Rate é de -41,74%, -2,59% superior ao bloco 32×32. Ainda, similarmente aos outros codificadores, o uso do preditor 16×16 melhorou consideravelmente o BD-Rate do LF "*Ceiling-Light*" de -23,14% para -30,83%. Já o "*Reeds*", também um dos piores casos, foi melhorado em pouco mais de -1%. Ainda, para o "*Swans-2*" houve uma perda de BD-Rate de aproximadamente 1%. Ademais, a sequência "*Ankylosaurus-1*" teve eficiência 3,26% menor para o preditor 16×16, isto por conter em sua grande parte áreas homogêneas que são favorecidas por blocos maiores. Estes comportamentos específicos serão estudados em breve com análises gráficas mais detalhadas. Contudo, antes devemos avaliar os resultados obtidos pela proposta final desta tese,

Tabela 19 – Comparação de BD-Rate e BD-PSNR para os dois tamanhos de bloco separadamente para LFs 16×16 no codificador VTM 23.1 modificado (BD-Rate e BD-PSNR em relação ao VTM 23.1).

Sequências	32×32		16×16	
	BD-Rate	BD-PSNR	BD-Rate	BD-PSNR
Ankylosaurus	-41,94	0,97	-37,8	0,84
Bikes	-48,92	2,32	-51,58	2,61
Black-Fence	-41,28	1,6	-45,8	1,88
Ceiling-Light	-23,14	0,5	-30,83	0,69
Danger-de-Mort	-41,27	2	-46,31	2,4
Friends-1	-32,55	1,02	-32,56	1,04
Houses-Lake	-33,44	0,93	-32,65	0,92
Reeds	-24,36	0,59	-25,4	0,65
Rusty-Fence	-42,21	2,02	-49,5	2,56
Slab-Lake	-45,78	2,21	-52,27	2,73
Swans-2	-52	2,06	-50,98	2,03
Vespa	-42,92	1,54	-45,2	1,69
Média	-39,15	1,48	-41,74	1,67

Tabela 20 – Resultados de eficiência de codificação do VTM 23.1 integrado com os preditores desenvolvidos para blocos 32×32 e 16×16 (BD-Rate e BD-PSNR em relação ao VVC).

Sequências	**Proposto 32+16**	
	BD-Rate	BD-PSNR
Ankylosaurus	-48,37	1,17
Bikes	-56,2	2,87
Black-Fence	-50,02	2,11
Ceiling-Light	-33,14	0,75
Danger-de-Mort	-49,13	2,55
Friends-1	-38,7	1,28
Houses-Lake	-40,4	1,19
Reeds	-29,31	0,75
Rusty-Fence	-52,41	2,74
Slab-Lake	-54,2	2,84
Swans-2	-60,47	2,54
Vespa	-50,29	1,91
Média	**-46,89**	**1,89**

que é o uso das redes treinadas para os tamanhos de bloco em conjunto.

A Tabela 20 apresenta os resultados do método proposto com preditores para ambos os tamanhos de bloco em conjunto para o codificador VTM. O BD-Rate médio obtido pelo método proposto é de -46,89% e BD-PSNR de 1,89dB. Ao utilizar ambas as redes em conjunto, tem-se uma melhora de -3,94% quando comparado ao uso do preditor 16×16, que havia obtido a melhor redução para este codificador. O "*Swans-2*" obteve -60,47% de BD-Rate, sendo esta a maior redução obtida para todos os casos de teste. Já o LF "*Reeds*" detém a menor redução dos casos de teste, mesmo sendo beneficiado em -3,91% quando comparado ao preditor 16×16 ou 32×32 isolados, o seu BD-Rate não superou -29,31%. Ainda, o "*Ceiling-Light*" atingiu um BD-Rate de -33,14%, uma melhora de -2,31% quando comparado aos preditores isoladamente. Ainda, se compararmos os resultados médios obtidos com os atingidos pelo JPEG-Pleno no modo MuLE (CARVALHO et al., 2018), superamos sua redução média em -8,77%.

Analisando os resultados de BD-PSNR, observa-se um ganho de eficiência de 1,89dB. Para esta métrica, os dois melhores casos foram as sequências "*Bikes*" e "*Slab-Lake*" com 2,87dB e 2,84dB, respectivamente, estando acima da sequência "*Swans-2*" em 0,33dB (12% superior) mesmo atingindo BD-Rates piores (~4% e ~6%).

6.1 Análise e Discussão dos Resultados

Para melhor entender como estes ganhos são obtidos pelo método sendo proposto, devemos investigar o comportamento do codificador e como este está explorando os preditores inseridos. Para isso, extraímos das codificações discutidas na tabela anterior e das codificações do VTM sem modificações, informações como qual modo foi aplicado para cada um dos blocos, estrutura de particionamento, tamanhos dos blocos, entre outras estatísticas.

Inicialmente buscamos averiguar se houve uma diminuição na quantidade de blocos necessários para codificar os LFs. Pois um maior número de blocos na codificação requer também um maior número de *bits* de sinalização para determinar as informações de cada bloco, como qual modo intra foi utilizado. Logo, quanto menos blocos forem utilizados no processo para um mesmo RD durante o RDO, melhor tende a ser a compressão do codificador. Para realizarmos esta análise, a Figura 51 apresenta o número total de blocos utilizados na codificação de cada QP para todas as sequências.

Observando-a, nota-se que as codificações utilizando os preditores desenvolvidos utilizam praticamente a metade de blocos para o QP 22 do que o codificador original, possuindo em média 41,3% blocos a menos, o que representa uma redução de 139,973 blocos para sinalizar. Para o QP 27 a quantidade de blocos segue praticamente a mesma, com o método proposto necessitando apenas 3% de blocos a mais.

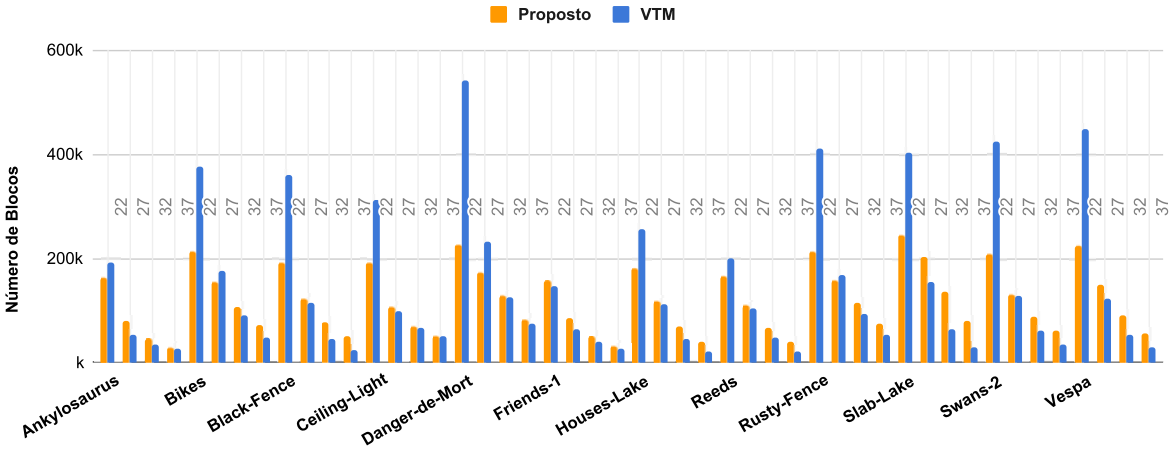


Figura 51 – Número total de blocos utilizados pelo VTM original e com os preditores desenvolvidos.
Fonte: Desenvolvido pelo autor.

Note que o único caso em que as modificações não ocasionaram redução do número de blocos no QP 22 é a sequência "*Friends-1*". Porém, note também que esta é uma das sequências com o menor número de blocos codificados, possuindo em média 82807 blocos, enquanto a média geral entre todas as sequências é de 118545 blocos. Ademais, apesar desta baixa redução em número de blocos, essa sequência tem um BD-Rate de -38,7%.

Já para os demais QPs, as modificações utilizaram 35,78% e 52,87% blocos a mais, respectivamente. Onde, para estes QPs, em nenhum caso de teste foi possível diminuir o número de blocos utilizados pelo VTM. Contudo, o VTM aproximadamente duplicou o número de blocos utilizados a cada decréscimo de QP entre os QPs 37 e 27, utilizando 36,477 blocos para o QP 37, 64,116 para o QP 32 e 127,561 para o QP 27. Já para o QP 22 este número aumentou 2,7x, totalizando 338,890 blocos. Frente a estes números e a alta redução atingida para o QP 22, os preditores desenvolvidos utilizaram um total de 92,864 blocos a menos que o VTM original para realizar todas estas codificações.

Para melhor entender estes comportamentos e por que o método proposto está utilizando um maior número de blocos para QPs mais altos, é necessário aprofundar a análise e averiguar onde os preditores estão sendo usados ou não e por quê. Para isso, coletamos a porcentagem de blocos que foram preditos utilizando os modos propostos e os demais modos do VTM que estão competindo.

A Figura 52 apresenta a parcela de blocos que foram preditos pelos modos propostos e pelos modos Planar, DC, MIPs, assim como os demais modos angulares. Observe que, para a grande maioria dos casos, os preditores desenvolvidos são majoritariamente selecionados. Para os QPs 22 e 27 do LF "*Swans-2*" os preditores desenvolvidos são utilizados para 97,54% e 97,91% dos blocos. Ainda, a figura deixa

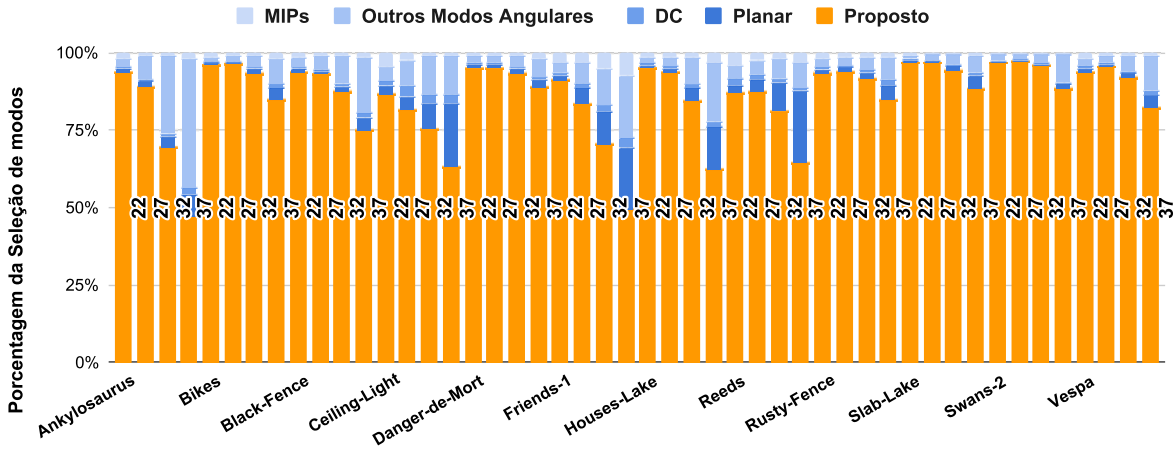


Figura 52 – Porcentagem da seleção dos modos intra e do preditor proposto em competição para cada um dos QPs avaliados.
Fonte: Desenvolvido pelo autor.

claro que, para os QPs maiores, os preditores tendem a ser menos utilizados. Em média, do QP mais baixo ao mais alto, os preditores desenvolvidos são utilizados 95,08%, 93,06%, 86,79% e 74,63%, respectivamente. O LF que menos utilizou os preditores foi o "*Ankylosaurus-1*" para o QP 37, com uma taxa de escolha de 47,88%. Vale ressaltar ainda que este foi o único caso em que os preditores foram utilizados para menos de 50% dos blocos.

6.2 Estudos de Caso

É possível investigar o porquê os modos intra propostos são menos utilizados para os QPs mais altos analisando as principais diferenças entre as imagens comprimidas com cada QP. A Figura 53 mostra o caso de teste "*Ankylosaurus-1*" que possui o BD-Rate mais próximo da média. Ainda este caso apresenta comportamentos interessantes a serem estudados, como o menor uso dos preditores para os QPs 32 e 37, além de possuir um aumento do número de blocos codificados para o QP 27. Através das figuras fica evidente que não é uma tarefa trivial notar as diferenças entre elas, não somente pela alta semelhança, mas também pelos padrões das MIs presentes nos LFs. Conforme o QP aumenta, a atenção do codificador às texturas complexas e detalhes começa a diminuir. Desta maneira, a imagem codificada com o QP 37 apresenta um efeito de desfoque em áreas como o focinho do dinossauro verde que, em conjunto com o efeito das MIs, pode até mesmo transmitir uma falsa sensação de maior nitidez.

Para melhor revelar as diferenças entre as codificações destes QPs, a Figura 54 apresenta o resíduo (diferenças) entre as imagens codificadas com o QP 22 e 37. Através dessa imagem, podemos observar que as principais diferenças encontram-se nas cabeças dos dinossauros e em seus contornos. Regiões estas que tendem

(a) QP 22 (b) QP 27

(c) QP 32 (d) QP 37

Figura 53 – Ankylosaurus-1 para todos QPs.
Fonte: Desenvolvido pelo autor.

a possuir texturas mais complexas devido à presença de detalhes e/ou diferenças de profundidade.

Outro resíduo que pode trazer informações sobre a codificação é a diferença entre o LF original e os codificados pelo VTM padrão e modificado. A Figura 55 apresenta resíduo entre o LF "*Ankylosaurus-1*" original e codificado pelo VTM padrão à esquerda e pelo VTM com os preditores desenvolvidos à direita. Ambos os resíduos são quase que completamente cinzas, evidenciando codificações com baixíssimas perdas visuais. Analisando os resíduos detalhadamente, é possível apenas identificar que enquanto as bordas dos dinossauros codificados com o VTM original apresentam resíduos mais escuros, os codificados utilizando os preditores desenvolvidos apresentam ruídos levemente mais claros.

Devido às poucas informações trazidas pelos resíduos, aprofundamos a discussão coletando também as informações dos tamanhos de todos os blocos codificados pelo VTM original e o modificado, assim como o modo intra utilizado por cada um destes blocos. Essas informações foram então sobrepostas sobre o LF original para permitir avaliarmos em quais regiões os preditores desenvolvidos estão sendo utilizados e entender melhor o motivo de sua menor utilização em QPs maiores. A Figura 56 apresenta à esquerda as superposições da estrutura de blocos e modos intra utilizados pelo VTM padrão e à direita pelo modificado. Para melhor identificar os preditores

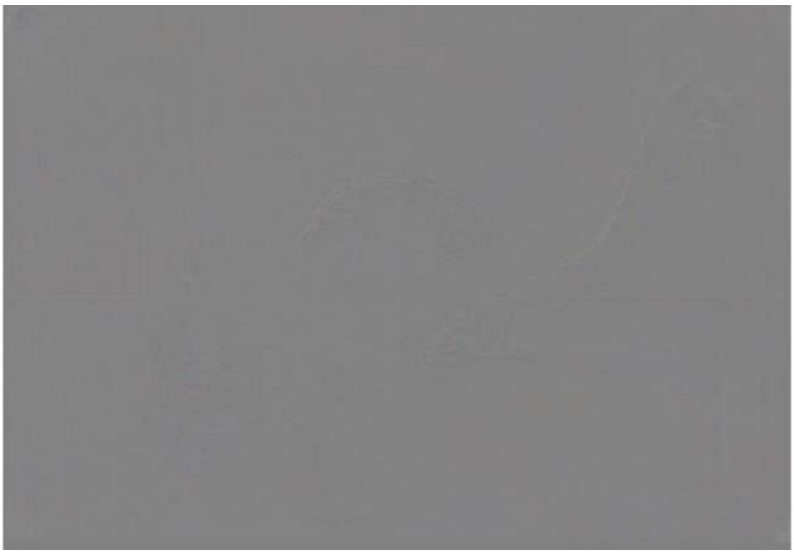
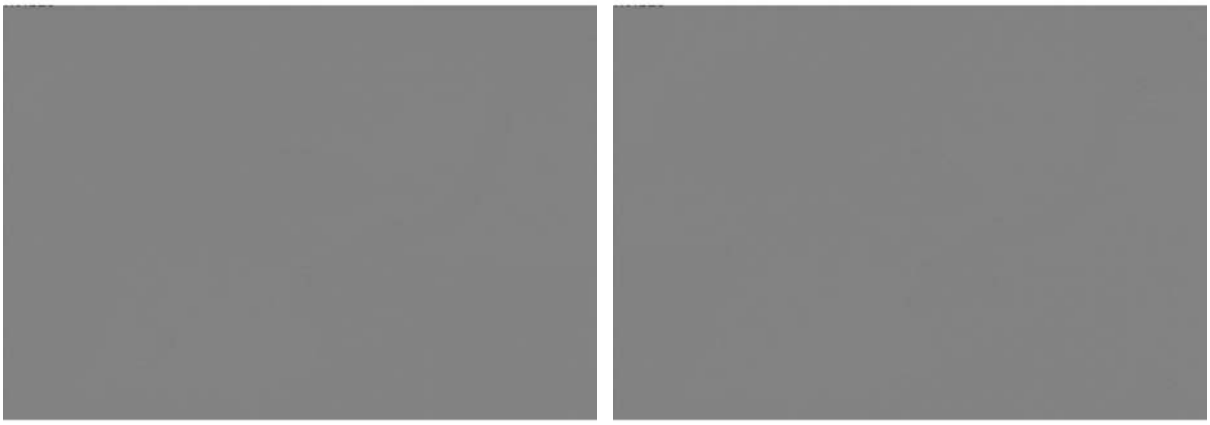


Figura 54 – Resíduos do LF Ankylosaurus-1 entre QP22 e QP37.
Fonte: Desenvolvido pelo autor.

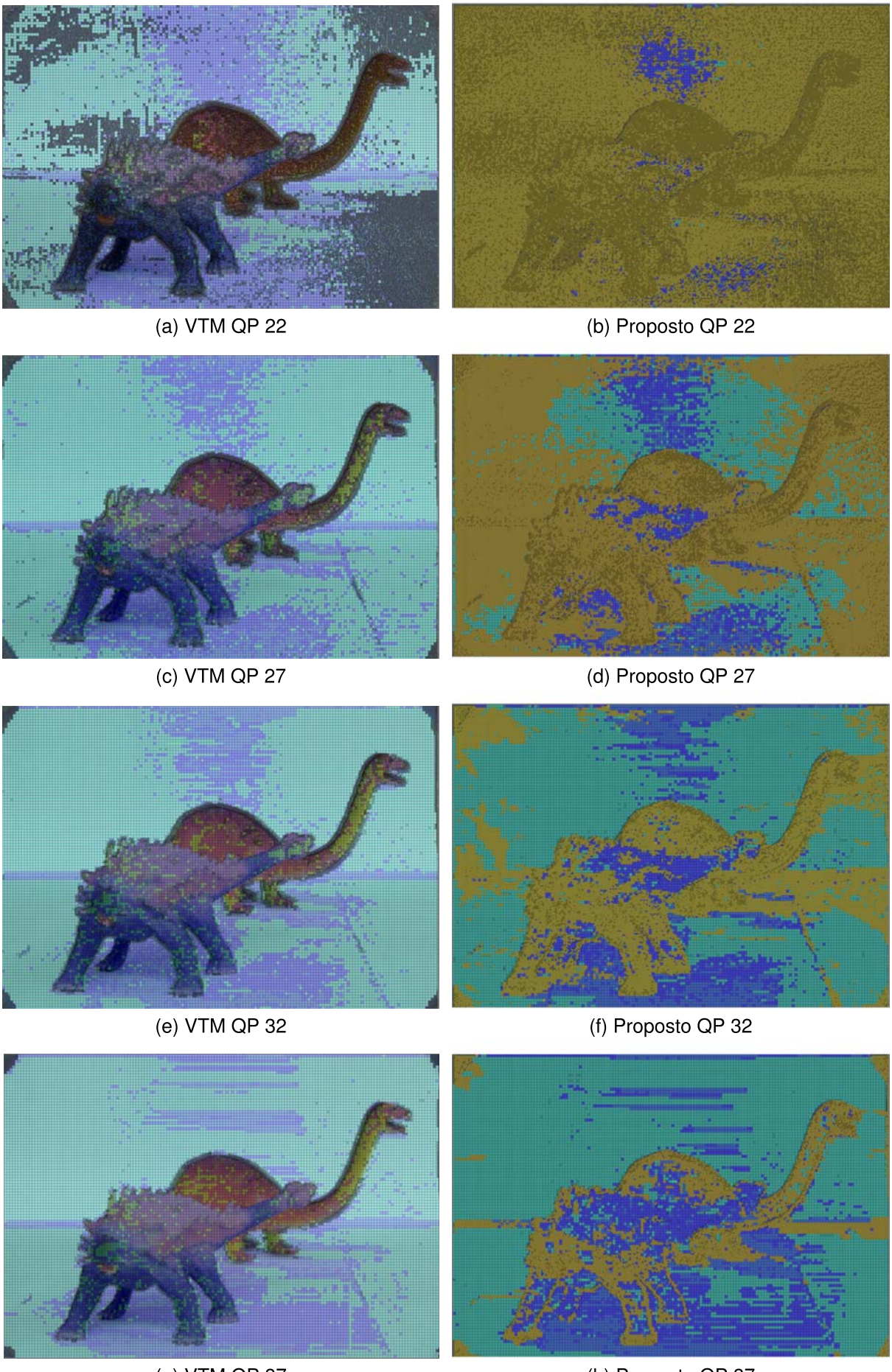


(a) VTM - Original (b) Proposto - Original

Figura 55 – Resíduos do Ankylosaurus-1 para o QP 22.
Fonte: Desenvolvido pelo autor.

desenvolvidos na escala de cor, definimos eles como amarelo, enquanto os modos padrões do VTM são apresentados no mapa de cores entre tons de azul e verde, azul sendo os modos Planar, DC e modos angulares com índices abaixo de 34, enquanto as cores azuis indicam modos intra angulares com índices acima de 34. Ainda, regiões amarelas mais escuras correspondem ao uso do preditor proposto para blocos 16×16, contudo, o efeito da cor escurecida é dado apenas pelas linhas de divisão dos blocos. Por fim, aconselhamos rever os LFs originais presentes na Figura 34 rever suas características a fim de melhor compreender as próximas discussões.

Analisando as figuras referentes ao QP 22, nota-se que, praticamente toda imagem é codificada com os preditores desenvolvidos, certificando o seu uso para mais de 95% dos blocos. A única região cujo uso do preditor não é proeminente é no centro acima do dinossauro mais ao fundo. Em vista de que é uma região homogênea



(a) VTM QP 22 (b) Proposto QP 22

(c) VTM QP 27 (d) Proposto QP 27

(e) VTM QP 32 (f) Proposto QP 32

(g) VTM QP 37 (h) Proposto QP 37

Figura 56 – Mapa dos blocos selecionados para o LF Ankylosaurus-1 para cada QP. Modos do VVC em tons de Azul; Modo proposto em Amarelo.
Fonte: Desenvolvido pelo autor.

predominada apenas pela cor branca, o codificador não determinou como necessário o uso de nenhum preditor além do Planar. Ao longo dos corpos dos dinossauros, que possuem texturas e detalhes complexos, o preditor proposto para o tamanho 16×16 foi utilizado com maior frequência, resultando numa diminuição também no número de blocos 8×8. Ainda, é possível perceber uma redução das regiões onde houve uma repartição maior em blocos de tamanhos menores como 16×16 e 8×8 como nos quatro vértices da imagem.

Para o QP 27 observa-se um aumento no uso dos modos planar e Angulares especialmente na região branca da parede que apresenta padrões angulares como linhas escuras. Isso se deve ao fato de que um QP maior permite um maior erro na predição da imagem, o que permitiu o codificador aplicar modos intra que utilizassem blocos maiores de tamanho 64×64 mesmo que estes providenciem uma predição menos precisa. Este padrão continua para os QPs 32 e 37, com os modos angulares sendo responsáveis por 25% e 42% dos blocos codificados para estes QPs respectivamente. Ressaltamos que não propomos nesta tese o uso de blocos de tamanho 64×64 por esta dimensão não se encontrar presente em outros codificadores como o HEVC e o EVC, o que tornaria a solução proposta menos genérica. Também ressaltamos que acreditamos ser possível aplicar o mesmo método sem modificação alguma para também treinar redes para blocos de tamanho 64×64. Ainda, como os BD-Rates em conjunto com as taxas de seleção demonstram, os ganhos mais relevantes encontram-se nos QPs menores.

Para facilitar a visualização do impacto dos preditores desenvolvidos na estrutura de repartição dos blocos, a Figura 57 traz uma ampliação das sobreposições anteriores para o QP 22 em uma região rica em detalhes como a cabeça do dinossauro ao fundo. Ao observar os particionamentos efetuados pelo codificador padrão, é possível perceber que, em exceção dos blocos 64×64 usando modos angulares nas regiões brancas a sua volta, os blocos possuem uma heterogeneidade muito grande entre modos, tamanhos de bloco e formato dos blocos (aplicando diversos blocos retangulares horizontais e verticais). Essa heterogeneidade requer mais *bits* de sinalização e também diminui a capacidade de compressão do codificador entrópico aplicado posteriormente. Já para a codificação utilizando os preditores desenvolvidos, a grande maioria dos blocos 64×64 angulares é substituída por blocos 32×32. Enquanto na região da cabeça a grande maioria dos blocos assumiram tamanhos 16×16 ou 32×32 sendo preditos pelas redes propostas. Estimamos que o aumento na homogeneidade das informações presentes em texturas complexas é o grande responsável pelos ganhos de BD-Rate e BD-PSNR obtidos pelo método proposto nesta tese.

Além do caso médio, analisaremos os mapas de particionamento também para o melhor e pior caso de teste e para dois outros casos que apresentam comportamentos e informações relevantes. Para cada um dos casos, consideraremos QPs distintos

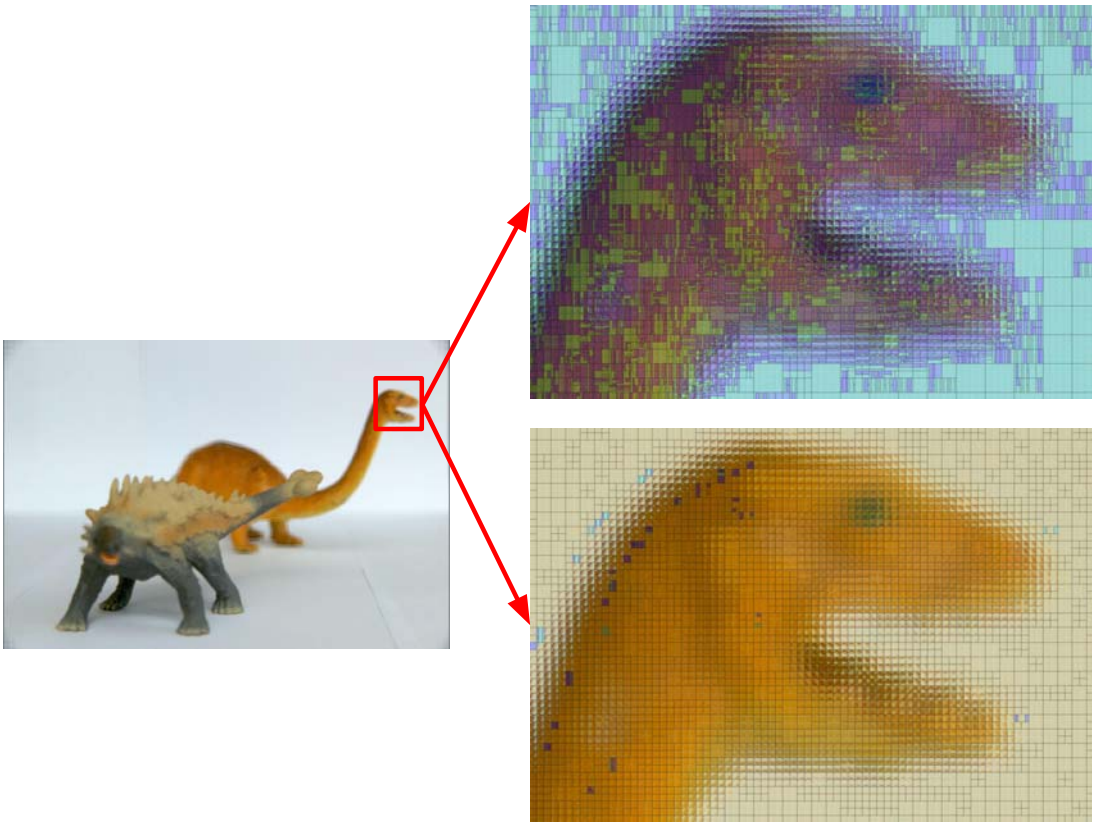
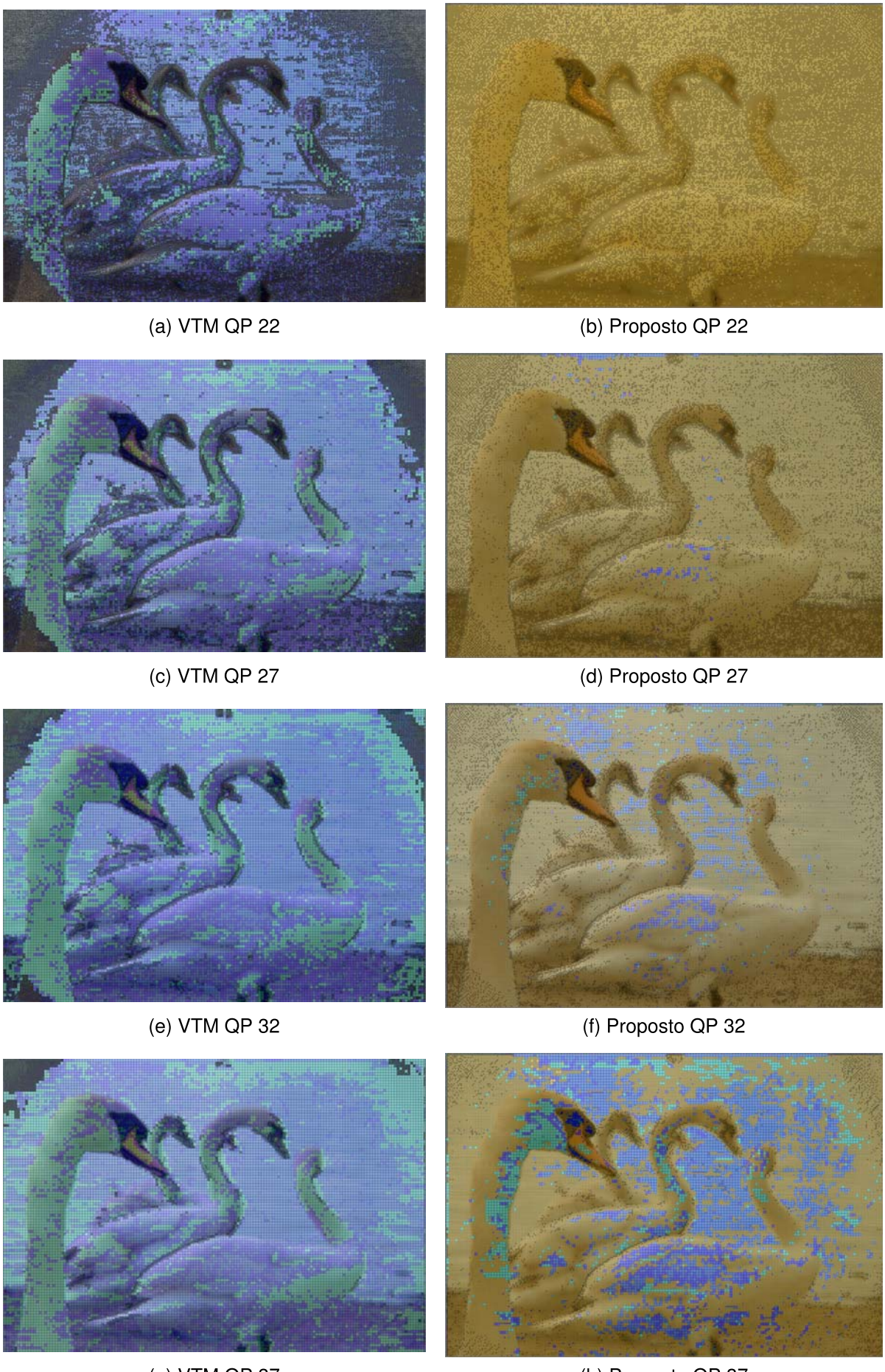


Figura 57 – Comparação entre tamanhos de bloco e modos intra em região de textura complexa entre VTM e Proposto para o LF Ankylosaurus-1 codificado no QP 22.
Fonte: Desenvolvido pelo autor.

para garantir análises em todos os QPs possíveis, isto, enquanto priorizamos também comportamentos relevantes em cada QP.

A Figura 58 apresenta os mapas de particionamento para o LF "*Swnas-2*", cujo é o caso de teste com o melhor BD-Rate (-60,47%). Para os QPs 22 e 27, o codificador utiliza os preditores desenvolvidos para todos os níveis de profundidade e complexidades de textura, desde as penas e bicos dos patos até o horizonte homogêneo coberto por água. As regiões com padrões complexos nos vértices da câmera que demandam uma grande quantidade de blocos menores e retangulares também começam a utilizar blocos de tamanhos menores, diminuindo assim a coloração preta predominante das linhas divisórias dos blocos. Já para os casos codificados com o VTM original, devido às texturas da água e das penas, a maioria dos blocos é predita com modos angulares, com presença praticamente nula (abaixo de 1%) dos modos Planar e DC.

Para os dois QPs mais altos, assim como esperado devido à menor atenção aos detalhes pelo codificador, áreas como a água e as penas dos cisnes começam a ter os detalhes de suas texturas ignorados, levando o codificador a utilizar blocos maiores com o modo Planar. Contudo, para esta sequência é evidente que, para o QP 32, os preditores desenvolvidos continuam sendo aplicados na predição da maior parte da imagem. Ainda, ao analisar o QP 37, pode-se chegar a conclusão de que aproxima-



(a) VTM QP 22

(b) Proposto QP 22

(c) VTM QP 27

(d) Proposto QP 27

(e) VTM QP 32

(f) Proposto QP 32

(g) VTM QP 37

(h) Proposto QP 37

Figura 58 – Mapa dos blocos selecionados para o LF *Swans* para cada QP. Modos do VVC em tons de Azul; Modo proposto em Amarelo.

Fonte: Desenvolvido pelo autor.

damente metade da imagem é predita com os preditores desenvolvidos, enquanto a Figura 52 diz que o preditor foi escolhido para 89% dos casos enquanto os Modos Planar e demais modos angulares foram utilizados aproximadamente 2% e 9% dos demais blocos. Esta diferença ocorre pelo fato de o preditor ser utilizado em regiões complexas para blocos menores de tamanhos 32×32 e 16×16 que, naturalmente, são mais numerosos que os 64×64. Por exemplo, para aplicar um bloco 64×64, o codificador substitui 4 blocos 32×32 ou 16 blocos 16×16. Por outro lado, ao codificar este LF com o VTM original, 26% de todos os blocos são codificados com o modo Planar e 66% com os outros modos angulares, o que fortalece a evidência sobre a relevância dos modos intra inseridos. Ainda, gostaríamos de relembrar que concentrar o máximo possível de blocos sob um mesmo modo e tamanho gera ganhos adicionais de sinalização, então trocar 66% de blocos que utilizam modos angulares diversos por um único modo capaz de predizer padrões em qualquer ângulo é o que também possibilita um ganho de BD-Rate tão expressivo como -60%.

Nesta sequência, considerou-se duas regiões específicas como de interesse. A região das patas dos cisnes por conterem diferenças de profundidade e texturas complexas, assim como a cabeça do dinossauro avaliada no caso anterior. E também a região das penas que, apesar de não possuir diferenças de profundidade, devido à sua textura, o VTM apresentou uma alta heterogeneidade de modos e tamanhos de blocos.

A Figura 59 apresenta o recorte das estruturas de blocos da região ampliada de uma das pernas do cisne para os QPs 22 e 27. Observe que no centro da pata (região mais escura), o VTM sem modificações acaba utilizando muitos blocos retangulares, em sua maioria verticais, para melhor reproduzir as texturas das patas. Ainda nesta região, os blocos com cores azul clara representam modos DC e Planar, os escuros abaixo de 34 e os verdes acima. Dito isto, observe que nas bordas das patas, para ambos os QPs, modos verticais são utilizados com maior frequência. Pois, nesta região, o VTM deve interpolar as linhas verticais pretas referentes ao contraste das bordas laterais das patas. Ao introduzir no codificador os dois preditores desenvolvidos, estes realizam a predição de praticamente todos os blocos.

Ainda, também há uma diminuição considerável na heterogeneidade dos tamanhos e formatos dos blocos que, em sua maioria, assumem tamanhos 16×16 para o QP 22 e 32×32 no QP 27. Havendo apenas 4 blocos 32×32 e outros 4 blocos 64×64 que não são preditos no QP 27 pelos modos inseridos, enquanto o VTM original codificou praticamente todos blocos com tamanhos 64×64 a possivelmente custo de precisão das texturas. Logo, afirma-se que os blocos 64×64 foram substituídos por blocos 32×32 e 16×16 pelo fato dos preditores desenvolvidos proverem predições muito superiores aos modos intra disponíveis para estes dois tamanhos de bloco no VTM. Embora o número de blocos aumente, a heterogeneidade dos modos escolhidos e

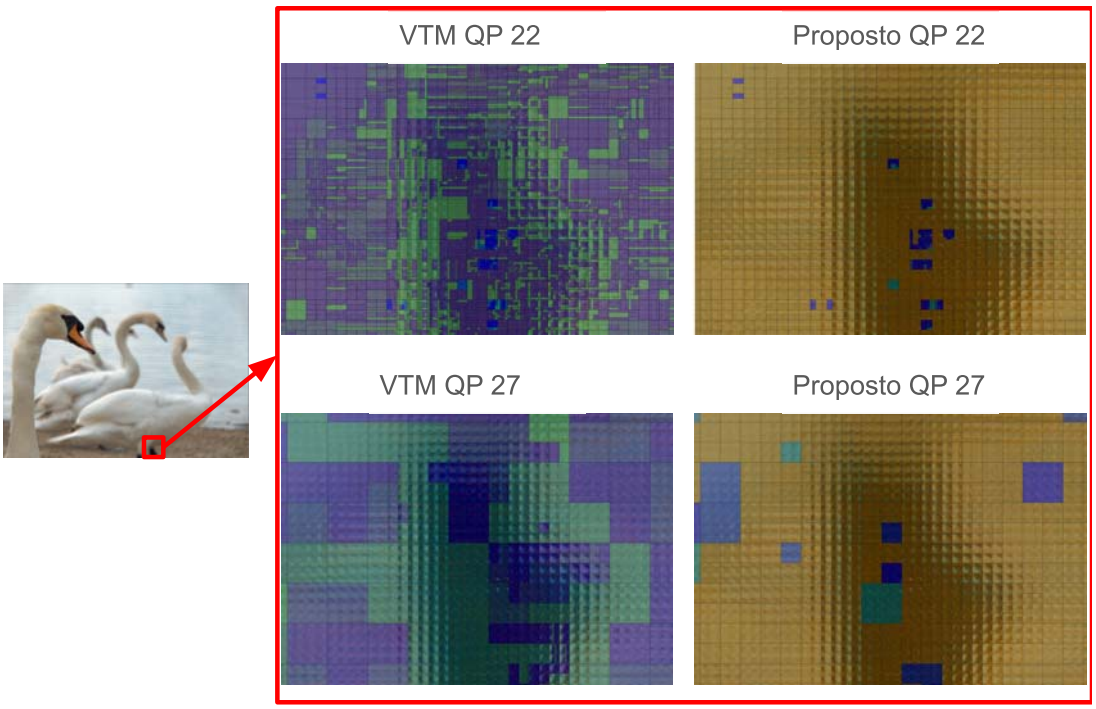


Figura 59 – Ampliação da estrutura de particionamento da sequência *Swans-2* n região da perna do cisne.
Fonte: Desenvolvido pelo autor.

tamanhos de bloco compensam o aumento.

Outra região interessante de ser analisada é a região do inínicio das asas do segundo cisne. Isto porque, a região apresenta apenas uma textura branca sem desníveis de profundidade, porém, ainda assim o VTM apresentou dificuldade predizê-la com blocos maiores, empregando diversos blocos retangulares (em sua maioria verticais), e diferentes modos angulares horizontais e verticais. Ainda, observando mais detalhadamente estes blocos retangulares, é possível ver que em alguns casos as MIs são estão sendo repartidas em diferentes partes, o que pode prejudicar ou dificultar a exploração de suas redundâncias angulares entre estes blocos. Por fim, perceba que ao inserir os preditores desenvolvidos, todos os blocos neste recorte foram preditos por eles e que todas MIs se encontram alinhadas com os blocos.

Para exemplificar como nossos preditores beneficiam a codificação em QPs mais altos e o porquê a sua utilização acarreta em um maior número de blocos, a figura apresenta a mesma região para o QP 37. Enquanto o VTM original utiliza apenas blocos 64×64 para predizer essa região homogênea, o que tende a implicar em perda de detalhes, o VTM contando com nossos preditores pode utilizar tamanhos 32×32 em todos blocos (com exceção de um), para obter uma maior atenção aos detalhes da textura sem prejudicar sua eficiência.

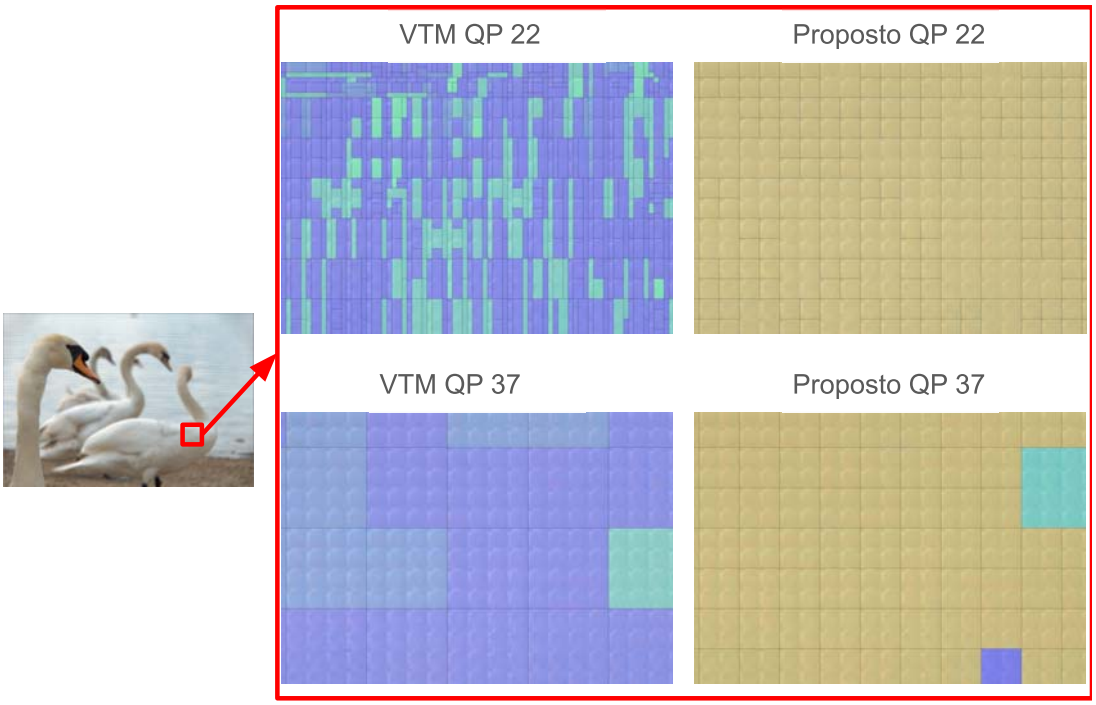
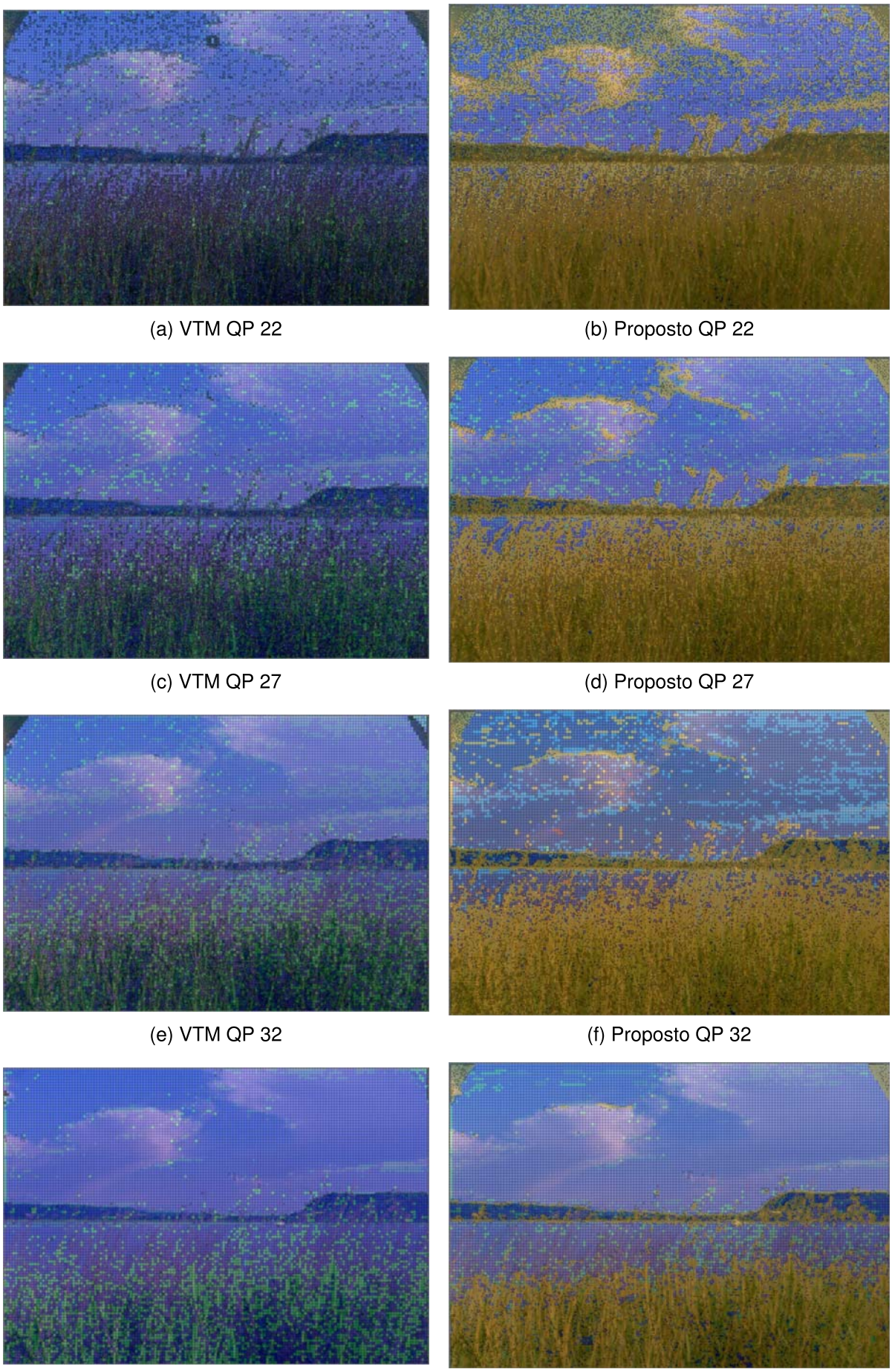


Figura 60 – Ampliação da estrutura de particionamento da sequência *Swans-2* n região da asa do cisne.
Fonte: Desenvolvido pelo autor.

A sequência "*Reeds*" que atingiu o menor ganho em BD-Rate (-29,31%) e também as menores taxas de redução de blocos possui três características importantes. Primeiramente tem-se na metade inferior do LF uma textura muito complexa constituída pelos matos próximos a câmera que possuem as mais diversas direções. Em contrapartida, ao fundo dos matos tem-se o céu homogêneo e distante no horizonte. Por fim, o céu ainda apresenta nuvens com suas texturas específicas.

Analisando o mapa de repartições para esta sequência na Figura 61, pode ser visto pelo tom quase preto gerado pelas linhas de repartição, que o VTM original emprega blocos pequenos de diversos formatos na região inferior da imagem. Nesta região também são aplicados diversos modos angulares horizontais e verticais para predizer as complexas texturas geradas pelos matos omnidirecionais. Já na parte superior da imagem, o codificador emprega em grande maioria o modo Planar em blocos 64×64 devido o céu apresentar apenas texturas simples, com gradientes de tons de azul muito propícios para o uso deste modo. Contudo, note que as texturas das nuvens são quase que ignorados, e estas são preditas também em grande parte pelo modo planar, apresentando raramente alguns modos angulares verticais.

Ao incluir os preditores desenvolvidos no codificador, as texturas complexas dos matos são preditas em sua totalidade por estes. Ainda, note que as nuvens são par-



(a) VTM QP 22

(b) Proposto QP 22

(c) VTM QP 27

(d) Proposto QP 27

(e) VTM QP 32

(f) Proposto QP 32

(g) VTM QP 37

(h) Proposto QP 37

Figura 61 – Mapa dos blocos selecionados para o LF *Reeds* para cada QP. Modos do VVC em tons de Azul; Modo proposto em Amarelo.

Fonte: Desenvolvido pelo autor.

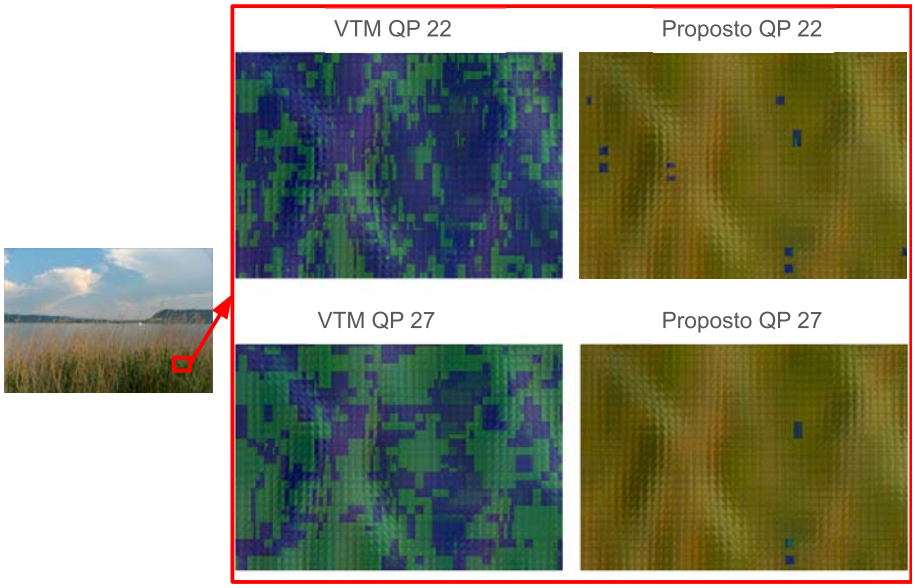


Figura 62 – Ampliação da estrutura de particionamento da sequência "*Reeds*".
Fonte: Desenvolvido pelo autor.

ticionadas em blocos menores que utilizam também os preditores desenvolvidos. Tal comportamento nos permite assumir que a melhora na predição das texturas das nuvens é significativa o suficiente para tornar o preditor proposto mais interessante que o modo planar, mesmo necessitando utilizar um maior número de blocos em vista da ausência de um preditor neural de 64×64 píxeis.

Para o QP 27, enquanto o VTM começa a usar modos angulares com maior frequência para predizer a parte inferior da imagem e o Planar para predizer as nuvens, os preditores desenvolvidos ainda são responsáveis por predizer toda a região inferior da imagem e as bordas das nuvens que contrastam com a parte mais escura do céu. No caso do QP 32, nota-se a utilização de blocos maiores na região dos matos para ambos os casos. Já na região do céu, enquanto o VTM original aumenta o uso do modo planar em toda a região superior da imagem, o VTM modificado segue aplicando os modos propostos nas regiões de bordas das nuvens e vértices da imagem. Por fim, ao codificar o LF com o QP 37, o modo Planar é utilizado na maior parte da imagem, subindo sua taxa de uso de 35% no QP 32 para 60%, enquanto modos angulares podem ser vistos mais frequentemente nas regiões inferiores e vértices da imagem. Note também que para este QP, a parte superior dos matos é predita por modos planares. O que, provavelmente, ocasionou perdas de qualidade ou artefatos indesejados nestas regiões. Por outro lado, o VTM modificado ainda aplicou os preditores desenvolvidos nas regiões inferiores e superiores aos matos, mesmo que estes utilizem blocos menores que 64×64.

A sequência "*Ceiling-Light*" demonstrou beneficiar-se mais de blocos 16×16 que

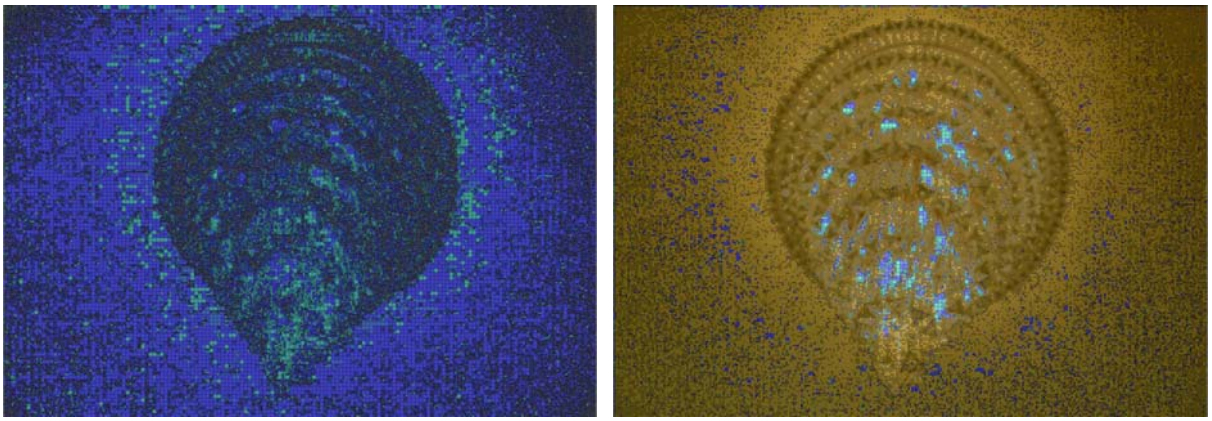
32×32 ao longo de nossos experimentos, apresentando o menor ganho de BD-Rate pelo preditor 32×32. Ainda, esta sequência apresentou a maior diferença de BD-Rate entre os dois tamanhos de bloco de todas as sequências avaliadas, com o uso do preditor 16×16 melhorando seu BD-Rate em -7,7% quando comparado à performance do 32×32.

Analisando a Figura 63, é possível perceber que os detalhes presentes no lustre na região central da figura apresentam diferenças bruscas e graduais de iluminação que, consequentemente, geram altos impactos nas amostras de luma. Estes comportamentos, levaram o codificador a intensificar as repartições de blocos nesta região, enquanto utilizou o modo Planar e blocos maiores na região mais homogênea em torno da região central. Outra região de interesse a se observar é a região inferior do lustre, onde blocos maiores e modos angulares são utilizados mesmo no QP22. Nesta região note que, ao utilizar os nossos preditores para o QP 22, o VTM seguiu aplicando os seus modos angulares apenas na parte superior da região. Isto por se tratar de uma zona consideravelmente clara e homogênea. Já para as demais áreas do lustre com maiores detalhes de luz e profundidade, os preditores desenvolvidos foram amplamente empregados. Ao comparar as duas Figuras 63a e 63b é possível inclusive perceber que todo o formato externo do lustre e seus detalhes de luz ficam perceptíveis na figura à direita devido à homogeneidade dos tamanhos de blocos aplicados, enquanto a complexidade das repartições de diversos formatos e tamanhos aplicados pelo VTM original dificulta a visualização do lustre na figura à esquerda.

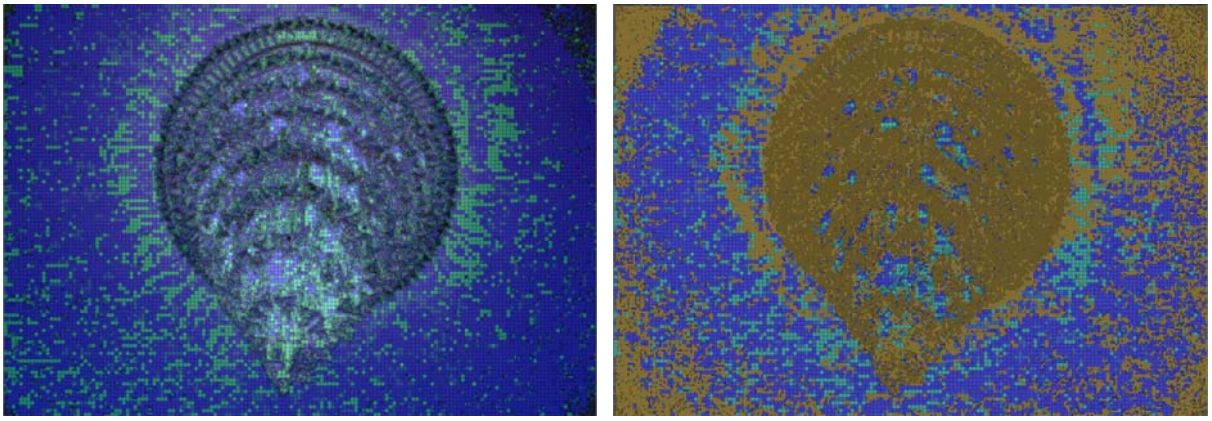
Analisando as regiões em torno do lustre constituídas apenas pelo teto branco com padrões gradientes de luz, é possível notar que os blocos foram em grande parte preditos pelos preditores desenvolvidos. Neste cenário, isto indica que os preditores têm uma boa capacidade de predizer também padrões de iluminação que antes necessitavam de diversos modos angulares distintos. Ainda, houve um uso maior de blocos 32×32 nas áreas próximas ao redor do lustre devido à maior iluminação e blocos 16×16 nas áreas mais escuras por estarem distantes do lustre.

Conforme os QPs aumentam, o uso de blocos 64×64 também aumenta, fazendo com que os preditores desenvolvidos deixem de ser utilizados nas áreas mais homogêneas em torno do lustre. Porém, na região do lustre que apresenta diversas bordas devido aos vários objetos presentes, o codificador ainda necessita aplicar um considerável número de blocos menores e retangulares. E ao ter os preditores desenvolvidos à disposição, o codificador ainda aplica largamente os preditores desenvolvidos nestas regiões que apresentam maior complexidade de detalhes, iluminação e profundidade. Desta forma, pode ser afirmado que os preditores são capazes de predizer padrões e texturas complexas também em cenas com contraste entre alta claridade e sombras, a ponto de beneficiar o codificador mesmo quando a atenção em tais detalhes é menor.

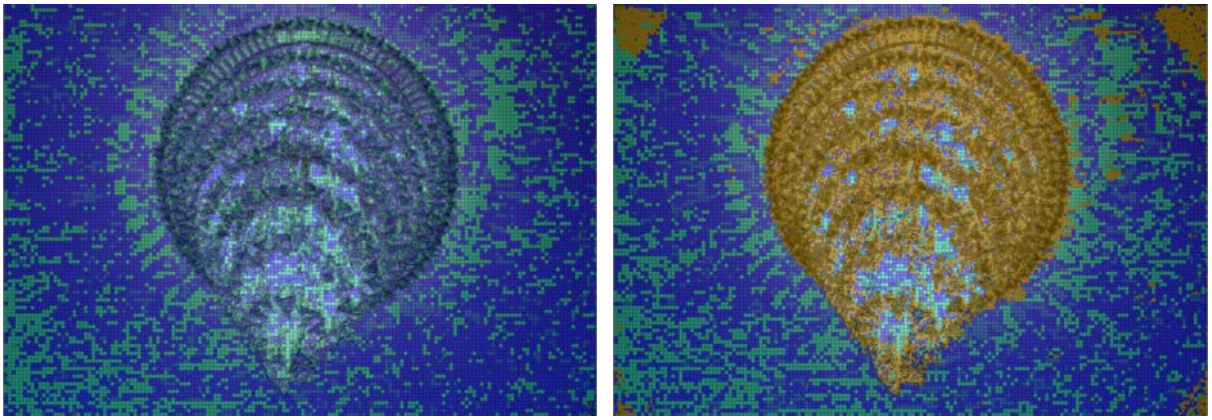
Para demonstrar mais detalhadamente os impactos positivos da inserção dos pre-



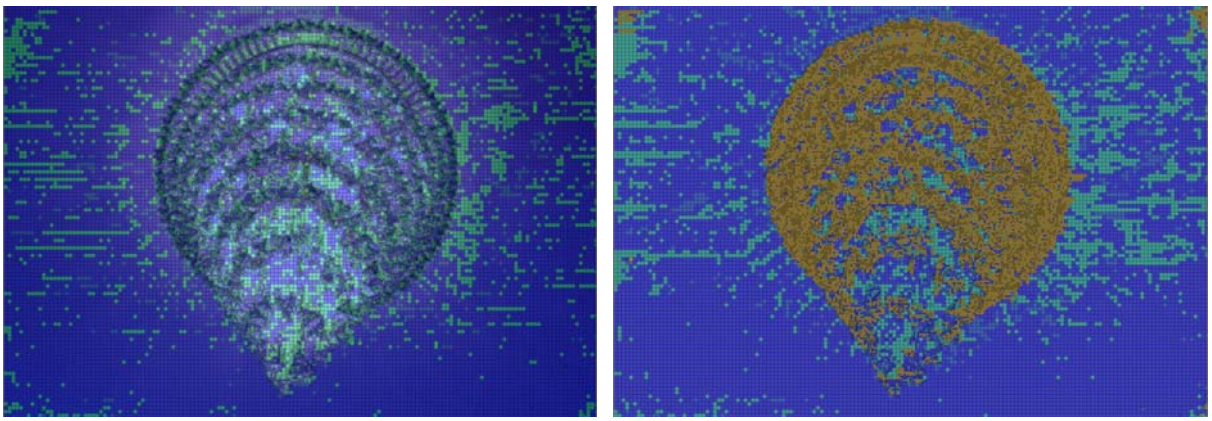
(a) VTM QP 22 (b) Proposto QP 22



(c) VTM QP 27 (d) Proposto QP 27



(e) VTM QP 32 (f) Proposto QP 32



(g) VTM QP 37 (h) Proposto QP 37

Figura 63 – Mapa dos blocos selecionados para o LF *Ceiling Light* para cada QP. Modos do VVC em tons de Azul; Modo proposto em Amarelo.
Fonte: Desenvolvido pelo autor.

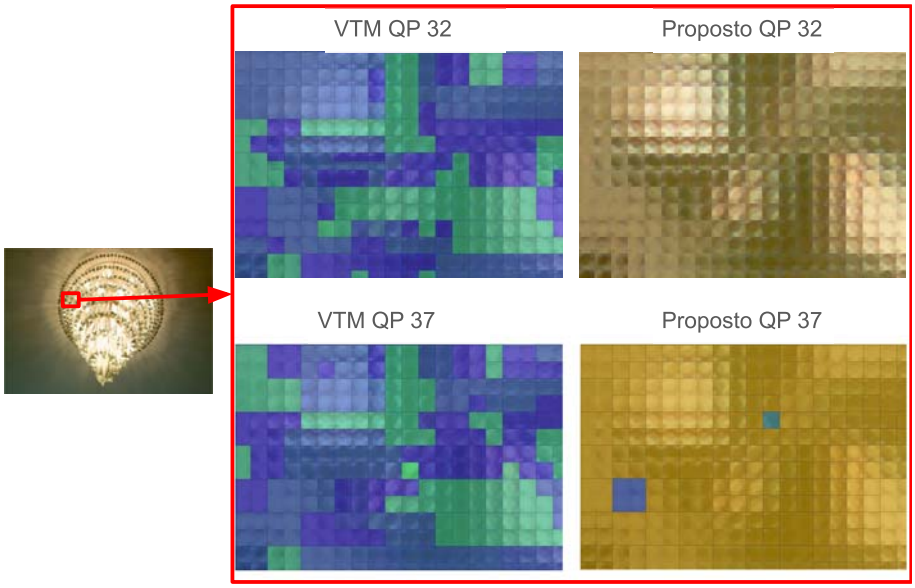


Figura 64 – Ampliação da estrutura de particionamento da sequência "*Ceiling-Light*".
Fonte: Desenvolvido pelo autor.

ditores, a Figura 64 apresenta a ampliação de uma região que seguiu utilizando um alto número de blocos 16×16 e 32×32 mesmo para os QPs 32 e 37 apresentados. A região para o QP 32 deixa evidente que, apesar de blocos 16×16 serem largamente mais presentes, o codificador ainda emprega blocos retangulares e modos intra dos índices mais baixos (tons azuis claros) até os mais altos (tons verdes escuros). Blocos e modos estes que são substituídos unicamente pelo preditor proposto de tamanho 16×16.

Para o QP 37, um padrão similar ocorre. Torna-se fácil visualizar na ampliação à esquerda que o codificador emprega modos intra que diferem consideravelmente entre MIs que são vizinhas. Já ao analisar a figura à direita, é perceptível o maior número de blocos 32×32 e que apenas 1 bloco 32×32 e 1 bloco 16×16 não foram preditos com o modos propostos. Em vista de que MIs próximas tendem a possuir formatos muito similares, esses comportamentos distintos apresentados entre o VTM original e o modificado demonstram a adequação do VVC de identificar e predizer as correlações espaciais e angulares entre as MIs providas pelos preditores neurais propostos neste trabalho.

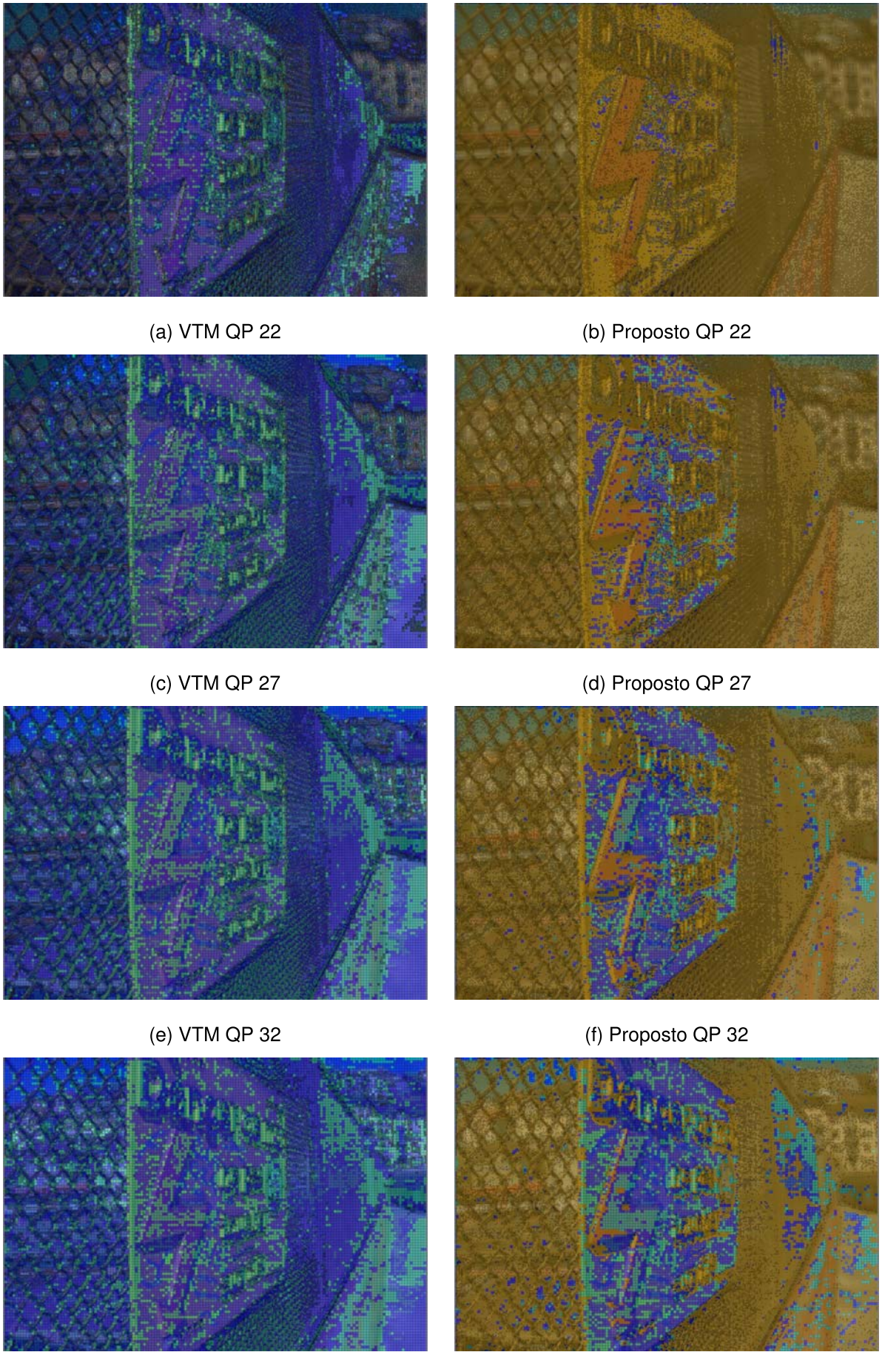
Outro caso de teste a ser estudado é o LF "*Danger de Mort*" que possuiu a maior redução de blocos utilizados na codificação do QP 22 e 27 e obtendo um BD-Rate de -49,13%, apenas 2,24% acima da média. Entre as particularidades de interesse na cena deste LF que pode ser observado na Figura 65, está a presença e ângulo da grade que não somente apresenta texturas complexas pela junção dos orifícios à medida que a grade se afasta gradualmente da câmera, como constantes contrastes

entre a grade e os prédios ao fundo. Estes contrastes são desafiadores de serem preditos pelo fato destas discrepâncias entre a profundidade e características da grade e dos prédios ao fundo ocasionarem em mudanças significativas entre MIs e píxeis próximos. Ainda, ao centro da imagem, o LF apresenta uma placa com cores homogêneas e profundidade estável. Este conjunto de características tornam este LF um caso de teste desafiador e completo.

Na região central do LF que apresenta a placa com texturas mais simples, a grande maioria dos blocos 64×64 são substituídos por blocos 32×32 e 16×16 preditos pelas redes inseridas no codificador. Restando nesta região apenas alguns detalhes que foram codificados principalmente com modos MIPs e planares. Ainda, é notável que as regiões como as bordas das letras, símbolos e da placa fora particionadas em blocos 16×16 em sua majoritária parte. Já na região das grades e prédios ao fundo, devido às complexas características mencionadas no parágrafo anterior, foram preditas quase que completamente pelo preditor treinado para blocos 16×16. Nota-se uma maior utilização de blocos 32×32 apenas em regiões de cores mais sólidas e homogêneas, como a calçada, muros, paredes e a grade mais ao fundo que apresenta apenas tons de cinza. É possível concluir também que a grande redução no número de blocos para o QP 22 ocorre pelo fato de todos os blocos 8×8, 16×16, retangulares ou não, que são necessários para predizer as texturas e profundidades complexas presentes em toda a região à esquerda do prédio ao fundo no LF terem sido substituídos por blocos 16×16 preditos unicamente pela rede neural proposta.

Observe também que, para as regiões da placa, do fundo da grade e parte do chão, o VTM original utiliza maioritariamente blocos 64×64 em todos os QPs. Com a maior mudança no comportamento das partições entre os QPs estando nos orifícios da grade mais próximos da câmera. Já o VTM com os modos propostos, os utiliza largamente nestas regiões até mesmo para o QP 37, sendo utilizado para mais de 90% dos blocos neste QP. Fato que evidencia ainda mais a complexidade de predizer estas regiões e a competência das redes propostas em fazê-lo.

Para o estudo com ampliação das estruturas de partição, selecionamos a região central da intersecção dos arames da grade justamente por apresentar o maior desafio para os preditores do codificador para este LF. Observando as ampliações presentes na Figura 66, fica visível o impacto que as diferenças de profundidade nas bordas do arame causam no deslocamento angular dos píxeis das MIs. Também fica evidente o maior uso de blocos retangulares e de menores dimensões nessas regiões, especialmente para o QP 22. Já ao analisar as ampliações das estruturas de particionamento do VTM com os preditores desenvolvidos, independentemente da complexidade da região, os blocos 16×16 foram completamente capazes de predizer as MIs. Havendo apenas um único bloco 16×16 codificado com outro modo. Por fim, na região menos complexa, os blocos 64×64 foram substituídos por conjuntos de blocos 32×32 e



(a) VTM QP 22

(b) Proposto QP 22

(c) VTM QP 27

(d) Proposto QP 27

(e) VTM QP 32

(f) Proposto QP 32

(g) VTM QP 37

(h) Proposto QP 37

Figura 65 – Mapa dos blocos selecionados para o LF *Danger de Mort* para cada QP. Modos do VVC em tons de Azul; Modo proposto em Amarelo.

Fonte: Desenvolvido pelo autor.

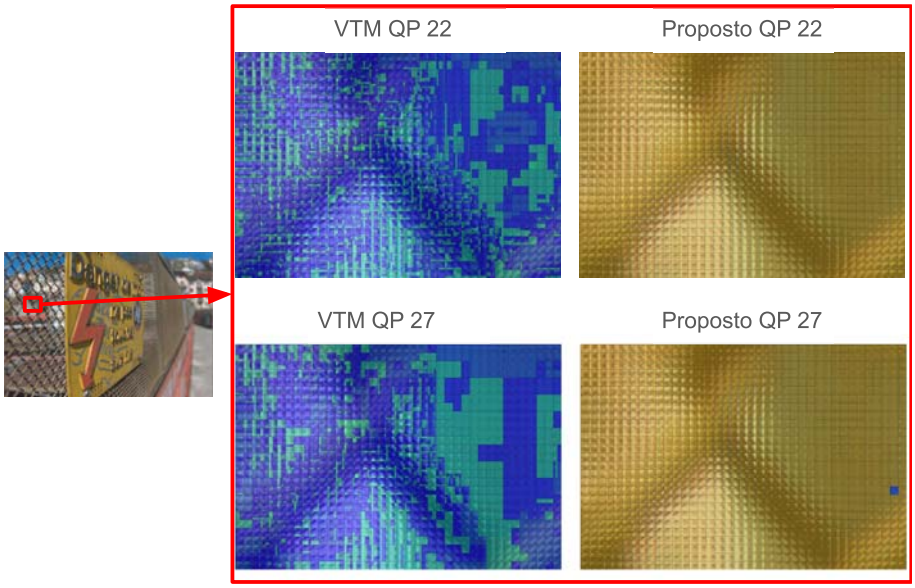


Figura 66 – Ampliação da estrutura de particionamento da sequência "*Danger-de-Mort*".
Fonte: Desenvolvido pelo autor.

16×16.

6.2.1 Validação dos Resultados: Dataset IRIS

Para validar a performance dos preditores desenvolvidos e a generalidade provida pelas técnicas de treinamento adotadas, o *dataset* IRIS (RODRIGO C. DAUDT, 2017) foi codificado. Todas as sequências relevantes para esta discussão estão presentes na Figura 67. Apesar dos 33 LFs presentes neste *dataset* terem sido obtidos com a mesma câmera *Lytro Illum*, suas cenas possuem características e desafios distintos do *dataset* EPFL. Como, por exemplo, as sequências "*LowLight*" (Figuras 67c-f), que possuem iluminação muito reduzida e apresentam uma alta quantidade de ruído. Ou ainda a "*MotionBlur*" (Figura 67g), que apresenta texturas distorcidas devido à alta movimentação. Além disso, nenhuma das sequências presentes nele foi utilizada no processo de treinamento.

Analisando os resultados de BD-Rate para este *dataset* apresentado pela Tabela 21, nota-se que os preditores melhoraram a codificação em -36,09%. Uma eficiência 10,8% menor do que para o *dataset* inicial EPFL. Esta perda de eficiência acaba sendo natural em vista dos novos desafios apresentados por este *dataset* conforme discutido no parágrafo anterior. Contudo, note que existem alguns casos discrepantes, como o "*LowLight-House*", "*LowLight-Street*" e "*LowLight-Flowers*" (Figuras 67c-f), que possuem BD-Rates em torno de -15%. Teorizamos que estes BD-Rates baixos ocorrem pela presença de ruídos intensos que acabam perturbando inclusive os padrões de MIs. Na sequência "*LowLight-House*" representada na Figura 67d, estes ruídos ficam

evidentes. Ademais, a sequência "*Texture*" (Figura 67h) é constituída por uma textura vermelha com detalhes muito simples, o que é propício para o uso do modo planar e tamanhos de bloco maiores. Conforme discutido nas análises anteriores, isto leva o VVC a não necessitar tanto dos preditores desenvolvidos para realizar as predições, o que leva a uma diminuição nos ganhos obtidos.

Tabela 21 – BD-Rate para o light fields do dataset IRIS.

Sequências	Proposto 32+16	
	BD-Rate	**BD-PSNR**
Bee-1	-32,5	1,03
Bee-2	-42,99	1,71
Building	-43,45	1,67
Bumblebee	-59,83	3,58
Checkerboard	-45,84	1,14
ChezEdgar	-21,14	0,68
Corridor	-27,63	0,85
DistantChurch	-22,87	0,67
Doves	-55,97	2,63
Duck	-54,74	2,99
Framed	-40,29	1,27
Fruits	-58,06	2,93
LensFlare	-41,89	1,94
LowLight-Flowers	-16,73	0,44
LowLight-House	-7,96	0,21
LowLight-Roundabout	-58,59	2,83
LowLight-Street	-14,89	0,28
MessyDesk	-40,81	1,2
Mini	-36,89	1,2
MotionBlur	-33,92	0,95
Perspective	-30,39	0,86
Plushies	-27,84	0,69
Posts	-26,32	0,69
Rond	-33,3	1,04
Sculpture	-36,03	1,09
Sign	-40,15	1,37
Statue	-32,54	1,06
Steps	-29,74	1,06
SunnyRose	-48,94	2,37
Symmetric	-35,24	0,88
Texture	-20,93	0,45
TinyMoon	-34,66	1,15
Translucent	-37,93	0,97
Média	-36,09	1,33

Já para os demais casos, a grande maioria das sequências atinge BD-Rates superior à -30%. Com os LFs "*Bumblebee*" (Figura 67a) e "*LowLight-Roundabout*" (Figura 67e) atingindo reduções de -59,83% e -58,59% respectivamente, sendo estas as duas maiores reduções obtidas para este *dataset*. O fato de o LF "*LowLight-Roundabout*" ser da categoria *LowLight* e ter atingido uma das melhores eficiências de codificação atesta que os preditores atingiram baixas reduções nos demais LFs da mesma categoria não pela sua baixa iluminação, mas sim pela presença de ruídos não naturais a LFs. Ainda, ressaltamos que apenas a sequência "*Swans-2*" os supera em ganhos de eficiência, atingindo um BD-Rate de -60,47%. Um exemplo interessante de se analisar é a sequência "*Doves*" , que por possuir pombas brancas com texturas similares aos cisneis da sequência "*Swans-2*" , acabou atingindo um BD-Rate similar de -55,97%.

As análises realizadas neste capítulo deixam claro que inserir preditores baseados em redes neurais como modo intra de codificadores de vídeo pode aumentar substancialmente suas eficiências de compressão ao codificar *light fields*. Esta adaptação do codificador estado da arte VTM para LFs abre espaço para diversas novas perguntas e maneiras de melhorá-lo neste tópico, considerando a presença dos preditores neurais e seus impactos no processo de codificação.

Neste capítulo, a capacidade de generalização dos preditores propostos foi testada através da inserção destes sem modificação alguma no codificador VVC. Para melhor entendermos de onde vêm os ganhos de performance, uma análise detalhada do comportamento do codificador, de seus blocos e de seu uso dos modos intra foi realizada para todos os LFs com comportamentos considerados relevantes. Além de validarmos os preditores em um novo codificador, também validamos seus desempenhos com um novo *dataset*, onde nenhuma de suas sequências esteve presente nos seus treinamentos. Além deste desafio, o *dataset* IRIS apresenta diversos novos desafios para os preditores por haver cenas com características distintas dos LFs presentes no *dataset* de treino EPFL. Apesar disto, os preditores propostos atingiram um BD-Rate expressivo de -36,09%. Com estes resultados, conclui-se as avaliações realizadas nesta tese, restando um novo âmbito de conhecimento para a extração de conclusões.

(a) BumbleBee

(b) Doves

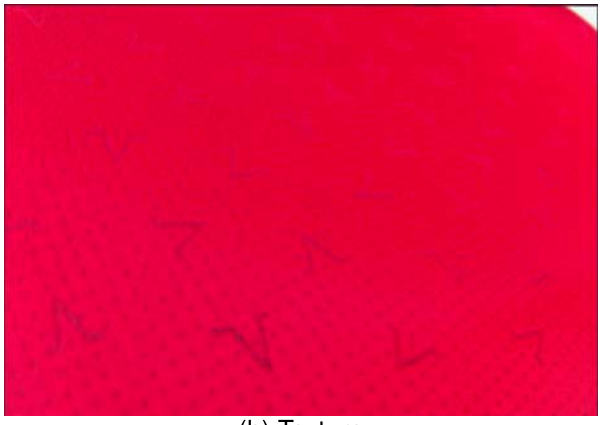
(c) LowLight-Flowers

(d) LowLight-House

(e) LowLight-Roundabout

(f) LowLight-Street

(g) MotionBlur



(h) Texture

Figura 67 – Sequências do dataset IRIS com comportamentos relevantes.
Fonte: Desenvolvido pelo autor.

7 CONCLUSÃO

Esta tese de doutorado apresentou um método de desenvolvimento e integração de preditores neurais intra-quadro para *Light Fields* capaz de adaptar codificadores de vídeo para a compressão deste tipo de imagens. No desenvolvimento deste método, foram desenvolvidas e avaliadas diferentes técnicas a fim de encontrar as mais eficientes para cada uma de suas etapas.

O treinamento das redes neurais foi dividido nas etapas de *Dataset*, *Data Augmentation*, *Learning Rate* e *Loss Function*. Entre os dois meios de adaptar o *dataset* para codificadores de vídeo avaliados, constatou-se que realizar preenchimento de vistas até uma dimensão angular de 16×16 provê ganhos médios em torno de -11% de BD-Rate. Conforme nossas análises posteriores revelaram, o VTM utiliza blocos maiores para o QP 22 quando tem à disposição os preditores desenvolvidos, logo, LFs com MIs 16×16 facilitam o uso de blocos 16×16 pelo codificador ao invés de 8×8, reduzindo o número de blocos necessários para codificar os LFs e, consequentemente, gerando este ganho médio de BD-Rate mesmo com o *overhead* das vistas copiadas. Sobre os hiperparâmetros de treinamento, concluiu-se que a técnica *data augmentation* em conjunto com um decaimento de *learning rate* exponencial e a *loss function* SATD providenciaram as melhores eficiências de codificação para cada uma destas etapas.

Após definidas as técnicas e hiperparâmetros mais eficientes para o treinamento das redes propostas, avaliamos modificações na topologia das redes neurais com a finalidade de melhorar seu desempenho e diminuir seu número de parâmetros. Entre as 5 topologias comparadas, foi possível verificar que utilizar camadas de convolução transpostas no lado do decoder providencia uma capacidade de predição 11% melhor do que utilizar camadas de convolução convencional em conjunto com camadas de *up sampling*. Este ganho em predição permitiu reduzirmos o tamanho da rede em 15% através da exclusão de uma camada de convolução com uma perda de apenas 4% de BD-Rate.

Com a rede e seu treinamento definidos, avaliamos os ganhos de desempenho com *pruning* não estruturado e estruturado para melhorar a eficiência de inferência

das redes e também reduzir o número de cálculos necessários para realizarem as inferências. Os resultados dos experimentos, unidos ao fato de a técnica de *pruning* estruturado possuir melhor eficiência em *hardware* levaram-nos a utilizá-lo com uma taxa de poda de 5% no método proposto por esta tese.

Ao inserir os preditores desenvolvidos no codificador VTM, foi possível melhorar sua eficiência de codificação para LFs em -46,89% de BD-Rate e -1,89dB de BD-PSNR. Após a inserção dos preditores desenvolvidos para blocos 32×32 e 16×16, o VTM os utilizou para mais de 90% em codificações com o QP 22 e mais de 50% para codificações com QP 37. Este comportamento fez com que o número de blocos utilizados para comprimir os LFs sob o QP 22 diminuísse em 41,3%, gerando assim os ganhos substanciais de BD-Rate. Ainda, nossas análises permitiram detectar que os preditores desenvolvidos só não são utilizados em regiões homogêneas e de baixa complexidade de predição, fazendo-os ser substituídos predominantemente por blocos 64×64, que não foram cobertos neste trabalho para mantê-lo genérico.

Ao compararmos os resultados obtidos no VVC por esta tese com os BD-Rates apresentados pelo JPEG Pleno no seu modo de transformadas focado para LFs densos, superamos seus resultados médios em -16,24%. Se compararmos os resultados obtidos para os três LFs em comum entre este trabalho e Carvalho et al. (2018), ainda atingimos resultados competitivos, os superando em -0,36%. Ademais, comparando os resultados obtidos no HEVC com a predição intra para LFs de dois estágios proposta para este codec em (MONTEIRO et al., 2017), os superamos em -17,27% de BD-Rate e 0,73 dB de BD-PSNR.

Com os resultados e análises desenvolvidos, abriu-se uma nova fronteira de experimentos que podem ser desenvolvidos para aprimorar ainda mais o método proposto e também especializá-lo para codificadores em específico como o VVC. Para aprimorar o desempenho no VVC, redes adicionais podem ser desenvolvidas para blocos com diferentes tamanhos e formatos, como os blocos quadrados 64×64, os quais são amplamente utilizados em altos QPs e oferecem potencial para ganhos na compressão e eficiência do codificador. Ajustes como a redução de modos intra angulares e a exclusão de blocos menores também podem acelerar o tempo de codificação. Ainda, o esquema de sinalização pode ser modificado considerando as taxas de uso dos preditores desenvolvidos.

Técnicas como quantização e *pruning* permitem otimizar redes neurais para codificação *end-to-end*, eliminando a necessidade de ferramentas externas ao comprimir o espaço latente, enquanto NERFs também possibilitam a compressão eficiente de *light fields*. Além disso, a viabilidade de execução em FPGAs é essencial para dispositivos móveis, podendo ser alcançada com técnicas como *knowledge distillation*, que reduzem o tamanho e a complexidade das redes. Avaliar os modelos em simuladores de hardware é indispensável para ajustar os preditores e garantir a eficiência em cenários

com recursos limitados.

7.1 Trabalhos Futuros

O método proposto nesta tese para adaptação para *light fields* de codificadores de vídeo, apesar dos altos ganhos obtidos, ainda pode ser melhorado em diversos níveis do processo. Entre estes níveis, pode-se considerar as redes, seu treinamento, o funcionamento dos codificadores e a eficiência da rede em FPGAs.

Pensando no contexto estudado nesta tese de aplicar as redes neurais como preditores intra em codificadores de vídeo, redes adicionais para outros tamanhos e formatos de bloco presentes no VVC podem ser desenvolvidas sem modificações ao método para especializar o que foi proposto para este codificador. Conforme foi discutido, o VTM faz largo uso de partições retangulares para melhor predizer regiões com texturas complexas. Da maneira que os treinamentos e redes foram implementados, treinar as redes para blocos retangulares é possível sem modificação alguma. Contudo, restam dúvidas se é um caminho de pesquisa válido de ser perseguido em vista de que o VTM utilizou blocos retangulares, na maioria das vezes, com dimensões 16×8, 8×16 ou menores, casos estes que sempre dividem uma MI em múltiplos blocos. O que nos leva a acreditar que blocos retangulares com dimensões menores que 32×16 ou 16×32 não vão gerar benefícios relevantes para o VTM. Ainda sobre redes para dimensões de bloco adicionais, ficou evidente nas análises que blocos 64×64 são largamente empregados na grande maioria dos casos de teste para os QPs mais altos. Desta maneira, conjecturamos que treinar redes para blocos quadrados de tamanho 64×64 vá providenciar ganhos substanciais para o VTM, fazendo com que os preditores desenvolvidos sejam utilizados mais frequentemente também nos QPs mais altos. Além da maior utilização do preditor gerando homogeneidade no número de modos escolhidos, a tendência de se utilizar blocos 64×64 também será aumentada, o que, por sua vez, reduz o número de blocos necessários para codificar uma imagem. Essa maior utilização do modo proposto, em conjunto com a redução no número de blocos codificados, leva o codificador de entropia a ser capaz de atingir uma maior taxa de compressão.

As análises deste capítulo também evidenciaram que os modos inseridos são utilizados em 90% dos blocos, sendo preditos para o QP 22 e acima de 50% para o QP 37. Considerando estes fatores, é possível adaptar o codificador para obter ganhos de compressão e tempo de codificação. Para obter ganhos de compressão, pode-se atribuir um sinal menor para o modo proposto. Por exemplo, pode-se atribuir a sinalização especial do modo planar, onde é gasto apenas 1 *bit* ao invés de 3 *bits*. Em vista de que o processo de codificação do VTM tem um custo temporal muito alto (em torno de 24 horas por LF), codificador também pode ser alterado para testar menos

modos ou formatos de bloco. Por exemplo, blocos 8×8 praticamente não são utilizados quando os preditores 32×32 e 16×16 propostos estão disponíveis ao VTM. Isto também pelo fato de que um bloco 8×8 divide uma MI 16×16 em quatro blocos separados. Ademais, avaliar blocos 8×8 demanda demasiado tempo, pois um bloco 64×64 pode ser repartido em 64 blocos 8×8 que precisam ser preditos e avaliados individualmente. Logo, ao se inserir os preditores desenvolvidos para predizer LFs com MIs de 16×16 a avaliação de blocos 8×8 pode ser evitada, gerando ganhos de tempo de codificação para o VTM. De maneira semelhante, diversos modos intra angulares podem ter sua avaliação pulada, especialmente para o QP 22, que os usa para menos de 1% dos blocos. Ao contrário de propor redes para blocos retangulares, o oposto também pode ser experimentado, onde blocos retangulares que dividiriam uma MI em múltiplos blocos não sejam avaliados.

Além de especializar o que foi proposto nesta tese para o codificador VTM, também é possível melhorá-lo ou alterá-lo para obter maiores ganhos de compressão. Com o avanço incessante de técnicas de aprendizado profundo, ainda restam alternativas não avaliadas que podem aperfeiçoar os treinamentos das redes. Técnicas de quantização como as avaliadas em Kulkarni et al. (2021); Zhou et al. (2018); Wu et al. (2020), podem aperfeiçoar a performance das redes assim como o processo de *pruning*. Ainda, com a união destas duas técnicas às redes estudadas aqui, abre-se o caminho para codificações *end-to-end*. Neste formato de codificação, o espaço latente gerado pela rede neural pode ser também quantizado e podado de maneira que vá ser constituído por pesos próximos de 0 e esparsos. Cenário propício para o espaço latente poder ser entropicamente comprimido e, desta maneira, a rede neural dispense a utilização de qualquer outra ferramenta de codificação como o VVC. Ainda, além desta vertente, o conceito de NERFs (*Neural Radiance Fields*) (MILDENHALL et al., 2021) pode ser adaptado para a compressão *end-to-end* de *light fields*. Nessa abordagem, redes neurais são treinadas propositalmente até sofrerem altos índices de *overfitting*, sendo assim capazes de lembrar de todos os aspectos necessários de um cenário 3D. Tendo em mente que LFs são cenas capturadas em 4D e, portanto, encapsulam uma cena 3D, o conceito de NERFs pode também ser aplicado para compressão e armazenamento eficiente de LFs (SHI; GUILLEMOT, 2023). Vale ressaltar que tal técnica traz consigo a vantagem de interpolação da cena entre as vistas do LF.

Ainda, conforme mencionado, é importante averiguar a possibilidade das redes neurais serem executadas em FPGA. FPGAs possuem recursos de memória e processamento limitados, dificultando realizar em tempo real todos os cálculos necessários para executar uma rede neural com muitos parâmetros. Para obter uma estimativa de quantos blocos por segundo as redes propostas conseguem predizer em uma FPGA é necessário realizar síntese e obter dados sobre sua performance e uso de recursos de hardware. Contudo, mesmo que as redes propostas sejam excessivamente grandes

para serem executadas em FPGA em tempo real, existem técnicas que possibilitam reduzir os tamanhos dos preditores desenvolvidos. Por exemplo, *knowledge distillation* possibilita treinar redes neurais menores a partir de redes previamente treinadas (GOU et al., 2021). Desta maneira, existe todo um ramo de pesquisa em torno de mensurar e otimizar o que foi proposto nesta tese para que sua execução em FPGAs possa ser realizada. Viabilizando assim a compressão eficiente de LFs em dispositivos móveis.

REFERÊNCIAS

ADELSON, E. H.; WANG, J. Y. Single lens stereo with a plenoptic camera. **IEEE transactions on pattern analysis and machine intelligence**, [S.l.], v.14, n.2, p.99–106, 1992.

AFONSO, V. **Desenvolvimento de uma arquitetura para estimação de movimento fracionária segundo o padrão emergente HEVC**. 2013. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal de Pelotas, Pelotas.

AGARAP, A. F. Deep learning using rectified linear units (relu). **arXiv preprint arXiv:1803.08375**, [S.l.], 2018.

AHMAD, W. et al. Shearlet transform-based light field compression under low bitrates. **IEEE Transactions on Image Processing**, [S.l.], v.29, p.4269–4280, 2020.

AHMAD, W.; OLSSON, R.; SJÖSTRÖM, M. Interpreting plenoptic images as multi-view sequences for improved compression. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 2017., 2017. **Anais...** [S.l.: s.n.], 2017. p.4557–4561.

AMIRPOUR, H.; GUILLEMOT, C.; GHANBARI, M.; TIMMERER, C. Advanced Scalability for Light Field Image Coding. **IEEE Transactions on Image Processing**, [S.l.], v.31, p.7435–7448, 2022.

ANWAR, S.; HWANG, K.; SUNG, W. Structured pruning of deep convolutional neural networks. **ACM Journal on Emerging Technologies in Computing Systems (JETC)**, [S.l.], v.13, n.3, p.1–18, 2017.

AOMEDIA. **Alliance for Open Media - An Alliance of Global Media Innovators**. Disponível em: <http://aomedia.org/> Acesso em: 30/01/2024.

ASHOK, A.; NEIFELD, M. A. Compressive light field imaging. In: THREE-DIMENSIONAL IMAGING, VISUALIZATION, AND DISPLAY 2010 AND DISPLAY TECHNOLOGIES AND APPLICATIONS FOR DEFENSE, SECURITY, AND AVIONICS IV, 2010. **Anais...** [S.l.: s.n.], 2010. v.7690, p.221–232.

ASTOLA, P.; TABUS, I. Wasp: Hierarchical warping, merging, and sparse prediction for light field image compression. In: EUROPEAN WORKSHOP ON VISUAL INFORMATION PROCESSING (EUVIP), 2018., 2018. **Anais...** [S.l.: s.n.], 2018. p.1–6.

ASTOLA, P.; TABUS, I. Light Field Compression of HDCA Images Combining Linear Prediction and JPEG 2000. In: EUROPEAN SIGNAL PROCESSING CONFERENCE (EUSIPCO), 2018., 2018. **Anais...** [S.l.: s.n.], 2018. p.1860–1864.

BALLé, J.; LAPARRA, V.; SIMONCELLI, E. P. Variational image compression with a scale hyperprior. **arXiv preprint arXiv:1802.01436**, [S.l.], 2018.

BANK, D.; KOENIGSTEIN, N.; GIRYES, R. Autoencoders. **Machine learning for data science handbook: data mining and knowledge discovery handbook**, [S.l.], p.353–374, 2023.

BAYER, J. **Development of a Modular Software Framework for Supporting Architects During Early Design Phases**. 2017. Tese (Doutorado em Ciência da Computação) — University of Kaiserslautern.

BJØNTEGAARD, G. Improvements of the BD-PSNR model. **ITU-T**, [S.l.], 2008.

BOLLES, R. C.; BAKER, H. H.; MARIMONT, D. H. Epipolar-plane image analysis: An approach to determining structure from motion. **International journal of computer vision**, [S.l.], v.1, n.1, p.7–55, 1987.

BOSSEN, F. Common Test Conditions and Software Reference Configurations. **(JCTVC-L1100)**, Geneva, 2013.

BOSSEN, F. et al. **VTM common test conditions and software reference configurations for SDR video**. JVET-T2010-v2. Disponível em: <https://jvet-experts.org/>.

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.

BURRUS, C. S.; PARKS, T. **Convolution Algorithms**. [S.l.]: Citeseer, 1985.

CARVALHO, M. B. de et al. A 4D DCT-based lenslet light field codec. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 2018., 2018. **Anais...** [S.l.: s.n.], 2018. p.435–439.

CARVALHO, M. B. de et al. Supporting Wider Baseline Light Fields in JPEG Pleno With a Novel Slanted 4D-DCT Coding Mode. **IEEE Access**, [S.l.], v.11, p.28294–28317, 2023.

CHELLAPILLA, K.; PURI, S.; SIMARD, P. High performance convolutional neural networks for document processing. In: TENTH INTERNATIONAL WORKSHOP ON FRONTIERS IN HANDWRITING RECOGNITION, 2006. **Anais...** [S.l.: s.n.], 2006.

CHEN, C.-C.; LU, Y.-C.; SU, M.-S. Light field based digital refocusing using a DSLR camera with a pinhole array mask. In: IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING, 2010., 2010. **Anais...** [S.l.: s.n.], 2010. p.754–757.

CHEN, X.; ZHU, J.; JIANG, J.; TSUI, C.-Y. Tight compression: Compressing CNN model tightly through unstructured pruning and simulated annealing based permutation. In: ACM/IEEE DESIGN AUTOMATION CONFERENCE (DAC), 2020., 2020. **Anais...** [S.l.: s.n.], 2020. p.1–6.

CHEN, Y. et al. Light Field Compression Using Global Multiplane Representation and Two-Step Prediction. **IEEE Signal Processing Letters**, [S.l.], v.27, p.1135–1139, 2020.

CHENG, Z.; SUN, H.; TAKEUCHI, Y.; KATTO, J. Learning image compression with soft vector quantization. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION (ICCV), 2020. **Anais...** [S.l.: s.n.], 2020.

CHERNYAK, R. I. Analysis of the intra predictions in H. 265/HEVC. **Applied Mathematical Sciences**, [S.l.], v.8, n.148, p.7389–7408, 2014.

CHOI, K. et al. An overview of the MPEG-5 essential video coding standard [standards in a nutshell]. **IEEE Signal Processing Magazine**, [S.l.], v.37, n.3, p.160–167, 2020.

CONCEICAO, R.; PORTO, M.; ZATT, B.; AGOSTINI, L. LF-CAE: Context-adaptive encoding for lenslet light fields using HEVC. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 2018., 2018. **Anais...** [S.l.: s.n.], 2018. p.3174–3178.

CORRÊA, D. S. **HCLE**: Codificador de Light Fields para altas taxas de compressão com predição baseada em Optical Flow e Phase Correlation. 2021. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal de Pelotas.

COVER, T. M.; THOMAS, J. A. Elements of information theory second edition solutions to problems. **Internet Access**, [S.l.], p.19–20, 2006.

DUMOULIN, V.; VISIN, F. A guide to convolution arithmetic for deep learning. **arXiv preprint arXiv:1603.07285**, [S.l.], 2016.

ELECTROSOME. **LFC** : Light Field Camera Take your snaps.. Focus later.. Acesso em: 21/02/2025. Disponível em: <https://electrosome.com/light-field-camera/>.

FARADAY, M. LIV. Thoughts on ray-vibrations. **The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science**, [S.l.], v.28, n.188, p.345–350, 1846.

FENG, B. Y.; VARSHNEY, A. SIGNET: Efficient Neural Representation for Light Fields. In: IEEE/CVF INTERNATIONAL CONFERENCE ON COMPUTER VISION (ICCV), 2021. **Proceedings...** [S.l.: s.n.], 2021. p.14224–14233.

FERRUGEM, A. P.; ZATT, B.; AGOSTINI, L. V. A Learning-Based Framework for Depth Perception using Dense Light Fields. In: BRAZILIAN SYMPOSIUM ON MULTIMEDIA AND THE WEB, 2022. **Proceedings...** [S.l.: s.n.], 2022. p.150–158.

FFmpeg Developers. **FFmpeg A complete, cross-platform solution to record, convert and stream audio and video**. Disponível em: <https://ffmpeg.org/>.

FUKUSHIMA, K. Neocognitron: A hierarchical neural network capable of visual pattern recognition. **Neural networks**, [S.l.], v.1, n.2, p.119–130, 1988.

GERSHUN, A. The light field. **Journal of Mathematics and Physics**, [S.l.], v.18, n.1-4, p.51–151, 1939.

GHANBARI, M. **Standard Codecs**: Image Compression to Advanced Video Coding. United Kingdon: The Institution of Electrical Engineers, 2003.

GOU, J.; YU, B.; MAYBANK, S. J.; TAO, D. Knowledge distillation: A survey. **International Journal of Computer Vision**, [S.l.], v.129, n.6, p.1789–1819, 2021.

GRELLERT, M. **Computational effort analysis and control in High Efficiency Video Coding**. 2014. Dissertação — Universidade Federal do Rio Grande do Sul, Porto Alegre.

GU, S.; MENG, W.; SUN, G. Streamlining YOLOv7 for Rapid and Accurate Detection of Rapeseed Varieties on Embedded Device. **Sensors (Basel, Switzerland)**, [S.l.], v.24, n.17, p.5585, 2024.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION, 2015. **Proceedings...** [S.l.: s.n.], 2015. p.1026–1034.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. In: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 2016. **Proceedings...** [S.l.: s.n.], 2016. p.770–778.

HE, Y.; ZHANG, X.; SUN, J. Channel pruning for accelerating very deep neural networks. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION, 2017. **Proceedings...** [S.l.: s.n.], 2017. p.1389–1397.

HELIN, P.; ASTOLA, P.; RAO, B.; TABUS, I. Sparse modelling and predictive coding of subaperture images for lossless plenoptic image compression. In: DTV-CONFERENCE: THE TRUE VISION - CAPTURE, TRANSMISSION AND DISPLAY OF 3D VIDEO (3DTV-CON), 2016., 2016. **Anais...** [S.l.: s.n.], 2016. p.1–4.

HONAUER, K.; JOHANNSEN, O.; KONDERMANN, D.; GOLDLUECKE, B. A dataset and evaluation methodology for depth estimation on 4d light fields. In: ASIAN CONFERENCE ON COMPUTER VISION, 2016. **Anais...** [S.l.: s.n.], 2016. p.19–34.

HORN, B. K.; BRIAN, G. R. Determining optical flow Artificial Intelligence Laboratory. **Massachusetts Institute of Technology, Cam bridge, MA**, [S.l.], v.2139, 1980.

HUANG, Z. et al. Rife: Real-time intermediate flow estimation for video frame interpolation. **arXiv preprint arXiv:2011.06294**, [S.l.], 2020.

IHRKE, I.; RESTREPO, J.; MIGNARD-DEBISE, L. Principles of light field imaging: Briefly revisiting 25 years of research. **IEEE Signal Processing Magazine**, [S.l.], v.33, n.5, p.59–69, 2016.

JASTRZĘBSKI, S. et al. Residual connections encourage iterative inference. **arXiv preprint arXiv:1710.04773**, [S.l.], 2017.

JIN, J.; HOU, J. Occlusion-aware unsupervised learning of depth from 4-d light fields. **IEEE Transactions on Image Processing**, [S.l.], v.31, p.2216–2228, 2022.

KINGMA, D. P.; BA, J. Adam: A Method for Stochastic Optimization. **arXiv preprint arXiv:1412.6980**, [S.l.], 2014.

KONAR, J.; KHANDELWAL, P.; TRIPATHI, R. Comparison of various learning rate scheduling techniques on convolutional neural network. In: IEEE INTERNATIONAL STUDENTS' CONFERENCE ON ELECTRICAL, ELECTRONICS AND COMPUTER SCIENCE (SCEECS), 2020., 2020. **Anais...** [S.l.: s.n.], 2020. p.1–5.

KUGLIN, C.; HINES, D. **The phase correlation image alignment method. InProceedings of the IEEE International Conference on Cyberneticsand Society**. [S.l.]: Piscataway, New Jersey: IEEE Press, 1975.

KULKARNI, U. et al. Performance improvements in quantization aware training and appreciation of low precision computation in deep learning. In: ADVANCES IN SIGNAL

PROCESSING AND INTELLIGENT RECOGNITION SYSTEMS: 6TH INTERNATIONAL SYMPOSIUM, SIRS 2020, CHENNAI, INDIA, OCTOBER 14–17, 2020, REVISED SELECTED PAPERS 6, 2021. **Anais...** [S.l.: s.n.], 2021. p.90–107.

LAUDE, T. et al. A comprehensive video codec comparison. **APSIPA Transactions on Signal and Information Processing**, [S.l.], v.8, p.e30, 2019.

LECUN, Y. A.; BOTTOU, L.; ORR, G. B.; MÜLLER, K.-R. Efficient backprop. In: **Neural networks**: Tricks of the trade. [S.l.]: Springer, 2012. p.9–48.

LECUN, Y.; BENGIO, Y. et al. Convolutional networks for images, speech, and time series. **The handbook of brain theory and neural networks**, [S.l.], v.3361, n.10, p.1995, 1995.

LECUN, Y. et al. Backpropagation applied to handwritten zip code recognition. **Neural computation**, [S.l.], v.1, n.4, p.541–551, 1989.

LECUN, Y. et al. Generalization and network design strategies. **Connectionism in perspective**, [S.l.], v.19, p.143–155, 1989.

LEVOY, M.; HANRAHAN, P. Light field rendering. In: COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, 23., 1996. **Proceedings...** [S.l.: s.n.], 1996. p.31–42.

LI, L.; JIN, X.; ZHONG, T. Imaging-Correlated Intra Prediction for Plenoptic 2.0 Video Coding. In: IEEE INTERNATIONAL CONFERENCE ON MULTIMEDIA AND EXPO (ICME), 2020., 2020. **Anais...** [S.l.: s.n.], 2020. p.1–6.

LI, Y.; WANG, Q.; ZHANG, L.; LAFRUIT, G. A lightweight depth estimation network for wide-baseline light fields. **IEEE Transactions on Image Processing**, [S.l.], v.30, p.2288–2300, 2021.

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, 2021.

LIANG, Z. et al. Weakly-Supervised Salient Object Detection on Light Fields. **IEEE Transactions on Image Processing**, [S.l.], v.31, p.6295–6305, 2022.

LIAO, Z.; QUÉTU, V.; NGUYEN, V.-T.; TARTAGLIONE, E. Can Unstructured Pruning Reduce the Depth in Deep Neural Networks? In: IEEE/CVF INTERNATIONAL CONFERENCE ON COMPUTER VISION, 2023. **Proceedings...** [S.l.: s.n.], 2023. p.1402–1406.

LU, G.; OUYANG, W.; XU, D.; ZHANG, X. Deep Kalman Filtering Network for Video Compression Artifact Reduction. **IEEE Transactions on Image Processing**, [S.l.], v.28, n.4, p.1773–1785, 2019.

LUXMAN, R. et al. LightBot: a multi-light position robotic acquisition system for adaptive capturing of cultural heritage surfaces. **Journal of Imaging**, [S.l.], v.8, n.5, p.134, 2022.

MACHADO, . D. **Software para o Treinamento de Preditores de Light Fields - STPLF**. Acesso em: 02 fev. 2025. Disponível em: <https://github.com/italoplk/predictorLF>.

MARTÍNEZ-RACH, M. O. et al. Performance Overview of the Latest Video Coding Proposals: HEVC, JEM and VVC. **Journal of Imaging**, [S.l.], v.7, n.2, p.39, 2021.

MCMILLAN, L.; BISHOP, G. Plenoptic modeling: An image-based rendering system. In: COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, 22., 1995. **Proceedings...** [S.l.: s.n.], 1995. p.39–46.

MENTZER, F. et al. Neural Video Compression using Spatiotemporal Priors. **IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)**, [S.l.], 2022.

MICHALSKI, R. S.; CARBONELL, J. G.; MITCHELL, T. M. **Machine learning**: An artificial intelligence approach. [S.l.]: Springer Science & Business Media, 2013.

MILDENHALL, B. et al. Nerf: Representing scenes as neural radiance fields for view synthesis. **Communications of the ACM**, [S.l.], v.65, n.1, p.99–106, 2021.

MIRJALILI, S.; MIRJALILI, S. Genetic algorithm. **Evolutionary algorithms and neural networks: theory and applications**, [S.l.], p.43–55, 2019.

MONTEIRO, R. J.; NUNES, P. J.; RODRIGUES, N. M.; FARIA, S. M. Light field image coding using high-order intrablock prediction. **IEEE Journal of Selected Topics in Signal Processing**, [S.l.], v.11, n.7, p.1120–1131, 2017.

MUKATI, M. U. et al. View synthesis-based distributed light field compression. In: IEEE INTERNATIONAL CONFERENCE ON MULTIMEDIA & EXPO WORKSHOPS (ICMEW), 2020., 2020. **Anais...** [S.l.: s.n.], 2020. p.1–6.

MUKATI, M. U. et al. Improved deep distributed light field coding. **IEEE Open Journal of Circuits and Systems**, [S.l.], v.2, p.325–337, 2021.

NAMAN, A.; MATHEW, R.; RUEFENACHT, D.; TAUBMAN, D. UNSW Depth Reciprocal Fields for the HDCA Dataset. In: **ISO/IEC JTC 1/SC29/WG1 JPEG, Doc. N78000**. [S.l.: s.n.], 2017.

NAVARRO, J.; SABATER, N. Learning occlusion-aware view synthesis for light fields. **Pattern Analysis and Applications**, [S.l.], v.24, n.3, p.1319–1334, 2021.

NIKLAUS, S.; MAI, L.; LIU, F. Video frame interpolation via adaptive separable convolution. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION, 2017. **Proceedings...** [S.l.: s.n.], 2017. p.261–270.

OH, K.-S.; JUNG, K. GPU implementation of neural networks. **Pattern Recognition**, [S.l.], v.37, n.6, p.1311–1314, 2004.

O'SHEA, K.; NASH, R. An introduction to convolutional neural networks. **arXiv preprint arXiv:1511.08458**, [S.l.], 2015.

PADIERNA, L. C. et al. A novel formulation of orthogonal polynomial kernel functions for SVM classifiers: the Gegenbauer family. **Pattern Recognition**, [S.l.], v.84, p.211–225, 2018.

PENNEBAKER, W. B.; MITCHELL, J. L. **JPEG Still Image Data Compression Standard**. 1st.ed. Norwell, MA, USA: Kluwer Academic Publishers, 1992.

PEREIRA, F. et al. JPEG Pleno light field coding common test conditions v3. 2. **Doc. ISO/IEC JTC**, [S.l.], v.1, 2019.

PFAFF, J. et al. Intra prediction and mode coding in VVC. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.31, n.10, p.3834–3847, 2021.

PINHEIRO, Á. F. **Inteligência Artificial Fundamentos e Aplicabilidades**. [S.l.: s.n.], 2019.

RAYTRIX. **Raytrix Power Tools**. Disponível em: <https://raytrix.de/>. Acesso em: 2022-18-05.

RERABEK, M.; EBRAHIMI, T. New light field image dataset. In: INTERNATIONAL CONFERENCE ON QUALITY OF MULTIMEDIA EXPERIENCE (QOMEX), 8., 2016. **Anais...** [S.l.: s.n.], 2016. n.CONF.

RICHARDSON, I. **Video Codec Design**: Developing Image and Video Compression Systems. Chichester: John Wiley and Sons, 2002.

RIZKALLAH, M.; MAUGEY, T.; GUILLEMOT, C. Graph-based spatio-angular prediction for quasi-lossless compression of light fields. In: DATA COMPRESSION CONFERENCE (DCC), 2019., 2019. **Anais...** [S.l.: s.n.], 2019. p.379–388.

RODRIGO C. DAUDT, C. G. **Alliance for Open Media - An Alliance of Global Media Innovators**. Disponível em: <http://www.irisa.fr/temics/demos/IllumDatasetLF/index.html> Acesso em: 30/01/2024.

RONNEBERGER, O.; FISCHER, P.; BROX, T. U-net: Convolutional networks for biomedical image segmentation. In: INTERNATIONAL CONFERENCE ON MEDICAL IMAGE COMPUTING AND COMPUTER-ASSISTED INTERVENTION, 2015. **Anais...** [S.l.: s.n.], 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.

RUCK, D. W.; ROGERS, S. K.; KABRISKY, M. Feature selection using a multilayer perceptron. **Journal of Neural Network Computing**, [S.l.], v.2, n.2, p.40–48, 1990.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. **Learning internal representations by error propagation**. [S.l.]: California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

SALMISTRARO, M.; ASCENSO, J.; BRITES, C.; FORCHHAMMER, S. A robust fusion method for multiview distributed video coding. **EURASIP Journal on Advances in Signal Processing**, [S.l.], v.2014, n.1, p.1–16, 2014.

SCHELKENS, P. et al. JPEG Pleno light field coding technologies. In: APPLICATIONS OF DIGITAL IMAGE PROCESSING XLII, 2019. **Anais...** [S.l.: s.n.], 2019. v.11137, p.391–401.

SHI, J.; GUILLEMOT, C. Light field compression via compact neural scene representation. In: ICASSP 2023-2023 IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING (ICASSP), 2023. **Anais...** [S.l.: s.n.], 2023. p.1–5.

SHI, J.; JIANG, X.; GUILLEMOT, C. A framework for learning depth from a flexible subset of dense and sparse light field views. **IEEE Transactions on Image Processing**, [S.l.], v.28, n.12, p.5867–5880, 2019.

SHI, W. et al. Is the deconvolution layer the same as a convolutional layer? **arXiv preprint arXiv:1609.07009**, [S.l.], 2016.

SIEBERT, C. R. **Otimizando Parâmetros de uma DenseNet**: através do controle de geração de mapas de características. [S.l.]: Editora Dialética, 2022.

SITZMANN, V. et al. Implicit neural representations with periodic activation functions. **Advances in Neural Information Processing Systems**, [S.l.], v.33, p.7462–7473, 2020.

SKODRAS, A.; CHRISTOPOULOS, C.; EBRAHIMI, T. The jpeg 2000 still image compression standard. **IEEE Signal processing magazine**, [S.l.], v.18, n.5, p.36–58, 2001.

SPADARO, G. et al. A Learnable EVC Intra Predictor Using Masked Convolutions. In: INTERNATIONAL CONFERENCE ON IMAGE ANALYSIS AND PROCESSING, 2023. **Anais...** [S.l.: s.n.], 2023. p.537–549.

SRIVASTAVA, R. K.; GREFF, K.; SCHMIDHUBER, J. Highway Networks. **CoRR**, [S.l.], v.abs/1505.00387, 2015.

SULLLIVAN, G. J. et al. 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.882 – 885, may 2012.

SZELISKI, R. **Computer vision**: algorithms and applications. [S.l.]: Springer Science & Business Media, 2010.

TANCIK, M. et al. Fourier features let networks learn high frequency functions in low dimensional domains. **Advances in Neural Information Processing Systems**, [S.l.], v.33, p.7537–7547, 2020.

TARG, S.; ALMEIDA, D.; LYMAN, K. Resnet in resnet: Generalizing residual architectures. **arXiv preprint arXiv:1603.08029**, [S.l.], 2016.

TAULLI, T. **Introdução à inteligência artificial**: uma abordagem não técnica. [S.l.]: Novatec Editora, 2020.

THEIS, L.; SHI, W.; CUNNINGHAM, A.; HUSZáR, F. Lossy image compression with compressive autoencoders. **arXiv preprint arXiv:1703.00395**, [S.l.], 2017.

VAN LAARHOVEN, P. J.; AARTS, E. H.; LAARHOVEN, P. J. van; AARTS, E. H. **Simulated annealing**. [S.l.]: Springer, 1987.

VARODAYAN, D.; AARON, A.; GIROD, B. Rate-adaptive codes for distributed source coding. **Signal processing**, [S.l.], v.86, n.11, p.3123–3130, 2006.

VIITANEN, M. et al. Complexity analysis of next-generation HEVC decoder. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS (ISCAS), 2012., 2012. **Anais...** [S.l.: s.n.], 2012. p.882–885.

WANG, L. et al. Unsupervised degradation representation learning for blind super-resolution. In: IEEE/CVF CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 2021. **Proceedings...** [S.l.: s.n.], 2021. p.10581–10590.

WANG, T. et al. Deep learning for light field saliency detection. In: IEEE/CVF INTERNATIONAL CONFERENCE ON COMPUTER VISION, 2019. **Proceedings...** [S.l.: s.n.], 2019. p.8838–8848.

WARNER, B.; MISRA, M. Understanding neural networks as statistical tools. **The american statistician**, [S.l.], v.50, n.4, p.284–293, 1996.

WILBURN, B. et al. High performance imaging using large camera arrays. In: **ACM SIGGRAPH 2005 Papers**. [S.l.: s.n.], 2005. p.765–776.

WU, G. et al. Light field reconstruction using deep convolutional network on EPI. In: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 2017. **Proceedings...** [S.l.: s.n.], 2017. p.6319–6327.

WU, H. et al. Integer quantization for deep learning inference: Principles and empirical evaluation. **arXiv preprint arXiv:2004.09602**, [S.l.], 2020.

X. JIN, M. T. Summary of the Evidences on Lenslet Video Coding. **JVET Meeting**, [S.l.], 2023.

ZHANG, J. et al. Light field saliency detection with deep convolutional networks. **IEEE Transactions on Image Processing**, [S.l.], v.29, p.4421–4434, 2020.

ZHANG, M. et al. LFNet: Light field fusion network for salient object detection. **IEEE Transactions on Image Processing**, [S.l.], v.29, p.6276–6287, 2020.

ZHANG, W.; HASEGAWA, A.; ITOH, K.; ICHIOKA, Y. Image processing of human corneal endothelium based on a learning network. **Applied Optics**, [S.l.], v.30, n.29, p.4211–4217, 1991.

ZHAO, J. et al. Light Field Image Sparse Coding via CNN-Based EPI Super-Resolution. In: IEEE VISUAL COMMUNICATIONS AND IMAGE PROCESSING (VCIP), 2018., 2018. **Anais...** [S.l.: s.n.], 2018. p.1–4.

ZHAO, S.; CHEN, Z.; YANG, K.; HUANG, H. Light field image coding with hybrid scan order. In: VISUAL COMMUNICATIONS AND IMAGE PROCESSING (VCIP), 2016., 2016. **Anais...** [S.l.: s.n.], 2016. p.1–4.

ZHONG, T.; JIN, X.; LI, L.; DAI, Q. Light field image compression using depth-based CNN in intra prediction. In: ICASSP 2019-2019 IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING (ICASSP), 2019. **Anais...** [S.l.: s.n.], 2019. p.8564–8567.

ZHOU, T. et al. Stereo magnification: Learning view synthesis using multiplane images. **arXiv preprint arXiv:1805.09817**, [S.l.], 2018.

ZHOU, Y.; MOOSAVI-DEZFOOLI, S.-M.; CHEUNG, N.-M.; FROSSARD, P. Adaptive quantization for deep neural network. In: AAAI CONFERENCE ON ARTIFICIAL INTELLIGENCE, 2018. **Proceedings...** [S.l.: s.n.], 2018. v.32, n.1.

8 ASSINATURAS

Italo Dombrowski Machado
Proponente

Bruno Zatt
Prof. Orientador