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

Dissertação

Predição de Light Fields Utilizando Redes Neurais Convolucionais

Matheus da Silva Jahnke

Pelotas, 2025

Matheus da Silva Jahnke

Predição de Light Fields Utilizando Redes Neurais Convolucionais

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

Orientador: Prof. Dr. Bruno Zatt

Pelotas, 2025

Universidade Federal de Pelotas / Sistema de Bibliotecas
Catalogação da Publicação

J25p Jahnke, Matheus da Silva

Predição de Light Fields utilizando Redes Neurais Convolucionais [recurso eletrônico] / Matheus da Silva Jahnke ; Bruno Zatt, orientador. — Pelotas, 2025.
48 f. : il.

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

1. Redes Neurais Convolucionais. 2. Light Fields. 3. Compressão. I. Zatt, Bruno, orient. II. Título.

CDD 005

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

RESUMO

JAHNKE, Matheus da Silva. **Predição de Light Fields Utilizando Redes Neurais Convolucionais**. Orientador: Bruno Zatt. 2025. 48 f. Dissertação (Mestrado em Ciência da Computação) – Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2025.

Imagens digitais são extremamente comuns e usadas em inúmeras aplicações atuais. No entanto, sua estrutura bidimensional é insuficiente para algumas aplicações como, por exemplo, representação de ambientes imersivos, mudança de foco e iluminação pós-captura, entre outras. Buscando tratar essa limitação, estuda-se outras representações como nuvens de pontos, imagens tridimensionais, imagens omnidirecionais e *light fields* (LFs) – este sendo o foco deste trabalho.

Light Fields são, essencialmente, imagens quadrimensionais. Dado um conjunto de pontos em um plano – referido como plano do sensor – LFs descrevem, para cada direção de incidência, uma intensidade diferente. Isso pode ser interpretado de várias formas, uma delas é como uma matriz bidimensional onde cada elemento é uma imagem bidimensional ou, nesse contexto, uma *sub-aperture image* (SAI). Dentro das SAIs, variamos as *dimensões espaciais* de um LF, enquanto que, ao variar as *dimensões angulares*, escolhemos uma SAI ou outra.

Como eles são estruturas quadrimensionais, o volume de dados necessário para armazená-los e transmiti-los exige uma compressão eficiente. Várias propostas foram feitas, em geral podendo ser classificadas em dois grandes grupos: aquelas que organizam as SAIs em uma sequência *pseudo temporal* – que então é comprimida utilizando algum codificador de vídeo, tal como o *High Efficiency Video Coding* (HEVC) – tratando uma matriz de SAIs como uma sequência de SAIs; e aquelas que exploram diretamente a natureza quadridimensional de um LF, tal como o *JPEG-Pleno*.

Umas das técnicas utilizadas para melhorar o desempenho da compressão é a predição. Ou seja, a partir de dados que foram (de)codificados anteriormente, tenta-se prever os próximos dados. Quanto melhor for a predição, menores serão os **resíduos** – a diferença entre o que foi predito com o que realmente é – deverão ser codificados e maior será a eficiência de codificação. O *JPEG-Pleno* não utiliza predição intra-quadro, mostrando que existe uma área a ser investigada.

Redes neurais já foram utilizadas com sucesso para uma miríade de problemas, inclusive no processamento de sinais visuais: identificação de números escritos, detecção e identificação de objetos, reconhecimento facial, entre outros. Em particular, foram desenvolvidas com elas abordagens de predição intra-quadro para compressão de imagens e vídeos. Com isso em mente, este trabalho propôs a criação de modelos de predição intra-LF utilizando redes neurais convolucionais. Obteve-se ganho de

Palavras-chave: Redes Neurais Convolucionais. Light Fields. Compressão.

ABSTRACT

JAHNKE, Matheus da Silva. **Light field prediction using neural networks**. Advisor: Bruno Zatt. 2025. 48 f. Dissertation (Masters in Computer Science) – PPGC, Federal University of Pelotas, Pelotas, 2025.

Digital images are extremely commons and used in myriad applications. However, their bidimensional structure is insufficient for some applications as, for instance, representing immersive environments, post-capture refocus and illumination. Trying to solve this limitation, other representations like point clouds, 3d and omnidirectional images and *light fields*(LFs) – the last being the focus of this work.

Light fields are, essentially, quadridimensional images. Given a set de points in a plane – referred as the sensor plane – LFs describe, for every incidence direction, a possibly different intensity. This can be interpreted in many ways, one of them being a bidimensional matrix where every entry is a bidimensional image, or, in this context, a *sub-aperture image*(SAI).

Given they are quadridimensional structures, the volume of data involved to store and transmit them require a efficient compression. Various proposals were made, in a broader view, they can be classified in two big groups: those who organize the SAIs in a *pseudo temporal* sequence – which is then compressed using some video encoder, for instance, the High Efficiency Video Coding(HEVC) – and those which directly explore the quadrimentional nature of LFs.

Neural networks were already used with success to a myriad of problems, including the processing of visual signals: identification of written numbers, detection and identification of objects, facial recognition, amongst others. In particular, intra-frame prediction for compression of images and video approaches were developed using neural networks. With that in mind, this work proposes the creation of intra-LF prediction models using convolutional neural networks.

Keywords: Convolutional Neural Networks. Light Fields. Compressão.

LISTA DE FIGURAS

1 Representação de um raio que passa por (u, v) e por (s, t), figura retirada de WU et al. (2017) 14
2 Um exemplo de uma das *sub-aperture images*. Esta SAI foi retirada do LF “Fountain & Vincent 2” 15
3 Um exemplo de do LF “Fountain & Vincent 2” representado em uma *super multi-view* 16
4 Um recorte do LF “Fountain & Vincent 2” representado em visão de mosca 17
5 Uma *Epipolar Plane Image* retirada do LF *Foutain and Vincent* . . . 18
6 Estrutura de predição utilizada em KHOURY; POURAZAD; NASIOPOULOS (2019), imagem adaptada do artigo 19
7 Imagem adaptada de ASTOLA; TABUS (2018) mostrando os níveis de codificação propostos 20
8 Um exemplo de um *kernel trick* tornando um problema linearmente separável. Figura retirada de ZHANG (2018) 23
9 *LeNet* proposta para o reconhecimento de caracteres LECUN et al. (1989) 26
10 Esquema que descreve uma *U-Net*. Adaptado de RONNEBERGER; FISCHER; BROX (2015) 26
11 Arquitetura utilizada em LI et al. (2017). Adaptada de LI et al. (2017) 29
12 Figura mostrando o processo de treinamento/inferência 32
13 Camadas convolucionais das arquiteturas propostas 34

LISTA DE TABELAS

1 Resultados dos trabalhos relacionados 30

2 Resultados da codificação sem *oracle* 36
3 Resultados da codificação com *oracle* 37
4 Resultados de BD-PSNR e BD-Rate por LF para os preditores propostos dentro do codificador comparado com a configuração utilizando apenas os preditores convencionais 37
5 Resultados dos custos reportados pelo codificador comparando os preditores convencionais (costEVC) com o baseado em redes neurais(costNN) por tamanho de bloco 38
6 Resultados dos custos reportados pelo codificador comparando os preditores convencionais (costEVC) com o baseado em redes neurais(costNN) por tamanho de bloco, considerando apenas os custos RD quando costNN < costEVC 38
7 Tabela com os resultados do custo RD nas diversas configurações testadas, considerando apenas quando o custo EVC é maior que o custoNN 40

LISTA DE ABREVIATURAS E SIGLAS

EPI	Epipolar Plane Image
EVC	Essential Video Coding
HEVC	High Efficiency Video Coding
LF	Light Field
MI	Micro Image
MPEG	Moving Pictures Expert Group
PSNR	Peak Signal to Noise Ratio
RD	Rate Distortion
RDO	Rate Distortion Optimization
SAI	Sub-Aperture Image
SAO	Simple Adaptive Offset
SMV	Super multi-view

SUMÁRIO

1 INTRODUÇÃO 11

2 FUNDAMENTAÇÃO TEÓRICA 13
2.1 Light Fields 13
2.2 Codificação de Light Fields 18
2.2.1 Abordagens baseadas em codificador de vídeo 18
2.2.2 Abordagens nativas a LF 20
2.3 Aprendizado de máquina e redes neurais 21
2.3.1 Perceptron 22
2.3.2 *Multilayer Perceptron* e retropropagação 23
2.3.3 Redes Neurais Convolucionais, *Res-nets* e *U-Nets* 25

3 TRABALHOS RELACINADOS 28

4 SOLUÇÕES PROPOSTAS 31
4.1 Metodologia 31
4.1.1 Preparando os conjuntos dados 31
4.1.2 Processo de treinamento 32
4.1.3 Processo de teste 33
4.2 Arquiteturas propostas 34

5 RESULTADOS E DISCUSSÕES 36

6 CONCLUSÃO 41

REFERÊNCIAS 42

7 ASSINATURAS 48

1 INTRODUÇÃO

Light Fields (LFs) são uma alternativa para mídias imersivas. Em contraste com imagens simples, que são bidimensionais, eles têm quatro dimensões, isso permite realizar diversas operações como, por exemplo, estimar a profundidade (BOLLES; BAKER; MARIMONT, 1987), mudar de perspectiva e mudar de foco após a captura (FU et al., 2015).

Suas quatro dimensões derivam da função **plenóptica**, que tem sete parâmetros: três dimensões para determinar uma posição no espaço, duas dimensões para determinar uma direção, uma para o tempo e uma para determinar a frequência, ou o comprimento de onda da luz. Fixando o tempo e assumindo que a luz se propaga em linha reta, sem perdas, eliminamos dois graus de liberdade, ao abstrairmos a frequência ao considerarmos os pixeis como a estrutura base (0-dimensional), temos quatro dimensões (WU et al., 2017).

As quatro dimensões, digamos $L(u, v, s, t)$ podem ser entendidas como um reta que passa por 2 pontos em planos paralelos: um ponto (u, v) no plano do **objeto** e um ponto (s, t) no plano do **receptor** (WU et al., 2017).

A sua alta dimensionalidade, apesar de permitir operações poderosas, implica em redundâncias e faz com que ocupe-se muito espaço para armazená-lo. Isso torna essencial a utilização de algoritmos de compressão que possam explorar essas redundâncias de modo a comprimir os LFs de maneira eficiente.

Metódos mais iniciais envolviam tornar o LF numa sequência de imagens, chamada ***pseudo-temporal***, e codificar como se fosse um vídeo, ou seja, utilizando um codificador comum tal como *High Efficiency Video Coding* (HEVC) (SULLIVAN et al., 2012). Essa abordagem pode variar na forma com que os LFs são ordenados, encontrada em HEBER; RANFTL; POCK (2013) ou KHOURY; POURAZAD; NASIOPOULOS (2019).

O padrão JPEG-Pleno (PERRA et al., 2020) define dois modos de codificação: o modo de transformada e o modo preditivo. O modo de transformada adota uma abordagem que explora a natureza quadridimensional dos *Light Fields* (LFs). Nesse contexto, aplica-se a transformada discreta do cosseno, *Discrete Cosine Transform* (DCT) em todas as dimensões do dado. Posteriormente, os coeficientes resultan-

tes são quantizados por meio de uma estrutura denominada *hexadeca-tree*, análoga à *quadtree*, porém estendida para quatro dimensões. Ressalta-se que o modo de transformada não utiliza técnica de predição alguma (PEREIRA et al., 2019). O modo preditivo, por sua vez, realiza a codificação das *Sub-Aperture Images* (SAIs) de forma hierárquica, iniciando pela SAI central, correspondente ao nível 0, e prosseguindo com os níveis subsequentes, os quais utilizam os anteriores como referência. A predição consiste na utilização da SAI mais próxima, ajustada mediante uma técnica de compensação de movimento, conhecida como *warping*. Destaca-se que a abordagem emprega predição em nível de SAI, e não em nível de bloco (ASTOLA; TABUS, 2018).

Um dos estágios frequentemente realizados em codificadores de imagem e vídeo é a **predição intra-quadro**. Nesse estágio, o codificador faz uso de preditores baseados em regiões vizinhas que já foram codificadas, isto é, uma **vizinhança causal**. Somente após essa etapa são aplicados os processos de transformada e quantização, conforme implementado pelo *software* de referência do *JPEG-Pleno*.

Uma das formas mais comuns de melhorar a eficiência de codificação é aprimorar os módulos de predição. Quanto mais parecido for o bloco predito com o bloco original, melhor será a eficiência da codificação. Isso significa que, com uma boa predição, é possível obter uma qualidade maior mantendo a mesma taxa de codificação, ou reduzir o tamanho dos dados – e, assim, diminuir a taxa de *bits* – mantendo a mesma qualidade visual (SULLIVAN et al., 2012).

Para esses fins, um codificador testa vários preditores com diferentes configurações e, em seguida, escolhe aquele que mais se aproxima do bloco que está sendo codificado. Por exemplo, o padrão Versatile Video Coding (VVC) define 71 formas distintas de realizar a predição intra-quadro (PFAFF et al., 2021).

Algoritmos de inteligência artificial, em especial as Redes Neurais (RN) – ainda mais especificamente, as redes neurais convolucionais (RNC) – têm sido utilizados recentemente para implementar preditores **intra-quadro** (CUI et al., 2018) (PFAFF et al., 2018). Essas redes têm se mostrado eficazes na previsão de blocos de imagem, devido à sua capacidade de capturar padrões espaciais complexos.

Diante do que foi discutido nesta seção, observa-se um potencial significativo para a aplicação de redes neurais convolucionais em implementações de preditores intra-LF, contribuindo assim para o avanço da codificação de campos de luz.

O restante desta dissertação está organizado da seguinte forma. O Capítulo 2 introduz conceitos de LFs, abordagens utilizadas para codificá-los e conceitos de RN e RNCs. O Capítulo 3 apresenta outras abordagens que utilizaram redes neurais para a predição. O Capítulo 4 descreve com mais detalhes a abordagem adotada e as soluções propostas. O Capítulo 5 apresenta os resultados e sua análise, e, finalmente, o Capítulo 6 encerra esta monografia.

2 FUNDAMENTAÇÃO TEÓRICA

Relevantes a este trabalho são as noções sobre *LFs*, os diferentes métodos de codificação aplicáveis a essas estruturas, bem como conceitos fundamentais de aprendizado de máquina.

2.1 Light Fields

Light Fields derivam da função plenóptica $P(x, y, z, \theta, \phi, t, \lambda)$, definida em GERSHUN (1939), inicialmente com o objetivo de estudar a natureza da luz, especialmente como ondas no espaço. As três primeiras variáveis, x, y e z, determinam uma posição em um espaço tridimensional; as duas seguintes, θ e ϕ, definem uma direção dentro desse espaço; o penúltimo argumento, t, corresponde ao tempo, e λ é o comprimento de onda, associado [1] à frequência da luz (LEVOY; HANRAHAN, 1996).

Para reduzir a dimensionalidade, fixamos um instante de tempo t para eliminar uma dimensão. Ao tratar a unidade básica, o ponto, como uma entidade de dimensão zero – mesmo quando se trata de imagens coloridas, onde os diferentes canais (como R, G, B) não são interpretados como uma dimensão adicional – eliminamos a dimensão associada à frequência. Isso reduz para 5 dimensões, $P(x, y, z, \theta, \phi)$, uma posição e uma direção dentro de um espaço tridimensional (WU et al., 2017). Assumindo que a luz se propaga em linha reta e que consideramos a luz incidindo em um plano, temos um grau de liberdade a menos. Após essas transformações, o resultado é uma função com quatro parâmetros $L(u, v, s, t)$ (LEVOY; HANRAHAN, 1996).

Uma maneira de interpretar um LF é através de 2 planos, um dos planos, (u, v) é o plano do objeto, cujas dimensões – intituladas *dimensões espaciais* – correspondem às dimensões de uma imagem bidimensional, e um dos planos, (s, t) é o plano das lentes, cujas dimensões – intituladas *dimensões angulares* – descrevem o ponto de vista, tal como está na figura 1.

Quando fixamos as dimensões angulares, obtemos uma estrutura semelhante a uma imagem 2D comum, como uma fotografia. Essa estrutura é chamada de *sub-*

[1] Considerando que a velocidade de propagação é constante

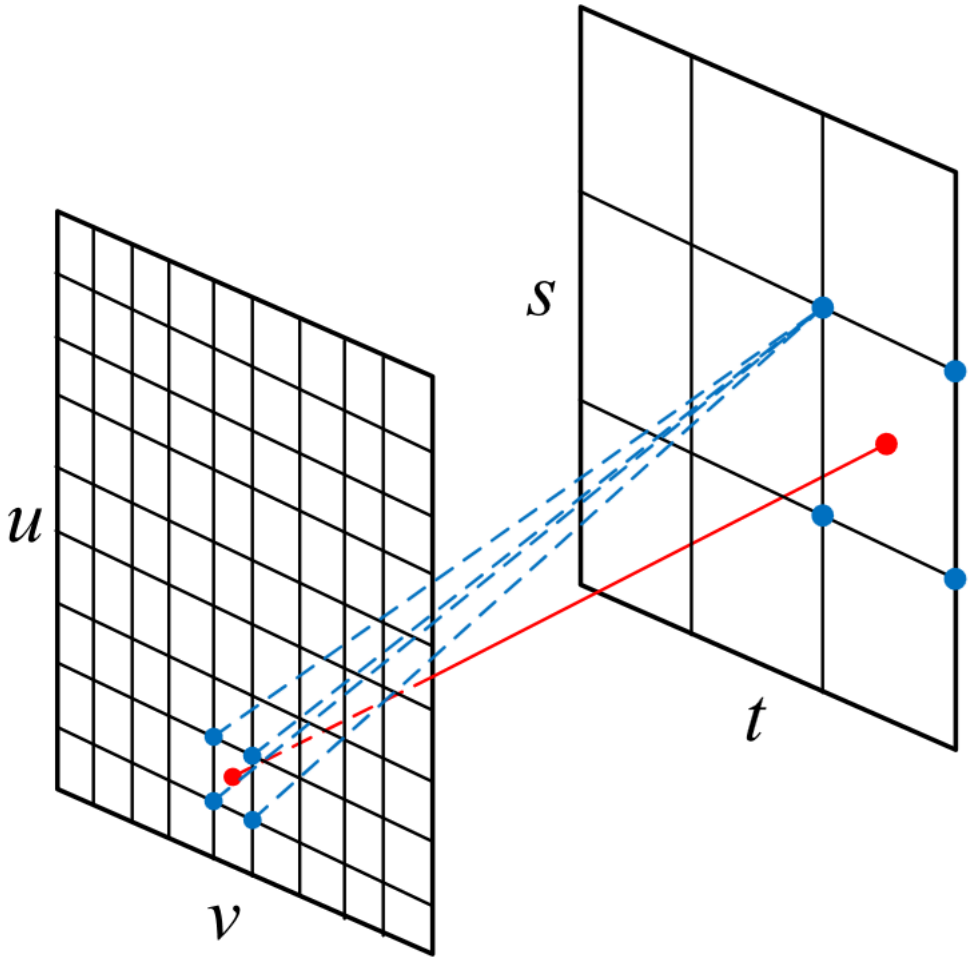


Figura 1 – Representação de um raio que passa por (u, v) e por (s, t), figura retirada de WU et al. (2017)

aperture image (SAI) ou *view*, conforme ilustrado na Figura 2. Ao concatenar diversas SAIs, formamos o *super-multiview* (SMV), que representa o conjunto completo de vistas do campo luminoso, conforme mostrado na Figura 3.

Figura 2 – Um exemplo de uma das *sub-aperture images*. Esta SAI foi retirada do LF "Fountain & Vincent 2"

Fixando as dimensões espaciais, temos uma estrutura, chamada de *micro-imagem* (MI), que mostra o *píxel* correspondente de cada SAI. Concatenando as MIs obtemos uma estrutura referida como **visão de mosca** ou **lenslet**, um recorte de um LF com esta representação pode ser visto na figura 4.

Fixando uma das dimensões espaciais e uma das dimensões angulares, com eixos perpendiculares, como por exemplo (v, s) ou (u, t), obtemos uma estrutura chamada *Epipolar Plane Image* (EPI), ilustrada na Figura 5. Na prática, essa estrutura revela linhas cujas inclinações indicam a profundidade dos objetos: objetos mais próximos correspondem a linhas mais inclinadas, enquanto objetos mais distantes geram linhas mais próximas da vertical. Isso ocorre porque objetos mais próximos sofrem maior deslocamento aparente com variações no ponto de vista, como discutido em BOLLES; BAKER; MARIMONT (1987).

Além da aplicação discutida anteriormente, o *light field* também pode ser considerado uma mídia imersiva. Isso significa que uma mudança no ponto de vista – dentro da faixa capturada pelo LF – corresponde à seleção de uma vista específica ou à

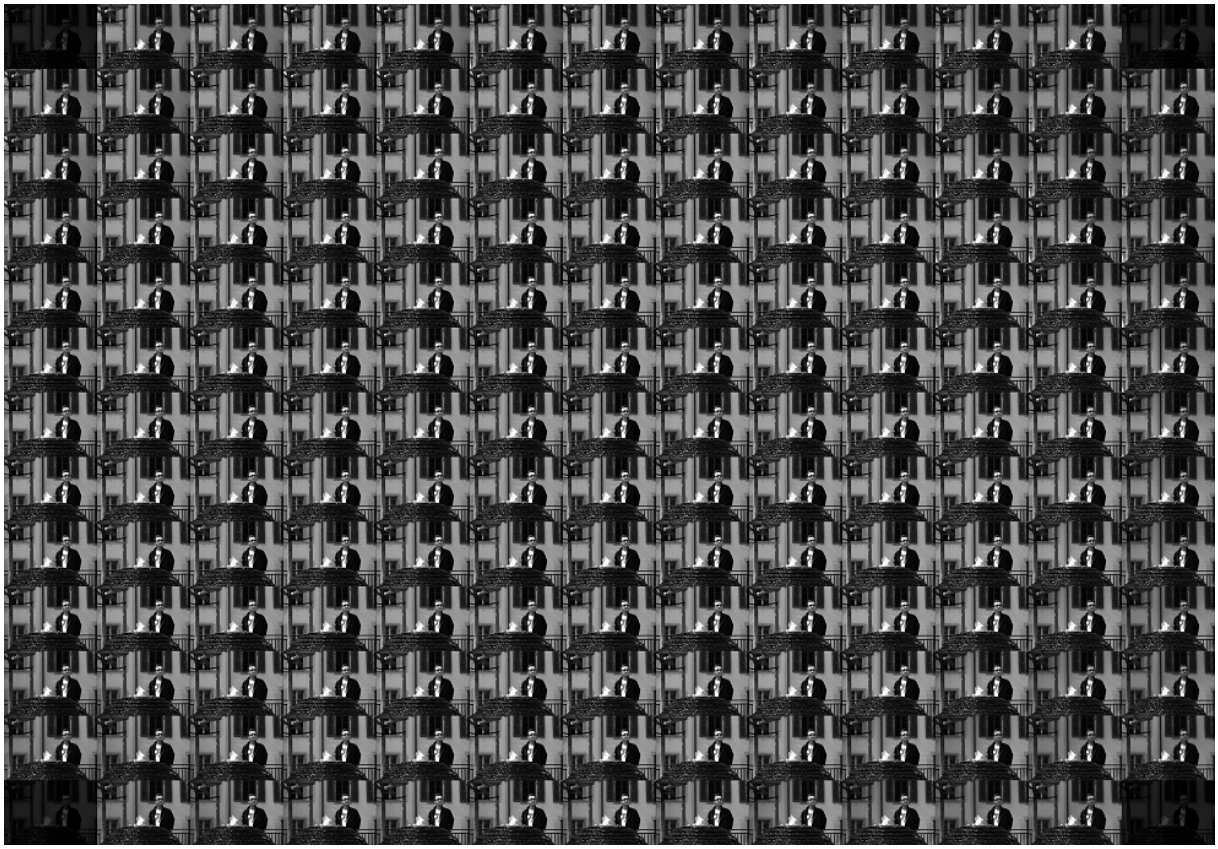


Figura 3 – Um exemplo de do LF “Fountain & Vincent 2” representado em uma *super multi-view*

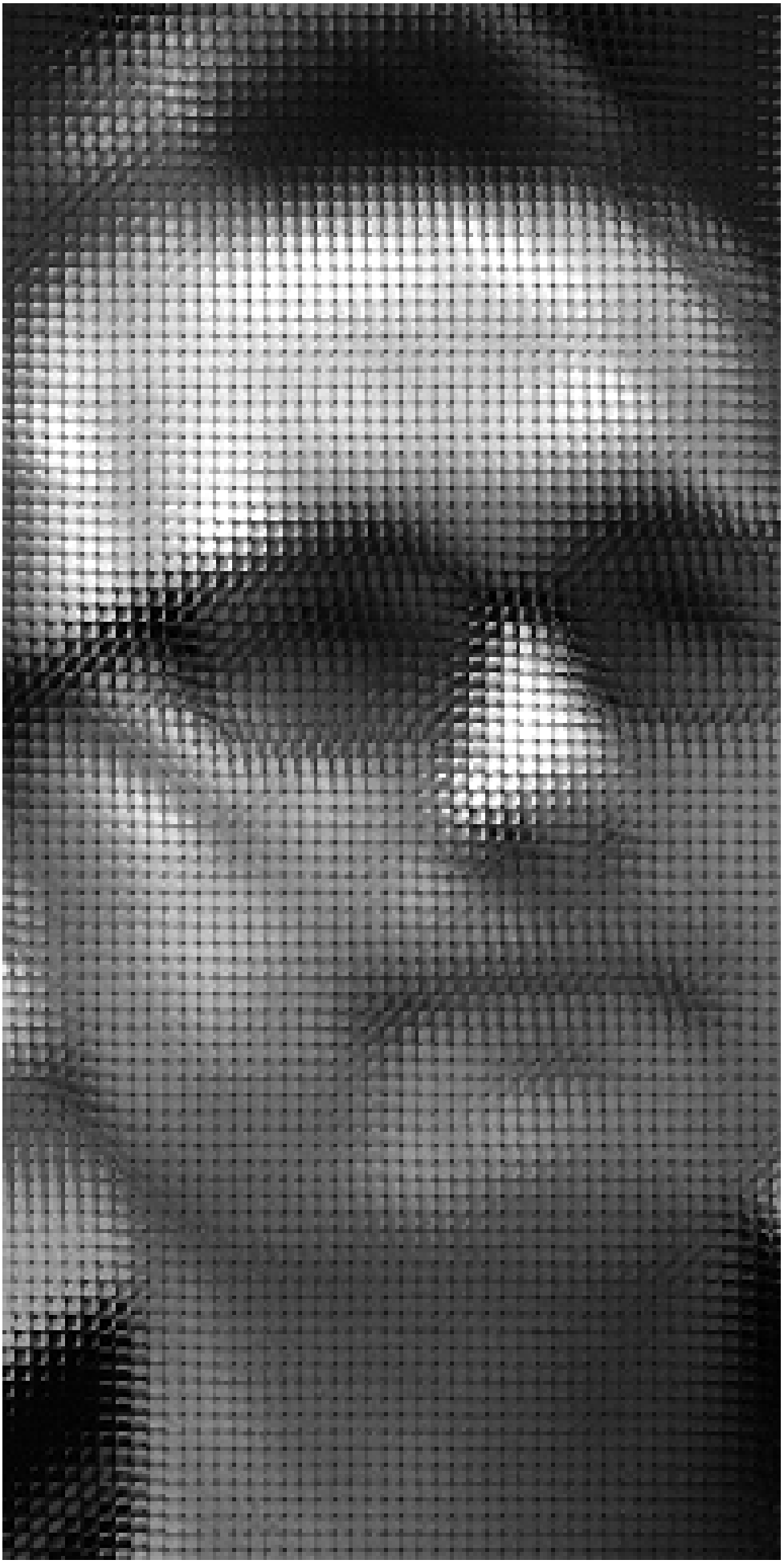


Figura 4 – Um recorte do LF “Fountain & Vincent 2” representado em visão de mosca

Figura 5 – Uma *Epipolar Plane Image* retirada do LF *Foutain and Vincent*

síntese de uma vista intermediária com base nas SAIs mais próximas, como descrito em LEVIN; DURAND (2010). Essa abordagem pode ser utilizada como alternativa à representação em uma nóvem de pontos, conforme definido em GROSSMAN; DALLY (1998).

Outra aplicação relevante dos *light fields* é a possibilidade de alterar a distância focal após a captura. A partir de um LF já adquirido, é possível ajustar o **plano focal** – ou seja, o plano em que a imagem aparece mais nítida. Em fotografias tradicionais, esse ajuste só pode ser feito antes do disparo da câmera. No entanto, como o LF contém informações suficientes para estimar diferentes pontos de foco, é viável simular a captura com diferentes profundidades de campo após o registro da cena (FU et al., 2015).

2.2 Codificação de Light Fields

Existem diferentes abordagens para a codificação de *Light Fields*. Algumas dessas abordagens utilizam codificadores de vídeo convencionais, aplicando-os a uma organização das SAIs em uma sequência linear. Outras são desenvolvidas especificamente para lidar com as características dos LFs, explorando sua estrutura multidimensional de forma mais eficiente.

2.2.1 Abordagens baseadas em codificador de vídeo

As abordagens que reorganizam as SAIs criam uma sequência pseudotemporal – como se fosse um vídeo ao longo do tempo, onde cada SAI é um quadro do pseudo-vídeo – e, em seguida, submetem essa sequência a um codificador de vídeo convencional, como o HEVC (SZE; BUDAGAVI; SULLIVAN, 2014) ou o VVC (BROSS et al., 2021).

Um dos primeiros exemplos é PERRA; ASSUNCAO (2016), que organiza as SAIs em formato serpentina, justamente para manter a similaridade entre as perspectivas, evitando saltos que ocorreriam na ordem de rasterização. Outra abordagem é discutida em CONCEICAO et al. (2018), entre aquelas que utilizam codificadores de vídeo convencionais. Nesse caso, em vez adotar uma ordem pré-definida, inicia-se pela codificação da SAI central e, enquanto houver SAIs a serem codificadas, escolhe-se aquela com maior eficiência de codificação, com base nas referências anteriores.

Outro trabalho em uma linha semelhante é GOMES (2019), no qual são propostas três ordens diferentes para as sequências: *raster*, serpentina e espiral. Além disso, o estudo introduz uma sequencialização arbitrária, com variações que utilizam várias

sequências codificadas de forma independente. Isso permite decodificar uma parte do LF sem precisar decodificar os demais trechos. Essa característica, conhecida como *random access*, é maior quanto menor for a porção necessária para acessar uma região específica do conjunto.

Outro trabalho é KHOURY; POURAZAD; NASIOPOULOS (2019), que propõe uma ordenação em "níveis": primeiro, codifica-se a vista central; em seguida, usando essa vista como referência, são codificadas quatro vistas nas bordas, e assim por diante, conforme ilustrado na Figura 6.

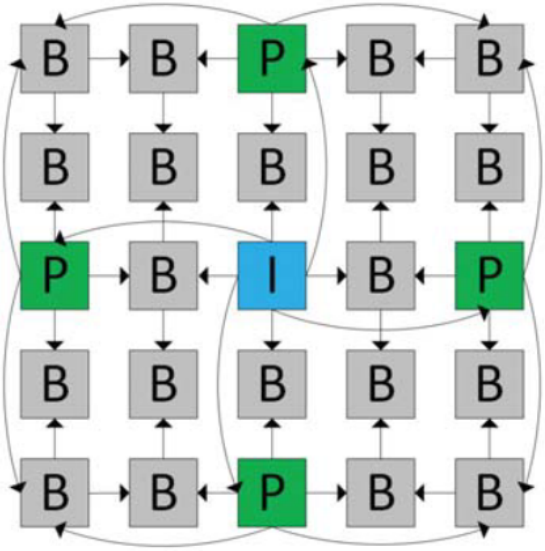


Figura 6 – Estrutura de predição utilizada em KHOURY; POURAZAD; NASIOPOULOS (2019), imagem adaptada do artigo

Uma abordagem que começa a explorar a estrutura de um LF é apresentada em GIROD et al. (2003), que parte do princípio de que as SAIs são imagens dos mesmos objetos, capturadas de pontos de vista ligeiramente diferentes. A técnica proposta analisa essas variações de perspectiva usando *disparity compensation* para predição entre quadros e, em seguida, aplica uma transformada *wavelet*. Essa compensação pelo deslocamento entre pontos de vista é semelhante à compensação de movimento usada em codificadores de vídeo.

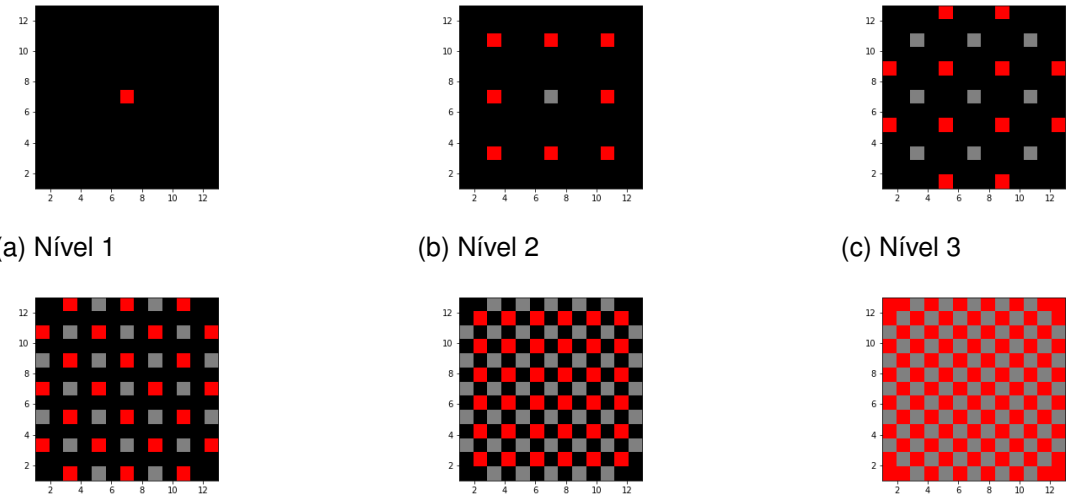
Nessa mesma linha, CHEN; HOU; CHAU (2017) também explora as similaridades entre as SAIs — as *Structural Key Views* — para que sejam codificadas diretamente como se fossem imagens (ou o primeiro *frame* de um vídeo). Ou seja, usa-se apenas predição intra nesse ponto e, a partir dessas imagens-chave, realiza-se a codificação das demais vistas, identificando e corrigindo a disparidade causada pelo deslocamento do ponto de vista, para então codificar os resíduos.

A maioria das abordagens discutidas não aproveita por completo a estrutura 4D dos LFs. Elas exploram a similaridade entre as SAIs, mas precisam linearizá-las para gerar uma sequência compatível com um codificador de vídeo.

2.2.2 Abordagens nativas a LF

Esse grupo de abordagens trata o LF como uma estrutura 4D. Por exemplo, CARVALHO et al. (2018) aplica a **transformada discreta do cosseno** (***discrete cosine transform***, DCT) em quatro dimensões. Em seguida, realiza quantização usando uma *hexadecatree*, que divide os blocos ao longo das quatro dimensões, e aplica codificação de entropia. Embora não empregue predição, esse método supera esquemas baseados em codificadores de vídeo que não exploram totalmente a estrutura quadridimensional do LF.

O artigo ASTOLA; TABUS (2018) propõe organizar as SAIs em camadas (*levels*). Primeiro, codificam-se as camadas inferiores; em seguida, as camadas superiores são codificadas usando as SAIs já codificadas como referência para registrar apenas as diferenças. Como mostra a Figura 7, aplica-se uma compensação de deslocamento (*warping*) para alinhar as imagens antes de codificar essas diferenças. Esse procedimento corresponde ao *Prediction Mode* do codificador de LFs do JPEG Pleno (SCHELKENS et al., 2019).



(a) Nível 1 (b) Nível 2 (c) Nível 3

(d) Nível 4 (e) Nível 5 (f) Nível 6

Figura 7 – Imagem adaptada de ASTOLA; TABUS (2018) mostrando os níveis de codificação propostos

Para facilitar a comparação entre os resultados, foram estabelecidas condições comuns de teste em PEREIRA et al. (2019). Em particular, define-se a taxa de *bits* o *Bits Per Pixel* (BPP) como o tamanho total do arquivo dividido pelo número de píxeis.

Define-se o *Peak Signal to Noise Ratio* (PSNR) geral como a média aritmética ponderada dos PSNRs dos canais Y, Cb e Cr, de formasendo que o PSNR do canal Y tem peso seis e os demais, peso um. O PSNR pode ser calculado conforme a equação (1), onde MAX representa o maior valor que o sinal pode assumir, neste caso, a maior intensidade de luz que a imagem pode atingir.

Como as imagens do *dataset* usado são representadas com dez *bits* por amostra, esse valor é 1023. O erro médio quadrático, ou *Mean Squared Error* (MSE), é uma métrica calculada entre dois vetores (ou matrizes), conforme a equação 2.

Para comparar dois ou mais codificadores ou configurações em um mesmo conjunto de sequências (LF), é necessário realizar múltiplas codificações para cada um, de modo que a qualidade e a taxa de bits variem, permitindo a obtenção de uma curva amostrada. Em seguida, calculam-se as métricas propostas por BJONTEGAARD (2001), conhecidas como métricas de Bjontegaard, que permitem extrair um único valor representativo da diferença entre as configurações. Essa diferença pode ser avaliada sob duas perspectivas: pela qualidade — "mantendo-se a mesma taxa de bits, qual é a diferença de qualidade?" — ou pela taxa de bits — "mantendo-se a mesma qualidade, qual é a diferença na taxa de bits?". Essas análises são representadas, respectivamente, pelas métricas BD-PSNR e BD-Rate.

$$\texttt{PSNR}(X, Y) = 20 \cdot \log_{10} \left(\frac{MAX}{\sqrt{\texttt{MSE}(X, Y)}} \right) \tag{1}$$

$$\texttt{MSE}(X, Y) = \frac{1}{n} \cdot \sum_{k=1}^{n} (X_k - Y_k)^2 \tag{2}$$

2.3 Aprendizado de máquina e redes neurais

Aprendizado de máquina é um ramo da inteligência artificial que estuda algoritmos capazes de analisar dados e "aprender" a realizar tarefas sem serem explicitamente programados para isso. O termo foi usado pela primeira vez em SAMUEL (1959), que apresentou um algoritmo para o jogo de damas. Esse algoritmo, ao receber uma lista de parâmetros – segundo os próprios autores, incompleta e redundante – aprendia a atribuir um grau de importância a cada um desses parâmetros. A partir deste ponto, esse grau será chamado de **peso** ou **coeficiente**.

Em algumas horas de jogo, junto a um algoritmo de *minimax* – que percorre N movimentos futuros para escolher a melhor opção, usando os pesos ajustados e assumindo que o oponente sempre fará a pior escolha possível para o jogador – o sistema alcançou um desempenho superior ao de um jogador iniciante, inclusive melhor que o das pessoas que o programaram.

Atualmente, o estado-da-arte em aprendizado de máquina são as **redes neurais**, que calculam os parâmetros mencionados anteriormente como somas ponderadas a entrada. Em seguida será discutido o *perceptron* que executa uma única iteração desse método, após serão discutidos os *multilayer perceptron* que realiza esse mesmo processo iterativamente e finalmente serão discutidos as **redes neurais convolucionais** combinam esse método com o conceito de **convolução** de

2.3.1 Perceptron

As redes neurais artificiais são compostas por neurônios artificiais - mais precisamente, blocos básicos inspiradas em neurônios. Cada neurônio artificial possui N entradas ou ***features*** $\vec{x} = [x_1, x_2, \cdots, x_n]$, e N, possivelmente $N + 1$, pesos, $\vec{w} = [w_1, w_2, \cdots, w_n]$, podendo ter ou não um viés b. Com isso, a saída desse neurônio é um valor y, que corresponde a uma combinação linear entre x e w, conforme descrito na equação 3.

$$y = b + \vec{w} \cdot \vec{x} = b + w_1 \cdot x_1 + w_2 \cdot x_2 + \cdots + w_n \cdot x_n \tag{3}$$

Um exemplo de neurônio artificial é o **perceptron**, que calcula y com base em uma entrada $\vec{x}$ e determina se a saída será ativada, caso $y > 0$, ou não será ativada, caso $y < 0$. Isso imita o comportamento de um neurônio biológico, que recebe sinais de outros neurônios e, com base neles, decide se deve ser ativado ou não (BLOCK, 1962). O perceptron pode ser compreendido como um classificador binário.

Um dos avanços no estudo das redes neurais foi o desenvolvimento de métodos para ajustar os pesos. Em BLOCK (1962), foi definida a **regra do perceptron**, que ajusta os pesos sempre que o modelo comete um erro — seja ao indicar que uma entrada pertence a uma classe quando na verdade não pertence (erro do tipo I ou **falso positivo**), seja ao indicar que uma entrada não pertence a uma classe quando na verdade pertence (erro do tipo II ou **falso negativo**).

Sempre que ocorre um erro, a regra do perceptron orienta o ajuste dos pesos somando w_i com η, um número pequeno chamado ***learning rate***, multiplicado por x_i (o valor da entrada associada ao peso a ser ajustado[2]) e por 1, no caso de erro do tipo I, ou -1 no caso de erro do tipo II. A equação 4 descreve como calcular o novo peso w_i', com base na saída esperada y e na saída prevista y_p.

$$w_i' = w_i + \eta \cdot (y - y_p) \cdot x_i \tag{4}$$

O perceptron tem limitações quanto à classe de problemas que ele resolve: apenas problemas de classificação binária e linearmente separáveis (ROSENBLATT, 1961). A adaptação para problemas de regressão será discutida a seguir. Para lidar com problemas não lineares, existem duas abordagens principais. A primeira consiste em aplicar transformações na entrada, de forma que o espaço de *features* se torne linearmente separável – técnica conhecida como *kernel trick* (HALAGUNDEGOWDA, 1980), exemplificada na figura 8. A segunda abordagem utiliza a saída numérica de um neurônio como entrada de outro, formando as chamadas redes ***multilayer perceptron*** (RUMELHART; HINTON; WILLIAMS, 1986), que serão discutidas a seguir. Essa abordagem,

[2]Para o viés, usa-se o valor 1

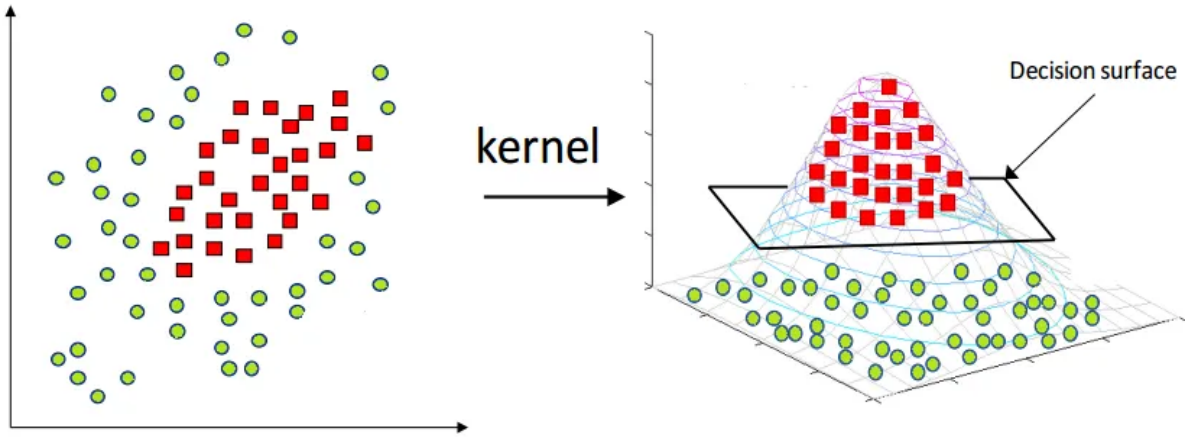


Figura 8 – Um exemplo de um *kernel trick* tornando um problema linearmente separável. Figura retirada de ZHANG (2018)

além de superar limitações do perceptron simples, é a base para arquiteturas mais complexas de redes neurais.

2.3.2 *Multilayer Perceptron* e retropropagação

Um perceptron multicamadas (multilayer perceptron, MLP) (RUMELHART; HINTON; WILLIAMS, 1986) amplia o perceptron simples ao empilhar diversas camadas de neurônios *fully connected*, nas quais as saídas de uma camada servem como entradas para a camada seguinte. Cada camada transforma seu vetor de entrada, ou *features*, por meio de uma combinação linear seguida de uma função de ativação não linear. Isso permite que as *features* passem a representar conceitos cada vez mais abstratos. Essa organização com camadas ocultas sucessivas permite ao modelo capturar relações não lineares complexas nos dados, superando as limitações de separabilidade linear do perceptron original.

Embora o perceptron seja um modelo mais simples de rede neural, ele pode ser entendido como a aplicação de uma combinação linear das entradas, seguida por uma função de ativação do tipo degrau, conforme a equação 5. Em muitos contextos, no entanto, utilizam-se outras funções de ativação, como a **sigmoide** (equação 6) ou a **Retificadora Linear Unidirecional (ReLU)** (equação 7) (SHARMA; SHARMA; ATHAIYA, 2017), que costumam apresentar melhor desempenho em tarefas de aprendizado de máquina.

$$f(x) = \begin{cases} 1, & \text{se } x \geq 0 \\ 0, & \text{caso contrário} \end{cases} \tag{5}$$

$$f(x) = \frac{e^x}{1 + e^x} = \frac{1}{1 + e^{-x}} \tag{6}$$

$$f(x) = \begin{cases} x, & \text{se } x \geq 0 \\ 0, & \text{caso contrário} \end{cases} \tag{7}$$

Além da função de ativação, também se utiliza uma **função de perda** (ou ***loss function***), que mede o quão distante a saída da rede está do valor esperado. Quanto menor essa distância, melhor o desempenho da rede. Mais importante do que o valor da perda em si é sua derivada em relação aos pesos, pois é ela que guia os ajustes nos pesos durante o treinamento (HASTIE et al., 2009).

Para problemas de classificação, uma função de perda comum é a entropia cruzada (ou *log-loss*); para problemas de regressão, um exemplo clássico é o **erro médio quadrático**, que corresponde à média dos quadrados das diferenças entre os valores reais e os previstos, conforme a equação 8, onde n é o tamanho do vetor, $\vec{y}$ é o vetor de referência e $\vec{y}_p$ é o vetor predito pela rede (JANOCHA; CZARNECKI, 2017).

$$\text{MSE}(\vec{y}, \vec{y}_p) = \frac{1}{n} \cdot ||\vec{y} - \vec{y}_p||^2 \tag{8}$$

Boa parte dos algoritmos otimizadores, usados para ajustar os pesos, baseia-se no **método da descida de gradiente**. Essa estratégia procura a direção que mais reduz o valor da função de perda conforme os parâmetros variam. Para isso, utiliza-se a derivada da função em relação aos pesos. Como geralmente há mais de um peso, emprega-se o **gradiente**, que é o vetor das derivadas parciais. A técnica consiste em dar um "passo" na direção que mais reduz o erro, ou seja, no sentido oposto ao gradiente, multiplicado pela **taxa de aprendizado** (LEMARÉCHAL, 2012).

$$\vec{w}_{n+1} = \vec{w}_n - \eta \cdot \vec{\nabla}_w \text{err}(\vec{w}) \tag{9}$$

O comportamento da derivada parcial dentro de uma rede neural permite que ela seja calculada "propagando" os gradientes dos valores finais para os iniciais. Considere uma rede neural com entrada $\vec{x}$, camadas intermediárias $\vec{v}_1$ e $\vec{v}_2$, e saída y. A regra da cadeia permite que, a partir da derivada da função de perda em relação a y, seja possível calcular o gradiente em relação a $\vec{v}_2$, conforme mostrado na equação, onde w_2 representa os pesos da segunda camada e f_2 é a função de ativação aplicada após essa camada (HECHT-NIELSEN, 1992).

$$\vec{\nabla}_{\vec{v}_2} \text{err} = \nabla_{\vec{v}_2}[y] \cdot \frac{\mathrm{d}\ \text{err}}{\mathrm{d}y} = \vec{w}_2 \cdot f_2'(\vec{w}_2 \cdot \vec{v}_2) \frac{\mathrm{d}\ \text{err}}{\mathrm{d}y} \tag{10}$$

O cálculo anterior pode ser generalizado: os gradientes de cada camada podem ser obtidos multiplicando [3] o gradiente da camada seguinte pelos pesos e pela derivada da função de ativação. Esse processo é conhecido como **retropropagação** ou

[3] Possivelmente usando multiplicação escalar por vetor, ou vetor por matriz

backpropagation. Para calcular a derivada de um peso específico, utiliza-se a derivada da saída em relação à variável que depende desse peso, multiplicada pelo valor da entrada correspondente (HECHT-NIELSEN, 1992).

Esse paradigma oferece uma base teórica sólida para a implementação de redes neurais: existem demonstrações de que redes com três camadas *fully connected* podem aproximar qualquer função que seja "comportada o suficiente", desde que haja neurônios em quantidade adequada (HECHT-NIELSEN, 1992). No entanto, há limitações em relação ao tempo de treinamento e à capacidade de generalização em dados com estrutura: por exemplo, em imagens, que são estruturas bidimensionais, uma rede totalmente conectada não diferencia *pixeis* próximos de distantes. Para lidar com essas limitações, podem ser usadas as **redes neurais convolucionais** (RNCs), que serão discutidas a seguir (LI et al., 2021).

2.3.3 Redes Neurais Convolucionais, *Res-nets* e *U-Nets*

Uma operação comum em processamento de sinais é a **convolução**. Ela combina duas sequências, $x[n]$ e $w[m]$, para produzir a sequência $y[n+m-1]$, conforme definida pela equação (11) (BURRUS; PARKS, 1985).

$$y(n) = \sum h(m)w(n-m) \tag{11}$$

Em uma camada de **rede neural convolucional** (RNCs), as entradas $x[k]$ são combinadas com os pesos (ou *kernel*) $w[k]$ por meio da operação de convolução, produzindo as saídas $y[k]$, às quais se aplica uma função de ativação. Essa mesma ideia vale para convoluções em várias dimensões, estendendo o processo de forma análoga.

Uma das principais vantagens dessa abordagem é o compartilhamento de pesos: para calcular qualquer $y[n]$, aplica-se o mesmo kernel em diferentes regiões da entrada. Com isso, um deslocamento na entrada resulta no mesmo deslocamento na saída, tornando o processo extbfinvariante a deslocamentos (*shift invariant*) (LECUN; BENGIO et al., 1995).

Um exemplo clássico de redes neurais convolucionais é a *LeNet*, proposta por LeCun et al. (LECUN et al., 1989) e mostrada na Figura 9. A arquitetura começa com uma camada convolucional seguida de uma operação de *pooling* para reduzir o tamanho dos mapas de ativação. Em seguida, aplicam-se mais duas camadas convolucionais. Por fim, os mapas de ativação são achatados em um vetor e processados por camadas *fully connected*.

Inicialmente, as redes neurais convolucionais (RNCs) foram usadas para reconhecer caracteres (FUKUSHIMA, 1988), classificando dez símbolos. Em imagens médicas, ZHANG et al. (1991) aplicou RNCs para segmentar automaticamente as células

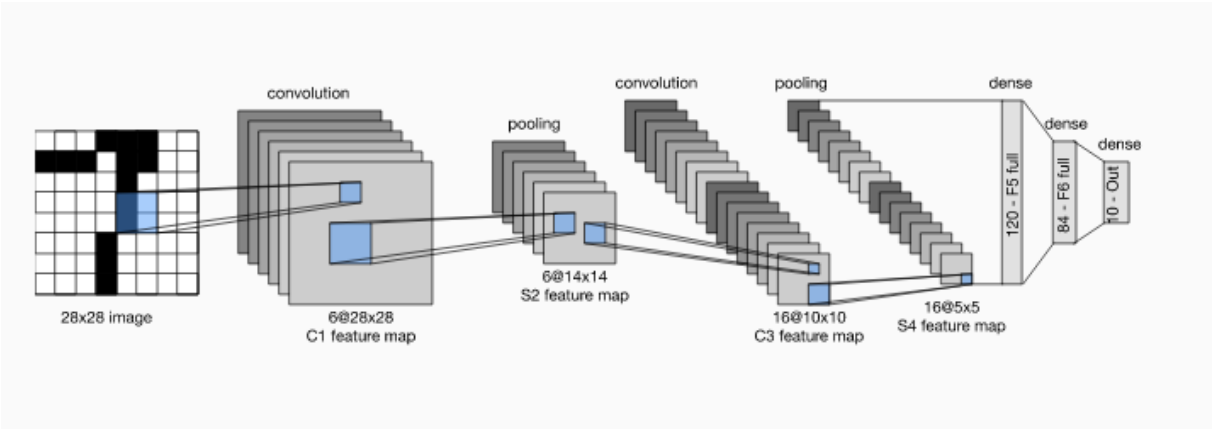


Figura 9 – *LeNet* proposta para o reconhecimento de caracteres LECUN et al. (1989)

de uma camada da córnea — tarefa antes feita manualmente.

Um caso especial de ResNet é a U-Net (RONNEBERGER; FISCHER; BROX, 2015), criada para segmentar imagens médicas. A U-Net tem duas partes: um caminho de contração, com camadas que reduzem a resolução, e um caminho de expansão, com camadas que recuperam essa resolução. Assim, a saída tem o mesmo tamanho da entrada (ver Figura 10).

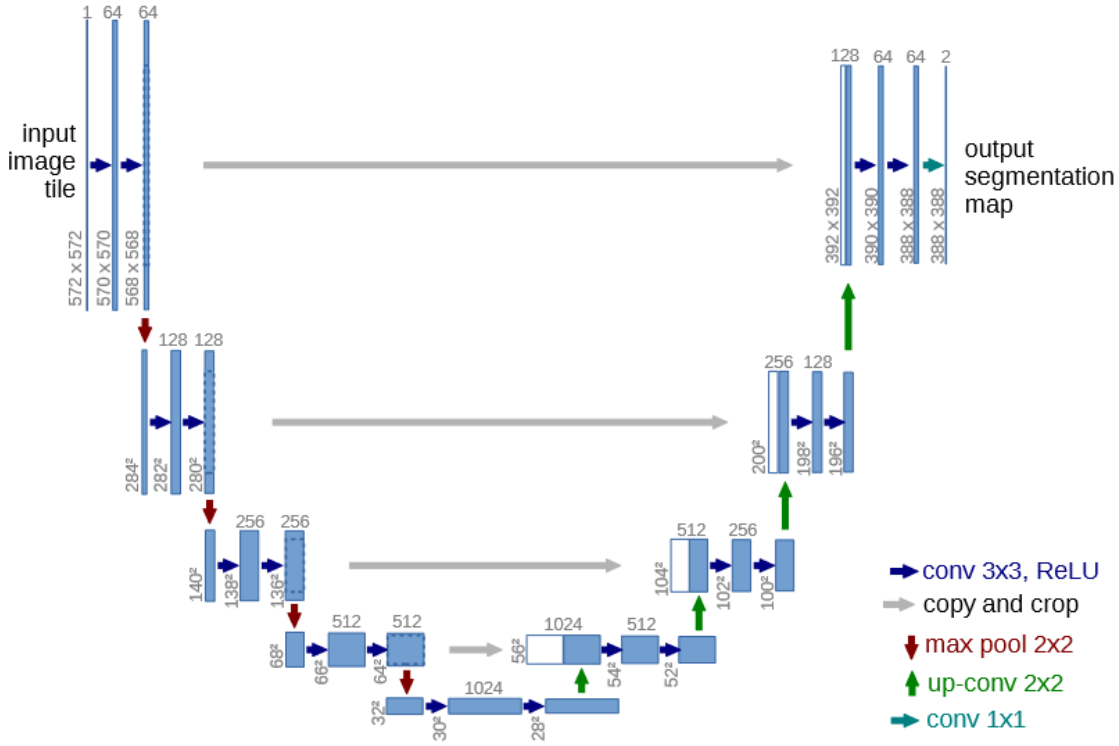


Figura 10 – Esquema que descreve uma *U-Net*. Adaptado de RONNEBERGER; FISCHER; BROX (2015)

Um dos fatores que limitou o desenvolvimento de redes neurais no passado foi a carência de recursos computacionais, especialmente na hora do treinamento, dado que isso em geral requer que sejam executados várias vezes o mesmo algoritmo sobre

um conjunto grande de imagens. O aumento exponencial no poder de processamento dos computadores permitiu que redes cada vez maiores se tornassem viáveis. Em particular, a utilização de *Graphics Processing Units* (GPUs) que permitem um número elevado de operações simultâneas, o que tornou o processo várias vezes mais rápido (OH; JUNG, 2004).

Resnets são redes nas quais o fluxo de dados não é estritamente sequencial. Nelas, após uma ou mais operações de convolução, o resultado, além de ser propagado para a camada imediatamente posterior, também será propagado e concatenado à saída de outra camada, abordagem referida como *skip-connections* ou *residual-connections* (TARG; ALMEIDA; LYMAN, 2016). Argumenta-se que o caminho com o menor número de conexões é ajustado inicialmente e, a medida que o treinamento progride, os caminhos mais longos são utilizados para refinar a predição, tal como foi sumarizado por JASTRZĘBSKI et al. (2017).

Neste capítulo discutimos aspectos sobre LFs, como eles são codificados e RNC que são importantes para o entendimento dos trabalhos relacionados que serão apresentadas no Capítulo 3 e das soluções propostas que serão apresentados no Capítulo 4.

3 TRABALHOS RELACINADOS

Existem diversas propostas que utilizam redes neurais para módulos específicos na codificação de imagens e de vídeos.

O trabalho apresentado por LI et al. (2017) utiliza uma rede neural *fully conected* para a predição intra de blocos do HEVC. Ela possui como entrada as amostras que estão acima e à esquerda do bloco a ser predito. Em particular, foram consideradas amostras até $L = 4$ linhas acima e à esquerda do bloco a ser predito, que, neste caso, tem tamanho 8×8, tal como está na Figura 11, obteve-se uma redução de 1.1% no BD-Rate para o componente Y.

PFAFF et al. (2018) seguem a mesma abordagem, porém com vários tamanhos de bloco possíveis, que levou a uma redução da taxa de bits de, aproximadamente, 3%.

Canvas PARK; KIM (2016) propõem substituir o *Simple Adaptive Offset* (SAO) do HEVC por uma rede neural convolucional, com o objetivo de melhorar a qualidade da imagem após a codificação. Observou-se que o SAO não funciona como um módulo preditor, pois, após a predição, **não** são gerados resíduos que passam por transformada e quantização. Em contraste, redes neurais podem reconstruir uma imagem mais próxima da original quando comparadas ao módulo SAO do HEVC. O estudo indicou que há tanto ganhos quanto perdas no PSNR, mas, de modo geral, os ganhos são mais evidentes na configuração *All-Intra*, com uma redução do BD-Rate de 4.8%, especialmente para valores mais altos dos parâmetros de quantização (*quantization parameters*).

A solução apresentada em ZHAO et al. (2018) implementa *Bi-Prediction* do HEVC via redes neurais convolucionais de forma a realizar a compensação de movimento de maneira melhor. Para tanto, foi treinada uma rede neural para cada *Quantization Parameter*. Observa-se que, mesmo no pior caso, houve reduções na BD-Rate de apenas 0.8%, e, nos melhores casos, quando os vídeos foram codificados com a configuração *Random Access* - houve uma redução de 3%

O trabalho descrito em SUN; FAN; ZHAO (2020) implementa uma RNC como uma possível etapa adicional na predição do *Audio Video Standard 3* (AVS3). Neste trabalho, depois de decidido qual o preditor (no caso do angular ou cópia), o bloco predito

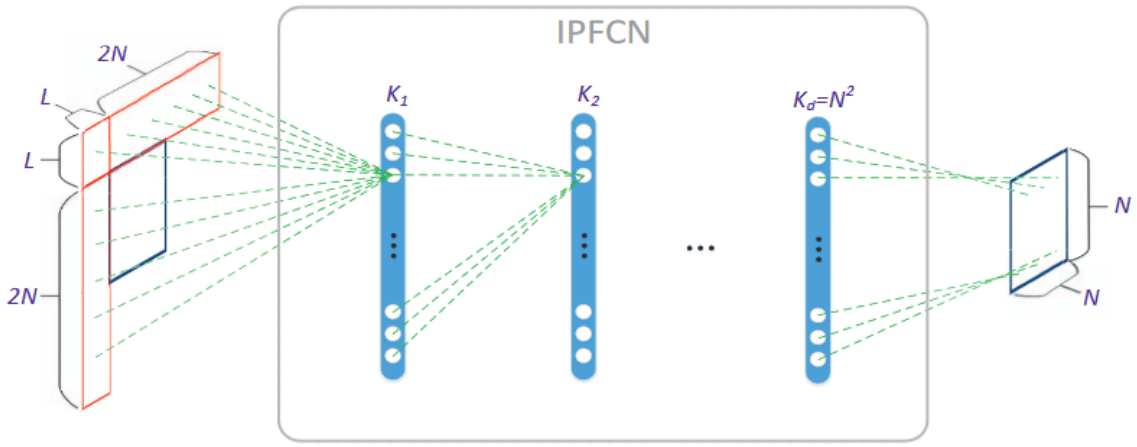


Figura 11 – Arquitetura utilizada em LI et al. (2017). Adaptada de LI et al. (2017)

pode, opcionalmente, passar por uma etapa que aplica uma RNC, recebendo tanto o bloco predito quanto alguma vizinhança acima e à esquerda na expectativa de melhorar a predição e reduzir os resíduos as serem codificados. Observa-se que a maior parte das reduções em taxa de bits ocorre no canal Y: em torno de 1.37%, se comparado a 0.58% e 0.64% nos canais U e V, respectivamente.

SPADARO et al. (2023) implementa predição intra utilizando *inpainting* – abordagens que preenchem "buracos" de imagens, em particular, utilizando **convoluções mascaradas**, que operam considerando quais pedaços já existem e quais devem ser preenchidos. Obteve-se redução de 6.55% no BD-rate.

Em HUANG; SCHIOPU; MUNTEANU (2019) é utilizado um conjunto de RNCs, cada uma para os modos de predição angular m_2, m_6, m_{10}, m_{14}, m_{18}, m_{22}, m_{26}, m_{30} e m_{34} do HEVC, colocando uma etapa na qual uma RNC pega o bloco predito e mais uma vizinhança de 16×16. Observou-se reduções no BD-Rate em torno de 0.85% considerando o canal Y, em média. Também chegaram a ganhos maiores a medida que a definição do vídeo aumentou.

Em JAHNKE (2021) são levantadas propostas similares as que são feita neste trabalho – utilizando redes neurais convolucionais para a predição de LFs, porém o trabalho não avalia os preditores dentro do contexto de um codificador.

Podemos observar o desempenho dos trabalhos relacionados na tabela 1.

Os trabalhos referenciados neste capítulo mostram que é possível utilizar redes neurais - inclusive convolucionais, para melhorar a qualidade da predição na codificação de vídeos. Nenhum dos trabalhos anteriores tratam de LFs, porém eles utilizam a mesma ideia central - utilizar redes neurais para obter preditores melhores.

Tabela 1 – Resultados dos trabalhos relacionados

Trabalho	BD-Rate	Padrão	Componente Implementado
LI et al. (2017)	1.1%	HEVC	Preditor Intra
PARK; KIM (2016)	4.8%	HEVC	*In-loop filter*
ZHAO et al. (2018)	3.0%	HEVC	*bi-predictor*
HUANG; SCHIOPU; MUNTEANU (2019)	0.85%	HEVC	Preditor Intra
SUN; FAN; ZHAO (2020)	1.3%	AVS3	Preditor Intra
JAHNKE (2021)	NaN	Nenhum	Preditor Intra
SPADARO et al. (2023)	6.55%	EVC	Preditor Intra

4 SOLUÇÕES PROPOSTAS

Este trabalho irá propor e avaliar preditores envolvendo redes neurais convolucionais. Inicialmente, será destacada a metodologia utilizada para treinar as redes e, com elas treinadas, testar o desempenho dentro do codificador MPEG-5; em seguida, serão descritas as arquiteturas propostas.

4.1 Metodologia

Será descrito como foi a preparação do conjuntos de dados a serem usados e como foi o processo de treinamento, respectivamente.

4.1.1 Preparando os conjuntos dados

Para montar um conjunto de dados para o treinamento, validação, utilizou-se pelo menos o *dataset* da École Polytechnique Fédérale de Lausanne (EPFL), com 115 LFs distintos (RERABEK; EBRAHIMI, 2016) retirando 14 LFs listados abaixo, pois estes serão utilizados nos testes para evitar que um eventual *sobreajuste* nos dados sobrestime a performance.

- *Ankylosaurus-Diplodocus*
- *head-anky*
- *Bikes*
- *Houses-Lake*
- *Black-Fence*
- *Reeds*
- *Ceiling-Light*
- *Rusty-Fence*
- *Danger-de-Mort*

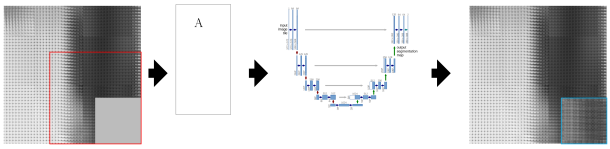


Figura 12 – Figura mostrando o processo de treinamento/inferência

- *Slab-Lake*
- *Friends*
- *Swans-2*
- *head2-anky*
- *Vespa*

Os LFs foram preprocessados com a remoção de duas linhas e duas colunas de SAIs nas bordas, além do ajuste dos valores das SAIs imediatamente internas, com o objetivo de corrigir o *vignetting*. Dessa forma, os LFs, originalmente compostos por 15 × 15 SAIs, foram reduzidos para 13 × 13 SAIs. Também foi utilizado um utilitário fornecido com o software de referência do JPEG Pleno, chamado *lenslet13x13shifter*. Esses LFs serão utilizados como amostras alvo, que as arquiteturas devem aprender a prever a partir da vizinhança.

As vizinhanças causais foram obtidas recortando regiões de 64×64, correspondentes a 4 blocos, a partir de um LF no formato *lenslet*. Em seguida, a área de 32×32 localizada no canto inferior esquerdo foi preenchida com pixels cinza de valor RGB $(127, 127, 127)$, similar ao que está na figura 12. Essa etapa visa evitar vazamento de dados, já que, durante a inferência, o modelo terá acesso apenas aos blocos reconstruídos pelo codificador ou decodificador. Para blocos menores, o contexto de 64×64 foi mantido, mas com a exclusão da região ainda não codificada. O treinamento considera apenas a região $n \times n$, a partir do quadrante inferior direito dessa vizinhança.

Com os dados, são treinados três modelos para cada arquitetura proposta, um para cada tamanho de bloco: 8×8, 16×16 e 32×32. O treinamento consiste em utilizar a vizinhança causal para prever o bloco atual. O processo de treinamento será descrito a seguir.

4.1.2 Processo de treinamento

O processo consiste em treinar redes neurais convolucionais para resolver o problema de predição. As entradas são os blocos vizinhos dos LFs codificados e decodificados, também chamados de vizinhança causal, e a saída esperada é o bloco correspondente no LF original.

A implementação dos modelos de redes foi feita com o auxílio da biblioteca *torch*, uma biblioteca *open source* desenvolvida em CUDA, C++ e Python, que oferece diversos recursos úteis para a criação e o uso de Redes Neurais.

Foi utilizando o otimizador Adam (KINGMA; BA, 2014) um *learning rate* inicial de 10^{-3}, com momento de 0.9, treinando por 100 épocas. A cada época, reduz-se o *learning rate* em 10%, estratégia conhecida como **decaimento exponencial**, dado que isso é equivalente a multiplicar 0.9. Essas decisões são, até certo ponto, arbitrárias, as redes são

Em cada época, são percorridos todos os LFs do conjunto de treinamento, recolhendo recortes aleatórios a serem utilizados. Após isso, para a validação, os LFs do respectivo grupo são escolhidos e, dentro de um LF, amostrados de maneira similar para obter o desempenho da validação.

Para agilizar o processo de treinamento, foram utilizadas unidades de processamento gráfico (GPUs) e o treinamento realizado em *mini-batches*. Nessa abordagem, os exemplos de treinamento são agrupados em mini-lotes, e a atualização dos pesos ocorre uma vez por *mini-batch*. Esse agrupamento permite a execução paralela de diversas operações, otimizando o uso computacional e acelerando o processo de aprendizagem do modelo.

4.1.3 Processo de teste

Para a realização dos experimentos foi utilizado o MPEG-5, ou *Essential Video Coding* (EVC) do Moving Pictures Expert Group(MPEG) modificado de forma a utilizar também os preditores propostos. Foi avaliado as métricas BD-Rate, BD-PSNR para os Parâmetros de quantização ou *Quantizations Parameters*(QPs) 22, 27, 32, 37 e 42. Parâmetros de quantização determinam o quão comprimido a entrada será: valores menores significam menos compressão, ou seja, maior tamanho do arquivo e qualidade, enquanto que valores maiores significam maior compressão, ou seja, menor tamanho de arquivo e qualidade(CHOI et al., 2020).

Uma configuração para o codificador modificado é o "*oracle*" que indica se o codificador deve substituir os preditores convencionais pelos baseados em redes neurais para aquele tamanho, ou se ambos os preditores coexistem, sendo um escolhido ou não acima do outro. variadas as arquiteturas, a serem discutidas com mais detalhes na seção 4.2.

A avaliação final foi feita sobre o modelo treinado sobre todo o conjunto de treinamento, utilizando os pesos da época em que ele teve o melhor desempenho no conjunto de validação. Será utilizado o codificador EVC (CHOI et al., 2020) modificado de forma que a utilizar também os preditores propostos.

É importante definir quais os tamanhos de blocos irão utilizar os preditores propostos. Podemos considerar os blocos de tamanho 32x32, 16x16, 8x8 e 4x4 independen-

temente um do outro. Como não foi treinado um preditor para tamanho de bloco 4x4, o blocos avaliados são de tamanho 32x32, 16x16 ou 8x8.

4.2 Arquiteturas propostas

Foram propostas duas arquiteturas que se distinguem através da forma com que elas tratam a entrada: uma delas, a chamada *angular-first* e outra chamada *space-first*.

Ambas as arquiteturas utilizam a mesma estrutura baseada em camadas convolucionais, organizadas no formato de uma *U-Net*, conforme ilustrado na Figura 13.

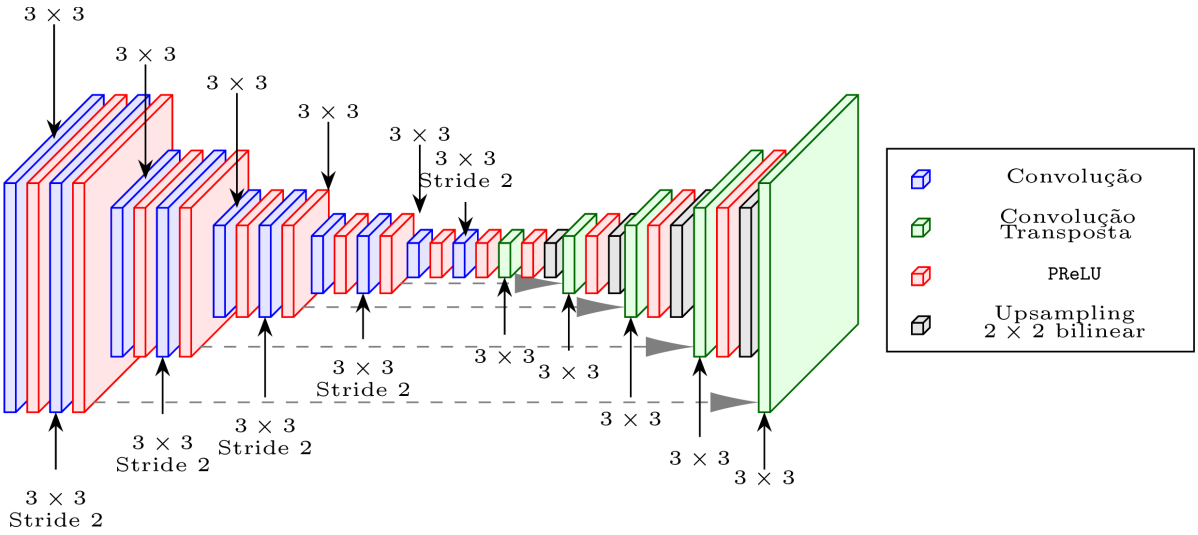


Figura 13 – Camadas convolucionais das arquiteturas propostas

O modelo aplica duas convoluções com *kernels* de tamanho 3×3, sendo que a segunda possui *stride* dois – ou seja, um deslocamento na entrada corresponde a metade do deslocamento na saída. o que reduz as dimensões por esse fator. Esse padrão se repete por quatro etapas consecutivas, reduzindo progressivamente o tamanho da representação.

Em seguida, a rede é espelhada: as camadas com *stride* dois são substituídas por operações de *upsampling*, que aumentam as dimensões da entrada, enquanto as camadas sem *stride* são substituídas por **convoluções transpostas**. Essas últimas são responsáveis por restaurar o tamanho original das ativações, mantendo a simetria da arquitetura.

Durante essa metade, as *skip-connections* desempenham um papel importante: elas transferem as ativações das camadas com *stride* dois da metade anterior e as combinam, concatenando, com as saídas correspondentes das camadas de *upsampling*, permitindo à rede recuperar detalhes espaciais perdidos nas etapas anteriores.

A arquitetura *angular-first* recebe a entrada em formato *lenslet* e usa tal como está. Enquanto isso, a arquitetura *spacefirst* recebe a entrada nesse formato mas a

rearranja os pixeis de tal forma que ela se assemelhe mais a uma *super-multiview*, colando pedaços de SAIs presentes e então, aplica a rede neural com a mesma disposição que a arquitetura anterior, e depois rearranja novamente os resultados numa rede em formato *lenslet* para que a resposta seja compatível.

Neste capítulo inicialmente discutimos a metodologia feita, qual conjunto de dados foi utilizado, em seguida foi descrito fluxo de dados, o treinamento e o teste. Em seguida foram propostas arquiteturas de RNCs que fazem a predição de blocos de LFs. Em sequência será apresentado os resultados no capítulo 5.

5 RESULTADOS E DISCUSSÕES

Conforme demonstra a Tabela 2, observamos que os ganhos de desempenho são mais significativos ao se utilizar blocos de tamanho 8. Nota-se, inclusive, que quando a opção *oracle* está desativada, a adoção de blocos de tamanho 16 e 32 resulta em perdas de desempenho, pois a rede neural apresenta desempenho inferior ao tentar prever blocos que abrangem áreas mais distantes. Além disso, é importante destacar que as arquiteturas *space-first* apresentam resultados superiores às arquiteturas *angle-first*, resultado que inclusive foi encontrado também em JAHNKE (2021).

Uma explicação para os erros maiores ao tentar prever blocos maiores é o fato de que a maior distância nos extremos entre a região conhecida e a região a ser predita implica numa conexão mais fraca, dependendo de caminhos que passam por mais camadas e não usufruindo tanto das *skip connections* presentes na arquitetura. Esse conceito é formalizado com a ideia de *receptive field* (LUO et al., 2016).

Outra explicação para o aumento do erro em soluções que envolvem blocos maiores é a maior ambiguidade na predição de elementos extensos: quanto maior a região a ser predita, mais possibilidades distintas de como preenchê-la existem, dificultando a predição da redes neurais devido à sua tendência a média dados múltiplos resultados plausíveis para uma mesma entrada (LI et al., 2024).

Um jeito de tratar esse problema seria através de modelos de difusão, que gradualmente reduzem o ruído(CHANG; KOULIERIS; SHUM, 2023), inclusive utilizados nas ferramentas de geração de imagens através de *prompt*, ferramentas tais como *DALL-E*

Tabela 2 – Resultados da codificação sem *oracle*

	Angle-First		Space-first	
Tamanhos de bloco	BD-Rate	BD-PSNR	BD-Rate	BD-PSNR
(32)	1.02	-0.05	0.47	-0.02
(32, 16)	1.02	-0.05	0.19	-0.01
(32, 16, 8)	1.73	-0.09	-2.14	0.11
(16)	0.28	-0.01	-0.69	0.03
(16, 8)	0.94	-0.05	-2.87	0.15
(8)	0.67	-0.03	**-2.98**	**0.16**

Tabela 3 – Resultados da codificação com *oracle*

	Angle-First (Oracle)		Space-first (Oracle)	
Tamanhos de bloco	BD-Rate	BD-PSNR	BD-Rate	BD-PSNR
32	0	0	0	0
(32, 16)	-0.06	0	-1.12	0.05
(32, 16, 8)	-0.17	0.01	**-4.52**	**0.22**
16	-0.06	0	-1.07	0.05
(16, 8)	-0.16	0.01	-4.49	0.22
8	-0.09	0	-4.1	0.2

Tabela 4 – Resultados de BD-PSNR e BD-Rate por LF para os preditores propostos dentro do codificador comparado com a configuração utilizando apenas os preditores convencionais

Configuração	Space(32,16,8)		Angle(32,16,8)	
LF	BD-Rate	BD-PSNR	BD-Rate	BD-PSNR
Ankylosaurus-Diplodocus	-5,66	0,24	-0,09	0
Bikes	-3,55	0,2	-0,12	0,01
Black-Fence	-3,35	0,24	-0,02	0
Ceiling-Light	-1,7	0,1	-0,1	0
Danger-de-Mort	-12,25	0,65	-0,54	0,03
Friends-1	-1,28	0,07	-0,02	0
Houses-Lake	-0,57	0,02	0,01	0
Reeds	-0,28	0,01	-0,06	0
Rusty-Fence	-8,37	0,39	-0,69	0,03
Slab-Lake	-2,27	0,1	-0,09	0
Swans-2	-11,98	0,53	-0,2	0,01
Vespa	-2,92	0,14	-0,07	0
Average	-4,52	0,22	-0,17	0,01

(BETKER et al., 2023).

Observa-se ainda na tabela que, caso os a opção *oracle* esteja desativada, ocorre perda de eficiência de codificação para os preditores *Angle-First*, com a menor perda ocorrendo quando estamos utilizando o tamanho de bloco 16.

O desempenho dos preditores varia significativamente de acordo com o LF que está sendo tratado, podemos observar isso na Tabela 4, Observando por exemplo que o LF *Danger-de-Mort* tem uma economia em BD-Rate de $12,25\%$ comparado com uma economia em *Reeds* de 0.28%.

Outro meio de avaliar o desempenho do preditor é pelos *logs* emitidos pelo codificador. Ele emite os custos, uma soma ponderada entre a distorção e a taxa de bits, o codificador otimiza essa medida para obter o melhor *trade-off* entre qualidade e compressão – alterando os pesos pode-se priorizar maior qualidade ao custo de uma compressão menos eficiente ou o contrário – referida como **otimização de taxa/distorção**, ou ***rate distortion optimization***(RDO).

Tabela 5 – Resultados dos custos reportados pelo codificador comparando os preditores convencionais (costEVC) com o baseado em redes neurais(costNN) por tamanho de bloco

Tamanho de bloco	Melhorados (%)	Custo EVC	Custo NN(Todos)	Redução(%)
32	4,75%	53194	74953,60	-40,91
16	20,76%	15575,1	18429	-18,32
8	42,19%	5044,06	4861,09	3,63

Tabela 6 – Resultados dos custos reportados pelo codificador comparando os preditores convencionais (costEVC) com o baseado em redes neurais(costNN) por tamanho de bloco, considerando apenas os custos RD quando costNN < costEVC

Tamanho de bloco	Melhorados (%)	Custo EVC	Custo NN (somente NN < EVC)	Redução(%)
32	4,75134%	4026,21	3561,27	11,55%
16	20,76%	4943,05	4171,93	15,6%
8	42,19%	3295,27	2527,39	23,30%

A Tabela 5 apresenta os resultados obtidos a partir dos *logs* gerados durante o processo de codificação, considerando os tamanhos de bloco 32, 16 e 8 com o uso de *oracle*.

Para cada tamanho de bloco, o codificador avalia tanto os preditores convencionais (costEVC) quanto os propostos (costNN), registrando os custos correspondentes. A média desses custos foi calculada e está apresentada na tabela. A coluna "Melhorados (%)"indica a proporção de blocos em que o preditor proposto resultou em menor perda, em comparação aos preditores convencionais. Já a coluna "Redução (%)"mostra, de forma proporcional, o quanto o custo associado ao *costNN* foi inferior ao do *costEVC*.

Os dados indicam que o desempenho é melhor com blocos menores. Especificamente, à medida que o tamanho do bloco diminui, aumenta a probabilidade de o preditor proposto superar os convencionais (em termos de porcentagem de blocos com menor custo), além de se observar uma redução média mais significativa no custo RD.

A tabela 5, apesar de já demonstrar a utilidade do preditor baseado em redes neurais, não considera um dos comportamentos mais relevantes de um codificador preditivo: testando mais de um preditor, utiliza-se o melhor, ou seja, aquele que tem a menor distorção. Considerando essa propriedade, a tabela 6 considera apenas os custos quando os preditores propostos são melhores que os do codificador. Observamos aqui uma redução expressiva no custo RD mesmo quando os tamanhos de bloco são maiores, ao custo de ocorrerem com menor frequência.

Os resultados completos de acordo com essa técnica encontram-se na tabela 7,

anota-se que as linhas com traços referem-se a tamanhos de bloco em que não foram utilizados os preditores propostos aqui, somente os convencionais. Observamos que os melhores resultados ocorrem com a arquitetura *space-first*. Na arquitetura *angle-first*, ocorre um comportamento anômalo: o comportamento dos preditor para os blocos 16×16 é aparentemente melhor, em termos de escolhas que os blocos 8×8, dado que aquele é escolhido 21% das vezes, comparados a este, que é escolhido em torno de 10%.

Neste capítulo discutiram-se os resultados obtidos pelas arquiteturas propostas, em seguida, no capítulo 6, serão feitas as considerações finais.

Tabela 7 – Tabela com os resultados do custo RD nas diversas configurações testadas, considerando apenas quando o custo EVC é maior que o custoNN

Configuração	Tamanho	Melhorados(%)	CostEVC	CostNN	Redução (%)
Angle-First	32	0,567033	801,735	783,44	2,28193
32	16	-	-	-	-
Oracle	8	-	-	-	-
Space-First	32	4,65409	3845,09	3409,81	11,3203
32	16	-	-	-	-
Oracle	8	-	-	-	-
Angle-First	32	-	-	-	-
16	16	21,3713	5945,04	5396,8	9,22177
No Oracle	8	-	-	-	-
Angle-First	32	-	-	-	-
16	16	21,3687	5952,55	5398,47	9,30813
Oracle	8	-	-	-	-
Space-First	32	-	-	-	-
16	16	20,3149	4812,82	4069,84	15,4375
No Oracle	8	-	-	-	-
Space-First	32	-	-	-	-
16	16	20,5236	4843,75	4091,95	15,521
Oracle	8	-	-	-	-
Angle-First	32	0,570968	811,682	793,021	2,29909
32,16	16	21,3665	5943,7	5390,24	9,31167
Oracle	8	-	-	-	-
Space-First	32	4,70593	3924,96	3468,69	**11,6248**
32,16	16	20,5268	4833,63	4083,17	15,5259
Oracle	8	-	-	-	-
Angle-First	32	-	-	-	-
8	16	-	-	-	-
Oracle	8	10,0727	1046,7	945,973	9,62321
Space-First	32	-	-	-	-
8	16	-	-	-	-
Oracle	8	**42,1976**	3314,19	2543,46	23,2555
Angle-First	32	-	-	-	-
16, 8	16	21,4209	5954,43	5398,36	9,33873
Oracle	8	10,0739	1042,16	940,986	9,708
Space-First	32	-	-	-	-
16,8	16	20,7499	4956,16	4185,69	15,5456
Oracle	8	42,1826	3294,68	2526,17	**23,3259**
Angle-First	32	0,566575	798,906	780,482	2,30618
32,16,8	16	**21,4222**	5956,65	5400,14	9,34273
Oracle	8	10,0706	1041,83	940,745	9,703
Space-First	32	**4,75134**	4026,21	3561,27	11,5477
32,16,8	16	20,7591	4943,05	4171,93	**15,6**
Oracle	8	42,1931	3295,27	2527,39	23,3022

6 CONCLUSÃO

Esta dissertação propôs preditores que utilizam RNCs. Introduziu-se conceitos de LFs, de aprendizado de máquina, de RN, de RNCs, *Res-nets* e *U-Nets*. Em seguida discutiu-se trabalhos que utilizam RNs em módulos específicos de codificadores.

Na linha de investigação, propuseram-se arquiteturas de redes como módulos preditores, avaliando a sua eficácia num codificador EVC, obtendo resultados ganhos na eficiência de codificação, observando que o preditor que planifica um bloco de um LF concatenando pedaços de SAIs, chamado neste texto de ***Space-first*** tem resultados melhores comparado ao preditor que planifica concatenando MIs, chamado neste texto de ***Angle-first***. Observamos ainda que arquiteturas que trabalham com tamanhos de blocos menores, no caso, 8×8 têm resultados melhores comparadas as que trabalham com blocos maiores, no caso 16×16 e 32×32.

Trabalhos futuros nessa linha de investigação incluem outras arquiteturas e topologias de redes neurais, outras regiões de vizinhança, outros meios de dispor os LFs e outros modos de considerar as convoluções. Para outras arquiteturas pode-se considerar redes mais profundas, ou seja, com mais camadas, possivelmente envolvendo outros canais de cores e algum tratamento adicional. Mas outro método consiste em utilizar outro tipo de arquitetura, tal como os modelos *transformers*, que utilizam um mecanismo de atenção; ou os modelos de difusão, que, ao invés de prever diretamente o imagem, preveem o ruído que ela tem, de forma que essa interação.

REFERÊNCIAS

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.

BETKER, J. et al. Improving image generation with better captions. **Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf**, [S.l.], v.2, n.3, p.8, 2023.

BJONTEGAARD, G. Calculation of average PSNR differences between RD-curves. **ITU SG16 Doc. VCEG-M33**, [S.l.], 2001.

BLOCK, H.-D. The perceptron: A model for brain functioning. i. **Reviews of Modern Physics**, [S.l.], v.34, n.1, p.123, 1962.

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.

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.

CHANG, Z.; KOULIERIS, G. A.; SHUM, H. P. On the design fundamentals of diffusion models: A survey. **arXiv preprint arXiv:2306.04542**, [S.l.], 2023.

CHEN, J.; HOU, J.; CHAU, L.-P. Light field compression with disparity-guided sparse coding based on structural key views. **IEEE Transactions on Image Processing**, [S.l.], v.27, n.1, p.314–324, 2017.

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.

CUI, W. et al. Convolutional neural networks based intra prediction for HEVC. **arXiv preprint arXiv:1808.05734**, [S.l.], 2018.

FU, W. et al. Implementing light field image refocusing algorithm. In: INTERNATIONAL CONFERENCE ON OPTO-ELECTRONICS AND APPLIED OPTICS (IEM OPTRONIX), 2015., 2015. **Anais...** [S.l.: s.n.], 2015. p.1–8.

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.

GIROD, B.; CHANG, C.-L.; RAMANATHAN, P.; ZHU, X. Light field compression using disparity-compensated lifting. In: IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH, AND SIGNAL PROCESSING, 2003. PROCEEDINGS.(ICASSP'03)., 2003., 2003. **Anais...** [S.l.: s.n.], 2003. v.4, p.IV–760.

GOMES, P. d. M. **Pseudo-Sequence Light Field Image Representation and Encoding for Improved Random Access and Scalability**. 2019. Tese (Doutorado em Ciência da Computação) — Universidade de Coimbra.

GROSSMAN, J. P.; DALLY, W. J. Point sample rendering. In: RENDERING TECHNIQUES' 98: PROCEEDINGS OF THE EUROGRAPHICS WORKSHOP IN VIENNA, AUSTRIA, JUNE 29—JULY 1, 1998 9, 1998. **Anais...** [S.l.: s.n.], 1998. p.181–192.

HALAGUNDEGOWDA, G. Evaluation of classification ability of support vector machine (SVM) in binary classification problems. **Training**, [S.l.], 1980.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. H.; FRIEDMAN, J. H. **The elements of statistical learning**: data mining, inference, and prediction. [S.l.]: Springer, 2009. v.2.

HEBER, S.; RANFTL, R.; POCK, T. Variational shape from light field. In: INTERNATIONAL WORKSHOP ON ENERGY MINIMIZATION METHODS IN COMPUTER VISION AND PATTERN RECOGNITION, 2013. **Anais...** [S.l.: s.n.], 2013. p.66–79.

HECHT-NIELSEN, R. Theory of the backpropagation neural network. In: **Neural networks for perception**. [S.l.]: Elsevier, 1992. p.65–93.

HUANG, H.; SCHIOPU, I.; MUNTEANU, A. Deep learning based angular intra-prediction for lossless HEVC video coding. In: DATA COMPRESSION CONFERENCE (DCC), 2019., 2019. **Anais...** [S.l.: s.n.], 2019. p.579–579.

JAHNKE, M. d. S. Predição de Light Fields utilizando Redes Neurais Convolucionais. **Trabalho de Conclusão de Curso (Graduação em Engenharia de Computação) — Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas**, [S.l.], p.49, 2021.

JANOCHA, K.; CZARNECKI, W. M. On loss functions for deep neural networks in classification. **arXiv preprint arXiv:1702.05659**, [S.l.], 2017.

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

KHOURY, J.; POURAZAD, M. T.; NASIOPOULOS, P. A New Prediction Structure for Efficient MV-HEVC based Light Field Video Compression. In: INTERNATIONAL CONFERENCE ON COMPUTING, NETWORKING AND COMMUNICATIONS (ICNC), 2019., 2019. **Anais...** [S.l.: s.n.], 2019. p.588–591.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. **arXiv preprint arXiv:1412.6980**, [S.l.], 2014.

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.

LEMARÉCHAL, C. Cauchy and the gradient method. **Doc Math Extra**, [S.l.], v.251, n.254, p.10, 2012.

LEVIN, A.; DURAND, F. Linear view synthesis using a dimensionality gap light field prior. In: IEEE COMPUTER SOCIETY CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 2010., 2010. **Anais...** IEEE, 2010. p.1831–1838.

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, B.; PAN, Z.; LYU, K.; LI, J. Feature Averaging: An Implicit Bias of Gradient Descent Leading to Non-Robustness in Neural Networks. **arXiv preprint arXiv:2410.10322**, [S.l.], 2024.

LI, J.; LI, B.; XU, J.; XIONG, R. Intra prediction using fully connected network for video coding. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 2017., 2017. **Anais...** [S.l.: s.n.], 2017. p.1–5.

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.

LUO, W.; LI, Y.; URTASUN, R.; ZEMEL, R. Understanding the effective receptive field in deep convolutional neural networks. **Advances in neural information processing systems**, [S.l.], v.29, 2016.

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

PARK, W.-S.; KIM, M. CNN-based in-loop filtering for coding efficiency improvement. In: IEEE 12TH IMAGE, VIDEO, AND MULTIDIMENSIONAL SIGNAL PROCESSING WORKSHOP (IVMSP), 2016., 2016. **Anais...** [S.l.: s.n.], 2016. p.1–5.

PEREIRA, F. et al. JPEG Pleno Light Field Coding Common Test Conditions. In: STANDARD ISO/IEC JTC 1/SC29/WG1N84025, 84TH MEETING, 2019. **Anais...** [S.l.: s.n.], 2019.

PERRA, C.; ASSUNCAO, P. High efficiency coding of light field images based on tiling and pseudo-temporal data arrangement. In: IEEE INTERNATIONAL CONFERENCE ON MULTIMEDIA & EXPO WORKSHOPS (ICMEW), 2016., 2016. **Anais...** [S.l.: s.n.], 2016. p.1–4.

PERRA, C.; FREITAS, P. G.; SEIDEL, I.; SCHELKENS, P. An overview of the emerging JPEG Pleno standard, conformance testing and reference software. In: OPTICS, PHOTONICS AND DIGITAL TECHNOLOGIES FOR IMAGING APPLICATIONS VI, 2020. **Anais...** SPIE, 2020. v.11353, p.207 – 219.

PFAFF, J. et al. Neural network based intra prediction for video coding. In: APPLICATIONS OF DIGITAL IMAGE PROCESSING XLI, 2018. **Anais...** [S.l.: s.n.], 2018. v.10752, p.1075213.

PFAFF, J. et al. Intra prediction and mode coding in VVC. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], 2021.

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.

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. **Principles of Neurodynamics**: Perceptrons and the Theory of Brain Mechanisms. [S.l.]: Spartan Books, 1961.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning internal representations by error propagation, parallel distributed processing, explorations in the microstructure of cognition, ed. de rumelhart and j. mcclelland. vol. 1. 1986. **Biometrika**, [S.l.], v.71, n.599-607, p.6, 1986.

SAMUEL, A. L. Some studies in machine learning using the game of checkers. **IBM Journal of research and development**, [S.l.], v.3, n.3, p.210–229, 1959.

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.111371G.

SHARMA, S.; SHARMA, S.; ATHAIYA, A. Activation functions in neural networks. **Towards Data Sci**, [S.l.], v.6, n.12, p.310–316, 2017.

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.

SULLIVAN, G. J.; OHM, J.-R.; HAN, W.-J.; WIEGAND, T. Overview of the high efficiency video coding (HEVC) standard. **IEEE Transactions on circuits and systems for video technology**, [S.l.], v.22, n.12, p.1649–1668, 2012.

SUN, C.; FAN, X.; ZHAO, D. Deep learning based intra prediction filter in avs3. In: IEEE INTERNATIONAL CONFERENCE ON MULTIMEDIA & EXPO WORKSHOPS (ICMEW), 2020., 2020. **Anais...** [S.l.: s.n.], 2020. p.1–6.

SZE, V.; BUDAGAVI, M.; SULLIVAN, G. J. High efficiency video coding (HEVC). In: **Integrated circuit and systems, algorithms and architectures**. [S.l.]: Springer, 2014. v.39, p.40.

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

WU, G. et al. Light field image processing: An overview. **IEEE Journal of Selected Topics in Signal Processing**, [S.l.], v.11, n.7, p.926–954, 2017.

ZHANG, G. **What is the kernel trick? Why is it important?** https://medium.com/@zxr.nju/what-is-the-kernel-trick-why-is-it-important-98a98db0961d.

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, Z. et al. Enhanced bi-prediction with convolutional neural network for high-efficiency video coding. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.29, n.11, p.3291–3301, 2018.

7 ASSINATURAS

gov.br
Documento assinado digitalmente
MATHEUS DA SILVA JAHNKE
Data: 13/06/2025 22:01:13-0300
Verifique em https://validar.iti.gov.br

Matheus da Silva Jahnke
Proponente

Bruno Zatt
Prof. Orientador