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

Dissertação

Aceleração do Particionamento de Quadros Intra no Codificador *Versatile Video Coding* (VVC) Utilizando Redes Neurais Profundas

Thiago Luiz Alves Bubolz

Pelotas, 2021

Thiago Luiz Alves Bubolz

Aceleração do Particionamento de Quadros Intra no Codificador *Versatile Video Coding* (VVC) Utilizando Redes Neurais Profundas

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

Orientador: Prof. Dr. Guilherme Ribeiro Corrêa
Coorientadores: Prof. Dr. Mateus Grellert da Silva
Prof. Dr. Bruno Zatt

Pelotas, 2021

Universidade Federal de Pelotas / Sistema de Bibliotecas
Catalogação na Publicação

B917a Bubolz, Thiago

Aceleração do particionamento de quadros intra no codificador Versatile Video Coding (VVC) utilizando redes neurais profundas / Thiago Bubolz ; Guilherme Ribeiro Corrêa, orientador ; Mateus Grellert da Silva, Bruno Zatt, coorientadores. — Pelotas, 2021.

71 f. : il.

Dissertação (Mestrado) — Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, 2021.

1. VVC. 2. Codificação. 3. Intra-quadro. 4. Redes neurais. 5. Computação. I. Corrêa, Guilherme Ribeiro, orient. II. Silva, Mateus Grellert da, coorient. III. Zatt, Bruno, coorient. IV. Título.

CDD : 005

Elaborada por Aline Herbstrith Batista CRB: 10/1737

Thiago Luiz Alves Bubolz

Aceleração do Particionamento de Quadros Intra no Codificador *Versatile Video Coding* (VVC) Utilizando Redes Neurais Profundas

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

Data da Defesa: 20 de Agosto de 2021

Banca Examinadora:

Prof. Dr. Guilherme Ribeiro Corrêa (orientador)

Doutor em Ciência da Computação pela Universidade de Coimbra.

Prof. Dr. Ricardo Matsumura Araujo

Doutor em Ciência da Computação pela Universidade Federal do Rio Grande do Sul.

Prof. Dr. Wagner Ishizaka Penny

Doutor em Engenharia de Sistemas Eletrônicos e Automação pela Universidade de Brasília.

Prof. Dr. Edson Mintsu Hung

Doutor em Biotecnologia pela Universidade Federal de Pelotas.

RESUMO

BUBOLZ, Thiago Luiz Alves. **Aceleração do Particionamento de Quadros Intra no Codificador *Versatile Video Coding* (VVC) Utilizando Redes Neurais Profundas**. Orientador: Guilherme Ribeiro Corrêa. 2021. 71 f. Dissertação (Mestrado em Ciência da Computação) – Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2021.

A predição intra-quadro desempenha um papel muito importante nas tecnologias atuais de codificação de vídeo. Assim como no atual padrão estado-da-arte em codificação de vídeo, o *High Efficiency Video Coding* (HEVC), o novo padrão *Versatile Video Coding* (VVC) adicionou novos modos de predição aliados a novos tipos de particionamento para a predição intra-quadro. A *Multi Type Tree* (MTT) possibilita uma melhoria no VVC sobre a estrutura de particionamentos em *quadtree* do padrão HEVC, gerando tamanhos de bloco mais flexíveis. Essas novas técnicas alcançam alta eficiência de codificação, mas também resultam em um custo computacional muito alto, por vezes até impedindo a adoção deste novo padrão na indústria. Para mitigar este problema, esta dissertação apresenta um algoritmo rápido para decisão de particionamento das *Coding Tree Units* (CTU) durante a codificação intra-quadro, possibilitando uma redução de custo computacional do processo. Uma das contribuições do trabalho consiste em formar um *dataset* com dados em quantidade e pluralidade significativa, com intuito de alimentar uma arquitetura de rede neural profunda capaz de classificar particionamentos de CTU. Os resultados experimentais mostraram que a solução proposta atingiu redução no tempo de codificação de até 21,41%, com perda na eficiência de codificação de 0,432%.

Palavras-chave: VVC. Codificação intra-quadro. Redes neurais. Complexidade.

ABSTRACT

BUBOLZ, Thiago Luiz Alves. **Accelerating Intra Frame Partitioning in Versatile Video Coding (VVC) Encoder Using Deep Neural Networks**. Advisor: Guilherme Ribeiro Corrêa. 2021. 71 f. Dissertation (Masters in Computer Science) – Technology Development Center, Federal University of Pelotas, Pelotas, 2021.

Intra-frame prediction plays a very important role in current video coding technologies. Therefore, the current state-of-the-art standard in video coding, High Efficiency Video Coding (HEVC), the new Versatile Video Coding (VVC) standard has added new prediction modes along with new types of partitioning for intra-frame prediction. The Multi Type Tree (MTT) enables an improvement in VVC over the quadtree partitioning structure of the HEVC standard, generating more flexible block sizes. These new techniques achieve high coding efficiency, but also result in a very high computational cost, sometimes even preventing the adoption of this new standard in the industry. To mitigate this problem, this dissertation presents a fast algorithm for the decision of Coding Tree Units (CTU) partitioning during intra-frame coding, enabling a reduction in the computational cost of the process. One of the contributions of the work is to form a dataset with data in significant quantity and plurality, in order to feed a deep neural network architecture capable of classifying CTU partitioning. The experimental results showed that the proposed neural network achieved a reduction in coding time of up to 21.41%, with a loss in coding efficiency of 0.432%.

Keywords: VVC. Intra Frame Encoding. Neural Networks. Complexity.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AMP	Adaptive Mode Pruning
CCLM	Cross Component Linear Model
CNN	Convolutional Neural Networks
CTC	Common Test Conditions
CTU	Coding Tree Units
CU	Coding Units
FME	Fractional Motion Estimation
FN	Falsos Negativos
FP	Falsos Positivos
FullHD	Full High Definition
GPU	Graphics Processing Unit
HD	High Definition
HEVC	High-Efficiency Video Coding
ISP	Intra Subpartition
JVET	Joint Video Experts Team
LIP	Line-Based Intra Prediction
MAE	Mean Absolute Error
MDT	Mode-Dependent Termination
ME	Motion Estimation
MIP	Matrix-based Intra Prediction
MPM	Most Probable Mode
MRL	Multiple Reference Line
MSE	Mean Squared Error
MTS	Multiple Transform Selection
MTT	Multi Type Tree
PSNR	Peak Signal-to-Noise Ratio

RD-Cost	Rate-Distortion Cost
RDO	Rate-Distortion Optimization
ReLU	Rectified Linear Unit
RGB	Red, Green and Blue
RMD	Rough Mode Decision
RT	Redução de Tempo
SATD	Sum of the Absolute Transformed Differences
SGD	Stochastic Gradient Descent
SI	Informação Espacial
TI	Informação Temporal
VN	Verdadeiros Negativos
VP	Verdadeiros Positivos
VTM	VVC Test Model
VVC	Versatile Video Coding
WAIP	Wide Angular Intra Prediction
YCbCr	Luminância, Crominância Azul e Crominância Vermelha

SUMÁRIO

1 INTRODUÇÃO

Os relatórios recentes da CISCO previram que o uso de *streaming* e download em serviços de conteúdo de vídeo ultrapassará 80% do tráfego da Internet até o ano de 2021 (INDEX, 2016). Devido a este alto tráfego de dados, a compressão de vídeo tornou-se fundamental para permitir o armazenamento e a transmissão desse tipo de conteúdo, especialmente com a popularização de vídeos de alta resolução espacial e altas taxas de quadros por segundo. Com a busca de se obter sempre uma melhor qualidade de experiência para o usuário com uma compressão do tamanho do vídeo cada vez maior, vários padrões de codificação de vídeo foram surgindo ao longo do tempo.

O *Versatile Video Coding* (VVC) é o novo padrão de codificação de vídeo, lançado em 2020 pelo *Joint Video Experts Team* (JVET) (JVET, 2020). Para os mesmos níveis de qualidade de imagem o VVC promete reduzir a taxa de bits em grande magnitude quando comparado ao seu antecessor, o padrão *High-Efficiency Video Coding* (HEVC) (JCTVC, 2013). No entanto, essas taxas de compressão são alcançadas com um custo significativo em termos de custo computacional, mostrando ser muito mais custoso que o HEVC comparação com o HEVC (Takamura, 2019).

O VVC introduziu diversas novas estruturas de particionamento mais flexíveis, sendo estas umas das principais responsáveis por ganhos de compressão obtidos pelo padrão, mas também responsáveis por grande parte do aumento substancial do custo computacional de codificação. Cada quadro do vídeo é inicialmente subdividido em regiões de tamanho 128$\times$128 pixels chamadas de *Coding Tree Units* (CTU), que podem ainda ser particionadas em regiões menores chamadas de *Coding Units* (CU).

As CUs são subdivididas recursivamente no formato de árvore quaternária, assumindo tamanhos possíveis de 64$\times$64 até 4$\times$4 pixels. Esta divisão quaternária inclui dois níveis a mais quando comparada com a estrutura quaternária permitida no codificador HEVC. Além desta divisão quaternária, o VVC ainda suporta divisões binárias e ternárias nas suas CUs.

Com estes novos particionamentos binários e ternários, CUs podem agora assumir formas que não sejam apenas quadradas, como nos padrões antigos. Portanto, é

possível observar CUs que podem ter tamanhos retangulares variados, como 128×4, 32×16, 8×64, etc. Esta nova estrutura de particionamento é chamada de *Multi Type Tree* (MTT).

Em um codificador de vídeo que busca a melhor relação entre taxa de bits e qualidade da imagem, a escolha de particionamento de cada CTU deve considerar a maior quantidade de possibilidades, em uma estratégia denominada Otimização Taxa-Distorção (*Rate-Distortion Optimization* - RDO) (SULLIVAN; WIEGAND, 1998). Neste caso, para cada CTU são testadas todas as profundidades possíveis de árvore quaternária (0, 1, 2, 3, 4, 5), juntamente com todas as combinações desses níveis da árvore quaternária com árvores binárias e ternárias filhas, de forma recursiva. Este processo busca encontrar o melhor particionamento e é responsável pela maior parte do custo computacional presente no codificador de referência do padrão VVC (JVET, 2020). Quando a estratégia RDO é utilizada, apenas após o codificador testar todas as possibilidades de particionamentos, a melhor combinação é escolhida de acordo com a métrica de Custo Taxa-Distorção (*Rate-Distortion Cost* - RD-Cost), que avalia o menor custo em termos de taxa de bits e distorção para a região a ser avaliada.

No Software de referência do padrão VVC, o *VVC Test Model* (VTM) (J. CHEN Y. YE, 2019), mesmo que haja uma probabilidade muito pequena de um formato de CU possuir o menor RD-Cost e ser escolhido, este particionamento é avaliado como todos os outros candidatos. Diversos trabalhos vêm sendo publicados com o objetivo de reduzir o grande custo computacional relacionado às decisões de particionamento do codificador VVC, como (TISSIER et al., 2019), (PARK; KANG, 2019), (LEI et al., 2019).

A área de aprendizado de máquina foca no desenvolvimento de modelos de aprendizagem computacionais capazes de aprender e melhorar a eficiência de determinada tarefa. Muito da sua popularidade atual se deve à recente viabilidade da área de *Deep Learning*, a qual engloba modelos mais complexos como Redes Neurais Convolucionais. O aprendizado de máquina vem sendo utilizado para auxiliar em soluções de redução de custo computacional na codificação de vídeo. Devido ao fato de o codificador VVC ter sido lançado recentemente, existem poucos trabalhos que exploram redução de custo computacional auxiliados por aprendizado de máquina e em sua maioria são trabalhos que exploram tarefas muito pontuais do codificador (FU et al., 2019), (TISSIER et al., 2020). Portanto, este trabalho tem como proposta desenvolver um modelo de Redes Neurais Profundas capaz de predizer particionamentos de CTU de acordo com o padrão VVC, de forma que não seja necessário realizar a busca exaustiva por todas as possibilidades de tamanho e formato de blocos.

1.1 Objetivo

O objetivo geral deste trabalho consiste em desenvolver uma solução com auxílio de Redes Neurais Profundas para redução de custo computacional da codificação vídeos com o novo padrão *Versatile Video Coding* (VVC). O escopo da estratégia proposta está concentrado nas decisões de tamanhos e formatos de blocos em CTUs de quadros do tipo Intra. A solução deverá explorar as características intrínsecas dos vídeos para realizar as decisões de particionamento sem que para isso seja necessário realizar uma busca exaustiva, atingindo dessa forma a redução de custo computacional. A solução poderá ainda ser utilizada em conjunto com outras técnicas de redução de custo computacional, aumentando assim a redução final. Os objetivos específicos para atingir este objetivo geral são enumerados abaixo:

Objetivo 1) Investigar o impacto dos novos tipos de particionamento permitidos no codificador VVC em termos de eficiência de compressão e custo computacional;

Objetivo 2) Gerar um *dataset* que permita detectar tendências de particionamentos de acordo com características dos vídeos;

Objetivo 3) Treinar modelos de Redes Neurais Profundas seguindo uma metodologia robusta a fim de evitar efeitos indesejados como *overfitting*;

Objetivo 4) Desenvolver e implementar uma técnica de redução de custo computacional no software de referência do codificador para avaliar o desempenho obtido;

Objetivo 5) Comparar os resultados obtidos com trabalhos relacionados.

Como esta solução busca acelerar o processo de particionamento na codificação de vídeo, ela poderá ser facilmente utilizada em conjunto com soluções de redução do custo computacional de outros processos do codificador, visando atingir níveis de redução de complexidade ainda mais altos. Os resultados de redução do custo computacional obtidos permitem facilitar a disseminação de vídeos de alta resolução em plataformas de distribuição de conteúdo multimídia e são de grande importância num cenário com resoluções, taxa de quadros por segundo e, consequentemente, tempos de processamento cada vez maiores.

1.2 Organização da Dissertação

A dissertação está organizada da seguinte forma. O capítulo 2 explica sobre o referencial teórico de vídeos, o VVC, aprendizado de máquina e trabalhos relacionados. O capítulo 3 apresenta uma análise dos novos particionamentos introduzidos no VVC e os seus impactos no desempenho do novo padrão de codificação. O capítulo 4 explica o processo de criação de um *dataset*, apresenta a etapa de treinamento dos modelos e a solução proposta nesta dissertação. O capítulo 5 apresenta e discute os resultados obtidos e mostra uma comparação com trabalhos relacionados. O capítulo 6 apresenta as conclusões e perspectivas de trabalhos futuros.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta o referencial teórico necessário para compreensão deste trabalho, com conceitos básicos sobre vídeos digitais, codificação de vídeo e aprendizado de máquina. Além disso, o capítulo apresenta os trabalhos relacionados mais importantes encontrados na literatura.

2.1 Conceitos Básicos de Vídeo Digital

Um vídeo é uma série de imagens sequencialmente representadas para o espectador com uma taxa de exibição de imagens por segundo suficiente que garanta uma suave transição visual. Todas as imagens individuais de um vídeo, mais conhecidas como quadros, possuem sua dimensão, tanto verticalmente como horizontalmente, medida em pixels. Pixel é o nome dado para a medida numérica de elementos dentro de uma imagem organizada de forma matricial.

Para propósitos de codificação, um quadro é normalmente dividido em blocos de dimensões *MxN* pixels, onde *M* é o número de linhas e *N* o número de colunas. Cada padrão de codificação define seus próprios tamanhos de blocos ou até mesmo uma variação de possíveis tamanhos. Isso é utilizado para facilitar a exploração de redundância de dados em áreas menores ao invés de se observar uma imagem por inteiro.

As dimensões de um quadro, também chamadas de resolução espacial, assumem diferentes tipos de proporções. Porém, existem alguns formatos predefinidos que são comumente usados nos nossos sistemas, como, por exemplo, *High Definition* (HD) 1024$\times$720 pixels, *Full High Definition* (FullHD) 1920$\times$1080 pixels e 4K, que podem ser representados por 4096$\times$2160 pixels ou 3840x2160 pixels. Quanto maior a resolução espacial, maior o número de pixels na imagem e consequentemente mais detalhada a percepção do conteúdo do vídeo.

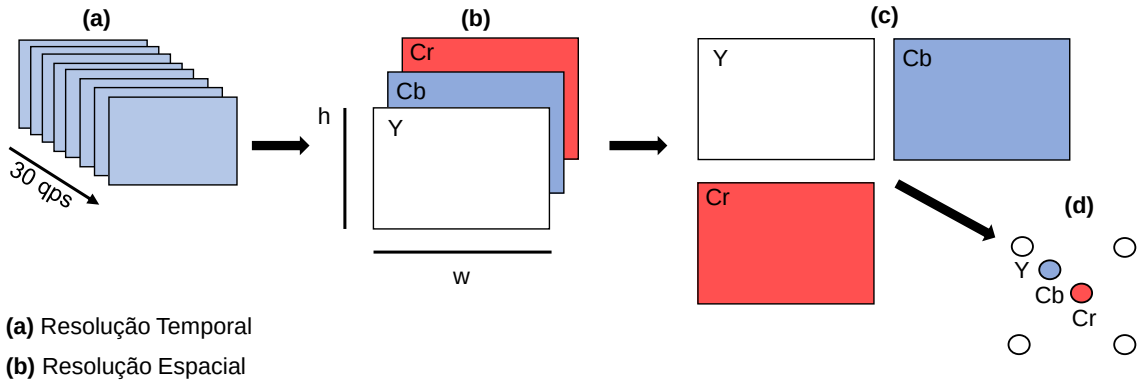
A taxa temporal, na qual os quadros do vídeo são apresentados, influenciam muito na qualidade de percepção visual. Quanto maior a quantidade de quadros por segundo apresentada, melhor e mais suave a percepção da mudança de movimento na

imagem. O número de quadros por segundo é chamado de resolução temporal. Normalmente esta resolução temporal varia entre 15 e 60 quadros por segundo, sendo necessários 24 quadros por segundo com uma resolução FullHD para que o ser humano tenha sensação de movimento real.

O sistema de cores e o padrão de subamostragem adotado pelo vídeo também é um importante aspecto a ser citado. Os sistemas de cores mais usados são o RGB (*Red, Green and Blue*) e o YCbCr (luminância, crominância azul e crominância vermelha). Milhares de cores podem ser representadas com a combinação de 3 canais. No RGB as cores são formadas por uma combinação dos componentes primários de vermelho, verde e azul. Porém o sistema visual humano é muito mais sensível a informações de luminância do que de cores, por isso o sistema YCbCr foi criado para se beneficiar desta característica (RICHARDSON, 2002).

O uso de subamostragem de imagens pode ser considerado por si só um tipo de compressão de vídeo, já que consiste em descartar parte da informação sem causar impactos perceptíveis na qualidade visual. As subamostragens mais comuns são 4:2:2, onde há 2 amostras de crominância azul e vermelha para cada 4 amostras de luminância, e 4:2:0, onde há apenas uma amostra de crominância azul e vermelha para cada 4 amostras de luminância (RICHARDSON, 2004).

A Figura 1 exemplifica de forma gráfica cada um dos conceitos anteriores: a resolução temporal (a), onde é definida a taxa de quadros por segundo apresentada ao usuário, a resolução espacial (b), onde se define o tamanho em pixels de cada quadro do vídeo, o espaço de cores (c), subdividido em Y (luminância), Cb (crominância azul) e Cr (crominância vermelha) e a quantidade de pixels de cada espaço de cor definido pela subamostragem do padrão (d).



(a) Resolução Temporal
(b) Resolução Espacial
(c) Espaço de Cores
(d) Subamostragem da Cor

Figura 1 – Resumo dos conceitos gerais de vídeos digitais

2.2 *Versatile Video Coding*

O padrão de codificação de vídeo *Versatile Video Coding* (VVC) foi lançado em Março de 2019 pela *Joint Collaborative Team on Video Coding* (JCT-VC), um projeto com a junção do ITU-T Video Coding Experts Group (VCEG) e ISO/IEC Moving Picture Experts Group (MPEG). O padrão foi planejado voltado para o crescimento da popularidade dos vídeos HD, FullHD e 4K, assim como vídeos para múltiplas vistas (JVET, 2020).

Atualmente, o padrão VVC consegue superar o seu concorrente direto HEVC em termos de eficiência de compressão, atingindo valores de compressão bem mais expressivos mantendo a qualidade da imagem. O VVC é capaz de reduzir o *bitrate* em até 50% se comparado ao seu antecessor. Isto é possível porque o VVC introduz uma série de novas ferramentas que melhoram a exploração de redundância de dados (SALDANHA et al., 2020a). A Figura 2 mostra um gráfico de comparação entre os padrões de codificação mais recentes em termos de taxa de bits (*bitrate*) e qualidade da imagem, medida em *Peak Signal-to-Noise Ratio* (PSNR) a qual é medida em decibéis (dB). Estas curvas trazem um valor em *BD-rate* onde quanto maior o valor, pior o resultado de eficiência de compressão

Consequentemente, quanto mais ferramentas e modos de operação são adicionados ao codificador, mais complexo ele se torna. Portanto, o VVC possui um custo computacional muito maior quando comparado ao HEVC e seus competidores, atingindo um custo computacional que pode ser de 21 até 31 vezes maior que o observado para o padrão HEVC (SALDANHA et al., 2020a).

Também vale ressaltar que grande parte deste custo computacional está associado

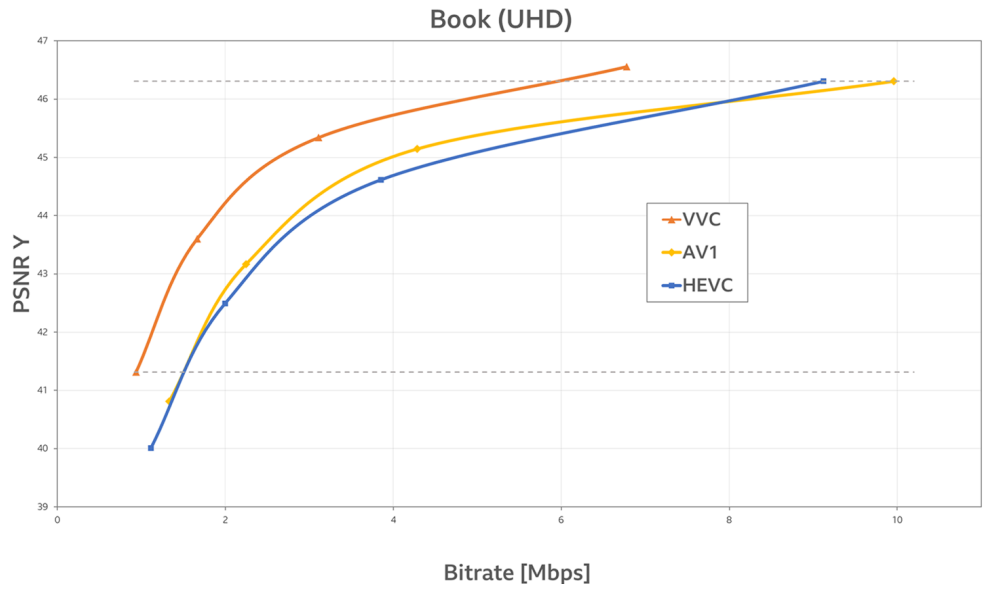


Figura 2 – Gráfico de comparação entre uma curva de quatro pontos de *BD-rate* para sequencias HD e UHD testadas em (JAMES BRUCE MARTA MRAK, 2019)

aos testes em diferentes tipos de particionamento adotados no VVC. Mais informações e conceitos a respeito dos dois tipos de predição existentes nos codificadores de vídeo serão abordados nas próximas seções deste trabalho. Também serão abordadas as estruturas de particionamentos adotadas pelo padrão HEVC.

2.2.1 Estruturas de Particionamento

O padrão VVC foi fortemente baseado no clássico esquema de particionamento de blocos, porém traz modificações significativas nas suas estruturas de particionamento de quadro. O VVC aumenta os níveis de particionamento em comparação ao seu antecessor e permite assim uma maior liberdade de escolha para um melhor particionamento, também permitindo que seja possível codificar de maneira mais eficiente vídeos com resoluções espaciais maiores, como 4K e 8K.

O padrão introduziu um esquema de particionamento de quadros muito mais flexível em comparação com seus antecessores, com o objetivo de permitir uma melhor eficiência de codificação para vários tipos de conteúdo e resolução. Inicialmente, cada quadro é particionado em várias estruturas quadradas chamadas *Coding Tree Units* (CTUs) de tamanho igual (tipicamente definido como 128x128). Em seguida, cada CTU pode ser particionada em várias *Coding Units* (CUs). As CUs podem apresentar vários tamanhos, podendo uma CTU inteira ser representada por apenas uma CU que engloba toda a região 128x128, até várias CUs 64x64, 32x32, 16x16, 8x8 e 4x4. O particionamento é realizado em um processo recursivo formando uma estrutura de divisão em árvore quaternária, binária ou ternária denominada *Multi Type Tree* (*MTT*), na qual cada CU é dividida em outras CUs, até que o tamanho mínimo de CU (4x4) seja atingido.

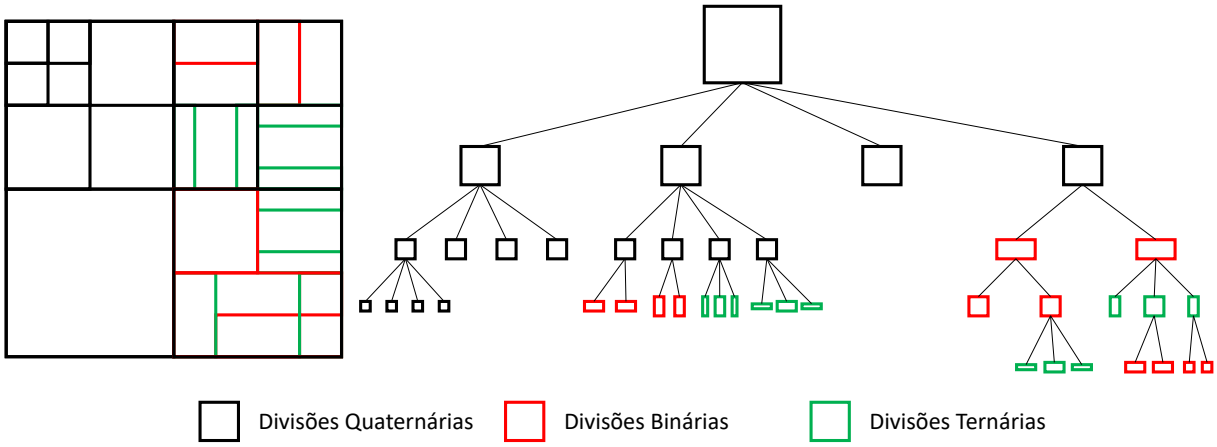


Figura 3 – Exemplo da estrutura da árvore MTT do VVC.

A Figura 3 mostra como a *MTT* inteira é formada para o cálculo recursivo das possibilidades de tamanhos de CU. No codificador VVC ótimo, ou seja, aquele codificador que atinge as melhores relações entre *bitrate* e qualidade de imagem através do uso

da estratégia de Otimização Taxa-Distorção (SULLIVAN; WIEGAND, 1998), o melhor particionamento para uma CTU é escolhido depois de testar todas as possibilidades da árvore quaternária, binária e ternária. Com base no que foi explicado até agora, é possível concluir que existem muitas combinações possíveis para codificar uma única CTU: além dos diferentes particionamentos possíveis, diferentes modos de predição e transformadas podem ser aplicados para cada bloco.

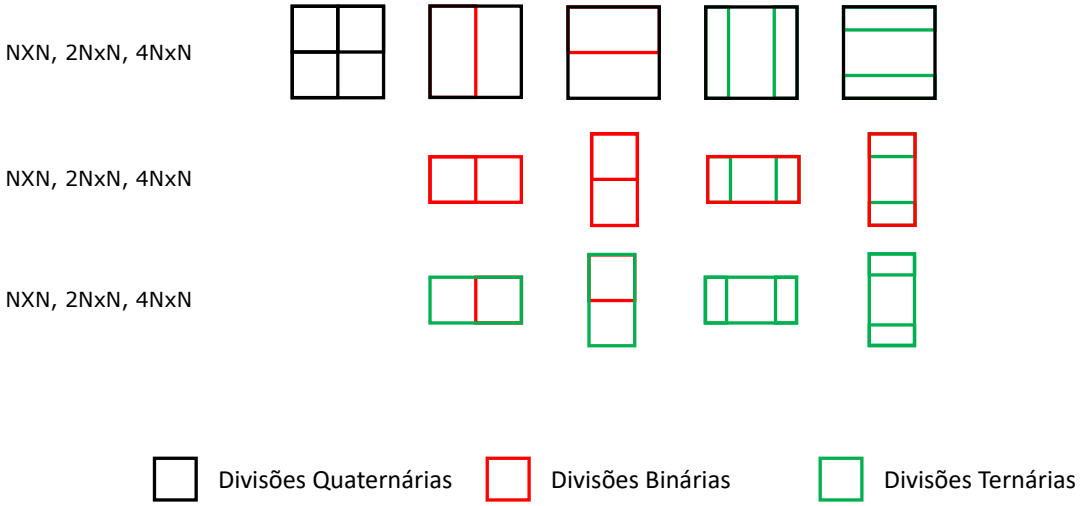


Figura 4 – Tipos de particionamentos aceitos pelo VVC.

A Figura 4 mostra todos os tipos possíveis de particionamento do VVC. Como apresentado na imagem, podemos particionar blocos quadrados em todos os tipos de particionamento, respeitando, é claro, os tamanhos mínimos de bloco. Porém, quando um bloco é particionado de forma binária ou ternária, a partir deste ponto ele só poderá ter novos particionamentos binários ou ternários. Assim, podemos ter particionamentos retangulares com tamanhos diversos, como 16x4, 8x32 e 16x8, por exemplo, abrangendo ainda mais as possibilidades de predições.

2.2.2 Predição Intra-Quadro

A predição intra-quadro é responsável por diminuir a redundância espacial predizendo amostras do bloco atual com os blocos vizinhos que já foram codificados no mesmo quadro. Esse modo de predição é responsável por compactar informações, explorando informações redundantes em um quadro, como regiões homogêneas ou padrões repetitivos.

O HEVC define 35 modos de predição intra responsáveis por predizer as informações de cada quadro, dos quais 33 são chamados de modos direcionais, úteis na repetição de padrões, como linhas diagonais ou retas. Os outros dois modos são chamados modos DC (que prevê que a média das amostras vizinhas é repetida em todo o bloco) e Planar (que repete informações de mais de uma borda, útil em padrões de gradiente).

A fim de minimizar os cálculos, uma heurística chamada *Rough Mode Decision* (RMD) é implementada no software HM (IECJCT1, 2014). No RMD, apenas um subconjunto dos 33 modos direcionais disponíveis é avaliado, em vez de testar todos eles. Depois que a melhor direção é selecionada, uma etapa de refinamento é aplicada, testando as direções angulares vizinhas.

No VVC os modos de predição intra angular do HEVC são estendidos de 33 para 65 para representar uma variedade de padrões de textura e alcançar maior precisão. Além disso, o VVC permite a predição intra de blocos retangulares devido aos formatos de CU obtidos com a estrutura de particionamento MTT. Neste caso, a predição intra VVC aplica a *Wide Angular Intra Prediction* (WAIP), uma vez que o ângulo dos 33 modos angulares convencionais foram definidos apenas para blocos quadrados. Assim, modos de predição intra com maior amplitude angular do que os modos angulares convencionais são permitidos para CUs retangulares. A Figura 5 representa os 67 modos de predição do VVC.

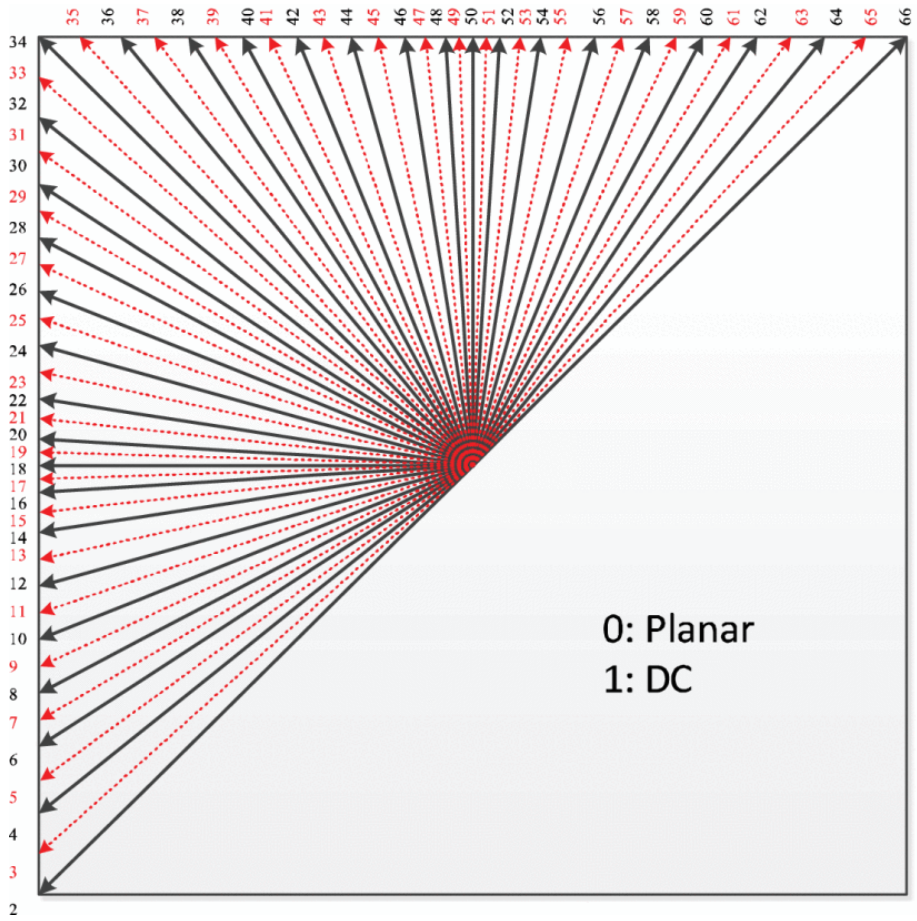


Figura 5 – Conjunto de 67 modos de predição intra-quadro: 65 modos direcionais, modo DC e modo planar

Como o HEVC, a predição intra do VVC emprega *Rough Mode Decision* (RMD) e

Most Probable Mode (MPM). O VVC divide o RMD em duas etapas (RMD-1 e RMD-2) para evitar uma avaliação exaustiva dos 67 modos de predição intra. O RMD-1 avalia os modos planar, DC e 33 angulares herdados do HEVC, por meio da Soma das Diferenças Transformadas Absolutas (SATD) entre o bloco original e o bloco predito e insere alguns modos com os valores de SATD mais baixos, ordenados, na lista RD. O RMD-2 executa uma etapa de refinamento para avaliar os SATDs dos modos angulares que são adjacentes aos modos angulares já incluídos na lista RD. Assim, um conjunto reduzido dos novos modos angulares VVC são avaliados para tentar encontrar uma predição mais precisa. O MPM obtém os modos padrão (os mais usados) e os modos nos blocos vizinhos à esquerda e acima codificados. A lista do MPM é composta por seis candidatos.

A predição intra *Multiple Reference Line* (MRL) (BROSS et al., 2018) permite o uso de mais linhas de referência, uma vez que, em alguns casos, as amostras de referência adjacentes podem diferir significativamente do bloco sob predição, levando a um erro de predição significativo. Além da linha de amostras diretamente adjacente, uma das duas linhas de referência não adjacentes, que são representadas na Figura 6, é usada como entrada para a predição intra MRL no VVC. O último uso de amostras de referência é conhecido como predição da MRL. Consequentemente, a codificação intra também avalia a predição do bloco usando amostras de referência mais distantes do que as amostras da linha e coluna diretamente adjacentes.

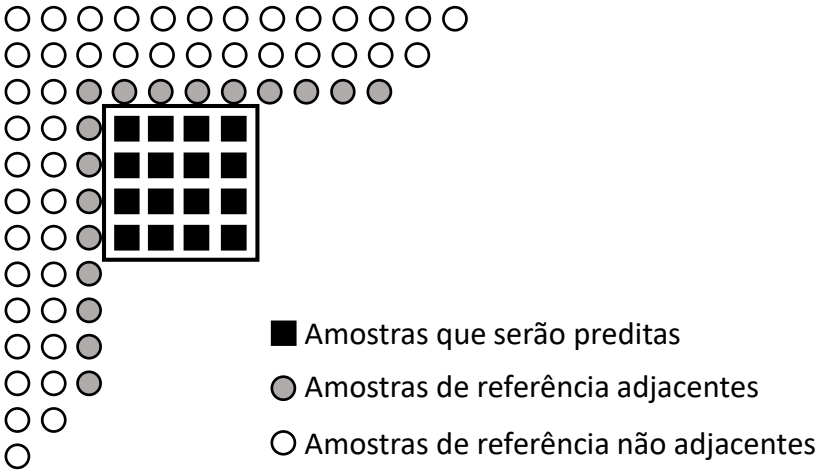


Figura 6 – Exemplo de predição intra MRL (BROSS et al., 2018)

A predição intra *Matrix-based Intra Prediction* (MIP) realiza a predição através da multiplicação de matrizes e interpolação de amostras, onde um conjunto de matrizes é avaliado de acordo com o tamanho do bloco e cada matriz representa um modo de predição. Os pesos dessas matrizes foram definidos por redes neurais, que foram

treinadas com um amplo conjunto de dados. A MIP melhora a eficiência da codificação permitindo predições que variam em mais de uma direção, o que não é possível com os modos angulares convencionais.

A predição *Intra Subpartition* (ISP) (RAMASUBRAMONIAN et al., 2019) melhora a eficiência de codificação explorando as correlações entre as amostras intrabloco. Para isso, o ISP divide o bloco atual, na vertical ou horizontal, em subpartições que são codificadas sequencialmente usando o mesmo modo de predição intra. Uma vez que uma subpartição foi codificada, suas amostras reconstruídas são usadas como amostras de referência para a próxima partição. Assim, mais amostras de referência correlacionadas são usadas para cada subpartição em comparação com a abordagem convencional, que localiza as amostras de referência nos limites esquerdo e acima do bloco atual.

A codificação residual abrange as etapas de transformação e quantização. Além do DCT-II, que tem sido empregado no HEVC, o VVC também inclui o *Multiple Transform Selection* (MTS) (ZHAO et al., 2021) para melhorar a eficiência da codificação residual. O MTS apresenta as matrizes de transformação primária DST-VII e DCT-VIII que podem ser aplicadas nas direções horizontal e vertical. Finalmente, a melhor combinação de transformada primária e secundária é escolhida por meio do menor (*Rate Distortion Cost* - RD-Cost) (HOANG; LONG; VITTER, 1998).

Isto introduz uma grande sobrecarga de computação, especialmente durante a predição, porque o RD-Cost de cada CU deve ser calculado durante a decisão da MTT da CTU. O processo de codificação completo, incluindo transformações e quantização é realizado muitas vezes apenas para decidir o melhor modo de predição. Portanto, reduzir o número de chamadas recursivas nas árvores quaternárias da CTU é uma maneira importante de reduzir o custo computacional geral (PFAFF et al., 2021).

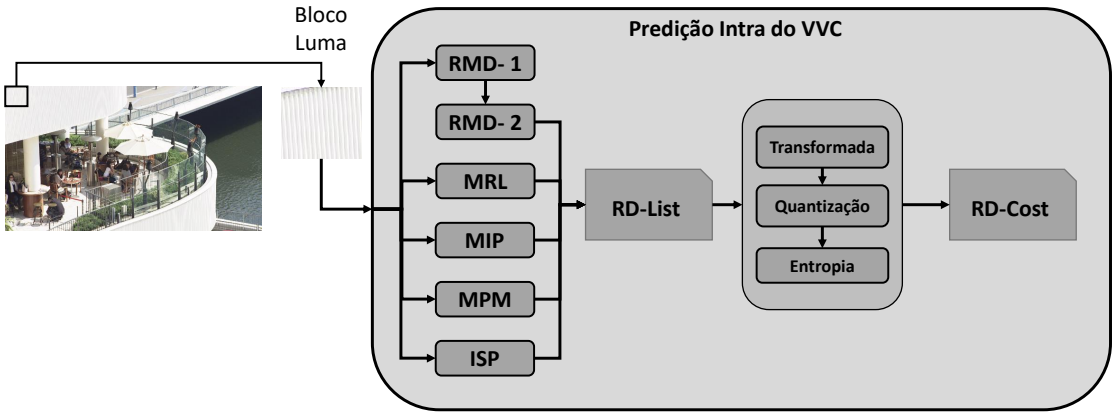


Figura 7 – Exemplo de uma predição intra para um bloco no padrão VVC

A Figura 7 exemplifica o fluxo da predição intra de um bloco no codificador VVC. A

predição passa pelas etapas citadas nesta seção para cada bloco dentro do quadro e cada tamanho e bloco têm o seu custo (RD-Cost) para definir como serão as partições.

2.2.3 Predição Inter-Quadros

A predição inter-quadros baseia-se na exploração de informações redundantes entre dois ou mais quadros em uma sequência. Altas taxas de quadros e ausência de movimento são duas situações típicas que aumentam as chances de alcançar uma boa compressão por meio da predição.

O processo de estimação de movimento (*Motion Estimation* - ME), realizado em blocos codificados com predição inter. Este processo é dividido em ME inteira (IME) e ME fracionária (FME). Durante a IME, o algoritmo de correspondência de bloco usa um ou mais quadros de referência para procurar a correspondência mais semelhante com o bloco atual. O mesmo processo é aplicado na FME, mas blocos com pixels fracionários, ou seja, interpolados em posições intermediárias a posições inteiras. A FME introduz vetores de movimento que representam movimentos menores que um pixel, o que aumenta a precisão da estimação de movimento. A Figura 8 apresenta um exemplo de quatro quadros dentro de um vídeo, onde três servem de referência para a predição inter-quadros entre os quadros.

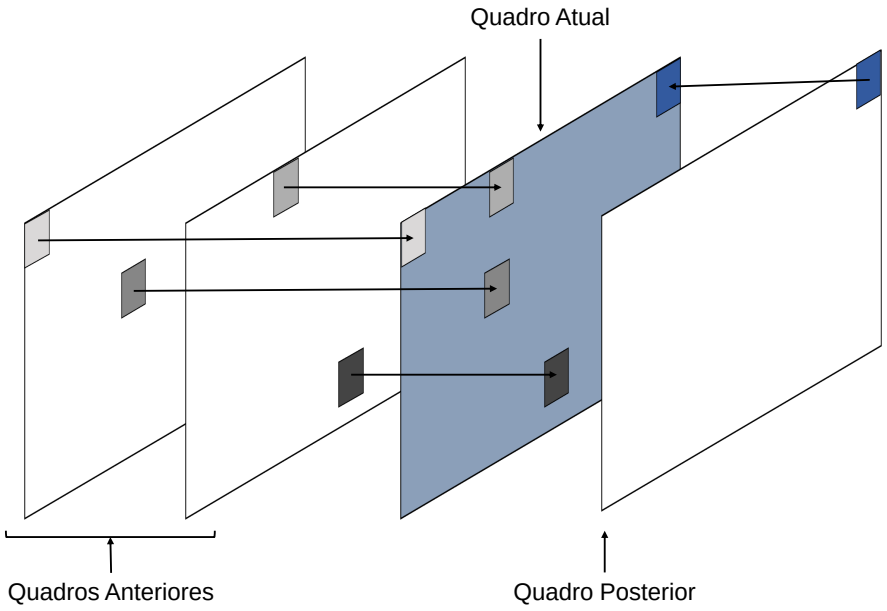


Figura 8 – Predição inter-quadros com múltiplos quadros de referência

As predições intra-quadro e inter-quadros possuem características próprias e são necessárias em diferentes tipos de aplicação. Serviços de *streaming*, por exemplo, necessitam criar segmentos curtos do vídeo que será transmitido, com o intuito de permitir ao usuário que solicite ao servidor os segmentos que melhor se adéquam às suas condições de rede, armazenamento e processamento local. Assim, cada um

desses segmentos deve conter pelo menos um quadro inicial codificado apenas com predição intra-quadro.

Porém, os quadros que são codificados com predição inter-quadros tendem a apresentar uma perda de qualidade significativa em relação a quadros puramente intra. Isto se deve ao fato da reutilização de dados provenientes dos quadros vizinhos e propagação de erros que causam perda de qualidade de um quadro para outro. Com isso, quadros inter possuem uma pior qualidade, apesar de reduzirem muito o *bitrate*. Quadros intra possuem um qualidade mais elevada, mas requerem um *bitrate* muito maior.

2.3 Aprendizado de Máquina

O aprendizado de máquina é um ramo da Inteligência Artificial que se ocupa do projeto de algoritmos capazes de obter conhecimento a partir de observações (RUSSEL; NORVIG, 2010). Em problemas computacionais, esse conhecimento é representado como um modelo que estima a saída de uma tarefa com base em alguns indicadores. Essa definição é intencionalmente ampla, porque mostra que o aprendizado de máquina pode ser aplicado em muitos casos, desde que os erros sejam permitidos em algum grau.

O conjunto de algoritmos possíveis a serem utilizados depende do tipo de treinamento (supervisionado, não supervisionado) e da saída desejada (classificação, regressão). O treinamento não supervisionado ocorre quando não há resultado previsto para as amostras. Nesse caso, as técnicas de agrupamento são aplicadas para separar exemplos em grupos e descobrir semelhanças ocultas entre eles. Este trabalho foca no aprendizado de máquina supervisionado com classificação que será explicado na subseção 2.3.1

O aprendizado de máquina está inserido em muitas ferramentas utilizadas na atualidade, desde pesquisas na web a filtragem de conteúdo em redes sociais, recomendações em sites de comércio eletrônico ou serviços de *streaming*. Também está cada vez mais presente em produtos de consumo, como câmeras e smartphones. Algoritmos de aprendizado de máquina são usados para identificar objetos em imagens, transcrição de falas em texto, postagens ou produtos com interesses em comum com os usuários assim como, seleciona resultados de pesquisa relevantes.

As técnicas convencionais de aprendizado de máquina são limitadas pela capacidade de processar dados naturais em sua forma bruta. Por décadas, a construção de um sistema de reconhecimento de padrões ou aprendizado de máquina exigia uma engenharia mais cuidadosa e uma considerável experiência de domínio para projetar um extrator de recursos que transforma os dados brutos em um classificador capaz de detectar ou classificar padrões na entrada. Devido à crescente quantidade e comple-

xidade dos dados processados na modelagem, modelos capazes de automatizar esse processo de extração de dados mais representativos passaram a ganhar importância. Esses modelos são construídos através de uma classe de técnicas chamadas aprendizado profundo (LECUN; BENGIO; HINTON, 2015). Essas técnicas se baseiam em arquiteturas de redes neurais complexas capazes de gerar representações latentes dos dados de entrada para gerar preditores mais eficientes.

O aprendizado profundo tem feito grandes avanços na resolução de problemas que resistiram às melhores tentativas da comunidade de inteligência artificial por muitos anos. Acabou performando de forma muito boa para aprender estruturas complexas com dados que possuem alta dimensão e, portanto, são aplicáveis a muitos domínios. Além disso, é usado com frequência para reconhecimento de imagem (KRIZHEVSKY; SUTSKEVER; HINTON, 2012a) e reconhecimento de voz (HINTON et al., 2012). Talvez mais surpreendentemente, o aprendizado profundo produziu resultados extremamente promissores para várias tarefas de compreensão de linguagem natural (COLLOBERT et al., 2011), classificação de tópicos, análise de sentimento, resposta a perguntas (BORDES; CHOPRA; WESTON, 2014) e tradução entre idiomas (JEAN et al., 2014).

2.3.1 Aprendizado Supervisionado em Redes Neurais

A forma mais comum de aprendizado de máquina é o aprendizado supervisionado. Neste modo, o modelo observa alguns exemplos de pares de entrada e saída, e aprende uma função que faz o mapeamento da entrada para a saída. Na prática, essas distinções nem sempre são tão nítidas. Mesmo os rótulos em si podem não ser os valores considerados verdadeiros que esperamos, por exemplo, quando temos imagens com rótulos errados que podem gerar ruído em modelos treinados.

O aprendizado supervisionado consiste em: dado um conjunto de treinamento com exemplos de entrada e saída, onde cada exemplo foi gerado por uma função desconhecida ou rotulado de alguma forma, descobrir uma função cuja saída se aproxime dos rótulos verdadeiros. Aprendizagem é uma busca através do espaço de hipóteses possíveis, mesmo em novos exemplos além do conjunto de treinamento. Para medir a precisão de uma hipótese, fornecemos um conjunto de testes de exemplos que são distintos do conjunto de treinamento. Dizemos que uma hipótese generaliza bem se prevê corretamente um rótulo para novos exemplos(RUSSEL; NORVIG, 2010).

Em alguns casos, a função é estocástica e o que temos de aprender é uma distribuição de probabilidade condicional. Quando a saída for de um conjunto finito de valores, como classes pré definidas, o problema da aprendizagem será chamado de classificação, e será chamado de classificação binária se houver apenas dois valores. Quando o valor da saída for um valor numérico, como por exemplo uma previsão do tempo da temperatura de amanhã, o problema de aprendizagem é chamado de re-

gressão. Neste trabalho, iremos tratar de problemas de classificação, pois o objetivo é prever o particionamento mais adequado dentro de um conjunto limitado de possibilidades.

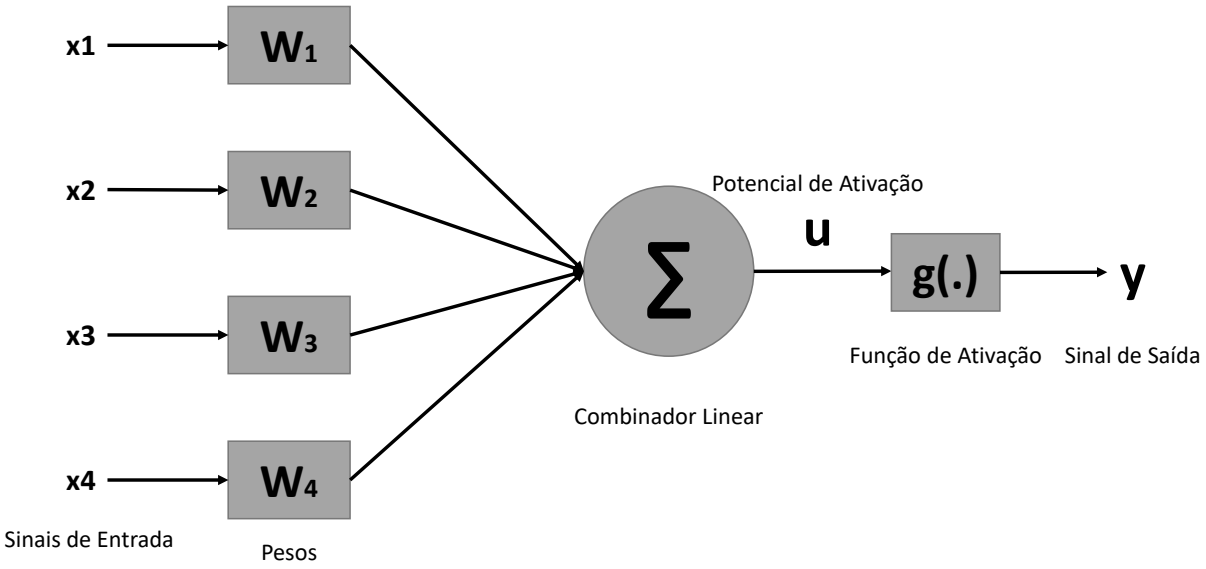


Figura 9 – Exemplo de um perceptron que recebe diversas entradas e calcula um valor de saída

Um exemplo de classificação binária simples é um *perceptron* como mostrado na Figura 9, o *perceptron* recebe uma quantidade n de entradas e reproduz uma saída de acordo com essas entradas através de uma função. Dentre as técnicas de classificação, temos as redes neurais, uma técnica extremamente versátil. Uma da característica deste campo da inteligência artificial é a capacidade de classificar imagens de diversas maneiras.

Para construir um modelo que pode classificar imagens, primeiro é necessário coletar um grande conjunto de dados de imagens categorizadas. Durante o treinamento, o modelo recebe um conjunto de imagens e produz uma saída na forma de um vetor de probabilidades, cada posição do vetor representa uma probabilidade para cada categoria.

Uma função que mede o erro entre os valores preditos e os valores reais é calculada seguindo alguma métrica de similaridade. Para problemas de regressão, costuma-se utilizar o erro médio absoluto (*mean absolute error* - MAE) ou quadrático (*mean squared error* - MSE).

O erro gerado é utilizado pelos algoritmos de treino, que ajustam os parâmetros dos modelos sendo treinados a fim de minimizar esse valor. Esses parâmetros ajustáveis, muitas vezes chamados de pesos, são números que definem a função de entrada-saída do modelo. Em um sistema típico de aprendizado profundo, pode haver centenas de milhões desses pesos ajustáveis e centenas de milhões de exemplos categorizados para treinar o modelo.

Para ajustar adequadamente o vetor de pesos, o algoritmo de aprendizado calcula um vetor gradiente que, para cada peso, indica por qual valor o erro aumentaria ou diminuiria caso o peso fosse aumentado em uma pequena quantidade. O vetor de peso é então ajustado.

Na prática, a maioria dos casos usa um procedimento chamado *stochastic gradient descent* (SGD). É chamado de estocástico porque cada pequeno conjunto de exemplos fornece uma estimativa ruidosa do gradiente médio de todos os exemplos. Este procedimento simples geralmente encontra um bom conjunto de pesos surpreendentemente rápido quando comparado com técnicas de otimização muito mais elaboradas (BOTTOU, 2010). Após o treinamento, o desempenho do sistema é medido em um conjunto diferente de exemplos, denominado conjunto de teste. Isso serve para testar a capacidade de generalização do modelo, sua capacidade de produzir respostas sobre novas entradas que nunca viu durante o treinamento.

Muitas das aplicações práticas atuais do uso de aprendizado de máquina usam classificadores lineares além de recursos de engenharia manual. Um classificador linear de duas classes calcula uma soma ponderada dos componentes do vetor de recursos. Se a soma ponderada estiver acima de um limite, a entrada é classificada como pertencente a uma determinada categoria.

Sabemos que os classificadores lineares só podem modelar seu espaço de entrada em regiões muito simples (DUDA; HART et al., 1973). Mas problemas como reconhecimento de imagem e voz exigem que a função de entrada-saída seja insensível a irrelevantes variações da entrada, como variações na posição, orientação ou iluminação de um objeto. Para tornar os classificadores mais robustos, a opção convencional é projetar extratores de boas características, o que requer uma quantidade considerável de habilidade e conhecimento de domínio. Mas isso tudo pode ser evitado se bons recursos puderem ser aprendidos automaticamente usando um procedimento de aprendizagem de propósito geral. Esta é a principal vantagem da aprendizagem profunda.

Uma arquitetura de aprendizado profundo é projetada como uma pilha de várias camadas de módulos simples, todos (ou a maioria) sujeitos a aprendizagem e muitos dos quais computam mapeamentos de entrada-saída não lineares. Uma maneira de obter invariância é normalizar os dados, ou girar os eixos da imagem de entrada.

2.3.2 *Backpropagation* para Treinar Arquiteturas de Múltiplas Camadas

Desde o princípio dos algoritmos de reconhecimento de padrões, o objetivo foi substituir recursos de engenharia manual por recursos treináveis por redes de múltiplas camadas, mas apesar de sua simplicidade, a solução não foi amplamente compreendida até meados dos anos 1980. Acontece que as arquiteturas de múltiplas camadas podem ser treinadas por um simples SGD. Contanto que os módulos sejam

funções relativamente suaves de suas entradas e de seus pesos internos, pode-se calcular gradientes usando o procedimento de *Backpropagation* (RUMELHART; HINTON; WILLIAMS, 1986).

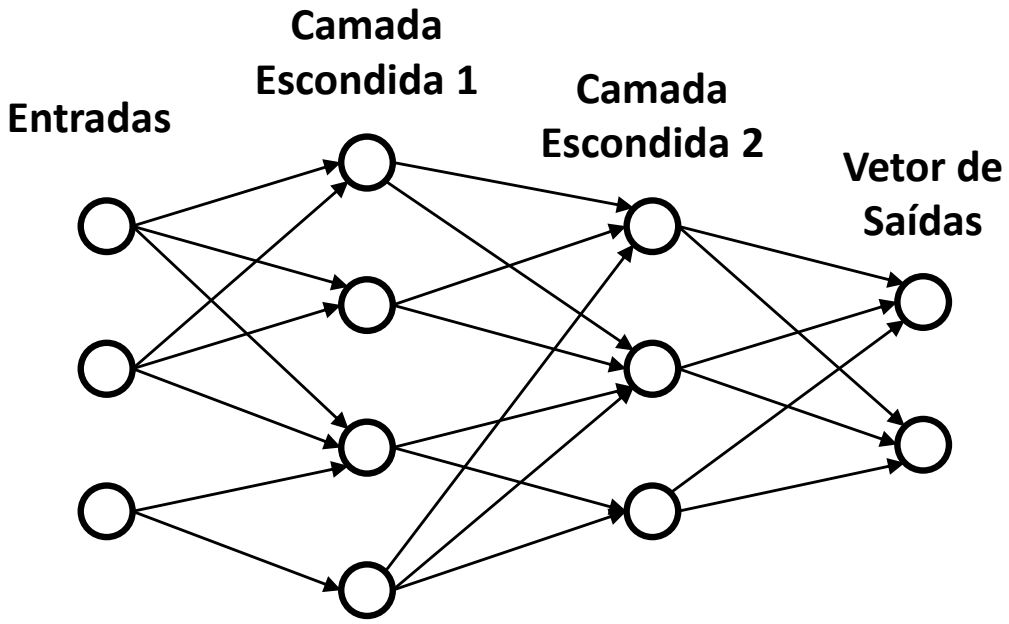


Figura 10 – Rede neural com múltiplas camadas

Muitas aplicações de aprendizagem profunda usam arquiteturas de rede neural *feedforward*, exemplificadas na Figura 10, que aprendem a mapear uma entrada de tamanho fixo (por exemplo, uma imagem) para uma saída de tamanho fixo (por exemplo, um vetor de probabilidade para cada uma das várias categorias). Para ir de uma camada para outra, um conjunto de unidades calcula uma soma ponderada de suas entradas (no caso, o resultado da camada anterior) e passa o resultado por uma função não linear. A função *Rectified Linear Unit* (ReLU) normalmente auxilia a rede a aprender muito mais rápido para redes com muitas camadas (GLOROT; BORDES; BENGIO, 2011). Unidades que não são as camadas de entrada ou saída são convencionalmente chamadas de camadas ocultas. As camadas ocultas podem ser vistas como uma maneira de distorcer a entrada de uma forma não linear de modo que as categorias se tornam linearmente separáveis pela última camada, como na Figura 10.

O interesse em redes *feedforward* profundas foi reavivado por um grupo de pesquisadores reunidos pelo Instituto Canadense de Pesquisa Avançada (CIFAR) (HINTON; OSINDERO; TEH, 2006). Os pesquisadores introduziram procedimentos de aprendizagem não supervisionados que poderiam criar camadas que apresentam detectores sem a necessidade de dados rotulados chamados de "pré-treinamento". Uma camada final pode então ser adicionada ao topo da rede e todo o sistema profundo pode ser ajustado usando *Backpropagation*. Isso funcionou muito bem para reconhecer dígitos ou para detectar pessoas, especialmente quando a quantidade de dados rotulados

eram muito limitados (SERMANET et al., 2013).

Este tipo de técnica foi possível com o advento das *Graphics Processing Unit* (GPUs) que eram convenientes para treinar redes 10 ou 20 vezes mais rápido (RAINA; MADHAVAN; NG, 2009). Para conjuntos de dados menores, o pré-treinamento não supervisionado ajuda a prevenir *overfitting* (BENGIO; COURVILLE; VINCENT, 2013), levando a generalização significativamente melhor quando o número de exemplos rotulados é pequeno.

2.3.3 Redes Neurais Convolucionais

As Redes Neurais Convolucionais (CNNs) são projetadas para explorar a redundância espacial entre os dados de entrada. Isso acontece, por exemplo, com imagens coloridas compostas de três matrizes 2D, cada uma contendo intensidades de pixel para cada canal de cor (RGB ou YCbCr). As CNNs são um tipo particular de rede profunda que é muito mais simples de treinar e generalizar do que redes com conectividade total entre camadas adjacentes. As CNN alcançaram muito sucesso prático e recentemente foram amplamente adotadas pela comunidade. Um exemplo de CNN está apresentado na Figura 11, que representa a arquitetura da Resnet18 (utilizada neste trabalho, como explicado posteriormente).

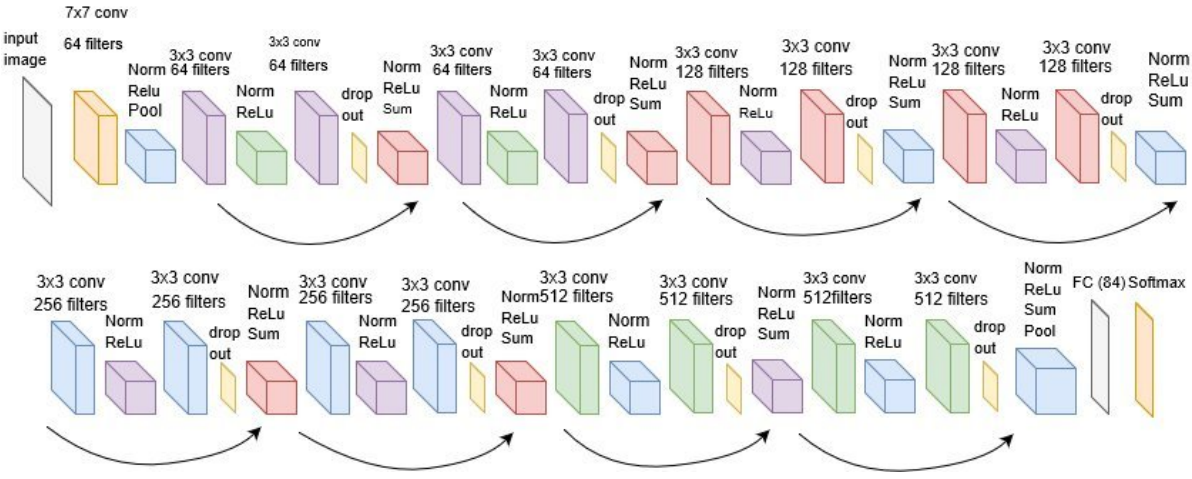


Figura 11 – Exemplo de CNN: arquitetura da Resnet18 (AL RABBANI ALIF; AHMED; HASAN, 2017)

As unidades em uma camada convolucional são organizadas em mapas de características, dentro dos quais cada unidade está conectado a *patches* locais da camada anterior através de um conjunto de pesos denominados filtros. O resultado disso, uma soma ponderada local, é então passado por uma função de ativação, como a ReLU. Podemos ter etapas de *Batch Normalization* onde é feita a normalização das entradas das camadas, centralizando e redimensionando. Também temos etapas de *dropout* que se refere ao termo "abandonar"aleatoriamente, ou omitir, unidades (ocultas ou

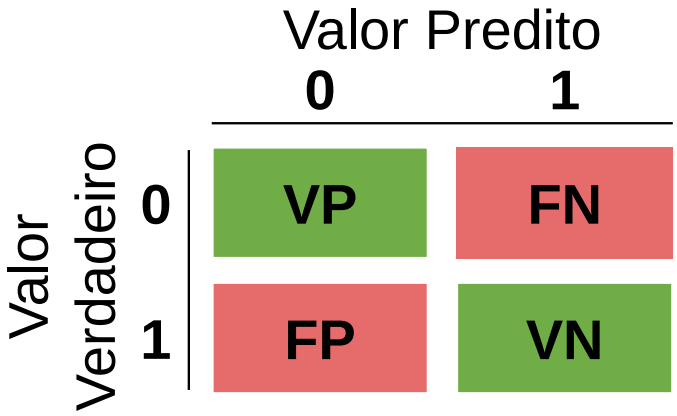


Figura 12 – Exemplo de uma matriz de confusão e o que representa cada posição

não) durante o processo de treinamento de uma rede neural como um tipo de regularização (DAHL; SAINATH; HINTON, 2013).

Matematicamente, a operação de filtragem realizada por um mapa de características é uma convolução discreta, que dá o nome a este tipo de rede neural. O papel da camada convolucional é detectar conjunções locais de recursos da camada anterior, já o papel da camada *pooling* é mesclar recursos semanticamente semelhantes (LECUN; BENGIO; HINTON, 2015).

As redes neurais convolucionais tornaram-se recentemente o modelo padrão em vários algoritmos baseados em problemas de visão computacional. O uso das redes convolucionais usadas pela comunidade de visão computacional têm crescido cada vez mais profundamente a cada ano desde o surgimento da AlexNet em 2012 (KRIZHEVSKY; SUTSKEVER; HINTON, 2012b). Uma das redes mais profundas na literatura é uma rede residual (ResNet) (HE et al., 2016) com 1.202 camadas treináveis. Essas redes foram treinadas usando o conjunto de dados de classificação ImageNet. Por outro lado, mostraram que é possível treinar redes muito mais rasas, porém mais amplas (ZAGORUYKO; KOMODAKIS, 2016).

2.3.4 Métricas de Avaliação de Modelos de Aprendizado de Máquina

Uma forma eficaz de avaliar o desempenho de modelos é através da matriz de confusão, a qual permite avaliar os erros e acertos para cada tipo de classe do problema. Cada linha da matriz representa os valores reais observados, enquanto que cada coluna representa os valores obtidos pelo algoritmo de classificação (ou vice-versa) (STEHMAN, 1997). Cada combinação (valor esperado, valor predito) é uma posição na tabela de contingência, como mostra a Figura 12.

Na Figura 12, as células destacadas em verde representam os Verdadeiros Positivos (VP), que são os valores positivos preditos corretamente, assim como os Verdadeiros Negativos (VN), que são os casos em que o valor predito e o valor observados

são ambos negativos. De forma análoga, os valores destacados em vermelho são os Falsos Positivos (FP), que ocorrem quando o valor esperado é negativo e o valor predito é positivo, assim como o os Falsos Negativos (FN), que ocorrem na situação inversa.

A partir da matriz de confusão é possível calcular algumas informações e para isso existem métricas responsáveis para classificar o quanto um modelo é bom para determinada tarefa. Com esses valores, várias medidas de desempenho podem ser calculadas, dentre as quais as mais comuns são:

- **Acurácia:** calculada conforme a equação (1), é a medida de desempenho mais intuitiva e representa uma proporção da observação corretamente prevista para o total de observações. É recomendada quando o conjuntos de dados é simétrico, ou seja, quando o número de exemplos positivos e negativos é similar.

$$Acuracia = \frac{(VP) + (VN)}{Total} \tag{1}$$

- **Precisão:** calculada conforme a equação (2), é a razão de observações positivas previstas corretamente para o total de observações positivas previstas. A métrica corresponde à proporção de todos os rótulos classificados como uma classe que realmente são desta classe. Uma alta precisão está relacionada à baixa taxa de falsos positivos.

$$Precisao = \frac{(VP)}{(VP) + (FP)} \tag{2}$$

- ***Recall*:** calculada conforme a equação (3), é a razão de observações positivas corretamente previstas para todas as observações na classe . A métrica indica, de todos instâncias com um rótulo no *dataset*, quantas realmente foram classificadas com o valor correto. O *recall* é a frequência com que o classificador encontra os exemplos de uma classe corretamente.

$$Recall = \frac{(VP)}{(VP) + (FN)} \tag{3}$$

- ***F1-score*:** calculada conforme a equação (4), considera tanto a precisão como o *recall* do teste para computar a pontuação. O *F1-score* é a média harmônica da precisão e do *recall*, em que uma pontuação de F1 atinge seu melhor valor em 1 (precisão e *recall* perfeitas) e o pior em 0. O *F1-score* é uma medida global de utilidade que se aplica a um conjunto de dados de avaliação como um todo

(JANSCHE, 2005).

$$F1 = \frac{2 \times Precisao \times Recall}{Precisao + Recall} \tag{4}$$

2.4 Trabalhos Relacionados

Por ser um padrão de codificação que se tornou estado da arte recentemente, o VVC possui uma constante evolução em comparação com seus antecessores. Também se deve levar em consideração que o VVC é uma evolução natural, com mais ferramentas, do HEVC, com isso é comum se esperar que trabalhos que obtiveram sucesso no HEVC também tenham bom desempenho no VVC. Por isso, neste capítulo, vamos focar em alguns temas principais, sendo eles: trabalhos que comparam os dois codificadores, algoritmos que buscam melhorias em ferramentas específicas dentro da predição intra e trabalhos que visam uma abordagem de redução do custo computacional em alto nível, focando nos particionamentos novos do VVC.

2.4.1 Visão Geral e Comparação Entre Codificadores

Por se tratar de um padrão de codificação recente, é possível ainda encontrar na literatura uma série de trabalhos recentes que buscam estudar o comportamento do VVC, e levantar estatísticas a respeito das etapas de codificação ou das ferramentas implementadas em cada módulo. Como o VVC herdou muitas particularidades do HEVC e também adicionou muitas ferramentas novas, que acarretam no aumento do custo computacional do codificador, soluções baseadas em redução do uso dessas ferramentas são comuns. Muitos trabalhos, também, visam explorar estas novas ferramentas, principalmente para avaliar o real uso da ferramenta dentro do codificador e o quanto a supressão desta ferramenta impacta na eficiência de codificação.

MANSRI et al. (2020) faz uma extensa avaliação da eficiência de codificação dos mais importantes padrões de codificação atuais. Além disso, os autores também incluem nas comparações algumas implementações otimizadas de codificadores já existentes, tais como o x264 e o x265, ambas implementações otimizadas dos codificadores H.264/AVC e H.265/HEVC, respectivamente. Nos resultados experimentais apresentados, é possível visualizar a superioridade do VVC em relação aos demais codificadores no quesito eficiência de codificação. Em contrapartida, os autores apresentam resultados quanto ao tempo de codificação do VVC em relação aos demais codificadores, que, dependendo da configuração escolhida, pode alcançar 15 vezes o tempo de codificação do HEVC.

Em (TISSIER et al., 2019), os autores trazem uma avaliação do VVC e trazem uma discussão sobre possíveis oportunidades que podem auxiliar na redução do custo computacional no VVC. O trabalho visa explorar similaridades do VVC com o HEVC

e também as novas ferramentas implementadas no VVC e faz uma abordagem sobre limites de redução do custo computacional para cada uma delas.

Em (PFAFF et al., 2021), os autores fazem um extenso estudo sobre todas as ferramentas da predição intra do VVC, assim como abordam os tipos de particionamento de blocos. Além disso, o trabalho ainda apresenta uma comparação em termos de eficiência de codificação do VVC comparando ele ao HEVC. Um ponto relevante é que o trabalho usa uma versão bem mais atual do software de referência do que a maioria dos trabalhos encontrados na literatura. Segundo os autores, o desempenho de compressão do VVC em relação a HEVC é de quase 25% de melhoria no *BD-rate* (BJONTEGAARD, 2001) para a configuração *AllIntra*. O VVC tem maior ganho em eficiência de codificação para altas resoluções, como sequências UHD. No geral, quando comparado com HEVC, há um benefício claramente significativo na eficiência de codificação para o VVC sobre todas as classes e tipos de sequências.

Os autores em (PFAFF et al., 2021) também fazem uma análise separada para cada ferramenta nova introduzida no VVC. A ferramenta *Cross Component Linear Model* (CCLM) fornece uma melhoria considerável na eficiência de codificação: redução de 1,5% em *BD-rate* para luminância e 14% para componentes de crominância. Os testes das ferramentas MIP, PDPC, ISP, MRL, apresentam os ganhos menos expressivos. Finalmente, os novos particionamentos fornecem um ganho de até 25% na eficiência de codificação.

Outra contribuição importante é a de CERVEIRA et al. (2020). Este trabalho apresenta um *profiling* de memória para o fluxo de codificação do VVC. O objetivo é analisar o impacto do uso de memória de cada módulo de codificação e destacar a influência das novas ferramentas de codificação introduzidas no VVC. Esse *profiling* é realizado por meio de dois conjuntos de experimentos: uma análise geral dos requisitos de memória, exibindo a distribuição dos acessos por módulo de codificação e uma avaliação específica das etapas de predição. Como resultados, é observado que: as etapas de predição realizam as operações que mais exigem acesso à memória no processo de codificação, representando de 47% a 58% de todos os acessos; as novidades do VVC em operações de transformações aumentam o overhead de memória em 20%, em média.

Por fim, SALDANHA et al. (2020a) traz uma visão geral à respeito dos trabalhos já publicados sobre projetos de hardware dedicados para a codificação de vídeo no padrão VVC. Com base em oito trabalhos publicados, os autores concluem que o principal desafio para esta geração de codificadores consiste em implementar as novas ferramentas em codificadores com demanda de alta vazão, como em aplicações em tempo real, e a codificação de vídeos com resoluções maiores.

2.4.2 Algoritmos para Ferramentas Específicas

Com a vasta adição de novas ferramentas no codificador VVC, existe um campo muito amplo de trabalhos que podem explorar cada ferramenta individualmente. Também graças ao sucesso do codificador em conseguir uma eficiência de compressão muito superior aos outros, essas ferramentas precisam ter uma atenção a mais, pois ao serem modificadas de alguma maneira, em busca de uma redução no custo computacional, podem ocasionar em um perda considerável na eficiência de compressão. Por isso alguns trabalhos focam em apenas estudar um tipo de ferramenta específica e de alguma forma trazer algum ganho em custo computacional sem afetar drasticamente a eficiência de compressão ou, por outro lado, buscar algum ganho em eficiência de compressão sem afetar muito o custo computacional do codificador que já é de grande escala.

O artigo (DE-LUXáN-HERNáNDEZ et al., 2019) propõe o uso do algoritmo *Intra Subpartition* (ISP). O ISP é uma versão atualizada do modo *Line-Based Intra Prediction* (LIP) que melhora o equilíbrio entre o ganho de codificação e o custo computacional do método original. O princípio básico do ISP consiste em subdividir um bloco intra predito em 2 ou 4 subpartições de pelo menos 16 amostras de acordo com as dimensões originais do bloco. O trabalho teve como resultado a obtenção de um ganho de 0,57% em eficiência de codificação para um aumento no tempo de codificação de 12% para a configuração All Intra e um ganho de 0,29% em eficiência de codificação com um aumento no tempo de codificação de 2%, para a configuração de *Random Access*. Este modo foi integrado posteriormente no software de referência, como mencionado na seção 2.2.2 e mencionado no documento (RAMASUBRAMONIAN et al., 2019).

Em (DONG et al., 2021), para a seleção de modo e finalização de predição, é proposto um algoritmo de decisão de modo intra rápido. O algoritmo consiste no *Adaptive Mode Pruning* (AMP) e o *Mode-Dependent Termination* (MDT). O AMP remove modos não promissores dos modos recém-introduzidos (IBC e ISP) e os modos normais durante a seleção. O MDT elimina a previsão desnecessária dos níveis restantes de profundidade com base no modo ideal selecionado. Resultados experimentais mostraram que essa abordagem pode reduzir em média o tempo de codificação em 51% com 1,08% de aumento em *BD-rate*.

Por fim, (NASIRI et al., 2020) explora o uso Redes Neurais Convolucionais (CNN) para melhorar a qualidade de quadros codificados no VVC após a decodificação. A principal contribuição deste trabalho é o uso de informações da codificação do *bitstream* compactado. Mais precisamente, a informação de predição intra é usada para treinar a rede, além da informação de reconstrução. O método proposto é aplicado em componentes de luminância e crominância de quadros intra. O estudo mostra que, tanto em taxas de bits baixas quanto altas, o uso de informações das predições podem melhorar o desempenho do *BD-rate* entre 1% e 6% para luma e croma.

2.4.3 Algoritmos de Particionamento de Blocos

Uma das otimizações mais exploradas em um codificador de vídeo para redução do esforço computacional é a simplificação, de maneira inteligente, dos níveis de particionamento de blocos, uma vez que para cada possível tamanho de bloco, é necessário executar diversas etapas complexas do codificador, como as etapas de predição, por exemplo. Além disso, a nova estrutura de particionamento inserida no VVC (MTT) contribuiu ainda mais para o acréscimo em esforço computacional. Neste sentido, alguns trabalhos já propõem melhorias neste quesito para o VVC, uma vez que é possível alcançar grandes taxas de redução de esforço computacional alterando somente uma etapa de codificação.

Em FU et al. (2019), é proposto um particionamento de CU mais rápido, com base em um modelo de *Random Forests Classifier* e um algoritmo de decisão de modo intra rápido com base em características de região de textura. Os resultados da simulação indicam que o esquema proposto pode economizar 54,91% do tempo de codificação com apenas 0,93% de aumento no *BD-rate*.

O trabalho SALDANHA et al. (2020b) propõe um novo esquema de decisão de particionamento rápido para codificação intra VVC para reduzir o custo computacional da estrutura MTT. Este esquema é composto por duas estratégias que exploram a correlação dos modos de predição intra e amostras da CU atual para decidir a direção de divisão de partições binárias e ternárias. Com base nessas informações, o esquema pode evitar avaliações desnecessárias de partições binárias e ternárias, reduzindo o tempo de codificação em 31,4% com perda na eficiência de compressão em *BD-rate* de 0,98% para a configuração *AllIntra*.

Em (LI et al., 2021) é proposta uma abordagem de *Deep Learning* para predizer a partição da CU baseada na MTT, acelerando drasticamente o processo de codificação intra do VVC. Em conjunto, foi projetada uma função de perda adaptativa para treinar o modelo da CNN, sintetizando o número incerto de modos de divisão e a meta do RD-Cost. Os resultados experimentais demonstram que esta abordagem pode reduzir o tempo de codificação do VVC entre 44,65% e 66,88% com uma perda de eficiência (em *BD-rate*) entre 1,322% e 3,188%.

No trabalho (TISSIER et al., 2020), é proposta uma técnica eficiente de redução do custo computacional baseada em CNN para o VVC. A CNN é alimentada com uma CU do bloco de luminância com tamanho de 64x64 pixels e cria um vetor que dá probabilidades de ter bordas nos limites do bloco 4×4. Este vetor de probabilidade é posteriormente explorado pelo codificador para pular divisões improváveis. Este trabalho alcança uma redução de tempo de codificação de até 51,5% para uma perda na eficiência de compressão de 1,45%.

Como pode ser visto, apesar de novo, o VVC atrai atenção de diversos pesquisadores que exploram suas diversas novidades em relação a outros codificadores. Por

ser um novo estado da arte é comum encontrar trabalhos que buscam objetivos próximos, ou até mesmo que sejam complementares sem de fato terem sido testados em conjunto. Também se pode pensar que cada nova ferramenta que leva a uma melhora na eficiência de codificação introduzida no novo padrão abre espaço para pelo menos uma possível solução que busca uma melhora no tempo de codificação. Com isso, podemos concluir que a implementação em software de codificadores VVC ainda pode ser muito otimizada e muitos trabalhos ainda podem surgir para contribuir com possíveis soluções.

3 ANÁLISES DOS PARTICIONAMENTOS DO VVC

Conforme explicado na subseção 2.2.1, o codificador de referência VVC avalia diversas possibilidades de particionamento de quadros. Vários cálculos são feitos nesse extenso processo, mas isso é importante para manter uma alta eficiência de codificação. Este capítulo apresenta uma análise estatística de ocorrências entre os particionamentos de *Coding Units*. Inicialmente, apresenta-se a metodologia empregada nos experimentos. Em seguida, a análise propriamente dita é apresentada.

3.1 Metodologia dos Experimentos

Todos os testes, análises e soluções propostas nesta dissertação foram desenvolvidos utilizando a versão 6.0 do codificador e decodificador VVC de referência, o *VTM Model* (JVET, 2020). Para obter os dados necessários para uma análise estatística e treinamento dos modelos foi necessário modificar tanto o codificador quanto o decodificador VVC para coletar informações de cada CU, que serviram posteriormente para as análises preliminares.

Foram usados para as análises cinco vídeos entre os indicados nas Condições Comuns de Testes (CTC) (F. BOSSEN J. BOYCE, 2019) do VVC. Os vídeos estão listados na Tabela 1 e englobam todas as resoluções acima de HD (*High Definition*). Todos os vídeos são codificados utilizando um parâmetro de quantização (*Quantization Parameter* - QP) nos valores de 22, 27, 32 e 37.

Tabela 1 – Configuração dos experimentos para análise de ocorrência de particionamento de CUs

Codec	VTM Model 6.0
Configuração	*All Intra*
QPs*	22, 27, 32, 37
Sequências de análises	*Cactus*, *PartyScene*, *BasketballPass*, *FourPeople*, *SlideEditing*

3.2 Análises dos Particionamentos do VVC

Como explicado anteriormente, os novos modos de particionamentos adicionados no VVC são uma das maiores ferramentas adicionadas ao novo padrão de codificação, tanto em termos de eficiência de codificação, quanto em complexidade computacional. Estes novos particionamentos conseguem extrair uma gama muito maior de possibilidades para o codificador, o que não era possível antes. Com isso é possível atingir números bem mais expressivos em eficiência de compressão. Porém, a carga computacional que estes particionamentos demandam é potencialmente elevada, tanto que muitos autores buscam uma abordagem mais conservadora que ainda assim não afete tanto a eficiência que eles trazem.

Como a quantidade de possíveis particionamentos dentro de uma CTU 128×128 é muito grande, o custo computacional é muito grande para conseguir encontrar um particionamento ideal. As CTUs podendo ser divididas tanto em blocos quadrados quanto retangulares acabam trazendo muitas possibilidades de divisões. No entanto, é difícil verificar se todos estes particionamentos candidatos serão úteis para o bloco antes de testá-los.

Também é necessário analisar a real utilidade dos dois principais novos particionamentos que são os binários e ternários, horizontais e verticais. Com isso em vista, uma análise dos particionamentos do VVC foi elaborada. Nesta análise se buscou observar se os novos tamanhos de bloco são realmente escolhidos com frequência. Para isso codificamos o conjunto de vídeos mencionados na Tabela 1 com as configurações descritas. Após, foi verificado o tamanho final de cada bloco dentro do vídeo tanto para codificação *AllIntra* e *RandomAccess*. Os resultados obtidos estão demonstrados nos mapas de calor da Figura 13.

A primeira nota importante a se mencionar sobre o mapa de calor é que na Codificação Intra o tamanho máximo de um bloco é de 64x64 pixels. Também vale ressaltar que não existem variações retangulares para Intra quando o tamanho no eixo X ou no eixo Y é igual a 64 pixels. Por isso, quando observamos no gráfico não encontramos nenhuma ocorrência de blocos com tamanho 64×32, 32×64, 64×16 e etc.

Dito isso, podemos notar que todos os tamanhos possíveis de bloco são utilizados, mesmo que numa proporção menor. O mapa de calor nos mostra a porcentagem de ocorrência de cada tamanho de bloco. É possível observar uma maior ocorrência de blocos quadrados. Também é possível perceber que quanto maior a diferença entre os tamanhos dos eixos, mais rara é a chance deste bloco ser escolhido. Isso se dá pelo fato de que blocos retangulares como 32×4 pixels exigem uma característica muito especifica de informação na imagem para que esta seja a melhor escolha, mesmo assim, este tipo de particionamento ainda é escolhido com certa frequência.

Por outro lado, percebemos que tamanhos diretamente adjacentes aos quadrados

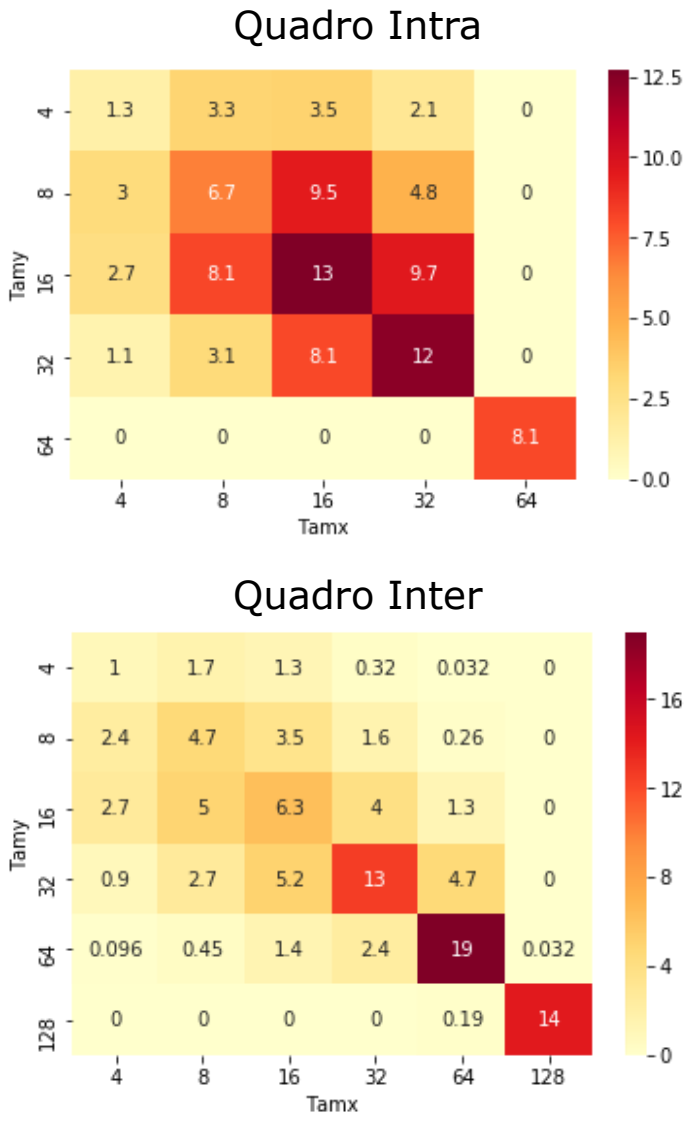


Figura 13 – Mapas de calor apresentando a porcentagem de ocorrência, de cada tamanho de bloco para quadros Intra e Inter, normalizados por tamanho de bloco. Tamy sendo o tamanho do bloco no eixo y e Tamx o tamanho do bloco no eixo x

como 32×16, e 16×8 já possuem uma ocorrência bem maior. Portanto, pode-se dizer que este novo tamanho de bloco de CU novo se faz muito necessário no VVC. Além disso, este tamanho de bloco só pode ser atingido através de um particionamento binário ou ternário, o que nos sugere que estes novos segmentos da árvore de particionamento se mostram importantes para a eficiência de compressão atingida no padrão.

Vale ressaltar também a diferença entre um quadro Inter e um quadro Intra. Como podemos perceber, ambos possuem características bem diferentes. Para um quadro Inter o tamanho de bloco máximo pode atingir 128×128, buscando redundâncias espaciais maiores. Naturalmente por apresentar maior número de possibilidades, o quadro Inter tem uma distribuição de probabilidades muito maior, ainda assim podemos observar uma grande predominância de blocos quadrados. Também temos um grande número de blocos com tamanho 128×128, mostrando que mesmo com vídeos de resolução inferiores a HD estes blocos já eram de uma grande importância e não eram explorados em codificadores passados.

Com isso, podemos concluir que todos os tamanhos de bloco possíveis parecem ser importantes para manter a eficiência de compressão. Contudo, ainda seguimos com o problema em que o custo computacional para testar essas possibilidades é elevado. Com isso, conduzimos mais uma análise onde o objetivo é verificar o real custo computacional e eficiência que as árvores binárias e ternárias trazem ao codificador. Para isso modificamos o software de referência removendo essas duas ferramentas, impedindo o codificador de formar blocos retangulares, basicamente limitando o algoritmo de particionamentos para funcionar como seu antecessor HEVC. Os resultados de redução de tempo e *BD-rate* desta análise podem ser observados nas Tabelas 2, 3, 4.

Tabela 2 – Resultados de Redução de Tempo (RT) e *BD-rate* com o algoritmo sem MTT para a configuração All Intra

Classe	**Vídeo**	**All Intra**	
		BD-rate **(%)**	**RT (%)**
B	**Cactus**	3,976	91,32
C	**PartyScene**	4,130	92,42
D	**BasketballPass**	4,920	86,80
E	**FourPeople**	5,835	89,53
F	**SlideEditing**	10,496	90,35
	Média	**5,872**	**90,08**

Como podemos observar nos resultados apresentados, as árvores binárias e ternárias são de extrema importância para o processo de codificação. Quando desativamos esse tipo de particionamento, e permanecemos apenas com blocos quadrados,

Tabela 3 – Resultados de Redução de Tempo (RT) e *BD-rate* com o algoritmo sem MTT para a configuração Low Delay

Classe	**Vídeo**	**Low Delay**	
		***BD-rate* (%)**	**RT (%)**
B	**Cactus**	5,065	81,79
C	**PartyScene**	5,713	82,91
D	**BasketballPass**	6,381	76,70
E	**FourPeople**	6,433	69,21
F	**SlideEditing**	19,933	57,87
	Média	**8,705**	**73,70**

Tabela 4 – Resultados de Redução de Tempo (RT) e *BD-rate* com o algoritmo sem MTT para a configuração Random Access

Classe	**Vídeo**	**Random Access**	
		***BD-rate* (%)**	**RT (%)**
B	**Cactus**	4,683	83,10
C	**PartyScene**	5,825	84,25
D	**BasketballPass**	6,920	80,92
E	**FourPeople**	6,555	73,20
F	**SlideEditing**	14,781	66,99
	Média	**8,520**	**76,34**

a eficiência de compressão tem uma queda brusca. Na Tabela 2 podemos observar que na média temos um acréscimo em *BD-rate* de 5,8%, isso indica que boa parte do ganho do VVC em relação ao HEVC está relacionado a estes novos particionamentos. Podemos observar que os resultados variam de acordo com a classe do vídeo, resolução, e também redundância espacial. Por exemplo, o vídeo Cactus, que possui resolução FullHD, foi o vídeo com a menor perda de eficiência de compressão, porém o vídeo SlideEditing, de conteúdo de tela com resolução HD, acabou com uma perda em eficiência de compressão de mais de 10%.

Por outro lado, o custo computacional que essas ferramentas trazem são expressivamente altss, como podemos observar a redução de tempo de codificação quando paramos de testar a MTT e usamos apenas a árvore quaternária, em média, fica próxima de 90% para a configuração *AllIntra*. Este alto custo computacional se dá pelo fato de todos esses possíveis particionamentos ainda terem que processar outras diversas ferramentas para encontrar o melhor RD-Cost. Com isso, apesar de ser extremamente custosa, essa busca pelo melhor particionamento é necessária, porém muitos possíveis particionamentos não terão chance alguma de serem escolhidos.

Quando avaliamos as outras configurações de codificação *Low Delay* e *Random Access*, percebemos que a perda em eficiência de compressão é ainda maior, chegando a quase 9%, praticamente metade do ganho em *BD-rate*, de 21%, que o VVC consegue em relação ao HEVC. Ou seja, grande parte da eficiência de compressão do codificador vem dos novos particionamentos, porém quase todo custo computacional está no grande número de comparações que eles geram. Assim, concluindo a análise, este ponto específico do codificador traz diversas oportunidades de trabalhos para melhorias em tempo de codificação. Ainda assim, vale ressaltar que qualquer modificação neste ponto do codificador é extremamente sensível, já que abandonando possíveis particionamentos também se abandonam todos os cálculos de outras ferramentas aliadas que servem para se obter uma predição mais precisa.

O estudo sobre os particionamentos do VVC é vastamente explorado. (SALDANHA et al., 2020a) estende as análises elaboradas nesta seção trazendo uma análise isolando cada uma das árvores binárias e ternárias. Os autores apresentam resultados em redução no tempo de codificação quando desativados separadamente cada um dos particionamentos.

Como mostrado na Tabela 5, é possível observar que o desligamento do particionamento binário tem um maior impacto no tempo de codificação, totalizando em média 75,3%, enquanto que os particionamentos ternários chegam em média a 47,6%. Vale ressaltar que partições ternárias e binárias podem ser calculadas tanto através de blocos quadrados quanto de blocos retangulares. O fato de particionamentos ternários terem uma parcela menor no tempo de codificação pode se dar pelo fato de particionar um bloco em 3 bloco menores de proporção 1:2:1. Com isso, é mais comum atingir

Tabela 5 – Tabela dos resultados desativando apenas a árvore binária ou a árvore ternária do codificador VVC (SALDANHA et al., 2020a)

Classe	Resolução	RT	
		Sem BT	Sem TT
A1	**3840x2160**	67,9%	39,9%
A2	**3840x2160**	77,1%	48,1%
B	**1920x1080**	75,7%	47,5%
C	**832x480**	79,3%	51,9%
D	**416x240**	77,5%	51,2%
E	**1280x720**	**74,4%**	**47,2%**
Média		**75,3%**	**47,6%**

um tamanho de bloco pequeno demais para ser divisível, reduzindo assim as possibilidades de particionamento deste tipo. Por outro lado, os particionamentos binários sempre dividem um bloco na proporção 1:1.

Além desta análise, os autores também exploram o fato dos blocos de luminância terem um fator impactante bem superior quanto ao tempo de codificação. Os quadros luma tomam em torno de 85% do tempo de codificação. É interessante analisar este fator, uma vez que mesmo tomando a maior parcela no tempo de codificação os blocos de luminância são os que trazem o menor ganho em eficiência de compressão, como é mostrado em (PFAFF et al., 2021). Porém, é interessante relembrar que o VVC traz uma codificação para croma baseada nos blocos de luminância, de forma que um erro de codificação neste bloco pode causar uma perda em eficiência nos blocos de crominância.

Também é explorado pelos autores o custo computacional para cada tamanho de bloco analisado. O maior custo computacional está nos blocos menores, principalmente quando o tamanho de um dos eixos é inferior à 16 pixels. Principalmente os blocos retangulares que ocupam boa parte do custo computacional como os blocos de tamanho 8×4 ou 16×8.

Por fim, podemos concluir que os particionamentos do VVC são ferramentas complexas que trouxeram melhorias muito significativas para ao padrão alcançar os objetivos que se propôs. Porém, modificar este fluxo da codificação é algo extremamente delicado, visto que boa parte da eficiência de compressão está ligada a ele. O menor dos erros pode significar em uma perda grande em *BD-rate*. Assim, qualquer técnica focada na redução do custo computacional do codificador que abordar os particionamentos deve explorar algo que consiga predizer de certa forma as possíveis combinações e tomar decisões inteligentes. Para este tipo de solução, o aprendizado de máquina se encaixa de maneira muito conveniente, uma vez que treinar modelos que consigam se adaptar ao fluxo da codificação e que consigam classificar de maneira prévia o tamanho de blocos ou modos de particionamento conseguirá não só reduzir

o custo computacional, como também manter a qualidade da imagem e a taxa de bits reduzida.

O próximo capítulo traz uma abordagem baseada em Redes Neurais Convolucionais que busca aprender como os particionamento do bloco funciona e, assim, tomar decisões prévias que possam acelerar o processo de codificação. Para isso, é necessária uma metodologia assertiva para que os resultados futuros alcancem resultados desejáveis, que possa permitir o desenvolvimento de um codificador rápido e ainda assim eficiente quando empregada em conjunto com outras soluções.

4 REDE NEURAL CONVOLUCIONAL PARA PARTICIONAMENTO DE QUADROS INTRA

Este capítulo apresenta a solução proposta com o uso de aprendizado de máquina, mais especificamente Redes Neurais Convolucionais, para acelerar a codificação intra do padrão VVC, direcionada às decisões de particionamentos. Como já foi comentado na seção 2.4.3, diversos trabalhos já abordam os particionamentos do VVC. Este assunto traz uma imensa procura por soluções por uma grande gama de possibilidades de melhoras em tempo de codificação, porém sem afetar a eficiência em compressão.

Por existir este fator complicante, é necessário tomar um cuidado especial com soluções propostas. Normalmente soluções que envolvam particionamentos têm por padrão reduzir o número de comparações entre possíveis particionamentos, ou realizar um corte na árvore quaternária, binária ou ternária. Assim, não modifica-se o fluxo natural do codificador, apenas restringem-se a quantidade de comparações que o processo de Otimização Taxa-Distorção realiza para decidir o melhor particionamento. Por isso, este tipo de abordagem é comum e pode ser encontrado na literatura para diversos outros codificadores.

Para realizar uma modificação no processo de decisão dos particionamentos, um dos ramos da computação mais utilizados é a Inteligencia Artificial, com o uso de algoritmos que sejam capazes de classificar com um certo nível de análise em dados. Desta forma, diversos trabalhos ao longo da literatura já empregaram este tipo abordagem como o uso de *Random Forests*, ou até mesmo CNNs como mencionado na subseção 2.4.3. Neste trabalho, utilizamos algoritmos de Redes Neurais Convolucionais para acelerar o processo de codificação de CUs para quadros intra, já que podemos utilizar as imagens como entrada para os modelos. A seguir, este capítulo apresenta como foram criados os *datasets* de treino e teste, assim como a metodologia empregada para escolha de rede e treinamento. Os passos empregados para se obter um melhor modelo foram baseadas no que é conhecido como "fluxo universal de aprendizado de máquina", como mostrado na Figura 14 e explicado em (CHOLLET, 2017). Cada passo deste fluxo é explicado em uma seção neste capítulo, assim como todos os desafios enfrentados em cada etapa.

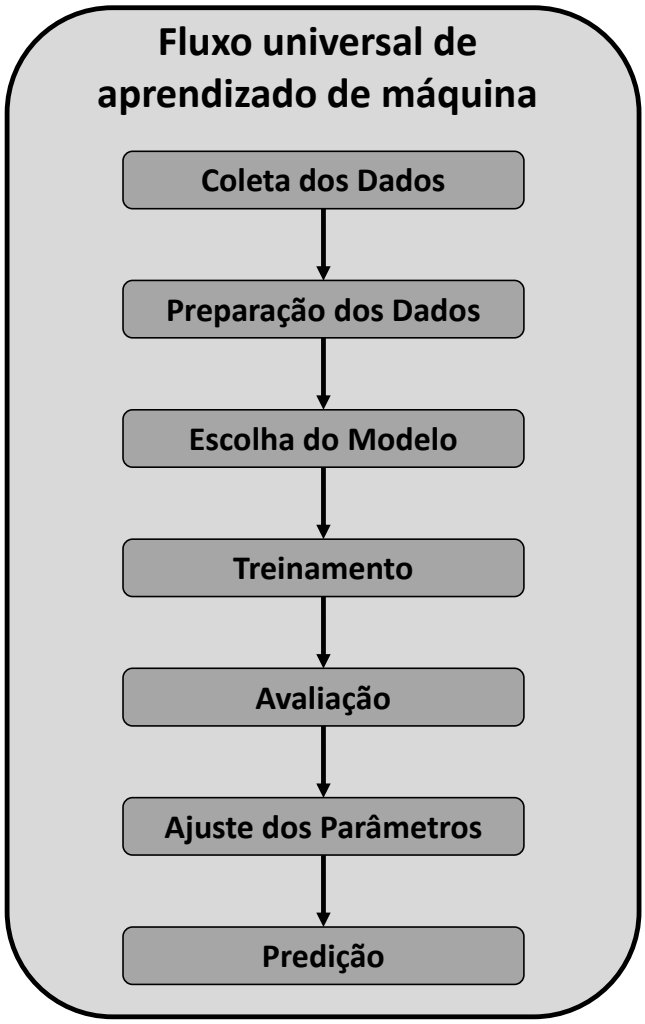


Figura 14 – Fluxo universal de aprendizado de máquina

4.1 Coleta de Dados

Primeiramente, é necessário construir um *dataset* que seja suficiente para que a rede consiga extrair o máximo de informações diferentes. Isso é necessário para que ela fique o mais generalizada possível, para conseguir ter uma probabilidade maior de predizer imagens não vistas durante o treinamento. Porém, conseguir definir um modo em que o *dataset* seja suficientemente genérico não é uma tarefa trivial, especialmente quando temos um número limitado de vídeos disponíveis que estão de acordo com as condições comuns de teste (*Common Test Conditions* - CTC) (F. BOSSEN J. BOYCE, 2019).

Ainda assim, para construir um *dataset* aceitável, ainda podemos recorrer às CTC de outros padrões, ou vídeos não comprimidos que não pertencem às CTC de nenhum padrão. Com isso, podemos reunir um conjunto com várias sequências disponíveis para avaliar quais são as que apresentam as melhores condições para construir um *dataset*.

Porém, como a CTC ainda inclui diversos tipos de vídeo que representam diversos conteúdos, como por exemplo *Screen Content*, onde a maior parte do vídeo tende a ser extremamente homogênea e com muitos blocos com informação extremamente similar, é necessário buscar outras sequências disponíveis. Este tipo de conteúdo não seria o ideal para estar contido num *dataset* mais limitado, pois diversos blocos teriam a mesma informação repetida, diversas vezes, podendo então vir a causar um vício na rede. Para fazer esta escolha, devemos levar em conta que temos poucas sequências disponíveis para trabalhar já que escolhemos trabalhar com as sequências que temos disponível das CTCs que temos acesso, dessas poucas sequências devemos explorar o máximo de informação diferente, blocos com o máximo de pixels distintos, que tivermos para construir um modelo mais robusto possível.

Analisando nossas limitações, tomamos como regra tentar encontrar, dentro das sequências que temos acesso, aquelas que consigamos extrair o maior número de blocos diferentes. Para isso utilizamos de um artifício muito usado na codificação de vídeo para classificar vídeos quando à sua heterogeneidade. Esta informação pode ser obtida através dos cálculos de Informação Espacial (SI) e Informação Temporal (TI), que nos dizem o quão heterogêneas são as informações de uma imagem, ou um vídeo.

O calculo do SI vai nos resultar em números que possuem uma maior diferença dos pixels dentro de um mesmo quadro. Portanto, vídeos que possuem muitas zonas iguais, como por exemplo um extenso céu azul, tenderão a ter um valor de SI muito baixo, enquanto vídeos que possuem muita informação diferente no mesmo quadro nos trarão um valor de SI muito maior.

Porém, não é suficiente termos apenas vídeos que possuem muita informação dentro de um mesmo quadro. Se esse vídeo, ao longo dos próximos quadros, não tiver muito movimento dos seus componentes, ou até mesmo um movimento de câmera, os blocos ainda assim permanecerão com a mesma informação ao longo do quadro. Portanto o que buscamos para este *dataset* são sequencias que não somente possuam muita informação diferente dentro do quadro, como também possua um maior movimento de objetos dentro da cena, ou então um movimento de câmera que vá mudando bastante o conteúdo.

O que é normalmente utilizado como SI e TI são os valores máximos de cada diferença de pixel. Este resultado por si só não é suficiente para a nossa avaliação. Tomemos, como exemplo, um quadro que seja muito homogêneo, mas tenha uma diferença grande entre dois pixels. Este vídeo terá um SI alto. Por outro lado, um vídeo que tenha uma mudança brusca em um ponto específico do quadro de um quadro para o outro também terá um TI alto, porém o resto da cena continua inalterada. Por isso, adaptamos o cálculo para algo mais adequado para o nosso caso. As equações 5 e 6 mostram como adaptamos estes cálculos para servir ao nosso propósito. O SI é

calculado através da média do desvio padrão do filtro *Sobel* calculado com base em um quadro do vídeo. O TI é calculado pela média do desvio padrão, da diferença de um pixel na posição i,j entre dois quadros adjacentes.

Ao invés de utilizarmos o valor mais alto de mudança de valor num pixel, utilizamos a média da diferença desses valores. Assim temos então valores mais condizentes, que podem nos indicar não apenas se os vídeos tendem a ser mais heterogêneos dentro do mesmo quadro, como também os vídeos que apresentam uma maior mudança entre os quadros.

$$SI = Media\{\sigma[Sobel(Quadro)]\} \tag{5}$$

$$TI = Media\{\sigma[D(i,j)]\} \tag{6}$$

Calculamos o SI e o TI médio para cada umas das 40 sequências de vídeo que tivemos acesso. As Figuras 15 e 16 mostram de forma resumida um gráfico de dispersão para as sequências em resolução 4K e FullHD. Valores mais altos de SI e TI médios indicam que os vídeos tendem a possuir um conteúdo mais heterogêneo.

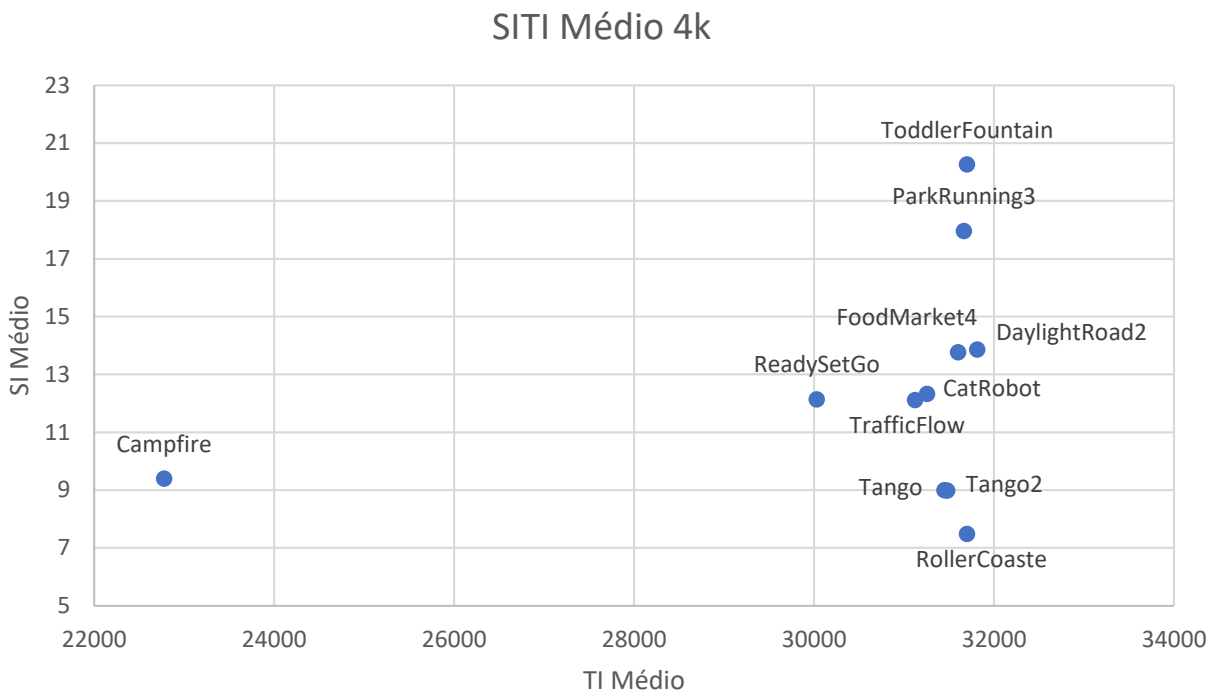


Figura 15 – SITI médio calculado para as sequências 4k

Como podemos observar nos gráficos, as sequências que se encontram no canto superior direito são as sequências que possuem um SITI médio mais alto, ou seja, tendem a ser mais heterogêneas. Nestes casos podemos observar que as sequências

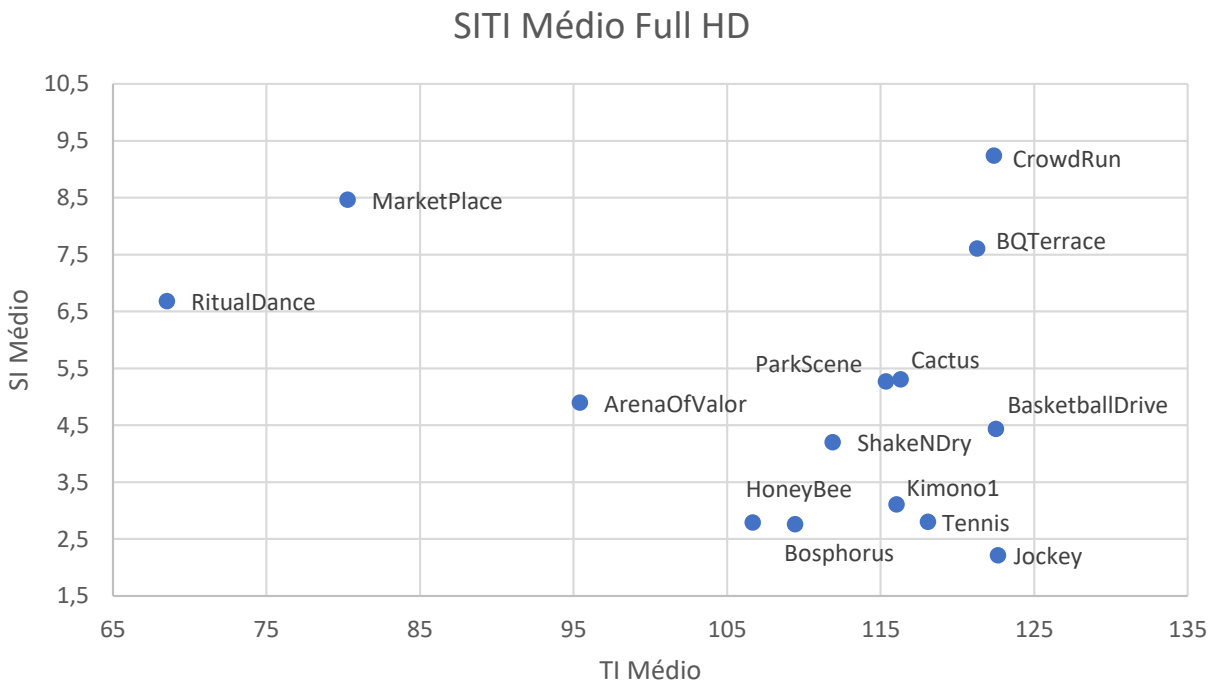


Figura 16 – SITI médio calculado para as sequências Full HD

ToddlerFountain e ParkRunning foram as sequências de resolução 4K com maior SITI médio, enquanto as sequências CrowdRun e BQTerrace são as sequências com um maior SITI médio para a resolução FullHD.

Temos diversas sequências com um TI alto, indicando uma heterogeneidade alta ao longo dos quadros, porém com um SI baixo, indicando que possuem mais pixels com valor igual ou muito próximos. Portanto, separamos as sequências com maiores valores para o *dataset* de treino e as sequências com valores menores para o *dataset* de teste. As sequências podem ser observadas na Tabela 6.

4.2 Preparação dos Dados

Durante a etapa de modelagem envolvendo redes neurais, diferentes tipos de entradas podem ser consideradas para alimentar o treinamento destes modelos. Dado que este trabalho está inserido no contexto de codificação de vídeo, diversas abordagens podem ser feitas, desde trabalharmos com imagens puras, até usarmos os próprios dados de codificação para predizer próximos passos de codificação. Como o foco deste trabalho é a codificação intra, e este tipo de configuração, para o padrão VVC, não aproveita dados de quadros vizinhos já codificados para tomar decisões em seus particionamentos, uma maneira, entre várias, simples de utilizar como entrada de uma rede seriam os próprios blocos que serão divididos.

Como mencionado na seção 2.2.1, existem muitos possíveis particionamentos dentro de um bloco 128×128. Com isso, decidimos restringir a entrada e o que esperamos da predição da rede para um nível mais alto. A entrada da Rede se dá por uma imagem 256×256 pixels: esta imagem contém o bloco atual a ser predito e os blocos diretamente vizinhos à esquerda, acima e na diagonal. Com isso, temos dados não apenas do bloco predito, mas também de seus vizinhos que possuem informações próximas e que o codificador poderia usar informações de blocos já codificados na configuração *All Intra*. A imagem 17 representa possíveis entradas para a rede, não são usados blocos vizinhos a direita pois na ordem de codificação de blocos estes vizinhos ainda não foram codificados quando o bloco atual está sendo codificado.

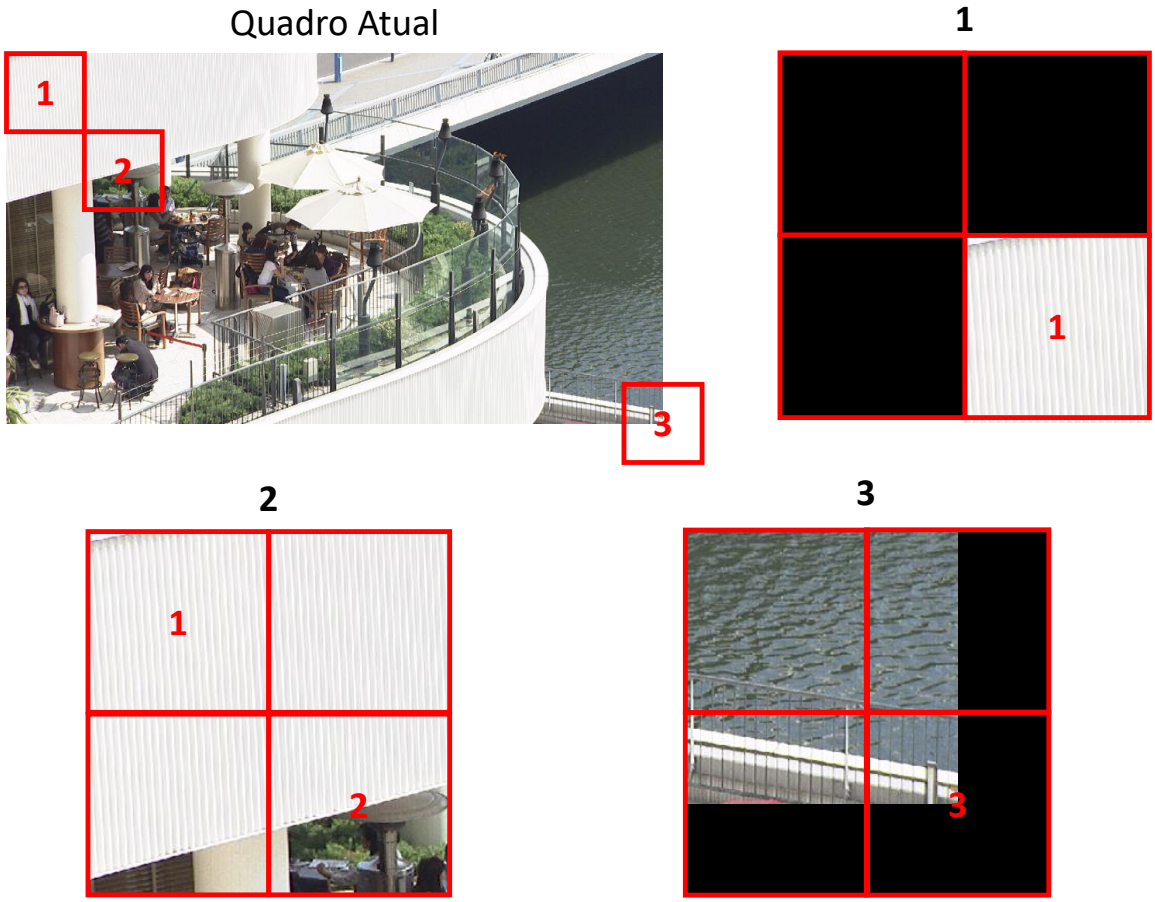


Figura 17 – Possíveis blocos do *dataset* criado, usados como entrada da rede neural

A Figura 17 mostra 3 exemplos dentro de um quadro de um vídeo, que representam blocos 128×128 pixels e que são complementados para 256×256 pixels com informações de blocos vizinhos. No exemplo 1 temos o primeiro bloco 128×128 dentro do quadro. Este bloco não possui blocos vizinhos à esquerda nem acima, portanto, para construir uma entrada que represente este bloco para uma rede, optamos por preencher o restante do bloco apenas com preto. Isso indica que ele é um bloco pertencente

a uma borda.

O exemplo 2 mostra um bloco comum no quadro, que possui informação nos seus 3 blocos vizinhos. Este é o tipo mais comum de bloco e consegue retirar informação dos blocos vizinhos para servir como entrada da rede neural. Por fim, o exemplo 3 mostra os exemplos de vídeos onde a resolução horizontal e/ou vertical do vídeo não é divisível por 128. Nestes casos, preenchemos bordas à direita e abaixo para indicar que este bloco também pertence a uma borda. Portanto a entrada da rede será uma imagem de 256×256 pixels, que representa um bloco de CTU, que o modelo deverá predizer o término precoce do particionamento da árvore quaternária baseado nas informações contidas nesse bloco. A escolha por este tamanho de bloco se baseia não só no bloco atual que está sendo predito, mas também para a o modelo conseguir extrair informações de blocos diretamente adjacentes.

Com a entrada da rede definida, resta categorizar cada uma dessas entradas para que seja possível aprender o padrão e classificar. Os exemplos de classificação a serem usados no treinamento precisam ser extraídos de *bitstreams* de vídeos já codificados ou do próprio processo de codificação, sob a forma da profundidade máxima da árvore quaternária escolhida pelo codificador original. Portanto, cada uma das sequências mostradas na Tabela 6 foi codificada na configuração All Intra. Note-se que essas sequências foram codificadas com cada um dos QPs que estão listados como padrão na CTC, conforme mostra a Tabela 6.

Portanto, cada QP terá o seu *dataset* separado, de treino e de teste. Escolhemos seguir este padrão porque os QPs possuem uma disparidade grande entre eles, onde o QP 22 tende a resultar em blocos muito menores do que o QP 37, de forma que generalizar com imagens iguais e valores de classificação tão distintos poderia resultar em um problema no aprendizado. Assim, para cada QP recomendado na CTC um modelo separado foi treinado. A única diferença entre eles será a própria classificação do codificador, que poderá variar no número de instâncias para cada profundidade da árvore quaternária. Por fim, a preparação dos dados se finaliza com 4 *datasets* completos de treino e teste, para cada um dos QPs usados na CTC, com os mesmos dados de entrada, ou seja, os mesmos blocos, e diferentes classificações que variam de acordo com o codificador. O número de exemplos para cada *dataset* é 172480, 271840, 371056, 183632 para respectivamente cada um dos QPs de 22 à 37, já ajustados com balanceamento igual para todas as classes.

4.3 Escolha do Modelo

Com o *dataset* escolhido e a preparação dos dados de entrada feitos, o próximo passo do pipeline é escolher um modelo para treinar os dados. Com o grande avanço das redes neurais convolucionais, possuímos uma gama muito grande de possíveis

Tabela 6 – Configuração dos experimentos Treinamento e teste dos modelos

Codec	VTM Model 6.0
Configuração	*All Intra*
QPs	22, 27, 32, 37
Sequências de Treino	*DaylightRoad2*, *ToddlerFountain*, *RollerCoaster*, *FoodMarket4*, *TrafficFlow*, *ReadySetGo*, *Jockey*, *BasketballDrive*, *CrowdRun*, *BQTerrace*, *Tennis*, *Kimono1*, *ParkScene*
Sequências de Teste	*Tango*, *ShakeNDry*, *Bosphorus*, *HoneyBee*, *ArenaOfValor*, *Vidyo3*, *Vidyo4*, *Flowervase*

modelos a serem escolhidos. A biblioteca do *pytorch* foi utilizada para treinar os modelos, de forma que alguns exemplares disponíveis na biblioteca foram considerados, como: Alexnet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012b), Resnet (HE et al., 2016), VGG (SENGUPTA et al., 2019), SqueezeNet (IANDOLA et al., 2016) e GoogleNet (SZEGEDY et al., 2015).

Para ser possível escolher um modelo dentre as opções citadas, é necessário realizar um treinamento com algumas épocas para verificar como cada modelo vai performar com o *dataset* criado. Com isso, será então possível verificar e analisar o tamanho da rede versus a acurácia alcançada por cada modelo. Portanto, cada modelo listado acima foi treinado com a biblioteca do *pytorch* por 50 épocas, com pré treinamento, para ter uma margem para comparação entre os modelos, de forma que se pudesse verificar qual deles conseguiria alcançar uma melhor acurácia. Foram considerados um tamanho do *batch* de 64, um *Learning-rate* de 10^{-5} utilizando o otimizador SGD com um *momentum* igual a 0,9, conforme a biblioteca *pytorch* define como padrão e modificando a última camada de classificação para uma saída de tamnho 4.

Como citado na Tabela 6, dividimos os vídeos que tinham maior SITI para treino e também separamos um conjunto diferente de vídeos para um conjunto de teste. Assim, conseguimos garantir que o modelo nunca tenha visto nenhuma imagem do conjunto de teste, já que caso optássemos por apenas extrair uma porcentagem das nossas imagens de entrada dos vídeos de treino, teríamos uma grande probabilidade de termos imagens exatamente idênticas ou muito similares devido à homogeneidade característica de vídeos.

A Tabela 7 mostra que, entre os modelos testados, o modelo que obteve o resultado mais satisfatório de acurácia foi o modelo da Resnet18. Além disso, este modelo é o mais simples entre os modelos da Resnet. Apesar de existirem modelos capazes de alcançar resultados mais expressivos, como um erro de classificação menor, porém quanto mais complexo o modelo, maior a rede, possuindo mais camadas, tornando o processo mais complexo. Por exemplo, a Resnet18 possui 18 camadas enquanto

Tabela 7 – Acurácia dos modelos treinados

Modelo	Acurácia (%)
Alexnet	43,2
Resnet	56,1
VGG	48,7
SqueezeNet	49,3
GoogleNet	52,4

versões mais robustas como Resnet50 e Resnet101 que possuem 50 e 100 camadas, respectivamente.

Como o principal objetivo do trabalho consiste em atingir uma redução no tempo de codificação, optamos por manter o modelo mais simples possível. Portanto, o modelo escolhido neste trabalho para predição do término precoce do particionamento da árvore quaternária é o Resnet18. Após a escolha, os próximos passos são treinar este modelo para os 4 QPs de configuração recomendada do VVC e verificar o seu desempenho.

4.4 Treinamento, Avaliação e Ajuste dos Hiperparâmetros

Esta seção trata do treinamento e avaliação de cada modelo (um para cada QP), treinados com a arquitetura do modelo pré-treinado da Resnet18. Portanto, foram usados modelos que já haviam sido pré-treinados e que são disponibilizados pela biblioteca do *pytorch*. Estes modelos pré-treinados possuem os pesos já ajustados quando treinados com o *dataset* da ImageNet, que possui mais de 1 milhão de imagens com 1000 categorias diferentes. Este tipo de abordagem é recomendada por já aproveitar de pesos adquiridos com estes treinamentos com um *dataset* de magnitude muito grande. O fluxo de treino é exemplificada na Figura 18. Porém estas redes pre treinadas devem seguir uma adaptação apra o problema tratado nesta dissertação, com isso adaptamos a última camada da rede para classificar 4 saídas e as renomeamos para receberem os labels adequados aos nosso problema.

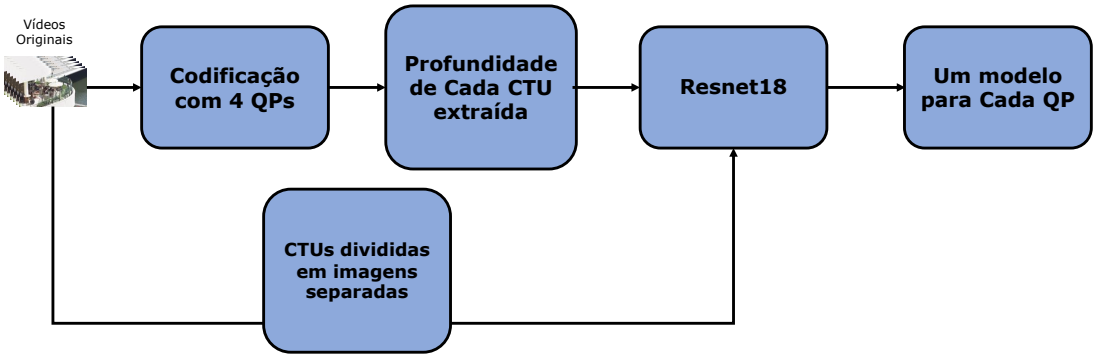


Figura 18 – Fluxo do Treinamento dos Modelos das Redes Neurais

A entrada da rede neural é uma imagem com 256×256 pixels. A rede deve prever, de acordo com a codificação prévia do vídeo, uma profundidade máxima para a árvore quaternária. Como a árvore quaternária não possui blocos de tamanho 128×128 na codificação de um quadro intra, assim como também não permite dividir de forma quaternária até blocos de tamanho 4×4, o número de possibilidades de classes são quatro. As possíveis classificações são: 64×64 como classe 1, 32×32 como classe 2, 16×16 como classe 3, e 8×8 como classe 4.

Todos os modelos foram treinados por 100 épocas. A Figura 19 mostra em gráfico os resultados de acurácia de treino e teste. Foram considerados um tamanho do *batch* de 64, um *Learning-rate* de 10^{-5} utilizando o otimizador SGD com um *momentum* igual a 0,9 que são definidos como padrão na biblioteca do *pytorch*. Como se pode observar, todos os modelos sofreram *overfitting* após algumas épocas de treinamento, onde a acurácia sobre o *dataset* de treinamento seguiu aumentando, porém a acurácia para o *dataset* de teste estagnou.

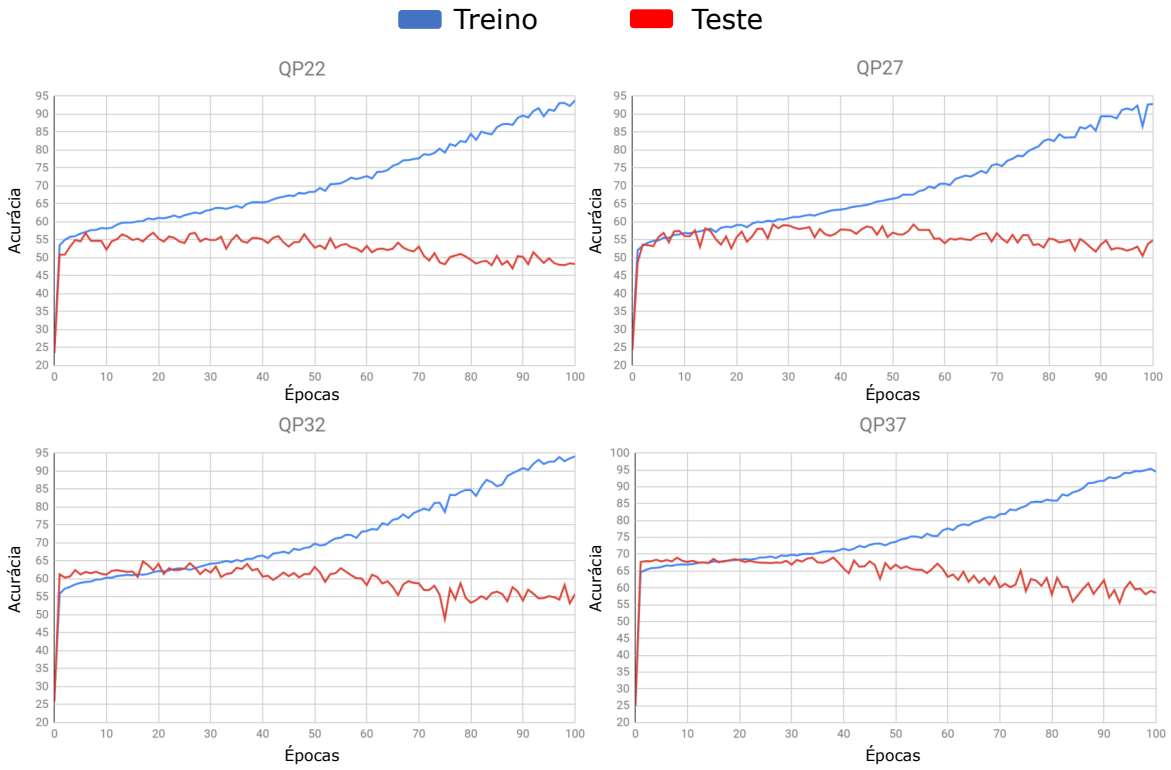


Figura 19 – Resultado dos treinamentos das redes neurais para QP 22, 27, 32 e 37. Onde o eixo X representa as épocas do treinamento e o eixo Y representa a acurácia do modelo

Por outro, lado a Figura 20 mostra como foi o comportamento da função de perda ao longo do treinamento. Os gráficos mostram que esta função teve um comportamento como esperado, reduzindo ao longo do treinamento e tendendo a zero, conforme o modelo vai se especializando. Como o desejável é não ter um modelo com *overfitting*, alguns parâmetros foram modificados para evitar o *overfitting*. Porém,

nenhum dos métodos tentados, como *dropout* de algumas camadas, alteração no *Learning-rate* e *momentum*, foram suficientes para alcançar uma acurácia de teste maior ou evitar que os modelos decorassem o conjunto de treino. Portanto, decidimos escolher os modelos na época com a maior acurácia no *dataset* de teste, antes que o modelo começasse a apresentar vício nos dados de treino.

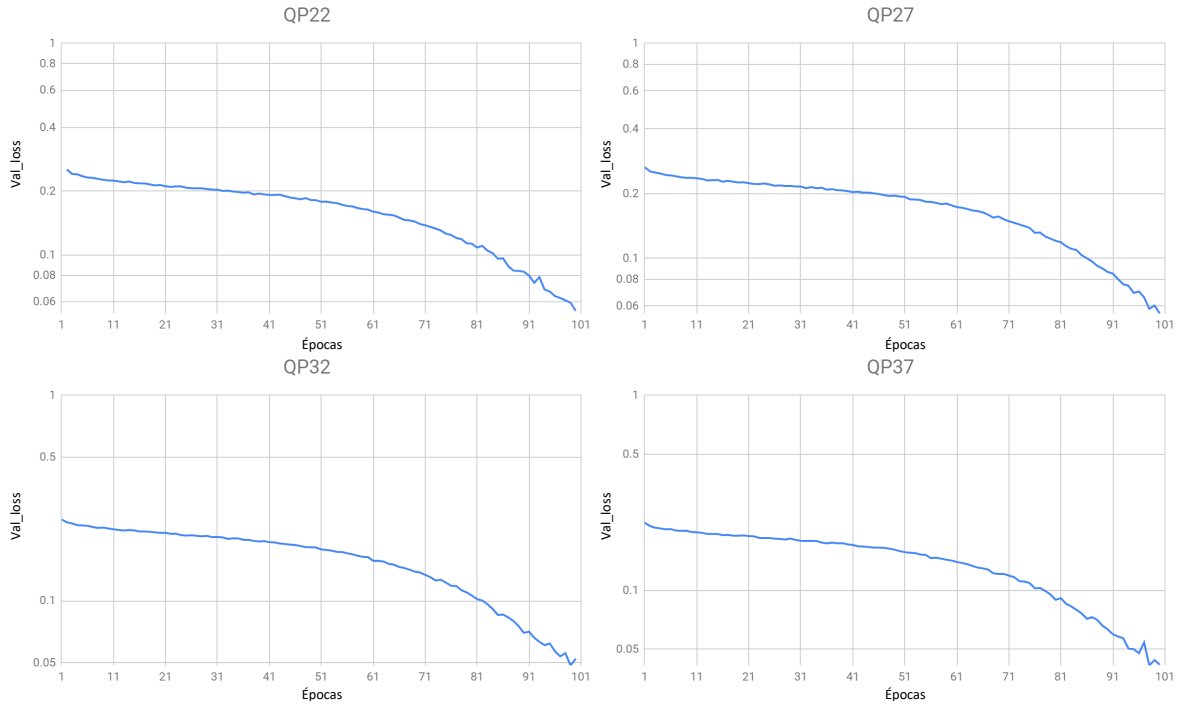


Figura 20 – Resultado da função de Loss por época de treinamento das redes neurais para QP 22, 27, 32 e 37

Observar apenas os resultados de acurácia nem sempre diz muito sobre os modelos, ainda mais quando se trata do caso em específico abordado neste trabalho. Para a árvore quaternária em especifico e a forma como ela é avaliada pelo codificador, uma parada mais cedo ou mais tarde não indica um erro necessariamente. Por exemplo, caso tenhamos uma classificação errônea para um determinado bloco indicando que ele pare o particionamento da árvore quaternária em um tamanho 8×8, quando a classe correta deveria ser 32×32, não estamos necessariamente perante um erro no processo de codificação. Na verdade, o codificador não deixará de testar a profundidade correta; apenas testará níveis a mais sem necessidade.

Por outro lado, caso deixemos de testar níveis mais abaixo na árvore quaternária quando deveríamos, estamos perdendo em eficiência de compressão. Porém, caso esta perda não seja significativamente alta, estaremos economizando em tempo de codificação. Portanto, para avaliar estes dois casos precisamos de uma métrica que nos indique o número de escolhas que tomamos erroneamente e que levam a perdas na eficiência de compressão e tempo. Para isso, calculamos a porcentagem de vezes que o modelo prediz uma classe com um valor inferior ao devido, por exem-

Tabela 8 – Resultados de Precisão, Recall, F1-Score, Erro-BD, Erro-Tempo e Acurácia

Modelos	Classes	Precisão	Recall	F1-Score	Erro-BD	Erro-Tempo	Acurácia
QP22	**Classe1**	0,77	0,60	0,67	31,71%	24,69%	56,12%
	Classe2	0,46	0,64	0,53			
	Classe3	0,47	0,53	0,50			
	Classe4	0,67	0,48	0,56			
	Média	0,59	0,56	0,57			
QP27	**Classe1**	0,88	0,61	0,72	30,81%	22,12%	58,33%
	Classe2	0,44	0,69	0,54			
	Classe3	0,52	0,65	0,58			
	Classe4	0,86	0,32	0,47			
	Média	0,68	0,57	0,58			
QP32	**Classe1**	0,84	0,74	0,79	24,07%	18,37%	65,08%
	Classe2	0,46	0,63	0,53			
	Classe3	0,53	0,54	0,53			
	Classe4	0,87	0,68	0,76			
	Média	0,67	0,65	0,65			
QP37	**Classe1**	0,83	0,67	0,74	17,73%	19,95%	68,31%
	Classe2	0,55	0,76	0,64			
	Classe3	0,67	0,52	0,59			
	Classe4	0,78	0,78	0,78			
	Média	0,71	0,69	0,69			

plo, classificou como 1 quando deveria ser 3, ou classificou com um valor superior ao que deveria, como por exemplo, classificou como 3 e deveria ser 1, Neste trabalho, chamamos estes erros de Erro-BD e Erro-Tempo respectivamente.

A Tabela 8 traz os resultados de todas as métricas explicadas na subseção 2.3.4 para os quatro modelos treinados. Os resultados estão separados por modelo, classes e a média do modelo. Para as métricas de Precisão, Recall e *F1-score*, temos os dados separados por classe e uma média. Já para as métricas de Acurácia, Erro-BD e Erro-Tempo apresentamos apenas um valor para ao modelo.

Podemos observar que em todos os modelos tivemos uma precisão maior nas classes 1 e 4. Isso indica que de uma maneira geral os modelos conseguem distinguir melhor blocos de tamanhos maior ou com o menor tamanho possível. Podemos afirmar isso pois por possuir uma precisão alta para esses casos, indica que quando predizemos um bloco ele tem uma probabilidade maior de ser de fato verdadeiro, já que uma alta precisão indica que um alto acerto do total de observações verdadeiras preditas.

Quando observamos os resultados de *Recall*, observamos que as classes tendem a ter valores mais equilibrados entre todos os modelos, com exceção do modelo para QP 27, que para a classe 4 aponta um valor de *Recall* bem inferior aos outros.

As métricas citadas acima conseguem nos representar o comportamento de cada modelo para cada classe. Porém, como comentado anteriormente, elaboramos duas

métricas para descrever como as escolhas do modelo afetam de fato no algoritmo de decisão da árvore quaternária do codificador. Quando analisamos a métrica Erro-BD, estamos olhando para a porcentagem de escolhas erradas no modelo que irão de fato afetar a eficiência de codificação. Isso porque estas escolhas deixarão de testar níveis que seriam escolhidos pelo codificados como particionamento ideal. Este erro acontece entre 17% e 31% das vezes no modelo, tendo o modelo do QP 22 como o pior entre eles, ou seja, provavelmente o modelo que causará uma perda maior na eficiência de compressão.

Por outro lado, temos o Erro-Tempo, que indica a porcentagem de erro em que testamos níveis a mais do que o necessário. Este erro vai afetar diretamente a técnica proposta na redução de tempo no processo de codificação, já que testando mais níveis abrimos espaço para mais possíveis particionamentos serem testados. Os modelos apresentam um Erro-Tempo entre 19% e 24%. Podemos ressaltar também que o maior valor de Erro-Tempo ainda é mais baixo que o maior valor de Erro-BD.

4.5 Emprego dos Modelos na Solução Proposta

As análises apresentadas no capítulo 3 e a rede neural treinada conforme apresentado nas seções anteriores deste capítulo levaram às conclusões que orientaram o algoritmo de aceleração da codificação intra apresentado nesta seção, visando reduzir o tempo de codificação com perdas insignificantes na eficiência de compressão. A solução proposta busca combinar ao máximo as informações obtidas na análise dos particionamentos das CUs e os modelos treinados que também se baseiam em trabalhos relacionados discutidos na seção 2.4.

A solução proposta neste trabalho emprega ao processo de codificação VVC diretamente o vetor de predições das redes neurais obtidos para cada CU, como mostrado na Figura 21. Note na figura uma segunda entrada de dados no codificador VVC, correspondente aos vetores de predições oriundas dos modelos treinados.

Assim, antes do processo de codificação, as CTUs são particionadas como explicado em 2.2.1. Essas CTUs servem como entrada da rede neural que irá classificar cada uma com uma profundidade máxima de divisão quaternária. Desta forma, quando o quadro é entregue ao codificador, é também fornecido um mapeamento das profundidades da CTU para este quadro correspondente e o codificador usa-o para acelerar a codificação.

O algoritmo de divisão da CU adaptado executado no codificador é apresentado na Figura 21. O fluxograma mostra que a busca recursiva pelo melhor tamanho de CU (função encode_CU) é interrompida se a profundidade atual $prof_i$ atingir um limiar $prof_{mapa}$, que é o profundidade da CU predita pelo modelo correspondente. Da mesma forma, se a profundidade atual estiver abaixo do limite $prof_{mapa}$, o processo de divisão

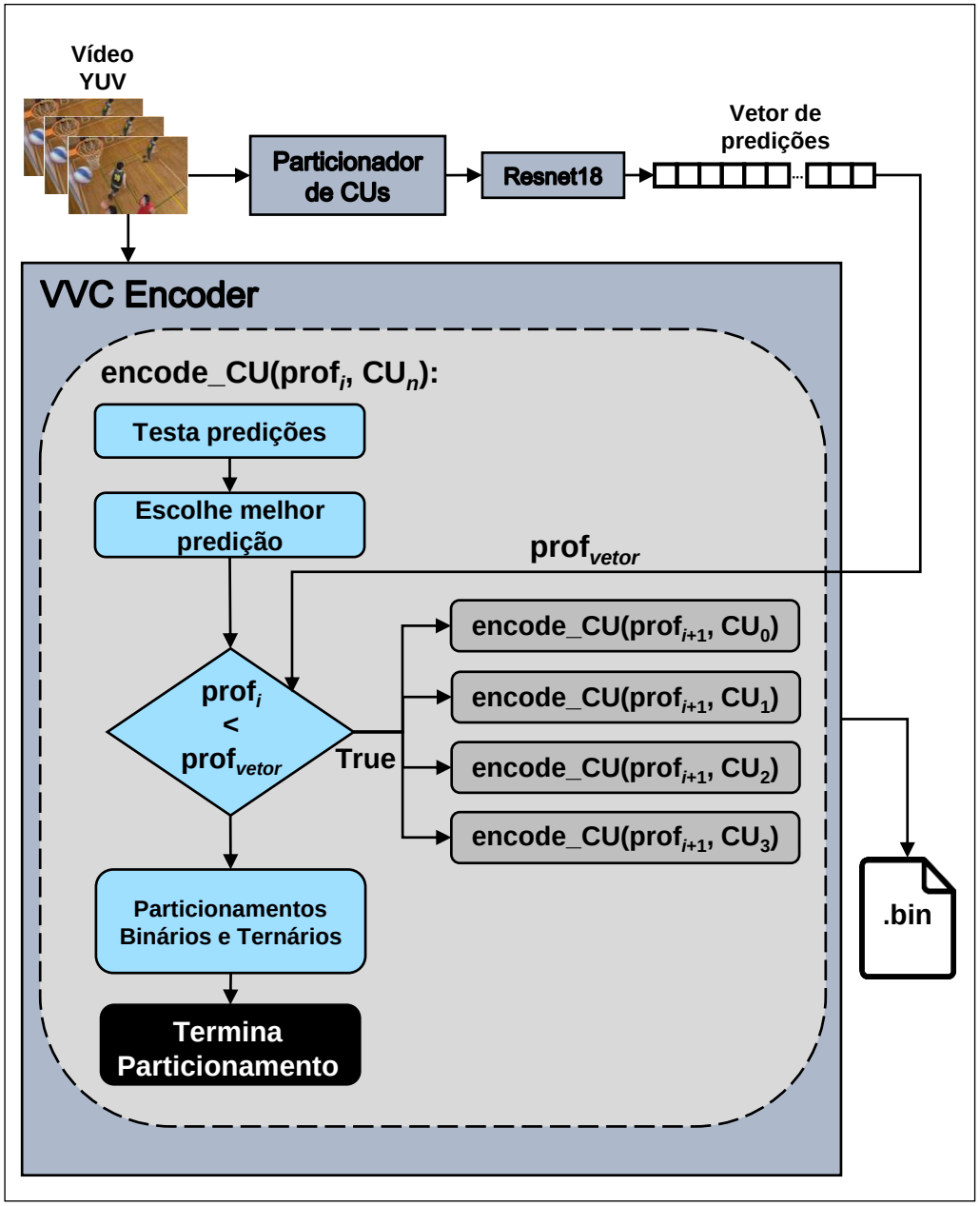


Figura 21 – Solução completa com uso dos modelos para predição dos particionamentos

da CU continuará. Por exemplo, se uma CU foi predita com a profundidade 1, somente as profundidades 0 e 1 serão testadas durante a codificação e, em seguida, o processo será interrompido.

É importante lembrar que para cara passo na recursão da árvore quaternária, ainda temos os testes de particionamentos binários e ternários. Como já explicado na seção 2.2.1, estes particionamentos possuem uma recursão própria para mais particionamentos binários ou ternários, porém nunca para particionamentos quaternários, mesmo que o tamanho da CU atual seja quadrado. É importante observar que quanto mais cedo cortarmos este processo, menos possíveis particionamentos são testados não só para tamanhos quadrados, como para os novos tipos de particionamento também.

É esperado que, devido aos resultados dos modelos apresentados na subseção 4.5, os erros nos quais deixamos de testar profundidades maiores não sejam tão expressivos. Assim, uma degradação na imagem ou aumento na taxa de bits deve ser de uma magnitude pequena. Como este é um corte em um nível mais alto da árvore e consequentemente outros tipos de particionamentos e predições serão evitados, ainda assim erros em blocos específicos podem trazer uma degradação indesejável.

A próxima seção apresenta os resultados obtidos por esta solução proposta e os compara com trabalhos relacionados encontrados na literatura. Também serão discutidas as similaridades que esta solução tem com outros trabalhos e casos em que ambos trabalhos poderiam se complementar em soluções conjuntas.

5 RESULTADOS EXPERIMENTAIS

Este capítulo apresenta os resultados experimentais obtidos através da solução proposta, apresentada no capítulo anterior 4.5. O capítulo também apresenta uma comparação e discussão de práticas conjuntas, entre os resultados alcançados neste trabalho e os trabalhos apresentados na seção 2.4.

Os experimentos foram conduzidos com base em condições próximas das análises dos particionamentos apresentada no capítulo 3, e do treinamento dos modelos, com exceção das sequências de vídeo que são diferentes daquelas para garantir resultados imparciais. Todas as sequências usadas nos experimentos são recomendadas na CTC (F. BOSSEN J. BOYCE, 2019) e estão listadas na Tabela 9. Observe que as 10 sequências pertencem a três classes de resolução espacial: HD (1280x720 e 1024×768 pixels), Full HD (1920×1080 pixels) e 4K (4096×2160 e 3840×2160 pixels). Vale ressaltar também que a versão do código de referência usado para os testes foi uma versão estável mais atual até o momento.

Tabela 9 – Configuração dos experimentos para os resultados experimentais

Codec	VTM Model 9.0
Configuração	*All Intra*
QPs	22, 27, 32, 37
Sequências de Teste	*Tango2*, *CatRobot*, *Cactus*, *MarketPlace*, *RitualDance*, *BasketballDrill*, *BQMall*, *PartyScene*, *SlideShow*, *SlideEditing*,

Todas as sequências foram codificadas primeiro com o algoritmo no seu estado padrão, com os 4 QPs, para todos os quadros Intra, lembrando que na configuração *All Intra* padrão recomendada nas CTC os quadros são subamostrados para aceleração dos testes: apenas um a cada 8 quadros é codificado e os demais são descartados. Após, cada sequência passou por um particionador de CUs como mostrado na Figura 21, onde as CUS são particionadas como explicado na seção 4.2. Após isso, cada CU serve como entrada nos modelos treinados, para então o vetor de predições servir como uma entrada no processo de codificação onde o software modificado, processa

Tabela 10 – Resultados de *BD-rate* e redução de tempo de codificação para as sequências escolhidas como teste

Sequência	*BD-rate (%)*	RT (%)
Tango2	0,456	12,49
CatRobot	0,618	10,66
Cactus	1,590	21,41
MarketPlace	0,015	6,56
RitualDance	0,450	9,95
BasketballDrill	0,266	11,62
BQMall	0,123	15,91
PartyScene	0,129	10,36
SlideShow	0,271	12,96
SlideEditing	0,398	15,33
Média	0,432	12,73

esses valores para avaliar no processo de particionamento dos blocos.

Portanto, os resultados refletem uma comparação entre o codificador padrão (sem nenhuma modificação) e o codificador modificado com a solução proposta, incluindo todas as etapas de pré-processamento e predições realizadas antes do processo de codificação do vídeo. Com isso, comparamos os dois modos em termos de *BD-rate* e redução de tempo (RT), onde RT é calculado através da razão entre o tempo de processamento do codificador de referência (original) e o tempo de processamento das etapas de pré-processamento, predição com modelos e execução do codificador modificado, conforme apresenta a equação 7.

$$RT = \frac{TO - TM}{TO} \tag{7}$$

A Tabela 10 mostra todos os resultados obtidos para cada sequência testada, assim como a média dos resultados de redução de tempo e *BD-rate*. Em média, nossa solução foi capaz de reduzir o tempo de codificação em 12,73% e levou a uma perda em eficiência de compressão de 0,432%, medida em *BD-rate*. Foi possível atingir uma redução considerável, levando em conta que o novo padrão de codificação VVC é extremamente mais complexo que seus antecessores. A perda em eficiência de compressão pode se considerar inexpressiva quando observamos que a redução em *BD-rate* que o padrão ja atingiu em relação ao HEVC foi de aproximadamente 25%.

Quando observamos dados isolados, a solução conseguiu atingir uma redução no tempo de codificação máximo de 21,41% para a sequência *Cactus*. Esta mesma sequência também foi a que atingiu a pior perda em eficiência de compressão, alcançando 1,590%, uma perda um pouco mais significativa dentro do ganho que o padrão atinge. Em contraponto, a pior redução em tempo de codificação se deu para a sequência *MarketPlace* que conseguiu alcançar apenas 6,56% de redução no tempo.

Em compensação, a perda de eficiência em compressão permaneceu desprezível.

Também vale ressaltar que, por mais que a rede neural acerte suas predições, ainda assim não é garantido uma boa redução de tempo. Tomemos, como exemplo, a sequência *MarketPlace*. Os valores em *BD-rate* indicam que a rede provavelmente teve um Erro-BD muito baixo, ou os erros que cometeu não afetaram tanto na eficiência de compressão. Porém, mesmo acertando muito, não houve uma redução no tempo de codificação tão expressiva. Este resultado pode ser explicado devido aos particionamentos serem, em sua grande maioria para esta sequência, de tamanhos pequenos, fazendo assim com que seja muito difícil acelerar o processo de codificação por não ser possível cortar níveis da árvore quaternária.

Podemos dizer também que o melhor resultado entre redução de tempo e *BD-rate* está descriminado na sequência *BQMall*, que atingiu uma redução no tempo de codificação de 15,91%, valor este que se encontra acima da média entre as outras sequências. Além disso, apresentou um *BD-rate* de 0,123%, bem abaixo da média de todas as sequências, atingindo assim a melhor relação custo-benefício entre estes dois fatores.

Como estamos incorporando novas etapas anteriores ao processo de codificação, que consistem em preparar os dados de entrada da rede e classificá-los, é importante também observar o tempo que estes dois processos em conjunto representam no tempo total da solução. Com isso, pode-se avaliar se estes dois passos representam uma parcela grande ou do processo completo de codificação e, caso seja, se podemos tomar medidas para reduzir este impacto.

Para realizar esta análise, a solução proposta foi executada duas vezes para todos os vídeos: uma vez com a classificação realizada por GPU e uma vez com a classificação realizada por CPU. A GPU utilizada foi a *NVIDIA GEFORCE GTX TITAN V*. A CPU utilizada foi um processador *Intel® Core™ i7-8700*. Em média o tempo de preparação dos dados, mais a predição realizada pelas redes neurais, condiz com 0,071% do tempo total do processo de codificação quando usamos GPU para a classificação das CUs. Quando utilizamos CPU para a mesma tarefa, a parcela de tempo aumenta para 1,06% do tempo total da solução completa. Podemos concluir, portanto, que o overhead gerado pelas etapas de pré-processamento e predição é muito pequeno em relação ao tempo de codificação.

5.1 Comparação com Trabalhos Relacionados

Outra forma de analisar os nossos resultados é comparamos o que conseguimos atingir com a nossa solução proposta com trabalhos similares na literatura. Nem todos os trabalhos são 100% comparáveis por existirem suas diferenças entre soluções. Além disso, apesar de apresentarmos comparações entre trabalhos, em muitos casos

Tabela 11 – Resultados comparados com trabalhos relacionados

Trabalhos Comparados	*BD-rate (%)*	RT (%)	Versão do Software
(FU et al., 2019)	1.16	45.6	1.0
(PARK; KANG, 2019)	1.03	45,0	3.0
(LEI et al., 2019)	1.02	33,7	4.0
(SALDANHA et al., 2020b)	1.10	30.6	5.0
(TISSIER et al., 2019)	1.02	33.7	6.1
(LI et al., 2021)	1.53	55.4	7.0
Solução Proposta	0.43	12.7	9.0

as soluções podem coexistir no codificador. Por exemplo, uma solução que foque apenas na redução de custo computacional da codificação de quadros intra pode coexistir com uma solução que foque em quadros inter, tentando assim atingir um valor maior na redução de tempo geral da codificação.

Também é necessário observar que como o VVC teve trabalhos publicados enquanto o padrão ainda estava em consolidação e o software de referência segue em constante modificação, os trabalhos comparados tendem a apresentar resultados para diferentes versões. Por isso, além de comparar os trabalhos, também informamos qual versão do software de referência é utilizada, para uma comparação mais justa. A Tabela 11 demonstra os dados médios de cada trabalho relacionado encontrado na literatura. Todos os trabalhos estão ligados por usarem técnicas para redução no tempo de codificação para quadros intra. Além disso, todas soluções envolvem, pelo menos em parte, a aceleração do processo de particionamento dos blocos.

Podemos observar pela tabela que os trabalhos relacionados utilizam versões diversas do software de referencia, o que pode não representar uma comparação tão fidedigna. Porém, como podemos observar, todos os trabalhos relacionados conseguem uma redução de tempo de codificação bem mais expressiva, com um custo na eficiência de compressão maior que 1%. Também vale ressaltar que estes trabalhos relacionados focam em reduzir, além dos particionamentos quaternários, os particionamentos binários e ternários também. Alguns destes trabalhos como (LEI et al., 2019) e (SALDANHA et al., 2020b) focam apenas nos particionamentos binários e ternários e seriam, portanto, soluções coexistentes com a solução proposta, podendo compor uma solução conjunta que resulte em uma maior redução de tempo.

No geral, a solução proposta não reduz o tempo de codificação com uma magnitude maior que os trabalhos relacionados. Entretanto, este resultado já era esperado, já que a solução não foca em todos os tipos de particionamento introduzidos no VVC. Além disso, a solução foi implementada e testada em uma versão mais recente do software de referência do VVC, que vem sendo otimizado a cada nova versão para redução de tempo de codificação. É importante destacar, porém, o baixo valor de *BD-rate* da solução proposta, que reflete um pequeno impacto negativo na eficiên-

cia de codificação. Este resultado permite concluir que é possível formar soluções compostas com trabalhos que explorem os novos particionamentos binário e ternário sem acrescentar perdas significativas na eficiência de codificação, mas ainda assim reduzindo o tempo de processamento.

6 CONCLUSÃO

A redução do tempo de codificação em codificadores atuais é um tema relevante no tópico de codificação de vídeo. Portanto, a codificação de quadros Intra é um dos pontos de grande interesse de pesquisa, uma vez que representa uma das etapas que demandam tempo de processamento. Além disso, um bom desempenho da codificação intra é importante pois estes quadros podem servir como referência para quadros Inter, dependendo do tipo de configuração.

Por outro lado, para obter uma eficiência de codificação superior ao seu antecessor, o VVC adicionou à sua etapa de particionamento das CUs novos particionamentos retangulares, com divisões binárias e ternárias em uma nova estrutura de particionamento que denominamos árvore MTT. Tal tarefa não é suportada nos padrões de codificação anteriores, devido ao seu alto requisito em processamento. Quando analisada esta estrutura, foi observado que os novos particionamentos podem representar até 90% do tempo total de codificação, sendo, portanto, a etapa que mais demanda esforço computacional do codificador. Também foi verificado que a perda em eficiência de codificação quando desativamos estes novos particionamentos em média está próximo a 6%.

Para reduzir o tempo de codificação presente nos particionamentos do codificador VVC, este trabalho apresentou um esquema baseado em aprendizado de máquina, mais especificamente utilizando redes neurais convolucionais. O esquema proposto define uma maneira de predizer a profundidade máxima na árvore de particionamentos, especificamente para a árvore quaternária. Os modelos propostos apresentaram uma acurácia de até 68,31% para o conjunto de testes avaliado. Como resultados finais, conseguimos atingir uma redução no tempo de codificação de 12,73% com uma perda na eficiência de compressão mensurada como *BD-rate* de 0,432%.

Em trabalhos futuros, é possível elaborar soluções que permeiam a solução apresentada nesta dissertação, como, por exemplo, explorar de uma forma diferente a saída das redes, utilizando as duas classificações com maior probabilidade, e, dependendo desta probabilidade, decidir se o algoritmo deve agir de uma forma mais agressiva ou mais conservadora. Assim como é possível aliar a esta solução uma

solução à parte para os novos particionamentos binários e ternários, somando assim o ganho em redução no tempo de codificação.

REFERÊNCIAS

AL RABBANI ALIF, M.; AHMED, S.; HASAN, M. Isolated Bangla handwritten character recognition with convolutional neural network. In: 2017. **Anais...** [S.l.: s.n.], 2017. p.1–6.

BENGIO, Y.; COURVILLE, A.; VINCENT, P. Representation learning: A review and new perspectives. **IEEE transactions on pattern analysis and machine intelligence**, [S.l.], v.35, n.8, p.1798–1828, 2013.

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

BORDES, A.; CHOPRA, S.; WESTON, J. Question answering with subgraph embeddings. **arXiv preprint arXiv:1406.3676**, [S.l.], 2014.

BOTTOU, L. Large-scale machine learning with stochastic gradient descent. In: **Proceedings of COMPSTAT'2010**. [S.l.]: Springer, 2010. p.177–186.

BROSS, B. et al. CE3: Multiple reference line intra prediction (Test 1.1. 1 1.1. 2 1.1. 3 and 1.1. 4). In: JVET 12TH MEETING, JVET-L0283, 2018. **Anais...** [S.l.: s.n.], 2018.

CERVEIRA, A.; AGOSTINI, L.; ZATT, B.; SAMPAIO, F. Memory Profiling of H.266 Versatile Video Coding Standard. In: IEEE INTERNATIONAL CONFERENCE ON ELECTRONICS, CIRCUITS AND SYSTEMS (ICECS), 2020., 2020. **Anais...** [S.l.: s.n.], 2020. p.1–4.

CHOLLET, F. **Deep learning with Python**. [S.l.]: Simon and Schuster, 2017.

COLLOBERT, R. et al. Natural language processing (almost) from scratch. **Journal of machine learning research**, [S.l.], v.12, n.ARTICLE, p.2493–2537, 2011.

DAHL, G. E.; SAINATH, T. N.; HINTON, G. E. Improving deep neural networks for LVCSR using rectified linear units and dropout. In: IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING, 2013., 2013. **Anais...** [S.l.: s.n.], 2013. p.8609–8613.

DE-LUXáN-HERNáNDEZ, S. et al. An Intra Subpartition Coding Mode for VVC. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 2019., 2019. **Anais...** [S.l.: s.n.], 2019. p.1203–1207.

DONG, X.; SHEN, L.; YU, M.; YANG, H. Fast Intra Mode Decision Algorithm for Versatile Video Coding. **IEEE Transactions on Multimedia**, [S.l.], p.1–1, 2021.

DUDA, R. O.; HART, P. E. et al. **Pattern classification and scene analysis**. [S.l.]: Wiley New York, 1973. v.3.

F. BOSSEN J. BOYCE, X. L. V. S. K. S. JVET common test conditions and software reference configurations for SDR video. **Preview document JVET-N1010 for Geneva meeting**, [S.l.], 2019.

FU, T.; ZHANG, H.; MU, F.; CHEN, H. Fast CU Partitioning Algorithm for H.266/VVC Intra-Frame Coding. In: IEEE INTERNATIONAL CONFERENCE ON MULTIMEDIA AND EXPO (ICME), 2019., 2019. **Anais...** [S.l.: s.n.], 2019. p.55–60.

GLOROT, X.; BORDES, A.; BENGIO, Y. Deep sparse rectifier neural networks. In: OF THE FOURTEENTH INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE AND STATISTICS, 2011. **Proceedings...** [S.l.: s.n.], 2011. p.315–323.

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.

HINTON, G. E.; OSINDERO, S.; TEH, Y.-W. A fast learning algorithm for deep belief nets. **Neural computation**, [S.l.], v.18, n.7, p.1527–1554, 2006.

HINTON, G. et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. **IEEE Signal processing magazine**, [S.l.], v.29, n.6, p.82–97, 2012.

HOANG, D. T.; LONG, P. M.; VITTER, J. S. Efficient cost measures for motion estimation at low bit rates. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.8, n.4, p.488–500, 1998.

IANDOLA, F. N. et al. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. **arXiv preprint arXiv:1602.07360**, [S.l.], 2016.

IECJCT1. **High Efficiency Video Coding (HEVC) Test Model 16 (HM 16) Encoder Description**. [S.l.]: (ISO/IECJCT1/SC29/WG11), 2014.

INDEX, C. V. networking. Forecast and methodology, 2016-2021, white paper. **San Jose, CA, USA**, [S.l.], v.1, 2016.

J. CHEN Y. YE, S. K. **Algorithm description for Versatile Video Coding and Test Model 6 (VTM 6)**. [S.l.]: (Joint Video Experts Team), 2019.

JAMES BRUCE MARTA MRAK, R. W. **Versatile Video Coding**: a Next-generation Video Coding Standard. Disponível em: <https://www.bbc.co.uk/rd/blog/2019-05-av1-codec-streaming-processing-hevc-vvc>. Acesso em: 2021-08-06.

JANSCHE, M. Maximum expected F-measure training of logistic regression models. In: OF THE CONFERENCE ON HUMAN LANGUAGE TECHNOLOGY AND EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING, 2005. **Proceedings...** [S.l.: s.n.], 2005. p.692–699.

JCTVC. **Recommendation ITU T H.265 High Efficiency Video Coding**.

JEAN, S.; CHO, K.; MEMISEVIC, R.; BENGIO, Y. On using very large target vocabulary for neural machine translation. **arXiv preprint arXiv:1412.2007**, [S.l.], 2014.

JVET. **VVC Draft 5 (JVET-N1001)**.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. **Advances in neural information processing systems**, [S.l.], v.25, p.1097–1105, 2012.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. **Advances in neural information processing systems**, [S.l.], v.25, p.1097–1105, 2012.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **nature**, [S.l.], v.521, n.7553, p.436–444, 2015.

LEI, M. et al. Look-Ahead Prediction Based Coding Unit Size Pruning for VVC Intra Coding. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 2019., 2019. **Anais...** [S.l.: s.n.], 2019. p.4120–4124.

LI, T. et al. DeepQTMT: A Deep Learning Approach for Fast QTMT-Based CU Partition of Intra-Mode VVC. **IEEE Transactions on Image Processing**, [S.l.], v.30, p.5377–5390, 2021.

MANSRI, I. et al. Comparative Evaluation of VVC, HEVC, H.264, AV1, and VP9 Encoders for Low-Delay Video Applications. In: FOURTH INTERNATIONAL CONFERENCE ON MULTIMEDIA COMPUTING, NETWORKING AND APPLICATIONS (MCNA), 2020., 2020. **Anais...** [S.l.: s.n.], 2020. p.38–43.

NASIRI, F. et al. Prediction-Aware Quality Enhancement of VVC Using CNN. In: IEEE INTERNATIONAL CONFERENCE ON VISUAL COMMUNICATIONS AND IMAGE PROCESSING (VCIP), 2020., 2020. **Anais...** [S.l.: s.n.], 2020. p.310–313.

PARK, S.-H.; KANG, J.-W. Context-Based Ternary Tree Decision Method in Versatile Video Coding for Fast Intra Coding. **IEEE Access**, [S.l.], v.7, p.172597–172605, 2019.

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

RAINA, R.; MADHAVAN, A.; NG, A. Y. Large-scale deep unsupervised learning using graphics processors. In: OF THE 26TH ANNUAL INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 2009. **Proceedings...** [S.l.: s.n.], 2009. p.873–880.

RAMASUBRAMONIAN, A. et al. CE3-1.6: On 1$\times$ N and 2$\times$ N subblocks of ISP. **Document JVET-O0106 of JVET**, [S.l.], 2019.

RICHARDSON, I. E. **Video codec design**: developing image and video compression systems. [S.l.]: John Wiley & Sons, 2002.

RICHARDSON, I. E. **H. 264 and MPEG-4 video compression**: video coding for next-generation multimedia. [S.l.]: John Wiley & Sons, 2004.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. **nature**, [S.l.], v.323, n.6088, p.533–536, 1986.

RUSSEL, S.; NORVIG, P. **Artificial Intelligence**: A Modern Approach. 3.ed. Upper Saddle River, NJ: Prentice Hall, 2010. (Series in Artificial Intelligence).

SALDANHA, M.; SANCHEZ, G.; MARCON, C.; AGOSTINI, L. Complexity Analysis Of VVC Intra Coding. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 2020., 2020. **Anais...** [S.l.: s.n.], 2020. p.3119–3123.

SALDANHA, M.; SANCHEZ, G.; MARCON, C.; AGOSTINI, L. Fast Partitioning Decision Scheme for Versatile Video Coding Intra-Frame Prediction. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS (ISCAS), 2020., 2020. **Anais...** [S.l.: s.n.], 2020. p.1–5.

SENGUPTA, A. et al. Going deeper in spiking neural networks: VGG and residual architectures. **Frontiers in neuroscience**, [S.l.], v.13, p.95, 2019.

SERMANET, P.; KAVUKCUOGLU, K.; CHINTALA, S.; LECUN, Y. Pedestrian detection with unsupervised multi-stage feature learning. In: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 2013. **Proceedings...** [S.l.: s.n.], 2013. p.3626–3633.

STEHMAN, S. V. Selecting and interpreting measures of thematic classification accuracy. **Remote sensing of Environment**, [S.l.], v.62, n.1, p.77–89, 1997.

SULLIVAN, G.; WIEGAND, T. Rate-distortion optimization for video compression. **IEEE Signal Processing Magazine**, [S.l.], v.15, n.6, p.74–90, 1998.

SZEGEDY, C. et al. Going deeper with convolutions. In: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 2015. **Proceedings...** [S.l.: s.n.], 2015. p.1–9.

Takamura, S. Versatile Video Coding: a Next-generation Video Coding Standard. **NTT Technical Review**, [S.l.], v.17, June 2019.

TISSIER, A. et al. Complexity Reduction Opportunities in the Future VVC Intra Encoder. In: IEEE 21ST INTERNATIONAL WORKSHOP ON MULTIMEDIA SIGNAL PROCESSING (MMSP), 2019., 2019. **Anais...** [S.l.: s.n.], 2019. p.1–6.

TISSIER, A. et al. CNN Oriented Complexity Reduction Of VVC Intra Encoder. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 2020., 2020. **Anais...** [S.l.: s.n.], 2020. p.3139–3143.

ZAGORUYKO, S.; KOMODAKIS, N. Wide residual networks. **arXiv preprint arXiv:1605.07146**, [S.l.], 2016.

ZHAO, X. et al. Transform coding in the VVC standard. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], 2021.