UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

EDUARDO BONOW BÖSEL

Avaliação de Modelos de Predição para Aproximações da Soma das Diferenças Transformadas Absolutas para Codificadores de Vídeo

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Prof. Dr. Cláudio Diniz
Co-orientador: Prof. Dr. Mateus Grellert
Co-orientador: Prof. Dr. Leonardo Soares

Porto Alegre
2025

RESUMO

O processo de codificação de vídeo é complexo e computacionalmente custoso. Muito tempo e esforço são investidos em pesquisas que visam de alguma forma melhorar esse processo, tornando-o mais rápido de ser executado e mais eficiente quanto a sua capacidade de compressão de dados. Dessa forma, esse trabalho tem como objetivo explorar e avaliar, através de conceitos de Aprendizado de Máquina, a possibilidade de desenvolver um modelo de predição capaz de escolher, durante o processo de codificação do codificador Vesatile Video Coding (VVC), qual aproximação do cálculo da Soma das Diferenças Transformadas Absolutas (SATD) deverá ser utilizada para melhorar a complexidade do cálculo. Uma pré-análise foi realizada em dados previamente coletados sobre o desempenho geral das aproximações do cálculo para selecionar os melhores candidatos. Dois conjuntos de dados distintos foram criados, variando em método de coleta e variáveis coletadas, e visando momentos e modos distintos de realizar a predição. Foram desenvolvidos modelos de predição utilizando cinco algoritmos de Aprendizado Supervisionado, afim de escolher o melhor equilíbrio entre acurácia e tempo necessário para a predição. Apesar dos modelos desenvolvidos obterem um desempenho abaixo do esperado, foi possível analisar e discutir algumas alternativas da forma que o assunto pode ser abordado.

Palavras-chave: Codificação de vídeo. Versatile Video Coding. SATD. Aprendizado de Máquina.

Evaluation of Prediction Models for Approximations of the Sum of Absolute Transformed Differences for Video Encoder

ABSTRACT

The video encoding process is complex and computationally costly. Significant time and effort are invested in research aimed at improving this process, making it faster to execute and more efficient in terms of data compression capability. Therefore, this work aims to explore and evaluate, using Machine Learning concepts, the possbility of developing a prediction model capable of selecting, during the encoding process of the Versatile Video Coding (VVC) encoder, which approximation of the Sum of Absolute Transformed Differences (SATD) calculation should be applied to improve calculation complexity. A preliminary analysis was conducted using previously collected data on the overall performance of the calculation approximations to select the best candidates. Two distinct datasets were created, varying in data collection methods and collected variables, targeting different moments and modes of prediction. Prediction models were developed using five Supervised Learning algorithms to select the one with the best balance between accuracy and prediction time. Although the developed models performed below expectations, it was possible to analyze and discuss some alternatives for how the topic can be approached.

Keywords: Video coding. Versatile Video Coding. SATD. Machine Learning.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AVC	*Advanced Video Coding*
BD-Rate	*Bjøntegaard Delta Bit Rate*
Cb	Crominância azul
Cg	Crominância verde
CIF	*Common Intermediate Format*
CODEC	*en**CO**der/**DEC**ode*
Cr	Crominância vermelha
CTC	*Common Test Conditions*
DSCQS	*Double Stimulus Continuous Quality Scale*
DVD	*Digital Versatile Disc*
FPS	*Frames Per Second*
HD	*High Definition*
HDR	*High Dynamic Range*
HEVC	*High-Efficiency Video Coding*
IEC	*International Electrotechnical Commission*
ISO	*International Organization for Standardization*
ITU	*International Telecommunication Union*
ITU-T	*Telecommunication Standardization Sector of ITU*
JVET	*Joint Video Experts Team*
K-NN	*K-Nearest Neighbors*
KFCV	*K-Fold Cross Validation*
ML	*Machine Learning*
MPEG	*ISO/IEC Moving Picture Experts Group*
MSE	*Mean Squared Error*

PSNR	*Peak Signal to Noise Ratio*
QP	*Quantization Parameter*
RGB	*Red*, *Green* e *Blue*
SAD	*Sum of Absolute Differences*
SATD	*Sum of Absolute Transformed Differences*
SD	*Standard Definition*
UHD	*Ultra High Definition*
VCEG	*ITU-T Video Coding Experts Group*
VS Code	*Visual Studio Code*
VTM	*VVC Test Model*
VVC	*Versatile Video Coding*
YCrCb	Luminância, crominância vermelha e crominância azul

SUMÁRIO

1 INTRODUÇÃO

Ao longo dos anos, juntamente ao aumento da capacidade e velocidade de unidades de armazenamento de informação digital e ao aumento de acesso à internet de maior velocidade, houve também um crescimento da capacidade de reprodução de vídeos digitais quanto a qualidade e resolução. Televisões com capacidade de reprodução de resoluções em Ultra-Alta Definição — *Ultra High Definition* (UHD) (maiores que 1920 *pixels* por 1080 *pixels*), por exemplo, estão cada vez mais comuns, e com esse avanço na capacidade de resolução de imagens, a necessidade de disponibilidade e capacidade de transmissão de conteúdos em resoluções cada vez maiores. Apenas 10 minutos de vídeo sem qualquer tipo de compressão, no formato *Sony 4K* (4096 *pixels* por 2160 *pixels*) com 30 quadros por segundo e 24 *bits* por *pixel* necessita 477 GB para ser armazenado (Diniz, 2015), o que dificulta o armazenamento e a transmissão de vídeos. Por isso, a compressão de vídeos através de codificadores de vídeos se torna necessária.

A compressão de vídeos através da codificação é um processo de alto custo computacional e é objeto de constante pesquisa e evolução. O processo tem o objetivo de reduzir a quantidade de *bits* necessários para representar o vídeo digital, sem que haja uma perda significativa de informações nessa representação. Para que os processos de codificação e decodificação seja utilizado de forma disseminada, padrões na realização desses processos são necessários. Um exemplo são os padrões definidos ao longo dos anos pelo Setor de Normatização das Telecomunicações — *International Telecommunication Union* (ITU) e pela Organização Internacional de Normalização — *International Organization for Standardization* (ISO), como o *Advanced Video Coding* (AVC), o *High-Efficiency Video Coding* (HEVC) e atualmente o *Versatile Video Coding* (VVC).

Pensando na otimização do processo de codificação de um ponto de vista de tempo, eficiência e energia, a computação aproximada é uma abordagem que já foi utilizada. Por exemplo, na simplificação do cálculo entre matrizes no processo de calcular a Soma das Diferenças Transformadas Absolutas — *Sum of Absolute Transformed Differences* (SATD) utilizando a Transformada de Hadamard, através da eliminação de parte das operações aritméticas (Lima et al., 2021).

Assim, utilizando os conceitos de Aprendizado de Máquina — *Machine Learning* (ML), esse trabalho tem como objetivo explorar e avaliar as possibilidades da utilização de modelos de predição. Utilizando cinco algoritmos diferentes de Aprendizado Supervisionado, treinados utilizando dois Conjuntos de Dados distintos e de desenvolvimento

próprio, os modelos devem escolher entre quatro possibilidades de aproximação do cálculo do SATD e a Soma das Diferenças Absolutas — *Sum of Absolue Differences* (SAD), qual deve ser utilizado para realizar a estimativa da distorção de cada bloco 8×8 durante o processo de codificação de vídeo, utilizando um modelo de teste do padrão de codificação VVC.

O trabalho está organizado em seis capítulos, sendo este o Capítulo 1, que serve de introdução ao trabalho. O Capítulo 2 discorre sobre a base teórica utilizada e trabalhos relacionados ao tema. O Capítulo 3 apresenta a metodologia utilizada para a realização do trabalho e as ferramentas utilizadas. No Capítulo 4, é apresentado o processo de desenvolvimento dos modelos de predição. No Capítulo 5, encontram-se as avaliações obtidas dos modelos de predição treinados. Por fim, no Capítulo 6, são expostas as conclusões obtidas e as possibilidades para trabalhos futuros.

2 REFERENCIAL TEÓRICO E TRABALHOS RELACIONADOS

Neste capítulo, são apresentados os conceitos teóricos do trabalho relativos a vídeo digital, codificação de vídeo e ML. No final do capítulo, são discutidos trabalhos relacionados ao tema.

2.1 Vídeo Digital

Um vídeo digital é formado por uma sequência de imagens (comumente chamadas de quadros ou *frames*) exibidas em uma determinada frequência; é a representação de uma cena visual do mundo real ou natural, amostrada espacialmente e temporalmente (Richardson, 2010).

Cada *frame* que constitui a amostragem espacial de um vídeo digital possui uma resolução que é definida pela quantidade de *pixels* em cada linha e cada coluna da imagem. Segundo Richardson (2010), o *pixel* (uma abreviação de *picture element*) é o elemento da imagem que contém as informações de brilho ou luminância e cor de cada ponto da imagem da amostra. Esses pontos são a representação por amostragem de uma cena visual.

A amostragem temporal do vídeo digital é dada pelo intervalo de tempo que os *frames* são coletados pela amostragem espacial. Quanto maior for a amostragem temporal, mais *frames* serão coletados e mais suave será a sensação de movimento (Richardson, 2010). Essa amostragem, medida em quadros por segundo — *Frames Per Second* (FPS), varia tipicamente entre 24 e 120 FPS, sendo que valores inferiores resultam em vídeos digitais muito irregulares, ficando mais evidente a falta de fluidez do vídeo.

2.1.1 Espaço de Cores

Enquanto uma imagem monocromática necessita apenas de um número para indicar o brilho ou luminância para cada *pixel*, uma imagem colorida necessita de pelo menos três números para representar a cor de cada *pixel*. O método de representar brilho ou luminância e cor é descrito como espaço de cor (Richardson, 2010). Uma amostra é a informação de luminância ou crominância de 1 *pixel*.

2.1.1.1 Espaço de cores RGB

No espaço de cor RGB, a cor de um *pixel* é representado por três números que indicam a proporção relativa da intensidade de cada uma das três cores-luz primárias: vermelho (***R**ed*), verde (***G**reen*) e azul (***B**lue*). Combinando essas três cores em diferentes proporções, é possível criar qualquer cor (Richardson, 2010).

2.1.1.2 Espaço de cores YCrCb

Já o espaço de cor YCrCb utiliza o fato que o sistema visual humano é menos sensível para cor do que para luminância (Richardson, 2010). Por isso, é mais eficiente separar a luminância de uma imagem e representa-lá com uma resolução maior que as informações de cor.

Nesse caso, Y é o componente de luminância da imagem e a informação da cor pode ser representada através dos componentes de crominância, onde cada componente será a diferença entre o componente de cor (R,G e B) e a luminância (Y):

$$\begin{aligned} Cr &= R - Y \\ Cg &= G - Y \\ Cb &= B - Y \end{aligned} \tag{2.1}$$

Como a soma dos componentes Cr, Cg e Cb é constante (sempre será 1, ou 100%), é necessário guardar e transmitir apenas dois dos componentes de crominância, pois sempre será possível calcular o terceiro. Dessa forma, no espaço YCrCb, os elementos utilizados são a luminância (Y), a crominância vermelha (Cr) e a crominância azul (Cb).

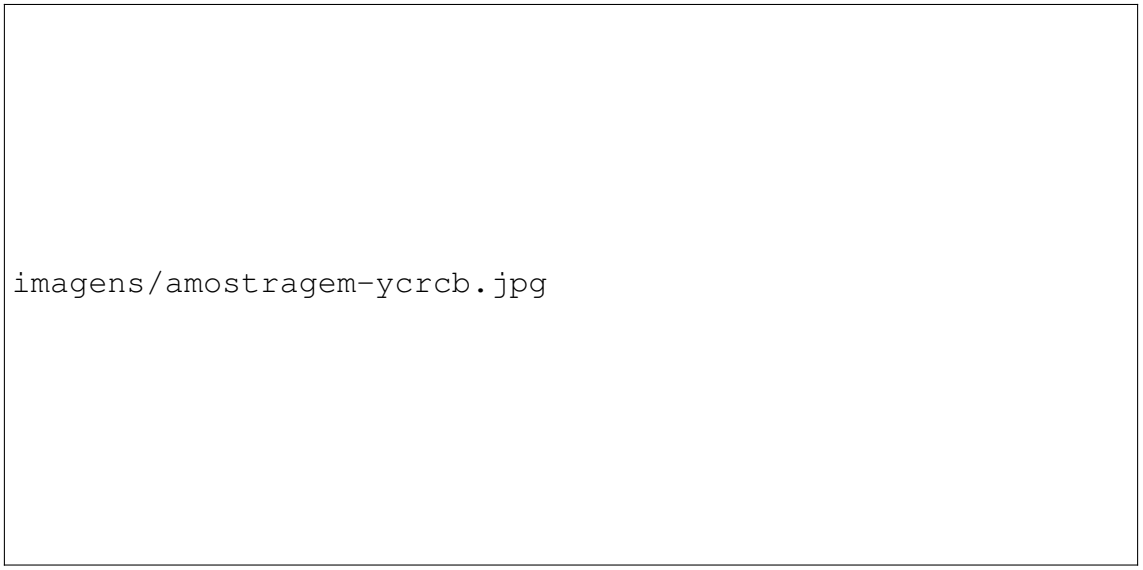
2.1.1.3 Amostragem do espaço YCrCb

Utilizando um sistema de amostragem, o espaço YCrCb consegue demonstrar sua eficiência ao diminuir até pela metade a quantidade de *bits* necessários para representar a mesma imagem (Richardson, 2010). As principais amostragens do espaço YCrCb são 4:4:4, 4:2:2 e 4:2:0. Na Figura 2.1 podemos entender um pouco melhor como cada uma delas funciona.

Na amostragem 4:4:4, para cada 4 *pixels* horizontais contendo informação de luminância, nós temos 4 *pixels* com informações de crominância vermelha e 4 *pixels* com informações de crominância azul. Nessa amostragem, as informações originais são man-

tidas e cada *pixels* utilizará três números para representá-los.

Figura 2.1 – Amostragens de cores no espaço YCrCb



Fonte: Wikipedia[1]

Na amostragem 4:4:4, para cada 4 *pixels* horizontais contendo informação de luminância, nós temos 4 *pixels* com informações de crominância vermelha e 4 *pixels* com informações de crominância azul. Nessa amostragem, as informações originais são mantidas e cada *pixels* utilizará três números para representá-los.

Na amostragem 4:2:2, ao invés de termos 4 *pixels* com informações para cada componente de crominância ao longo de 4 *pixels* horizontais, possuímos apenas 2 *pixels* com informações que são replicados para os *pixels* vizinhos horizontalmente. Nessa amostragem, o vídeo irá possuir $2/3$ do tamanho original (caso utilizado a amostragem 4:4:4), pois cada camada de crominância terá metade do tamanho em *bits*.

Na amostragem 4:2:0, além de replicarmos a informação de crominância para os vizinhos horizontais, as informações são replicadas para os vizinhos verticais, tornando as informações de 1 *pixel* responsável pela crominância de 4 *pixels*. Nesse caso, o vídeo terá metade do tamanho quando comparado com a amostragem 4:4:4, pois cada camada de crominância terá $1/4$ do tamanho em *bits*.

Como dito anteriormente, o sistema visual humano é mais sensível à luminância que à cor, e por isso é possível utilizar a amostragem 4:2:0 sem uma perda perceptiva de qualidade. A amostragem 4:2:0 é amplamente utilizada em aplicações de consumo como vídeo conferência, televisões digitais e DVDs, por exemplo (Richardson, 2010).

[1] Autor: Stevo-88, disponível em: https://en.wikipedia.org/wiki/Chroma_subsampling

2.1.2 Formatos de Vídeo

Codificadores de vídeo são capazes de trabalhar com uma grande variedade de resoluções de quadro do vídeo, mas, na prática, é comum os vídeos serem capturados ou convertidos para formatos intermediários padronizados antes do processo de codificação (Richardson, 2010), como o Formato Intermediário Comum — *Common Intermediate Format* (CIF) e suas variações (Tabela 2.1).

Tabela 2.1 – Formato CIF

Formato	**Resolução (horizontal x vertical)**
Sub-QCIF	128 x 96
QCIF	176 x 144
CIF	352 x 288
4CIF	704 x 576

Fonte: Richardson (2010)

Além do CIF, o Setor de Normatização das Telecomunicações da ITU — *Telecommunication Standardization Sector of ITU* (ITU-T), é responsável por desenvolver padrões conhecidos como *ITU-T Recommendations*, definindo faixas de resoluções (em *pixels*) para enquadrar em formatos como a Definição Padrão — *Standard Definition* (SD), a Alta Definição — *High Definition* (HD) e UHD como é possível ver na Tabela 2.2.

Tabela 2.2 – Formatos SD, HD e UHD

Formato	**Faixa vertical (em *pixels*)**
Abaixo de SD	180-270
SD	360-540
HD	720-1080
Acima de HD (UHD)	1440-2160

Fonte: ITU-T P.1204.3[2]

2.1.3 Qualidade do Vídeo

Para que seja possível especificar, avaliar e comparar sistemas de transmissão de vídeo, é necessário determinar a qualidade da imagem de um vídeo (Richardson, 2010). Medir a qualidade de um vídeo é algo difícil e impreciso, dado que muito do que temos como qualidade, normalmente, é algo muito mais subjetivo do que objetivo.

[2]Disponível em: https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-P.1204.3-202001-I!!PDF-E&type=items

Alguns dos fatores externos que influenciam a qualidade subjetiva de um vídeo são a interação entre o sistema visual humano e o cérebro, o ambiente onde a avaliação ocorre e a composição das imagens do vídeo (Richardson, 2010). Além disso, existem testes que tentam avaliar e padronizar essas medidas, como o método da Escala de Qualidade Contínua de Estímulo Duplo — *Double Stimulus Continuous Quality Scale*[3] (DSCQS), que busca avaliar a qualidade do vídeo através de um processo de avaliação comparativa entre vídeos.

A complexidade e custo no processo de avaliar a qualidade subjetiva de um vídeo torna mais atraente a ideia de realizarmos, através de algoritmos, uma medição de qualidade de forma mais objetiva (Richardson, 2010). Uma medida amplamente utilizada, por exemplo, é a Razão Pico de Sinal-Ruído — *Peak Signal to Noise Ratio* (PSNR), que é feita na escala logarítmica e depende do Erro Quadrático Médio — *Mean Squared Error* (MSE) entre uma imagem e essa mesma imagem com algum tipo de ruído ou processamento.

2.2 Codificação de Vídeo

Na segunda parte do capítulo, são apresentados os conceitos utilizados neste trabalho relativos à área de Codificação de Vídeo. Falaremos sobre conceitos básicos do processo de codificação de vídeo, fluxo de um codificador de vídeo, padrão de codificação VVC, traremos alguns conceitos de qualidade de imagem e compressão de vídeos codificados e, também, sobre métricas de similaridade de blocos.

2.2.1 Conceitos Básicos da Codificação de Vídeo

Um *frame* de um vídeo em resolução HD (1280 x 720), onde cada amostra ocupa 8 *bits*, totalizando 24 bits por *pixel* em uma amostragem, utilizando o espaço YCrCb e uma amostragem 4:2:0, possui o seguinte tamanho em *bits*:

$$\begin{aligned} 1280 * 720 * 8 &= 7372800 && \text{(\textit{bits} para a camada } Y\text{)} \\ 640 * 360 * 8 &= 1843200 && \text{(\textit{bits} para cada camada Cr e Cb)} \\ 7372800 + 2 * (1843200) &= 11059200 && \text{(tamanho total em \textit{bits})} \end{aligned} \tag{2.2}$$

[3]Procedimento presente no ITU-R BT.500-11, disponível em: https://www.itu.int/dms_pubrec/itu-r/rec/bt/R-REC-BT.500-11-200206-S!!PDF-E.pdf

Isso significa que um *frame* possui aproximadamente 11 *Mbits*, e consequentemente, um segundo desse vídeo, em 30 *FPS*, resultaria em aproximadamente 331 *Mbits*. Levando em conta um filme de 1 hora e 30 minutos de duração, ele ocuparia 1,79 *Tbits*, ficando clara a necessidade de uma forma de compressão de dados para que seja feito o armazenamento e a transferência de vídeos (Richardson, 2010).

Essa compressão, ou codificação, envolve um par complementar de sistemas: o codificador (*encoder*) e o decodificador (*decoder*). O *encoder* faz a compressão dos *bits* de um arquivo visando diminuir o seu tamanho, enquanto o *decoder* faz a descompressão e retorna os dados em seu formato para visualização. O par é normalmente chamado de CODEC (*en**CO**der*/*DEC**oder*).

A compressão de dados é feita através da remoção de redundância desses dados, e pode ser uma compressão com perdas ou sem perdas. A compressão sem perdas resultará numa cópia fiel dos dados após o processo de descompressão. Já, na compressão com perdas, os dados não serão iguais após o processo de descompressão, em compensação torna o processo muito mais eficaz quanto a compressão de dados (Richardson, 2010).

Segundo Richardson (2010), a codificação de vídeos utiliza o conceito de qualidade subjetiva para poder realizar uma compressão com perdas, sem que isso afete a percepção de qualidade do vídeo após o processo de decodificação.

A maioria dos métodos de codificação utilizam a redundância espacial e temporal em um vídeo para a compressão. Em termos espaciais, existe um alto grau de correlação e similaridade entre *pixels* próximos, resultando em áreas que possuem dados parecidos. Já em termos temporais, existe uma alta similaridade entre *frames* que foram capturados temporalmente próximos. *Frames* sucessivos tendem a ter um alto nível de correlação e similaridades, principalmente em casos com uma amostragem temporal maior (FPS maiores).

A maioria dos codificadores de vídeo dos últimos anos, apesar de suas diferenças, são baseados no mesmo modelo de CODEC. Esse modelo consiste no uso de predição e/ou compensação de movimento baseado em bloco, transformadas, quantização e codificação de entropia (Richardson, 2010).

2.2.2 Fluxo da Codificação de Vídeo

Todos os padrões de codificação de vídeo desde o H.261 em 1988 são baseados no mesmo princípio de codificação híbrido (Liou, 1991). O termo híbrido faz referência

a combinação de duas maneiras de reduzir a redundância em um sinal de vídeo, sendo elas a predição e a transformada com quantização do residual da predição (Bross et al., 2021a).

Na Figura 2.2 é possível observar o processo de codificação e decodificação de sinal de vídeo. O processo de codificação pode ser divido em seis etapas: Divisão em Blocos, Predição, Transformadas, Quantização, Codificação de Entropia e Filtros de Laço.

Figura 2.2 – Diagrama de blocos de um CODEC híbrido

Fonte: Bross et al. (2021a)

Na etapa de Divisão em Blocos, o *frame* é dividido em blocos menores. A forma que essa divisão é feita mudou ao longo do tempo, tanto em relação ao tamanho dos blocos quanto em relação ao seus formatos, variando entre codificadores. Cada vez mais essa etapa tem sido foco de otimização de padrões atuais. Utilizando blocos menores se tem uma divisão mais precisa, que atende as necessidade de maior detalhamento na cena, em troca de uma maior taxa de *bits* para representar o particionamento. Em contrapartida, blocos maiores proporcionam uma redução na taxa de bits para representar o particionamento, ao custo de uma menor qualidade para representação do mesmo (Bross et al., 2021a).

A etapa de Predição é formada por duas etapas: Predição Intra-quadro e Predição Inter-quadros, que reduzem a redundância espacial e temporal, respectivamente. A primeira delas utiliza blocos que já foram codificados no quadro como referência para a predição, e a segunda, conta com duas etapas menores: a etapa de Estimação de Movi-

mento e a etapa de Compensação de Movimento. Utilizando quadros já codificados como referência para a predição, a Estimação de Movimento tenta encontrar equivalências entre os blocos de quadros sequenciais, e assim, gerar um vetor de movimento, baseado em onde o bloco estava no quadro de referência e onde ele está no quadro atual. A Compensação de Movimento faz o processo inverso, a partir de um vetor de movimento ela reconstrói o bloco atual com base no quadro de referência.

Na etapa de Transformadas, o resíduo formado pela diferença entre o bloco original e o bloco predito é levado do domínio espacial para um domínio transformado, normalmente o domínio da frequência (Bross et al., 2021a).

Na etapa de Quantização, os coeficientes resultantes da etapa de Transformada são simplificados afim de diminuir a quantidade de dados necessários para representá-los. O grau de quantização é determinado por um valor chamado Parâmetro de Quantização — *Quantization Parameter* (QP), que irá determinar o passo da quantização. Um QP menor significa que os valores quantizados estarão mais próximos do valor original, e por consequência, mantendo uma qualidade mais próxima à original, com um taxa de compressão menor. Já um QP maior significa que os valores quantizados sofreram mais distorções em relação aos valores originais, em troca de uma taxa de compressão maior.

Na etapa de Codificação de Entropia, algoritmos são aplicados para, sem perda, realizar uma compressão na sequência de *bits*, baseado em propriedades estatísticas de frequência relativa (Bross et al., 2021a).

Por fim, na etapa de Filtros de Laço, são aplicados filtros nos quadros reconstruídos utilizando os dados quantizados, através de um processo inverso de quantização e transformada, juntamente com os quadros preditos. Os filtros são utilizados para diminuir os impactos do processo de quantização e diminuir o número de artefatos e erros de reconstrução (Bross et al., 2021a). Esses quadros são os quadros utilizados como referência na Predição Inter-Quadros.

2.2.3 Versatile Video Coding

O VVC, padronizado no ITU-T como *Recommendation H.266* e na ISO e na Comissão Eletrotécnica Internacional — *International Electrotechnical Commission* (IEC) como *International Standard 23090-3* (MPEG-I Part 3), é a nova geração de padrões internacionais de codificação de vídeo desenvolvida em conjunto pelo *ITU-T Video Coding Experts Group* (VCEG) e pelo *ISO/IEC Moving Picture Experts Group* (MPEG) (Bross

et al., 2021b) sob o nome *Joint Video Experts Team* (JVET).

O predecessor do VVC, o HEVC (também conhecido como H.265 e MPEG-H Part 2) foi finalizado em 2013, oferecendo aproximadamente 50% na redução da taxa de *bits* em relação ao seu predecessor, o AVC (também conhecido como H.264 e MPEG-4 Part 10). O padrão do VVC apresenta mais de 30% na redução da taxa de *bits* em relação ao HEVC, com implementações que podem chegar a 50% (Bross et al., 2021b).

Além da maior capacidade de compressão, outras características do VVC incluem melhorias em vídeos com resoluções altas (maiores que HD) e em Grande Alcance Dinâmico — *High Dynamic Range* (HDR) com a possibilidade de estruturas de blocos maiores e mais flexíveis, facilitação de *streaming* com latência baixa e eficiência na codificação de vídeos em 360º (Bross et al., 2021a).

2.2.4 Qualidade de Imagem e Compressão de Vídeos

Conforme citado anteriormente, segundo Richardson (2010), para compararmos dois sistemas de transmissão e/ou codificação de vídeo é necessário determinar a qualidade da imagem e da compressão do vídeo, principalmente quando a codificação é com perdas.

2.2.4.1 PSNR

A métrica mais utilizada para a avaliação da qualidade de imagem objetiva de um vídeo é o PSNR. O PSNR, medido em decibéis (dB), leva em conta o valor máximo de uma amostra utilizando a fórmula $(2^n - 1)^2$, onde n é o números de *bits* da amostra, e o MSE entre a imagem referência e a imagem analisada, resultando na Equação (2.3).

$$PSNR = 10 \log_{10} \frac{(2^n - 1)^2}{MSE} \tag{2.3}$$

De forma geral, um PSNR mais alto indica uma imagem de maior qualidade, enquanto um valor menor indica uma imagem com pior qualidade (Richardson, 2010). O PSNR pode ser facilmente e rapidamente calculado, porém possui a limitação de que é necessária a imagem original para avaliar a qualidade uma codificação.

2.2.4.2 BD-Rate

Além da avaliação da qualidade de imagem, a qualidade de compressão também é muito importante quando falamos em comparações de codificadores de vídeo. Diferente da qualidade de imagem, a qualidade de compressão é inerentemente mais objetiva, geralmente sendo representada em número de *bits* ou em taxa de *bits* (*bit rate*).

Dessa forma, utilizando tanto o PSNR quanto o *bitrate*, e explorando as opções de QP na etapa de quantização, é possível criar curvas de taxa de distorção (do inglês *rate distortion curves*) para analisarmos, em um mesmo codificador com configurações diferentes ou entre codificadores, a relação entre as métricas de qualidade de imagem e compressão. Para facilitar essa comparação, temos o *Bjøntegaard Delta Bit Rate* (BD-Rate), que traduz para um único número a taxa de aumento do *bit rate* considerando quatro níveis de quantização diferentes (Bjontegaard, 2001).

2.2.5 Métricas de Similaridade Entre Blocos

Nas etapas de Predição Intra-quadro e Predição Inter-quadros, existe a necessidade de comparar e estimar a distorção entre dois blocos. Para realizar essa comparação, algumas métricas são utilizadas, sendo duas delas o SAD e o SATD, das quais falaremos sobre a seguir.

2.2.5.1 SAD

O SAD é uma métrica facilmente e rapidamente calculável, sendo a soma dos valores absolutos das diferenças entre os dois blocos, ou seja, uma sucessão de adições e subtrações. A Equação (2.4) apresenta como o SAD é calculado, onde O é o bloco original, R é o bloco de referência e h e w são as dimensões dos blocos.

$$SAD = \sum_{i=1}^{h} \sum_{j=1}^{w} |O_{i,j} - R_{i,j}| \tag{2.4}$$

Pela sua velocidade, ele é muito indicado para etapas executadas muitas vezes pelo codificador e é comumente implementado diretamente em *hardware*. Apesar de sua velocidade, normalmente é uma métrica com um desempenho baixo para a estimação de distorções.

2.2.5.2 SATD

O SATD, diferente do SAD, é uma métrica com uma complexidade maior e que necessita de mais esforço computacional para ser calculada. O SATD é obtido pela soma absoluta dos coeficientes da matriz resultante da aplicação da Transformada de Hadamard na matriz das diferenças dos blocos. Primeiro, obtemos a diferença entre os blocos na Equação (2.5), onde O é o bloco original, R é o bloco de referência, h e w são as dimensões dos blocos e D é o bloco de resíduos resultante da diferença, que pode ser representado por uma matriz.

$$D_{h,w} = O_{h,w} - R_{h,w} \tag{2.5}$$

Em seguida, devemos obter a matriz de coeficientes transformados pela transformada de Hadamard, obtido através da Equação (2.6), onde D é a matriz das diferenças, H é a matriz da transformada de Hadamard, $H\top$ é a matriz da transformada de Hadamard transposta e HT é a matriz de coeficientes transformados. A matriz de Hadamard relevante para o contexto desse trabalho possui as dimensões 8×8 e é apresentada na Equação (2.7).

$$HT = H \cdot D \cdot H^T \tag{2.6}$$

$$H = \frac{1}{4}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \tag{2.7}$$

Para obtermos o SATD temos a Equação (2.8), onde HT é a matriz de coeficientes transformados e h e w são as dimensões da matriz.

$$SATD = \sum_{i=1}^{h}\sum_{j=1}^{w} |HT_{i,j}| \tag{2.8}$$

O SATD, apesar de mais complexo, possui uma boa relação entre custo computa-

cional e estimação de distorção, sendo também amplamente utilizado.

2.2.5.3 SATD Aproximado

Sendo o SATD uma métrica com melhor desempenho de estimação de distorção que o SAD, é possível pensar em formas de utilizar computação aproximada para tentar simplificar o processo complexo de multiplicação de matrizes com a menor perda de qualidade de estimação de distorção possível.

Através de um processo de eliminação de operações na transformada de Hadamard, podemos ter um ganho em relação ao custo computacional da transformada enquanto mantemos um desempenho melhor que do SAD. Esse processo se utiliza do fato que os coeficientes da transformada possuem importância diferente para o desempenho do resultado final (Soares et al., 2016).

Dessa forma, é possível gerar um grupo de aproximações para o SATD e avaliar seus desempenhos, como pode ser observado em Stigger et al. (2020). Porém, as diferentes aproximações possuem desempenhos diversos em vídeos distintos, não sendo possível eleger a melhor aproximação para ser utilizada em todos os casos.

2.3 Aprendizado de Máquina

Na terceira parte do capítulo falaremos sobre os conceitos relativos à área de Aprendizado de Máquina utilizados neste trabalho, como conceitos básicos de Aprendizado de Máquina, tipos de aprendizado, Aprendizado Supervisionado, alguns algoritmos de predição, tipos de atributo, normalização, conjuntos de treinamento e conjuntos de teste, e avaliação e métricas de desempenho.

2.3.1 Conceitos Básicos de Aprendizado de Máquina

Desde que os computadores foram inventados, existe a dúvida sobre se eles seriam capazes de serem construídos para aprender. Caso fosse possível programá-los para aprender, ou seja, se aperfeiçoarem com o tempo de uso, o impacto seria dramático (Mitchell, 1997). Segundo Faceli et al. (2011), se da o nome Aprendizado de Máquina ao processo de indução de uma hipótese ou aproximação de função a partir da experiência passada pela máquina.

Ainda segundo Faceli et al. (2011), em ML, os computadores são programados para aprender com a experiência passada. Para isso, utilizam um princípio de inferência denominado indução, onde a partir de um conjunto particular de exemplos se obtêm conclusões genéricas. Dessa forma, algoritmos de ML aprendem a induzir uma função ou hipótese capaz de resolver um problema a partir de dados que representam instâncias do mesmo problema a ser resolvido. Esses dados formam um conjunto, denominado conjunto de dados.

Cada instância nesse conjunto de dados pode representar um objeto real ou um conceito abstrato. O conjunto de propriedades e características desse objeto/conceito é traduzido para atributos no conjunto de dados. Pensando em uma matriz, cada linha seria uma instância representando um objeto/conceito, e as colunas são os atributos que representam as características e propriedades desse objeto/conceito (Faceli et al., 2011).

2.3.2 Tipos de Aprendizado de Máquina

Existem, pelo menos, três tipos diferentes de aprendizados dentro do universo de ML: o Aprendizado Supervisionado, o Aprendizado Não Supervisionado e o Aprendizado por Reforço (Abu-Mostafa; Magdon-Ismail; Lin, 2012). Falaremos do Aprendizado Supervisionado posteriormente nesse capítulo e, aqui, uma breve explicação dos outros tipos.

No Aprendizado Não Supervisionado, os dados são fornecidos para o algoritmo sem qualquer informação de classificação para cada instância (Abu-Mostafa; Magdon-Ismail; Lin, 2012). Dessa forma, o algoritmo busca padrões e semelhanças entre os atributos das instâncias, formando conjuntos (*clusters*) de onde é possível extrair algumas informações relevantes. O Aprendizado Não Supervisionado pode ser visto como uma forma de criar uma representação dos dados num nível maior de abstração (Abu-Mostafa; Magdon-Ismail; Lin, 2012).

O Aprendizado por Reforço funciona como uma criança aprendendo o que deve ou não fazer através da experiência de tomar ações (Abu-Mostafa; Magdon-Ismail; Lin, 2012). Através de um sistema de recompensa por ações, o algoritmo de Aprendizado por Reforço aprende as melhores ações a tomar em determinados estados. Na Figura 2.3, é possível observar o ciclo do agente realizando uma ação em um determinada situação no ambiente e a partir dessa ação, recebendo uma recompensa r e um novo estado s onde ele deve tomar outra ação, e assim sucessivamente.

Figura 2.3 – Modelo de Aprendizado por Reforço

Fonte: Adaptado por Santos (2019)

2.3.3 Aprendizado Supervisionado

Os modelos de Aprendizado Supervisionado podem ser divididos entre Modelos de Classificação e Modelos de Regressão (Faceli et al., 2011). Diferente dos outros tipos de aprendizado, o Aprendizado Supervisionado funciona com a premissa de uma instância no conjunto de dados, baseada em seus atributos, possuir um valor de saída ou classificação (Abu-Mostafa; Magdon-Ismail; Lin, 2012). Por exemplo, na Figura 2.4 temos os atributos relacionados a cada paciente (ou instância), presentes entre as colunas Idade e Estado, e temos a classificação do estado de saúde daquele paciente na coluna Diagnóstico.

2.3.3.1 Modelos de Classificação

Modelos de Classificação buscam, através da análise dos atributos de uma instância, classificá-la com um rótulo de valor nominal (Faceli et al., 2011). Usando o mesmo exemplo da Figura 2.4, através do conjunto de atributos relevantes Idade, Sexo, Peso, Manchas, Temperatura, Número de Internações e Estado, o modelo deve analisá-lo e realizar a classificação entre os rótulos possíveis, nesse caso, Saudável ou Doente. Essa classificação é feita encontrando, uma ou mais, fronteiras de decisões que sejam capazes de separar as instâncias entre as classes possíveis para classificação (Faceli et al., 2011).

Figura 2.4 – Conjunto de dados de pacientes de um hospital

Fonte: Faceli et al. (2011)

Diferentes algoritmos de ML trabalham de forma diferente na busca das fronteiras de decisões e podem encontrar fronteiras diferentes entre si, além de que diferenças nos conjuntos de treinamento de um modelo, variações na ordem de apresentação dos exemplos durante esse treinamento e os próprios processos internos estocásticos podem fazer um mesmo algoritmo encontrar fronteiras diferentes a cada execução (Faceli et al., 2011). Os algoritmos de classificação abordados nesse trabalhos serão o *Decision Tree*, o *K-Nearest Neighbors* (*K-NN*), o *Naive Bayes* e o *Random Forest*.

É importante notar que o balanceamento dos dados é importante pois vários algoritmos de ML podem ter seu desempenho prejudicado caso haja um desbalanceamento na quantidade de dados pertencentes a uma ou mais classes em relação as outras classes. Por exemplo, é possível que um conjunto de dados divididos em duas classes A e B, onde a distribuição do Conjunto de Dados é de 90% das instâncias pertencendo a classe A e apenas 10% das instâncias pertencendo a classe B, seja afetado e possua um viés para a classe A.

Para lidar com o desbalanceamento podemos utilizar algumas técnicas como redefinir o tamanho do conjunto de dados para que a distribuição entre as classes seja mais próxima, ou utilizar diferentes custos de classificação para diferentes classes, adicionando um custo para um preditor avaliar uma instância como a classe que possui uma ocorrência muito maior.

2.3.3.2 Modelos de Regressão

Modelos de Regressão, por sua vez, buscam, através da análise dos atributos de uma instância, gerar uma função e retornar um valor em um conjunto infinito e ordenado de valores (Faceli et al., 2011). Olhando para o exemplo na Figura 2.5, o algoritmo de

regressão geraria uma função que, com os atributos Fertilidade, Agricultura, Educação e Renda como os parâmetros de entrada, retornaria um valor para o atributo alvo Mortalidade.

A geração dessa função também varia de acordo com o algoritmo de regressão utilizado. É importante notar que a função gerada pelo modelo é uma aproximação de uma função desconhecida que reflete a realidade e retorna sempre o valor correto. O algoritmo de regressão abordado nesse trabalho será o *Logistic Regression*.

Figura 2.5 – Conjunto de dados de populações

Fonte: Faceli et al. (2011)

2.3.4 Algoritmos de Predição

Existem diversos algoritmos de predição de Aprendizado Supervisionado, como *Decision Tree*, *K-Nearest Neighbor*, *Naive Bayes*, *Random Forest* e *Logistic Regression*.

2.3.4.1 Decision Tree

O algoritmo *Decision Tree* utiliza a estratégia dividir para conquistar para resolver um problema de decisão. Um problema complexo é divido em problemas mais simples, aos quais recursivamente é aplicada a mesma estratégia (Faceli et al., 2011). Utilizando o conceito de árvores computacionais, cada instância analisada irá percorrer o caminho do primeiro Nó de Divisão até um Nó Folha, podendo passar por diversos nós intermediários. No exemplo da Figura 2.6, para descobrir se uma linha de crédito deve ser concedida a um indivíduo, a árvore avalia os atributos Idade, Estudante e Renda Mensal. Dependendo da combinação de valores dos atributos, a árvore avaliará se o crédito deve ou não ser concedido.

Figura 2.6 – Exemplo de Decision Tree

Fonte: Medium[4]

Todo Nó de Divisão, incluindo o Nó Raiz, terá a instância sendo analisada como entrada e terá duas ou mais saídas, levando a outros nós. Relacionada a cada Nó de Divisão teremos um teste condicional, desenvolvido pelos processos internos da implementação do algoritmo, que pode ser sobre um ou mais atributos, que irá avaliar para qual saída a instância deve ser enviada. No caso do exemplo citado anteriormente, no primeiro Nó de Divisão 'Idade', é avaliado se a idade do solicitante é menor que 25 anos. Caso seja, a instância irá para o Nó de Divisão 'Estudante?', caso contrário, irá para o Nó de Divisão 'Renda Mensal', e assim por diante.

Todo Nó Folha terá uma classificação relacionada, podendo ser igual a de outros Nós Folha, sendo a classificação atribuída a cada instância que eventualmente alcançá-lo ao percorrer a árvore. Ainda no exemplo citado anteriormente, ao alcançar um Nó Folha, a instância será classificada como 'Concede' ou 'Não Concede', dependendo da classificação relacionada ao nó.

[4]Disponível em: https://medium.com/@msremigio/Árvores-de-decisão-decision-trees-4cb6857671b3

2.3.4.2 K-Nearest Neighbors

O algoritmo K-NN utiliza um plano de N dimensões, sendo N o número de atributos do conjunto de dados, para, ao analisar os atributos de uma instância, atribuir uma posição espacial para essa instância. Em seguida, ele avalia as outras K instâncias mais próximos dessa instância sendo avaliada, e a classifica baseado na classificação dos seus vizinhos mais próximos.

Observando a Figura 2.7, é possível ver como o K influencia na decisão de classificação. Se considerarmos o K como sendo 3 (3-NN), a instância pertenceria a classe Doente. Já se considerarmos o K como sendo 5 (5-NN), a instância pertenceria a classe Saudável. Por isso, a escolha do parâmetro K é de extrema importância e pode influenciar diretamente o desempenho do algoritmo.

Figura 2.7 – Exemplo de K-NN

Fonte: Faceli et al. (2011)

2.3.4.3 Naive Bayes

O algoritmo *Naive Bayes* utiliza a probabilidade estatística para realizar a classificação, utilizando os métodos probabilísticos bayesianos. O teorema de Bayes (Equação 2.9) mostra como calcular a probabilidade $P(A|B)$, ou seja, a probabilidade de um evento A ocorrer, dado um evento B. Para isso é necessário conhecer a probabilidade $P(A)$ de um evento A ocorrer, a probabilidade $P(B|A)$ de um evento B ocorrer dado um evento

A e a probabilidade $P(B)$ de um evento B ocorrer.

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \tag{2.9}$$

Por exemplo, para calcularmos a probabilidade de chover dado que há nuvens no céu, ou seja, $P(Chover|Nuvens\ no\ céu)$, seria necessário calcular primeiramente a probabilidade $P(Chover)$, a probabilidade $P(Nuvens\ no\ céu)$ e a probabilidade $P(Nuvens\ no\ céu|Chovendo)$. Atribuindo os valores $0,54$, $0,62$ e 1 para as últimas três probabilidades citadas, respectivamente, o resultado pode ser observado na Equação 2.10.

$$\begin{aligned} P(Chover|Nuvens\ no\ céu) &= \frac{1 \cdot 0,54}{0,62} \\ P(Chover|Nuvens\ no\ céu) &= 0,87 \end{aligned} \tag{2.10}$$

2.3.4.4 Random Forest

O algoritmo *Random Forest* utiliza o algoritmo *Decision Tree* para gerar um número N de árvores para no final realizar um votação, onde cada árvore votará na classificação do Nó Folha que a instância sendo analisada alcançou. No final, a classificação com mais votos será escolhida para classificar a instância sendo analisada (Figura 2.8).

2.3.4.5 Logistic Regression

O algoritmo *Logistic Regression*, apesar de ser um algoritmo de Regressão, aqui, será utilizado como um algoritmo de Classificação. Essa prática é bem comum dado que esse algoritmo trabalha com probabilidades (Abu-Mostafa; Magdon-Ismail; Lin, 2012).

O algoritmo define limites de valores para cada categoria. Dependendo entre quais limites o resultado da função de regressão ficar, ele será considerado da classe atribuída aquela faixa, funcionando como um algoritmo de Classificação.

2.3.5 Tipos de Atributo

O tipo de um atributo define se ele irá representar quantidades ou qualidades de um objeto. Quando um atributo representa a quantidade de algo, ele é chamado de quan-

[5]Disponível em: https://www.ibm.com/br-pt/topics/random-forest

Figura 2.8 – Exemplo de Random Forest

Fonte: IBM[5]

titativo, e quando ele representa uma qualidade categórica, ele é chamado de qualitativo (Faceli et al., 2011). A principal diferença entre os tipos de atributos é a possibilidade de realizar operações aritméticas com cada um, não sendo possível realizá-las com os atributos qualitativos, apenas com os quantitativos. Atributos quantitativos ainda podem ser classificados como contínuos ou discretos.

Exemplos de atributos quantitativo contínuos seriam a altura ou peso de uma pessoa, ou a temperatura que essa pessoa se encontra. Atributos contínuos podem assumir um número infinito de valores e, normalmente, são resultados de medidas (Faceli et al., 2011).

Já como exemplos de atributos quantitativos discretos temos a idade de uma pessoa, ou a quantidade de filhos que uma pessoa tem. Atributos discretos contêm um número finito ou infinito contável de valores (Faceli et al., 2011).

No caso de atributos qualitativos, temos rótulos categóricos que não possuem valor aritmético, mas que podem ser utilizados para identificação ou agrupamentos. Como exemplos, temos o nome de uma pessoa, a cor do cabelo ou o estado civil dela.

2.3.6 Normalização

Atributos quantitativos podem ter diferentes grandezas absolutas nos seus valores, indo de valores muito pequenos, como o tempo que demora para um computador executar uma função simples a números muito grandes, como o tamanho em *bits* de um arquivo de vídeo não codificado. Neste exemplo, a grandeza pode ir de milésimos no caso do tempo a centenas de milhões no caso do tamanho. Para evitar dar maior peso a um atributo no processo de treinamento do modelo, quando isso não é desejável, realizamos o processo de normalização. Existem alguns métodos de normalização, como por amplitude ou por distribuição.

Segundo Faceli et al. (2011), um dos métodos de normalização por amplitude é a normalização por reescala (também conhecido como *min-max*). Inicialmente, é necessário definir os valores mínimo (min) e máximo (max), usualmente 0 e 1 respectivamente, para os novos valores de cada atributo (Faceli et al., 2011). Depois de obtermos o menor valor ($menor$) e o maior valor ($maior$) que um determinado atributo assume no conjunto de dados, podemos aplicar a Equação (2.11) para obter o valor normalizado (v_{norm}) de um valor atual de determinada instância (v_{atual}).

$$v_{norm} = min + \frac{v_{atual} - menor}{maior - menor} * (max - min) \tag{2.11}$$

2.3.7 Conjunto de Treinamento e Conjunto de Teste

Da mesma forma que um professor fornece exercícios com as respostas aos alunos para que estudem e treinem antes de aplicar um teste com exercícios diferentes, mas que são resolvidos da mesma forma que os exercícios fornecidos anteriormente, um modelo também precisa "estudar" e treinar em cima de um conjunto de dados para que possa fazer a predição das novas instâncias (Abu-Mostafa; Magdon-Ismail; Lin, 2012). Porém, após treinar um modelo, como podemos saber o seu desempenho e seu nível de confiabilidade antes de utilizá-lo para fazer predições de instâncias que não sabemos a classificação?

Por isso, a prática mais comum é dividir o Conjunto de Dados, onde temos conhecimento da classificação de cada instância, em dois conjuntos: o Conjunto de Treinamento e o Conjunto de Teste. Dessa forma podemos utilizar o Conjunto de Treinamento para treinar o modelo e o Conjunto de Testes para fazer uma avaliação do seu desempenho, já

que sabemos quais as classificações esperadas para cada instância.

É importante manter em mente o balanceamento de classes em casos de Classificação. Como cada um dos conjuntos de treino e teste seriam amostras do conjunto original, podemos utilizar técnicas de amostragem. Uma delas é a amostragem estratificada, onde, ao realizar a divisão do Conjunto de Dados entre Conjunto de Treinamento e Conjunto de Teste, os conjuntos devem manter a mesma proporção de instâncias de cada classe. Dessa forma, se o Conjunto de Dados possui 40% das instâncias pertencendo a classe *A*, 35% a classe *B* e 25% a classe *C*, ambos conjuntos de Treinamento e de Teste devem manter a mesma proporção de distribuição, independente da quantidade de instâncias selecionadas para cada conjunto.

2.3.8 Avaliação e Métricas de Desempenho

Apesar de ser possível apenas dividirmos o Conjunto de Dados em Conjunto de Treinamento e Conjunto de de Teste, existem algumas técnicas ligeiramente mais avançadas e que costumam apresentar resultados melhores. Uma dessas técnicas é a Validação Cruzada *K-Fold — K-Fold Cross-Validation* (KFCV), que além da divisão dos conjuntos, também lida com a avaliação dos modelos.

2.3.8.1 Validação Cruzada K-Fold

O método KFCV consiste em dividir um Conjunto de Dados em K grupos de tamanhos iguais (*folds*). Depois que os grupos são definidos, o algoritmo de treinamento é executado K vezes, onde um dos grupos é escolhido para ser o Conjunto de Teste enquanto todos os outros grupos são o Conjunto de Treinamento. Após K execuções, onde todos os grupos foram o Conjunto de Teste uma vez, é realizada uma média dos desempenhos das K execuções (Faceli et al., 2011).

Por exemplo, se realizarmos um KFCV onde o K é 10, executaremos o algoritmo de treinamento 10 vezes, cada uma vez elegendo um grupo diferente para ser o Conjunto de Teste e os outros 9 grupos como o Conjunto de Treinamento.

Quando utilizamos o KFCV, o tamanho dos conjuntos de Treinamento e de Teste de cada execução também são determinados por K. Por exemplo, quando o K for 10, em cada execução do algoritmo o Conjunto de Testes será 10% do tamanho do Conjunto de Dados, e consequentemente, o Conjunto de Treinamento de cada execução será 90%. Já

se o K for 5, as proporções passam a ser 20% e 80% respectivamente. É importante notar que a divisão do KFCV pode ser ou não estratificada (Faceli et al., 2011).

Técnicas de Validação Cruzada são principalmente utilizadas para avaliar a habilidade de generalização de um modelo preditivo e para prevenir o *overfitting*. O *overfitting* ocorre quando um modelo preditivo fica especializado demais no Conjunto de Treinamento utilizado, tendo dificuldade de fazer a predição de novas instâncias desconhecidas.

2.3.8.2 Métricas de Desempenho

Segundo Faceli et al. (2011), de maneira geral, pode-se afirmar que não é possível estabelecer previamente que uma técnica de ML em particular se sairá melhor na resolução de qualquer tipo de problema. Dessa forma, existe a necessidade de experimentação, avaliação e comparação no domínio de ML. As métricas utilizadas para avaliação e comparação são variadas e podem ser realizadas sobre diversos aspectos, como a acurácia do modelo gerado, tempo de aprendizado, requisitos de armazenamento do modelo, entre outros (Faceli et al., 2011).

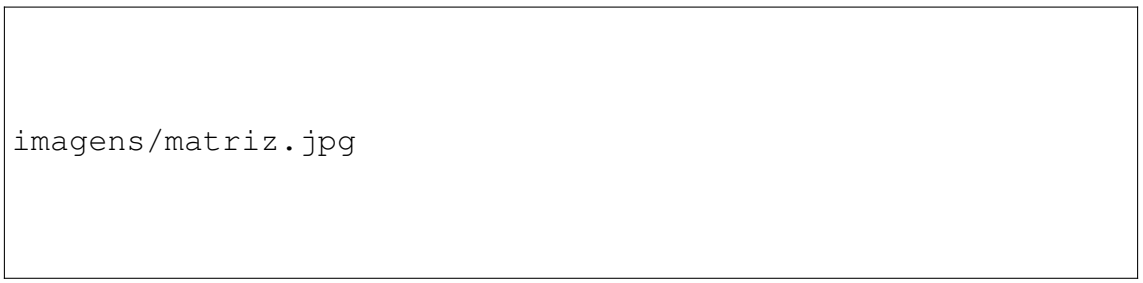
Através da matriz de confusão de um modelo, é possível extrair diversas métricas de desempenho para comparação de modelos. Utilizando como exemplo um caso de duas classes, a classe *Positiva* (+) e a classe *Negativa* (-), temos a matriz de confusão da Figura 2.9, onde:

- *VP* corresponde ao número de verdadeiros positivos, instâncias da classe *Positiva* corretamente classificadas.
- *VN* corresponde ao número de verdadeiros negativos, instâncias da classe *Negativa* corretamente classificadas.
- *FP* corresponde ao número de falsos positivos, instâncias da classe *Negativa* erroneamente classificadas como classe *Positiva*.
- *FN* corresponde ao número de falsos negativos, instâncias da classe *Positiva* erroneamente classificadas como classe *Negativa*.

Várias métricas de desempenhos podem ser extraídas de uma matriz de confusão, e as utilizadas nesse trabalho foram:

- Acurácia: A Taxa de Acerto ou Acurácia mostra quantas instâncias foram corretamente classificadas pelo modelo. Para calcular a Acurácia, somamos todos as classificações verdadeiras e dividimos pelo número total de classificações n (Equa-

Figura 2.9 – Exemplo de matriz de confusão



Fonte: Faceli et al. (2011)

ção 2.12).

$$Acurácia = \frac{VP + VN}{n} \tag{2.12}$$

- Precisão: A Precisão representa a proporção de instâncias classificadas como positivas enquanto de fato pertenciam a classe *Positiva* em relação a todas instâncias classificadas como positivas (Equação 2.13).

$$Precisão = \frac{VP}{VP + FP} \tag{2.13}$$

- Revocação: A Revocação representa a taxa de acerto da classe *Positivo*, ou seja, a taxa de instâncias positivas que foram classificadas como positivas em relação a todas as instâncias pertencentes a classe *Positiva* (Equação 2.14).

$$Revocação = \frac{VP}{VP + FN} \tag{2.14}$$

- Medida-F: A Medida-F busca, através da média harmônica ponderada da Precisão e da Revocação, resolver os problemas de que a Precisão não consegue refletir as instâncias da classe *Positiva* que não foram classificadas corretamente (*FN*) e que a Revocação não consegue refletir as instâncias da classe *Negativa* que foram classificadas como positivas (*FP*). Como é possível observar na Equação (2.15), através da variável w podemos dar mais importância para a Precisão ou para a Revocação. Quando damos a mesma importância para ambas métricas, w assume o valor de 1, e a métrica é comumente chamada de Medida-F1. Valores acima de 1 (normalmente 2) atribuem uma importância maior a Revocação enquanto valores abaixo de 1 (normalmente 0,5) atribuem uma importância maior a Precisão.

$$Medida\text{-}F = \frac{(w+1) * Revocação * Precisão}{Precisão + w * Revocação} \tag{2.15}$$

2.4 Trabalhos Relacionados

Conforme citado anteriormente neste capítulo, existe um alto custo computacional para calcular o SATD durante o processo de codificação, mais especificamente o cálculo envolvendo matrizes presente na Transformada de Hadamard. Portanto, é comum essa etapa do processo ser objeto de pesquisas que visam otimizá-lo, principalmente a nível de *hardware*. Porém, não foi encontrado, até a finalização deste trabalho, referências publicadas que tenham utilizado alguma variação de SATD Aproximado e ML a nível de código do codificador para tentar realizar essa otimização. Contudo, existem trabalhos que envolvem o uso de computação aproximada no cálculo do SATD a nível de *hardware*.

Em Stigger et al. (2019), os autores propõem um *hardware* acelerador para o cálculo do SATD, utilizando o cálculo aproximado da Transformada de Hadamard em blocos 4×4. A proposta é de utilizar um acelerador que realize apenas as transformadas horizontais do cálculo da Transformada de Hadamard, arquitetura intitulada SATD 4×4 2D, ao invés de executar ambas as transformadas horizontais e verticais, arquitetura intitulada SATD 4×4 1D. Utilizando a arquitetura proposta, os autores conseguiram reduzir em 66,86% a energia dissipada em comparação a um acelerador de *hardware* para SATD calculado de forma precisa. Além disso, o BD-Rate teve um aumento de menos de 0,748%, em comparação ao aumento de até 0,830% ao utilizar um acelerador de *hardware* para o cálculo do SAD, ambos em relação ao BD-Rate obtido utilizando o SATD calculado de forma precisa.

Em Lima et al. (2021), os autores propõem um *hardware* acelerador configurável para o cálculo do SATD, com quatro possibilidades de configuração, sendo elas o SATD calculado de forma precisa, SATD Aproximado excluindo 3 colunas de somadores/subtratores, SATD Aproximado excluindo 5 colunas de somadores/subtratores e utilizando o SAD. Através de um *MUX4-1*, os autores escolhem em qual ponto do cálculo os valores serão utilizados nos somador de coeficientes. Com isso, a dissipação de energia nas configurações que excluem 3 colunas, que excluem 5 colunas e SAD apresenta uma queda de 19,87%, 32,33% e 39,16% respectivamente quando comparado a dissipação de energia do SATD calculado de forma precisa, enquanto o aumento do BD-Rate

apresentado foi, em média, de 0,0954%, 0,1620% e 0,2534% respectivamente.

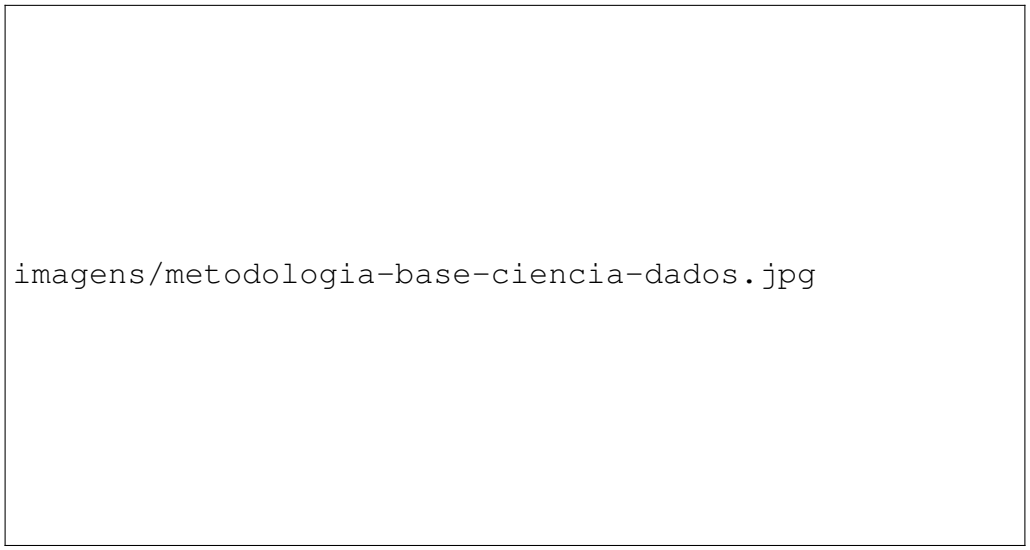
3 METODOLOGIA E FERRAMENTAS UTILIZADAS

Neste capítulo, são apresentadas a metodologia e as ferramentas utilizadas no processo de desenvolvimento deste trabalho.

3.1 Metodologia

A metodologia adotada neste trabalho foi a Metodologia de Base para Ciência de Dados (Rollins, 2015), que divide o processo em **estágios**. A metodologia consiste em 10 estágios que formam um processo cíclico, com diversos pontos de reiteração (Figura 3.1).

Figura 3.1 – Metodologia de Base para Ciência de Dados



Fonte: Rollins (2015)

- **Entendimento dos negócios**: neste estágio, ocorre a definição do problema, os objetivos do projeto e os requisitos da solução, de um ponto de vista do negócio. No contexto do trabalho, o problema definido foi a necessidade e possibilidade de otimização do processo de cálculo do SATD utilizando o SATD Aproximado e ML, enquanto o objetivo foi definido como a otimização do processo de codificação de vídeo, considerando qualidade do vídeo, a taxa de *bits* do vídeo codificado, além do tempo de codificação.
- **Abordagem Analítica**: neste estágio, é onde ocorre a definição de quais serão as técnicas empregadas para a resolução do problema. No contexto do trabalho, a definição de um problema de classificação ofereceu a possibilidade da utilização do

Aprendizado Supervisionado, tanto pela natureza do problema quanto pela familiaridade com este tipo de técnicas.

- **Requerimentos de Dados**: neste estágio é definido, determinado pela Abordagem Analítica, quais dados serão utilizados, como eles devem estar formatados e representados. No contexto do trabalho, foram definidos quais atributos seriam utilizados para o treinamento do modelo de predição. Dada a natureza exploratória do trabalho, este se mostrou o estágio mais desafiador, o qual foi necessário retornar posteriormente, depois de etapas de avaliação do modelo, para uma redefinição dos requerimentos e quais atributos seriam utilizados.
- **Coleta de Dados**: neste estágio, é feita a coleta de dados. Os recursos de dados disponíveis, estruturados ou não, relevantes para o domínio do problema são reunidos. No contexto do trabalho, dada a inexistência de dados disponíveis para o desenvolvimento do trabalho, foi necessário realizar a coleta de dados direto do processo de codificação de vídeo. Essa coleta é demorada devido ao processo de codificação de vídeo ser demorado. Foram necessárias duas grandes etapas de coleta de dados, dado que o trabalho passou por uma grande mudança de Requerimento de Dados em certo ponto.
- **Entendimento dos Dados**: neste estágio, é feito o uso de estatísticas descritivas e técnicas de visualização para entender o conteúdo dos dados, avaliar a qualidade dos dados e ter algumas compreensões iniciais sobre os dados. No contexto do trabalho, ambos os conjuntos de dados coletados eram constituídos de atributos numéricos que continham informações sobre os blocos 8×8 onde cada aproximação do cálculo de SATD foi aplicada no processo de codificação de vídeo.
- **Preparação de Dados**: neste estágio, é feito toda a preparação do conjunto de dados que será utilizada na modelagem subsequente através de atividades como limpeza de dados, combinação de dados de diversas fontes e transformação de dados em variáveis mais úteis. No contexto do trabalho, como todos os dados utilizados no trabalho são próprios e coletados direto do processo, foi possível realizar essa coleta de forma que este estágio não fosse tão complexo. Foi necessário um processamento dos dados coletados a fim de obter uma avaliação inicial e descobrir quais aproximações tiveram os melhores desempenhos por média de blocos 8×8 por *frame* para o primeiro conjunto de dados e comparação direta entre blocos 8×8 no segundo conjunto de dados obtidos. Além disso, todos os dados foram normalizados.

- **Modelagem**: neste estágio, é feita a modelagem com base na Abordagem Analítica definida anteriormente. No caso de ML, é normal modelar diversos algoritmos com seus parâmetros a fim de obter o melhor modelo que atende as necessidades do projeto. No contexto do trabalho, utilizamos os seguinte algoritmos de classificação de Aprendizado Supervisionado: *Decision Tree*, *K-NN*, *Logistic Regression*, *Naive Bayes* e *Random Forest*. Além de da eficácia na predição, outros fatores importantes na decisão de qual modelo utilizar foram a velocidade de predição e a possibilidade e complexidade da implementação do modelo em *hardware* dedicado.
- **Avaliação**: neste estágio, é feita a avaliação do modelo feito no estágio anterior, antes de implementá-lo, a fim de assegurar que ele trate de maneira completa e adequada o problema. Essa avaliação é realizada com medidas de diagnóstico computadas a partir dos resultados dos modelos. No contexto do trabalho, para a obtenção dos dados necessários para realizar a avaliação, os modelos foram treinados e testado utilizando o método KFCV. As métricas obtidas e utilizadas para comparar os modelos foram a acurácia e o tempo de predição.
- **Implementação**: neste estágio, é realizada a implementação do modelo após a avaliação e aprovação de algum dos modelos. Essa implementação normalmente ocorre em um ambiente de produção ou em um ambiente de testes que seja comparável. No contexto do trabalho, nenhum dos modelos obtiveram avaliações boas o suficientes que justificassem a sua implementação no código do VVC.
- ***Feedback***: neste estágio, é onde dados da performance do modelo implementado são colhidos, além da avaliação do impacto no ambiente de implementação. A análise do *feedback* permite o refinamento do modelo para melhorar sua performance e utilidade. No contexto do trabalho, como nenhum modelo obteve uma avaliação satisfatória suficiente para uma implementação, esse estágio não foi realizado.

Os procedimentos relevantes de cada estágio serão discutidos mais profundamente nos Capítulos 4 e 5.

3.2 Ferramentas Utilizadas

Para o desenvolvimento do trabalho foram usadas as seguintes ferramentas:

- **VVC Test Model**: O *VVC Test Model* (VTM) é o *software* referência do padrão de codificação VVC. Desenvolvido pelo JVET, o *software* é composto de do codifica-

dor e decodificador e tem como objetivo demonstrar as capacidades e auxiliar no estudo, desenvolvimento, implementação e testagem do VVC. A versão utilizada do VTM foi a 14.0.

- **C++**: Linguagem de programação utilizada para desenvolver o VTM. Foram necessárias alterações no código do VTM para realizar a coleta dos dados utilizados na construção dos Conjuntos de Dados utilizados no treinamento dos modelos.
- **Python**: Linguagem de programação utilizada para realizar o pré-processamento dos dados, a configuração, treinamento e avaliação dos modelos de predição desenvolvidos. As principais bibliotecas utilizadas foram a (1) *pandas*, que auxilia na manipulação de dados principalmente em formas de tabelas, e a (2) *scikit-learn*, biblioteca com foco em ML que apresenta diversos módulos, funções e algoritmos para a análise e pré-processamento de dados, treinamento e avaliação de modelos, com uma grande variedade de algoritmos de predição.
- **Visual Sudio Code**: O *Visual Studio Code* (VS Code) é um editor de código-fonte desenvolvido pela Microsoft. Ele possui suporte para diversas linguagens, entre elas C++ e Python. Foi a ferramenta utilizada para auxiliar na escrita de todos os códigos de modificação no VTM e os códigos utilizados para o desenvolvimento dos modelos de predição.
- **Overleaf**: O *Overleaf* é uma plataforma *online* de edição colaborativa de *LaTeX*. Foi a ferramenta utilizada para a escrita e edição desta monografia, facilitando o compartilhamento com os orientadores, bem como as revisões necessárias.

4 DESENVOLVIMENTO

Neste capítulo serão apresentados os passos tomados para o desenvolvimento do trabalho. Partindo da análise feita em cima de dados previamente coletados, passando pela definição dos atributos que foram utilizados nos modelos, pelo processo de coleta de dados realizado para as duas definições de atributos, e por fim, pelo treinamento dos modelos.

4.1 Pré-análise

Após a definição proposta e de uma pesquisa inicial por bancos de dados relevantes para o desenvolvimento da pesquisa, foi observada a necessidade da coleta de dados e a criação de um banco de dados próprio. Dessa forma, utilizando dados coletados previamente, que continham informações do BD-Rate de cada *frame* de alguns vídeos codificados utilizando várias aproximações do SATD (Stigger et al., 2020), foi realizada uma análise inicial.

Figura 4.1 – Exemplo de dados de BD-Rate do SATD Aproximado

Fonte: Os Autores

Os dados coletados foram referentes a 5 vídeos do *Common Test Conditions* (CTC) do JVET: *BQMall*, *BasketballDrive*, *BasketballPass*, *RaceHoreses* e *RitualDance*, coletando um número parcial e variado de *frames* de cada vídeo. O BD-Rate de cada *frame*

codificado foi calculado para as combinações das 3 configurações diferentes: *All Intra*, *Random Access* e *Low Delay*; e 15 modificações do SATD para SATD Aproximado. Nessas modificações havia tanto aproximações do SATD para blocos 4×4, blocos 8×8 e para ambos ao mesmo tempo. Um exemplo dos dados pode ser observado na Figura 4.1[1].

Os *frames* codificados somaram 160, com 3 configurações possíveis e 15 aproximações, totalizando 6.782 *frames* coletados pois em algumas combinações de configurações e aproximações, alguns *frames* foram pulados no processo de codificação. Com esses dados, foi possível avaliar quais aproximações tiveram um melhor desempenho, nesse caso, a quantidade de vezes que uma aproximação obteve o melhor valor de BD-Rate em um *frame* em comparação as outras aproximações (Figura 4.2[2]).

Figura 4.2 – Avaliação das aproximações por BD-Rate

Fonte: Os Autores

[1] A coluna ***satd*** é a aproximação do SATD utilizada na codificação, a coluna ***video*** é o nome do vídeo codificado, a coluna ***cfg*** é a configuração utilizada, a coluna ***frame*** é o *frame* do vídeo em questão (o valor -1 representa o vídeo inteiro) e a coluna ***bd rate*** é o valor de BD-Rate do *frame* (ou do vídeo inteiro no caso do *frame* ser -1)

[2] A coluna ***satd*** é a Aproximação SATD em questão, a coluna ***# best*** é a quantidade de vezes que a aproximação obteve o melhor BD-Rate no *frame* e a coluna ***%*** é a porcentagem em relação ao total de *frames*.

4.1.1 Definição de Aproximações Utilizadas

Para otimizar tempo de análise dos dados e execução dos experimentos e melhor delimitar o escopo do trabalho, foi definida a utilização de aproximações relativas ao bloco 8×8 (não utilizando nenhum tipo de aproximação no cálculo de blocos 4×4) e apenas a configuração *All Intra*. Dessa forma, foi definida a lista de aproximações utilizadas na elaboração do trabalho: as 4 aproximações do bloco 8×8 que obtiveram o melhor desempenho (Figura 4.2) e a aproximação **SAD**, que substitui o cálculo do SATD pelo cálculo do SAD. A lista completa consta na Tabela 4.1.

Tabela 4.1 – Aproximações de SATD selecionadas

Aproximação[3]	**Número de melhores *frames* (desempenho**[4]**)**
RdCost8x8-1	21 (13,13%)
RdCost8x8-2	22 (13,75%)
RdCost8x8-3	15 (9,38%)
RdCost8x8-4	15 (9,38%)
RdCost8x8-SAD	9 (5,63%)

Fonte: Os Autores

Com essa definição, as 5 aproximações escolhidas acabam por se tornar os possíveis valores do atributo-alvo do modelo de predição.

4.2 Definição de Atributos

Com as aproximações escolhidas e os atributos-alvos definidos, o próximo passo foi determinar quais atributos seriam utilizados na construção do banco de dados de treino e, posteriormente, do modelo implementado.

Essa foi uma das etapas mais desafiadoras do projeto, dada a dificuldade de encontrar materiais com relevância para este estudo. O estágio de definir atributos foi revisitado depois da avaliação dos primeiros modelos desenvolvidos, o que resultou em duas definições de atributos diferentes.

[3]Para simplificação, os nomes das aproximações não utilizam o prefixo ***4x4-***, pois as aproximações afetam apenas o bloco 8×8 e o prefixo existia apenas para mostrar que nada foi feito no bloco 4×4.

[4]Em relação ao total de 160 *frames* avaliados

4.2.1 Primeira Definição de Atributos

Na tentativa de coletar dados relevantes para o cálculo do SATD do bloco 8×8 de forma rápida, para tentar manter o aumento de tempo do processo de codificação o menor possível, o primeiro conjunto de atributos foi definido como um conjunto de manipulações básicas em cima do bloco de resíduos que é utilizado na Transformada de Hadamard dentro do SATD. Além disso, além de coletar os dados de forma relativa (mantendo o sinal resultando do processo de cálculo do bloco de resíduos), também coletamos os dados de forma absoluta (transformando valores negativos no bloco de resíduos em positivos). Todos os atributos são formados por 1 número.

Tabela 4.2 – Conjunto de Atributos Preliminares

Atributo	**Descrição**
min	Menor valor do bloco de resíduo
max	Maior valor do bloco de resíduo
sum	Soma dos valores do bloco de resíduo
avg	Média dos valores do bloco de resíduo
abs_min	Menor valor absoluto do bloco de resíduo
abs_max	Maior valor absoluto do bloco de resíduo
abs_sum	Soma dos valores absolutos do bloco de resíduo
abs_avg	Média dos valores absolutos do bloco de resíduo

Fonte: Os Autores

Esse processo gerou o Conjunto de Atributos Preliminares, como pode ser visto na Tabela 4.2. Porém, como o processo de coleta deveria ser feito para as 5 aproximações escolhidas como atributos-alvo, dada a forma que o processo de codificação faz o agrupamento e divisão de blocos do *frame*, seria impossível garantir que a coleta de todas as aproximações seriam feitas sobre a mesma divisão do *frame* e, por consequência, que os blocos 8×8 seriam coletados na mesma posição no *frame* entre todas as aproximações.

Por isso, foi definido que os valores coletados nos blocos 8×8 seriam processados a nível de *frame*, obtendo assim uma média para cada quadro codificado. O cálculo da média é um processo que resulta em perda de informação, porém, foi a única forma que encontramos de garantir que iriamos conseguir comparar os dados coletados no mesmo contexto entre as aproximações.

Para essa comparação, coletamos também o PSNR e o *bit rate* de cada frame. Calculando a divisão entre estes dois parâmetros ($PSNR/$*bit rate*), obtivemos uma métrica que leva em consideração a qualidade objetiva do *frame* e o quão eficaz foi a sua compressão. Assim, obtivemos o Primeiro Conjunto de Atributos, como pode ser visto na Tabela

4.3.

Tabela 4.3 – Primeiro Conjunto de Atributos

Atributo	Descrição
min	Média dos menores valores dos blocos de resíduo do *frame*
max	Média dos maiores valores dos blocos de resíduo do *frame*
sum	Média das somas dos valores do blocos de resíduo do *frame*
avg	Média da médias dos valores dos blocos de resíduo do *frame*
abs_min	Média dos menores valores absolutos dos blocos de resíduo do *frame*
abs_max	Média dos maiores valores absolutos dos blocos de resíduo do *frame*
abs_sum	Média das somas dos valores absolutos dos blocos de resíduo do *frame*
abs_avg	Média das médias dos valores absolutos dos blocos de resíduo do *frame*

Fonte: Os Autores

4.2.2 Segunda Definição de Atributos

No segundo processo de definir os atributos, decidimos diminuir a importância da velocidade no processo de obtenção dos dados dentro do bloco e optamos por utilizar uma maior quantidade de atributos e atributos mais complexos.

Para o segundo conjunto de atributos, começamos com alguns atributos utilizados na Primeira Definição de Atributos como média do bloco, maior valor do bloco, menor valor do bloco e a soma dos valores do bloco. Além desses atributos, outros foram escolhidos e podem ser vistos na Tabela 4.4.

Tabela 4.4 – Segundo Conjunto de Atributos

Atributo	Descrição	Colunas[5]
avg	Média dos valores do bloco	1
sd	Desvio padrão da média	1
max	Maior valor do bloco	1
min	Menor valor do bloco	1
sum	Soma dos valores do bloco	1
ent_sh	Valor de entropia do bloco em *shannons*	1
ent_nat	Valor de entropia do bloco em unidades naturais	1
ent_hart	Valor de entropia do bloco em *hartleys*	1
fft	Transformada de Fourier em cada elemento do bloco	64
hu	Momentos Invariantes de Hu	7
skew	A obliquidade de cada linha do bloco	8
kurt	A curtose de cada linha do bloco	8
freq	O espectro de magnitude de cada elemento do bloco	64
Total		159

Fonte: Os Autores

[5]Diferente da Primeira Definição de Atributos, alguns atributos desta definição são formado por mais

O maior desafio dessa definição de atributos é o tamanho do bloco. Muitos atributos tem problemas em operar em imagens pequenas, como por exemplo, imagens 8×8, o tamanho do nosso bloco de estudo.

4.3 Coleta de Dados

O processo de coleta de dados envolvendo codificadores de vídeos é, na maior parte das vezes, um processo demorado devido ao tempo necessário no próprio processo de codificação. Nesta etapa, também foi definido o conjunto de vídeos que foi utilizado para a coleta de dados e treinamento dos modelos. Como temos duas definições de atributos, naturalmente, temos dois processos de coleta de dados e dois conjuntos de dados coletados. O QP utilizado em ambos conjuntos de dados foi 30, que é o valor definido como padrão pelo VTM.

4.3.1 Definição do Conjunto de Vídeos de Treino

Ao compor o conjunto de vídeos utilizados na etapa de coleta de dados e treinamento do modelo (Tabela 4.5[6,7]), buscamos vídeos com alguma variedade de resolução de *frame* e FPS. Os vídeos foram obtidos da *Xiph.Org Foundation*[8], que possui acervos de diferentes tipos de vídeos, entre eles, um acervo de vídeos não codificados[9].

A decisão de não utilizar vídeos do CTC na coleta de dados e no treinamento e avaliação dos modelos foi tomada para utilizar esse conjunto de vídeos padronizados na etapa de avaliação do modelo implementado no codificador.

4.3.2 Primeiro Conjunto de Dados

Para realizar a coleta do Primeiro Conjunto de Dados, o código do VTM foi alterado para que toda vez que o cálculo do SATD para algum bloco 8×8 fosse chamado, os dados daquele bloco seriam coletados.

---

informações que apenas 1 número. Essas informações foram espalhadas por colunas ao longo da tabela.

[6]Com exceção do vídeo **DucksTakeOff**, que possui subamostragem YCrCb de 4:4:4, todos os outros vídeos do conjunto possuem subamostragem 4:2:0

[7]Todos os vídeos possuem 8 de *bit depth*

[8]Disponível em: https://www.xiph.org/

[9]Disponível em: https://media.xiph.org/video/derf/

Tabela 4.5 – Conjunto de Vídeos de Treinamento

Nome	**Resolução**	**FPS**	**Quantidade de *Frames***
BlueSky	1920x1080	25	217
Coastguard	352x288	30	300
CrowdRun	3840x2160	50	500
DucksTakeOff	1280x720	50	500
Foreman	352x288	30	300
News	352x288	30	300
ParkJoy	1920x1080	50	500
Silent	352x288	30	300
Station2	1920x1080	25	313
Sunflower	1920x1080	25	500

Fonte: Os Autores

Os dados foram colocados em *arrays* para que, no final da codificação de cada *frame*, as médias fossem calculadas e serem salvas em um arquivo CSV, junto com o valor de PSNR dividido pelo *bit rate* do *frame*.

Além dos atributos, também consta no CSV o nome do vídeo, qual aproximação foi utilizada e a qual *frame* do vídeo os dados da linha pertencem (Tabela 4.6[10]).

Tabela 4.6 – Organização do Primeiro Conjunto de Dados não preparado

Campo	**Tipo de Dado**	**Descrição**
video	Identificador	Nome do vídeo
aprox	Identificador	Aproximação utilizada
frame	Identificador	Numero do *frame*
min	Atributo	Atributo *min*
max	Atributo	Atributo *max*
sum	Atributo	Atributo *sum*
avg	Atributo	Atributo *avg*
abs_min	Atributo	Atributo *abs_min*
abs_max	Atributo	Atributo *abs_max*
abs_sum	Atributo	Atributo *abs_sum*
abs_avg	Atributo	Atributo *abs_avg*
psnr/bitrate	Valor de Comparação	Valor de comparação entre *frames*

Fonte: Os Autores

Foram coletados os 60 *frames* iniciais de cada vídeo do conjunto (do *frame* 0 até o *frame* 59), resultando em 600 *frames* para cada aproximação, totalizando 3.000 *frames*.

[10]Todos os campos utilizam 1 coluna na tabela

4.3.3 Segundo Conjunto de Dados

Para realizar a coleta do Segundo Conjunto de Dados, mudamos completamente a abordagem de coleta. Para podermos realizar uma comparação bloco a bloco e não por média do *frame*, dividimos os *frames* do vídeo original (antes da codificação) em blocos 8×8 e dividimos os *frames* do vídeo depois de ser codificado com cada aproximação e decodificado, também em blocos 8×8.

Com os *frames* divididos em blocos, para cada aproximação, coletamos os dados da Segunda Definição de Atributos e calculamos o MSE entre o bloco original e o bloco depois do processo de codificação e decodificação, ambos da camada de luminância Y. Um arquivo CSV é gerado com essa informações além do nome do vídeo, a aproximação, o *frame*, a linha e a coluna relativa a posição do bloco 8×8[11] (Tabela 4.7).

Tabela 4.7 – Organização do Segundo Conjunto de Dados não preparado

Campo	Tipo de Dado	Descrição	Colunas
video	Identificador	Nome do vídeo	1
aprox	Identificador	Aproximação utilizada	1
frame	Identificador	Numero do *frame*	1
linha	Identificador	Linha do bloco	1
coluna	Identificador	Coluna do bloco	1
avg	Atributo	Atributo *avg*	1
sd	Atributo	Atributo *sd*	1
max	Atributo	Atributo *max*	1
min	Atributo	Atributo *min*	1
sum	Atributo	Atributo *sum*	1
ent_sh	Atributo	Atributo *ent_sh*	1
ent_nat	Atributo	Atributo *ent_nat*	1
ent_hart	Atributo	Atributo *ent_hart*	1
fft	Atributo	Atributo *fft*	64
hu	Atributo	Atributo *hu*	7
skew	Atributo	Atributo *skew*	8
kurt	Atributo	Atributo *kurt*	8
freq	Atributo	Atributo *frq*	64
mse	Valor de Comparação	Valor de comparação entre *frames*	1

Fonte: Os Autores

Também mudamos quais *frames* de cada vídeo foram analisados, e agora, coletamos 1 *frame* por segundo do vídeo. Como cada vídeo possui um *frame rate* e uma quantidade de *frames*, a quantidade de *frames* analisada é diferente para cada vídeo.

[11] Como todo o *frame* foi dividido em blocos 8×8, a linha e a coluna representam a posição relativa a essa divisão do bloco. A posição absoluta desse bloco no *frame*, em *pixels*, é obtida multiplicando os valores de linha e coluna por 8.

Com esse conjunto de atributos, estamos trabalhando com coleta a nível de bloco, e neste caso, a quantidade de dados obtida foi muito maior. O total de blocos coletados foi de 3.188.160 e a distribuição deles entre os vídeos pode ser vista na Tabela 4.8.

Tabela 4.8 – Distribuição dos Blocos pelos Vídeos

Vídeo	**Quantidade de Blocos Coletados**
BlueSky	291.600
Coastguard	15.840
CrowdRun	1.296.000
DucksTakeOff	144.000
Foreman	15.840
News	15.840
ParkJoy	324.000
Silent	15.840
Station2	421.200
Sunflower	648.000

Fonte: Os Autores

4.4 Preparação dos Dados

A vantagem de realizar a própria coleta de dados é a possibilidade de organizar os dados de forma que o processamento na etapa de Preparação de Dados seja facilitada. Os processos entre os dois conjuntos de dados foram similares, mas falaremos deles de forma separada.

4.4.1 Preparação do Primeiro Conjunto de Dados

Com os dados organizados como na Tabela 4.6, o processo de preparação foi simples. Para cada *frame* de cada vídeo, comparamos os valores do campo ***psnr/bitrate*** entre as 5 aproximações. A aproximação que possuísse o maior valor, era considerada a melhor e seus atributos eram colocados no conjunto de dados final. Os atributos então foram normalizados utilizando a normalização por reescala *min-max*. Além disso, o campo ***aprox*** se torna o atributo-alvo dessa instância do conjunto de dados. O conjunto final possui 600 instâncias e sua organização está na Tabela 4.9[12] e a distribuição das aproximações está na Tabela 4.10.

[12]Todos os campos utilizam 1 coluna na tabela

Tabela 4.9 – Organização do Primeiro Conjunto de Dados

Campo	Tipo de Dado	Descrição
min	Atributo	Atributo *min*
max	Atributo	Atributo *max*
sum	Atributo	Atributo *sum*
avg	Atributo	Atributo *avg*
abs_min	Atributo	Atributo *abs_min*
abs_max	Atributo	Atributo *abs_max*
abs_sum	Atributo	Atributo *abs_sum*
abs_avg	Atributo	Atributo *abs_avg*
aprox	Atributo-alvo	Classe da Instância

Fonte: Os Autores

Tabela 4.10 – Distribuição do Primeiro Conjunto de Dados

Aproximação	Número de *Frames*	Proporção
RdCost8x8-1	149	24,83%
RdCost8x8-2	125	20,83%
RdCost8x8-3	104	17,33%
RdCost8x8-4	108	18,00%
RdCost8x8-SAD	114	19,00%

Fonte: Os Autores

4.4.2 Preparação do Segundo Conjunto de Dados

Utilizando os dados organizados conforme a Tabela 4.7, comparamos o valor de ***mse*** para cada bloco. A aproximação que tivesse o menor valor era considerada a melhor e seus atributos eram colocados no conjunto de dados final. Da mesma forma, os atributos então foram normalizados utilizando a normalização por reescala *min-max*. Como na primeira preparação, o campo ***aprox*** se tornou o atributo-alvo de cada instância. Sua organização está na Tabela 4.11 e a distribuição dos blocos em relação as aproximações na Tabela 4.12.

4.5 Treinamento dos Modelos

Para o treinamento dos modelos de predição, foi utilizada a biblioteca *scikit-learn* da linguagem *Python*. Os algoritmos foram escolhidos pela sua popularidade em trabalhos envolvendo ML, pela familiaridade do autor com tais algoritmos e na tentativa de utilizar algoritmo de implementação mais simplificada com a ideia futura de realizar a implementação do código no codificador e em um *hardware* acelerador dedicado. Para ambos conjuntos de dados, foram treinados modelos utilizando os seguintes algoritmos:

Tabela 4.11 – Organização do Segundo Conjunto de Dados

Campo	Tipo de Dado	Descrição	Colunas
avg	Atributo	Atributo *avg*	1
sd	Atributo	Atributo *sd*	1
max	Atributo	Atributo *max*	1
min	Atributo	Atributo *min*	1
sum	Atributo	Atributo *sum*	1
ent_sh	Atributo	Atributo *ent_sh*	1
ent_nat	Atributo	Atributo *ent_nat*	1
ent_hart	Atributo	Atributo *ent_hart*	1
fft	Atributo	Atributo *fft*	64
hu	Atributo	Atributo *hu*	7
skew	Atributo	Atributo *skew*	8
kurt	Atributo	Atributo *kurt*	8
freq	Atributo	Atributo *frq*	64
aprox	Atributo-alvo	Classe da Instância	1

Fonte: Os Autores

Tabela 4.12 – Distribuição do Segundo Conjunto de Dados

Aproximação	Número de Blocos	Proporção
RdCost8x8-1	630.881	19,79%
RdCost8x8-2	613.219	19,23%
RdCost8x8-3	638.141	20,02%
RdCost8x8-4	683.416	21,44%
RdCost8x8-SAD	622.503	19,53%

Fonte: Os Autores

Decision Tree, *K-NN*, *Logistic Regression*, *Naive Bayes* e *Random Forest*.

4.5.1 Definição de Hiperparâmetros

Para determinar alguns dos hiperparâmetros dos algoritmos *Decision Tree*, *K-NN*, *Logistic Regression* e *Random Forest*, a fim de otimizar o desempenho de cada modelo, foi utilizada a biblioteca *scikit-optimize*, que tenta, através de combinações e tentativas de um conjunto prévio de possibilidades de valores, estimar os melhores valores de hiperparâmetros para um algoritmo de ML. Na Tabela 4.13 é possível observar as possibilidades dos hiperparâmetros para cada algoritmo que foram exploradas nos dois conjuntos de dados.

Nas Tabelas 4.14 e 4.15 são demonstrados os valores devolvidos pelo algoritmo *skopt* (da biblioteca *scikit-optimize*) para o Primeiro Conjunto de Dados e para o Segundo

[13] Valor nulo, no caso da não utilização de penalidade

Tabela 4.13 – Possibilidades de Valores para Hiperpârametros

Hiperparâmetro	**Possibilidades**
Decision Tree	
criterion	gini, entropy, log_loss
splitter	best, random
K-NN	
algorithm	auto, ball_tree, kd_tree, brute
n_neighbors	Valor inteiro entre 1 e 24
weights	uniform, distance
Logistic Regression	
max_iter	Valor inteiro entre 100 e 1000
penalty	None[13], l2
solver	lbfgs, newton-cg, sag, saga
Random Forest	
criterion	gini, entropy, log_loss
n_estimators	Valor inteiro entre 50 e 500

Fonte: Os Autores

Conjunto de Dados respectivamente, em cada algoritmo.

Tabela 4.14 – Otimização de Hiperparâmetros do Primeiro Conjunto de Dados

Hiperparâmetro	**Valor Otimizado**
Decision Tree	
criterion	gini
splitter	best
K-NN	
algorithm	kd_tree
n_neighbors	22
weights	distance
Logistic Regression	
max_iter	834
penalty	l2
solver	newton-cg
Random Forest	
criterion	gini
n_estimators	115

Fonte: Os Autores

Na biblioteca *scikit-learn*, utilizando a versão *Gaussian Naive Bayes*, não existem hiperparâmetros para serem escolhidos, por isso, o algoritmo não faz parte do processo de otimização.

Tabela 4.15 – Otimização de Hiperparâmetros do Segundo Conjunto de Dados

Hiperparâmetro	**Valor Otimizado**
Decision Tree	
criterion	entropy
splitter	random
K-NN	
algorithm	auto
n_neighbors	23
weights	uniform
Logistic Regression	
max_iter	219
penalty	None
solver	lbfgs
Random Forest	
criterion	log_loss
n_estimators	402

Fonte: Os Autores

4.5.2 Treinamento e Avaliação

Para o treinamento e avaliação dos modelos, utilizamos a técnica KFCV. Os valores de K escolhidos foram 5 e 10 no caso do Primeiro Conjunto de Dados e 10 no caso do Segundo Conjunto de Dados. A análise foi feita sobre a média e o desvio padrão dos valores entre cada conjunto de *folds*, para cada métrica que será avaliada. Sendo assim, cada métrica é calculada pela média dos 5 ou 10 resultados de cada algoritmo para aquela métrica, junto com o desvio padrão obtido desse conjunto. Esse processo foi realizado utilizando a função *cross_validate*, do módulo *model_selection*, da biblioteca *scikit-learn*.

As métricas usadas são Acurácia, Precisão, Sensibilidade, Medida-F1 e Tempo de Predição. O Tempo de Predição é o tempo que o modelo levou para realizar a predição das instâncias de teste para cada modelo. É um valor referencial para comparar os algoritmos entre si em cada conjunto de dados, não sendo possível afirmar que possui algum valor avaliativo fora desse contexto.

As métricas que mais importam para a avaliação entre modelos, no contexto do codificador de vídeo, são a Acurácia e o Tempo de Predição, sendo que as outras métricas foram coletadas para fins de análises adicionais, caso necessário ou desejável.

5 RESULTADOS

Neste capítulo, são apresentados os resultados do treinamento dos modelos de cada algoritmo, para cada um dos conjuntos de dados, além de falarmos brevemente sobre uma possível implementação de algum dos modelos no codificador.

5.1 Avaliação do Primeiro Conjunto de Dados

Primeiro, falaremos da avaliação utilizando o número de *folds* igual a 10 no KFCV e, ao analisarmos os resultados, decidimos realizar novamente a avaliação utilizando o número do *folds* igual a 5.

5.1.1 KFCV Com 10 Folds

Na Tabela 5.1, são apresentados os dados relativos a avaliação dos modelos treinados com o Primeiro Conjunto de Dados com K igual a 10. As métricas estão em valores de porcentagem (de 0 a 100), com exceção do Tempo (abreviado de Tempo de Predição) que está em milissegundos. Além do valor da métrica, o valor de desvio padrão está entre parenteses depois de cada valor.

Os melhores resultados para cada métrica estão em negrito e as métricas mais importantes para nossa avaliação estão sublinhadas (Acurácia e Tempo).

Tabela 5.1 – Avaliação do Primeiro Conjunto de Dados (K = 10)

Modelo	Acurácia	Precisão	Sensibilidade	Medida-F1	Tempo (ms)[1]
Decision Tree	13,6 (6,1)	8,8 (4,5)	12,5 (5)	9,6 (3,9)	**5,89 (0,26)**
K-NN	16,4 (5,9)	10,5 (5,7)	14,9 (4,1)	10 (3,3)	6,67 (0,26)
Logistic Regression	**24,5 (8,4)**	15,7 (10,2)	22,3 (7,4)	14,1 (6,4)	6,03 (0,13)
Naive Bayes	23,8 (8,6)	16 (11,9)	**22,7 (8,3)**	**14,8 (7,9)**	6,36 (0,1)
Random Forest	17 (6,9)	**18,6 (10,3)**	15,7 (5,5)	12,6 (3,9)	12,66 (0,66)

Fonte: Os Autores

Como é possível observar, o modelo utilizando *Logistic Regression* obteve a melhor Acurácia e o modelo utilizando *Decision Tree* obteve o melhor Tempo. Se analisarmos o Tempo do modelo utilizando *Logistic Regression*, a média é ligeiramente maior que do melhor resultado, sendo 0,14 milissegundos mais lento. Já a Acurácia do melhor

[1] Em milissegundos

Tempo, o modelo utilizando *Decision Tree*, é muito pior que o desempenho do melhor resultado, perdendo 10,9% de Acurácia na média. Dessa forma, o modelo mais indicado levando em conta essas duas métricas seria o que utilizou o algoritmo *Logistic Regression*.

Observando os dados de forma geral, os desvios padrões tiveram um valor elevado se comparados com os valores das médias, chegando a 74,3% do valor da média no caso da Precisão do modelo utilizando *Naive Bayes* (16 de média e 11,9 de desvio padrão). Entre as possibilidades da causa, temos o número de instâncias utilizadas para o treinamento e avaliação do modelo (600 instâncias no caso do Primeiro Conjunto de Dados) somado ao número de folds, ou uma possível falta de correlação entre os atributos e o atributo-alvo.

Como os modelos possuem 5 possibilidades de atributo-alvo, a melhor média conseguiu ser apenas ligeiramente melhor que uma escolha aleatória entre as aproximações, que possui uma Acurácia de 20%, com o modelo utilizando o *Logistic Regression* com 24,5% de Acurácia.

Essa análise nos levou a realizar um novo treinamento e avaliação dos modelos utilizando o valor de K igual a 5, buscando, dessa forma, aumentar a quantidade de instâncias em cada *fold* e, por consequência, a quantidade de instâncias para teste.

5.1.2 KFCV Com 5 Folds

A avaliação do Primeiro Conjunto de Dados utilizando o K igual a 5 está na Tabela 5.2. Da mesma forma que os dados foram apresentados na Tabela 5.1, as métricas estão em valores de porcentagem, com exceção do Tempo que está em milissegundos, e os valores de desvios padrões estão entre parênteses juntos a cada valor.

Novamente, os melhores resultados para cada métrica estão em negrito e as métricas mais importante para a nossa avaliação estão sublinhados.

Tabela 5.2 – Avaliação do Primeiro Conjunto de Dados (K = 5)

Modelo	Acurácia	Precisão	Sensibilidade	Medida-F1	Tempo (ms)[2]
Decision Tree	19,3 (4,8)	16 (8,3)	18,1 (3,7)	13,9 (4,7)	**6,08 (0,62)**
K-NN	17,8 (4,3)	13,6 (6)	17 (2,9)	12 (2,9)	7,36 (0,26)
Logistic Regression	23,3 (4,4)	15,9 (2,8)	21,3 (3,1)	14,2 (3,3)	6,53 (0,37)
Naive Bayes	**25,6 (3,8)**	**21,9 (3,2)**	**24,1 (2,8)**	**18,2 (4,6)**	6,14 (0,21)
Random Forest	15,5 (2,8)	14 (5)	15,3 (2,6)	12,2 (2,8)	15,49 (0,8)

Fonte: Os Autores

[2]Em milissegundos

Os valores das médias das métricas permaneceram parecidos com os valores obtidos com o K igual a 10. Aqui vemos os modelos utilizando *Logistic Regression* e *Naive Bayes* trocaram de posições na questão da Acurácia, com o *Naive Bayes* obtendo o melhor resultado com 25,6% de Acurácia. O melhor tempo ainda foi obtido pelo modelo utilizando *Decision Tree*.

É nos desvios padrões onde é possível ver uma mudança maior. Com exceção de três valores, todos os valores de desvio padrão foram menores, o que pode ser considerado um aceno na direção das possibilidades levantadas antes sobre a causa dos valores altos de desvio padrão.

Outro ponto de interesse é o desempenho do modelo utilizando *Naive Bayes*, que obteve o melhor resultado em todas métricas, exceto o Tempo. Porém, seu resultado foi apenas 0,06 milissegundos pior que o melhor resultado, enquanto sua Acurácia foi 6,3% do modelo mais rápido. Levando isso em conta, o modelo utilizando o algoritmo *Naive Bayes* se mostra como a melhor alternativa nesse conjunto.

Apesar do desempenho melhor de certa forma, a melhor Acurácia ainda foi abaixo do desejado, sendo ainda ligeiramente melhor que uma escolha aleatória na média. Com esses resultados em mãos, optamos por tentar uma nova abordagem com o objetivo de obtermos modelos com melhor desempenho. E assim mudamos nosso foco para o desenvolvimento do Segundo Conjunto de Dados.

5.2 Avaliação do Segundo Conjunto de Dados

Apesar da diferença dos atributos e método de coleta dos dados, o Segundo Conjunto de Dados foi avaliado da mesma forma que o Primeiro Conjunto de Dados, como é possível ver na Tabela 5.3.

O K do KFCV foi estabelecido em 10, e como nos casos anteriores, as métricas estão em valores de porcentagem. A única diferença aqui está na métrica Tempo, que está em segundos. Além disso, temos alguns casos onde o desvio padrão foi muito menor que o valor da média e, por isso, desconsiderado.

O melhor desempenho de Acurácia foi do modelo utilizando *Logistic Regression*, mas importante notar que a diferença entre ele e o penúltimo modelo é de menos de 1%. O melhor Tempo foi do modelo utilizando *Naive Bayes*, com *Decision Tree* e *Logistic Regression* com valores bem próximos. Os outros dois modelos obtiveram resultados bem maiores no Tempo, se mostrando bem mais sensíveis ou à quantidade de instâncias ou à

quantidade de atributos. Os valores de Tempo aumentaram significativamente para este Conjunto de Dados pela quantidade total maior de inferências realizadas pelos modelos na etapa de teste.

Tabela 5.3 – Avaliação do Segundo Conjunto de Dados

Modelo	Acurácia	Precisão	Sensibilidade	Medida-F1	Tempo (s)[3]
Decision Tree	20,2 (0,1)	20,2 (0,1)	20,2 (0,1)	20,2 (0,1)	0,49 (-)[4]
K-NN	20,6 (0,2)	20,5 (0,2)	20,5 (0,2)	20,4 (0,2)	9,11 (0,05)
Logistic Regression	**21,3 (0,3)**	20,3 (0,3)	20,6 (0,2)	18,1 (0,2)	0,54 (-)[4]
Naive Bayes	19,4 (-)[5]	**21,2 (6,7)**	20,2 (-)[5]	8 (0,4)	**0,44 (-)**[4]
Random Forest	21,1 (0,2)	20,6 (0,2)	**20,8 (0,2)**	**20,5 (0,1)**	5,48 (-)[4]

Fonte: Os Autores

Em comparação com os resultados do Primeiro Conjunto de Dados, podemos notar grandes diferenças com facilidade. As médias das métricas muito mais próximas umas das outras, com apenas uma discrepância (a Medida-F1 do modelo *Naive Bayes*), em métricas como o Sensibilidade, a diferença foi na casa decimal da porcentagem.

Da mesma forma, os valores de desvios padrões foram muito menores no Segundo Conjunto de Dados, em casos sendo até descartado pelo seu valor muito menor que a média. Da mesma forma que as médias, tivemos apenas uma discrepância (a Precisão do modelo *Naive Bayes*). Isso reforça mais a possibilidade da quantidade de instâncias estar diretamente ligado com a variação das médias.

O desempenho dos modelos, apesar de mais estáveis, foram menores que o melhor resultado do Primeiro Conjunto de Dados, ficando mais próximos ainda de uma escolha aleatória. Apesar do desempenho inferior, a estabilidade dos modelos do Segundo Conjunto de Dados não pode ser descartada. Dessa forma, existem contextos onde eles poderiam ser mais indicados que os outros modelos.

5.3 Implementação

Levando em conta o desempenho de ambos conjuntos de dados, com o melhor modelo do Primeiro Conjunto de Dados tendo uma Acurácia média de 24,5% para K igual a 10 e 25,6% para K igual a 5, e o melhor modelo do Segundo Conjunto de Dados tendo uma Acurácia média de 21,3%, foi avaliado que a possibilidade de ganho é muito baixa para o aumento de complexidade, processamento e tempo que o codificador teria

[3] Em segundos
[4] Valor menor que 0,01
[5] Valor menor que 0,1

com a implementação de um modelo de predição.

Assim, optamos por não realizar a implementação de nenhum dos modelos de predição no código do codificador. Outra possibilidades foram debatidas e elas serão discutidas um pouco mais a fundo no Capítulo 6, quando falarmos sobre possíveis trabalhos futuros.

6 CONCLUSÃO

Neste trabalho, foram apresentados modelos de predição que tinham como objetivo selecionar entre aproximações do SATD e o SAD, qual cálculo deveria ser realizado para obter a estimativa de distorção de blocos 8×8 durante o processo de codificação. Utilizamos dois Conjuntos de Dados completamente distintos, com atributos, quantidade de instâncias e método de coleta diferentes, afim de tentar uma abordagem diferente após obtermos resultados pouco promissores com a avaliação do primeiro Conjunto de Dados.

O maior desafio foi o desenvolvimento de um Conjunto de Dados capaz de fornecer as informações necessárias para os modelos de predição. A princípio, era imaginado que seria possível estabelecer uma correlação entre as características de um bloco e a aproximação que apresentou o melhor resultado em sua codificação, seja ele a razão entre a qualidade objetiva e a taxa de *bits* (PSNR e *bit rate*) ou a menor média do erro quadrático (MSE). Porém, com os resultados da avaliação dos modelos, não foi possível comprovar essa correlação, ao menos para os atributos selecionados na produção de ambos os Conjuntos de Dados.

Assim, no final da etapa de avaliação dos modelos treinados, o modelo com melhor Acurácia acabou tendo um desempenho ligeiramente melhor (5,6%) que uma escolha aleatória, um desempenho que não conseguiu justificar a implementação de nenhum dos modelos ao código do codificador afim de uma avaliação durante o processo de codificação.

Entre os possíveis motivos para o desempenho abaixo do desejado, é possível destacar os atributos escolhidos no desenvolvimento dos dois Conjuntos de Dados, o tipo de aprendizado utilizado na aplicação do ML e o número elevado de possibilidades de classificações para uma diferença muito sutil de valores de atributos entre as instâncias da classe.

6.1 Trabalhos Futuros

Considerando os possíveis motivos que contribuíram para o desempenho abaixo do desejado dos modelos de predição, é possível pensar em algumas possibilidades para trabalhos futuros sobre o tema.

6.1.1 Aprendizado Não Supervisionado ou Aprendizado por Reforço

Uma possibilidade seria a utilização de outros tipos de aprendizado, como Aprendizado Não Supervisionado ou Aprendizado por Reforço. Em uma abordagem de Aprendizado Não Supervisionado, seria possível tentar identificar os *clusters* formados pelos algoritmos e realizar uma análise para tentar compreender quais as principais características que poderiam apontar para o uso de uma ou outra aproximação do SATD. Já em uma abordagem de Aprendizado por Reforço, uma das possibilidades seria utilizar a razão entre o PSNR e o *bit rate* de cada *frame* como medida avaliativa da ação tomada pelo modelo durante a codificação desse *frame*.

6.1.2 Análise de Correlação do Bloco e a Qualidade

Outra possibilidade seria o aprofundamento de uma análise das informações que podem ser obtidas em um bloco durante o processo de cálculo do SATD e métricas de avaliação da qualidade da imagem (PSNR), da compressão (*bit rate*) ou ambos (BD-Rate). Além dos atributos utilizados neste trabalho, existem inúmeras outras abordagens que podem ser tomadas para uma tentativa de obtenção de informações relevantes dos blocos, como por exemplo, um processamento mais complexo do que os utilizados aqui, afim de buscar entender melhor essa possível correlação já citada.

6.1.3 Quantidade de Possibilidades de Aproximação

Outra possibilidade seria abordar outro número de possíveis aproximações que o modelo deve fazer a predição. Por exemplo, uma abordagem de predição binária entre duas aproximações de SATD ou entre uma aproximação de SATD e o SAD. É possível que um número menor de aproximações apresente um comportamento melhor para uma abordagem de Aprendizado Supervisionado.

REFERÊNCIAS

ABU-MOSTAFA, Y.; MAGDON-ISMAIL, M.; LIN, H. **Learning from Data: A Short Course**. AMLBook, 2012. ISBN 9781600490064. Disponível na Internet: <https://books.google.com.br/books?id=iZUzMwEACAAJ>.

BJONTEGAARD, G. Calculation of average psnr differences between rd curves. **ITU-T SG16/Q6, 13th VCEG Meeting, Austin, Texas, USA, April 2001**, 2001. Disponível na Internet: <https://cir.nii.ac.jp/crid/1570572700525852416>.

BROSS, B. et al. Developments in international video coding standardization after avc, with an overview of versatile video coding (vvc). **Proceedings of the IEEE**, v. 109, n. 9, p. 1463–1493, 2021.

BROSS, B. et al. Overview of the versatile video coding (vvc) standard and its applications. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 31, n. 10, p. 3736–3764, 2021.

DINIZ, C. M. **Dedicated and reconfigurable hardware accelerators for high efficiency video coding standard**. Thesis (PhD) — Universidade Federal do Rio Grande do Sul, 2015.

FACELI, K. et al. **Inteligência artificial: uma abordagem de aprendizado de máquina**. 1. ed. Rio de Janeiro: LTC, 2011.

LIMA, V. H. S. et al. Configurable approximate hardware accelerator to compute satd and sad metrics for low power all-intra high efficiency video coding. In: **2021 34th SBC/SBMicro/IEEE/ACM Symposium on Integrated Circuits and Systems Design (SBCCI)**. [S.l.: s.n.], 2021. p. 1–6.

LIOU, M. Overview of the p×64 kbit/s video coding standard. **Commun. ACM**, Association for Computing Machinery, New York, NY, USA, v. 34, n. 4, p. 59–63, abr. 1991. ISSN 0001-0782. Disponível na Internet: <https://doi.org/10.1145/103085.103091>.

MITCHELL, T. **Machine Learning**. [S.l.]: McGraw-Hill Education, 1997. (McGraw-Hill international editions - computer science series). ISBN 9780070428072.

RICHARDSON, I. E. G. **The H.264 Advanced Video Compression Standard**. 2. ed. [S.l.]: John Wiley & Sons, 2010. ISBN 978-0-470-51692-8.

ROLLINS, J. B. **Metodologia de Base para Ciência de Dados**. 2015. Disponível na Internet: <https://www.ibm.com/downloads/cas/B1WQ0GM2>.

SANTOS, E. **Alocação de Recursos baseada em Clustering com Aprendizado de Características e Orientação a QoS em Redes LTE-Advanced**. Thesis (PhD) — Universidade Federal de Uberlândia, 06 2019.

SOARES, L. B. et al. A novel pruned-based algorithm for energy-efficient satd operation in the hevc coding. In: **2016 29th Symposium on Integrated Circuits and Systems Design (SBCCI)**. [S.l.: s.n.], 2016. p. 1–6.

STIGGER, M. F. et al. Approximate sum of absolute transformed differences hardware accelerator. In: **2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS)**. [S.l.: s.n.], 2019. p. 53–56.

STIGGER, M. F. et al. Approximate satd hardware accelerator using the 8×8 hadamard transform. In: **2020 IEEE 11th Latin American Symposium on Circuits 'I& Systems (LASCAS)**. [S.l.: s.n.], 2020. p. 1–4.