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

Tese

Redução do Tempo de Codificação na Predição Interquadros do Padrão *Versatile Video Coding* (VVC)

Marta Breunig Loose

Pelotas, 2025

Marta Breunig Loose

Redução do Tempo de Codificação na Predição Interquadros do Padrão *Versatile Video Coding* (VVC)

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

Orientador: Prof. Dr. Luciano Volcan Agostini
Coorientadores: Prof. Dr. Guilherme Ribeiro Corrêa
Prof. Dr. Gustavo Freitas Sanchez

Pelotas, 2025

Universidade Federal de Pelotas / Sistema de Bibliotecas
Catalogação da Publicação

L863r Loose, Marta Breunig

Redução do tempo de codificação na predição interquadros do padrão *Versatile Video Coding* (VVC) [recurso eletrônico] / Marta Breunig Loose ; Luciano Volcan Agostini, orientador ; Guilherme Ribeiro Corrêa, Gustavo Freitas Sanchez, coorientadores. — Pelotas, 2025.
205 f. : il.

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

1. VVC. 2. Interquadros. 3. Tempo de codificação. 4. Aprendizado de Máquina. I. Agostini, Luciano Volcan, orient. II. Corrêa, Guilherme Ribeiro, coorient. III. Sanchez, Gustavo Freitas, coorient. IV. Título.

CDD 005

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

Marta Breunig Loose

Redução do Tempo de Codificação na Predição Interquadros do Padrão *Versatile Video Coding* (VVC)

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

Data da Defesa: 28 de março de 2025

Banca Examinadora:

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

Prof. Dr. Daniel Munari Vilchez Palomino
Doutor em Computação pela Universidade Federal do Rio Grande do Sul.

Prof. Dr. Claudio Machado Diniz
Doutor em Computação pela Universidade Federal do Rio Grande do Sul.

Prof. Dr. Iago Coelho Storch
Doutor em Computação pela Universidade Federal do Rio Grande do Sul.

AGRADECIMENTOS

Agradeço primeiramente a Deus, autor da vida e criador do universo. Pela graça infinita, misericórdia diária e pelo amor incondicional.

Agradeço imensamente ao meu esposo Luís, que me apoiou nesse período do doutorado. Obrigada por ser um marido e pai presente, apoiando efetivamente quando eu mais precisei. Pelos conselhos e abraços que sempre transmitiram paz e acolhimento. Pela paciência, carinho e amor.

Agradeço de todo o coração ao meu filho Mathias, que nesses cinco anos cresceu, passando pelas fases de ser um bebê até um menino “grande” e independente em várias atividades. Passamos por tantos desafios e dificuldades, principalmente neste último ano. Obrigada que, mesmo sem entender muito bem, você teve paciência em todos esses momentos e conseguiu se adaptar quando foi necessário. Você me motiva e me ensina a ser uma pessoa melhor a cada dia. Mathias, amo você e sou muito feliz por ser sua mãe! Agora eu terminei meu trabalho do doutorado!

Agradeço aos meus pais, Clari e Carlos, pela vida e por contribuírem, cada um da sua maneira, para minha formação pessoal e profissional. Obrigada mãe, por ser exemplo de motivação, força e resiliência. Obrigada pai, pelo carinho, apoio e por sempre ter me incentivado a estudar.

A todos meus familiares, pela torcida, motivação e apoio durante esse período. Em especial aos meus sogros, Haidi e Sídio, por cuidarem do Mathias em alguns momentos que precisei me dedicar ao doutorado e também aos meus cunhados, Laís e Moizes, pela troca de experiências e pelo incentivo nesse período.

À minha prima-amiga-irmã Dani, pelo carinho e amizade incondicionais. Obrigada por cada mensagem que, mesmo indiretamente, me trouxe apoio, alegria e alívio nesse período.

À minha psicóloga Ana, agradeço por me ajudar a superar cada desafio e a enxergar cada potencialidade minha. Por contribuir imensamente no meu processo de autoconhecimento e na reconstrução da minha autoestima. Sua dedicação e apoio tornaram esse período do doutorado mais leve.

Também gostaria de agradecer às minhas colegas queridas do IFFar, Lara, Manuela, Andréa, Cristiane, Marieli e Andressa, pela torcida e pelo apoio de sempre!

Agradeço em especial ao meu orientador, professor Luciano Volcan Agostini, pela acolhida e pelo incentivo desde o primeiro momento. Obrigada pela sua conduta humanizada nesse período, fez toda a diferença para que eu não desistisse. Agradeço também pelas orientações, correções e contribuições valiosas para o trabalho.

Agradeço aos coorientadores, professores Guilherme Corrêa e Gustavo Freitas Sanchez, pelas orientações, correções e contribuições que foram fundamentais para a finalização desta pesquisa.

Não poderia deixar de agradecer imensamente ao Ramiro, que desde o início me apoiou efetivamente em várias tarefas, desde estudar sobre o VVC ou rodar experimentos no servidor da UFPel, até elaborar imagens e organizar a escrita dos artigos. Você merece todo o sucesso!

Agradeço também ao Fernando, que durante o seu mestrado me auxiliou em várias tarefas, contribuindo para o aprendizado sobre o VVC. Aproveito para agradecer aos alunos de graduação, Rafael e Bianca, que pude ser coorientadora de seus TCCs, sendo que seus trabalhos também contribuíram para esta tese.

A todos os colegas do ViTech, em especial Adson, Alex, Mário e Matheus, pelas contribuições para este trabalho.

Também agradeço às colegas do doutorado, Renira, Karlise, Bianca e Roberta, pela troca de experiências, mas principalmente pela empatia e companheirismo, demonstrados em cada conversa. Desejo sucesso a vocês!

Agradeço ao IFFar, por realizar a parceira com a UFPel e ofertar o Dinter em Ciência da Computação. Agradeço à direção e aos colegas do IFFar campus Santo Ângelo, por viabilizar as horas de afastamento semanal para minha participação na ação de desenvolvimento em serviço.

Agradeço a UFPel e ao Programa de Pós-graduação em Computação (PPGC), por oportunizar o curso de Doutorado em Ciência da Computação, até mesmo durante a Pandemia. De maneira especial, agradeço aos docentes do PPGC, que sempre demonstraram profissionalismo e excelência em todas as aulas e atividades.

*"Essa **tese** foi feita sob a influência da erva do chimarrão, que se chama Ilex Paraguariensis".*
— HUMBERTO GESSINGER (PARÁFRASE)

RESUMO

LOOSE, Marta Breunig. **Redução do Tempo de Codificação na Predição Interquadros do Padrão *Versatile Video Coding* (VVC)**. Orientador: Luciano Volcan Agostini. 2025. 205 f. Tese (Doutorado em Ciência da Computação) – Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2025.

Atualmente há uma crescente demanda por vídeos digitais de altas resoluções, principalmente através da Internet. Diferentes plataformas permitem a transmissão, o compartilhamento e o consumo de vídeos online. Durante a pandemia pela COVID-19, esse contexto se intensificou, demonstrando a importância da infraestrutura de rede e de estratégias para a compressão de vídeos. Nesse sentido, o padrão *Versatile Video Coding* (VVC) fornece novas ferramentas que possibilitam maiores taxas de compressão, comparado aos padrões anteriores, além de maior adaptabilidade aos diferentes tipos de vídeos. O padrão VVC, assim como os demais codificadores atuais, é baseado no modelo híbrido de codificação de resíduos de predições. Suas principais etapas consistem na predição intraquadro e interquadros, Transformada, Quantização e Codificação de Entropia. Entretanto, juntamente com essas características, o VVC detém um alto grau de complexidade, acarretando em um tempo de codificação de vídeo muito alto. Sendo assim, são muitos os desafios que ainda estão em aberto, a fim de diminuir esse problema. Diante disso, esta tese visa apresentar soluções para a redução do tempo de codificação na predição interquadros do padrão VVC. Além da contextualização sobre o padrão VVC e da interquadros em si, também são apresentados conceitos de aprendizado de máquina, além de resultados de uma densa análise experimental e de uma revisão sistemática da literatura. Por fim, são apresentadas quatro soluções de otimização focadas na predição interquadros, sendo uma heurística, com base em análise estatística e outras três soluções que utilizam aprendizado de máquina. A heurística é focada nas predições Unidirecional, Bidirecional e *Affine*, sendo configurável para três pontos de operação. Essa solução obteve resultados de redução no tempo de codificação de até 26,5%, para as etapas Unidirecional e Bidirecional, e de até 22,71% para a etapa *Affine*, com perda de eficiência de codificação de 0,9% e 0,44%, respectivamente. A solução usando *Random Forest* focada na Bidirecional, alcançou 92% em média de redução de tempo na etapa e 2% no tempo total de codificação, com perda de eficiência de codificação de 0,75%. Já a solução ampliada, adicionando modelos *Decision Tree* focados na *Affine* à solução anterior, alcançou redução de tempo total de 3,9%, com perda de eficiência de codificação de 0,72%. Por fim, a solução focada na otimização da *Affine*, utilizando modelos *Decision Tree*, obteve redução média de 42,1% no tempo da etapa, 3,4% no tempo total de codificação, com impacto de

0,25% na eficiência de codificação. Esses resultados demonstram que a utilização de técnicas de aprendizado de máquina são caminhos muito promissores para alcançar redução no tempo de codificação da predição interquadros com impactos mínimos na eficiência de compressão.

Palavras-chave: VVC; Interquadros; Tempo de codificação; Aprendizado de Máquina.

ABSTRACT

LOOSE, Marta Breunig. **Encoding Time Reduction in Versatile Video Coding (VVC) Inter-frame Prediction**. Advisor: Luciano Volcan Agostini. 2025. 205 f. Thesis (Doctorate in Computer Science) – Technology Development Center, Federal University of Pelotas, Pelotas, 2025.

There is a growing demand for high-resolution digital videos, mainly over the Internet. Different platforms allow the transmission, sharing, and consumption of videos online. During the COVID-19 pandemic, this context intensified, demonstrating the importance of network infrastructure and strategies for video compression. In this sense, the Versatile Video Coding (VVC) standard provides new tools that enable higher compression ratios than previous standards and greater adaptability to different types of videos. The VVC standard and other modern encoders are based on the hybrid model of prediction residual coding. Its main steps include intra-frame and inter-frame prediction, Transforms, Quantization, and Entropy coding. However, along with these characteristics, VVC is highly complex, resulting in a very high video encoding time. Therefore, many challenges are still open to reduce this problem. In light of this, this thesis aims to present solutions for reducing the encoding time in inter-frame prediction of the VVC standard. In addition to providing context on the VVC standard and inter-frame prediction itself, concepts of machine learning are also presented, along with results from a dense experimental analysis and a systematic literature review. Finally, four optimization solutions focused on inter-frame prediction are presented: one heuristic based on statistical analysis and three other solutions employing machine learning. The heuristic focuses on unidirectional, bidirectional, and Affine predictions, and can be configured for three operating points. This solution achieved encoding time reductions of up to 26.5% for the unidirectional and bidirectional stages and up to 22.71% for the Affine stage, with a coding efficiency loss of 0.9% and 0.44%, respectively. The solution using *Random Forest* focused on bidirectional achieved an average reduction of 92% in time per step and 2% in total encoding time, with a coding efficiency loss of 0.75%. The expanded solution, adding *Decision Tree* models focused on *Affine* to the previous solution, achieved a total time reduction of 3.9%, with a coding efficiency loss of 0.72%. Finally, the solution focused on Affine optimization, using *Decision Tree* models, obtained an average reduction of 42.1% in time per step, 3.4% in total time, with an impact of 0.25% on coding efficiency. Therefore, these results demonstrate that the use of machine learning techniques are the most promising ways to achieve a reduction in interframe coordination time with minimal impacts on update efficiency.

Keywords: VVC; Inter-frame; Encoding time; Machine Learning.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACM	*Association for Computing Machinery*
ALF	Filtro de Laço Adaptável, do inglês *Adaptive Loop Filter*
AME	Estimação de Movimento *Affine*, do inglês *Affine Motion Estimation*
AMVP	Predição Avançada do Vetor de Movimento, do inglês *Advanced Motion Vector Prediction*
AMVR	Resolução de Vetor de Movimento Adaptável, do inglês *Adaptive Motion Vector Resolution*
AOMedia	Alliance for Open Media
AVC	Codificação de Vídeo Avançada, do inglês *Advanced Video Coding*
BCW	Bi-predição com Pesos a nível de Unidade de Codificação do inglês *Bi-prediction with CU-level Weights*
BDOF	Fluxo Óptico Bidirecional, do inglês *Bi-directional Optical Flow*
BDBR	*Bjøntegaard Delta Bit Rate*
CABAC	Codificação Aritmética Binária Adaptativa Baseada em Contexto, do inglês *Context-based Adaptive Binary Arithmetic Coding*
CB	Bloco de Codificação, do inglês *Coding Block*
CBF	Sinalizador de Bloco Codificado, do inglês *Coded Block Flag*
CC-ALF	Filtro de *loop* adaptativo entre componentes, do inglês *Cross-Component Adaptive Loop Filter*
CIIP	Predição Inter/Intraquadro Combinada, do inglês *Combined Inter/Intra-picture Prediction*
CNN	Rede Neural Convolucional, do inglês *Convolutional Neural Network*
CP	Ponto de Controle, do inglês *Control Point*
CPMV	Vetor de Movimento do Ponto de Controle, do inglês (*Control Point Motion Vector*
CU	Unidade de Codificação, do inglês *Coding Unit*
CTB	Bloco de Árvore de Codificação, do inglês *Coding Tree Block*
CTC	Condições Comuns de Teste, do inglês *Common Test Conditions*

CTU	Unidade de Árvore de Codificação, do inglês *Coding Tree Unit*
DBF	Filtro de Deblocagem, do inglês *Deblocking Filter*
DCT	Transformada de Cosseno Discreto, do inglês *Discrete Cosine Transform*
DMVR	Refinamento de Vetor de Movimento no Decodificador, do inglês *Decoder-side Motion Vector Refinement*
DST	Transformada de Seno Discreto, do inglês *Discrete Sine Transform*
DT	Árvore de Decisão, do inglês *Decision Tree*
FI	Importância do Recurso, do inglês *Feature Importance*
FIFO	Primeiro a Entrar e Primeiro a Sair, do inglês *First In First Out*
FPME	Prévia da Estimação de Movimento Fracionária, do inglês *Fractional Preview Motion Estimation*
FME	Estimação de Movimento Fracionária, do inglês *Fractional Motion Estimation*
FN	Falsos Negativos, do inglês *False Negatives*
FP	Falsos Positivos, do inglês *False Positives*
FHD	Alta Definição Completa, do inglês *Full High Definition*
GOP	Grupo de Quadros, do inglês *Group of Pictures*
GPM	Modo de Particionamento Geométrico, do inglês *Geometric Partitioning Mode*
HEVC	*High Efficiency Video Coding*
HD	Alta Definição, do inglês *High Definition*
HM	*HEVC Test Model*
HMVP	Derivação de Candidatos ao *Merge* com Base no Histórico, do inglês *History-based merge candidates derivation*
IA	Inteligência Artificial
IEEE	*Institute of Electrical and Electronics Engineers*
IG	Ganho de informação, do inglâs *Information Gain*
IME	Estimação de Movimento Inteira, do inglês *Integer Motion Estimation*
IPME	Prévia da Estimação de Movimento Inteira, do inglês *Integer Preview Motion Estimation*
ISP	Sub-Partições Intra, do inglês *Intra Sub-Partitions*
LNN	Rede Neural Leve, do inglês *Lightweight Neural Network*
MC	Compensação de Movimento, do inglês *Motion Compensation*
ME	Estimação de Movimento, do inglês *Motion Estimation*
MV	Vetor de Movimento, do inglês *Motion Vector*
MVP	Predição do Vetor de Movimento, do inglês *Motion Vector Prediction*

MVD	Diferença do Vetor de Movimento, do inglês *Motion Vector Difference*
MIP	Predição Intra Ponderada por Matriz, do inglês *Matrix-based Intra Prediction*
MPEG	*Moving Picture Experts Group*
MRL	Múltiplas Linhas de Referência, do inglês *Multiple Reference Line*
MTS	Seleção Múltipla de Transformada, do inglês *Multiple Transform Selection*
POC	Contagem da Ordem da Imagem, do inglês *Picture Order Count*
PSNR	Razão Pico de Sinal para Ruído, do inglês *Peak Signal-to-Noise Ratio*
PU	Unidade de Predição, do inglês *Prediction Unit*
QP	Parâmetro de Quantização, do inglês *Quantization Parameter*
QTMT	Árvore Quádrupla com Árvore de Tipos Múltiplos Aninhada, do inglês *Quad-tree with nested Multi-type Tree*
RD	Taxa-distorção, do inglês *Rate-Distortion*
RDO	Otimização da taxa-distorção, do inglês, *Rate-Distortion Optimization*
RF	Florestas Aleatórias, do inglês *Random Forests*
RSL	Revisão Sistemática da Literatura
SAD	Soma das Diferenças Absolutas, do inglês, *Sum of Absolute Differences*
SAO	Compensação de Amostra Adaptável, do inglês *Sample Adaptive Offset*
SbTMVP	Predição de Vetor de Movimento baseada em Sub-blocos Temporais, do inglês *Subblock-based Temporal Motion Vector Prediction*
SVM	Máquina de Vetores de Suporte, do inglês *Support Vector Machine*
SMVD	Codificação MVD Simétrica, do inglês *Symmetric Motion Vector Difference Coding*
SVH	Sistema Visual Humano
TN	Verdadeiros Negativos, do inglês *True Negatives*
TP	Verdadeiros Positivos, do inglês *True Positives*
UHD	Ultra Alta Definição, do inglês *Ultra High Definition*
UVG	*Ultra Video Group*
VCEG	*Video Coding Experts Group*
VVC	*Versatile Video Coding*
VTM	*VVC Test Model*
WP	Predição Ponderada, do inglês *Weighted Prediction*
YUV	Luminância (Y), Crominância azul (U), Crominância Vermelha (V)

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos, o acesso e o consumo de vídeos digitais através da Internet aumentaram significativamente. De acordo com a Cisco, em 2023 o número de usuários conectados à Internet chegou a 5,3 bilhões, ou seja, cerca de 66% da população global. Já o número de dispositivos conectados neste mesmo ano alcançou a marca de 29 bilhões, isto é, 3,6 por pessoa (Cisco, 2023). São várias as plataformas disponíveis para a transmissão, o compartilhamento e o consumo de vídeos online. Esse cenário tornou-se ainda mais drástico no contexto da pandemia pela COVID-19, pois, ao permanecer mais tempo em casa, as pessoas utilizaram ainda mais esses serviços. Nesse sentido, a adoção do trabalho remoto é um fator que influenciou o aumento significativo de reuniões por vídeo chamada, por exemplo. Uma pesquisa da Nielsen (Dooley, 2020) indicou que os americanos consumiram mais de 123 bilhões de minutos em transmissão de vídeos na semana do dia 20 de julho de 2020, quando as regras de distanciamento social já estavam mais brandas. Já no Brasil, em uma pesquisa da mesma empresa (Brasil, 2020), 93,2% dos entrevistados afirmaram que utilizam a Internet todos os dias para acessar vídeos, filmes e programas de TV.

Diante desse cenário, é possível perceber a importância da infraestrutura de rede e da utilização de estratégias de compressão de vídeos, visto que são necessários muitos dados para representá-los (Zhang; Mao, 2019). Em geral, a expectativa é de que, ao consumir vídeos digitais, os mesmos apresentem uma boa qualidade e rapidez de acesso, sem gastar muitos recursos (banda e energia). Neste contexto, os codecs (codificadores e decodificadores) de vídeo têm um papel fundamental, pois realizam a compressão dos sinais dos vídeos, garantindo a eficiência de armazenamento e transmissão dos mesmos (Zhang; Mao, 2019).

Alguns padrões de codificação de vídeo foram definidos ao longo do tempo e são resultantes de um trabalho integrado entre grupos como o MPEG (*Moving Picture Experts Group*) [1] e VCEG (*Video Coding Experts Group*) [2]. Fazem parte desses resultados os padrões H.264/AVC (*Advanced Video Coding*) (Sullivan; Wiegand, 2005),

[1] `https://www.mpeg.org/`

[2] `https://www.itu.int/en/ITU-T/studygroups/2013-2016/16/Pages/video/vceg.aspx`

H.265/HEVC (*High Efficiency Video Coding*) (Sullivan, et al., 2012) e, o mais recente, H.266/VVC (*Versatile Video Coding*) (Bross; Wang; Ye; Liu; Chen; Sullivan; Ohm, 2021) lançado em julho de 2020. Além disso, também existem outros codificadores, desenvolvidos de maneira independente por grandes empresas, tais como Google, Microsoft, Mozilla e Cisco. Em meados de 2015, essas e outras empresas formaram a *Alliance for Open Media* (AOMedia), com o objetivo de desenvolver um codificador de vídeo livre de royalties, chamado de AV1 (*AOMedia Video 1*) (Chen et al., 2018).

Este trabalho está focado no padrão VVC, visto que o mesmo é o atual padrão estado da arte e tende a ser utilizado pela indústria a fim de suprir as novas necessidades em relação à compressão de vídeo. Assim como os demais codecs, o VVC inclui novas ferramentas que aumentam a taxa de compressão. Em comparação com o seu antecessor HEVC, o VVC é capaz de reduzir em 50% a taxa de bits, mantendo a mesma qualidade do vídeo (BROSS et al., 2021). Entretanto, essas mesmas ferramentas contribuem para o aumento significativo no custo computacional e, consequentemente, no tempo de codificação (Saldanha et al., 2020). Esse fator pode prejudicar a adoção do padrão em situações que exigem alta performance em relação ao tempo de codificação e uso de energia, como, por exemplo, transmissões de vídeo em tempo real e acesso através de dispositivos móveis.

1.1 Questão e Hipótese de Pesquisa

Considerando o que foi exposto até aqui, é possível perceber que a questão do custo computacional do VVC tende a ser um desafio importante para a ampla adoção deste padrão de compressão de vídeos. Então, esta tese explora soluções heurísticas e com o uso de aprendizado de máquina para reduzir esse custo computacional elevado do VVC, visando sempre o menor impacto possível em termos de eficiência de codificação. O foco das soluções desenvolvidas está na predição interquadros, em função do seu elevado custo computacional, conforme ficará mais claro no decorrer deste texto. A partir dessa definição inicial, foi elaborada a questão de pesquisa que norteou este trabalho:

Como reduzir o custo computacional da predição interquadros do VVC, com impactos mínimos na eficiência de codificação?

Com base nessa questão, a hipótese é que o uso de heurísticas baseadas em análise estatística e, principalmente, baseadas no uso de aprendizado de máquina, seria o caminho mais promissor para responder à questão de pesquisa.

Todas as soluções desenvolvidas e apresentadas nesta tese investigam esta hipótese. Cada solução investiga essa hipótese sob um determinado ponto de vista.

Os focos de investigação ficaram concentrados na predição Bidirecional e na predição *Affine*, ambas ferramentas com elevado custo computacional. A predição *Affine*, que será mais bem discutida no restante deste texto, é uma das principais novidades do VVC.

Ao final do trabalho, foi possível demonstrar que a hipótese apresentada acima é válida para as ferramentas investigadas neste trabalho.

1.2 Objetivos e Contribuições

Este trabalho visa desenvolver soluções de otimização algorítmica focadas em ferramentas da predição interquadros do padrão de codificação VVC, a fim de obter a redução no tempo de codificação sem impactos significativos na eficiência de codificação.

De maneira mais específica, esta proposta tem os seguintes objetivos:

1. Realizar uma análise experimental sobre o tempo de codificação e do uso dos tamanhos de bloco em diferentes ferramentas da predição interquadros do VVC.
2. Realizar uma revisão sistemática da literatura para capturar o estado-da-arte da pesquisa nesta temática.
3. Desenvolver uma heurística com uso de análise estatística focada na redução do custo computacional das etapas Unidirecional, Bidirecional e *Affine* do VVC.
4. Desenvolver modelos de aprendizado de máquina para reduzir o custo computacional da predição Bidirecional do VVC.
5. Desenvolver modelos de aprendizado de máquina para reduzir o custo computacional da predição *Affine* do VVC.
6. Desenvolver modelos de aprendizado de máquina para reduzir o custo computacional, de forma conjunta, das predições Bidirecional e *Affine* do VVC.
7. Analisar os resultados da implementação da heurística e modelos de aprendizado de máquina no *software* VTM (*VVC Test Model*) em relação a redução no tempo de codificação e perdas na eficiência de codificação.

1.3 Apresentação do Texto

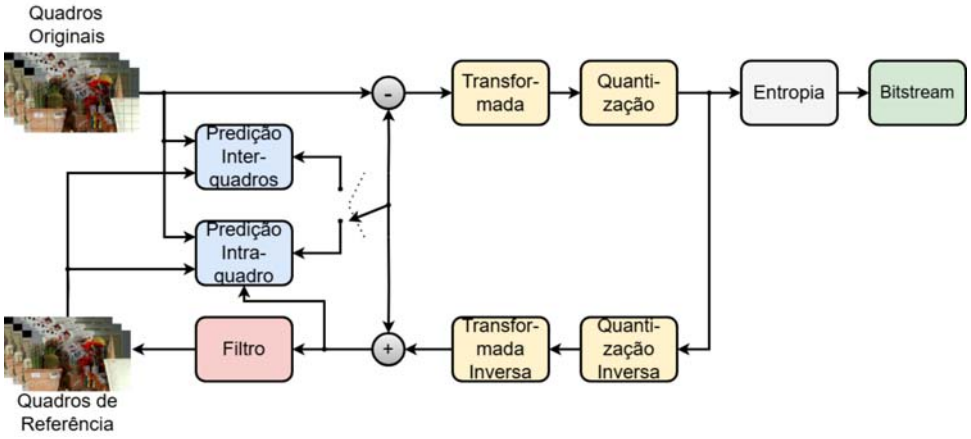
O texto desta tese está organizado da seguinte maneira: no Capítulo 2 são abordados os conceitos sobre o padrão VVC; o Capítulo 3 aprofunda a descrição dos principais conceitos relacionados à etapa da predição interquadros, acompanhada da

apresentação das respectivas novas ferramentas disponíveis no padrão VVC; já o Capítulo 4 engloba os principais conceitos, métodos e técnicas relacionadas ao aprendizado de máquina; no Capítulo 5 é apresentada uma análise experimental sobre a predição interquadros do padrão VVC e a avaliação das suas principais ferramentas em relação ao tempo de codificação e utilização dos tamanhos de bloco; já o Capítulo 6 aborda uma revisão sistemática da literatura, incluindo a descrição e análise de trabalhos relacionados com foco no HEVC e VVC; por fim, os Capítulos 7, 8, 9 e 10 apresentam as soluções para a redução do tempo de codificação, usando análise estatística e aprendizado de máquina, seguidos pela conclusão no Capítulo 11.

2 CONCEITOS DE CODIFICAÇÃO DE VÍDEO E O PADRÃO VVC

O *Versatile Video Coding* (VVC) é o padrão mais recente definido pela ITU-T e ISO/IEC e foi disponibilizado em julho de 2020. De acordo com Bross et al. (2021), além da maior eficiência de codificação, o VVC fornece e aperfeiçoa funcionalidades voltadas a uma série de aplicações emergentes, tais como: vídeos de alta definição; conteúdo de tela ou gerado por computador; transmissão *ultra-low-delay*; transmissões adaptativas; vídeos 360°; codificação multicamadas; entre outras. De maneira geral, o VVC contempla as etapas de um codificador de vídeo híbrido, baseado em resíduos de predições, contendo: predição interquadros e intraquadro, Transformadas, Quantização, codificação de Entropia e Filtros de laço. As Figuras 1 e 2 ilustram a estrutura básica do codificador e do decodificador do VVC, respectivamente.

Figura 1 – Estrutura Básica do Codificador VVC



Fonte: Elaborada pelo autor.

Conforme ilustrado na Figura 1, cada imagem do vídeo é dividida em blocos, que seguem um esquema de particionamento, sendo que estes são utilizados em todas as etapas da codificação.

A predição intraquadro é responsável pela redução da redundância espacial (Browne; Ye; Kim, 2022), que ocorre quando píxeis vizinhos em um mesmo quadro

possuem valores semelhantes. No VVC, os modos direcionais, que no HEVC eram 33, são expandidos para 65, sendo que o novo padrão também continua contando com os modos Planar e DC (média) (Browne; Ye; Kim, 2022). Além disso, novas ferramentas foram introduzidas nessa etapa, tais como Múltiplas Linhas de Referência (*Multiple Reference Line* - MRL) (Chang et al., 2019), Sub-Partições Intra (*Intra Sub-Partitions* - ISP) (De-luxán-hernández et al., 2019), Previsão Intra Ponderada por Matriz (*Matrix-based Intra Prediction* - MIP) (Schäfer et al., 2019), entre outras.

A predição interquadros trata da redundância temporal (PALAU et al., 2021), que ocorre entre quadros temporalmente vizinhos. Essa etapa da predição será descrita com mais detalhes no próximo capítulo, por ser o foco desta tese.

Após essas etapas, ocorre a subtração entre o bloco predito, resultante da predição intraquadro ou interquadros, e o bloco original. Os resíduos dessa subtração são tratados pelas demais etapas. Na etapa das Transformadas, esses valores são transformados do domínio espacial para o domínio das frequências, preparando-os para a próxima etapa. O VVC incluiu novas ferramentas para a etapa das Transformadas, tais como: transformada para blocos não quadrados; seleção múltipla de transformada (*Multiple Transform Selection* - MTS), incluindo as Transformadas DCT-VIII e DST-VII, além da DCT-II presente no HEVC. Já na Quantização, ocorre a eliminação das frequências menos relevantes ao sistema visual humano (SVH), sendo também a etapa que determina a taxa de compressão e as perdas de qualidade em um codificador de vídeos (Agostini, 2007).

Após esse processo, os valores resultantes são comprimidos na codificação de Entropia, através do uso de algoritmos de compressão sem perdas. O objetivo dessa etapa é representar a maior quantidade de dados com a menor quantidade de bits possível, reduzindo a redundância entrópica (Agostini, 2007). Sendo assim, a saída final do processo de codificação é chamada de *bitstream* e, nos padrões de codificação, segue um fluxo e sintaxe específicos. No VVC, o método usado para a codificação de Entropia é o CABAC. Esse método já era utilizado no H.264 e no HEVC, porém foram incorporadas algumas melhorias, tais como a codificação de coeficiente aprimorada e a estimação de probabilidade de múltiplas hipóteses de alta precisão (BROSS et al., 2021).

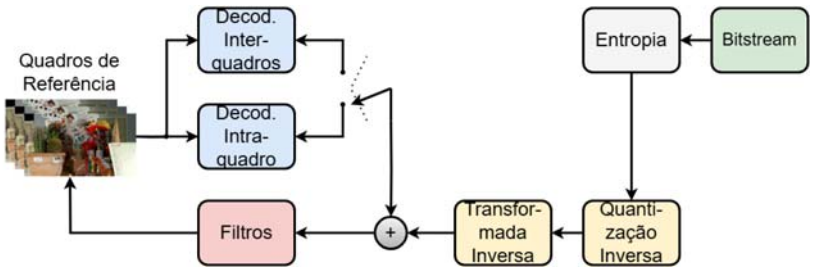
Como é possível visualizar na Figura 1, há um caminho de decodificação, inserido após a etapa de Quantização. Esse processo de decodificação, que inclui as etapas inversas da Quantização e das Transformadas e dos Filtros de Laço, ainda durante a codificação, é necessário pelo fato de que a Quantização gera perdas de informações. Portanto, os blocos que já foram codificados são reconstruídos considerando as predições realizadas pelas etapas Intra ou interquadros, formando assim os quadros de referência. Então, estes quadros reconstruídos são utilizados como base para a codificação de outros quadros originais do vídeo. Isso garante que tanto o codificador

quanto o decodificador irão usar exatamente os mesmos quadros de referência, o que evita a degradação da qualidade na compressão (PALAU et al., 2021).

Os Filtros de Laço ou *In-loop Filters* servem para melhorar a qualidade subjetiva dos vídeos, sendo necessários pois o processo de codificação insere alguns artefatos nas imagens, tais como: efeito de bloco (*blocking*), granulação (*ringing*) e desfoque (*blurring*) (PALAU et al., 2021). O VVC implementa três filtros, sendo eles: filtro de deblocagem (*Deblocking Filter* - DBF); compensação de amostra adaptável (*Sample Adaptive Offset* - SAO) e filtro de laço adaptável (*Adaptive Loop Filter* - ALF). O filtro ALF é composto por luma ALF, chroma ALF e componentes cruzados (*Cross-component ALF* - CC-ALF) (Browne; Ye; Kim, 2022).

O processo de decodificação dos quadros de um vídeo é ilustrado na Figura 2. Basicamente, a partir do *bitstream*, são realizadas as etapas de Entropia, Quantização e Transformada inversas, sendo estas somadas às decisões já tomadas pelas etapas de predição Intra ou interquadros. É importante destacar que, mesmo sendo um processo mais simples, um decodificador deve estar apto a interpretar qualquer decisão tomada pelo codificador. Este, por sua vez, pode ser implementado com alguma simplificação para fins de redução de custo computacional, por exemplo, desde que gere um *bitstream* que seja possível de ser decodificado (PALAU et al., 2021).

Figura 2 – Estrutura Básica do Decodificador VVC

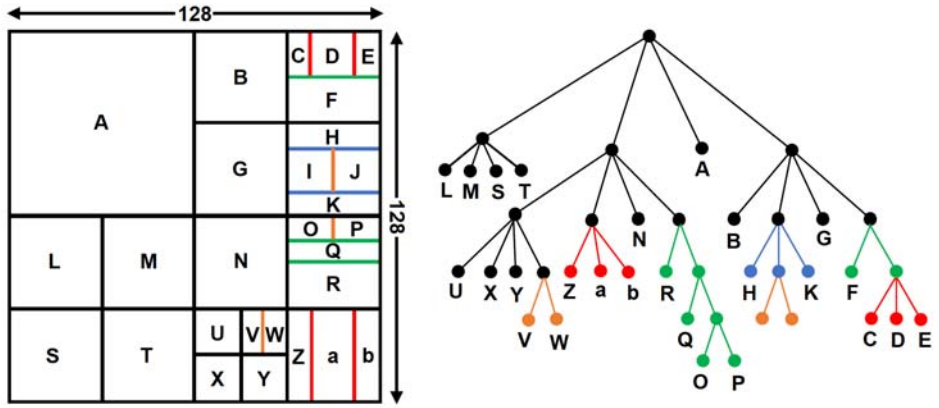


Fonte: Elaborada pelo autor.

No VVC, um novo esquema de particionamento, derivado do seu antecessor o HEVC, aplica divisões quadráticas, binárias ou ternárias, seguindo um particionamento de Árvore Quaternária com Árvore de Tipos Múltiplos Aninhada (*Quad-tree with nested Multi-type Tree* - QTMT) (Browne; Ye; Kim, 2022). Mais especificamente, cada quadro é dividido em Unidades de Árvore de Codificação (*Coding Tree Units* - CTU) que são blocos quadrados de tamanhos iguais, cujo tamanho máximo é 128×128 píxeis. Cada CTU é composta por três Blocos de Árvore de Codificação (*Coding Tree Blocks* - CTB), um para cada componente de cor, ou seja, um para luminância (Y), um para crominância azul (Cb) e outro para crominância vermelha (Cr) (Browne; Ye; Kim, 2022). Posteriormente, cada CTU é recursivamente dividida em Unidades de Codificação (*Coding Units* - CU), com seus três componentes de cor. A divisão recursiva segue um esquema de árvore QTMT, conforme ilustrado na Figura 3. Primeiramente, são ob-

tidas as CUs quadradas, em uma árvore quaternária (*quad-tree* - QT). Após isso, cada um dos nós folha da QT é associado a uma Árvore de Tipos Múltiplos (*multi-type tree* - MTT), podendo sofrer divisões binárias horizontais ou verticais, divisões ternárias horizontais ou verticais, ou ainda, não sofrer divisão. A partir do momento em que o particionamento MT é utilizado nas CUs, as mesmas não poderão utilizar mais subdivisões quadráticas. As CUs podem ter o tamanho máximo de 128×128 amostras e mínimo de 4×4 amostras. Os componentes de cor em uma CU são chamados de Blocos de Codificação (*Coding Blocks* - CB). Em se tratando de amostras de luminância, o tamanho mínimo dos CBs pode ser de 4×4. Já para as amostras de crominância, o tamanho de CB pode variar de 64×64 até 4×4 amostras (BROSS et al., 2021).

Figura 3 – Esquema de particionamento QTMT



Fonte: Elaborada pelo autor.

Segundo (Browne; Ye; Kim, 2022), todas as etapas do processo de codificação podem ser executadas em cada CB candidato. Dessa forma, a codificação torna-se um processo extremamente complexo, visto que ocorre em um número expressivo de possibilidades, considerando os 27 tamanhos de CBs possíveis com a árvore QTMT, além das várias ferramentas e modos disponíveis no VVC. Esse processo é chamado de otimização do custo taxa-distorção (*Rate-distortion Optimization* - RDO), que busca o menor custo taxa-distorção (*Rate-distortion cost* - RD cost) (Sullivan; Wiegand, 1998) para codificar cada CTB do quadro. Esse custo considera a relação entre a distorção (erro) do bloco predito em relação ao bloco original e o número de bits necessários para sua representação. O *software* de referência do VVC implementa diferentes estratégias, chamadas heurísticas, na tentativa de reduzir o esforço computacional desse processo, diminuindo as possibilidades testadas. Mesmo assim, o VVC apresenta um custo computacional muito elevado e seu *software* de referência tem um tempo de codificação de cinco a 31 vezes maior em comparação com o software de referência do HEVC, dependendo da configuração (Pakdaman et al., 2020).

No software de referência do VVC estão disponíveis três configurações que são definidas antes da codificação dos vídeos, são elas: *All Intra*, *Low Delay* e *Random*

Access (Bossen et al., 2020). Na configuração *All Intra* todos os quadros do vídeo são codificados somente pela predição intraquadro, não sendo executada a predição interquadros. Já nas configurações *Low Delay* e *Random Access*, ambas as predições estão disponíveis. Essas configurações organizam conjuntos de quadros (*Group of Pictures* - GOP), em que o primeiro quadro sempre é codificado exclusivamente com a predição intraquadro, e as CBs dos demais quadros podem ser preditos com a interquadros ou a intraquadro. A configuração *Random Access* permite que a ordem de codificação dos quadros seja fora da ordem de captura. Assim, essa configuração permite o uso da predição Bidirecional, onde CBs de dois quadros podem ser usados como referência simultaneamente para gerar a predição da CB atual. A configuração *Low Delay* codifica os quadros na ordem de captura e tem duas variações: *Low Delay P* e *Low Delay B* (Bossen et al., 2020). A configuração *Low Delay P* permite apenas a predição Unidirecional, assim, uma CB dos quadros de referência pode ser usada para gerar a predição do bloco atual. Já a configuração *Low Delay B* permite a predição Bidirecional, porém considerando somente quadros temporalmente passados. A configuração *Random Access* tem um custo computacional e uma latência maior, mas atinge uma eficiência de codificação mais elevada, sendo mais útil para aplicações como *streaming*. Por outro lado, a configuração *Low Delay* tem um custo computacional e uma latência menores e também uma menor eficiência de codificação, mas é útil para codificar vídeos ao vivo, onde a latência é um ponto crítico.

A área de codificação de vídeos utiliza as métricas Bjøntegaard Delta Bit Rate (BDBR) e Bjøntegaard Delta PSNR (BDPSNR) para avaliar a eficiência de codificação (Bjøntegaard, 2001). A métrica BDPSNR é usada para comparar dois codificadores onde, para uma mesma quantidade de bits usada para codificar um vídeo, é medido o aumento ou a redução da sua qualidade objetiva (em PSNR) após a codificação. Os resultados são expressos em decibéis. Então, números positivos em BDPSNR indicam um aumento na qualidade objetiva para uma mesma taxa de bits, indicando um ganho na eficiência de codificação. A métrica BDBR é complementar à métrica BDPSNR e também é usada para comparar dois codificadores. Neste caso, para uma mesma qualidade objetiva (em PSNR), é medido o percentual de aumento ou redução do número de bits necessários para representar um vídeo. Assim, números negativos em BDBR indicam redução na taxa de bits para uma mesma qualidade objetiva, indicando um ganho na eficiência. Essa é a métrica mais usada na área de codificação de vídeos e será usada neste trabalho.

3 PREDIÇÃO INTERQUADROS DO PADRÃO VVC

A etapa de predição interquadros do VVC e as suas ferramentas serão descritas com mais detalhes a seguir. Basicamente, a redução da redundância temporal é o objetivo da predição interquadros e esse objetivo é alcançado através de dois estágios principais: Estimação de Movimento (*Motion Estimation* - ME) e Compensação de Movimento (*Motion Compensation* - MC).

Para cada CB de luminância em um quadro, a ME procura o bloco com melhor correspondência em quadros codificados previamente. Quando este bloco é encontrado, pelo menos um vetor de movimento (*motion vector* - MV) é gerado para indicar a localização do bloco. Nos codificadores atuais, a ME tem dois passos principais: a Estimação de Movimento Inteira (*Integer Motion Estimation* - IME) e a Estimação de Movimento Fracionária (*Fractional Motion Estimation* - FME) (Palau et al., 2021). Primeiramente, a ME é aplicada sobre amostras inteiras dos quadros de referência. Quando a melhor correspondência é encontrada, uma interpolação é feita para gerar amostras fracionárias e a FME é aplicada como uma segunda etapa da ME sobre amostras fracionárias. A FME é capaz de capturar movimentos sutis, aumentando a eficiência de codificação. Se a FME é usada, o MV pode assumir valores fracionários (Agostini, 2007). O último estágio é a MC que utiliza os MVs para localizar os blocos referenciados e copiá-los, reconstruindo a imagem. Nesse caso, os MVs gerados para o CB de luminância são reaproveitados também para os CBs de crominância e, caso alguma subamostragem de cor (Agostini, 2007) tenha sido utilizada, o fracionamento respectivo é aplicado a estes vetores (Browne; Ye; Kim, 2022). A MC é necessária porque a correspondência gerada pela ME não é perfeita e, então, a diferença entre o bloco original e o bloco apontado pelo MV não pode ser descartada. Essa diferença é chamada de resíduo e é processada pelas outras etapas de codificação e enviada no *bitstream* (Agostini, 2007).

O modo ME básico é chamado de ME Unidirecional, no qual a predição só pode usar blocos em um quadro de referência como a melhor correspondência. Nesses caso, 27 tamanhos de bloco são permitidos (Browne; Ye; Kim, 2022), ficando de fora apenas o tamanho 4×4. A predição ME Bidirecional, por outro lado, pode apontar

para dois blocos em dois quadros de referência diferentes. Uma predição ponderada entre esses dois blocos é realizada para gerar a predição final (BROSS et al., 2021). No caso da ME Bidirecional no VTM, são suportados 25 tamanhos de bloco, ficando de fora tamanhos 4×4, 4×8 e 8×4 (Browne; Ye; Kim, 2022).

Como em outros padrões, o VVC também permite a codificação de quadros fora de ordem. Sendo assim, os quadros de referência usados na predição interquadros podem ser quadros passados ou futuros na ordem de apresentação, mas todos devem ser previamente codificados (BROSS et al., 2021). Especificamente, na configuração *Random Access* do *software* VTM (*VVC Test Model*), os quadros temporalmente passados e futuros ao quadro que está sendo codificado são armazenados em dois *buffers*, chamados Lista 0 (L0) e Lista 1 (L1), respectivamente.

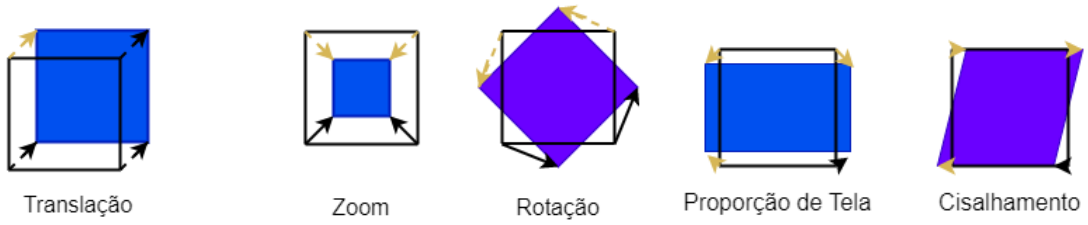
O VVC inclui outras ferramentas novas na predição interquadros, tais como: Estimação de Movimento *Affine*, Modos *Merge* Estendidos (*Extended Merge Modes*), Modo de Particionamento Geométrico (*Geometric Partitioning Mode* - GPM), Predição Inter/intraquadro Combinada (*Combined Inter/Intra-picture Prediction* - CIIP), Resolução de Vetor de Movimento Adaptável (*Adaptive Motion Vector Resolution* - AMVR), Bi-predição com Pesos a nível de Unidade de Codificação (*Bi-prediction with CU-level Weights* - BCW), Fluxo Óptico Bidirecional (*Bi-directional Optical Flow* - BDOF), e Refinamento do Vetor de Movimento no Decodificador (*Decoder-side MV Refinement* - DMVR). Essas ferramentas serão discutidas nas próximas seções do texto.

3.1 Estimação de Movimento *Affine*

Conforme mencionado anteriormente, uma das principais novidades do VVC é a Estimação de Movimento *Affine*. Esse modo é usado para mapear movimentos não translacionais, tais como dimensionamento, rotação e cisalhamento, melhorando a eficiência de codificação. A Figura 4 representa os diferentes tipos de movimentos e seus respectivos Pontos de Controle (*Control Points* - CP), representados pelas setas amarelas. É possível notar que, quanto mais complexo o movimento, mais Pontos de Controle serão necessários para mapeá-lo.

Figura 4 – Tipos de movimentos e seus respectivos Pontos de Controle

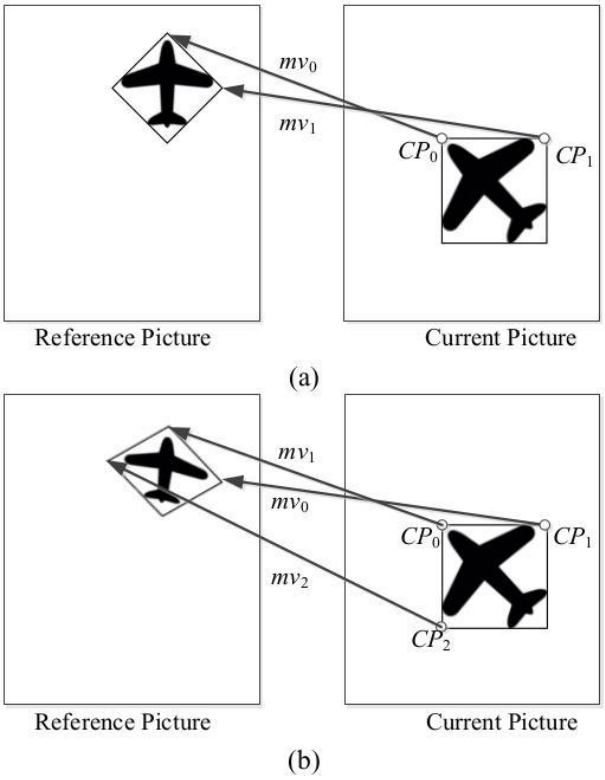
(a) Um Ponto de Controle (b) Dois Pontos de Controle (c) Três Pontos de Controle



Fonte: Gonçalves (2021).

O VVC define duas etapas *Affine*, conforme mostrado na Figura 5: o modelo 4-parâmetros e o modelo 6-parâmetros. O primeiro é usado para modelar movimentos mais simples, tais como dimensionamento e rotação, enquanto o segundo é usado para movimentos mais complexos, tais como cisalhamento. Dessa forma, são necessários dois ou três Vetores de Movimento, conforme os Pontos de Controle (*Control Points* - CP) do bloco (Browne; Ye; Kim, 2022).

Figura 5 – Modelos *Affine* 4-parâmetros (a) e 6-parâmetros (b).



Fonte: Zhang; Mao (2019).

A seguir, são definidas as equações para a derivação dos Vetores de Movimento, considerando a localização por amostra (x, y) (Browne; Ye; Kim, 2022). Para o modelo *Affine* de quatro parâmetros, considera-se a equação (1). Já para o modelo *Affine* de seis parâmetros, considera-se a equação (2).

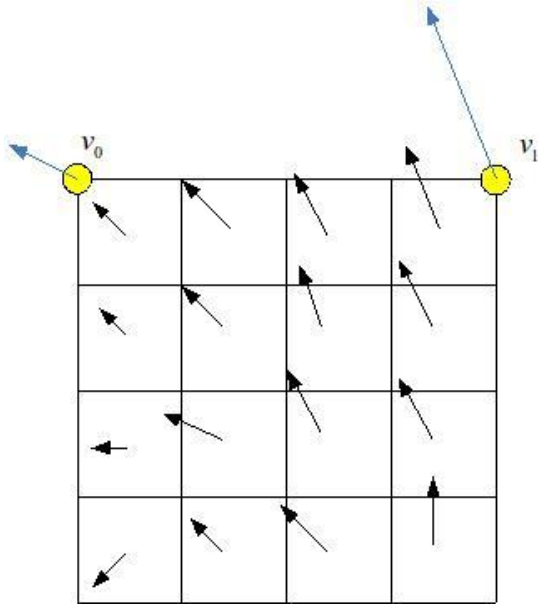
$$\begin{cases} mv_x = \frac{mv_{1x}-mv_{0x}}{W}x + \frac{mv_{0y}-mv_{1y}}{W}y + mv_{0x} \\ mv_y = \frac{mv_{1y}-mv_{0y}}{W}x + \frac{mv_{1x}-mv_{0x}}{W}y + mv_{0y} \end{cases} \tag{1}$$

$$\begin{cases} mv_x = \frac{mv_{1x}-mv_{0x}}{W}x + \frac{mv_{2x}-mv_{0x}}{H}y + mv_{0x} \\ mv_y = \frac{mv_{1y}-mv_{0y}}{W}x + \frac{mv_{2y}-mv_{0y}}{H}y + mv_{0y} \end{cases} \tag{2}$$

Onde (mv_{0x}, mv_{0y}) é o Vetor de Movimento do CP superior esquerdo (CP_0), (mv_{1x}, mv_{1y}) do CP superior direito (CP_1), (mv_{2x}, mv_{2y}) do CP inferior esquerdo (CP_2), H e W se referem à altura e largura do bloco, respectivamente (Browne; Ye; Kim, 2022).

Entretanto, por simplificação, a predição *Affine* não é baseada em amostras, mas em sub-blocos de luminância 4×4 (Browne; Ye; Kim, 2022). Conforme ilustra a Figura 6, o MV da amostra central de cada sub-bloco é calculado de acordo com as equações anteriores e arredondado para a precisão 1/16. Então, na compensação de movimento, os filtros de interpolação são aplicados para gerar a predição de cada sub-bloco com base no MV derivado. Os sub-blocos de crominância também têm o tamanho 4×4, sendo que o seu MV é calculado a partir da média dos MVs dos sub-blocos de luminância superior esquerdo e inferior direito, na região co-localizada 8×8 de luminância.

Figura 6 – Predição *Affine* por sub-blocos



Fonte: Browne; Ye; Kim (2022).

O modo *Affine* do VVC pode ser aplicado sobre 12 tamanhos de blocos (Browne; Ye; Kim, 2022) que devem ter, pelo menos, 16 píxeis de altura e largura, ou seja, do bloco 16×16 até o bloco 128×128. Assim, a ME do VVC, além de possuir as etapas Unidirecional e Bidirecional, como em outros codificadores recentes, também possui a ME para os dois modelos *Affine*, 4 e 6 parâmetros. Sendo assim, se o bloco atual é de um dos 12 tamanhos suportados, a *Affine* 4-parâmetros é executada após a etapa Bidirecional. Entretanto, no VTM, o modelo *Affine* de 6-parâmetros somente é executado se o custo do modelo de 4-parâmetros for até 5% maior que o menor custo obtido pelas etapas Unidirecional e Bidirecional. Independente do modelo *Affine* executado, durante a Estimação de Movimento *Affine* (*Affine Motion Estimation* - AME), são executadas as etapas Unidirecional e Bidirecional específicas, tanto para o modelo de 4-parâmetros quanto para o de 6-parâmetros. Naturalmente, a complexidade final da ME, que já era grande em codificadores anteriores, cresce muito com estas novas opções trazidas pelo VVC.

Quanto à precisão das amostras para estimar o movimento fracionário, é utilizado o AMVR (*Adaptive Motion Vector Resolution*), que adapta a precisão das amostras de luminância dependendo do modo. Para a *Affine*, a precisão das amostras pode ser de pixel inteiro ou entre 1/4 e 1/16 de pixel.

É importante ressaltar que há um modo *Merge* específico para o *Affine*. Nesse caso, podem ser usadas CUs com largura e altura iguais ou maiores que oito (Browne; Ye; Kim, 2022). Ao utilizar o modo *Affine Merge*, os Vetores de Movimento dos Pontos de Controle (*Control Point Motion Vector* - CPMV) são gerados com base nas informações de movimento das CUs vizinhas espaciais. Podem existir até cinco candidatos, sendo sinalizado o índice daquele que será usado para gerar o CPMV.

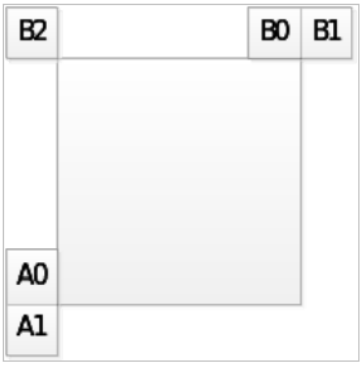
3.2 *Extended Merge prediction*

Esse algoritmo gera um único conjunto de parâmetros de movimento para uma região inteira de blocos contíguos compensados por movimento. Desde o padrão HEVC o modo *Merge* é aplicado na predição interquadros, tendo o objetivo de identificar blocos com características semelhantes de movimento, gerando regiões de blocos contíguos, explorando a redundância nos parâmetros de movimento que devem ser transmitidos (Helle et al., 2012). No VVC a lista de candidatos ao *Merge* é constituída a partir de cinco tipos de candidatos, conforme segue: Predição de Vetor de Movimento Espacial com Unidades de Codificação vizinhas espaciais (*Spatial MVP from spatial neighbor CUs*); Predição de Vetor de Movimento Temporal para Unidades de Codificação Co-localizadas (*Temporal MVP from collocated CUs*); Predição de Vetor de Movimento baseada em histórico em tabela FIFO (*History-based MVP from a FIFO table*); Predição de Vetor de Movimento por média de pares (*Pairwise average MVP*) e *Zero MVs*.

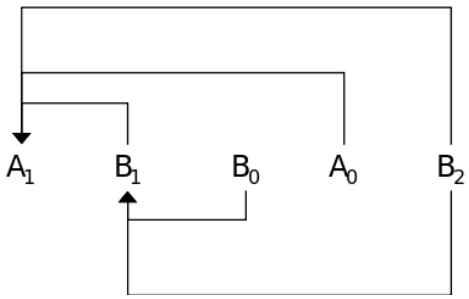
De acordo com (Browne; Ye; Kim, 2022), a definição dos blocos espacialmente candidatos ao *Merge* (*Spatial MVP from spatial neighbour CUs*) é semelhante ao processo realizado no HEVC, sendo trocadas as posições dos dois primeiros candidatos. Conforme mostra a Figura 7a, podem ser selecionados, no máximo, quatro candidatos localizados e derivados na seguinte ordem: B0, A0, B1, A1 e B2. A posição B2 é considerada somente se alguma das demais não estiver disponível ou é codificada como intra. A fim de melhorar a eficiência de codificação, é realizada uma verificação de redundância de informação de movimento, após a inclusão do candidato A1. Essa verificação é feita considerando alguns pares de candidatos, sendo que, para ser adicionado à lista, o bloco deve ter informações de movimento diferentes dos blocos correspondentes (Figura 7b).

Um candidato ao *Merge* temporal (*Temporal MVP from collocated CUs*) é adicionado à lista, a partir da derivação de um vetor de movimento em escala com base na

Figura 7 – Derivação dos candidatos espaciais ao *Merge*



(a) Posições dos candidatos espaciais à mesclagem



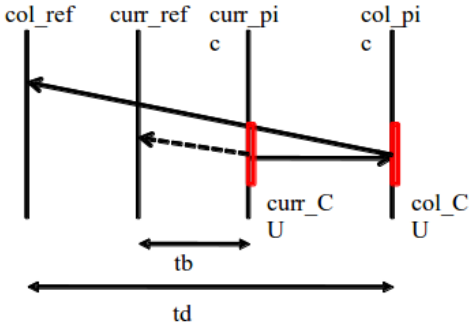
(b) Pares de candidatos

Fonte: Browne; Ye; Kim (2022).

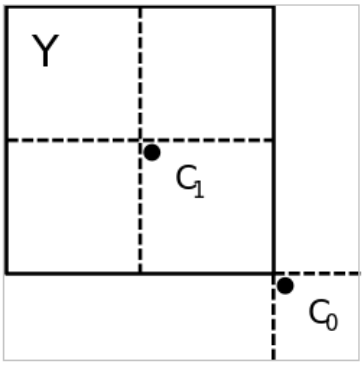
CU co-localizada pertencente à imagem de referência co-localizada. Conforme ilustrado na Figura 8a, a linha pontilhada indica o vetor de movimento dimensionado para o candidato à mesclagem temporal. Ele é obtido usando a distância tb, que é a diferença do POC (*Picture Order Count*) entre a imagem de referência e a imagem atual, e a distância td, que é a diferença POC entre a imagem de referência co-localizada e a imagem co-localizada. A partir das duas posições, C_0 e C_1 (Figura 8b), o candidato ao *Merge* temporal é derivado, sendo que se C_0 não está disponível, é codificado como intra ou está fora da linha atual de CTUs, a posição C_1 é utilizada (Browne; Ye; Kim, 2022).

Figura 8 – Definição dos candidatos temporais ao *Merge*

(a) Ilustração de dimensionamento de Vetor de Movimento



(b) Posições dos candidatos C_0 e C_1



Fonte: Browne; Ye; Kim (2022).

Na derivação de candidatos à mesclagem como base no histórico (*History-based merge candidates derivation* - HMVP) é gerada uma tabela com informações de movimento de blocos previamente codificados no modo inter, sendo então utilizados como MVP para o bloco atual. Essa tabela é mantida durante a codificação e decodificação,

sendo reiniciada quando uma nova linha de CTU é encontrada. É possível armazenar até cinco candidatos na tabela que segue a restrição de funcionamento FIFO (*First In First Out*). Antes de um novo elemento ser inserido na tabela, é realizada uma verificação de redundância e, caso seja encontrado um candidato idêntico, o mesmo é retirado e todos os candidatos posteriores a ele são movidos adiante. Os candidatos HMVP podem ser usados na lista de candidatos do modo *Merge*, sendo adicionados após os candidatos ao *Merge* temporal. Antes disso, eles são verificados quanto à redundância: os dois últimos elementos são comparados com os candidatos espaciais A1 e B1, respectivamente; caso o total de candidatos ao *Merge* chegue ao máximo permitido menos um, o processo para HMVP é encerrado (Browne; Ye; Kim, 2022).

Segundo (Browne; Ye; Kim, 2022), em outro caso é realizada a média de pares de candidatos ao *Merge* pré-definidos como $p0Cand$ e $p1Cand$, usando os dois primeiros candidatos da lista respectivamente (*Pair-wise average merge candidate derivation*). Os cálculos dos MVs médios são realizados de acordo com a disponibilidade do MV de $p0Cand$ e $p1Cand$ para cada lista de referência separadamente. Caso ambos os MVs dos candidatos estejam disponíveis para uma lista, é calculada a média entre eles, sendo definida a imagem de referência como a mesma do $p0Cand$. Porém, se apenas um MV estiver disponível, o mesmo será usado diretamente. Quando nenhum MV estiver disponível, a lista é invalidada. Além disso, caso os índices do filtro de interpolação *half-pel* forem diferentes para ambos os candidatos, ele será definido como zero. Por fim, após os candidatos de mesclagem por média de pares serem adicionados, caso a lista geral de candidatos ao *Merge* não esteja cheia, a mesma é preenchida com MVPs (*Motion Vector Prediction*) zerados.

3.3 *Adaptive Motion Vector Resolution* (AMVR)

A ferramenta de Resolução Adaptativa de Vetor de Movimento (*Adaptive Motion Vector Resolution* - AMVR) possibilita que o MVD (*Motion Vector Difference*) da CU seja codificado em diferentes precisões (Browne; Ye; Kim, 2022). A seleção da precisão depende do modo AMVP (*Advanced MV Prediction*) para a CU atual. No HEVC, o AMVP faz a sinalização explícita de um dos dois potenciais candidatos a MVP, que são derivados de cinco MVs espacialmente vizinhos e dois temporalmente colocalizados (MVs usados ao codificar um local correspondente em uma determinada imagem previamente decodificada). Também precisa da indicação da aplicação da ME Unidirecional ou Bidirecional e, para cada MV, é atribuída a indicação de qual imagem de referência será usada (BROSS et al., 2021). As diferenças de MV são calculadas entre o vetor de movimento predito (pelo AMVP) e o vetor de movimento (gerado pelas buscas Unidirecional ou Bidirecional). As precisões de amostras de luminância disponíveis para o modo AMVP normal são: 1/4, 1/2, amostra inteira ou 4 amostras. Já para

o modo AMVP *Affine*, a precisão dos CPMVs podem ser: 1/4, amostra inteira ou 1/16. Se todos os componentes MVD (horizontais e verticais para a Lista 0 e Lista 1) forem zero, é sinalizada a precisão 1/4 de amostra de luminância. Os demais casos são condicionados a pelo menos um componente do MVD ser diferente de zero (Browne; Ye; Kim, 2022).

3.4 *Symmetric MVD Coding* (SMVD)

Já a Codificação Simétrica de Diferença de Vetor de Movimento (*Symmetric MVD coding* - SMVD) explora as situações em que o movimento do bloco atual está em uma trajetória constante, considerando uma imagem de referência passada e outra futura. Nesses casos, tanto os MVs quanto os índices das imagens de referência tendem a ser simétricos (BROSS et al., 2021). Dessa forma, na predição Bidirecional usando SMVD, o codificador sinaliza apenas o MVD da Lista 0, sendo que o MVD da Lista 1 e os índices das imagens de referência são derivados. O MVD da Lista 1 é definido como o inverso do MVD da Lista 0 (- MVD0) e os índices de referência para Lista 0 e Lista 1 são definidos, respectivamente, como sendo iguais ao par de imagens de referência (Browne; Ye; Kim, 2022). No codificador, a Estimação de Movimento MVD simétrico começa com a avaliação inicial do MV. Um conjunto de candidatos iniciais, tais como os MVs obtidos das pesquisas das predições Unidirecional e Bidirecional, além dos MVs da lista AMVP. Aquele com o menor custo de taxa-distorção é escolhido para ser o MV inicial para a busca de movimento MVD simétrico (Browne; Ye; Kim, 2022).

3.5 *Bi-Prediction with CU-Level Weights* (BCW)

A ferramenta Bi-predição com Pesos a nível de Unidade de Codificação (*Bi-prediction with CU-level Weights* - BCW) é uma extensão da predição Bidirecional, que fornece o cálculo de uma média ponderada dos dois sinais de predição, além da média simples já existente (Browne; Ye; Kim, 2022). São definidos cinco pesos fixos iguais a $\{-2, 3, 4, 5, 10\}/8$ e um índice deste conjunto é sinalizado para especificar o peso w selecionado para a predição de bloco da lista 1. Já para a lista 0, o peso é definido como $1 - w$. Todos os cinco pesos podem ser usados na configuração *low-delay*, cujas imagens de referência são todas temporalmente anteriores. Nos demais casos, apenas o subconjunto $\{3, 4, 5\}/8$ pode ser considerado (BROSS et al., 2021). O BCW é aplicado somente em CUs que tenham 256 amostras de luminância ou mais. Em uma mesma CU no VVC não pode-se aplicar o modo BCW e o modo Combinação da Predição inter e interquadros (*Combined Inter/Intra-picture Prediction* - CIIP) em conjunto. O modo CIIP será explicado mais adiante neste texto. Quando o modo CIIP é utilizado, o índice BCW é setado como 2, indicando pesos iguais.

3.6 *Bidirectional Optical Flow* (BDOF)

O Fluxo Óptico Bidirecional (*Bidirectional Optical Flow* - BDOF) é uma técnica que visa refinar a predição Bidirecional no nível de subblocos 4×4. Esta técnica é baseada no conceito de fluxo óptico, assumindo que o movimento de um objeto na CU atual é suave ou homogêneo (BROSS et al., 2021). O BDOF pode ser aplicado em CUs codificadas no modo *Merge* ou AMVP, porém não é habilitado quando os modos *Affine*, *Merge* SbTMVP (*Subblock-based Temporal Motion Vector Prediction*) ou CIIP são utilizados. Além disso, o BDOF também é desabilitado quando o índice BCW indica pesos desiguais ou a predição ponderada (*Weighted Prediction* - WP) é usada em qualquer imagem de referência da imagem atual. A ferramenta BDOF é aplicada em CUs com altura e largura iguais ou maiores que 8 amostras de luminância, ou seja, os blocos precisam ter pelo menos 64 amostras de luminância. Para cada subbloco 4×4 é calculado um refinamento de movimento que é usado para ajustar os valores da respectiva amostra predita na etapa Bidirecional (Browne; Ye; Kim, 2022).

3.7 *Decoder Side MV Refinement* (DMVR)

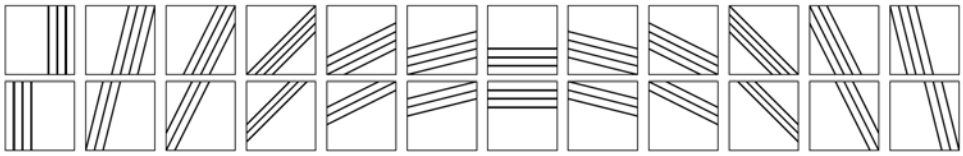
Já o Refinamento de Vetor de Movimento no Decodificador (*Decoder side MV Refinement* - DMVR) é outra técnica que realiza o refinamento do MV na etapa da predição Bidirecional no decodificador do VVC, a fim de aumentar a precisão do MV no modo *Merge*. A DMVR é executada com uma busca em aproximadamente duas amostras de luminância em torno dos MV iniciais da Lista 0 e Lista 1, caso estes sejam inteiros, sendo calculado o SAD (*Sum of Absolute Differences*) (Browne; Ye; Kim, 2022). No caso dos MV iniciais serem fracionários, utiliza-se um filtro de interpolação bilinear 2-tap para gerar as amostras e o filtro de interpolação normal 8-tap é aplicado para gerar a predição final. A DMVR é aplicada somente em CUs com mais de 64 amostras de luminância, sendo que, quando sua altura ou largura for maior que 16, a CU é dividida e a técnica é aplicada em blocos 16×16 separadamente (BROSS et al., 2021).

3.8 *Geometric Partitioning Mode* (GPM)

O Modo de Particionamento Geométrico (*Geometric Partitioning Mode* - GPM) é uma variação do Modo *Merge* que possibilita a compensação de movimento em partições não retangulares dos blocos. Suporta 64 esquemas de partições diferentes para cada tamanho de CU a partir de 8×8 até 64×64, exceto 8×64 e 64×8 (Browne; Ye; Kim, 2022). Quando o GPM é usado, uma CU é dividida em duas partes por uma linha reta geometricamente localizada, conforme ilustrado na Figura 9. Essa localização é derivada matematicamente a partir dos parâmetros de ângulo e deslocamento de uma partição específica. Somente a predição Unidirecional é permitida para cada

partição, sendo que cada uma possui um Vetor de Movimento e um índice de imagem de referência. Após a predição das partes, as amostras são combinadas usando um processamento de mesclagem com pesos adaptativos ao longo da linha de partição geométrica (BROSS et al., 2021).

Figura 9 – Exemplos de divisões do GPM agrupadas por ângulos



Fonte: Browne; Ye; Kim (2022).

3.9 *Combined Inter/Intra-Picture Prediction* (CIIP)

A Combinação da Predição Intra e Interquadros (*Combined Inter/Intra-picture Prediction* - CIIP), como o próprio nome representa, é uma ferramenta que combina as predições interquadros e intraquadro (Browne; Ye; Kim, 2022). Essa técnica usa os resultados obtidos pelo modo *Merge* regular - da predição inter - e do modo Planar - da predição intra - efetuando uma média ponderada. O peso usado no cálculo da CIIP é definido com base nos modos de codificação dos blocos vizinhos acima e à esquerda do bloco atual, da seguinte maneira: se esses dois blocos forem intra, o peso será três; se apenas um desses blocos for intra, o peso será dois e se nenhum deles for intra, o peso será um (Equação (3)).

$$P_{CIIP} = ((4 - wt) * P_{inter} + wt * P_{intra} + 2) \gg 2 \tag{3}$$

A ferramenta CIIP é habilitada para blocos com pelo menos 64 amostras de luminância e que tenham altura e largura menores que 128, sendo que uma *flag* adicional é sinalizada indicando a utilização da CIIP.

4 CONCEITOS BÁSICOS DE APRENDIZADO DE MÁQUINA

Considerando que esta tese apresenta modelos de aprendizado de máquina para reduzir o custo computacional da predição interquadros do VVC, este capítulo discute alguns conceitos necessários para compreender as soluções desenvolvidas.

Inicialmente, é importante esclarecer o conceito de Inteligência Artificial (IA), que segundo Norvig; Russell (2014) "(...) é o estudo de agentes que recebem percepções do ambiente e executam ações". Nesse sentido, a área de Aprendizado de Máquina pode ser considerada um ramo da IA, em que esses agentes "(...) podem melhorar seu comportamento através do estudo diligente de suas próprias experiências"(Norvig; Russell, 2014). Outra definição está presente em (Raschka, 2015), onde afirma que a área de Aprendizado de Máquina engloba o desenvolvimento de algoritmos capazes de obter conhecimento e fazer previsões com base em dados de entrada.

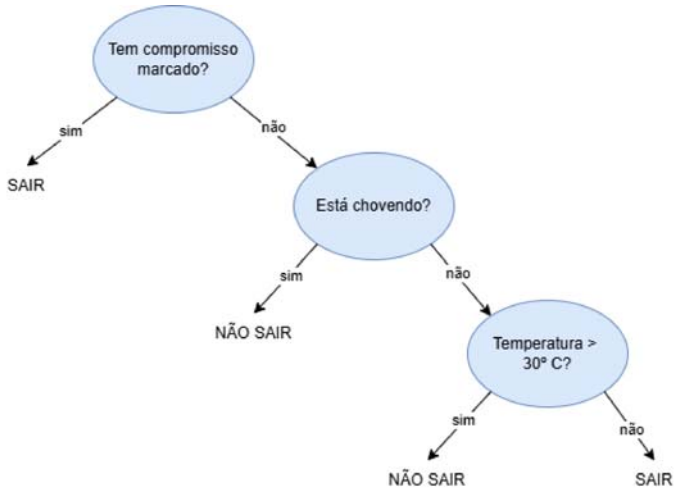
Raschka (2015) também define os três principais tipos de Aprendizado de Máquina: supervisionado, não supervisionado e por reforço. O aprendizado supervisionado consiste em treinar um modelo com base em dados já rotulados, ou seja, cuja saída já é conhecida, permitindo que este modelo faça previsões sobre novos dados. Já o aprendizado não supervisionado precisa utilizar técnicas específicas, como agrupamento, por exemplo, para extrair informações significativas a partir de dados não rotulados. Por fim, o aprendizado por reforço engloba o desenvolvimento de sistemas (agentes) que melhoram gradativamente o seu desempenho a partir das interações com o ambiente, sendo essas interações as medidas do quão bem as ações foram realizadas. Nesta tese, serão considerados algoritmos de aprendizado supervisionado, visto que os dados são extraídos durante o processo de codificação juntamente com o rótulo determinado.

Um subconjunto importante do aprendizado supervisionado engloba os problemas de classificação. Nesses casos, os modelos precisam aprender, a partir de dados previamente rotulados em classes categóricas, e predizer qual é a classe de novos exemplos (Raschka, 2015). Esses problemas podem ser de "classificação binária", quando há apenas duas classes possíveis, ou "classificação multi-classe", quando há várias classes. No contexto da etapa interquadros da codificação de vídeo, as solu-

ções envolvendo aprendizado supervisionado podem ser tanto de classificação binária quanto multi-classe. Por exemplo, se a solução pretende definir qual das etapas da interquadros deve ser executada para determinado bloco, será multi-classe, visto que envolverá as possibilidades Unidirecional, Bidirecional ou *Affine*. Por outro lado, um exemplo de classificação binária, que está inserido nesta tese, é quando a solução define se a etapa Bidirecional deve ou não ser executada para determinado bloco.

Existem diversas técnicas possíveis de serem utilizadas para o treinamento dos modelos de aprendizado de máquina. Nesta tese, serão utilizadas as técnicas Árvore de Decisão (*Decision Tree*) e Florestas Aleatórias (*Random Forests*), visto que são modelos de baixo custo computacional e, assim, acrescentam um custo computacional mínimo ao processo de codificação. Segundo Norvig; Russell (2014), uma Árvore de Decisão é composta por nós internos, representando os testes realizados a partir de valores dos atributos, e nós folha, que representam as classes/decisões. A Figura 10 apresenta um exemplo simplificado de uma árvore sobre a decisão de "sair de casa", contendo três testes a partir de atributos (*features*), com as possíveis classes "sair"ou "não sair".

Figura 10 – Exemplo de Árvore de Decisão



Fonte: Elaborada pelo autor.

Conforme (Raschka, 2015), para definir os testes em cada nó e realizar as divisões para os próximos nós, o algoritmo de Árvore de Decisão usa o conceito de ganho de informação, do inglês *information gain* - IG. O ganho de informação pode ser considerado como a diferença entre a impureza/incerteza do conjunto de dados disponível em um nó pai e a soma das impurezas/incertezas dos conjuntos de dados dos nós filhos. A impureza ou incerteza se refere ao quanto o conjunto de dados está dividido entre as classes em um determinado nó. Quanto mais equilibradas as proporções entre as classes, maior a incerteza na decisão. Por exemplo, se um conjunto de dados disponível em um nó da árvore está dividido com 50% dos registros para a classe 0 e 50% para a classe 1, significa que a incerteza da decisão é alta. Já no momento

em que a proporção entre as classes está desbalanceada, menor é a incerteza na decisão. Ou seja, se o conjunto de dados disponível em um nó da árvore está dividido em 10% dos registros para a classe 0 e 90% para a classe 1, significa que a incerteza da decisão é baixa, já que é muito provável que a mesma seja pela classe 1. Portanto, quanto menor a incerteza, maior é o ganho de informação. Dessa forma, em cada nó da árvore, o objetivo é escolher a *feature* que garante o maior ganho de informação possível.

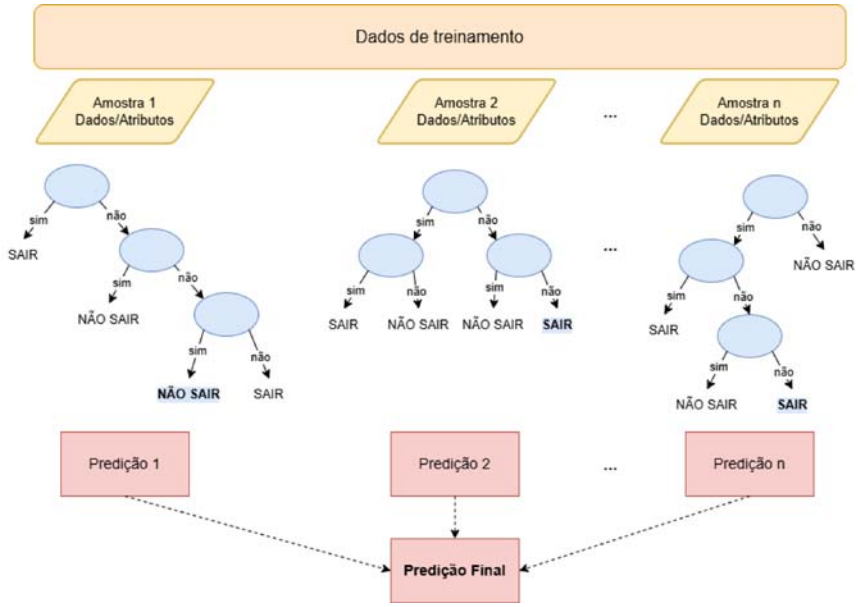
Nesta tese, são utilizados dois critérios disponíveis para a definição do ganho de informação: "*entropy*"e "*gini*". Basicamente, o critério *entropy* ou entropia, mede o grau de desordem de um conjunto de dados. Ou seja, a entropia é máxima (igual a 1) quando a proporção das classes é uniforme e mínima (igual a 0) quando uma das classes é majoritária. O critério *gini* é semelhante, medindo a impureza de um conjunto de dados, ou seja, a probabilidade de erro na classificação. Sendo assim, um valor do índice gini próximo a 0 indica maior pureza do conjunto de dados, já quando é próximo a 1, indica maior impureza. Resumidamente, quanto menor a impureza, independente do critério utilizado, maior será o ganho de informação (Raschka, 2015).

Uma das preocupações ao se utilizar aprendizado de máquina é a possibilidade de sobreajuste (ou *overfitting* em inglês) do modelo aos dados de treinamento (Raschka, 2015). Esse problema ocorre quando o modelo gerado tem pouca capacidade de generalização, ou seja, se especializou nos dados de treinamento e tem baixa performance para dados novos. No caso das árvores de decisão, a definição de uma profundidade máxima para as divisões da árvore é uma das estratégias que podem ser utilizadas para evitar o sobreajuste (Raschka, 2015). Outra questão importante é o balanceamento das classes no conjunto de dados usado para treinamento, ou seja, que o número de registros das classes seja equilibrado. Esse balanceamento visa garantir que o modelo não priorize o aprendizado sobre uma classe em detrimento da outra, garantindo também uma capacidade de generalização (Raschka, 2015).

Outro algoritmo de aprendizado de máquina é o de Florestas Aleatórias, que são conjuntos de Árvores de Decisão, e estão inseridas nos modelos de aprendizado por *ensembles*. Esse tipo de modelo tem o objetivo de combinar modelos mais simples para gerar modelos mais robustos, genéricos e com menor possibilidade de sobreajuste (Raschka, 2015). Basicamente, diferentes árvores de decisão são definidas aleatoriamente, a partir de amostras dos dados e atributos que garantem o maior ganho de informação a cada divisão. Ao final, as decisões de cada árvore são agregadas e a classe que obtiver a maioria dos resultados será a classe final retornada pelo modelo. A Figura 11 ilustra um exemplo simplificado de Floresta Aleatória, conforme a descrição anterior.

De maneira geral, o processo de aprendizado de máquina supervisionado envolve algumas etapas importantes, tais como pré-processamento dos dados, treinamento

Figura 11 – Exemplo de Floresta Aleatória



Fonte: Elaborada pelo autor.

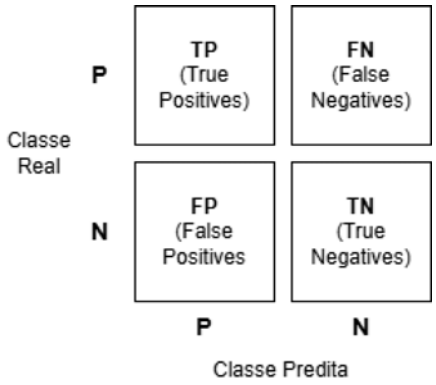
dos modelos, avaliação e, por fim, a implementação do modelo para novos dados (Raschka, 2015). Durante o pré-processamento dos dados ocorre a extração das *features*, ou seja, os atributos de cada amostra de dados devidamente rotulados. Nessa fase também pode existir a necessidade de adaptar ou corrigir os dados, realizando processos como a exclusão de valores nulos ou incorretos e a criação ou transformação de atributos. Além disso, é importante que os registros das classes sejam balanceados e o conjunto de dados (*dataset*) seja dividido em duas partes, uma para o treinamento e outra para os testes.

No processo de treinamento e avaliação dos modelos, são utilizadas métricas para análise da performance dos modelos, técnicas de validação, além da busca pelos melhores hiperparâmetros. Por fim, a implementação consiste na aplicação e real utilização do modelo treinado sobre novos dados.

Conforme mencionado anteriormente, a utilização de métricas para avaliação da performance dos modelos é uma parte fundamental do processo de aprendizado de máquina. Uma vez treinado, com base no conjunto de treinamento, o modelo poderá ser testado, ou seja, vai predizer as classes para o conjunto de teste. Dessa forma, é possível analisar o desempenho do modelo através da matriz de confusão (Raschka, 2015), por exemplo. Nesse caso, são geradas as pontuações para cada classe, nomeadas como Positiva (P) e Negativa (N), através de uma matriz quadrada. Conforme ilustra a Figura 12, a matriz de confusão mostra, na diagonal principal, a quantidade de predições corretas (*True*) para ambas as classes, ou seja, a quantidade de Verdadeiros Positivos (*True Positives* - TP) e Verdadeiros Negativos (*True Negatives* - TN). Já na diagonal secundária, são mostrados os erros das predições (*False*), ou seja, a quantidade de Falsos Positivos (*False Positives* - FP) e Falsos Negativos (*False Ne-*

gatives - FN). Naturalmente, valores mais altos na diagonal principal e mínimos na diagonal secundária demonstram um bom desempenho do modelo.

Figura 12 – Exemplo de Matriz de Confusão



Fonte: Elaborada pelo autor.

A partir da matriz de confusão e seus respectivos valores, é possível calcular outras métricas relacionadas ao desempenho do modelo (Raschka, 2015). A acurácia (ACC), conforme demonstrado na equação (4), consiste na taxa de acertos gerais do modelo, sendo a soma das predições corretas dividida pelo total das predições.

$$ACC = \frac{TP + TN}{FP + FN + TP + TN} \tag{4}$$

Já a Precisão (PRE) é a relação entre os acertos e o total das predições para uma classe, conforme exemplo na equação (5).

$$PRE = \frac{TP}{TP + FP} \tag{5}$$

O *Recall* (REC), descrito pela equação (6), pode ser considerado como a taxa de acertos em relação aos valores reais de uma classe.

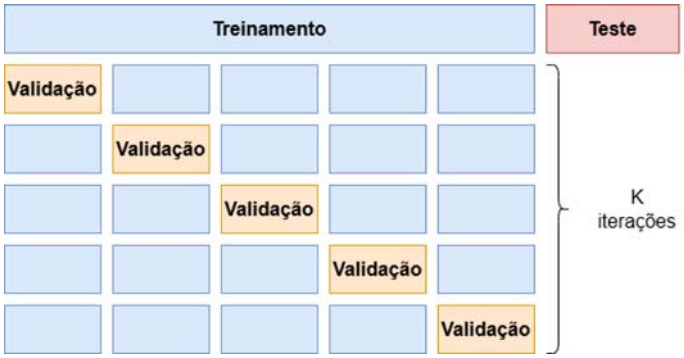
$$REC = \frac{TP}{FN + TP} \tag{6}$$

Por fim, o *F1-score* engloba tanto a Precisão quanto o *Recall*, calculando a média harmônica entre os mesmos, conforme mostrado na equação (7).

$$F1 = 2\frac{PRE \times REC}{PRE + REC} \tag{7}$$

Além da divisão do *dataset* entre conjunto de treinamento e teste, também pode ser efetuada a validação do modelo, através da técnica de validação cruzada ou K-fold (Boisberranger et al., 2025a). Nessa técnica, o conjunto de treinamento é dividido em K partes, em que uma dessas partes é utilizada como validação e as demais para o treinamento. A partir disso, ocorre um processo iterativo de treinamento e teste, sendo gerada uma performance geral de K modelos, conforme ilustrado na Figura 13.

Figura 13 – Exemplo de utilização do método K-fold



Fonte: Elaborada pelo autor.

Outra ação importante ao longo do processo de treinamento é a busca pelos melhores hiperparâmetros para o modelo. Cada tipo de algoritmo terá os seus hiperparâmetros específicos, ou seja, configurações próprias que vão influenciar o modelo construído na etapa de treinamento. O algoritmo de Árvore de Decisão, disponível na biblioteca `scikit-learn` (PEDREGOSA et al., 2011), possui hiperparâmetros como, por exemplo, o critério para realizar as divisões (*criterion*), a profundidade máxima da árvore (*max_depth*), o número mínimo de amostras para um nó ser dividido (*min_samples_split*), o número mínimo de amostras de um nó folha (*min_samples_leaf*), dentre outros.

A fim de automatizar os testes e a busca pelos hiperparâmetros, são utilizados nesta tese os métodos de busca *Random Search* (Boisberranger et al., 2025b) e *Grid Search* (Boisberranger et al., 2025c) combinados, conforme a metodologia descrita no trabalho de Andrades; Grellert; Fonseca (2019). Primeiramente, para o método *Random Search* ser utilizado, é necessário definir faixas de valores para os hiperparâmetros que serão foco da busca. Também é definida a quantidade de iterações que serão realizadas, ou seja, quantas combinações diferentes, de valores selecionados aleatoriamente dentro das faixas estipuladas, serão testadas. Além disso, em cada iteração, é utilizada a técnica de validação cruzada (K-fold). Ao final, o método retorna a combinação de valores com a melhor performance, de acordo com a métrica escolhida. Nesta tese, foram estipuladas 500 repetições, utilizando validação cruzada igual a 5, conforme indicado em Andrades; Grellert; Fonseca (2019). Já a métrica escolhida para definir o melhor resultado foi a F1-score, pois representa a média entre o *recall* e a precisão, sendo mais significativa para a tomada de decisão. Além disso, foram calculados os coeficientes de correlação de Pearson para cada hiperparâmetro testado em relação à F1-score, conforme indicado no trabalho (Andrades; Grellert; Fonseca, 2019). Basicamente, o coeficiente de Pearson (Schober; Boer; Schwarte, 2018) é a medida da correlação entre duas variáveis, podendo ser positivo ou negativo. No contexto deste e de outros trabalhos ((Andrades; Grellert; Fonseca, 2019), (Duarte, 2021)), a correlação positiva indica que valores mais altos de determinado

hiperparâmetro tendem a ser associados a valores maiores de F1-score. Já a correlação negativa indica que valores mais baixos para determinado hiperparâmetro tendem a aumentar o F1-score.

Já o método *Grid Search* consiste em realizar a busca exaustiva utilizando todas as combinações possíveis a partir das faixas de valores de cada hiperparâmetro. Nesta tese, foi definido que os dois hiperparâmetros com maior valor de correlação no *Random Search* teriam faixas de valores mais amplas no *Grid Search*, mesma estratégia usada no trabalho (Andrades; Grellert; Fonseca, 2019). Sendo assim, para os demais hiperparâmetros foi utilizado o valor padrão do modelo e, se for o caso, o valor retornado pelo melhor resultado no *Random Search*.

Deste modo, neste capítulo foram apresentados os principais conceitos de aprendizado de máquina e as metodologias relacionadas às soluções desenvolvidas nesta tese. Esses conhecimentos teóricos e técnicos fornecem o suporte necessário para a construção, treinamento e avaliação dos modelos que serão apresentados adiante.

5 ANÁLISE EXPERIMENTAL DA PREDIÇÃO INTERQUADROS DO VVC

Neste capítulo, serão apresentados e discutidos os resultados obtidos através de experimentos de análise da codificação no padrão VVC. O experimento foi realizado em um servidor com processador Intel Xeon E5-2650v4 2,20GHz com 48 GB de RAM. Foram utilizadas todas as sequências de vídeo das Condições Comuns de Teste (*Common Test Conditions* - CTCs) (Bossen et al., 2020), conforme consta na Tabela 1.

Tabela 1 – Sequências de vídeos utilizadas no experimento

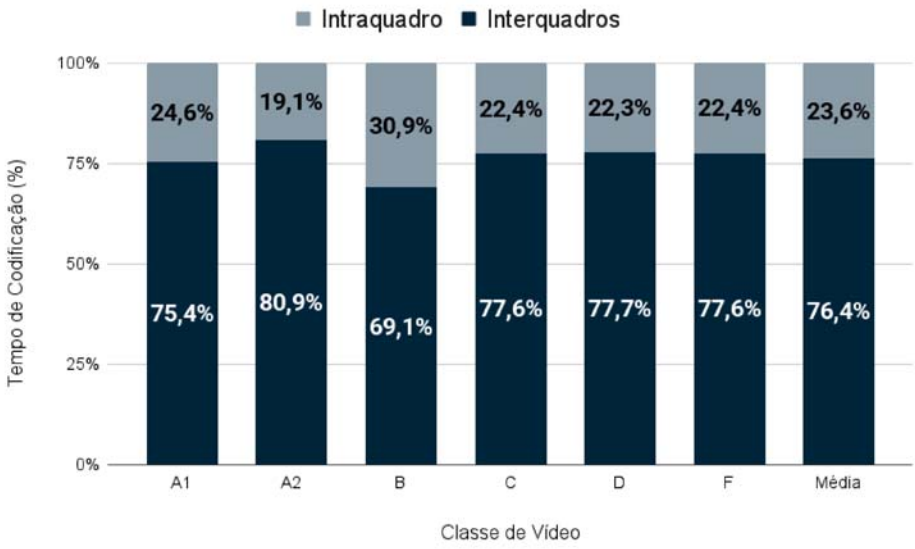
Nome	Classe	Resolução	Nº Quadros	Taxa (fps)
Campfire	A1	3840x2160	300	30
FoodMarket4			300	60
Tango2			294	60
CatRobot	A2	3840x2160	600	60
DaylightRoad2			600	60
ParkRunning3			600	50
BasketballDrive	B	1920x1080	500	50
BQTerrace			600	60
Cactus			500	50
MarketPlace			600	60
RitualDance			600	60
BasketballDrill	C	832x480	500	50
BQMall			600	60
PartyScene			500	50
RaceHorsesC			300	30
BasketballPass	D	416x240	500	50
BlowingBubbles			500	50
BQSquare			600	60
RaceHorses			300	30
ArenaOfValor	F	1920x1080	600	60
BasketballDrillText		832x480	500	50
SlideEditing		1280x720	300	30
SlideShow		1280x720	500	20

Esses vídeos foram codificados por inteiro, ou seja, com todos os quadros, para os quatro Parâmetros de Quantização (*Quantization Parameters* - QPs) 22, 27, 32 e 37, na configuração *Random Access*. Essa configuração foi definida por ser a mais usada em cenários reais e, também, por ser a mais completa e, assim, com maior custo computacional. A versão do *software* de referência utilizada foi a VTM 16.2 e o código foi alterado em alguns trechos para a captura dos dados de interesse.

5.1 Avaliação das Principais Etapas da Predição

Nesta seção é apresentada uma avaliação inicial, visando melhorar a compreensão das principais etapas de predição do VVC: predição interquadros e intraquadro. São aprofundadas as análises do tempo de codificação e do número de CBs avaliados para ambas as etapas de predição. Especificamente, divide-se o tempo total do codificador entre predição interquadros e intraquadro, sendo que os resultados apresentados levam em consideração todas as etapas de codificação necessárias para lidar com um bloco predito. Isso abrange não apenas a predição em si, mas também processos subsequentes, como Transformadas, Quantização, codificação de Entropia e Filtros.

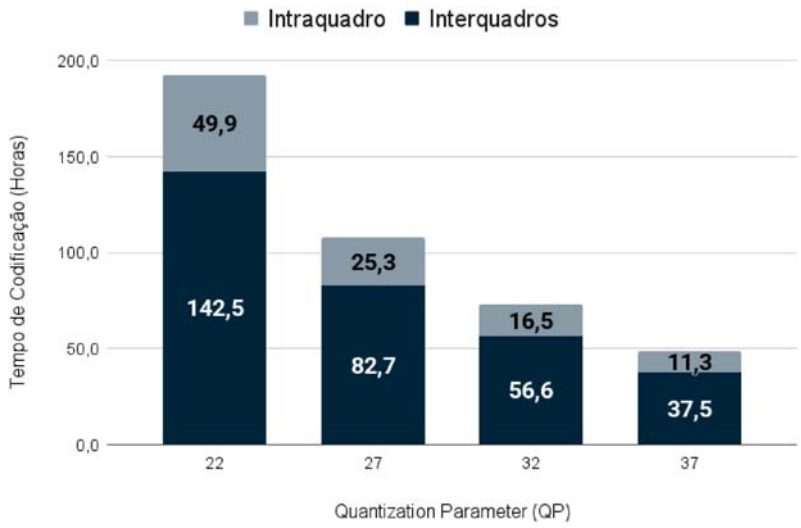
Figura 14 – Percentual do tempo de codificação das etapas interquadros e intraquadro (por Classe de Vídeo).



Fonte: Elaborada pelo autor.

A Figura 14 ilustra a porcentagem de tempo de codificação para cada classe de vídeo. É evidente que a etapa de predição interquadros domina o tempo de codificação em todas as classes de vídeo, ultrapassando a marca de 70% em cinco das seis classes. A média geral indica que a predição interquadros consome 76,4% do tempo total de codificação. Por outro lado, o estágio intraquadro varia de 19,1% (classe A2) a 30,9% (classe B), com média geral de 23,6% do tempo total de codificação. A

Figura 15 – Tempo de Codificação (em horas) das etapas interquadros e intraquadro (por QP).



Fonte: Elaborada pelo autor.

diferença mais significativa é observada na classe A2, onde a predição interquadros requer 4,2 vezes mais tempo do que a predição intraquadro. Em contraste, a menor diferença ocorre na classe B, onde a predição interquadros requer 2,2 vezes mais tempo de codificação do que a predição intraquadro.

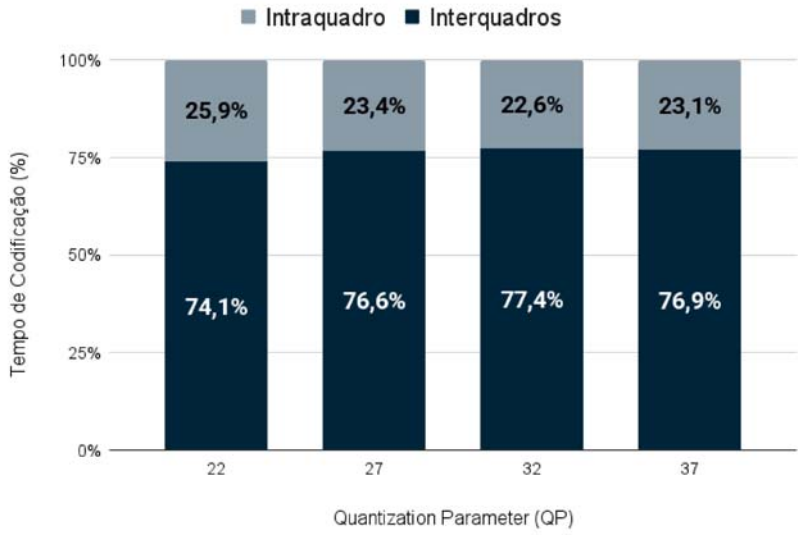
A Figura 15 apresenta uma análise secundária do tempo de codificação, mas agora agrupando os resultados com base no valor de QP. Neste cenário, o tempo médio de codificação para todas as sequências é levado em consideração para cada QP. Uma observação importante aqui é a diminuição substancial no tempo de codificação à medida que o valor de QP aumenta. A codificação com QP 22 leva quase quatro vezes mais tempo do que a codificação com QP 37.

A Figura 16 ilustra as mesmas informações da Figura 15, mas agora em termos de valores percentuais, que são mais significativos para fins de avaliação. As avaliações posteriores apresentadas neste trabalho utilizarão resultados percentuais. Neste caso, torna-se evidente que, à medida que o valor de QP aumenta, o percentual de tempo gasto na predição interquadros também tende a aumentar, embora com uma ligeira variação, como observado na Figura 16.

Uma segunda análise considerou o número de CBs avaliados tanto na predição interquadros quanto na intraquadro. Os resultados exibem uma correlação natural com as descobertas anteriores (ver Figura 15 e Figura 16), já que um maior número de CBs avaliados geralmente corresponde a um aumento no tempo de codificação. No entanto, considerando a complexidade distinta das ferramentas de codificação para os dois tipos de predição, esta análise tem um significado particular.

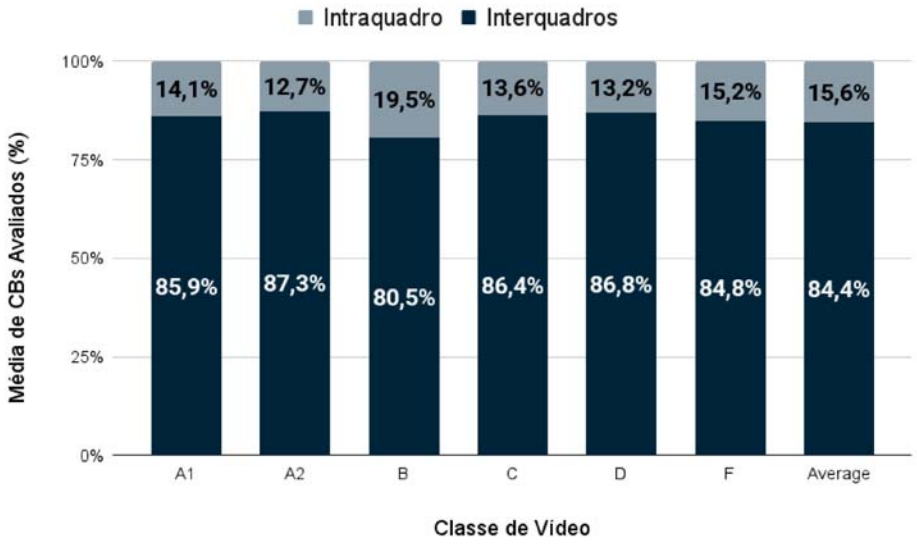
A Figura 17 ilustra os resultados percentuais categorizados por Classe de Vídeo. Em média, a predição interquadros envolve a avaliação de 5,4 vezes mais CBs do que a predição intraquadro. Nota-se que as maiores e menores disparidades foram

Figura 16 – Percentual do tempo de codificação das etapas interquadros e intraquadro (por QP).



Fonte: Elaborada pelo autor.

Figura 17 – Percentual de CBs avaliados nas predições interquadros e intraquadro (por Classe de Vídeo).



Fonte: Elaborada pelo autor.

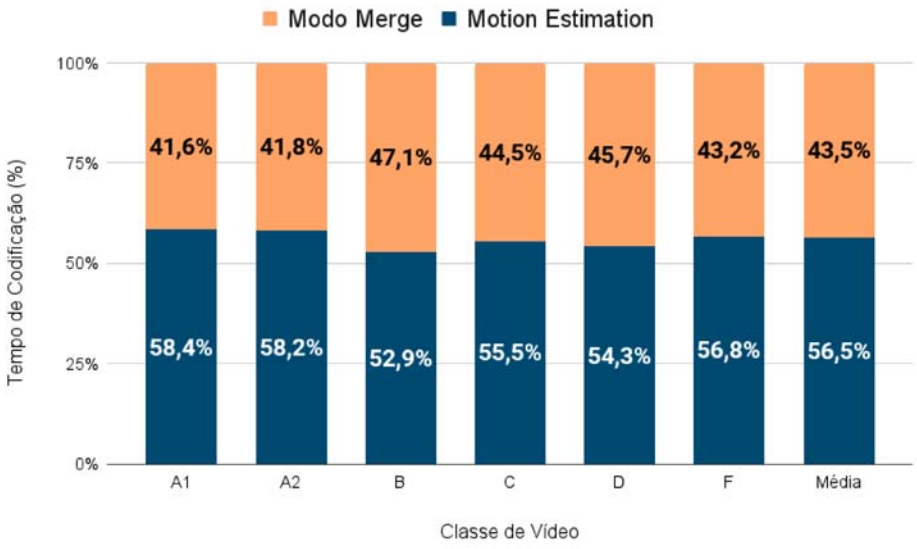
novamente observadas nas classes A2 e B, respectivamente, refletindo as tendências na Figura 14. Contudo, neste caso, as discrepâncias são mais distintas: 6,9 vezes para a classe A2 e 4,1 vezes para a classe B. Com base nisso, é possível concluir que o esforço computacional necessário para avaliar cada CB é menor na predição interquadros do que na intraquadro, uma vez que a diferença no tempo de codificação é menor que a diferença no número de CBs avaliados. Então, mesmo com um tempo total de codificação muito maior, a predição interquadros requer um esforço computacional menor para prever cada CB processado do que a predição intraquadro.

Outro experimento considerou o percentual de CBs avaliados na predição interquadros e intraquadro, agrupados por QP. Neste caso, o valor do QP não impacta significativamente a proporção de CBs avaliados em ambas as ferramentas de codificação. Em todos os casos, o percentual de CBs avaliados na intraquadro foi próximo de 15,6% e na interquadros foi de cerca de 84,4%. Uma comparação com a Figura 16 também revela que, para todos os QPs, o percentual de CBs avaliados é superior ao percentual de tempo gasto na codificação destes CBs, conforme ilustrado na Figura 16. Isto reforça a conclusão anterior de que as ferramentas de predição interquadros requerem um esforço computacional menor por CB processado do que a predição intraquadro. O número de CBs avaliados na predição interquadros é mais de cinco vezes maior do que aqueles avaliados na predição intraquadro. Por outro lado, o tempo gasto na predição interquadros é apenas cerca de três vezes maior do que aquele gasto na predição intraquadro.

5.2 Avaliação do Tempo de Codificação por Classe de Vídeo

Nesta seção, será analisado o tempo de codificação para modos específicos da predição interquadros, categorizados por classe de vídeo. Esses modos incluem a Estimação de Movimento (*Motion Estimation* - ME), que compreende a ME convencional e a ME *Affine*, e o *Merge*, abrangendo funções que mesclam blocos que exibem características de movimento semelhantes.

Figura 18 – Percentual do tempo de codificação dos principais modos da predição interquadros (por Classe de Vídeo).



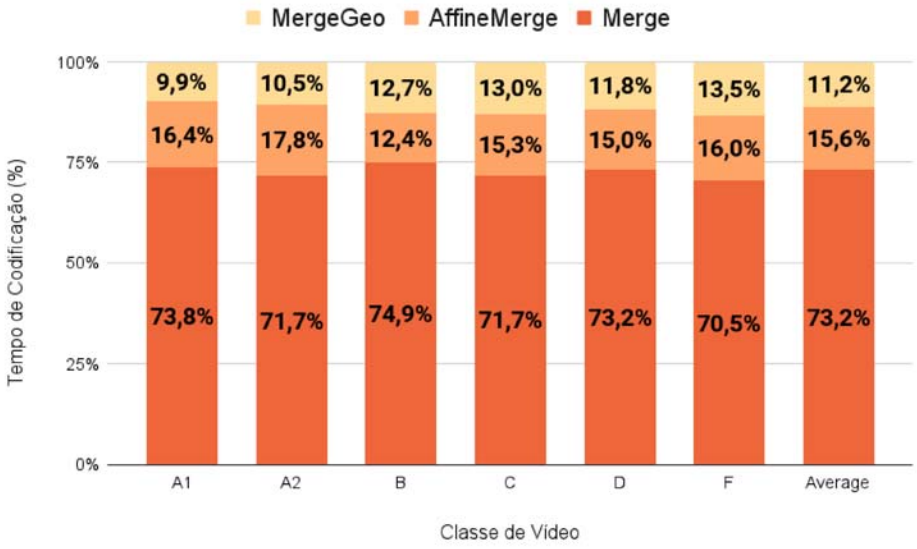
Fonte: Elaborada pelo autor.

A Figura 18 ilustra o tempo médio de codificação para esses dois principais modos de codificação na predição interquadros, ME e *Merge*, em diferentes Classes de Vídeos. No geral, a ME é responsável pela maior parte do tempo da predição inter-

quadros, representando aproximadamente 57% do tempo total. Os valores percentuais apresentam apenas uma ligeira variação entre as diferentes classes de vídeo. Na verdade, é possível notar uma pequena tendência de diminuição do tempo gasto na ME à medida que a resolução do vídeo diminui.

Uma análise detalhada, considerando a distribuição do tempo de codificação para o modo *Merge*, é apresentada na Figura 19. Dentro da predição interquadros do VVC, existem três funções *Merge*. A função *Merge* representa a aplicação padrão do modo *Merge*, mesclando blocos com características de movimento semelhantes. A função *AffineMerge* é um modo *Merge* especializado, projetado para blocos com movimentos não translacionais. Por último, a função *MergeGeo* aplica particionamento geométrico, outra forma do modo *Merge*, dividindo blocos em partições angulares (Browne; Ye; Kim, 2022).

Figura 19 – Percentual do tempo de codificação das funções do Modo *Merge* (por Classe de Vídeo).



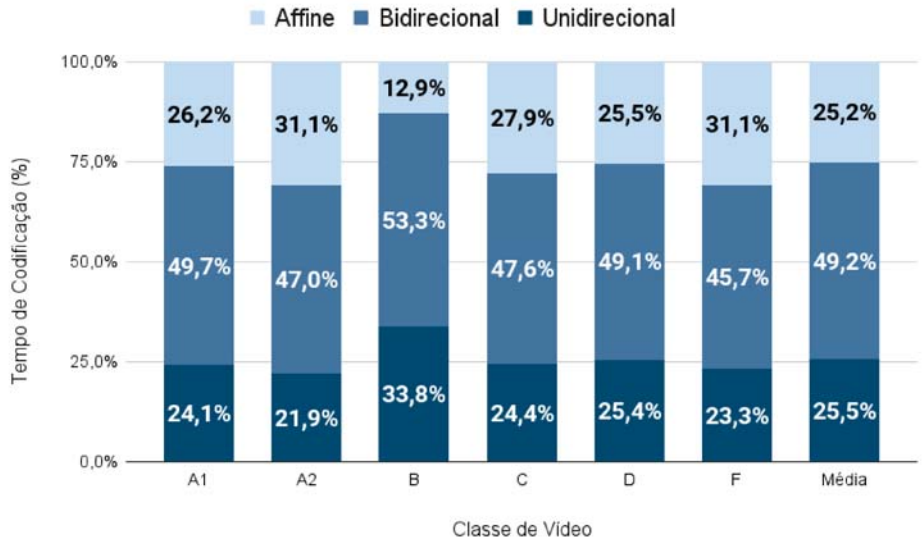
Fonte: Elaborada pelo autor.

A função *Merge* padrão consome a maior parte do tempo de codificação, representando em média 73,2% do tempo total do modo *Merge*. Isto se deve principalmente à implementação de *Extended Merge Prediction* do VVC, onde a lista de candidatos a *Merge* inclui cinco tipos: *Spatial MVP from spatial neighboring CUs*, *Temporal MVP from collocated CUs*, *History-based MVP from a FIFO table*, *Pairwise average MVP* e *Zero MVs*. As funções *AffineMerge* e *MergeGeo* requerem uma parcela menor de tempo de codificação desse modo, com média de 15,6% e 11,2%, respectivamente. Conforme discutido anteriormente, essas funções são executadas para uma quantidade menor de tamanhos de bloco, 19 e 14, respectivamente. Nota-se que o percentual de tempo gasto em cada ferramenta *Merge* permanece consistente em diferentes classes de vídeo, conforme ilustrado na Figura 19.

De maneira similar, foi realizada uma análise com foco no tempo de codificação da ME, considerando as etapas Unidirecional, Bidirecional e *Affine*, categorizados por classe de vídeo. Essas etapas contribuem significativamente para o tempo geral de execução da predição interquadros.

A Figura 20 ilustra os resultados médios de cada classe de vídeo. Em termos da média global, é evidente que o modo Bidirecional é responsável pela maior parte do tempo de codificação ME, compreendendo 49,2% do tempo total. Este elevado esforço computacional é resultado da complexidade inerente associada ao uso de dois referenciais para construir os blocos candidatos. Também é influenciado pelas novas ferramentas introduzidas no VVC, como SMVD, BCW e BDOF, por exemplo.

Figura 20 – Percentual do tempo de codificação das etapas ME (por Classe de Vídeo).



Fonte: Elaborada pelo autor.

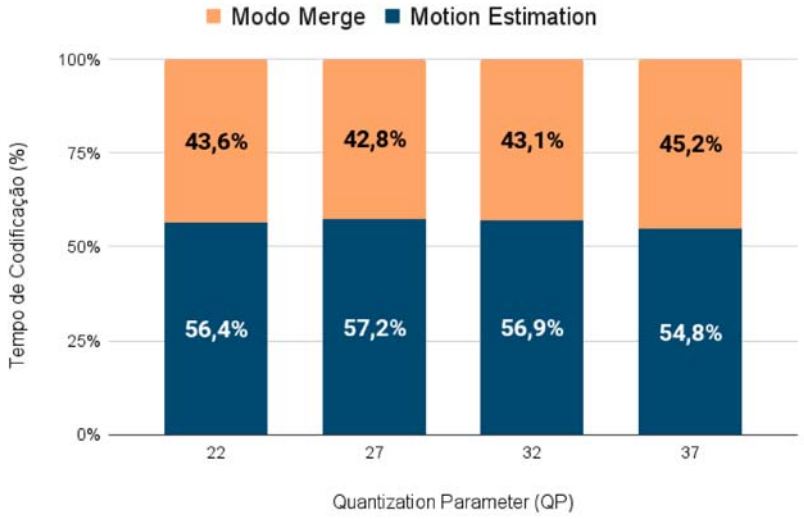
As etapas Unidirecional e *Affine* apresentam resultados médios semelhantes, constituindo 25,5% e 25,2% do tempo médio de codificação, respectivamente. A *Affine* ME requer um esforço computacional maior que a ME Unidirecional para cinco das seis classes de vídeo avaliadas; a única exceção é a classe B.

5.3 Avaliação do Tempo de Codificação por QP

Esta avaliação também se concentra no tempo de codificação, mas agora os resultados são agrupados por QP. Neste caso, o tempo de codificação obtido é observado de uma perspectiva diferente, correlacionando o tempo de codificação e o QP para diferentes ferramentas de predição interquadros.

A Figura 21 exibe porcentagens médias de tempo de codificação considerando as duas principais funções da predição interquadros: *Motion Estimation* (ME) e o Modo *Merge*. Os resultados indicam que o ME requer um tempo de codificação ligeiramente

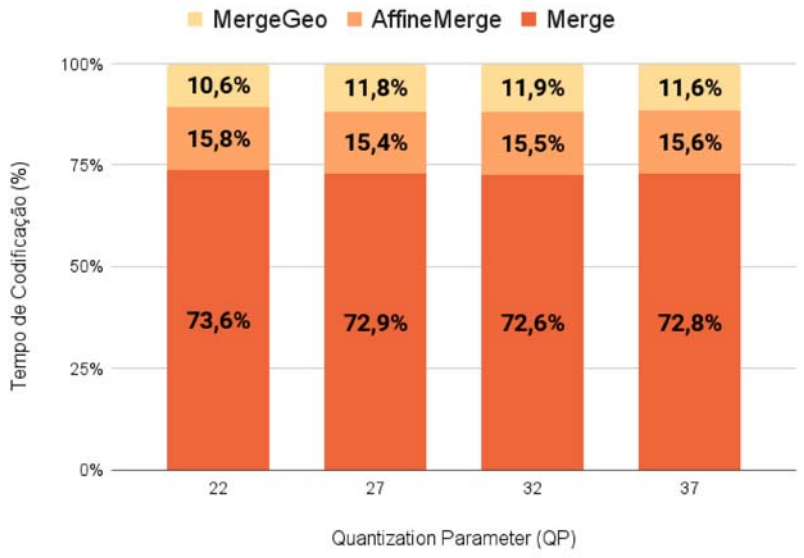
Figura 21 – Percentual do tempo de codificação dos principais modos da predição interquadros (por QP).



Fonte: Elaborada pelo autor.

maior do que o modo *Merge* para todos os QPs avaliados, consistente com os resultados apresentados na Figura 18. Outra observação é a estabilidade da distribuição percentual do tempo entre essas ferramentas para os quatro QPs avaliados, com pequena tendência de aumento do tempo gasto no modo *Merge* à medida que o QP aumenta.

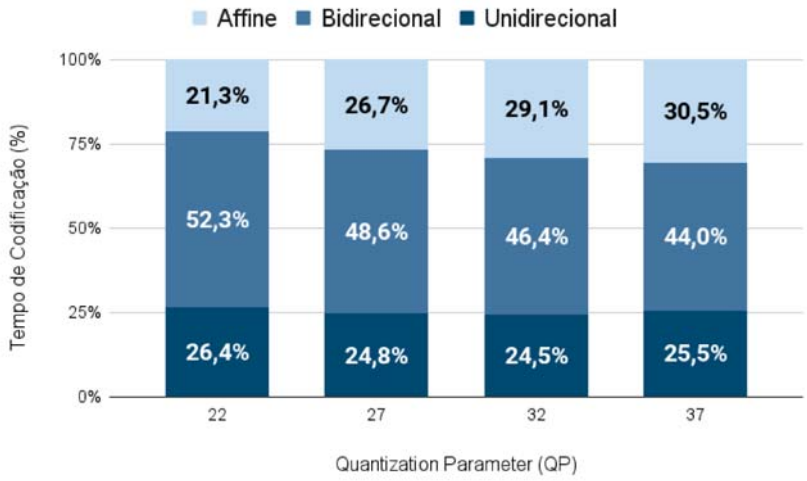
Figura 22 – Percentual do tempo de codificação das funções do Modo *Merge* (por QP).



Fonte: Elaborada pelo autor.

A Figura 22 apresenta a distribuição do tempo de codificação para as principais funções do modo *Merge*. Neste caso, variações no valor do QP não resultam em alterações significativas no percentual de tempo de codificação por função *Merge*. Esses resultados estão alinhados com os apresentados na Figura 19, e a distribuição per-

Figura 23 – Percentual do tempo de codificação das etapas ME (por QP).



Fonte: Elaborada pelo autor.

centual do tempo para cada QP avaliado está próxima do valor médio ali apresentado.

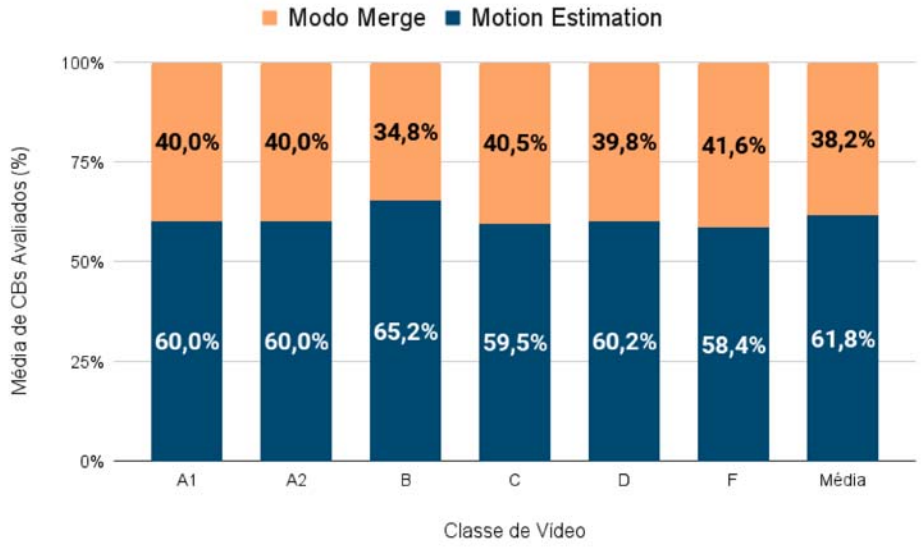
Os resultados para a codificação do tempo nas etapas Unidirecional, Bidirecional e *Affine* são ilustrados na Figura 23, categorizados por QP. Notavelmente, alterações no valor do QP resultam em alterações perceptíveis nos resultados. Um QP mais alto corresponde a uma porcentagem maior de tempo alocado para ME *Affine*, enquanto o inverso é verdadeiro para a ME Bidirecional, ou seja, valores mais altos de QP se correlacionam com menor tempo gasto nesta etapa. Apesar dessas variações, a ME Bidirecional consome consistentemente a maior parte do tempo de codificação em todos os QPs avaliados. Por outro lado, o tempo gasto na ME Unidirecional permanece relativamente constante em diferentes QPs. Além disso, uma observação crucial é que a *Affine* ME utiliza o menor percentual de tempo de codificação no QP mais baixo. Porém, em todos os outros casos, a ME Unidirecional ocupa esta última posição em termos de utilização de tempo.

5.4 Análise dos CBs Avaliados por Classe de Vídeo

Esta seção fornece uma análise do número de CBs avaliados por classe de vídeo, considerando as diferentes ferramentas de codificação discutidas anteriormente. A avaliação visa melhorar a compreensão de como o tempo de codificação se correlaciona com o número de CBs avaliados.

A avaliação inicial é representada na Figura 24, ilustrando o percentual de CBs avaliados nas duas funções principais de predição interquadros nas diferentes classes de vídeo. A *Motion Estimation* (ME) avalia consistentemente um número maior de CBs em comparação ao modo *Merge*, ultrapassando 60% em média para todas as classes, exceto C e F. Ao comparar os resultados da Figura 24 com os da Figura 18, fica evidente um alto grau de correlação em ambas as figuras, como esperado.

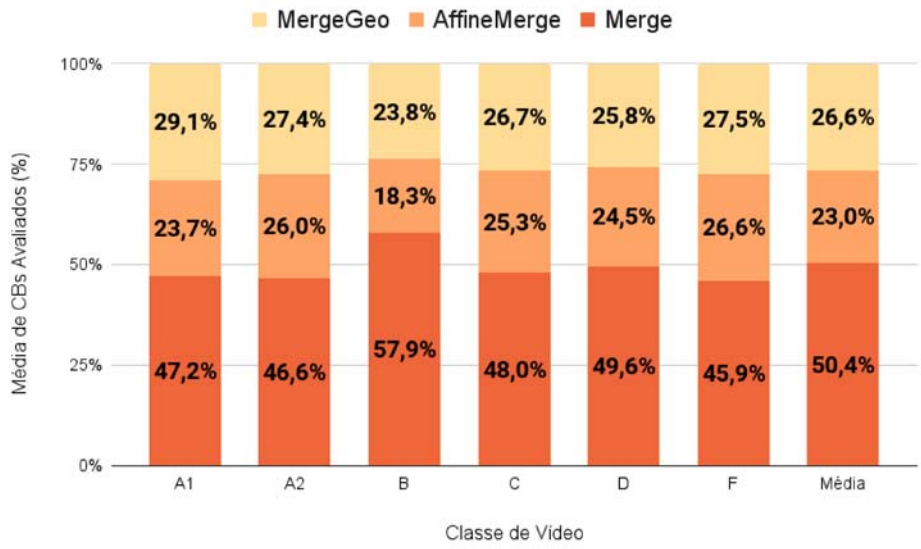
Figura 24 – Percentual de CBs avaliados nos principais modos da predição interquadros (por Classe de Vídeo).



Fonte: Elaborada pelo autor.

À medida que o número de CBs avaliados aumenta, também aumenta o tempo de processamento necessário. No entanto, existe uma diferença notável entre a parcela do tempo necessário (Fig. 18) e os CBs avaliados (Fig. 24) para o modo *Merge*. Isto indica que o tempo requerido pelo modo *Merge* para processar um CB é maior do que o tempo necessário para a ME processar um CB.

Figura 25 – Percentual de CBs avaliados pelas funções do Modo *Merge* (por Classe de Vídeo).



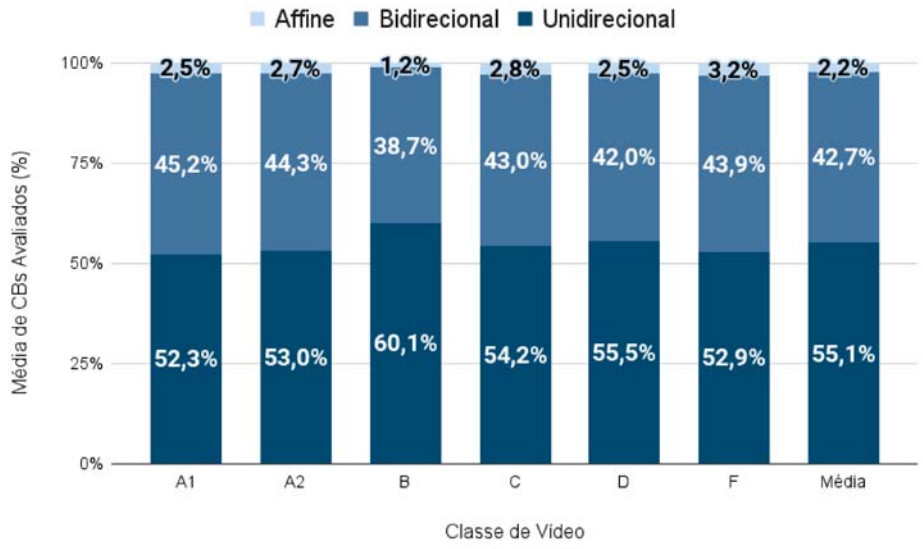
Fonte: Elaborada pelo autor.

Já a Figura 25 ilustra o número médio de CBs avaliados nas funções do modo *Merge*. Em todas as classes de vídeo, a função *Merge* convencional é responsável pela maior porcentagem de CBs avaliados, com média de 50,4%. Em contrapartida, as funções *AffineMerge* e *MergeGeo* compartilham a metade restante da porcentagem,

com 23% e 26,6% dos CBs avaliados em média, respectivamente. O mesmo comportamento observado na análise anterior fica evidente aqui, onde os vídeos da Classe B apresentam as diferenças mais significativas nos resultados médios. Neste caso, o *Merge* convencional atingiu o valor mais alto, com quase 60% dos CBs avaliados, enquanto o *AffineMerge* registrou o menor percentual de CBs avaliados.

Vale ressaltar que ao comparar os resultados apresentados na Figura 25 com os da Figura 19 observa-se que as proporções diferem entre os resultados dessas três ferramentas. Conforme ilustrado na Figura 19, a função *Merge* convencional requer mais de 70% do tempo total de codificação do modo *Merge*, mas avalia pouco mais de 50% dos CBs. Isto implica que a função *Merge* convencional aplica operações mais complexas sobre os CBs avaliados do que as outras ferramentas do modo *Merge*. Por outro lado, as funções *MergeGeo* e *AffineMerge* apresentam comportamento oposto: embora avaliem uma porcentagem maior dos CBs (26,6% e 23% em média), elas requerem menos tempo de processamento (11,2% e 15,6% em média). Isto sugere que a complexidade das operações aplicadas nas funções *MergeGeo* e *AffineMerge* tende a ser menor do que aquela aplicada pela função *Merge* convencional. Outra observação interessante é que as funções *MergeGeo* e *AffineMerge* trocam de posição ao comparar seus resultados na Figura 25 e na Figura 19. *MergeGeo* é a função que avalia a segunda maior porcentagem de CBs, mas fica em último lugar na avaliação do tempo de processamento. Isso indica que, apesar de avaliar muitos CBs, *MergeGeo* possui um nível de complexidade menor comparado às outras funções *Merge*.

Figura 26 – Percentual de CBs avaliados nas etapas ME (por Classe de Vídeo).



Fonte: Elaborada pelo autor.

A Figura 26 ilustra o percentual de CBs avaliados na ME, considerando as etapas Unidirecional, Bidirecional e *Affine* ME. A etapa Unidirecional mostra consistentemente os valores mais altos em todas as classes, seguida pelas etapas Bidirecional e *Affine*.

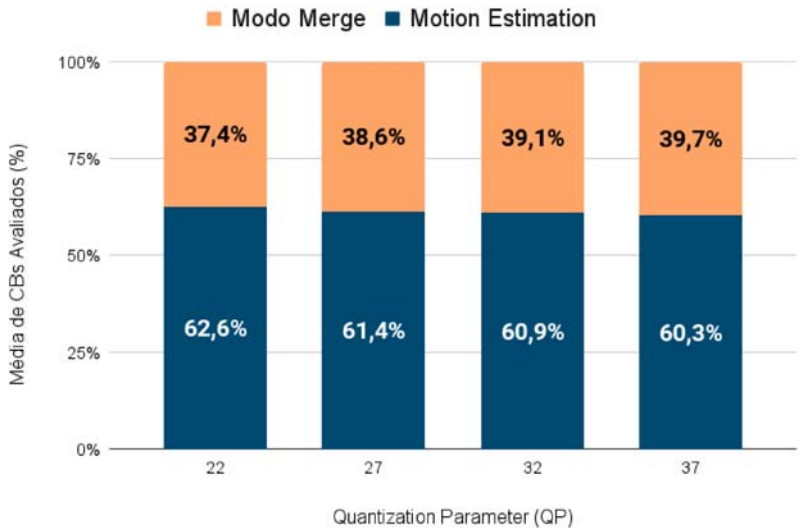
O comportamento permanece relativamente consistente entre as diferentes classes de vídeo, com as principais diferenças observadas nos vídeos Classe B, onde a ME Unidirecional avalia a maior porcentagem de CBs.

Os resultados representados na Figura 26 são particularmente intrigantes quando comparados com os da Figura 20. Um ponto notável de comparação surge ao examinar a *Affine* ME. Notavelmente, a *Affine* ME avalia um percentual baixo de CBs em contraste com as outras etapas da ME, embora com operações mais complexas. Apesar de avaliar apenas uma média de 2,2% dos CBs, a *Affine* ME é responsável por 25,2% do tempo total de codificação do ME. Em contraste, a ME Bidirecional constitui 49,2% do tempo total de codificação do ME, processando 42,7% de todos os CBs avaliados, refletindo uma distribuição equilibrada dos recursos de processamento. Concluindo, a ME Unidirecional, que consome 25,5% do tempo médio de codificação, tem a tarefa de avaliar a maioria dos CBs (55,1% em média). Isto significa que as suas operações são caracterizadas pela menor complexidade entre as três principais ferramentas da ME.

5.5 Análise dos CBs Avaliados por QP

As análises apresentadas nesta seção também levam em consideração o número de CBs avaliados. Entretanto, os resultados são categorizados com base no valor do QP, com o objetivo de explorar relações entre esta variável e a avaliação dos CBs.

Figura 27 – Percentual de CBs avaliados nos principais modos da predição interquadros (por QP).



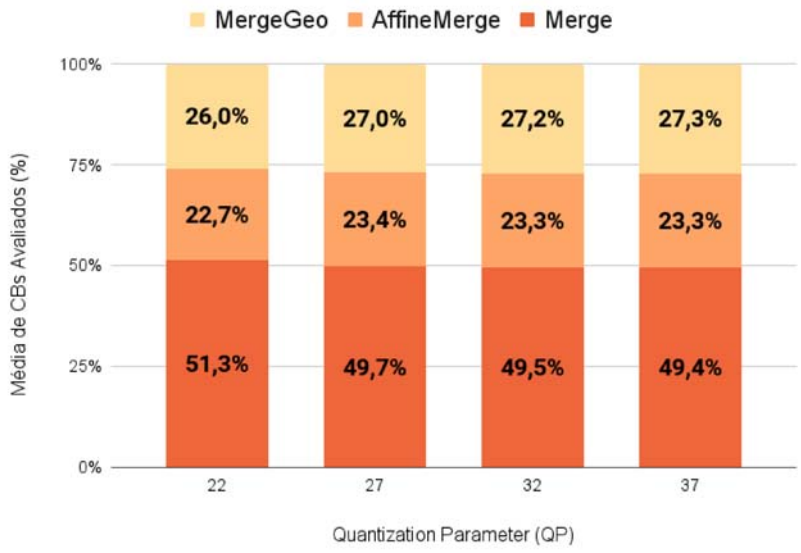
Fonte: Elaborada pelo autor.

A Figura 27 ilustra a porcentagem de CBs avaliados pela ME e pelo modo *Merge*. Os resultados indicam que a ME avalia mais de 60% do total de CBs em todos os QPs, e os valores permanecem consistentemente próximos para diferentes QPs. Adicional-

mente, a figura revela que à medida que o valor de QP aumenta, há uma tendência de ligeira diminuição na percentagem de CBs avaliados na ME, embora essa variação seja mínima.

Ao comparar os resultados da Figura 27 com os da Figura 21, fica evidente uma alta correlação. No entanto, a ME apresenta percentuais mais elevados na Figura 27 do que na Figura 21. Esse padrão também é consistente com os resultados agrupados por classe de vídeo, indicando que a ME avalia mais CBs, embora com operações por CB que sejam menos complexas do que aquelas empregadas pelo modo *Merge*.

Figura 28 – Percentual de CBs avaliados pelas funções do Modo *Merge* (por QP).



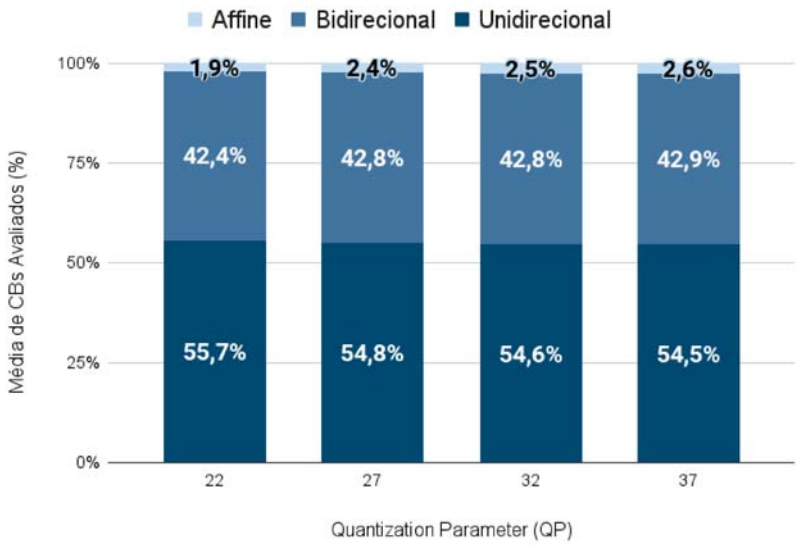
Fonte: Elaborada pelo autor.

Na Figura 28 a análise sobre o percentual de CBs avaliados em cada função do modo *Merge* por valor de QP mostrou que existe uma pequena variação entre os diferentes QPs considerados. O percentual médio de CBs avaliados foi de 26,8% no Merge Geo, 23,2% no *Affine Merge* e 50% no Merge convencional, considerando os quatro QPs. À medida que o QP aumenta, o percentual de CBs avaliados pela função *MergeGeo* tende a subir um pouco, variando em 1,3% do menor para o maior QP avaliado. Por outro lado, os resultados convencionais de *Merge* mostram a tendência oposta, com uma variação de 1,9% entre os quatro QPs. Apesar destas variações, as diferenças geralmente são mínimas. As principais informações inéditas aqui são encontradas ao comparar esses resultados com os resultados da Figura 22. Notavelmente, o modo convencional *Merge* se destaca. Embora consuma mais de 70% do tempo de codificação do modo *Merge*, ele avalia apenas cerca de 50% dos CBs. Isto implica, como discutido anteriormente, que o *Merge* convencional é a ferramenta mais complexa entre os cálculos de *Merge*, exigindo proporcionalmente mais tempo do que o número de CBs avaliados. Outra observação importante diz respeito aos modos *MergeGeo* e *AffineMerge*, que mudam de posição. Estes resultados reforçam a con-

clusão de que *MergeGeo* é a ferramenta Merge com menor complexidade por CB. Ele avalia mais CBs que o *AffineMerge* mas ainda exige menos tempo de codificação.

A Figura 29 descreve o percentual de CBs avaliados pelas principais ferramentas da ME, organizados por QP. Mais uma vez, os resultados apresentam variação mínima entre os diferentes QPs. A porcentagem de CBs avaliados pela ME Unidirecional sofre uma ligeira diminuição com o aumento do QP, enquanto o percentual avaliado pela ME *Affine* cresce de acordo com o QP. Os resultados para ME Bidirecional não mostram alterações significativas. A comparação destes resultados com os da Figura 23 reforça as conclusões quanto à baixa complexidade da ME Unidirecional. Apesar de avaliar mais de 50% dos CBs, requer apenas cerca de 25% do tempo de codificação. Outra conclusão reafirmada diz respeito à *Affine* ME, que processa um percentual relativamente pequeno de CBs mas consome mais de 20% do tempo total de codificação.

Figura 29 – Percentual de CBs avaliados nas etapas ME (por QP).



Fonte: Elaborada pelo autor.

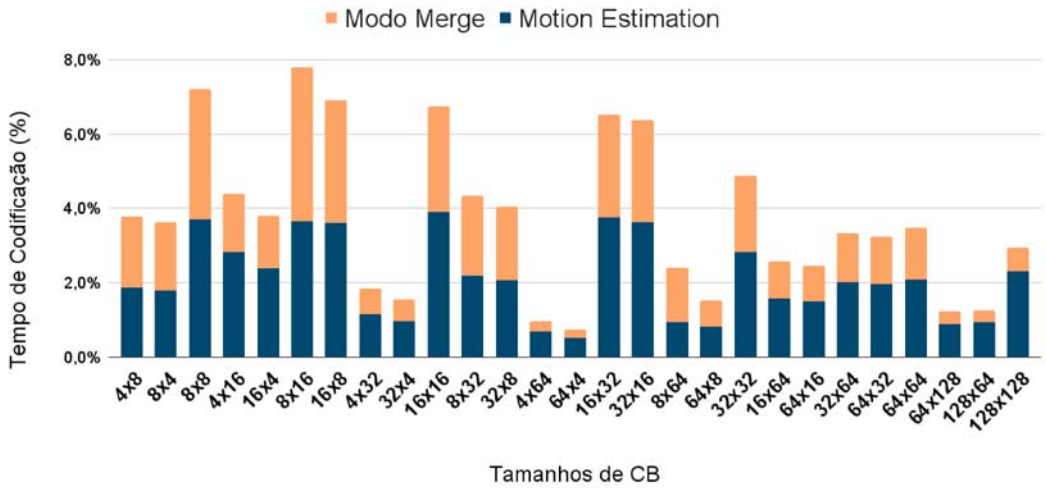
5.6 Avaliação do Tempo de Codificação por Tamanho de Bloco (CB)

Esta seção fornece uma avaliação abrangente do tempo de codificação por tamanho de CB. A análise inicial é mostrada na Figura 30, ilustrando a porcentagem de tempo de codificação necessária para cada tamanho de CB tanto na ME quanto no modo *Merge*. A verificação dos resultados permite inferir uma correlação entre o tamanho, a forma e o tempo de codificação do CB. Vale ressaltar que tamanhos de blocos menores e fatores de forma mais próximos de 1:1 tendem a resultar em tempos de codificação mais elevados para o CB. Isso ocorre porque, em uma área qualquer do quadro, é necessário avaliar mais blocos menores e mais quadrados do que blocos

maiores e mais retangulares.

Além disso, observa-se que a ME geralmente exige mais tempo de codificação em comparação com o modo *Merge* na maioria dos tamanhos de CB. Em casos específicos, como os blocos 8×8 e 8×64, o modo *Merge* requer mais tempo de codificação. Por outro lado, em casos como os blocos 4×8, 8×16, 8×32 e 32×8, os tempos são semelhantes. Para os demais tamanhos de CB, a ME domina a maior parte do tempo.

Figura 30 – Percentual do tempo de codificação dos principais modos da predição interquadros (por tamanho de CB).

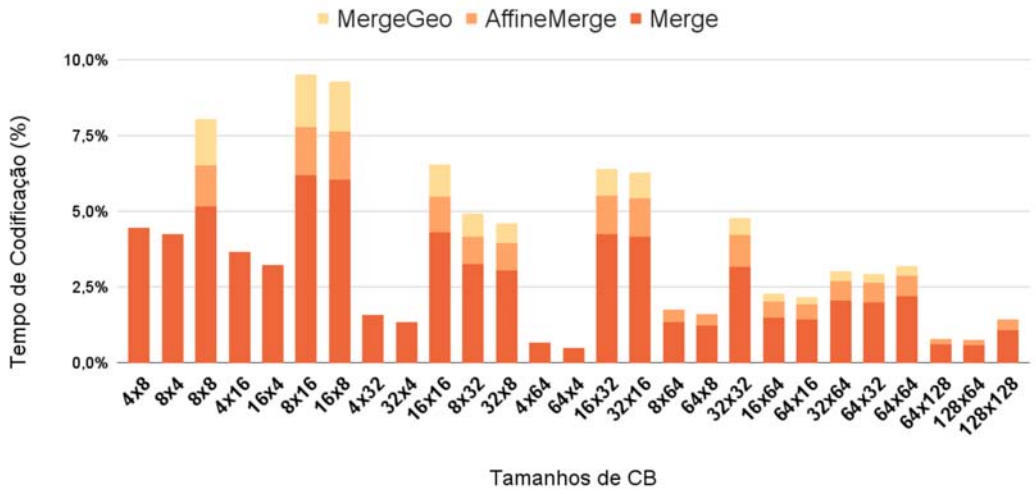


Fonte: Elaborada pelo autor.

Na Figura 31, o gráfico ilustra o percentual de tempo de codificação por tamanho de CB para as funções específicas do modo *Merge*. É fundamental ressaltar que a função convencional *Merge* é testada para todos os 27 tamanhos de CB, enquanto as funções *AffineMerge* e *MergeGeo* são testadas para 19 e 14 tamanhos CB, respectivamente. Consequentemente, os CBs com os tempos de codificação mais elevados são geralmente aqueles onde todas as funções *Merge* estão disponíveis. Outra observação importante é a correlação entre o tamanho do CB e o fator de forma do CB, conforme identificado na análise anterior, o que também fica evidente nestes resultados.

A Figura 32 apresenta o tempo médio de codificação para as etapas Unidirecional, Bidirecional e *Affine* da ME por tamanho de CB. Conforme mencionado anteriormente, a *Affine* ME está habilitada exclusivamente para CBs com dimensões (largura e altura) a partir de 16 amostras, resultando na avaliação de apenas 12 tamanhos de CB. Da mesma forma, a ME Bidirecional não é ativada para CBs com dimensões de 4×8 e 8×4, conforme discutido anteriormente. Ao examinar os resultados, torna-se aparente que, quando aplicável, a ME Bidirecional possui o maior tempo de codificação para CBs menores que 32×32 amostras. Para CBs maiores, a *Affine* ME incorre no maior tempo de codificação. Além disso, a porcentagem de tempo de codificação exigida pela *Affine* ME aumenta proporcionalmente conforme o tamanho do CB. Em contraste,

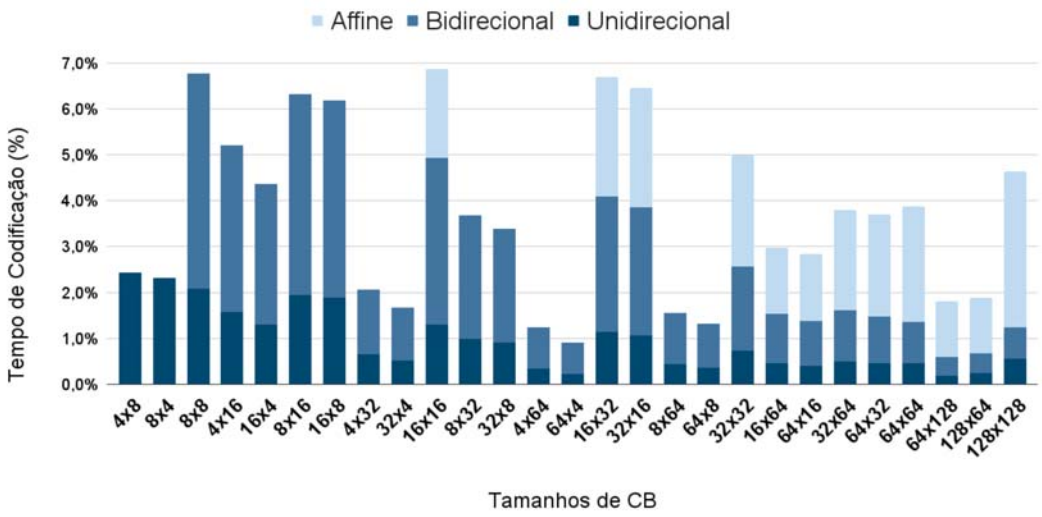
Figura 31 – Percentual do tempo de codificação das funções *Merge* (por tamanho de CB).



Fonte: Elaborada pelo autor.

as etapas Unidirecional e Bidirecional demonstram um padrão inverso: quanto menor o tamanho do CB e quanto mais próximo o formato do CB estiver de uma proporção de 1:1, maior tende a ser o percentual de tempo de codificação para essas ferramentas.

Figura 32 – Percentual do tempo de codificação nas etapas ME (por tamanho de CB).



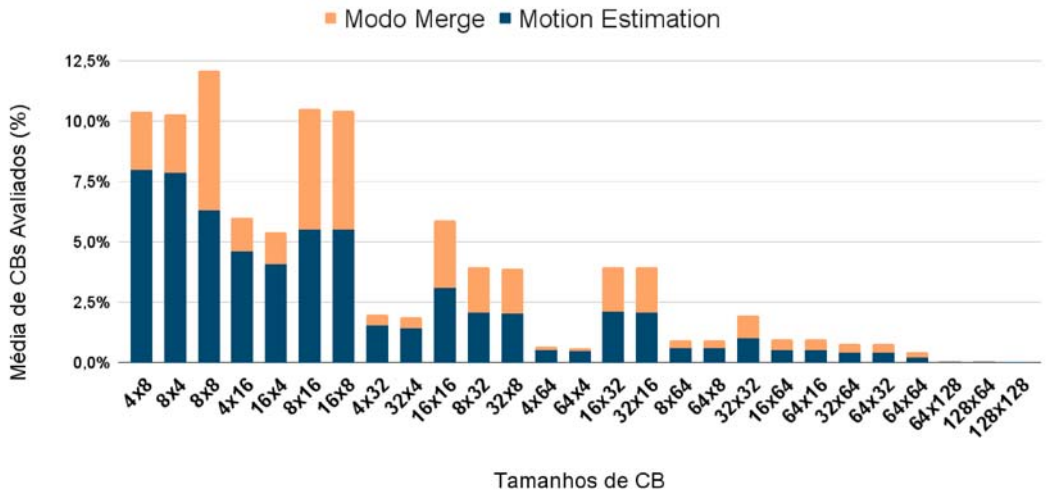
Fonte: Elaborada pelo autor.

5.7 Análise da Distribuição da Avaliação por Tamanho de Bloco (CB)

Esta seção fornece uma análise da distribuição das avaliações dos CBs com base nos seus respectivos tamanhos. A análise inicial é ilustrada na Figura 33, que representa o número médio de CBs avaliados na ME e no modo *Merge* em vários tamanhos de CB. Ao analisar esses resultados, torna-se evidente que os CBs menores e aque-

les com fator de forma próximo a 1:1 são avaliados com mais frequência do que os CBs de outros tamanhos. Esta tendência é observada em ambos os modos da predição interquadros, com maior destaque na ME. Além disso, os resultados indicam que as disparidades primárias no número de CBs avaliados entre os dois modos de predição são evidentes para CBs menores (4×8, 8×4) e CBs maiores de formas retangulares. Nestes casos, a ME avalia um número significativamente maior de CBs em comparação com o modo *Merge*.

Figura 33 – Percentual de CBs avaliados nos principais modos da predição interquadros (por tamanho de CB).



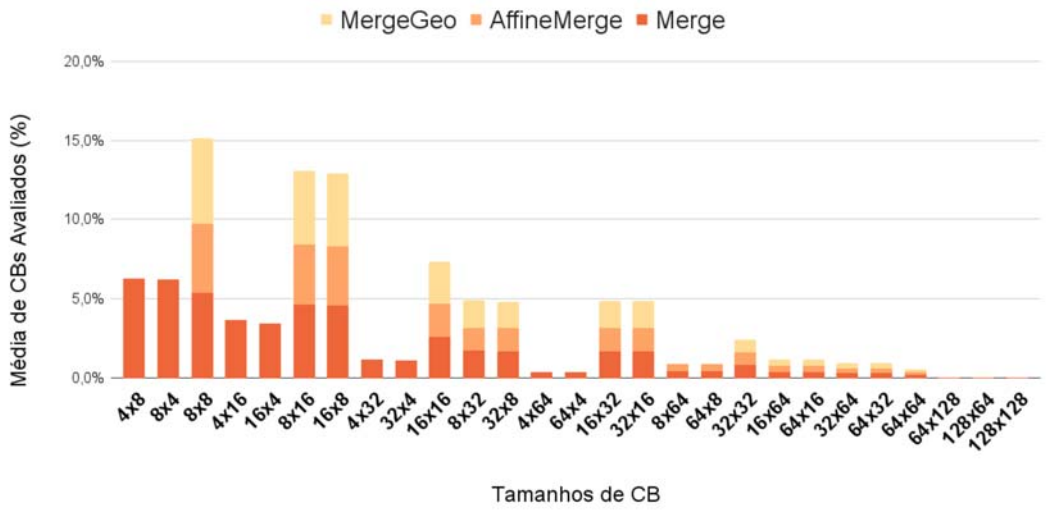
Fonte: Elaborada pelo autor.

Os resultados apresentados na Figura 33 e na Figura 30 levam à conclusão de que o tempo de codificação e o número de CBs avaliados não estão correlacionados. Na verdade, CBs maiores raramente são avaliados, mas requerem uma fatia importante do tempo de codificação. Por outro lado, CBs menores, apesar de serem avaliados com mais frequência, não apresentam aumento proporcional no uso do tempo de codificação. Esta observação está alinhada com as expectativas, pois um maior número de amostras dentro de um CB tende a corresponder a um maior tempo de processamento para aquele bloco específico.

Na Figura 34, o gráfico ilustra o percentual de CBs avaliados para as funções *Merge* específicas. É importante reiterar que a disponibilidade das funções *AffineMerge* e *MergeGeo* não é uniforme em todos os tamanhos de CB. Quando todas as funções *Merge* estão acessíveis, os tamanhos de CB com essas funções tendem a passar por mais avaliações em comparação com outros CBs com um número equivalente de amostras. Esta observação elucida por que tamanhos de CB como 8×8, 8×16, 16×8 e 16×16 são avaliados com mais frequência no contexto dos modos de *Merge*.

Para a função *Merge* convencional, blocos menores como 4×8 e 8×4 são predominantemente avaliados. CBs menores apresentam maior probabilidade de avaliação no modo *Merge* convencional em comparação com seus equivalentes maiores. Além

Figura 34 – Percentual de CBs avaliados nas funções *Merge* (por tamanho de CB).



Fonte: Elaborada pelo autor.

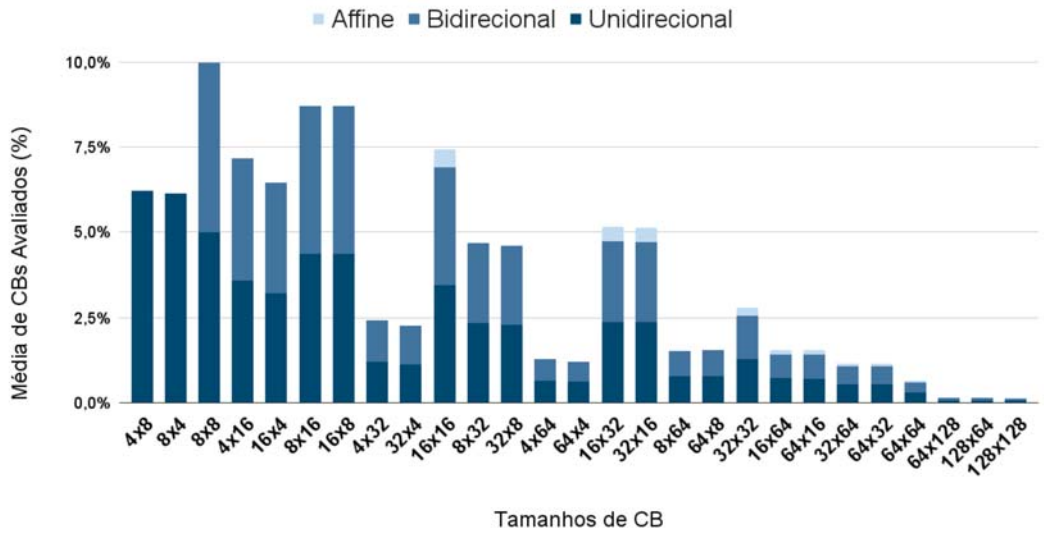
disso, o fator retangular de um CB se correlaciona inversamente com sua frequência de avaliação, o que significa que os CBs com um fator retangular mais alto têm menor probabilidade de serem avaliados quando comparados aos CBs com o mesmo número de amostras. Notavelmente, ambos os modos *MergeGeo* e *AffineMerge* exibem uma tendência decrescente no uso à medida que o tamanho do CB aumenta.

Outra observação decorrente da comparação destes resultados com aqueles apresentados na Figura 31 é que CBs maiores tendem a passar por menos avaliações enquanto exigem uma quantidade substancial de tempo de codificação. Este fenômeno está alinhado com a discussão anterior, uma vez que os CBs maiores envolvem inerentemente o processamento de um maior número de amostras, contribuindo para o aumento do tempo de codificação necessário.

A Figura 35 apresenta o desempenho médio dos CBs avaliados nas etapas da ME Unidirecional, Bidirecional e *Affine*. É importante observar que a ME *Affine* e Bidirecional não estão disponíveis para todos os tamanhos de CB. Ao examinar os resultados apresentados, fica evidente que CBs menores, caracterizados por menor número de amostras, são avaliados com maior frequência. Em todas as ferramentas ME, o tamanho CB 8×8 surge como o mais comumente avaliado, seguido pelos tamanhos CB 8×16, 16×8 e 16×16. Notavelmente, cada um desses quatro tamanhos de CB é avaliado em mais de 7% dos casos.

Para os dois menores CBs representados na Figura 35, apenas a ME Unidirecional está disponível. Nos casos em que a ME Bidirecional é uma opção, há um equilíbrio no número de CBs avaliados entre a ME Unidirecional e Bidirecional. No entanto, conforme discutido anteriormente, para dois tamanhos significativos de CB (4×8 e 8×4), apenas a ME Unidirecional está disponível. Esta circunstância explica os resultados apresentados anteriormente, indicando que a ME Unidirecional é a etapa com maior

Figura 35 – Percentual de CBs avaliados nas etapas ME (por tamanhos de CB).



Fonte: Elaborada pelo autor.

nível de avaliação dos CBs. Além disso, vale ressaltar que a *Affine* ME é utilizada com menos frequência em todos os cenários onde esta ferramenta está disponível. Isso contribui para o nível notavelmente baixo de CBs avaliados apresentado na Figura 35 quando a *Affine* é levada em consideração.

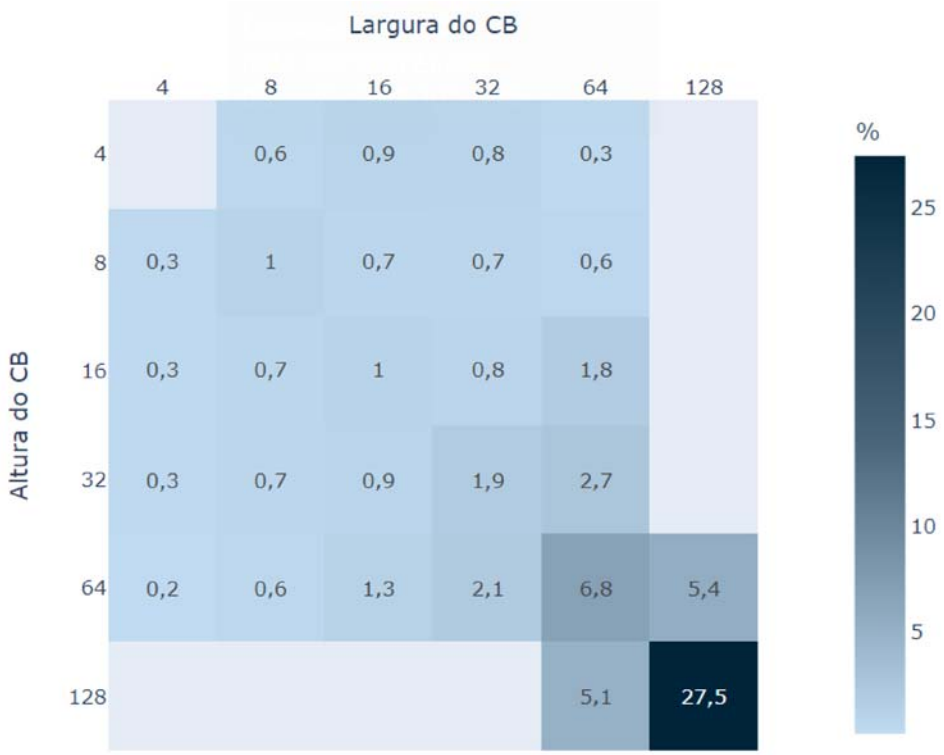
Por outro lado, ao comparar estes resultados com os apresentados na Figura 32, fica evidente que o tempo de codificação e o número de CBs avaliados não estão correlacionados. Notavelmente, o modo *Affine*, onde os CBs são avaliados com pouca frequência, contribui significativamente para o tempo geral de codificação. Uma observação paralela pode ser feita para a ME Bidirecional, que, apesar de ser menos utilizada que o ME Unidirecional, demanda um tempo de codificação substancialmente maior.

Finalmente, é apresentada uma análise, baseada em CBs de luminância, que fornece informações sobre com que frequência um determinado tamanho de CB é selecionado como a melhor representação para uma área de quadro. É importante observar que múltiplas combinações de tamanhos de CB devem ser avaliadas para cada CTB para determinar a combinação mais eficiente em termos de taxa-distorção.

A Figura 36 apresenta a porcentagem de CBs selecionados como a melhor opção em relação ao número de vezes que o tamanho do CB foi avaliado. Esses valores são calculados para toda a etapa interquadros em todos os 27 tamanhos CB, sem quaisquer considerações de ponderação. As células cinza indicam combinações de CB não suportadas pela interquadros do padrão VVC. Destaca-se o tamanho 128×128 CB, sendo selecionado 27,5% das vezes que é avaliado. Da mesma forma, CBs 64×64, 64×128 e 128×64 também mostram destaque em comparação com outros tamanhos de CBs. Naturalmente, esses CBs maiores são avaliados com menos frequência, pois

o particionamento de blocos segue uma partição de árvore recursiva de CBs maiores para menores (por exemplo, um 128×128 CB compreende 256 8×8 CBs). Os resultados na Figura 36 esclarecem a eficiência limitada do VVC na definição das melhores opções de codificação para predição interquadros. Consequentemente, isto também explica o tempo de codificação substancial exigido pelo VVC. O cenário mais desafiador é observado para o CB 4×64, onde apenas 0,2% das avaliações resultam na melhor seleção. Em termos práticos, isso implica que 500 CBs devem ser avaliados para selecionar apenas um para uso, sendo descartados os outros 499 cálculos.

Figura 36 – Percentual de vezes que cada tamanho de CB é selecionado como o melhor CB em relação ao número de vezes que esse tamanho de CB é avaliado.



Fonte: Elaborada pelo autor.

5.8 Avaliação do Uso de CBs Ponderado pelo Número de Amostras

Esta seção investiga resultados experimentais adicionais relativos à utilização de CBs, levando em consideração uma ponderação baseada no número de amostras em cada tamanho de CB. As duas primeiras análises foram agrupadas por resolução de vídeo (e não por classe de vídeo) porque a análise com a ponderação aplicada torna-se mais interessante quando o tamanho do vídeo (resolução) é correlacionado com o tamanho do CB.

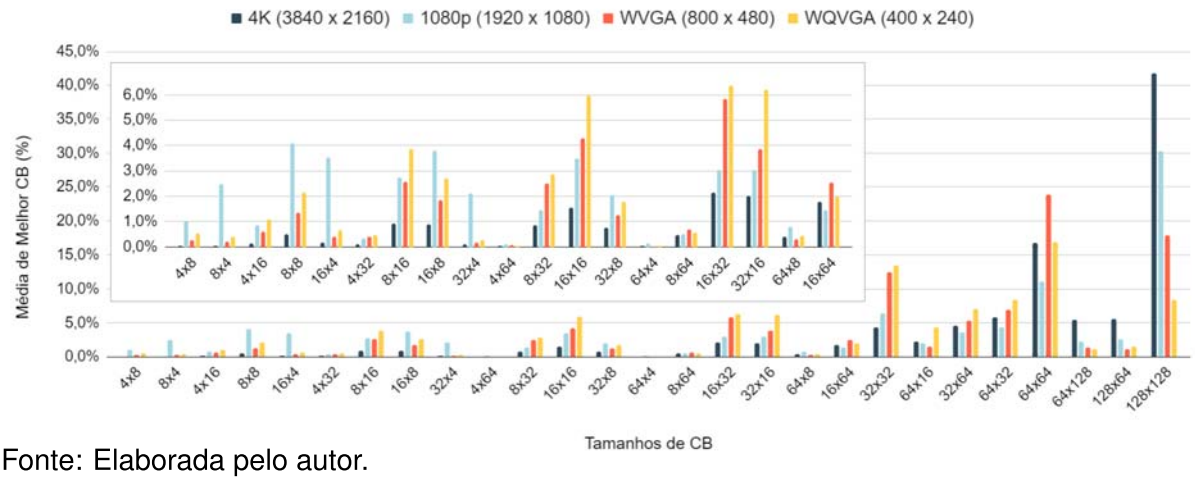
A Figura 37 ilustra a porcentagem de amostras de vídeo codificadas com cada

tamanho de CB. A média dos valores é calculada em todos os QPs para quatro resoluções de vídeo. Para melhor compreensão, a figura inclui uma seção ampliada com valores menores. O principal destaque deste experimento é que CBs maiores (32×32 ou maiores) são os tamanhos de CB mais comumente usados para codificação de amostras de vídeo, mesmo em vídeos com resoluções mais baixas. Uma observação secundária é que resoluções de vídeo mais altas possuem uma porção maior da área de vídeo sendo codificada com CBs maiores usando predição interquadros. Especificamente, para vídeos 4K, mais de 40% das amostras são codificadas usando 128×128 CBs. Em contraste, este tamanho CB codifica menos de 10% das amostras em vídeos WQVGA. Por outro lado, tamanhos menores de CB codificam menos de 5% das amostras de vídeo. O padrão inverso surge para vídeos de resolução mais baixa: à medida que a resolução do vídeo diminui, há uma tendência de maior utilização de CBs menores. Uma exceção surge para vídeos 1080p, onde, em alguns casos, é necessária uma quantidade maior de CBs menores em comparação com vídeos WVGA e WQVGA. Notavelmente, os resultados para CBs 16×16 e 128×128 CBs exibem uma inversão na ordem de uso entre as resoluções de vídeo. Este resultado está alinhado com as expectativas, uma vez que resoluções de vídeo mais altas têm maior probabilidade de conter regiões com texturas homogêneas, codificadas eficientemente usando CBs maiores. Naturalmente, vídeos de resolução mais baixa tendem a apresentar áreas com texturas heterogêneas, onde CBs menores alcançam melhor eficiência de codificação.

À primeira vista, os resultados apresentados na Figura 37 podem parecer inconsistentes com os da Figura 33, mas não o são. Os CBs mais pequenos são, de fato, avaliados com muito mais frequência do que os CBs maiores. No entanto, quando estes CBs menores são escolhidos, eles codificam apenas algumas amostras em comparação com CBs maiores. Por exemplo, selecionar um CB de 128×128 como o tamanho ideal representa uma área de quadro com 16.384 amostras. Por outro lado, escolher um CB 4×8 codifica apenas 32 amostras do quadro. Isto resulta numa diferença substancial de 512 vezes e justifica a variação entre os dois gráficos.

A análise final discutida nesta seção é apresentada na Figura 38. Esta figura mostra a porcentagem de amostras de vídeo codificadas com cada tamanho de CB, agrupando os resultados por QP. Além disso, a figura inclui uma seção ampliada com valores menores para melhor visualização. Os resultados na Figura 38 alinham-se com as expectativas: à medida que o QP aumenta, o número de amostras de vídeo codificadas usando CBs maiores tende a aumentar, enquanto a tendência oposta é observada para QPs menores. Este comportamento é esperado porque valores mais elevados de QP resultam em regiões mais homogêneas em quadros reconstruídos, que são codificados de forma mais eficiente usando CBs maiores. Por outro lado, QPs mais baixos exibem um comportamento oposto, preservando detalhes de textura de

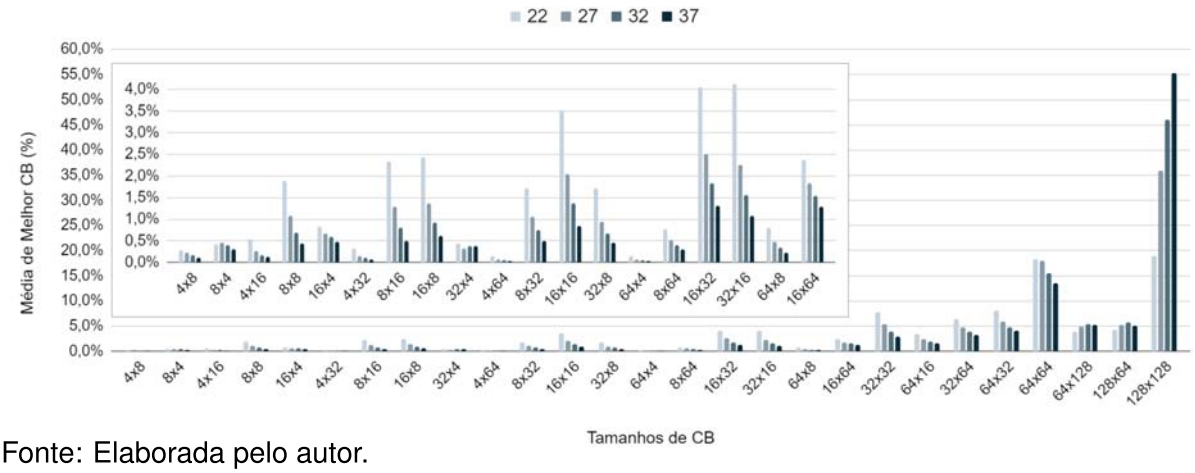
Figura 37 – Percentual de amostras de vídeo codificadas com cada tamanho de CB (por resolução de vídeo).



Fonte: Elaborada pelo autor.

forma mais eficaz, enquanto CBs menores normalmente demonstram melhor eficiência de codificação. Um equilíbrio entre todos os QPs é evidente para tamanhos de CB de 64×128 e 128×64. Notavelmente, o maior CB (128×128) codifica a maioria das amostras de vídeo quando o QP mais alto (37) é aplicado.

Figura 38 – Percentual de amostras de vídeo codificadas com cada tamanho de CB (por QP).



Fonte: Elaborada pelo autor.

5.9 Considerações sobre a Avaliação Experimental da Predição interquadros

Este capítulo forneceu uma avaliação abrangente e aprofundada da predição interquadros VVC. O capítulo inicia com os resultados experimentais de uma avaliação geral detalhada das ferramentas de predição inter e intraquadro do VVC. Além disso, é discutido o tempo de codificação exigido pelas principais ferramentas de predição interquadros, categorizado por classe de vídeo e QP. A quantidade de CBs avaliados por

classe de vídeo e QP também é apresentada e discutida. A seguir, o capítulo fornece uma avaliação aprofundada do tempo de codificação e do número de CBs avaliados, com resultados detalhados por tamanho de CB. Por fim, a seleção dos CBs é examinada, levando em consideração o número de amostras por bloco, e considerando diferentes resoluções de vídeo e QPs.

As análises apresentadas neste capítulo produziram diversas conclusões sobre a predição interquadros do VVC, o que pode ser significativo para estudos futuros neste campo.

Em primeiro lugar, os resultados indicam que a predição interquadros domina o tempo de codificação em todas as classes de vídeo e QPs quando comparada à predição intraquadro, representando mais de 76% do tempo total de codificação. Além disso, existe uma relação inversamente proporcional observada entre o valor do QP e o tempo de codificação, com o QP mais baixo exigindo quase quatro vezes mais tempo de processamento do que o QP mais alto. Por outro lado, embora com uma ligeira variação, valores mais elevados de QP estão associados a uma maior percentagem de tempo gasto na predição interquadros. Ao analisar o número de CBs avaliados, observou-se que a predição interquadros avalia 5,4 vezes mais CBs do que a predição intraquadro. Isto implica que, apesar de um custo computacional global significativamente mais elevado, a predição interquadros implica um custo computacional mais baixo por CB avaliado, uma vez que a diferença no tempo de codificação é menor do que a diferença no número de CBs avaliados.

A avaliação do tempo de codificação para as ferramentas primárias de predição interquadros revela um equilíbrio entre a avaliação dos modos *Motion Estimation* (ME) e *Merge*, com ME demonstrando um custo computacional ligeiramente superior. Curiosamente, este custo tende a apresentar uma redução marginal com o aumento da resolução. O tempo de processamento exigido pelas ferramentas *Merge* mostra pequenas variações entre diferentes classes de vídeo e QPs. Notavelmente, o modo convencional *Merge* é responsável por mais de 70% do tempo de codificação, seguido por *AffineMerge* e *MergeGeo*. Ao examinar as principais ferramentas de ME, fica evidente que a ME Bidirecional demanda a maior parte do tempo de codificação em todas as classes e QPs, seguido pelo ME Unidirecional e *Affine*. Ao avaliar os resultados organizados por QP, fica evidente que há estabilidade no tempo de codificação gasto pela ME Unidirecional entre diferentes QPs. Em contraste, a *Affine* ME demanda, proporcionalmente, mais tempo de acordo com valores mais elevados de QP, enquanto uma tendência inversa é observada para ME Bidirecional.

A análise do número de CBs avaliados revela que a ME avalia mais de 60% dos CBs, em média, em todas as classes de vídeo e QPs. Além disso, à medida que o valor do QP aumenta, o número de CBs avaliados na ME diminui. No entanto, uma observação crucial é que o modo Merge envolve operações mais complexas para processar

um único CB em comparação com a ME. O modo Merge avalia 38,2% dos CBs, em média, mas consome, em média, 43,5% do tempo de codificação. Em contraste, a ME exibe a tendência oposta. Ao analisar as funções Merge, surge um padrão consistente entre as diferentes classes e QPs. O modo convencional *Merge* avalia mais de 50% dos CBs em média, seguido por *MergeGeo* e *AffineMerge*. Notavelmente, a função *Merge* convencional executa operações mais complexas por CB do que outras ferramentas Merge, utilizando 73,2% do tempo total de codificação para processar 50,4% dos CBs, em média. Como esperado, os modos *MergeGeo* e *AffineMerge* exibem o comportamento oposto, com *MergeGeo* demonstrando menor complexidade por CB avaliado do que outros modos Merge. Em experimentos com foco nas principais funções da ME, fica evidente que a ME Unidirecional avalia o maior número de CBs (mais de 50%, em média), seguida pela ME Bidirecional. A *Affine* ME avalia apenas 2,2% dos CBs em média. Uma descoberta importante é que, apesar de avaliar um pequeno número de CBs (apenas 2,2%, em média), a *Affine* ME demanda um tempo de processamento considerável (25,2%, em média), indicando maior complexidade por CB do que outras ferramentas de ME. Por outro lado, a ME Unidirecional processa 55,1% dos CBs, mas requer apenas 25,5% do tempo de codificação. Os resultados demonstram estabilidade em diferentes classes de vídeo e QPs.

A análise do tempo de codificação e o número de CBs avaliados por tamanho de CB produziu conclusões significativas. Em primeiro lugar, existe uma falta de correlação entre o tempo de codificação e o número de CBs avaliados, uma vez que CBs maiores tendem a ser avaliados com menos frequência, embora ainda exijam uma quantidade considerável de tempo de codificação. Isto é esperado, dado que os CBs maiores envolvem o processamento de um maior número de amostras do que os CBs mais pequenos. Em segundo lugar, os CBs com formatos retangulares maiores são menos utilizados e requerem o menor tempo de codificação. Em terceiro lugar, o tempo de codificação do CB está intrinsecamente ligado à relação entre o tamanho do CB e o fator de forma. Tamanhos de bloco menores e fatores de forma mais próximos de 1:1 tendem a incorrer em tempos de codificação mais elevados para os respectivos CBs. Esta tendência também é evidente na avaliação do número de CBs, onde os tamanhos de CBs mais pequenos são avaliados com mais frequência do que os maiores. Este comportamento é consistente em vários experimentos que detalham ferramentas de predição interquadros por tamanho de CB. As observações indicam que a ME geralmente requer mais tempo de codificação e avalia mais CBs do que os modos *Merge* para a maioria dos tamanhos de CB. A investigação nas principais ferramentas ME revela que a ME Bidirecional detém o maior tempo de codificação para CBs de até 32×32. Para CBs maiores, a *Affine* ME assume o maior tempo de codificação, com a parcela do tempo de codificação aumentando à medida que o tamanho do CB aumenta. Apesar da *Affine* ME avaliar um pequeno número de CBs grandes,

reforça-se a conclusão de que a *Affine* ME incorre em um alto custo computacional por CB avaliado. As etapas Unidirecional e Bidirecional, por outro lado, apresentam tendência oposta. Para tamanhos de CB menores e fatores de forma mais próximos de 1:1, a porcentagem de CBs avaliados e a porcentagem de tempo de codificação gasto nessas ferramentas tendem a ser maiores. Na análise final, explorou-se uma correlação entre o número de vezes que o tamanho do CB foi avaliado e o número de vezes que o tamanho do CB foi selecionado como a melhor opção. Esta análise contribui para a compreensão do considerável tempo de codificação exigido pelo VVC, pois, no pior caso, apenas 0,2% das avaliações resultam na melhor seleção.

A avaliação focada na porcentagem de amostras codificadas com cada tamanho de CB revela que CBs maiores são consistentemente os tamanhos mais usados em todas as resoluções de vídeo, uma tendência particularmente pronunciada em vídeos com resoluções mais altas. Adicionalmente, observou-se que vídeos de menor resolução tendem a favorecer o uso de CBs menores. Essa tendência pode ser atribuída ao fato de que vídeos de alta resolução normalmente contêm mais regiões com texturas homogêneas, tornando-os codificados de forma mais eficiente usando CBs maiores. Em contrapartida, vídeos de resolução mais baixa apresentam frequentemente áreas com texturas mais heterogêneas, favorecendo o uso de CBs menores. Embora CBs menores sejam avaliados com mais frequência, CBs maiores são empregados para codificar um número significativamente maior de amostras de vídeo. Quando os resultados são analisados com base no QP, surge um padrão claro e consistente: valores mais elevados de QP correspondem ao aumento do uso de CBs maiores, enquanto o oposto se aplica a QPs menores. Isto pode ser explicado pelo fato de que valores mais elevados de QP tendem a aumentar a homogeneidade da textura, favorecendo mais uma vez a codificação eficiente de regiões homogêneas utilizando CBs maiores.

Os resultados apresentados neste capítulo oferecem uma análise abrangente da predição interquadros definida no padrão VVC. Os experimentos e as discussões subsequentes levam a diversas conclusões, sendo que esses conhecimentos serão valiosos no apoio a futuros esforços de investigação relacionados ao VVC e podem até contribuir para decisões relativas à definição de padrões futuros.

6 REVISÃO SISTEMÁTICA DA LITERATURA

Neste capítulo será descrito o processo de Revisão Sistemática da Literatura (RSL) realizado sobre o tema deste trabalho. As etapas seguidas foram baseadas em (Petersen; Feldt; Mujtaba; Mattsson, 2008): (i) definição das questões de pesquisa; (ii) busca de artigos; (iii) seleção dos artigos; (iv) definição de classificação e extração de dados; e (v) mapeamento e análise dos dados.

6.1 Questões de Pesquisa

Os objetivos específicos de uma RSL são representados através das questões de pesquisa. Esta é uma etapa fundamental, visto que a operacionalização das demais etapas acontecerá com base nesta etapa (Petersen; Feldt; Mujtaba; Mattsson, 2008). É importante que as questões de pesquisa sejam respondidas ao final do processo, para que o objetivo da RSL seja atingido. Especificamente, neste trabalho, foram definidas as seguintes questões de pesquisa:

1) Quais são os trabalhos de pesquisa que apresentam soluções de otimização para a predição interquadros do HEVC?

2) Quais são os trabalhos de pesquisa que apresentam soluções de otimização para a predição interquadros do VVC?

3) Quais são as principais estratégias de aprendizado de máquina utilizadas nas soluções de otimização para a predição interquadros do HEVC e VVC?

4) Quais são as oportunidades de pesquisa ainda abertas na predição interquadros do padrão VVC?

Nesse sentido, ao serem respondidas, as questões de pesquisa possibilitarão um melhor entendimento do estado da arte referente às soluções para a predição interquadros do padrão VVC. Além disso, será possível identificar lacunas de pesquisa que ainda podem ser exploradas.

6.2 Busca por Trabalhos

Esta etapa consiste na busca de trabalhos que auxiliem a responder as questões de pesquisa. Para este trabalho, a pesquisa foi realizada através de palavras-chave de busca, inseridas no site *Google Scholar*, que fornece acesso indireto às bases de dados da IEEE, ACM, Springer e Elsevier, entre outras. A busca foi finalizada em 09/02/2024. Após um processo de refinamento, foram definidas e utilizadas duas *strings*, de acordo com os escopos de trabalhos voltados a soluções de otimização do HEVC (*string* X) e do VVC (*string* Y):

X) Soluções para o HEVC: (“HEVC inter prediction” OR “HEVC inter-frame prediction” OR “HEVC inter frame prediction” OR “HEVC Motion Estimation”) AND ("optimization” OR “machine learning”)

Y) Soluções para o VVC: (“VVC” OR “Versatile Video Coding” OR “H.266”) AND (“inter prediction” OR “inter-frame prediction” OR “inter frame prediction” OR “Motion Estimation” OR “Affine”) AND (“optimization” OR “machine learning” OR “complexity reduction” OR “time reduction” OR “fast”)

Essa divisão na *string* de busca foi necessária devido ao espaço limitado para a sua inserção no site *Google Scholar*. Para a *string* Y, referente ao VVC, foi necessário realizar uma especificação mais detalhada nos termos, visando obter mais resultados. O período considerado de publicação dos trabalhos para a *string* X foi de 2013 até 2023, de acordo com o lançamento do padrão. Já para a *string* Y, o período da busca foi de 2019 até fevereiro de 2024, também iniciando a partir da data de lançamento do padrão. A diferença entre o final de cada período foi pela necessidade de delimitar mais a pesquisa sobre os trabalhos com foco no HEVC, dispondo de maior atenção para os trabalhos com foco no VVC.

6.3 Seleção dos Trabalhos

A partir do resultado inicial da busca, foi realizada uma seleção dos trabalhos através da aplicação dos critérios de inclusão e exclusão. A Tabela 2 especifica esses critérios, que foram definidos com o intuito de atender às questões de pesquisa.

No total, foram encontrados 2758 trabalhos, sendo 908 deles resultantes da *string* X e 1850 da *string* Y. Neste resultado inicial estão incluídos artigos, livros, capítulos de livros e teses. Os critérios de inclusão e exclusão dos trabalhos foram aplicados em duas etapas: i) após a busca inicial, a partir da leitura do título e, se necessário, do resumo dos artigos; ii) refinamento a partir da leitura do resumo, resultados e conclusões. Na Tabela 3 constam os dados quantitativos resultantes dessas etapas.

É importante ressaltar que, devido ao grande número de trabalhos resultantes da pesquisa para o HEVC, foi aplicado um critério específico para definir quais dos 156 trabalhos seriam descritos em mais detalhes na próxima seção. Nesse sentido, fo-

Tabela 2 – Critérios de inclusão e exclusão de trabalhos

Critérios de Inclusão	Critérios de Exclusão
Estudos com foco nos padrões HEVC ou VVC.	Trabalhos que não estão disponíveis na íntegra.
Estudos que apresentam soluções para a etapa interquadros.	Estudos focados em outros codificadores.
Trabalhos escritos em inglês ou português.	Estudos que apresentam soluções em hardware ou paralelismo.
Artigos científicos publicados em periódicos ou eventos	Trabalhos focados na intraquadro, vídeos 360, *light fields* ou *point clouds*.
Trabalhos com análise específica da solução para a etapa interquadros	Trabalhos que não usam as Condições Comuns de Teste (CTCs)
(HEVC) Trabalhos com resultados específicos para a interquadros	(HEVC) Trabalhos focados em melhorar algoritmos de busca da ME.

Tabela 3 – Resultado da seleção de trabalhos

String de Busca	Total inicial	Total refinamento (i)	Total refinamento (ii)
X : HEVC	908	156	7
Y : VVC	1850	51	34
Total	**2758**	**207**	**41**

ram selecionados 7 trabalhos que apresentaram resultados específicos de redução de tempo para a etapa interquadros, podendo ou não relatar também resultados de redução de tempo geral.

6.4 Classificação e Descrição dos Trabalhos - HEVC

Na Tabela 4 é possível visualizar um resumo das técnicas utilizadas e o foco da aplicação dos trabalhos selecionados que são voltados ao padrão HEVC. Também são abordadas as informações quanto à versão do *software* de referência do HEVC (*HEVC Test Model* - HM), a média de redução do tempo de codificação da etapa específica e as perdas de eficiência de codificação (BDBR). A seguir, esses trabalhos serão descritos mais detalhadamente.

A proposta definida por (Pan; Lei; Zhang; Sun; Kwong, 2016) consiste em um método de Estimação de Movimento (ME) rápida, baseado na análise estatística da correlação do melhor vetor de movimento entre os diferentes modos de predição. Primeiramente, é realizada a classificação das PUs como sendo pai (2Nx2N) ou filhas (2N×N, N×2N, N×N, 2N×nU, 2N×nD, nL×2N e nR×2N). Assim, o processo de busca ME das PUs filhas é ignorado de maneira adaptativa quando a PU pai tem o ponto de busca inicial como o seu melhor vetor de movimento. Em comparação com o *software* de referência, os experimentos apresentaram, em média, os seguintes resultados: para a configuração *Low Delay Main*, redução de 20,12% no tempo de codificação da

ME e 15,04% no tempo total de codificação, com BDBR de 0,55%; para a configuração *Random Access*, redução no tempo de codificação de 18,52% para a etapa ME e 12,29% para o tempo total, com impacto de 0,86% em BDBR.

Tabela 4 – Resumo das técnicas, aplicações e resultados dos trabalhos para o HEVC

Trabalho	Versão HM	Média de Redução de Tempo	BDBR	Técnica Utilizada	Foco da Aplicação
Pan et al. (2016)	12.0	20,12%	0,55%	Análise Estatística	Busca ME
Saurty; Catherine; Soyjaudah (2016)	10.0	70,80%	0,20%	Análise Estatística	Particionamento
Cebrián-márquez; et al. (2017)	16.6	88,05%	0,33%	Análise Estatística	ME Inteira
Li et al. (2018)	15.0	45,00%	2,91%	CNN	Decisão de modo
Pan et al. (2018)	12.0	63,82%	1,41%	Análise Estatística	ME Fracionária
Liu; Lin (2019)	16.0	29,60%	0,09%	SVM	Particionamento
Bouaafia et al. (2020)	16.5	53,99%	-0,19%	SVM e CNN	Particionamento

No artigo (Saurty; Catherine; Soyjaudah, 2016) é apresentada uma solução de encerramento prévio do particionamento de CU e de alguns modos, considerando CUs maiores que 8×8. O trabalho realiza uma análise estatística e definição de limiares estáticos com base no valor da média quadrada (*Mean Square* - MS) dos resíduos. São implementados três testes com base na relação entre o maior valor de MS, valor do QP e ocorrências de *Early* CUs, outros modos de PU e particionamento de CU. Sendo assim, após a codificação dos modos *Merge* e *Skip*, o primeiro teste define uma *Early* CU, que em caso positivo, segue para o teste do modo 2Nx2N ou, caso negativo, segue para o teste de particionamento. O teste do modo 2Nx2N, por sua vez, ao retornar positivo, ignora os demais modos (Nx2N, 2NxN, 2NxnU, 2NxnD, nLx2N, nRx2N). Da mesma forma, ocorre com o teste do particionamento. Os resultados experimentais dos três testes aplicados conjuntamente demonstram a redução de 62,2% e 70,8% no tempo total de codificação e no tempo da etapa ME, respectivamente. O impacto em BDBR apresentou o valor de 1,14%.

O trabalho de (Cebrián-márquez; Martínez; Cuenca, 2017), por sua vez, propõe alterações no algoritmo de pré-análise a fim de fornecer informações de codificação para as etapas Intra e interquadros. A etapa de pré-análise realiza uma Estimação de Movimento rápida. Na proposta, essa pré-análise divide o quadro atual em blocos de todos os tamanhos, resultando na estimação dos MVs e seu custo para cada bloco e quadro de referência. Posteriormente, o módulo ME faz uso dessas informações para reduzir o número de quadros de referência testados e prever o ponto inicial da busca

de movimento. São realizadas algumas adaptações para fins de correspondência entre os tamanhos de PU pré-analisados e os codificados. O algoritmo padrão TZSearch é substituído pelo refinamento baseado no padrão hexágono, reduzindo as operações de SAD (*Sum of Absolute Differences*) realizadas na ME. Nesse sentido, os resultados experimentais para a configuração *Random Access* demonstram uma redução de 16,19% no tempo total de codificação, com impacto de BDBR de 0,44%, média de 28,32% de quadros de referência ignorados e 83,25% de redução nas operações SAD na IME. Na configuração *Low Delay* a redução no tempo total é de 20,29%, com impacto em BDBR de 0,61%, média de 30,05% de quadros de referência ignorados e 87,31% de redução nas operações de SAD da IME. O trabalho não apresenta valores de redução de tempo da ME, porém demonstra em gráficos a redução proporcional das diferentes etapas, sendo notável uma redução significativa na etapa IME, o que certamente demonstra um impacto positivo da solução.

A proposta apresentada em (Li; Liu; Ji; Wang, 2018) consiste em um algoritmo de decisão rápida sobre o particionamento de CU, baseado em redes neurais convolucionais (*Convolutional Neural Network* - CNN), para reduzir o tempo de codificação da interquadros. Considera-se que uma CU 2Nx2N, sendo $N \in \{32, 16, 8\}$, pode ser codificada como um todo ou ser dividida em quatro subCUs NxN, cada uma codificada individualmente. Na proposta, é verificado se a CU atual pertence a um bloco homogêneo. Nesse caso, somente o modo 2Nx2N é verificado. Do contrário, é utilizada a CNN para definir o modo de partição de CU, sendo que o seu retorno é HOMO ou SPLIT. Dessa forma, se a CNN retorna SPLIT, o modo 2Nx2N é ignorado; caso contrário, a busca do modo NxN é ignorada. A CNN possui três redes internas: Rede I de análise de textura; Rede II para análise residual da IME e Rede III para análise de vetor de movimento da IME. Os resultados obtidos nos experimentos demonstram a redução de 45% em média no tempo de codificação da predição interquadros, com um impacto de BDBR de 2,91%.

No trabalho (Pan; Lei; Zhang; Wang, 2018) é proposto um esquema de terminação prévia para a Estimação de Movimento Fracionária (FME), elaborado com base em uma análise estatística. Primeiramente, os modos das Unidades de Predição (*Prediction Units* - PU) são classificados como raiz (2Nx2N) e filhos (outros sete modos PU inter). A análise constatou que a maioria dos modos PUs raiz obtém o melhor MV como sendo da Estimação de Movimento Inteira (IME) e suas PUs filhas, por sua vez, têm grande probabilidade de obter o mesmo resultado. O esquema consiste em: codificar a CU atual com o modo *Merge/Skip*; codificar o modo PU raiz com a IME e FME; se o melhor MV for resultante da IME, os modos PU filhos são codificados somente com a IME; caso contrário, são executadas IME e FME para os modos PU filhos; após isso, a codificação da CU atual é finalizada com a aplicação dos modos intra. Os resultados obtidos para a configuração *Low Delay Main* foram a redução no tempo total

e no tempo da ME de 28,82% e 62,60%, respectivamente, com um impacto de BDBR de 1,51%. Já para a configuração *Random Access*, foi relatada a redução de 24,35% no tempo total e 63,82% no tempo da ME, com o impacto de BDBR de 1,41%.

Já o trabalho de (Liu; Lin, 2019) apresenta uma solução de decisão rápida para a etapa interquadros usando o modelo de aprendizado de máquina SVM (*Support Vector Machine*). A solução é composta por uma decisão de tamanho de CU, que utiliza informações como variância de MV, profundidade da CU vizinha e CBF (*Coded Block Flag*), e outra decisão do modo PU, baseada em informações de variância de MV, *flag Skip* e taxa RDO. Especificamente para a decisão de modo de PU, é utilizada uma SVM para classificar os modos de PU em dois subgrupos: SG7 (*Skip/Merge* e modo 2Nx2N inter) e SG8 (11 modos PU). Na proposta, os modos *Skip/Merge* e 2Nx2N inter PU são executados para cada CTU. Posteriormente a isso, as respectivas informações são usadas para classificar a CTU atual. Caso a CTU esteja no grupo SG7, os demais modos de PU não são executados. Se a CTU for classificada como grupo SG8, os demais modos de PU são executados a fim de encontrar o melhor modo de PU. Os resultados experimentais do esquema de decisão de modo PU demonstram uma redução de 9,5% no tempo da inter predição, com impacto de 0,04% em BDBR.

Por fim, em (Bouaafia; Khemiri; Sayadi; Atri, 2020) são definidos e comparados dois métodos de particionamento de CU rápidos, baseados em modelos de aprendizado de máquina. Primeiramente, um modelo de SVM treinado on-line é definido para encerrar antecipadamente o processo de particionamento da CU. No treinamento, alguns quadros são codificados sem restrições, gerando rótulos de classes e recursos para o treinamento do modelo, que aplica as predições sobre a profundidade nos próximos quadros. Já o modelo de CNN organiza camadas de pré-convolução que recebem CUs residuais dos três primeiros níveis (64, 32 e 16), sendo que esse bloco é subtraído pelos valores médios de intensidade para fins de redução da variação das amostras de entrada. Após isso, três camadas convolucionais extraem recursos dos dados em todos os níveis. Os experimentos demonstram o resultado da aplicação dos dois métodos comparados com o *software* de referência na configuração *Low Delay P*. Os modelos CNN e SVM demonstraram uma redução no tempo de codificação da interquadros de 53,99% e 52,28%, com impacto em BDBR de -0,195% e 1,928%, respectivamente. A conclusão do trabalho indica a utilização do modelo CNN, pois o mesmo, além de reduzir o tempo de codificação, obteve melhor desempenho em termos de BDBR.

6.4.1 Discussão dos Resultados - HEVC

A partir da descrição dos trabalhos de otimização focados no HEVC e do resumo apresentado na Tabela 4, é possível responder às questões de pesquisa 1 e 3. Quanto ao foco da otimização, é notável o interesse em reduzir o tempo de codificação na

tarefa do Particionamento dos blocos, presente em três trabalhos. Além disso, outro foco importante de aplicação é a Estimação de Movimento, também presente em três trabalhos, sendo dois voltados especificamente para a busca inteira e fracionária, respectivamente. Em relação às técnicas utilizadas, percebe-se que a maioria dos trabalhos utiliza Análise Estatística de dados da codificação. Entretanto, três dos sete trabalhos utilizam técnicas de Aprendizado de Máquina. Destaca-se a utilização de Redes Neurais Convolucionais (CNN) e Máquinas de Vetores de Suporte (SVM). É importante ressaltar que, para o HEVC, foram selecionados apenas os trabalhos que constavam resultados de redução de tempo de codificação para a etapa específica da interquadros. Dessa forma, apenas sete trabalhos foram selecionados.

6.5 Classificação e Descrição dos Trabalhos - VVC

Esta seção contém a descrição e classificação dos trabalhos de otimização focados no VVC, resultantes das etapas de busca e refinamento. Com o intuito de facilitar o entendimento, a organização da seção acontece em duas partes, de acordo com a técnica utilizada nos trabalhos. Sendo assim, primeiramente são apresentados os 16 trabalhos que utilizam Análise Estatística (ou outra técnica) e, após, os 18 trabalhos que utilizam Aprendizado de Máquina.

6.5.1 Trabalhos que Utilizam Análise Estatística

Especificamente nesta seção serão classificados e descritos os trabalhos de redução de complexidade com foco no VVC e que utilizam técnicas de Análise Estatística. Primeiramente, na Tabela 5 é possível visualizar um resumo sobre os trabalhos, incluindo informações quanto à versão do *software* de referência do VVC (*VVC Test Model* - VTM), a média de redução do tempo de codificação total e da etapa específica, as perdas de eficiência de codificação (BDBR) e o foco de cada trabalho. Os dados da tabela estão dispostos em ordem crescente de ano e sobrenome do primeiro autor. Ressalta-se que alguns trabalhos só apresentam um dos resultados de redução de tempo de codificação, sendo que estes estão representados com um "hífen"(-) na tabela. A seguir, os trabalhos serão descritos com um detalhamento maior.

A solução de Pan et al. (2019) é focada em otimizar a ME. Primeiramente, é proposto um algoritmo de configuração de intervalo de pesquisa adaptativo que é orientado ao conteúdo. Assim, o tamanho do intervalo de pesquisa das CUs filhas é definido de maneira adaptativa usando a informação do melhor MV da CU pai. Para reduzir ainda mais o tempo de codificação, também é definido um algoritmo rápido de decisão quanto à direção do quadro de referência, com base na alta correlação espacial e características semelhantes presentes na CU pai. Nesse caso, são consideradas informações da CU pai, tais como a direção do melhor frame de referência,

diferença entre o MV predito e o de melhor custo, e a direção da predição (Unidirecional L0, L1 ou Bidirecional). A versão do VTM utilizada foi a 1.0, na configuração *Low Delay B*. O trabalho obteve média de 34,27% de redução do tempo total de codificação e 40,79% de redução do tempo de codificação da ME. O resultado para o BDBR é de 0,49%.

Em (Park; Kang, 2019), um método rápido para Estimação de Movimento *Affine* (AME) é proposto, usando recursos extraídos de uma análise estatística para ignorar processos AME redundantes. A solução é dividida em duas etapas. Primeiramente, a partir da análise do melhor modo de predição de um bloco pai é definido se todo o processo AME será ignorado ou não para o bloco atual. Posteriormente, caso o processo AME não seja ignorado, é verificada a direção da predição do melhor quadro de referência obtido na Estimação de Movimento convencional (Unidirecional e Bidirecional). Se o menor custo for obtido por um quadro de referência da predição Unidirecional na Lista 0, então este quadro é usado como valor máximo para o intervalo dos quadros de referência usados no processo AME. Os resultados experimentais demonstram redução de 37% no tempo de codificação da etapa *Affine*, 5% no tempo total de codificação, com uma perda de eficiência de codificação (BDBR) de 0,1%.

O trabalho de Tang et al. (2019) propõe uma otimização para o particionamento QTMT tanto na etapa intraquadro quanto na interquadros. Para a intraquadro é utilizado o algoritmo *Canny*, detector de bordas, em nível de bloco para ignorar os modos de particionamento vertical ou horizontal, realizando o encerramento prévio dos mesmos. Já para a interquadros é utilizado o método de diferença de três quadros para determinar se o bloco atual contém um objeto em movimento e a partição pode ser encerrada antecipadamente. Nesse caso, é calculada uma taxa a partir das diferenças encontradas entre o bloco nos diferentes quadros de referência. Se essa taxa for maior que 0,95, indicando que no bloco não houve movimento, o particionamento interquadros será ignorado para o mesmo. Caso contrário, o algoritmo *Canny* será utilizado para tomar a decisão. Os experimentos utilizaram o VTM 4.0, na configuração *Random Access*. Considerando somente a proposta para o particionamento da interquadros, o trabalho obteve média de 31,43% de redução no tempo total de codificação e BDBR de 1,34%.

Em Li et al. (2020) é apresentado um método para acelerar o modo InterIMV. O método ignora processos redundantes e é baseado na verificação *bottom-up* do melhor nível de profundidade e na relação taxa-distorção temporal para a CU. Primeiramente, o método verifica o modo InterIMV desde a profundidade mais baixa até as profundidades superiores e encontra a profundidade adequada para a área atual. Assim, a verificação do InterIMV em profundidades acima da profundidade adequada é removida sem trazer muitos danos à qualidade. Posteriormente, para cada CU, a proporção da taxa no custo RD do melhor modo temporal é usada para julgar se deve-se

Tabela 5 – Classificação dos trabalhos do VVC - Análise Estatística/Outros

ID	Trabalho	Versão VTM	Redução de Tempo (Total e Etapa)	BDBR	Técnica Utilizada	Foco da Aplicação
1	Pan et al. (2019)	1.0	34,27% 40,79%	0,49%	Análise Estatística	ME
2	Park; Kang (2019)	3.0	5,00% 37,00%	0,10%	Análise Estatística	*Affine* ME
3	Tang et al. (2019)	4.0	31,43% -	1,34%	Diferença de três *frames*	Particionamento
4	Li et al. (2020)	6.0	10,10% 79,78%	0,37%	Análise Estatística	InterIMV
5	Ren; He; Cui (2020)	7.0	6,00% 30,00%	0,21%	Análise Estatística	*Affine* ME
6	Guan; Sun (2021)	11.0	15,50% 12,50%	0,54%	Análise Estatística	Particionamento, Modo e Precisão
7	Jung; Jun (2021)	10.0	5,00% 33,00%	0,04%	Análise Estatística	*Affine* ME
8	Kuang et al. (2022)	14.0	40,46% -	1,08%	Análise Estatística	Particionamento
9	Li et al. (2022)	7.0	10,11% 40,85%	0,16%	Análise Estatística	*Affine* ME
10	Li; Luo; Zhu (2022)	6.0	23,19% -	0,97%	Análise Estatística	Particionamento
11	Wang; Yang (2022)	14.0	48,85% -	0,17%	Busca Hexagonal	Busca IME
12	Hong et al. (2023)	10.0	6,22% 24,79%	0,76%	*Canny Edge Detection*	*Affine* ME
13	Pejman et al. (2023)	14.0	10,60% -	0,88%	Análise Estatística	*Affine* ME
14	Shang et al. (2023)	11.0	40,08% -	1,56%	Análise Estatística	Particionamento
15	Yang et al. (2023)	16.0	- 13,79%	0,15%	Análise Estatística	*Geometric Partition Mode* (GPM)
16	Yu; Yang (2023)	VVEnc 1.0.0	0,23% -	0,12%	Análise Estatística	TZSearch

pular a verificação do InterIMV, uma vez que esta relação está fortemente associada à seleção do InterIMV. A solução foi implementada no VTM 6.0, com a configuração *Random Access*. Os resultados obtidos foram 10,10% de redução no tempo total de codificação e 79,78% no tempo da InterIMV. Já o BDBR obtido foi de BDBR 0,37% em média.

Já no trabalho (Ren; He; Cui, 2020) são propostos algoritmos baseados na simetria da descida de gradiente iterativo para a estimação rápida de movimento *Affine*. Basicamente, a solução limita o número máximo de iterações da predição Unidirecional e Bidirecional dentro da *Affine*. Para o modelo *Affine* rápido de quatro parâmetros, a Unidirecional itera até três vezes e a Bidirecional itera até cinco vezes. Já para o modelo *Affine* rápido de seis parâmetros, a Unidirecional itera até três vezes e a Bidirecional itera até quatro vezes. Além disso, é definida uma constante que acelera o processo de busca de gradiente, quando é identificada a mesma direção em mais de uma iteração. Dentre os resultados apresentados, o trabalho obteve perto de 30% de redução no tempo de codificação da etapa de compensação de movimento *Affine*, em média 6,65% de redução no tempo total de codificação, com um impacto no BDBR de 0,21%.

O artigo de (Guan; Sun, 2021) apresenta uma solução composta para o particionamento MTT, decisão de modo e ainda ajuste da precisão para a *Affine*. Na primeira etapa da solução, há a decisão do nível de divisão da CU atual na MTT, além de determinar se o melhor modo de predição da CU pai atual e da CU filho é consistente entre os mesmos. No modo de previsão de projeção, caso o gradiente de textura for menor que o valor médio do nível de luminância, ele será encerrado antecipadamente. Correspondentemente, quando a CU pai e a CU filha atuais estão ambas no modo de predição *Affine*, é realizada divisão adicional, o que garante que a descrição do primeiro plano de movimento seja mais precisa. Na segunda parte, os modos *Affine* de 4 e 6-parâmetros são otimizados em relação à precisão do MV, utilizando as precisões 1/4 e 1 para o modelo de 4-parâmetros, e as precisões 1/4 e 1/16 para o modelo de 6-parâmetros. O trabalho usa o VTM 11.0, nas configurações *Random Access* (RA) e *Low Delay P* (LDP). Quanto à redução no tempo total de codificação, foi alcançada média de 15,5% (RA) e 16% (LDP). Já a redução na etapa *Affine* foi de 12,5% (RA) e 5% (LDP). Os valores obtidos para BDBR-Y são de 0,54% (RA) e 0,84% (LDP).

Já no trabalho (Jung; Jun, 2021), é apresentado um método baseado na correlação de contexto entre blocos pai e atual para a predição *Affine*. O método possui três testes de terminação prévia. Primeiramente, toda a predição *Affine* é ignorada caso o bloco pai não tenha sido codificado como *Affine* e se os seus coeficientes transformados diferentes de zero não existirem, o que indica uma área estática ou movimento translacional. Em seguida, a predição *Affine* também é ignorada caso o quadro atual seja o maior da camada temporal (TL == 4) e se o bloco atual não foi codificado como

Affine Merge. Se nenhuma das condições for atendida, a predição *Affine* de quatro parâmetros é executada. Após isso, o modo *Affine* de seis parâmetros só é executado se o último teste verificar que o melhor modo para o bloco pai não foi o *Affine* de quatro parâmetros e se o bloco atual não teve o modo *Affine Merge* de quatro parâmetros como o menor custo até então. Os resultados obtidos mostram uma redução de 33% no tempo de codificação da etapa *Affine* e 5% no tempo total de codificação, com uma perda de eficiência de codificação (BDBR) de 0,04%.

Em Kuang et al. (2022) é definida uma solução unificada para o particionamento intraquadro e interquadros. A proposta utiliza informações históricas sobre diferentes caminhos de particionamento, sendo dividida em quatro partes. A primeira, chamada de *BT-direction-based QT and TT skip*, ignora o particionamento QT e TT na direção oposta, quando o particionamento BT segue apenas uma direção (horizontal ou vertical). As demais partes seguem uma lógica semelhante, mas são baseadas em posição, profundidade e histórico das divisões anteriores. Os experimentos foram realizados no VTM 14.0, com as configurações *Random Access*(RA), *Low Delay P* (LDP) e *All Intra* (AI). Considerando a solução combinada das 4 partes, os resultados obtidos para as configurações RA e LDP, respectivamente, são: 40,46% e 40,38% de redução do tempo total de codificação, com 1,08% e 1,18% de BDBR.

O trabalho de Li et al. (2022) apresenta um algoritmo rápido de Estimação de Movimento *Affine* (AME), baseado em informações de codificação de blocos adjacentes. É proposto um término antecipado da AME a partir da análise estatística da relação entre os modos *Affine* e *Skip* (quando não há movimento). Essa análise é realizada com base nos dados de codificação coletados de vídeos de diferentes resoluções. A cada bloco codificado, foi considerada a contagem de blocos adjacentes à esquerda, acima e co-localizados no quadro de referência, cujo modo *Skip* tenha sido o modo com menor custo. Os resultados obtidos no trabalho, em comparação com a versão original do VTM 7.0, atingem em média a redução de 10,11% e 40,85% no tempo de codificação total e da etapa AME, com um impacto de BDBR de 0,16%.

Já no trabalho (Li; Luo; Zhu, 2022) é proposto um método de particionamento rápido de blocos inter, baseado em um modelo de predição temporal. Na solução são apresentadas terminações prévias para particionamento QTMT, definidas a partir de análise estatística da distribuição de tamanhos de bloco ótimos na predição interquadros. Basicamente, são extraídas informações de particionamento do bloco ótimo da posição correspondente aos quadros codificados anteriormente e então é definido um modelo para prever o particionamento do bloco atual. Também é extraída a diferença do vetor de movimento (MVD) do bloco para definir se o particionamento QTMT pode ser encerrado antecipadamente, a fim de reduzir o erro cumulativo do modelo. Como resultados, o trabalho obteve em média 23,19% de redução no tempo de codificação, com perda na eficiência de codificação (BDBR) de 0,97%.

No artigo (Wang; Yang, 2022) o algoritmo TZSearch, usado na Estimação de Movimento, é alterado a fim de obter uma redução no tempo de codificação. As alterações realizadas no VTM 14.0 consistem em: utilizar o modelo de busca hexagonal substituindo o modelo de busca diamante do algoritmo padrão; definir um modelo de hexágono de rotação, com mais pontos iniciais de correspondência; e um limite razoável de término antecipado é definido para a Estimação de Movimento. Os resultados demonstram uma diminuição no tempo de codificação de 48,85% em média, com um impacto no BDBR de 0,17%.

Já em Hong et al. (2023), é proposta uma solução para a Estimação de Movimento *Affine*, utilizando o algoritmo *Canny Edge Detection* para a detecção rápida de bordas. O método apresentado tem duas etapas. Primeiramente, são definidas condições de teste para o conjunto de vetores de movimento *Affine* candidatos, com o intuito de ignorar codificações redundantes. A outra etapa consiste em utilizar o algoritmo de detecção de bordas na Estimação de Movimento *Affine*, com o objetivo de obter o gradiente da imagem de maneira mais rápida. Os resultados experimentais obtidos, em comparação com o VTM 10.0, são de 24,79% e 6,22% de redução no tempo de codificação da etapa *Affine* e total, respectivamente. Quanto ao impacto na eficiência de codificação, o trabalho atingiu a média de 0,76% em BDBR.

O trabalho de Pejman et al. (2023) analisa primeiramente a relação entre os custos RD AMVP, RD *Affine* e translacionais para uma série de vídeos disponíveis publicamente. Em seguida, é desenvolvido um método que acelera o processo ME, ignorando a *Affine* ME para determinados blocos. Esse método tem como base um valor limite que é calculado usando o modelo MLR (*Multiple Linear Regression*) sobre o custo TME RD. Esse limite pode ser ajustado de acordo com os requisitos do usuário sobre o compromisso entre o custo RD e a aceleração. Os experimentos utilizaram o VTM 14.0, na configuração Random Access (RA). O trabalho apresenta a média de redução do tempo total de codificação de 10,6%, com impacto de BDBR-Y de 0,88%.

No artigo de Shang et al. (2023) é utilizada Análise Estatística para acelerar o processo de particionamento e decisão de modos interquadros. Para a aplicação do método proposto, são extraídas informações durante o processo de codificação. Quanto ao particionamento, o trabalho propõe duas terminações prévias: quanto às divisões quaternárias, a partir de informações de área dos blocos vizinhos; quanto à direção das divisões ternárias, a partir da informação da melhor divisão binária temporária. Também é realizada uma estimativa do tipo de movimento do bloco, podendo este ser simples, médio e complexo. Com base nessa estimativa, mais modos interquadros são testados, conforme a complexidade aumenta. Dessa forma, em comparação com a versão 11.0 do VTM, o trabalho obteve 40,08% de redução no tempo de codificação total, com impacto de BDBR de 1,56%.

A solução presente em Yang et al. (2023) é focada em otimizar o modo GPM

(*Geometric Partition Mode*), a partir de um novo algoritmo rápido que detecta bordas baseado em texturas dinâmicas e estáticas. São utilizadas as técnicas *Histograms of Oriented Gradients* (HOG) e *Motion Boundary Histograms* (MBH) para detecção das bordas. As possibilidades de particionamento GPM são filtradas de acordo com as bordas detectadas pelo algoritmo. Além disso, todo o processo GPM é ignorado quando não existem bordas detectadas. A versão do VTM utilizada foi a 16.0, na configuração *Random Access*. Foi obtida a média de 13,79% de redução no tempo de codificação do modo GPM, com o BDBR de 0,15%.

Por fim, o artigo (Yu; Yang, 2023) apresenta uma otimização para o algoritmo TZ-Search. Especificamente quanto ao ponto de busca inicial, o trabalho considera que o modo Merge seleciona o MV final, por isso, neste método, o mesmo é complementado como um candidato. Quanto à otimização do padrão de busca, o método altera o mesmo a partir da terceira rodada de busca, considerando a direção do ponto de busca da rodada anterior com o menor custo. O software utilizado para testes foi o VVenC 1.0.0. Como resultado, o trabalho apresenta a média de 0,23% de redução no tempo total de codificação com BDBR de 0,12%.

6.5.2 Trabalhos que Utilizam *Machine Learning*

Nesta seção em específico, serão descritos os trabalhos de otimização com foco no VVC e que utilizam técnicas de *Machine Learning*. Conforme a organização da seção anterior, na Tabela 6 é apresentado um resumo dos trabalhos, com informações sobre a versão do VTM, os resultados de redução de tempo e BDBR, além da técnica específica utilizada e foco do trabalho. Assim como na tabela anterior, esses dados também estão organizados por ano e sobrenome, em ordem crescente. Novamente, os trabalhos que possuem apenas um resultado de redução de tempo estão representados com um "hífen"(-) na tabela. Nos próximos parágrafos, cada trabalho é apresentado de maneira mais detalhada.

No artigo (Kulupana; Kumar m; Blasi, 2021) é proposta uma otimização no esquema de particionamento da interquadros. O trabalho baseia-se em um novo esquema de seleção de *features*, que são inseridos em classificadores baseados em *Random Forests* e um critério de terminação prévia. Primeiramente, é definido o classificador QT/MTT, que decide se deve seguir o particionamento QT ou MTT. Depois, o classificador MTT, caso esse tipo de divisão for selecionado, decide entre testar divisões horizontais ou verticais, para BT e TT. Há também dois classificadores específicos para divisões TT horizontais e verticais, para decidir se devem ou não ser testadas. Por fim, foi definido um critério de terminação prévia TT, que define se serão avaliadas as partições ternárias para o bloco ou não. Para os classificadores, o trabalho utiliza um limite de confiança para a decisão, que, caso esteja abaixo do definido, ignora o resultado da classificação e realiza a avaliação do VTM convencional. O VTM

Tabela 6 – Classificação dos trabalhos do VVC - *Machine Learning*

ID	Trabalho	Versão VTM	Redução de Tempo (Total e Etapa)	BDBR	Técnica Utilizada	Foco da Aplicação
17	Kulupana et al. (2021)	8.0	41,00% -	1,33%	RF	Particionamento
18	Li et al. (2021)	6.0	- 66,79%	0,01%	DNN	*Motion Vector Prediction*
19	Pan et al. (2021)	6.0	30,63% -	3,18%	CNN	Particionamento e Merge
20	Yeo; Kim (2021)	11.0	11,53% -	1,01%	CNN	Particionamento
21	Duarte et al. (2022)	9.0	8,49% 46,94%	0,18%	RF	*Affine* ME
22	Li; Zhang; Yang (2022)	VVEnc 1.0.0	15,27% -	1,87%	RF	Particionamento
23	Liu et al. (2022)	11.0	19,34% -	0,83%	CNN	Particionamento
24	Lindino et al. (2022)	9.0	34,76% -	1,03%	RF	Particionamento
25	Shen; Yang; Wang (2022)	7.0	51,40% -	1,65%	DT	Particionamento
26	Tissier et al. (2022)	10.2	51,00% -	3,79%	CNN e DT	Particionamento
27	Xie et al. (2022)	VVEnc 1.0.0	7,71% -	1,48%	RF	Interquadros
28	Xu; He (2022)	13.0	37,30% -	1,41%	RF	Particionamento
29	Chan; Im (2023)	13.0	11,45% -	2,86%	RNN	Interquadros\ *Reference Frame*
30	Huang et al. (2023)	10.0	10,33% -	0,14%	Análise Estatística e DT	Unidirecional, Bidirecional e *Affine* ME
31	Lee; Jun (2023)	11.0	32,00% 33,00%	0,26%	LNN com MLP	*Bi-prediction with CU-level weight* (BCW)
32	Peng; Shen (2023)	16.0	44,50% -	1,94%	CNN e DT	Particionamento
33	Thanh et al. (2023)	18.0	33,34% -	0,31%	SVM	*TZSearch*
34	Sagrilo et al. (2023)	16.2	2,89% 19,64%	0,07%	RF	*Affine* ME

8.0 foi utilizado para os experimentos, com a configuração *Random Access*. Foram testados os critérios de confiança *low*, *medium* e *fast*, que obtiveram os respectivos resultados: 32%, 41% e 59% de redução no tempo total de codificação, e BDBR de 0,82%, 1,33% e 3,31%.

Li et al. (2021) propõe uma predição de Vetor de Movimento (MV) aprimorada baseada em Redes Neurais para a ME. Primeiramente, pesos dinâmicos são propostos no processo de seleção do melhor MVP para a ferramenta AMVP. Em segundo lugar, é definido um modelo de predição de MV baseado em Rede Neural Profunda (*Deep Neural Network*). Por fim, o modelo é aplicado no VVC para adquirir um MVP mais preciso e diminuir a complexidade da codificação. Os experimentos utilizaram o VTM 6.0 na configuração *Low Delay P* (LDP). O trabalho obteve média de 66,79% de redução do tempo de codificação da etapa AMVP e 5,5% da etapa *Motion Search*, com um BDBR de 0,01%.

Em Pan et al. (2021) é proposta uma *Multi-information fusion* CNN (MF-CNN) para finalização prévia do particionamento QTMT, que usa um conjunto de informações multi-domínio. Esse conjunto consiste em informações sobre o componente de luminância, resíduos e campo de movimento Bidirecional da CU atual. Também é proposta uma decisão de modo *Merge* com base nos resíduos da codificação e confiança da MF-CNN. O trabalho foi aplicado no VTM 6.0, com a configuração RA. A média de redução do tempo total de codificação ficou em 30,53%, para a proposta completa, e 24,83% para a otimização do particionamento. Já o BDBR obtido foi de 3,18% e 2,52%, respectivamente.

O trabalho (Yeo; Kim, 2021) apresenta uma solução para redução do tempo de codificação da etapa interquadros, abordando o problema do particionamento dos blocos. Para a solução, é introduzida uma rede neural convolucional (CNN) chamada de *Multi-level tree architecture* (MLT-CNN), sendo que os blocos 128×128 são as entradas da rede. Os resultados experimentais demonstram uma redução do tempo de codificação de 11,53% em média, com uma perda de eficiência de codificação de 1,01%. Para os experimentos, foram utilizados cinco QPs 22, 27, 32, 37 e 42, três vídeos da classe A (resolução 4K UHD), cinco vídeos da classe B (resolução Full HD), quatro vídeos da classe C (resolução WVGA), quatro vídeos da classe D (resolução VGA) e quatro vídeos da classe F (resolução HD). Os melhores resultados foram obtidos para os vídeos da classe A e B, de maior resolução, reduzindo de 9,81% até 26,14% do tempo de codificação total com aumento entre 0,95% e 3,28% no BDBR.

O artigo de Duarte et al. (2022) é originado a partir da dissertação de mestrado presente em (Gonçalves, 2021). Esse trabalho define um esquema rápido para o modo *Affine* baseado no modelo de Aprendizado de Máquina de Florestas Aleatórias (*Random Forests*). A solução inclui dois testes de terminação prévia, gerados a partir dos modelos treinados: um antes de executar a etapa *Affine* de quatro parâmetros e

outro antes de executar a etapa *Affine* de seis parâmetros. Os modelos foram treinados a partir de 33 *features* extraídas de 15 sequências de vídeo de *datasets* fora das CTCs. Para o treinamento, também foram considerados os diferentes tamanhos de bloco, sendo que foram agrupados, de acordo com a análise da importância da *features*, em três conjuntos para cada modelo, resultando em seis modelos de *Random Forests* treinados. O trabalho obteve uma redução de 46,94% no tempo de processamento da etapa *Affine* e 8,49% no tempo total de codificação, com uma perda de eficiência de codificação de 0,18%.

O trabalho apresentado em (Li; Zhang; Yang, 2022) é aplicado ao particionamento QTMT da interquadros, especificamente para os tamanhos de bloco 64×64, 32×32 e 16×16. A solução é baseada em *Random Forest* (RF) e é aplicada após a execução do modo *Merge*. As *features* extraídas levam em consideração informações do domínio temporal e também do conteúdo do bloco atual. Além disso, é calculado um coeficiente a partir de valores da etapa das Transformadas. Sendo assim, o modelo RF referente ao tamanho de bloco retorna se o mesmo deve ou não ser dividido. Os experimentos utilizaram o VVenC 1.0, com a configuração RA. Foi atingida a média de redução no tempo total de codificação de 15,27% e o BDBR de 1,87%.

Já o trabalho de Liu et al. (2022) utiliza CNN para acelerar o processo de particionamento na interquadros. A CNN opera em nível de CTU, sendo que a mesma é dividida em uma grade fixa de blocos 8×8. Cada bloco é então associado à informações de profundidade de particionamento pela CNN. De maneira específica, o método é aplicado para decidir sobre ignorar os testes para as divisões do tipo MT e NS (sem divisão), quando as mesmas provavelmente não seriam selecionadas. Para isso, a partir da saída da CNN (mapa de profundidade) é calculada a média de profundidade predita, sendo a mesma comparada com um limite subtraído da profundidade do bloco atual. Esse limite é flexível, podendo ser estabelecido conforme a necessidade de redução de complexidade. Para os experimentos foi utilizado o VTM 11.0, na configuração RA. Os resultados para redução no tempo total de codificação e BDBR foram de 19,34% e 0,83%, respectivamente.

No artigo de Lindino et al. (2022) é proposto um esquema de decisão rápida para o particionamento QTMT, baseado em Aprendizado de Máquina. A partir de dados coletados durante a codificação de vídeos selecionados, quatorze modelos de Florestas Aleatórias foram treinados e aplicados para decidir quando os particionamentos horizontal e vertical são necessários para determinado bloco. Para o treinamento, os dados foram agrupados de acordo com os tamanhos de bloco. Como resultado, em comparação com o VTM 9.0, o trabalho obteve em média a redução de 34,76% do tempo de codificação e aumento de 1,03% no BDBR.

Outro trabalho, descrito em (Shen; Yang; Wang, 2022), descreve um *framework* multinível para o particionamento QTMT, dividindo-o em 6 problemas distintos com

duas classificações possíveis. A técnica utilizada para cada nível é a Árvore de Decisão. Com uma prioridade maior, o classificador para partições quadráticas (QT) é aplicado. Na sequência, para diminuir a complexidade, um classificador para a direção do particionamento multi tipo (MT) é aplicado, prevendo a divisão horizontal ou vertical. Cada uma das duas possibilidades possui outros dois subclassificadores para prever as divisões binárias ou ternárias. Além disso, o *framework* adota um mecanismo de proteção contra classificação incorreta, utilizando uma estratégia menos rígida em cada classificador e executando em paralelo a verificação dos subclassificadores binário e ternário. Os resultados obtidos a partir da comparação com o VTM 7.0, apontam um acréscimo de BDBR de 1,65% e uma redução de 51,40% no tempo total de codificação.

A proposta de Tissier et al. (2022) também é voltada para o Particionamento. Primeiramente, a partir de dados de luminância da CTU atual e das duas CTUs de referência (com menor custo RD) é utilizada uma CNN para gerar um vetor de informações espaciais, que corresponde ao particionamento da CTU. Após isso, várias Árvores de Decisão, chamadas *Multi-class Classifiers* que são baseadas no *framework LightGBM* (LBGM) (KE et al. 2017), determinam a probabilidade de cada divisão possível, a partir das informações do bloco processado. A tomada de decisão sobre quais divisões serão efetivamente testadas pelo VVC é feita com base em uma configuração adaptável, que determina quantas das divisões com maior probabilidade serão consideradas. São três configurações distintas, C1, C2 e C3 que visam representar diferentes compromissos de redução de complexidade e perda de eficiência de codificação. Em C1 e C3, por exemplo, são testadas as 4 e as 3 divisões com melhores probabilidades para todos os tamanhos de bloco, respectivamente. Para os experimentos foi utilizado o VTM 10.2, com a configuração RA. Os resultados obtidos para a redução no tempo total de codificação ficaram entre 31,3% até 51%. Já o BDBR obtido foi de 1,65% até 3,79%.

O trabalho de Xie et al. (2022) utiliza três modelos *Random Forests* treinados *offline* para cada um dos três tamanhos de blocos: 128×128, 64×64 e 32×32. A proposta realiza a terminação prévia da etapa interquadros. As *features* são extraídas durante o processo do modo *Merge* e aplicadas aos três modelos que retornam, para cada tamanho de bloco, se a predição interquadros deve ser finalizada. Além das *features* obtidas no modo *Merge*, também são calculadas *features* a partir de informações do bloco temporalmente co-localizado e de quatro blocos adjacentes (esquerda, superior esquerdo, superior e superior direito) ao bloco atual. A proposta foi testada no VVenC 1.0.0, na configuração RA e atindiu a média de 7,71% de redução no tempo total de codificação, com BDBR de 1,48%.

O artigo (Xu; He, 2022) também apresenta uma solução de otimização para o particionamento QTMT, porém, utilizando a técnica de Aprendizado de Máquina de

Florestas Aleatórias. Para o treinamento do modelo, são extraídos e calculados alguns dados a partir da codificação de blocos elegíveis, tais como, variância, diferença de adjacência, vetor de movimento, etc. Além disso, também é extraído o particionamento ideal após a codificação de cada CTU (*Coding Tree Unit*). Os autores alteraram a saída do modelo de Florestas Aleatórias, de modo que o mesmo retorna uma matriz de probabilidade de predição de diferentes categorias. Dessa forma, também é adicionado um mecanismo de intervalo de risco, definido como a probabilidade de 0,65, em função da maior acurácia obtida. Caso a probabilidade resultante do modelo para determinada divisão seja maior que o valor 0,65, altera-se a pilha de execução do particionamento. Dessa forma, o trabalho alcança resultados de 37,30% no tempo total de codificação, com aumento de 1,41% no BDBR.

Já o trabalho (Chan; Im, 2023) utiliza uma Rede Neural Recorrente (*Recurrent Neural Network* - RNN) para a geração de um quadro de referência, chamado de NF-Frame, que é utilizado na predição interquadros. A rede pré-treinada YOLOv5 é utilizada para obter o rastreamento de objetos dentro de um quadro. Em seguida, esses objetos e suas informações de movimento são usadas para prever uma posição futura que será apresentada no NF-Frame proposto. Sendo assim, a busca de movimento para esses objetos pode ser determinada de maneira mais rápida, equanto as demais partes do quadro seguem a execução padrão do codificador. A partir da comparação com o VTM 13.0, o trabalho obteve um impacto de BDBR de 2,86% e melhora de 11,45% no tempo de codificação, na configuração *Random Access*. Já na configuração *Low Delay*, os resultados foram de 0,61% e 2,89% em BDBR e tempo de codificação, respectivamente.

O artigo de Huang et al. (2023) também apresenta uma solução de otimização para a Estimação de Movimento Convencional e *Affine*. A solução abrange três estágios principais. Primeiro, a partir de uma análise estatística, é definido um teste de terminação prévia da etapa AME caso o modo de codificação ótimo do bloco pai tenha sido a ME convencional. Especificamente na AME, são aplicados três otimizações: terminação prévia do processo AME, caso os dois vetores de movimento gerados no modelo *Affine* de quatro parâmetros sejam paralelos; terminação prévia do processo de iteração da busca de gradiente, a partir de um limite calculado com base na SAD (*Sum of Absolute Difference*), média da matriz de erro, altura e largura do bloco; além disso, um esquema rápido para a busca de granularidade fina é aplicado com base na comparação do custo a cada duas direções. Por fim, um modelo de árvore de decisão é treinado a partir de 8 *features* referentes a textura, *zoom*, gradiente e energia, para decidir qual o modo de predição, convencional ou *Affine*, será testado. Como resultado de BDBR o trabalho obteve 0,14% na configuração *Random Acces* e 0,12% na *Low Delay*. Já a redução de tempo foi de 10,33% e 10,2%, respectivamente.

Já o artigo (Lee; Jun, 2023) apresenta uma solução de decisão rápida para a fer-

ramenta BCW (*Bi-prediction with CU-level Weights*), presente na etapa Bidirecional da Estimação de Movimento. Os autores utilizam uma Rede Neural Leve (Lightweight Neural Network - LNN) com MLP (*multilayer perceptron*) treinada a partir de quatro categorias de *features*: correlação do BCW com os blocos pai e vizinho; informações sobre Parâmetro de Quantização (QP); proporções quanto ao tamanho do bloco e informações de outras ferramentas de codificação. A arquitetura denominada BCW-MLP foi implementada para determinar quando o modo BCW deve ser executado. Os resultados experimentais, na configuração *Random Access*, demonstraram redução de 33% e 32% no tempo do BCW e total, respectivamente. Já o BDBR em média atingiu 0,26%. Na configuração *Low Delay* a redução foi de 49% e 34% no tempo BCW e total, com impacto de 0,06% em BDBR.

Em (Peng; Shen, 2023) é proposta a utilização de um *framework* que combina classificação e predição para processar diferentes CTUs através de redes com capacidades adequadas. O objetivo é acelerar o particionamento QTMT na interquadros. Este trabalho também considera um mapa de homogeneidade de particionamento (*Partition Homogeneity Map* - PHM) das CTUs, mapeando a estrutura de particionamento para cada sub-bloco 8×8. Primeiramente, uma CNN é utilizada para classificar uma CTU em simples, moderada ou complexa, de acordo com o PHM. Assim, de acordo com a classificação, uma das três CNNs treinadas especificamente para o tipo de CTU é utilizada para prever o PHM. Após isso, um modelo que utiliza Árvore de Decisão define se haverá particionamento ou não, tomando como base, além do PHM, as informações da codificação do modo *Affine Merge*. Os experimentos foram realizados utilizando o VTM 16.0, com a configuração RA. Foi obtida a média de 44,5% de redução do tempo total de codificação, com BDBR de 1,94%.

O trabalho de Thanh et al. (2023) é focado em otimizar o TZSearch especificamente para vídeos de vigilância. Primeiro, os autores estudam e definem uma lista de características espaciais e temporais que indicam as características de movimento e textura do vídeo de vigilância. Esses recursos são usados junto com um algoritmo de aprendizado de máquina para atribuir adequadamente um intervalo de pesquisa para a pesquisa de movimento VVC. Em segundo lugar, para reduzir os pontos de busca, os autores propõem uma busca adaptativa na Zona de Teste (TZ) na qual as etapas da TZ são encerradas antecipadamente seguindo a variação das características espaço-temporais. O trabalho utilizou o VTM 18.0. Na configuração RA, obteve a média de 33,34% de redução do tempo total de codificação, com BDBR de 0,31%.

Por fim, no artigo de Sagrilo et al. (2023), é apresentada outra solução voltada para a *Affine* ME, utilizando a técnica de Aprendizado de Máquina *Random Forests* (RF). Foram definidos e treinados doze modelos de RF, um para cada tamanho de bloco suportado na *Affine*. Os modelos foram treinados a partir de *features* extraídas do bloco que está sendo codificado, do bloco pai, e vizinhos à esquerda e acima,

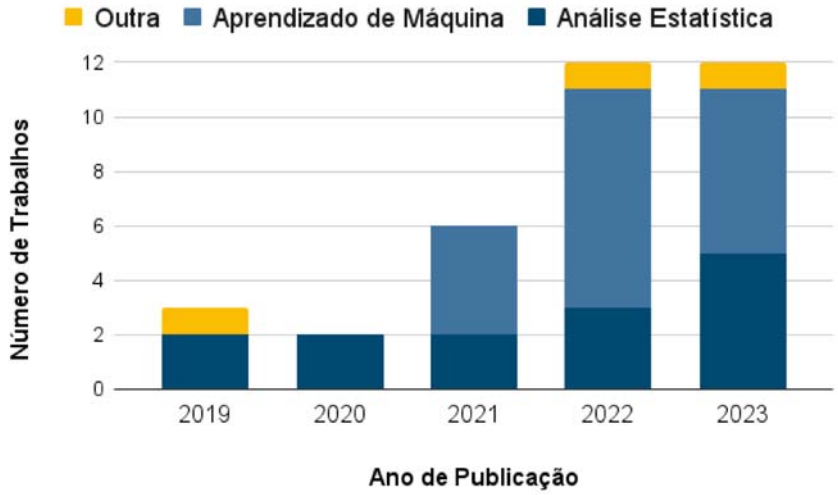
previamente codificados. Algumas dessas *features* se referem ao custo RD (*Rate-Distortion*), posição do bloco no quadro, melhor modo de predição, precisão do vetor de movimento (IMV), etc. Quanto aos resultados experimentais, em comparação com o VTM 16.2 na configuração *Random Access*, o trabalho obteve redução de 19,64% e 2,89% de tempo de codificação da AME e total, com BDBR de 0,07%.

6.5.3 Discussão dos Resultados - VVC

De maneira geral, ao analisar as técnicas utilizadas pelos trabalhos, percebe-se um equilíbrio entre o uso de Análise Estatística e Aprendizado de Máquina, com 13 e 18 trabalhos, respectivamente. Além disso, três trabalhos utilizam outras técnicas. Quanto ao foco da otimização, a maioria é para o particionamento, com 15 trabalhos, e *Affine* ME, com 8 trabalhos. Já a ME e o TZSearch possuem 2 trabalhos cada. As demais ferramentas, tais como *Merge*, GPM, BCW, MV *Prediction*, InterIMV, IME e interquadros, foram foco de apenas um trabalho cada.

Nesse sentido, a Figura 39 demonstra a evolução das técnicas utilizadas ao longo do tempo. Percebe-se que o uso de Análise Estatística se mantém estável entre 2019 e 2021, crescendo significativamente em 2023. Já as técnicas de Aprendizado de Máquina aparecem só em 2021, mas na maioria dos trabalhos. Também é possível notar o crescimento duplicado desses trabalhos em 2022 e uma leve diminuição em 2023. Esses dados demonstram o crescente interesse em técnicas mais robustas, seguindo a tendência de utilização de Aprendizado de Máquina nas soluções de otimização, visando também o menor impacto na eficiência de codificação.

Figura 39 – Número de Trabalhos por Ano e Técnica Utilizada

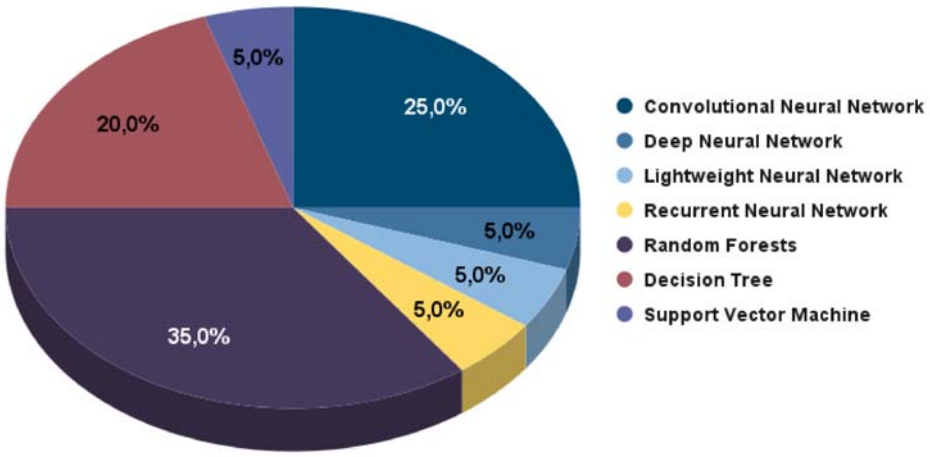


Fonte: Elaborada pelo autor.

De maneira específica, pode-se analisar as diferentes técnicas de Aprendizado de Máquina que foram utilizadas nos trabalhos, conforme ilustrado na Figura 40. É possível notar que as técnicas mais utilizadas são *Random Forests*, *Convolutional Neural Network* (CNN) e *Decision Tree*. Entretanto, a utilização de outros tipos de

Redes Neurais aparece em mais três trabalhos, o que aumenta o percentual para 40%. Também é importante ressaltar que o gráfico considera que nos trabalhos Tissier et al. (2022) e Peng; Shen (2023) são utilizadas tanto a técnica CNN quanto *Decision Tree*.

Figura 40 – Distribuição das Técnicas de Aprendizado de Máquina



Fonte: Elaborada pelo autor.

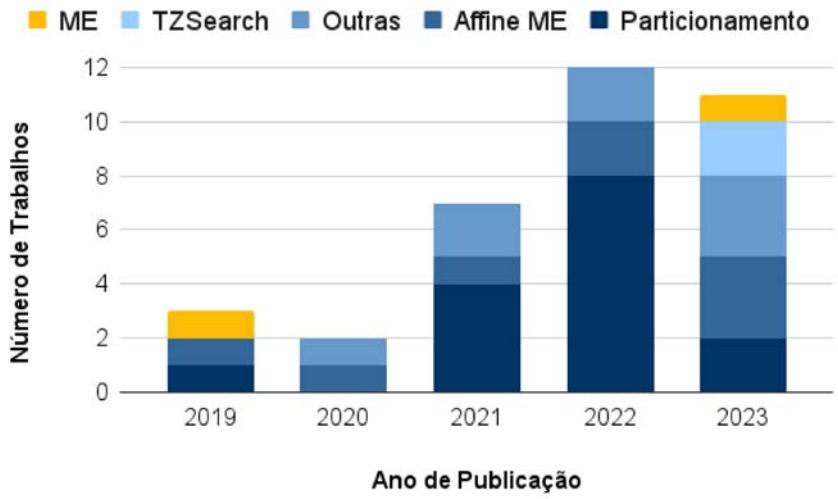
Já na Figura 41 é ilustrada a evolução das ferramentas foco de otimizações ao longo dos anos. Quanto ao Particionamento, é possível perceber o crescente interesse, principalmente nos anos de 2021 e 2022, com uma diminuição considerável no ano de 2023. Também é notável que a *Affine* ME aparece em todos os anos, com um leve crescimento em 2022 e 2023. O interesse em outras ferramentas da interquadros é praticamente estável entre 2020 e 2023, com leve aumento nos três últimos anos. O TZSearch, por sua vez, aparece como foco de trabalhos apenas em 2023. Por fim, o foco na ME como um todo surge apenas duas vezes, uma em 2019 e outra em 2023.

Outra análise pode ser realizada a partir das Figuras 42 e 43, que apresentam a relação entre a redução de tempo de codificação total e o BDBR obtido nos trabalhos que usam Análise Estatística e Aprendizado de Máquina, respectivamente. Apenas dois trabalhos, indicados com *, têm a representação da redução do tempo da etapa específica, por não fornecerem a informação quanto ao tempo total.

Para os trabalhos que usam Análise Estatística, presentes no gráfico 42, é possível notar que a maioria atingiu menos de 20% de redução no tempo com impacto de BDBR menor que 1%. Outros trabalhos apresentam maior redução no tempo, porém com impacto maior no BDBR (em torno de 1% e 1,5%). Entretanto, destacam-se os trabalhos de Pan et al. (2019) e Wang; Yang (2022), que atingiram redução no tempo de codificação de 34,37% e 48,85%, respectivamente, com menos de 0,5% de impacto de BDBR.

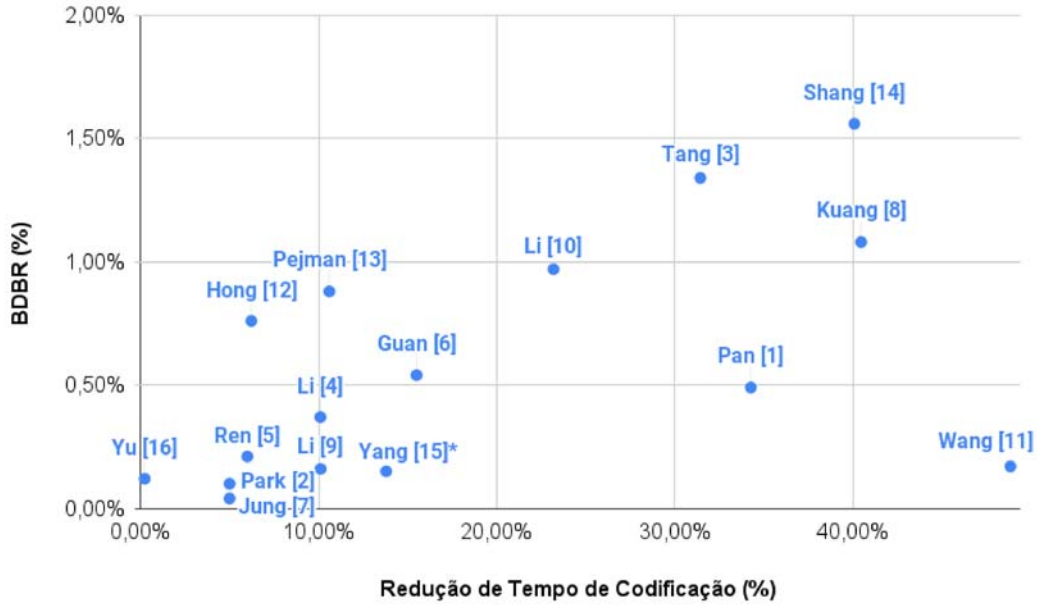
Por outro lado, na Figura 43, nota-se uma distribuição mais variada dos trabalhos.

Figura 41 – Número de Trabalhos por Ano e Foco da Otimização



Fonte: Elaborada pelo autor.

Figura 42 – Distribuição (Tempo x BDBR) dos trabalhos que utilizam Análise Estatística

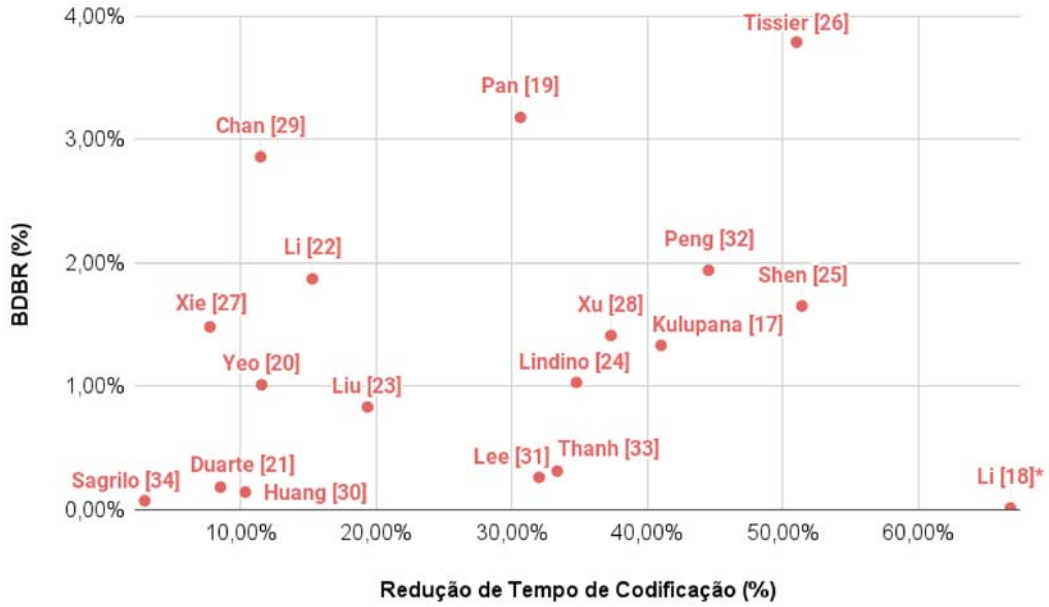


Fonte: Elaborada pelo autor.

Quanto à redução no tempo de codificação, 8 trabalhos atingiram até 20% e os outros 10 trabalhos ficaram na faixa de 30% a 60%. Quanto ao BDBR, metade dos trabalhos chegou até 1,03% e a outra metade foi de 1,33% até 3,79%. Dentre os melhores resultados, destacam-se os trabalhos Lee; Jun (2023) e Thanh et al. (2023), com 32% e 33,34% de redução de tempo de codificação, com impacto de BDBR de 0,26% e 0,31%, respectivamente. Outros 5 trabalhos (Lindino et al. (2022), Xu; He (2022), Kulupana; Kumar m; Blasi (2021), Peng; Shen (2023) e Shen; Yang; Wang (2022)) também apresentam resultados significativos de redução de tempo (entre 34,76% e 51,4%), porém o BDBR fica entre 1,03% e 1,94%. Além disso, o trabalho Li et al. (2021) se destaca com redução significativa (66,79%) no tempo de codificação da

etapa *MV Prediction*, com baixo BDBR (apenas 0,01%).

Figura 43 – Distribuição (Tempo x BDBR) dos trabalhos que utilizam Aprendizado de Máquina



Fonte: Elaborada pelo autor.

A partir das descrições e das análises sobre os trabalhos, é possível responder às questões de pesquisa 3 e 4. A questão 3 se refere a quais estratégias de Aprendizado de Máquina são utilizadas nos trabalhos. Nesse sentido, tanto a Tabela 6 quanto a Figura 40, demonstram que as Redes Neurais, principalmente CNN, são as mais utilizadas. Além disso, técnicas como *Random Forests* e *Decision Tree* também se destacam.

Já a questão 4 foca em quais as oportunidades de pesquisa ainda estão abertas na predição interquadros do padrão VVC. A partir das Tabelas 5 e 6 e da Figura 41, percebe-se o maior interesse na otimização da ferramenta *Affine* e do Particionamento QTMT, possivelmente pelo fato de que ambos são novidade no VVC e também apresentam alto custo computacional. Entretanto, é possível notar que há poucos trabalhos que focam em toda a ME, ou seja, abordando as etapas Unidirecional, Bidirecional e *Affine*. É importante destacar que a ME ainda é uma etapa custosa no processo de codificação de vídeos, sendo naturalmente foco de soluções de otimização. Além disso, a Unidirecional usa o TZSearch na busca inteira, que, apesar de ser um algoritmo rápido, ainda apresenta alta complexidade na sua execução. O TZSearch é abordado em apenas dois trabalhos, o que indica uma oportunidade de desenvolvimento de soluções de otimização. Já a Bidirecional contém novas ferramentas, como SMVD, BCW e BDOF, o que pode ocasionar um maior tempo de codificação. Diante disso, para além do Particionamento interquadros e da *Affine* ME, a etapa Bidirecional e suas respectivas novas ferramentas também surgem como alternativas para trabalhos de otimização.

7 HEURÍSTICA CONFIGURÁVEL PARA REDUÇÃO DE CUSTO COMPUTACIONAL DA PREDIÇÃO INTERQUADROS

Este capítulo descreve o primeiro resultado gerado a partir do desenvolvimento desta tese. É uma heurística para redução de consumo de energia configurável e fácil de implementar em hardware para a predição interquadros do codificador VVC, incluindo as etapas Unidirecional, Bidirecional e *Affine* da Estimação de Movimento. Essa heurística foi publicada em Loose et al. (2022).

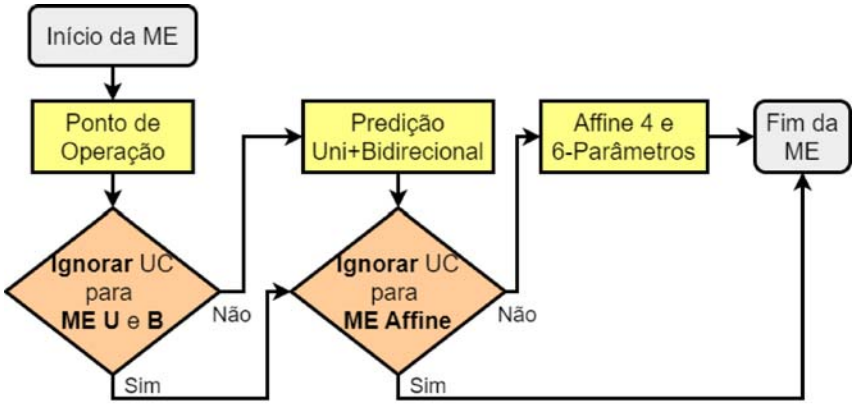
Esta heurística é baseada em restrições adaptativas das CUs disponíveis, de acordo com a resolução do vídeo e com o ponto de operação selecionado. Três pontos de operação estão disponíveis, considerando o nível da redução do consumo de energia: Mínimo (Min), Intermediário (Int) e Máximo (Max). Naturalmente, quanto maior é a redução no consumo de energia, maiores são as perdas de eficiência de codificação.

O fluxograma da heurística é mostrado na Figura 44. Primeiramente, o ponto de operação precisa ser configurado. Esta configuração pode ser feita em tempo de projeto ou em tempo de execução. O primeiro caso é útil quando a meta de consumo de energia é definida antes da implementação do hardware. O segundo caso é útil quando uma solução adaptativa é necessária, por exemplo, considerando o nível de bateria de um dispositivo móvel. Neste segundo caso, as unidades de processamento podem ser desligadas de forma adaptativa de acordo com a definição do ponto de operação.

O ponto de operação e a resolução do vídeo definem quais tamanhos de CUs são permitidos para cada ferramenta da Estimação de Movimento. A primeira avaliação define se o tamanho de CU deve ser ignorado para a ME Unidirecional e Bidirecional (*ME U e B*, na Figura 44). Se o tamanho de CU é ignorado, a ME Unidirecional e Bidirecional não é aplicada para aquele tamanho de CU e ambas as unidades podem ser desligadas. Por outro lado, se o tamanho de CU não é ignorado, a ME Unidirecional e Bidirecional são executadas sem restrições. Depois disso, a segunda avaliação define se o tamanho de CU é considerado para a ME *Affine*. Se o tamanho de CU é ignorado, a ME *Affine* pode ser desligada; caso contrário, a ME *Affine* é aplicada sem

restrições.

Figura 44 – Solução proposta aplicada à predição interquadros



Fonte: Elaborada pelo autor.

Observando a Figura 44, pode-se concluir que esta heurística é fácil de ser implementada em *hardware*. Em tempo de projeto, as unidades de processamento ME podem ser projetadas considerando que alguns tamanhos de CU não serão suportados, e a área do *hardware* pode ser economizada, além de energia. Em tempo de execução, duas comparações precisam ser implementadas para definir quais unidades serão desligadas para economizar energia de acordo com o ponto de operação e a resolução do vídeo.

A heurística foi definida em dois passos principais: extração das *features* e definição dos tamanhos de CUs ignorados. Quinze vídeos do conjunto de dados UVG (Mercat; Viitanen; Vanne, 2020) e do conjunto de dados NETVC (Daede; Norkin; Brailovskiy, 2019) foram utilizados nos experimentos para definirem a heurística. Estes quinze vídeos foram selecionados considerando os seus valores para SI (Informação Espacial) e TI (Informação Temporal) (ITU-T, 2021), garantindo que eles possuem características distintas. As sequências das Condições de Teste Comum (CTC) (Bossen et al., 2020) do grupo JVET foram reservadas para avaliar os resultados, garantindo assim que a solução foi avaliada sem viés. Todas as sequências foram codificadas sob as configurações descritas na Seção 7.3.

7.1 Extração das *Features*

As *features* consideradas foram o tempo de codificação e a contagem de cada tamanho de CU como o bloco escolhido (ou seja, se um tamanho de CU levou ao melhor custo de RD para essa etapa (BROSS et al. 2021)). O tempo de codificação foi utilizado como uma aproximação da métrica do consumo de energia. A ideia principal é utilizar estas *features* para encontrar a melhor combinação de tamanhos de CUs para alcançar as reduções do tempo de codificação com o menor impacto na eficiência

de codificação. A premissa é que quanto menor é o uso de um determinado tamanho de CU, menor é o impacto de ignorar este tamanho de CU no processo de codificação. Estas *features* foram extraídas considerando as ferramentas da ME em dois grupos: (i) ME Unidirecional e Bidirecional e (ii) ME *Affine* com quatro e seis parâmetros. Este agrupamento foi feito considerando a similaridade das operações necessárias para estes passos do codificador.

Os vídeos utilizados para extrair as *features* foram divididos em três resoluções: HD (1280×720), FHD (1920×1080) e 4K UHD (3840×2160). As sequências HD foram *Dark*, *Netflix DrivingPOV*, *Vidyo4*, *Netflix DinnerScene*, e *KristenAndSara*. As sequências FHD foram *Netflix TunnelFlags*, *Jockey*, *Beauty*, *Touchdown Pass*, e *Rush Field Cuts*. Por fim, as sequências 4K UHD foram *ToddlerFountain*, *SunBath*, *Lips*, *BuildingHall2* e *Netflix Dancers*. Cada sequência foi codificada quatro vezes, considerando os quatro valores dos parâmetros de quantização (QP) definidos pelas CTCs (22, 27, 32 e 37). Foram codificados os primeiros 32 quadros de cada vídeo, usando a configuração *Random-Access* (RA) (Bossen et al., 2020). Para cada vídeo foram calculadas as médias resultantes para as duas *features*.

7.2 Definição dos Tamanhos de CU Ignorados

Na etapa de análise de dados, um algoritmo de força-bruta foi desenvolvido para testar todas as combinações entre os 27 tamanhos de CU permitidos para a ME Unidirecional e Bidirecional, e entre os 12 tamanhos de CU para ME *Affine*. Para cada combinação de tamanhos de CU e para cada resolução, o algoritmo somou seus respectivos percentuais de tempo de codificação e contagem. Para cada meta, a combinação de CU com a menor contagem que atingiu ou ultrapassou a meta foi selecionada para ser ignorada, considerando os dois grupos de ferramentas ME (como mostrado na Tabela 7 e Tabela 8). A suposição básica é que, quanto menor é a contagem de uma CU (ou um grupo de CUs) ignorado, menores serão as perdas de eficiência de codificação, pois menos vezes a mesma foi escolhida como tendo o menor custo RD.

Aqui, a heurística foi avaliada usando três pontos de operação, chamados Mínimo (Min), Intermediário (Int), e Máximo (Max), respectivamente, em referência à redução de energia alcançada por cada meta de redução de tempo de codificação. Neste caso, as metas foram definidas como 5%, 15%, e 20% para os dois grupos de ferramentas ME, mas outras metas também podem ser definidas para permitir um nível maior de configurabilidade ou um nível maior de economia de energia.

As Tabelas 7 e 8 mostram os tamanhos de CU ignorados para cada ponto de operação (OP) e para cada resolução suportada para (i) ME Unidirecional e Bidirecional e (ii) ME *Affine*, respectivamente. Como discutido anteriormente, a heurística é especializada para três resoluções: HD, FHD e 4K UHD. Como pode-se concluir observando

as Tabelas 7 e 8, o número e os tipos de tamanhos de CUs ignorados variam de acordo com o ponto de operação, a resolução específica e a ferramenta ME. Isto ocorre exatamente porque a heurística foi definida através de uma avaliação exaustiva da melhor combinação de tamanhos de CU para cada caso.

Tabela 7 – Tamanhos de CUs ignoradas: ME Unidirecional e Bidirecional

OP	**Resolução**		
	HD	**FHD**	**4K UHD**
Min	4×16 4×64 8×8	4×16 4×64	4×16 8×8
Int	4×16 4×64 16×4 128×128	4×16 4×64 8×8 16×4 128×64	4×16 4×32 4×64 8×8 128×64
Max	4×8 8×4 4×16 8×8 4×32 16×4 4×64 32×4 128×128	4×8 8×4 4×16 8×8 4×32 16×4 32×4 128×128	4×8 8×4 4×16 8×8 4×64 16×4 128×128

Tabela 8 – Tamanhos de CUs Ignoradas: ME *Affine*

OP	**Resolução**		
	HD	**FHD**	**4K UHD**
Min	16×64	16×64	64×16
Int	16×64 128×64	16×32 64×128	128×128
Max	16×64 64×16 128×64	64×128 128×128	16×16 128×128

7.3 Resultados e Comparações da Heurística Proposta

A heurística desenvolvida com seus pontos de operação respectivos Min, Int, e Max foi implementada no *software* de referência VTM 14.0 para os dois grupos ME: (i) ME Unidirecional e Bidirecional e (ii) ME *Affine*. Esta implementação permitiu a avaliação experimental das reduções do tempo de codificação e de perdas da eficiência de codificação considerando os pontos de operação e as resoluções dos vídeos, para cada grupo de ferramentas da ME. Os experimentos foram realizados em um servidor com processador Intel Xeon E5-2650v4 2,20GHz com 48 GB de RAM. Oito sequências de teste foram selecionadas das CTCs do VVC (Bossen et al., 2020), divididas em duas sequências HD, três sequências FHD e três sequências 4K UHD. As

duas sequências HD da Classe F da CTC foram *SlideShow* e *SlideEditing*. As três sequências FHD foram *RitualDance*, *Cactus* e *BQTerrace* da Classe B da CTC. Por fim, as três sequências 4K UHD da Classe A da CTC também utilizadas foram *Tango2*, *FoodMarket4* e *CatRobot*. Como discutido anteriormente, todas estas sequências utilizadas para testar são diferentes daquelas utilizadas para a definição da heurística. Os primeiros 32 quadros de cada sequência foram codificados para os quatro QPs, também usando a configuração *Random Access* no VTM 14.0, como nos experimentos anteriores.

Também é apresentada uma estimativa da redução no consumo de energia por ponto de operação e resoluções dos vídeos da heurística proposta. Esta estimativa foi feita em um alto nível de abstração e sem um *hardware* específico em mente, mas considerando a adaptabilidade do tempo de execução (quando o ponto de operação é definido em tempo de execução). Dois casos foram considerados: (i) quando cada ferramenta da ME tem uma unidade operacional para processar cada tamanho de CU (totalmente paralelo) e (ii) quando cada ferramenta ME tem uma unidade operacional genérica que é usada para processar todos os tamanhos de CU (totalmente sequencial). No caso totalmente paralelo, cada tamanho de CU ignorado implica em desligar a sua unidade operacional respectiva, economizando energia. Na maneira totalmente sequencial, cada tamanho de CU ignorada implica em menos ciclos de uso para esta unidade genérica, novamente economizando energia. Então, o ganho de consumo de energia é diretamente proporcional ao número de CUs ignoradas independentemente do nível de paralelismo utilizado. A estimativa da redução de energia considerou que todas as quatro ferramentas da ME (Unidirecional, Bidirecional, e *Affine* com quatro ou seis parâmetros) consomem a mesma energia, por simplicidade. Desta forma, o ganho da Estimação de Movimento considera a porcentagem das CUs ignoradas em cada caso, em comparação com o número total de tamanhos de CU originalmente disponíveis no VVC: 27 para ME Unidirecional e Bidirecional, e 12 para cada ME *Affine*. Sabe-se que essa estimativa apresenta algumas fragilidades, principalmente porque ainda não considera que as ferramentas e os tamanhos de bloco podem consumir níveis diferentes de energia. Mesmo assim, essa estimativa é utilizada para fins de validação inicial da solução desenvolvida.

A Tabela 9 apresenta os resultados alcançados, incluindo a média da redução do tempo para ME Unidirecional e Bidirecional (TR_{UB}), a média da redução de tempo para ME *Affine* (TR_{AF}), a média de redução do tempo total de codificação (TR_{Total}), a estimativa da redução do consumo de energia ($EECR$), e as perdas totais da eficiência de codificação, usando a métrica Bjøntegaard-Delta bitrate ($BDBR$) (Bjøntegaard, 2001). Estes resultados presentes na Tabela 9 para cada resolução (Res) e cada ponto de operação (OP) são um percentual de ganhos ou perdas ao comparar a heurística proposta com o VTM original.

Tabela 9 – Resultados da redução de tempo, estimação da redução do consumo de energia e o aumento do BDBR

OP	Resolução	TR_{UB} (%)	TR_{AF} (%)	TR_{Total} (%)	EECR (%)	BDBR (%)
Min	HD	3,57	6,73	1,56	10,26	0,04
	FHD	10,56	4,70	1,24	7,69	0,19
	4K UHD	6,19	8,12	3,90	7,69	0,11
Int	HD	12,55	14,15	2,80	15,39	0,35
	FHD	14,50	10,23	2,34	17,95	0,25
	4K UHD	11,12	16,81	5,92	15,39	0,30
Max	HD	16,09	19,55	2,91	30,77	1,00
	FHD	26,50	14,75	2,72	25,64	0,90
	4K UHD	15,68	22,71	7,44	23,08	0,44

Os resultados na Tabela 9 mostram que, em todos os casos, a heurística apresentada alcançou economias importantes de tempo e energia com poucos impactos em termos de eficiência de codificação. Além disso, os resultados na Tabela 9 mostram que a heurística proposta é capaz de dimensionar a economia de energia de acordo com os pontos de operação. As melhores economias de energia foram alcançadas no ponto de operação Max em todos os casos, como esperado, alcançando mais que 30% de redução no consumo de energia para sequências HD, para um aumento de 1% no BDBR. Por outro lado, também conforme esperado, as menores perdas de eficiência de codificação foram alcançadas no ponto de operação Min e novamente as sequências HD devem ser destacadas, pois para este ponto de operação as sequências HD atingiram a maior redução de consumo de energia (10,26%), com o menor impacto de BDBR (somente 0,04%). Outra observação é que as reduções do tempo de codificação apresentam algumas imprecisões em comparação com as metas originais para os dois grupos de ferramentas da ME. Ainda assim, mesmo com tais imprecisões, a redução do tempo de codificação varia consistentemente com os pontos de operação, que era o objetivo principal.

Não é fácil fazer comparações com trabalhos relacionados de maneira justa, já que nenhum dos trabalhos na literatura foca em soluções amigáveis a *hardware* ou solução configurável. Como discutido anteriormente, existem alguns trabalhos focando na redução do tempo de codificação da predição interquadros do VVC usando uma variedade de técnicas, como os trabalhos Duarte et al. (2022), Jung; Jun (2021) e Park; Kang (2019), que miram na ferramenta ME *Affine* do VVC. Estes trabalhos relacionados não estão preocupados com os problemas da implementação de *hardware* e, por isso, os métodos propostos nestes trabalhos não são fáceis de serem implementados em *hardware*. Em uma aplicação real onde a solução em *hardware* é obrigatória, como na maioria das aplicações que suportam codificadores de vídeo atuais, as soluções apresentadas nestes trabalhos não serão diretamente aplicáveis. Esses

trabalhos relacionados também não suportavam nenhum nível de configurabilidade, o que também é um recurso importante para aplicativos do mundo real, principalmente aqueles executados em dispositivos alimentados por bateria. Além disso, esses trabalhos utilizaram diferentes versões do VTM, evitando também comparações diretas.

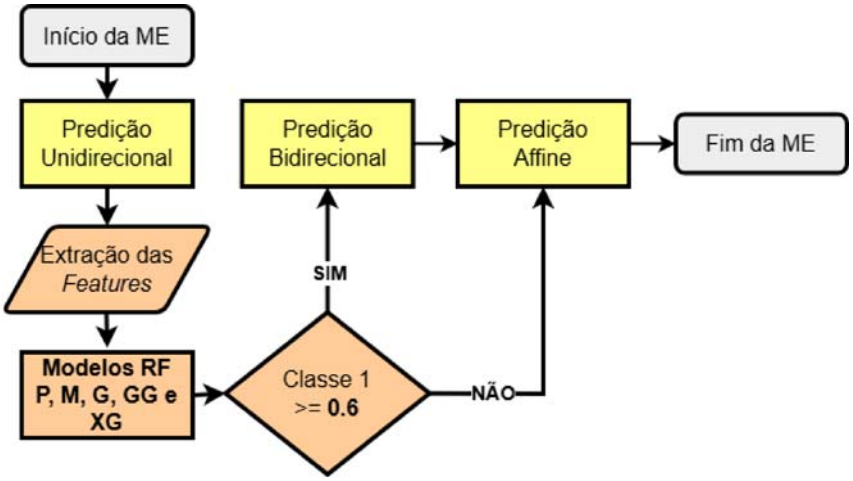
Mesmo com um foco diferente dos trabalhos relacionados, a heurística proposta alcançou resultados interessantes. Enquanto trabalhos relacionados têm foco total em otimizações de *software*, esta heurística implementável em *hardware* e configurável alcançou quase 23% de redução do tempo de codificação para a ferramenta de ME *Affine* com um aumento de 0,44% de BDBR quando considerando as sequências 4K UHD e o ponto de operação Max. Considerando apenas as otimizações de *software*, estes três trabalhos relacionados Duarte et al. (2022), Jung; Jun (2021) e Park; Kang (2019) alcançaram reduções de tempo maiores (de 33% (Jung; Jun, 2021) até 47% Duarte et al. (2022)) e menores resultados de BDBR (de 0,04% (Jung; Jun, 2021) até 0,18% Duarte et al. (2022)) para ME *Affine*. Mas estes trabalhos relacionados focam somente em uma ferramenta da ME e não consideram problemas de implementação em *hardware* e nem possuem configurabilidade. Além disso, como esperado, estes trabalhos não discutem problemas de consumo de energia.

8 REDUÇÃO DE CUSTO COMPUTACIONAL DA PREDIÇÃO BIDIRECIONAL USANDO APRENDIZADO DE MÁQUINA

Este capítulo apresenta uma solução de otimização para a predição Bidirecional da predição interquadros do VVC. Essa solução é baseada em aprendizado de máquina, utilizando modelos de florestas aleatórias (*random forests*). O objetivo da proposta é a redução do tempo de codificação da predição Bidirecional, com baixo impacto na eficiência de codificação.

A Figura 45 ilustra a proposta de otimização da predição Bidirecional a partir de modelos de aprendizado de máquina baseados em florestas aleatórias. O fluxo da ME inicia pela predição Unidirecional, que é executada normalmente e gera dados que são utilizados como *features* para os modelos treinados. Além disso, também são utilizados como *features* dados do próprio bloco atual que está sendo testado, além dos dados dos blocos pai, vizinho à esquerda e vizinho acima, que foram previamente codificados. As *features* extraídas são utilizadas como entrada para os modelos de florestas aleatórias treinados previamente (*off-line*).

Figura 45 – Solução de otimização aplicada à predição Bidirecional



Fonte: Elaborada pelo autor.

Foram implementados cinco modelos distintos, treinados a partir de *features* de

diferentes tamanhos de bloco, conforme detalhado na Tabela 10. De acordo com o tamanho do bloco atual que está sendo codificado, um dos cinco modelos é executado, retornando uma pontuação para as classes 1 (executar a predição Bidirecional) e 0 (não executar a predição Bidirecional). Para este trabalho, após alguns testes preliminares comparando o desempenho dos modelos com os valores 0,5, 0,6 e 0,7, foi definido o ponto de corte de 0,6 para a execução da Bidirecional. A descrição sobre a definição do ponto de corte será detalhada na seção 8.4. Sendo assim, quando o modelo retornar a pontuação da classe 1 igual ou maior que 0,6, a predição Bidirecional é executada; caso contrário, não é executada. Por fim, independentemente da decisão do modelo, as etapas *Affine* de 4 e 6-parâmetros são executadas conforme definido no padrão VVC.

A Tabela 10 apresenta os tamanhos de bloco suportados para cada modelo em específico. É possível notar que a organização ocorre de acordo com o tamanho dos blocos, em número de amostras de luminância. O Modelo P engloba blocos pequenos (de 64 e 128 amostras). O Modelo M abrange blocos médios (de 256 e 512 amostras). O Modelo G possui blocos grandes (de 1024 e 2048 amostras). O Modelo GG atende a blocos muito grandes (de 4096 e 8192 amostras). Por fim, o Modelo XG representa o bloco extra grande (de 16384 amostras).

Tabela 10 – Modelos e seus respectivos tamanhos de bloco

Modelo	Blocos	Tamanho (amostras)
P	8×8 4×16 16×4	64
	8×16 16×8 4×32 32×4	128
M	16×16 8×32 32×8 4×64 64×4	256
	16×32 32×16 8×64 64×8	512
G	32×32 16×64 64×16	1024
	32×64 64×32	2048
GG	64×64	4096
	64×128 128×64	8192
XG	128×128	16384

Todos os experimentos relacionados a essa solução de redução de custo computacional foram realizados utilizando um servidor com processador Intel Xeon Gold 5118 2,30 GHz com 56 GB de RAM. Para o treinamento dos modelos, foi utilizada a lingua-

gem Python, juntamente com a biblioteca `scikit-learn` (PEDREGOSA et al., 2011). Além disso, também foi utilizada a biblioteca `m2cgen` (Zeigerman, 2022) para realizar a exportação dos modelos de florestas aleatórias para a linguagem C++.

8.1 Extração de Dados

Na fase de extração das *features* foram utilizados os mesmos 15 vídeos descritos na seção 7.1. Esses vídeos são das bases de dados UVG (Mercat; Viitanen; Vanne, 2020) e NETVC (Daede; Norkin; Brailovskiy, 2019) e possuem valores distintos de informação espacial e temporal (SI e TI). Foram codificados os primeiros 16 quadros de cada vídeo no VTM 22.0, utilizando a configuração *Random Access*.

A Tabela 11 apresenta as *features* que foram extraídas durante a codificação, incluindo o nome da *feature*, a quantidade de cada *feature* e uma breve descrição de cada uma das *features*. No total, foram 57 features inicialmente selecionadas. A extração ocorreu logo após a execução da predição Unidirecional. Especificamente para as *features* bloco_atual_bidirecional, bidirecional_pai, bidirecional_viz_esq e bidirecional_viz_acima, o valor é obtido ao final da predição interquadros, indicando se o bloco em questão foi codificado, ou seja, teve o menor custo obtido pela predição Bidirecional. Além disso, as *features* sum, media, vari, desvioPadrao, grad, razao_grad, grad_razao_pixeis e dist_uni_L0_L1 são calculadas a partir dos dados do bloco que está sendo codificado. A utilização das *features* relacionadas aos blocos vizinhos foi inspirada pelo trabalho de Sagrilo et al. (2023). Já os trabalhos Lindino et al. (2022) e Saldanha (2021) motivaram a utilização das *features* calculadas a partir de dados do bloco, bem como o trabalho de Huang et al. (2023) que utilizou informações dos gradientes e desvio padrão. Por fim, o trabalho de Gonçalves (2021) foi considerado para o cálculo da distância entre os vetores de movimento. As demais *features*, tais como os vetores de movimento, seu custo e quantidade de bits, além da precisão, foram utilizadas por serem dados gerados pela predição Unidirecional e estarem disponíveis para a extração.

Tabela 11 – Relação das *Features* extraídas

Feature	**Qtd**	**Descrição**
tamBlocoWidth	1	Largura do bloco em píxeis
tamBlocoHeight	1	Altura do bloco em píxeis
numQP	4	Parâmetro de quantização de entrada
widthVideo	1	Largura do vídeo em píxeis
heightVideo	1	Altura do vídeo em píxeis
cu_pos	2	Posição X e Y do bloco dentro do quadro
imv	4	Precisão do Vetor de Movimento (AMVR)
mv_uni	4	Vetor de Movimento (X, Y) gerado pela predição Unidirecional para a Lista 0 e Lista 1
mv_pred_uni	4	Vetor de Movimento (X, Y) gerado pela predição AMVP Unidirecional para a Lista 0 e Lista 1
cost_mv_uni	2	Custo do Vetor de Movimento gerado pela predição Unidirecional para a Lista 0 e Lista 1
bits_mv_uni	2	Quantidade de bits do Vetor de Movimento gerado pela predição Unidirecional para a Lista 0 e Lista 1
atual_QP	1	Parâmetro de quantização utilizado no momento
bidirecional_pai	1	Indica se o bloco pai foi codificado pela predição Bidirecional
custo_pai	1	Custo gerado pela predição do bloco pai
IMV_pai	5	Valor do IMV do bloco pai
bidirecional_viz_esq	1	Indica se o bloco vizinho a esquerda foi codificado pela predição Bidirecional
custo_viz_esq	1	Custo gerado pela predição do bloco vizinho a esquerda
IMV_viz_esq	5	Valor do IMV do bloco vizinho a esquerda
bidirecional_viz_acima	1	Indica se o bloco vizinho acima foi codificado pela predição Bidirecional
custo_viz_acima	1	Custo gerado pela predição do bloco vizinho acima
IMV_viz_acima	5	Valor do IMV do bloco vizinho acima
sum	1	Somatório das amostras do bloco atual
media	1	Média das amostras do bloco atual
vari	1	Variância das amostras do bloco atual
desvioPadrao	1	Desvio padrão das amostras do bloco atual
grad	2	Gradientes Horizontal e Vertical do bloco atual (Sobel)
razao_grad	1	Gradiente Horizontal divido pelo Gradiente Vertical
grad_razao_píxeis	1	Soma dos gradientes divido pelo número de píxeis
dist_uni_L0_L1	1	Distância entre os Vetores de Movimento resultantes da predição Unidirecional da Lista 0 e Lista 1
bloco_atual_bidirecional	1	Indica se o bloco atual possui o melhor modo de predição como sendo Bidirecional

8.2 Elaboração e Seleção do *Dataset*

Após a extração das *features* foi realizada a etapa de balanceamento de dados. Primeiramente, foi definido o número mínimo de 1000 exemplos para cada decisão,

em cada tamanho de bloco, por vídeo e por QP. Esse número foi definido com o intuito de garantir um maior conjunto de dados em comparação com outros trabalhos da área. Em Gonçalves (2021) foi definido o número mínimo de 500 exemplos para cada tamanho de bloco e decisão. Em outros trabalhos, como Duarte (2021) e Saldanha (2021), apenas cita-se que os dados estão balanceados de acordo com o número de exemplos de cada quadro, tamanho de bloco, QP e classe. Duarte (2021) traz ainda o número de exemplares total extraído em cada caso, mas sem especificar o número mínimo considerado. O intuito de disponibilizar mais exemplos no conjunto de dados é possibilitar uma eficiência maior na etapa de treinamento do modelo.

Sendo assim, cada um dos 15 vídeos possui 1.000 exemplos da decisão 0 (não executar a Bidirecional) e 1.000 exemplos da decisão 1 (executar a Bidirecional), para cada um dos quatro QPs e em cada um dos 25 tamanhos de bloco. Dessa forma, foram selecionados o total de 3.000.000 de exemplos extraídos do processo de codificação, 1.500.000 para cada decisão.

Entretanto, em 20 dos 25 tamanhos de bloco ocorre o problema de não existir o mínimo de 1.000 exemplos para alguns vídeos e QPs. Em alguns casos extremos, há poucos ou nenhum exemplo para cada classe. Como exemplo, o tamanho de bloco 16×4, para o vídeo Netflix DinnerScene no QP 37, possui apenas seis exemplos para a classe 0 (não executa a Bidirecional) e nenhum exemplo para a classe 1 (executa a Bidirecional).

Para contornar esse problema, foram utilizados alguns critérios para selecionar exemplos extras, no intuito de garantir o número mínimo de 1.000 exemplos de cada tamanho de bloco, decisão, QP e vídeo. O primeiro critério $i)$ define que os exemplos extras serão de outros vídeos da mesma resolução e QP. Caso ainda não existam exemplos suficientes, o critério $ii)$ define que os exemplos extras podem ser de vídeos de outra resolução, porém do mesmo QP. Esses critérios foram assim definidos a fim de obter exemplos extras com valores de *features* mais próximos possíveis dos exemplos originais, sem prejudicar o balanceamento. As tabelas completas com os dados sobre o balanceamento estão disponíveis no Apêndice A.

8.2.1 Organização dos *Datasets*

A partir do total de 3.000.000 de exemplos selecionados, foram definidas seis opções de *datasets*, com diferentes organizações dos dados por tamanhos de blocos, conforme apresentado na Tabela 12. O objetivo foi definir qual seria o melhor agrupamento dos *datasets* para o treinamento dos modelos. Neste caso, cada *dataset* é usado no treinamento de um modelo de aprendizado de máquina para definir se a predição Bidirecional será ou não usada. As organizações **A** e **B** representam os dois extremos: **A** contém 25 *datasets*, sendo um para cada tamanho de bloco, já **B** possui somente um *dataset*, que engloba todos os tamanhos de bloco. Ou seja, se a orga-

Tabela 12 – Organizações de *Datasets*

Amostras	Tamanho	A	B	C	D	Tamanho	E	Tamanho	F
64	8×8	**1**	**S**	**P**	**S1**	8×8	**1x1**	8×8	**Qua**
64	4×16	**2**				16×16		16×16	
64	16×4	**3**				32×32		32×32	
128	8×16	**4**			**S2**	64×64		64×64	
128	16×8	**5**				128×128		128×128	
128	4×32	**6**				4×16	**1x4**	4×16	**Ver**
128	32×4	**7**				8×32		8×32	
256	16×16	**8**		**M**	**S3**	16×64		16×64	
256	8×32	**9**				16×4	**4x1**	8×16	
256	32×8	**10**				32×8		16×32	
256	4×64	**11**				64×16		32×64	
256	64×4	**12**				8×16	**1x2**	64×128	
512	16×32	**13**			**S4**	16×32		4×32	
512	32×16	**14**				32×64		8×64	
512	8×64	**15**				64×128		4×64	
512	64×8	**16**				16×8	**2x1**	16×4	**Hor**
1024	32×32	**17**		**G**	**S5**	32×16		32×8	
1024	16×64	**18**				64×32		64×16	
1024	64×16	**19**				128×64		16×8	
2048	32×64	**20**			**S6**	4×32	**1x8**	32×16	
2048	64×32	**21**				8×64		64×32	
4096	64×64	**22**		**GG**	**S7**	32×4	**8x1**	128×64	
8192	64×128	**23**			**S8**	64×8		32×4	
8192	128×64	**24**				4×64	**1x16**	64×8	
16384	128×128	**25**		**XG**	**S9**	64×4	**16x1**	64×4	

nização **A** for usada, então serão 25 modelos diferentes, um para cada tamanho de bloco. Por outro lado, se a organização **B** for usada, apenas um modelo será definido, independente do tamanho de bloco. Os conjuntos **C** e **D** têm como base o tamanho em amostras dos blocos. **C** engloba blocos a cada dois tamanhos de amostras diferentes, gerando cinco *datasets*. Por outro lado, **D** agrupa os blocos que têm a mesma quantidade de amostras, resultando em nove *datasets*. Já a organização **E** considera o fator de forma, possuindo nove *datasets*. Por fim, **F** baseia-se no formato dos blocos, que podem ser quadráticos, verticais ou horizontais, resultando em três *datasets*.

Quanto à preparação dos *datasets* para o treinamento, foi utilizada a técnica *one-hot* (Géron, 2019) nas colunas numQP, imv, IMV_pai, imv_vizEsq, imv_acima. Sendo assim, a partir dessas *features* que originalmente eram categóricas, foram geradas novas colunas binárias para cada categoria. Nessas colunas, apenas é atribuído o valor 1 quando o exemplo é da referida categoria, sendo que as demais recebem o valor 0, conforme ilustrado no exemplo da Tabela 13.

É importante destacar que em alguns exemplos extraídos, não há valores referentes aos blocos vizinhos acima ou à esquerda. Isso ocorre quando um bloco testado

Tabela 13 – Resultado da transformação da *feature* categórica **IMV** com a técnica *one-hot*

IMV_0	IMV_1	IMV_2	IMV_3
0	0	1	0
0	1	0	0
0	0	1	0
0	0	0	1
1	0	0	0

está localizado nas bordas superior ou esquerda do quadro. Em outros casos, os blocos não possuem um bloco pai, como, por exemplo, para blocos do tamanho 128×128. Esse fato também pode ocorrer em outros tamanhos de bloco (64×64, por exemplo) que se localizam na borda inferior do quadro e quando são os maiores tamanhos de bloco possíveis de serem testados.

Nessas situações, os registros extraídos possuem valores padrões gerados durante a extração (`4294967295` para o `custo` e `2147483647` para o `IMV`). Sendo assim, para que esses valores não prejudiquem o treinamento e a performance dos modelos, os mesmos precisam ser ajustados. No caso das *features* `IMV` dos blocos vizinhos e bloco pai, a categoria *null* também foi considerada no processo de binarização com a técnica *one-hot*. Já para as *features* `custo_vizEsq`, `custo_acima` e `custo_pai`, foi calculada e atribuída a respectiva média dos custos válidos entre todos os registros.

Outro ajuste realizado nos *datasets* se refere aos custos resultantes da predição Unidirecional para as Listas 0 e 1. Nos *datasets* **B**, **C**, **D**, **E** e **F**, que englobam mais do que um tamanho de bloco, os valores das *features* `cost_Mv_Uni_L0` e `cost_Mv_Uni_L1` foram ponderados de acordo com o número de amostras do bloco.

8.2.2 Seleção da Melhor Organização de *Dataset*

Para definir qual seria a melhor opção dentre as seis organizações de *datasets* já apresentadas, foi realizado um treinamento padronizado para todas as seis organizações. Os hiperparâmetros considerados nesse treinamento padrão foram: o critério de ganho de informação (criterion='gini'), o número de árvores (n_estimators = 50) e a profundidade máxima das árvores (max_depth = 15). Os demais hiperparâmetros não foram alterados, sendo que consideram os valores padrão. Os *datasets* referentes aos seis tipos de organização foram separados em treino e teste, com a proporção de 80% e 20%, respectivamente.

Os resultados são apresentados na Tabela 14, com o percentual obtido para as métricas de Acurácia, Precisão, Recall e F1-score, além dos valores para Falsos Negativos e Falsos Positivos. No caso dos *datasets* **A**, **C**, **D**, **E** e **F**, com mais de um tamanho de bloco, foi calculada a média das métricas para consideração do desempenho geral dos modelos. Também foram calculados os percentuais de falsos negativos, que estão relacionados ao possível impacto do modelo na eficiência de codificação,

visto que consistem nas predições do modelo para não realizar a Bidirecional, porém o correto seria realizar a Bidirecional. Já o percentual de falsos positivos indica uma perda de eficiência do modelo na redução do custo computacional, pois representa os erros do modelo quando prevê realizar a Bidirecional, mas o correto seria não realizar.

Tabela 14 – Resultado do treinamento padrão para as seis organizações de *Datasets*

Tipo Dataset		**Acurácia**	**Precisão**	**Recall**	**F1-score**	**Falsos Negativos**	**Falsos Positivos**
A	*0*	67,7%	69,6%	62,9%	66,1%	13,6%	18,6%
	1		66,4%	72,8%	69,4%		
B	*0*	66,0%	68,0%	58,0%	63,0%	13,6%	20,8%
	1		64,0%	73,0%	68,0%		
C	*0*	**67,6%**	**69,8%**	**59,8%**	**64,8%**	**13,0%**	**20,1%**
	1		**65,0%**	**74,2%**	**69,0%**		
D	*0*	68,2%	70,1%	61,8%	65,6%	13,1%	19,1%
	1		66,1%	73,9%	69,9%		
E	*0*	67,0%	68,9%	60,6%	64,3%	13,7%	19,7%
	1		64,8%	72,4%	68,3%		
F	*0*	66,3%	68,7%	58,7%	63,3%	13,3%	20,6%
	1		64,0%	73,3%	68,3%		

Conforme os resultados apresentados na Tabela 14, percebe-se que as organizações dos *datasets* não apresentam diferenças significativas nos resultados prévios obtidos. Para definição de qual organização utilizar, foi escolhida a métrica do percentual de Falsos Negativos, visto que a mesma tem relação com o impacto do modelo na eficiência de codificação. Isso porque, apesar da solução ser voltada para otimização do tempo de codificação, é importante manter as perdas na eficiência de codificação tão baixas quanto possível, a fim de que a solução seja passível de utilização prática. Sendo assim, foi definida a utilização da organização de *dataset* **C**, pois a mesma possui um menor percentual de Falsos Negativos.

Outra questão importante que pesou nessa decisão diz respeito ao número de modelos de aprendizado de máquina necessários para a tomada da decisão no uso da predição Bidirecional. As outras duas organizações de *datasets* com resultados mais próximos da organização **C** são as organizações **A** e **D**. Mas estas organizações implicam no uso de um número muito maior de modelos, quando comparado à organização **C**. A organização **C** usa cinco modelos distintos, enquanto as organizações **A** e **D** usam, respectivamente, 25 e nove modelos. O uso de um número menor de modelos é importante para reduzir o impacto no custo computacional da solução completa.

8.3 Treinamento dos Modelos

Após a decisão pelo uso da organização de *dataset* C, com cinco modelos, foi realizado um conjunto de experimentos para aprimorar o treinamento destes cinco modelos, incluindo a busca por hiperparâmetros mais relevantes, a seleção de *features* com maior ganho de informação, para então os modelos serem treinados novamente. Estas etapas são descritas nesta seção.

8.3.1 Busca de Hiperparâmetros

Para o refinamento dos hiperparâmetros para o treinamento dos modelos, os *datasets* foram separados em treino e teste, com a proporção de 75% e 25% dos dados. A Tabela 15 apresenta o número de registros dessas divisões em cada *dataset*. O total de registros de cada *dataset* corresponde à proporção do número de tamanhos de blocos incluídos no mesmo, considerando que cada um possui 1.000 registros para cada classe (0 e 1).

Tabela 15 – Número de registros de cada *Dataset*

Modelo	Nº de Blocos	Treino (75%)	Teste (25%)	Total
P	7	630.000	210.000	840.000
M	9	810.000	270.000	1.080.000
G	5	450.000	150.000	600.000
GG	3	270.000	90.000	360.000
XG	1	90.000	30.000	120.000

Primeiramente, foi realizada a busca de hiperparâmetros utilizando o método *Random Search*. Para cada hiperparâmetro, foi definida uma faixa de valores, conforme mostrado na Tabela 16. Essas faixas de valores foram definidas a partir dos trabalhos de Duarte (2021) e Andrades; Grellert; Fonseca (2019), com algumas alterações visando adaptar os espaços de busca. Sendo assim, ao ser executado, o método seleciona 500 combinações de valores diferentes de maneira aleatória, realizando o treinamento e o teste. A representação das faixas de valores na tabela segue a indicação do valor inicial, final e a iteração (ou “passo”), sendo que esta última é considerada 1 quando não é indicada. Por exemplo, para o hiperparâmetro ***n_estimators*** é gerada a faixa de valores entre 2 e 25. Já para o hiperparâmetro ***min_samples_split*** é considerado valor padrão 2 e os valores da faixa entre 10 e 100, variando a cada 10 (ou seja, 10, 20, 30, etc, até 100).

A Tabela 17 apresenta a combinação de hiperparâmetros, realizada através do método *Random Search*, que resultou no maior F1-score para cada modelo. Na Tabela, estão destacados em amarelo os dois hiperparâmetros com maior valor de correlação em cada modelo. Nesse sentido, a profundidade máxima (***max_depth***) destaca-se em todos os modelos. Já o número de árvores (***n_estimators***) também apresenta

Tabela 16 – Faixas de valores utilizadas no *Random Search*

Hiperparâmetro	Faixa de valores
n_estimators	[2..25]
criterion	[gini, entropy]
min_samples_split	[2] OU [10..100, 10]
min_samples_leaf	[1] OU [10..100, 10]
max_features	[sqrt] OU [1..total]
max_depth	[2..10]

alta correlação para os modelos P e M. Na sequência, o critério (***criterion***) destaca-se no modelo G, e o número máximo de *features* (***max_features***) para os modelos GG e XG.

Tabela 17 – Melhor resultado do *Random Search* por modelo

Modelo/ Hiperparâmetro	P	M	G	GG	XG
n_estimators	24	20	19	22	20
criterion	gini	gini	gini	gini	gini
min_samples_split	90	2	70	40	30
min_samples_leaf	100	1	100	1	10
max_features	35	33	29	46	27
max_depth	10	10	10	10	10
F1-score	**0,69**	**0,63**	**0,64**	**0,65**	**0,67**

Posteriormente, esses resultados foram utilizados como base para aplicar o método *Grid Search*, que resulta na combinação de hiperparâmetros com maior F1-score, considerando todas as combinações possíveis. A Tabela 18 apresenta a faixa de valores utilizada em cada hiperparâmetro e modelo no método *Grid Search*. Nota-se que os hiperparâmetros destacados na Tabela 17 são configurados para explorar mais possibilidades, visto que os mesmos tiveram maior correlação com o *F1-score*. Os demais hiperparâmetros são testados com o valor padrão do modelo *Random Forest* e o valor resultante do *Random Search*. Essas definições são semelhantes aos valores presentes nos trabalhos de Duarte (2021) e Andrades; Grellert; Fonseca (2019), com alguns ajustes a fim de diminuir a quantidade de treinamentos necessários.

Tabela 18 – Faixas de valores utilizadas no *Grid Search*

Modelo/ Hiperparâmetro	P	M	G	GG	XG
n_estimators	[2..24]	[2..20]	[19]	[22]	[20]
criterion	[gini]	[gini]	[gini, entropy]	[gini]	[gini]
min_samples_split	[2] OU [90]	[2]	[2] OU [70]	[2] OU [40]	[2] OU [30]
min_samples_leaf	[1] OU [100]	[1]	[1] OU [100]	[1]	[1] OU [10]
max_features	[sqrt] OU [35]	[sqrt] OU [33]	[sqrt] OU [29]	[sqrt] OU [46..57]	[sqrt] OU [27..50]
max_depth	[1..10]	[1..10]	[1..10]	[1..10]	[1..10]
Nº candidatos	**1840**	**380**	**160**	**260**	**1000**

É importante destacar que inicialmente o *Grid Search* gerou os melhores resulta-

dos para combinações com o hiperparâmetro max_depth igual a 10. Entretanto, ao realizar o treinamento e gerar os modelos finais para implementação no VTM, os mesmos se mostraram excessivamente grandes, atingindo até Gigabytes. Isso se deve ao fato da combinação da profundidade 10 com o número de árvores dos modelos, que acarreta no crescimento exponencial do modelo. Essa característica é considerada um problema, visto que acarreta dificuldades já na etapa de implementação dos modelos no VTM, exigindo uma alta capacidade de memória RAM, além de necessitar um hardware dedicado muito amplo para uma possível aplicação real. Para contornar esse problema, foi adotada uma estratégia de seleção alternativa baseada na média dos desempenhos. Inicialmente, foram analisados os melhores resultados de *F1-score* para cada valor de max_depth no intervalo de 1 a 10, conforme as combinações já exploradas pelo *Grid Search*. Em seguida, foi calculada a média desses valores e selecionado o valor de max_depth cuja métrica estivesse mais próxima dessa média. Essa abordagem permitiu reduzir significativamente o tamanho dos modelos, sem grandes impactos na performance preditiva.

Sendo assim, as combinações de hiperparâmetros selecionadas a partir dessa estratégia estão representadas na Tabela 19. Também são apresentados os valores de F1-score obtidos durante o processamento do *Grid Search* para as respectivas combinações. Os valores dos hiperparâmetros foram utilizados para o treinamento final dos modelos, que serão descritos na próxima seção.

Tabela 19 – Hiperparâmetros utilizados para o treinamento final

Modelo/ Hiperparâmetro	**P**	**M**	**G**	**GG**	**XG**
n_estimators	13	11	19	22	20
criterion	gini	gini	gini	gini	gini
min_samples_split	90	2	2	2	2
min_samples_leaf	100	1	1	1	1
max_features	sqrt	33	29	46	36
max_depth	6	7	8	2	8
F1-score	**0,66**	**0,64**	**0,67**	**0,67**	**0,71**

8.3.2 Seleção das *Features*

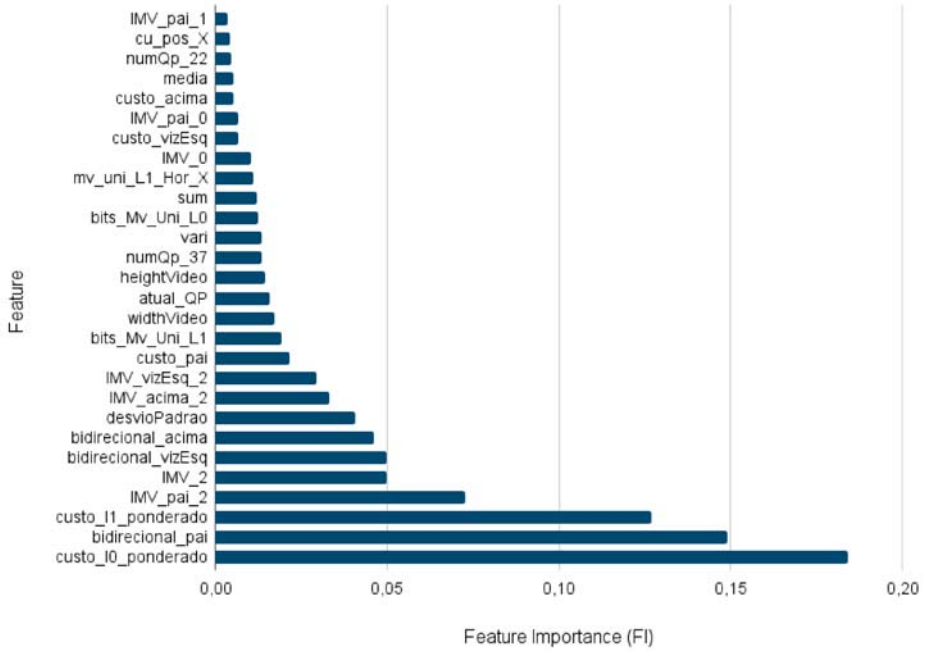
Após a definição dos hiperparâmetros, foi realizado um processo de seleção das 57 *features* inicialmente definidas, com o objetivo de simplificar os modelos finais. Para isso, foi realizado um treinamento utilizando os hiperparâmetros da Tabela 19 e, posteriormente, foram gerados os valores de *Feature Importance* (FI) para cada modelo.

Assim, nesta tese, estão apresentados os gráficos de FI dos cinco modelos. Para facilitar a compreensão e manter um padrão visual dos gráficos, os mesmos são apresentados com os valores das 28 *features* de maior relevância em cada modelo, dentre

as 57 *features* disponíveis. A quantidade de 28 *features* foi utilizada como referência a partir dos resultados gerados para o modelo XG, considerando apenas as *features* com valores de FI a partir de 0,01. Portanto, para os demais modelos, algumas *features* são representadas nos gráficos, porém as mesmas possuem menos de 0,01 de importância.

Na Figura 46 é ilustrado o gráfico de FI para o modelo P. Nota-se que apenas quatro *features* possuem impacto maior que 5%, sendo duas delas relacionadas ao bloco pai (IMV_pai_2 e bidirecional_pai) e as outras duas relacionadas ao custo ponderado da etapa Unidirecional (custo_l1_ponderado e custo_l0_ponderado). Esta última atinge 18% de importância nas predições.

Figura 46 – *Feature Importance* - Modelo P



Fonte: Elaborada pelo autor.

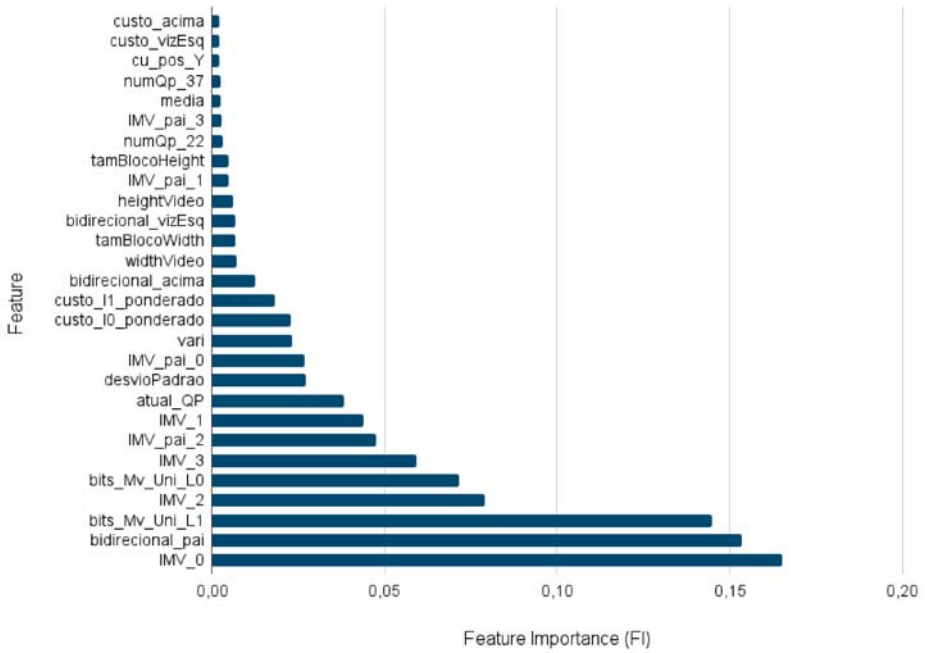
Já o gráfico da Figura 47 apresenta a distribuição da importância para as *features* do modelo M. Nesse caso, seis *features* atingem mais do que 5% de relevância; entretanto, duas delas se destacam com mais de 15% (bidirecional_pai e IMV_0).

Os valores de importância das *features* para o modelo G são apresentados na Figura 48. Em contraste com os casos anteriores, no modelo G a *feature* IMV_0 possui impacto de 37% nas predições, destacando-se das demais.

A Figura 49 ilustra os valores de impacto para as *features* do modelo GG. Notavelmente, este modelo se diferencia dos demais com apenas sete *features* com importância a partir de 1%. Novamente, o IMV_0 se destaca, porém, nesse caso, com 81% de relevância para as predições do modelo GG.

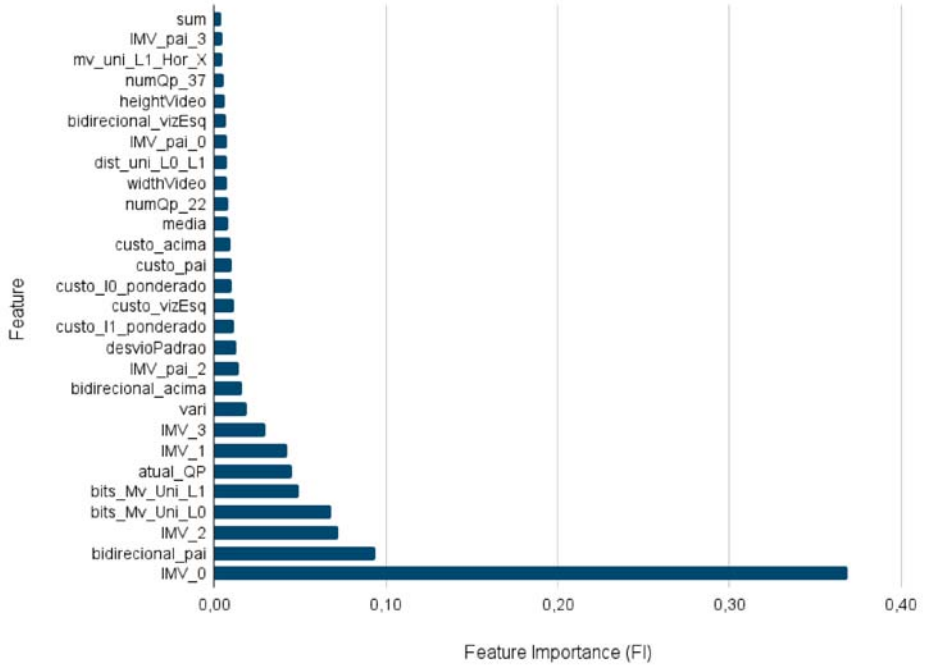
Por fim, o gráfico da Figura 50 representa os valores de relevância para as *features* do modelo XG. É importante destacar que, ao se referir aos blocos 128×128, esse

Figura 47 – *Feature Importance* - Modelo M



Fonte: Elaborada pelo autor.

Figura 48 – *Feature Importance* - Modelo G

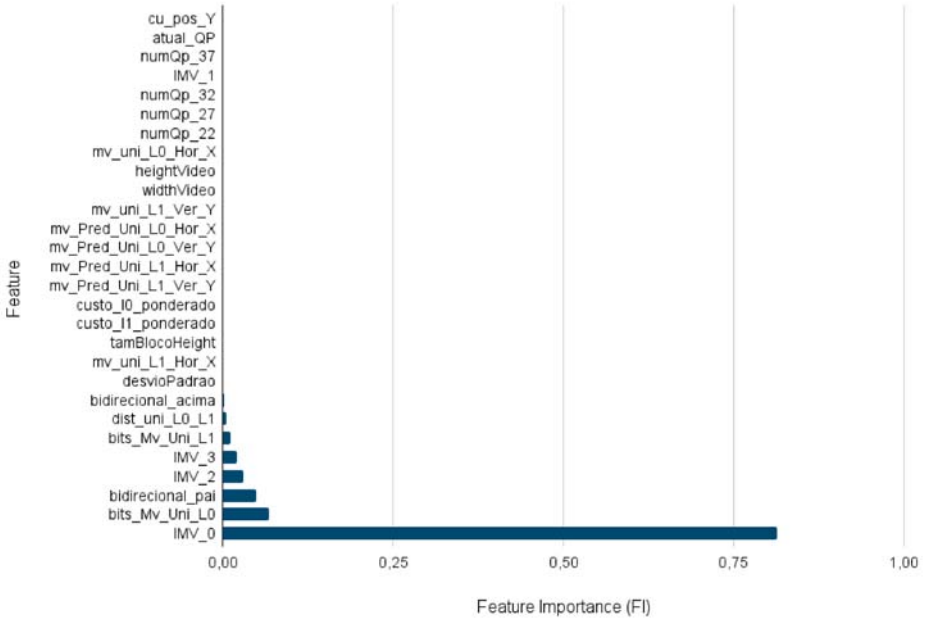


Fonte: Elaborada pelo autor.

modelo não utiliza *features* de bloco pai. Mais uma vez, a *feature* IMV_0 possui maior relevância, com 42%, seguido das demais *features* com 8% ou menos.

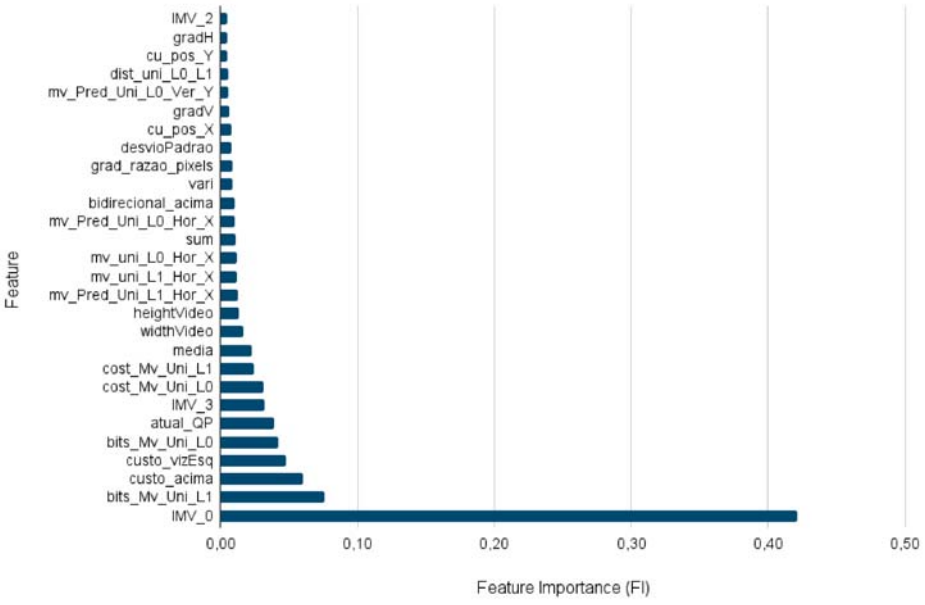
A partir da análise dos gráficos anteriores, é possível destacar o comportamento distinto entre os modelos, corroborando com a proposta de definir modelos separados de acordo com os diferentes tamanhos de bloco. Além disso, notam-se algumas

Figura 49 – *Feature Importance* - Modelo GG



Fonte: Elaborada pelo autor.

Figura 50 – *Feature Importance* - Modelo XG



Fonte: Elaborada pelo autor.

semelhanças entre os modelos M até XG, no que se refere às *features* com maior relevância. Também foi possível observar que várias das features originalmente definidas possuem pouca ou nenhuma relevância para todos os modelos, indicando uma oportunidade de simplificação desses modelos.

Assim, após a análise dos resultados, optou-se pela exclusão de *features* com menor relevância geral para os modelos e que demandam mais tempo de extração no VTM (*features* relacionadas aos blocos pai e vizinhos, e *features* calculadas a partir dos valores do bloco, como média, variância, soma, desvio padrão e gradientes). A

partir dessa análise, foram selecionadas para exclusão as *features* com menos de 1% de FI ou cuja soma do grupo de *features* fosse menor que 5%.

A Tabela 20 apresenta o grupo de *features* excluídas em cada um dos cinco modelos. O 'X' representa que o grupo de *features* foi excluído do referido modelo. Para os blocos pai e vizinhos, o grupo de *features* se refere às respectivas informações sobre Bidirecional, custo e IMV. Já as *features* relacionadas aos gradientes são os valores horizontal, vertical, razão e normalização do gradiente. Importante destacar, também, que foram excluídas as *features* referentes ao IMV nulo para blocos pai e vizinhos, visto que apresentaram 0% de FI em todos os modelos. Após esse processo, o número de *features* diminuiu de 57 para 50 no *dataset* P, 38 nos *datasets* M, G e GG, e 44 *features* no *dataset* XG.

Tabela 20 – *Features* excluídas do treinamento final

Feature\ Modelo	Bloco Pai	Vizinho Esquerda	Vizinho Acima	Gradientes
P				X
M		X	X	X
G		X	X	X
GG		X	X	X
XG	X			X

8.3.3 Treino Final dos Modelos

Com base na seleção das *features* e dos hiperparâmetros (Tabela 19) os cinco modelos foram treinados e posteriormente testados com os mesmos *datasets* utilizados desde a etapa de busca com o método *Random Search*. Ou seja, os testes foram realizados com os 25% de dados restantes do *dataset* original, garantindo que os resultados antes da implementação no VTM fossem mais próximos à realidade. Nesse sentido, a Tabela 21 apresenta os resultados das métricas de desempenho deste teste para os modelos. Após esses testes, foi realizado o treinamento final dos modelos, considerando os *datasets* originais, formados pelos dados dos grupos de treino (75%) e teste (25%).

Em geral, nota-se que para a classe 1 (que ativa a Bidirecional) os modelos têm melhor *recall*, com média de 74%. Porém, a classe 0 (que desativa a Bidirecional) tem *recall* médio com valor baixo, com 57,6%. Isso demonstra uma tendência dos modelos de priorizar a correta ativação da predição Bidirecional, alinhada ao objetivo de não impactar na eficiência de codificação. Em compensação, os modelos acabam prevendo execuções desnecessárias da predição Bidirecional, considerando a média de 21,3% de Falsos Positivos.

Especificamente, o modelo P tem os melhores resultados, com a maior acurácia, atingindo 69%. Além disso, há um aparente equilíbrio entre as demais métricas, inclu-

Tabela 21 – Resultados do treinamento e teste finais

Métrica/ Modelo		Acurácia	Precisão	Recall	F1-score	Falsos Negativos	Falsos Positivos
P	0 1	69%	69% 68%	67% 71%	68% 69%	14,7%	16,7%
M	0 1	64%	66% 62%	58% 70%	61% 66%	15,0%	21,2%
G	0 1	65%	68% 61%	54% 75%	61% 68%	12,5%	22,9%
GG	0 1	66%	69% 63%	55% 75%	61% 68%	12,3%	22,5%
XG	0 1	66%	72% 63%	54% 79%	62% 70%	10,5%	23,1%
Média	**0** **1**	**66,0%**	**68,8%** **63,6%**	**57,6%** **74,0%**	**62,6%** **68,2%**	**13,0%**	**21,3%**

sive nas taxas de Falsos Negativos e Falsos Positivos.

Já o modelo M apresentou a menor acurácia, com 64%, e o maior percentual de falsos negativos (15%). Além disso, o F1-score para a classe 1 é o mais baixo (66%), indicando que o modelo possui limitações ao errar mais nas previsões de não execução da predição Bidirecional.

Os modelos G e GG possuem resultados semelhantes, com acurácia de 65 e 66%, respectivamente. O *recall* de 75% indica alta eficiência nas ativações da predição Bidirecional. Entretanto, essa métrica é baixa quando se refere a não ativar a Bidirecional (classe 0), indicando efetividade em apenas metade das vezes. As altas taxas de falsos positivos também indicam uma tendência de ativações excessivas.

Por fim, o modelo XG possui melhor desempenho na classe 1, com *recall* de 79% e a menor taxa de Falsos Negativos (10,5%). Porém, para a classe 0, o *recall* é de apenas 54%, com a maior taxa de falsos positivos (23,1%), demonstrando uma tendência a ativações desnecessárias.

8.4 Definição do Ponto de Corte

A fim de definir o ponto de corte a ser implementado no VTM, foram realizados alguns testes simplificados utilizando o vídeo Cactus das CTCs, que possui resolução FHD. Esse vídeo foi escolhido pelo fato de possuir uma relação de SI/TI mediana, quando comparado aos outros vídeos da mesma classe. Foram testadas as possibilidades de ponto de corte 0,5, 0,6 e 0,7, sendo que os dois últimos testes já consideram a seleção das *features*. Por padrão, os modelos retornam a pontuação de cada classe, ou seja, do total de 1,0, qual é a pontuação para a classe 0 e 1. Por exemplo, um modelo pode retornar as pontuações 0,35, para a classe 0, e 0,65 para a classe 1. Nesse caso, essa pontuação significa que o modelo indica a classe 1 como a mais provável

predição para os dados de entrada.

Sendo assim, os resultados de cada modelo passaram por três testes com os pontos de corte diferentes: a Bidirecional era executada normalmente quando a pontuação da classe 1 fosse maior ou igual a 0,5 (1º teste), 0,6 (2º teste) ou 0,7 (3º teste). Dessa forma, foi aumentado o ponto de corte para que a Bidirecional fosse executada, ou seja, mesmo com uma pontuação mais baixa, a Bidirecional foi ignorada mais vezes. Os resultados de BDBR, redução de tempo da Bidirecional e total, de cada teste são apresentados na Tabela 22. Nota-se que, para o ponto de corte 0,5, a redução de tempo total foi mínima, assim como o BDBR, com apenas 48,4% de redução no tempo da Bidirecional. Já para os pontos de corte 0,6 e 0,7 houve significativa redução no tempo da etapa Bidirecional, com redução semelhante no tempo total. O tempo da extração das *features* (MT_{feat}) reduziu para os dois últimos testes em função da seleção das *features*. Conforme destacado na tabela, o ponto de corte 0,6 foi escolhido para a implementação, pois o desempenho dos modelos foi melhor em termos da relação de redução de tempo de codificação (MRT_{Total}) e perda de eficiência de codificação ($BDBR$).

Sendo assim, ficou definido na implementação final no VTM que a etapa Bidirecional é executada somente se a pontuação (*score*) retornada pelo modelo na classe 1 é igual ou maior que 0,6 (conforme ilustrado anteriormente na Figura 45). Ou seja, a etapa Bidirecional será omitida sempre que a pontuação para a classe 0 for maior que 0,4.

Tabela 22 – Testes realizados com diferentes pontos de corte

Ponto de Corte	MRT_{bi} (%)	MRT_{Total} (%)	$BDBR$ (%)	MT_{RF} (%)	MT_{feat} (%)
0,5	48,4	0,2	0,156	0,3	1,2
0,6	**89,5**	**4,8**	**0,362**	**0,3**	**0,9**
0,7	99,8	5,6	0,475	0,3	0,9

8.5 Implementação no VTM

Para as avaliações, foram geradas três versões do software de referência: VTM original, VTM com custo reduzido e VTM sem a predição Bidirecional. A versão do VTM original é usada como âncora para as comparações. A versão do VTM com custo reduzido é usada para avaliar o desempenho dos modelos treinados em um cenário real. Por fim, a versão do VTM sem a predição Bidirecional é importante para indicar os limites máximos de redução de tempo de execução e de perda de eficiência de codificação com a remoção total da predição Bidirecional. Uma solução ideal de redução de custo computacional teria uma redução de tempo mais próxima possível da atingida pela exclusão da predição Bidirecional, mas com perda de eficiência de

codificação o mais próximo possível de zero.

Para viabilizar a implementação dos modelos no VTM, que utiliza a linguagem C++, foi necessário utilizar a biblioteca `m2cgen` (*Model 2 Code Generator*) (Zeigerman, 2022). Através da função `export_to_c` é possível realizar a tradução do modelo estatístico para a linguagem C++. Adicionalmente, o VTM também foi modificado para implementar a extração das *features* que são usadas como entrada para os modelos. Também foram implementadas rotinas para contabilizar as decisões dos modelos para as classes 0 e 1. Cada modelo foi implementado como uma classe na pasta `CommonLib` do VTM. Após o bloco atual ser testado pela etapa Unidirecional, as *features* são extraídas e passadas para um dos cinco modelos, de acordo com o seu tamanho do bloco.

A versão do VTM sem a predição Bidirecional exigiu uma modificação no VTM para excluir a predição Bidirecional e manter o restante das funcionalidades do codificador.

Por fim, as três versões do VTM foram modificadas para incluir rotinas de extração de tempo de processamento para algumas etapas específicas, tais como: predições intraquadro e interquadros (com as respectivas predições Unidirecional, Bidirecional e *Affine*) e ferramenta Merge.

8.6 Resultados no VTM

Nesta seção, serão apresentados e discutidos os resultados obtidos a partir da implementação dos cinco Modelos de *Random Forest* para a redução de custo computacional da etapa Bidirecional do VVC. Para a obtenção desses resultados, foram codificados os 32 primeiros quadros de 15 sequências de vídeos das CTCs (Tabela 23), com a configuração *Random Access*. Cada vídeo foi codificado para os QPs 22, 27, 32 e 37, considerando as três versões do software de referência previamente apresentadas.

Para a verificação do desempenho dos modelos, foram consideradas as métricas de tempo de codificação e BDBR. Das métricas disponibilizadas pelo VTM, são consideradas o tempo total, a taxa de bits e o YUV-PSNR. Os demais dados foram extraídos através das modificações no VTM já comentadas na seção anterior.

A Redução de Tempo para cada Etapa (RT_{etapa}) é calculada conforme a equação (8), onde o tempo de codificação da etapa específica do VTM com custo reduzido (VTM_{β}) ou sem Bidirecional (VTM_{Ω}) é dividido pelo tempo de codificação da etapa específica do VTM original (VTM_{α}). De maneira semelhante, também é calculada a Redução de Tempo Total (RT) de codificação, conforme a equação (9). Esses cálculos são realizados para cada sequência de vídeo e cada QP.

$$RT_{etapa} = 1 - \frac{TempoEtapa_{(VTM_{\beta||\Omega})}}{TempoEtapa_{(VTM_{\alpha})}} \times 100 \tag{8}$$

Tabela 23 – Sequências de vídeos utilizadas nos testes de otimização

Nome	Classe	Resolução	Nº Quadros	Taxa (fps)
Campfire	A1	3840x2160	300	30
FoodMarket4			300	60
Tango2			294	60
CatRobot	A2	3840x2160	600	60
DaylightRoad2			600	60
ParkRunning3			600	50
BasketballDrive	B	1920x1080	500	50
BQTerrace			600	60
Cactus			500	50
MarketPlace			600	60
RitualDance			600	60
ArenaOfValor	F	1920x1080	600	60
BasketballDrillText		832x480	500	50
SlideEditing		1280x720	300	30
SlideShow			500	20

$$RT = 1 - \frac{TempoTotal_{(VTM_{\beta||\Omega})}}{TempoTotal_{(VTM_{\alpha})}} \times 100 \tag{9}$$

Para a obtenção das médias, consideram-se os valores de Redução de Tempo para cada Etapa (RT_{etapa}) e total (RT) nos quatro QPs ($QP_i \in \{22, 27, 32, 37\}$). Esses cálculos são demonstrados nas equações 10 e 11, respectivamente, sendo MRT_{etapa} a Média da Redução de Tempo das etapas específicas e MRT a Média de Redução de Tempo Total de codificação.

$$MRT_{etapa} = \frac{1}{4} \sum_{QP_i \in \{22,27,32,37\}} RT_{etapa(QP_i)} \tag{10}$$

$$MRT = \frac{1}{4} \sum_{QP_i \in \{22,27,32,37\}} RT_{(QP_i)} \tag{11}$$

Também foram calculados os valores de *overhead* da solução, conforme as equações 12 e 13. Nessas fórmulas, T_{modelo} e MT_{modelo} representam, respectivamente, a taxa e a média do tempo gasto pela solução, sendo que esses valores podem ser referentes à execução das *Random Forest* (RF) ou à extração das *features* ($feat$).

$$T_{modelo} = \frac{Tempo_{RF(VTM_{\beta})}||Tempo_{feat(VTM_{\beta})}}{TempoTotal_{(VTM_{\beta})}} \times 100 \tag{12}$$

$$MT_{modelo} = \frac{1}{4} \sum_{QP_i \in \{22,27,32,37\}} T_{modelo}(QP_i) \tag{13}$$

Já os resultados referentes à eficiência de codificação são obtidos através do cál-

culo do BDBR (Bjøntegaard, 2001), que considera os valores da taxa de bits (*bitrate*) e YUV-PSNR dos quatro QPs em cada vídeo. A fim de acrescentar mais um aspecto na análise dos resultados, também foi calculada a relação entre a redução do tempo de codificação total e a perda na eficiência de codificação. Essa relação foi obtida por $MRT_{Total}/BDBR$, sendo que a mesma vai indicar a proporção de redução de tempo total obtida ao custo de cada ponto percentual do BDBR.

Na sequência, os resultados comparativos entre a solução de redução de custo para a etapa Bidirecional e o VTM original serão apresentados em três tabelas. Primeiramente, a Tabela 24 apresenta os resultados de redução de tempo para a etapa Bidirecional e para todo o codificador, além do *overhead* da solução. Já a Tabela 25 apresenta uma visão mais ampla dos resultados obtidos, abrangendo os números para as etapas específicas Bidirecional, Unidirecional, *Affine*, além de incluir as etapas interquadros, intraquadro, *Merge* e o tempo total. A análise do impacto da solução na eficiência de compressão juntamente com o tempo total pode ser vista na Tabela 26. Por fim, ainda é exibida na Tabela 27 uma comparação da solução de redução de custo da Bidirecional com uma versão do VTM sem a ferramenta.

A partir dos valores apresentados na Tabela 24 é possível afirmar que a solução proposta atingiu uma redução muito significativa no tempo de codificação da etapa Bidirecional (MRT_{bi}), com médias maiores que 90% em todas as classes de vídeos. Pode-se destacar o resultado do vídeo `FoodMarket4` da classe A1, que obteve a maior redução, com 95,6%. Por outro lado, a redução do tempo total do codificador (MRT) foi bem mais modesta, atingindo um máximo de 3,9% para o vídeo `SlideEditing` da classe F e um mínimo de 0,5% para o vídeo `MarketPlace` da classe B. É importante destacar que, nestes tempos, já está somado o custo computacional do modelo de aprendizado de máquina. Esse baixo impacto no tempo total do codificador causou surpresa e, em função disso, novos experimentos foram feitos para entender o que estava acontecendo e essa discussão será melhor abordada na sequência deste texto.

A Tabela 24 também apresenta o *overhead* da solução implementada, ou seja, o percentual de tempo necessário para extrair as *features* (MT_{feat}) e para executar os modelos (MT_{RF}). O MT_{RF} apresenta valores médios que não ultrapassam 0,3% do tempo total de codificação, enquanto que o MT_{feat} apresentou uma participação maior no tempo total do codificador, chegando a um máximo de 1,7%. Quando os dois valores são somados, o pior caso chega a 2,1% do tempo total de codificação. Esse valor é elevado e afetou diretamente os resultados de ganho de tempo total (MRT). Neste ponto, é importante destacar que, nos testes preliminares descritos anteriormente, já havia sido notado um resultado de *overhead* significativo. Portanto, os esforços de simplificação dos modelos de RF, através da utilização de profundidades de árvores menores e da seleção das *features*, contribuíram para diminuir esse impacto, porém ainda deixando oportunidade para melhorias.

Tabela 24 – Impacto dos modelos na redução de tempo de execução

Vídeo/ Classe	MRT_{bi} **(%)**	MRT_{Total} **(%)**	MT_{RF} **(%)**	MT_{feat} **(%)**
ArenaOfValor	92,5	2,8	0,4	1,7
BasketballDrillText	92,0	1,0	0,4	1,7
SlideEditing	91,8	3,9	0,2	1,2
SlideShow	93,1	3,5	0,3	1,2
Média Classe F	**92,4**	**2,8**	**0,3**	**1,5**
BasketballDrive	95,5	1,5	0,3	1,3
BQTerrace	92,7	2,6	0,3	1,4
Cactus	92,4	2,2	0,4	1,6
MarketPlace	85,7	0,5	0,2	1,3
RitualDance	90,9	1,8	0,3	1,4
Média Classe B	**91,4**	**1,7**	**0,3**	**1,4**
Campfire	91,6	2,8	0,4	1,7
FoodMarket4	95,6	0,7	0,2	0,9
Tango2	93,1	0,7	0,3	1,3
Média Classe A1	**93,4**	**1,4**	**0,3**	**1,3**
CatRobot	91,9	1,4	0,3	1,3
DaylightRoad2	91,1	2,7	0,3	1,4
ParkRunning3	90,4	1,6	0,4	1,6
Média Classe A2	**91,2**	**1,9**	**0,3**	**1,4**
Média Geral	**92,0**	**2,0**	**0,3**	**1,4**

Já a Tabela 25 traz os resultados de maneira mais ampliada, permitindo observar as implicações que a implementação do modelo RF para redução de custo na predição Bidirecional trouxe para outas etapas da predição interquadros e também para outras ferramentas do codificador. Através dos resultados apresentados, é possível destacar a redução de tempo de codificação da etapa interquadros ($Inter$), que fica entre 8,4% e 12,3%, em função dos resultados expressivos de redução de tempo da predição Bidirecional (Bi). Como foco desta tese, pode-se dizer que esta otimização da etapa interquadros é significativa e corrobora para a confirmação da hipótese de pesquisa. A etapa *Affine* (Aff), por sua vez, demonstrou uma ampliação relevante no seu tempo de execução, com um tempo quase 34% maior, em média, na Classe B. Esse resultado explica o motivo dos ganhos elevados apresentados na predição Bidirecional (Bi) não terem um impacto tão positivo na redução de tempo da predição interquadros como um todo ($Inter$). Ocorre que o codificador, no processo de otimização taxa-distorção, acabou usando mais a predição *Affine* para compensar a redução de eficiência de codificação causada pelo menor uso da predição Bidirecional. Por outro lado, a etapa Unidirecional (Uni) não foi impactada de maneira significativa com a simplificação na Bidirecional, chegando a reduzir seu tempo de execução em cerca de 2% para as classes B e A2, porém com tempos piores nas classes F e A1. As outras ferramentas avaliadas foram a predição intraquadro ($Intra$) e *Merge* ($Merge$). Em am-

bas as ferramentas, as modificações na predição Bidirecional causaram, na maioria dos casos, uma ampliação no tempo de execução que chegou até a 5,4%, enquanto que, em alguns casos, aconteceram reduções no tempo, mas sempre inferiores a 1%. Neste ponto cabe destacar a ferramenta intraquadro, que com as modificações na predição Bidirecional, teve, em todos os casos, uma ampliação no seu tempo de execução que chegou até a 5,4% para o vídeo `Tango2`.

Com estes resultados é possível entender o motivo para que uma redução expressiva do tempo de execução na predição Bidirecional, traga resultados modestos de redução de tempo na predição interquadros e ainda menores na redução do tempo total. Como já explicado, na predição interquadros, a predição *Affine* tem seu tempo de execução sempre ampliado de forma significativa quando é aplicada a redução de custo computacional na predição Bidirecional. Por outro lado, no codificador completo, é a predição intraquadro que amplia seu tempo de execução quando a Bidirecional simplificada é utilizada. Assim, os resultados de redução de tempo na predição interquadros não trazem os impactos que seriam esperados na redução de tempo total. Nos dois casos, o que o ocorre é que o codificador, quando a predição Bidirecional estava desabilitada, compensou essa falta usando mais outras ferramentas, com destaque para a *Affine* e a intraquadro, para manter a eficiência de codificação.

Tabela 25 – Impacto dos modelos no tempo total e das principais ferramentas do codificador

Vídeo/ Classe	MRT						
	Bi **(%)**	Uni **(%)**	Aff **(%)**	$Inter$ **(%)**	$Intra$ **(%)**	$Merge$ **(%)**	$Total$ **(%)**
ArenaOfValor	92,5	1,4	-22,4	9,2	-1,5	0,6	2,8
BasketballDrillText	92,0	0,9	-34,7	5,3	-2,8	0,4	1,0
SlideEditing	91,8	-2,5	-15,8	19,7	-1,2	-0,2	3,9
SlideShow	93,1	-2,2	-10,9	15,1	-2,8	-0,3	3,5
Média Classe F	**92,4**	**-0,6**	**-20,9**	**12,3**	**-2,1**	**0,1**	**2,8**
BasketballDrive	95,5	1,3	-29,1	7,9	-3,4	-1,2	1,5
BQTerrace	92,7	7,7	-45,0	8,5	-1,6	0,4	2,6
Cactus	92,4	2,9	-26,9	7,9	-1,6	0,0	2,2
MarketPlace	85,7	-0,8	-38,7	6,9	-2,0	-1,1	0,5
RitualDance	90,9	-0,6	-29,4	10,6	-1,7	-0,3	1,8
Média Classe B	**91,4**	**2,1**	**-33,8**	**8,4**	**-2,1**	**-0,4**	**1,7**
Campfire	91,6	-1,3	-27,5	14,4	-3,2	-1,3	2,8
FoodMarket4	95,6	-0,5	-28,0	8,0	-3,6	-4,2	0,7
Tango2	93,1	-1,3	-28,6	7,9	-5,4	-3,6	0,7
Média Classe A1	**93,4**	**-1,0**	**-28,0**	**10,1**	**-4,1**	**-3,0**	**1,4**
CatRobot	91,9	1,5	-23,5	7,6	-3,8	-2,4	1,4
DaylightRoad2	91,1	0,7	-18,1	9,5	-3,4	-1,6	2,7
ParkRunning3	90,4	3,4	-33,5	8,2	-3,8	-0,8	1,6
Média Classe A2	**91,2**	**1,9**	**-25,0**	**8,4**	**-3,7**	**-1,6**	**1,9**
Média Geral	**92,0**	**0,7**	**-27,5**	**9,8**	**-2,8**	**-1,0**	**2,0**

Já a Tabela 26 apresenta uma análise da redução no tempo total (MRT_{Total}), o impacto na eficiência de codificação ($BDBR$), além da relação entre essas duas métricas. Pode-se afirmar que, os valores mais altos resultantes de $MRT_{Total}/BDBR$, indicam que a solução obteve significativa redução no tempo total ao custo de baixo impacto na eficiência de codificação. Ao analisar somente o $BDBR$, o mesmo pode ser considerado baixo, visto que em média, foi menor que 1% em todas as classes. Nota-se também que o $BDBR$ médio aumenta de acordo com a resolução dos vídeos de cada classe.

Analisando cada Classe de vídeos, fica evidente que a solução obteve o melhor desempenho na Classe F, cujos vídeos são de conteúdo de tela. Especialmente nos vídeos `SlideEditing` e `SlideShow` ocorrem as maiores reduções no tempo total (MRT_{Total}), de 3,9 e 3,5%, respectivamente. Uma explicação para esse resultado é que os vídeos da Classe F possuem menor complexidade de movimentos, além de menores resoluções (832x480, 1280x720 e 1920x1080), o que pode naturalmente demandar menos esforço da etapa Bidirecional.

Ao considerar as Classes B, A1 e A2 é possível notar um desempenho mais modesto. De modo geral, esses vídeos apresentam valores mais baixos de redução no tempo total de codificação e impacto de BDBR relativamente maior, o que consequentemente, atribui uma relação $MRT_{Total}/BDBR$ com valor menor. Uma possível explicação para esse desempenho é que os vídeos dessas classes possuem complexidade de nível moderado a muito alto de movimentos. Ou seja, ao desabilitar a etapa Bidirecional, o codificador é forçado a utilizar mais as etapas *Affine* e intraquadro, por exemplo. Nesse sentido, os dados da Tabela 25 demonstram que, de fato, a etapa *Affine* (MRT_{aff}) obteve os maiores percentuais de aumento no tempo de codificação nessas classes (de 25% a 33,8%). Da mesma forma, a intraquadro (MRT_{Intra}) também foi impactada, apesar dos percentuais serem menores (de 2,1% a 4,1%). Sendo assim, o fato de o codificador demandar mais da predição *Affine* e intraquadro, contribuiu para o desempenho modesto da solução nessas classes.

A seguir, a Tabela 27 apresenta uma comparação dos resultados do VTM com custo reduzido (“Com RF” na Tabela) e da versão do VTM sem a ferramenta Bidirecional (“Sem Bi” na Tabela). Essa comparação foi realizada com o objetivo de determinar um limite superior máximo em termos de redução do tempo de codificação e impacto na eficiência de codificação que qualquer solução de redução de custo focada na etapa Bidirecional poderia alcançar. Isso porque, ao desabilitar a execução de toda a ferramenta Bidirecional, espera-se alcançar uma redução significativa nos tempos de codificação, ao custo de um impacto também significativo na eficiência de codificação.

Os resultados de BDBR presentes na Tabela 27 revelam que, em todos os casos, a perda de eficiência de codificação atingida com o uso dos modelos de RF desenvolvidos foram inferiores aos da completa remoção da ferramenta Bidirecional, mas

Tabela 26 – Impacto dos modelos na eficiência de codificação

Vídeo/ Classe	MRT_{Total} **(%)**	$BDBR$ **(%)**	$MRT_{Total}/BDBR$
ArenaOfValor	2,8	0,380	7,3
BasketballDrillText	1,0	0,824	1,2
SlideEditing	3,9	-0,065	-60,1
SlideShow	3,5	0,584	6,0
Média Classe F	**2,8**	**0,260**	**-11,4**
BasketballDrive	1,5	1,060	1,4
BQTerrace	2,6	1,170	2,2
Cactus	2,2	0,548	3,9
MarketPlace	0,5	0,255	1,8
RitualDance	1,8	0,804	2,2
Média Classe B	**1,7**	**0,767**	**2,3**
Campfire	2,8	0,287	9,6
FoodMarket4	0,7	1,215	0,6
Tango2	0,7	1,466	0,5
Média Classe A1	**1,4**	**0,989**	**3,6**
CatRobot	1,4	0,690	2,1
DaylightRoad2	2,7	1,151	2,3
ParkRunning3	1,6	0,914	1,8
Média Classe A2	**1,9**	**0,918**	**2,1**
Média Geral	**2,0**	**0,752**	**-1,1**

com resultados muito próximos. Estes resultados são coerentes com a redução de tempo na etapa Bidirecional, já discutidos, que estiveram sempre acima dos 90%. Mas estes resultados também indicam que a eficiência dos modelos de aprendizado de máquina ficou abaixo do esperado, uma vez que os resultados de perda de eficiência de codificação ficaram muito próximos. Outra constatação advinda da análise deste resultado é que se confirma a hipótese anteriormente apresentada sobre o uso maior de outras ferramentas como adaptação compensatória do codificador para manter elevada a eficiência de codificação. Estes resultados apontam para a necessidade de, como trabalhos futuros, treinar modelos mais precisos para a decisão do uso ou não da Bidirecional, que permitam um corte importante no tempo de execução, mas que conduzam a resultados superiores de BDBR.

Sobre os resultados de redução de tempo, é possível perceber que o uso dos modelos RF não foi capaz de atingir resultados próximos aos do que são atingidos com a remoção total da Bidirecional. Aqui a explicação é em função de dois fatores, já discutidos: (i) os modelos trazem um *overhead* significativo no tempo de codificação e (ii) com o uso dos modelos, outras ferramentas de codificação passam a ser mais usadas.

Assim, é possível concluir que o uso da solução de aprendizado de máquina para reduzir o tempo de processamento da Bidirecional gera uma redução de tempo não

Tabela 27 – Comparação do VTM com redução de custo e do VTM sem a predição Bidirecional

Vídeo/ Classe	MRT_{Total}		$BDBR$	
	Com RF (%)	Sem BI (%)	Com RF (%)	Sem BI (%)
ArenaOfValor	2,8	7,3	0,380	0,456
BasketballDrillText	1,0	5,3	0,824	1,007
SlideEditing	3,9	8,4	-0,065	0,056
SlideShow	3,5	6,3	0,584	0,925
Média Classe F	**2,8**	**6,9**	**0,260**	**0,611**
BasketballDrive	1,5	5,4	1,060	1,083
BQTerrace	2,6	6,5	1,170	1,118
Cactus	2,2	7,1	0,548	0,585
MarketPlace	0,5	5,9	0,255	0,348
RitualDance	1,8	6,5	0,804	0,820
Média Classe B	**1,7**	**6,3**	**0,767**	**0,791**
Campfire	2,8	9,1	0,287	0,303
FoodMarket4	0,7	4,7	1,215	1,382
Tango2	0,7	5,6	1,466	1,494
Média Classe A1	**1,4**	**6,5**	**0,989**	**1,060**
CatRobot	1,4	6,3	0,690	0,950
DaylightRoad2	2,7	7,9	1,151	1,218
ParkRunning3	1,6	8,0	0,914	1,051
Média Classe A2	**1,9**	**7,4**	**0,918**	**1,073**
Média Geral	**2,0**	**6,7**	**0,752**	**0,853**

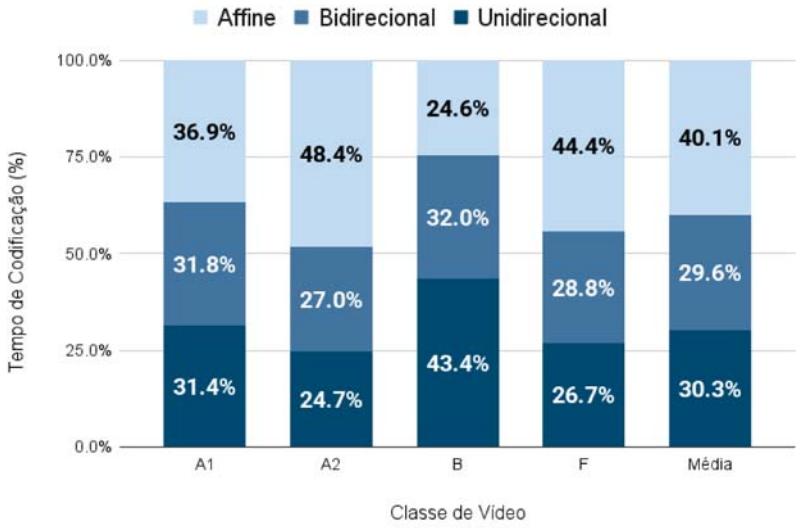
muito significativa e a perda na eficiência de codificação fica próxima à versão sem a Bidirecional. Apesar disso, é importante salientar que a solução com RFs se manteve inserida dentro dos limites de resultados apresentados pela versão com a Bidirecional removida, tanto no que diz respeito à redução de tempo de execução, quanto à perda na eficiência de codificação.

As possibilidades que explicam esse desempenho podem estar relacionadas às restrições aplicadas aos 12 tamanhos de bloco suportados pela *Affine* (de 16×16 a 128×128), que acabam sendo testados cerca de 5% das vezes pela Bidirecional, e o restante, fica a cargo da Unidirecional e da *Affine*. Nesse sentido, é necessário retomar a análise já demonstrada no capítulo 5, especificamente na Figura 32, em que é visível o elevado tempo de codificação desses blocos na etapa *Affine* mesmo com a etapa Bidirecional sendo executada normalmente. Assim, ao ignorar a etapa Bidirecional cerca de 95% das vezes e aumentar significativamente o tempo de execução da *Affine*, a solução de RFs não foi capaz de atingir reduções expressivas de tempo total.

Outra possibilidade é que alterações presentes na versão 22.0 do VTM, realizadas pelo JVET (*Joint Video Experts Team*), tenham feito com que a etapa Bidirecional não tivesse mais tanta relevância no tempo total de codificação quanto aquela observada na análise presente no capítulo 5, que considerou a versão VTM 16.2. Assim,

um novo experimento foi realizado com o VTM 22.0 para avaliar essa hipótese. A Figura 51 apresenta o resultado deste experimento, que usou exatamente as mesmas configurações do experimento apresentado na Figura 20 do capítulo 5, mas agora na versão 22.0 do VTM. Esse gráfico comprova a hipótese, pois o percentual de tempo total ocupado pela Bidirecional no VTM 22.0 original não ultrapassa 32%. Por outro lado, ao comparar as três ferramentas da interquadros, a *Affine* se destaca como a que ocupa o maior percentual de tempo em relação ao tempo total de codificação. Além disso, conforme apresentado na Tabela 27 os baixos valores de BDBR (menores que 1%) obtidos ao se retirar a ferramenta Bidirecional no VTM 22.0 demonstram que a mesma tem pouco impacto na eficiência de codificação. Ambos os resultados em conjunto comprovam que, na versão 22.0 do VTM, a Bidirecional é muito menos relevante para o codificador do que na versão 16.2 do VTM.

Figura 51 – Percentual do tempo de codificação das etapas ME do VTM Original



Fonte: Elaborada pelo autor.

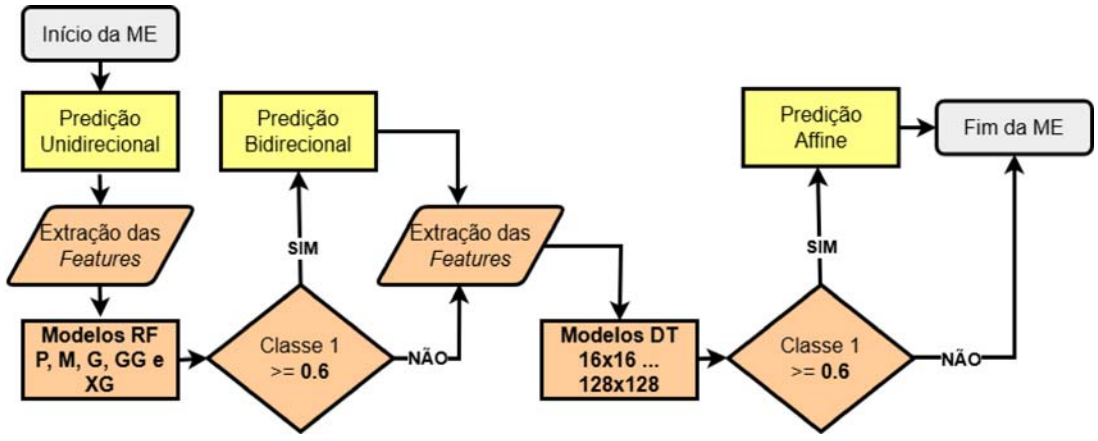
Apesar de não terem sido abordados trabalhos diretamente relacionados à solução proposta neste capítulo, os resultados podem ser comparados com o trabalho de Lee e Jun (2023), cujo foco é a ferramenta BCW, que está contida na Bidirecional. O trabalho de Lee e Jung (2023), é aplicado na versão 11.0 do VTM, anterior a versão 22.0 utilizada neste trabalho. Os autores utilizam uma Rede Neural com multicamadas *perceptrons*, ou seja, uma técnica de aprendizado profundo, para acelerar a ferramenta BCW. Como resultado, o trabalho atinge 33% de redução de tempo na ferramenta BCW, naturalmente, sem considerar o tempo necessário para o processamento da rede neural. O impacto em BDBR foi de 0,26%. A comparação com esses valores demonstra que a solução proposta neste capítulo é competitiva em termos de redução no tempo de codificação, alcançando 92% em média de redução na etapa Bidirecional. Mesmo o impacto de 0,75% em BDBR sendo maior, é preciso conside-

rar que a versão do VTM 22.0 é mais recente e apresenta diversas modificações e otimizações ao comparar-se com a versão 11.0.

9 REDUÇÃO DE CUSTO COMPUTACIONAL DAS PREDIÇÕES BIDIRECIONAL E *AFFINE* USANDO APRENDIZADO DE MÁQUINA

Este capítulo apresenta a solução de otimização para as etapas Bidirecional e *Affine* da predição interquadros de forma conjunta, utilizando modelos baseados em *Decision Trees* (Árvores de Decisão). O objetivo é ampliar a otimização apresentada no Capítulo 8, obtendo redução no tempo de codificação total e na etapa interquadros, com baixo impacto na eficiência de codificação. Nessa solução, optou-se por não utilizar modelos baseados em RFs para a otimização ampliada da *Affine*. Ao invés disso, foram desenvolvidos modelos baseados em DTs, para não impor mais custo computacional à solução ampliada, visto que os modelos RFs utilizados na otimização da Bidirecional, descritos no capítulo anterior, apresentaram essa característica.

Figura 52 – Solução de otimização aplicada à predição Bidirecional e *Affine*



Fonte: Elaborada pelo autor.

Conforme ilustrado na Figura 52, esta solução ampliada utiliza os modelos de RFs desenvolvidos para otimizar a Bidirecional, extraindo novas *features* após sua execução. Então, essas *features* são utilizadas como entrada para 1 dos 12 modelos de *Decision Tree* (DT), treinados previamente (*off-line*), que retorna se a predição *Affine* deve ou não ser executada para o bloco atual. É importante destacar que a solução com DTs considera “executar *Affine*” como Classe 1 e “não executar *Affine*” como

Classe 0. Além disso, também foi considerado o ponto de corte de 0,6 para a Classe 1, mantendo a equivalência com a solução de RFs para a Bidirecional. Dessa maneira, a predição *Affine* é executada somente se a pontuação retornada pelo modelo DT for maior ou igual a 0,6. Os 12 modelos DT foram definidos conforme os tamanhos de blocos que são suportados pela etapa *Affine*, ou seja, os blocos com altura e largura maiores que oito amostras de luminância (de 16×16 até 128×128).

Assim como na otimização focada na etapa Bidirecional, os experimentos relacionados à solução ampliada foram realizados em um servidor com processador Intel Xeon Gold 5118 2,30 GHz com 56 GB de RAM. Da mesma forma, o treinamento dos modelos DT foi realizado a partir da linguagem Python, com a biblioteca `scikit-learn` (PEDREGOSA et al., 2011). A biblioteca `m2cgen` (Zeigerman, 2022) foi novamente utilizada para a exportação dos modelos DT para a linguagem C++, possibilitando sua implementação no VTM 22.0.

9.1 Extração de Dados

Para a extração das *features* da solução ampliada foram utilizados nove vídeos das bases de dados UVG (Mercat; Viitanen; Vanne, 2020) e NETVC (Daede; Norkin; Brailovskiy, 2019). O número de vídeos usados nesta fase foi reduzido (de 15 para 9) a fim de agilizar o processo de extração e obter arquivos menores. Dessa forma, foram descartados dois vídeos de cada resolução, sendo um que possuía o valor mais baixo de TI (pouca informação temporal) e o outro cuja relação SI/TI fosse mais baixa (pouca informação espacial em relação à temporal). Esses critérios foram utilizados visando descartar vídeos com características mais extremas, ou seja, pouco movimento ou baixa relação entre textura e movimento, garantindo que as *features* extraídas fossem mais relevantes para o modelo. Sendo assim, as sequências utilizadas para a extração das *features* foram: *Dark*, *Netflix DrivingPOV* e *Vidyo4* (HD, resolução 1280×720), *Netflix TunnelFlags*, *Jockey* e *Touchdown Pass* (FHD, resolução 1920×1080), *ToddlerFountain*, *SunBath* e *Lips* (4K UHD resolução 3840×2160). Esses vídeos foram codificados quatro vezes, uma para cada QP (22, 27, 32 e 37), com seus primeiros 16 quadros, usando a versão 22.0 do VTM na configuração *Random Access*.

A seguir, a Tabela 28 apresenta as *features* extraídas após a execução da etapa Bidirecional, já considerando a otimização com RFs. Nota-se que a maior parte das *features* são iguais às da solução anterior, pois são dados básicos sobre o bloco que está sendo testado no momento da extração. Adicionalmente, foram extraídas outras informações, com base em trabalhos de pesquisa que também focam na otimização da etapa *Affine* ou pela análise do código do VTM. As *features* relacionadas à profundidade de particionamento do bloco (depth, qtDepth e mtDepth) foram escolhidas com

base no trabalho de (Gonçalves, 2021). Além dos dados sobre blocos vizinhos, os valores específicos extraídos da etapa Bidirecional, tais como vetores de movimento (mv_bi e mv_pred_bi), custo (cost_mv_bi), número de bits (bits_mv_bi), uso da ferramenta SMVD e direção do movimento (interDir) também foram utilizados no trabalho de (Sagrilo; Loose; Viana; Sanchez; Corrêa; Agostini, 2023). Já as *features* checkAffine e affineModeSelected estão presentes no código do VTM, mais especificamente, no teste original que decide se a etapa *Affine* será ou não executada.

9.2 Elaboração dos *datasets*

Diferentemente da solução para a Bidirecional (Capítulo 8), nesta otimização ampliada optou-se pela utilização de *datasets* separados para os 12 tamanhos de bloco suportados pela predição *Affine*. Essa decisão se baseia no fato de que outras organizações de *datasets* provavelmente apresentariam desempenhos muito semelhantes, conforme já demonstrado na seção 8.2.2 da otimização anterior. Sendo assim, para acelerar o processo de desenvolvimento da solução, optou-se pela organização com 12 *datasets* distintos, um para cada tamanho de bloco (a partir de 16×16 até 128×128).

Com as *features* extraídas, realizou-se o balanceamento dos dados de maneira semelhante à otimização do capítulo 8. Para isso, também foram considerados 1.000 exemplos para cada combinação de vídeo, QP, tamanho de bloco e decisão. O objetivo foi igualar o número mínimo de exemplos da solução com RFs para a Bidirecional, visando maior eficiência na etapa de treinamento do modelo. Assim, cada um dos nove vídeos inclui 1.000 exemplos da decisão 0 (não executar a *Affine*) e 1.000 exemplos da decisão 1 (executar a *Affine*) para cada um dos quatro QPs e em cada um dos 12 tamanhos de bloco. Dessa maneira, foram selecionados 864.000 exemplos do total extraído do processo de codificação, sendo 432.000 para cada decisão.

Porém, novamente enfrentou-se o problema de não existirem 1.000 exemplos para alguns vídeos e QPs. Nesse caso, a falta de exemplos ocorreu em sete dos 12 tamanhos, sem ocorrências de faltas extremas ou zeradas. Por exemplo, o menor número disponível é de 380 exemplos e acontece com a decisão 1 para o bloco 16×16, no QP 37 da sequência `Vidyo4`. Além disso, é importante citar que, no geral, apenas os registros da decisão 1 apresentaram esse problema, porém é mais frequente nos blocos 128×128, 64×128 e 128×64, ocorrendo em pelo menos um QP de todos os vídeos. Já para os blocos 64×16, 16×64, 32×16 e 16×16, o problema aparece em apenas um QP na resolução HD. Mesmo assim, foi possível contornar a falta de exemplos e garantir o balanceamento utilizando a mesma estratégia descrita na seção 8.2, com os critérios: $i)$ em que serão selecionados exemplos extras de outros vídeos da mesma resolução e QP; e, caso necessário $ii)$ os exemplos extras serão obtidos de vídeos de outra resolução, porém do mesmo QP. A tabela completa com o número

Tabela 28 – Relação das *Features* extraídas após a etapa Bidirecional

Feature	**Qtd**	**Descrição**
tamBlocoWidth	1	Largura do bloco em píxeis
tamBlocoHeight	1	Altura do bloco em píxeis
numQP	4	Parâmetro de quantização de entrada
widthVideo	1	Largura do vídeo em píxeis
heightVideo	1	Altura do vídeo em píxeis
cu_pos	2	Posição X e Y do bloco dentro do quadro
depth	1	Profundidade do bloco no particionamento QTMTT
qtDepth	1	Profundidade do bloco no particionamento QT
mtDepth	1	Profundidade do bloco no particionamento MTT
indiceBcw	5	Índice relacionado ao peso usado na BCW
imv	4	Precisão do Vetor de Movimento (AMVR)
atual_QP	1	Parâmetro de quantização utilizado no momento
mv_uni	4	Vetor de Movimento (X, Y) gerado pela predição Unidirecional para a Lista 0 e Lista 1
mv_pred_uni	4	Vetor de Movimento (X, Y) gerado pela predição AMVP Unidirecional para a Lista 0 e Lista 1
cost_mv_uni	2	Custo do Vetor de Movimento gerado pela predição Unidirecional para a Lista 0 e Lista 1
bits_mv_uni	2	Quantidade de bits do Vetor de Movimento gerado pela predição Unidirecional para a Lista 0 e Lista 1
mv_bi	4	Vetor de Movimento (X, Y) gerado pela predição Bidirecional para a Lista 0 e Lista 1
mv_pred_bi	4	Vetor de Movimento (X, Y) gerado pela predição AMVP Bidirecional para a Lista 0 e Lista 1
cost_mv_bi	1	Custo do Vetor de Movimento gerado pela predição Bidirecional
bits_mv_bi	1	Quantidade de bits do Vetor de Movimento gerado pela predição Bidirecional
SMVD	1	Indica se o SMVD está sendo testado ou não
interDir	1	Indica a direção da predição
checkAffine	1	Indica se *Affine* deve ou não ser testada, de acordo com o valor atual do imv
affineModeSelected	1	Indica se o modo *Affine* já foi selecionado
bidirecional_pai	1	Indica se o bloco pai foi codificado pela predição Bidirecional
custo_pai	1	Custo gerado pela predição Bidirecional do bloco pai
IMV_pai	5	Valor do IMV do bloco pai
affine_pai	1	Indica se o bloco pai foi codificado pela predição *affine*
custo_affine_pai	1	Custo gerado pela predição *Affine* do bloco pai
bidirecional_viz_esq	1	Indica se o bloco vizinho a esquerda foi codificado pela predição Bidirecional
custo_viz_esq	1	Custo gerado pela predição do bloco vizinho a esquerda
IMV_viz_esq	5	Valor do IMV do bloco vizinho a esquerda

affineVizEsq	1	Indica se o bloco vizinho à esquerda foi codificado pela predição *Affine*
custo_affine_VizEsq	1	Custo gerado pela predição *Affine* do bloco vizinho à esquerda
bidirecional_viz_acima	1	Indica se o bloco vizinho acima foi codificado pela predição Bidirecional
custo_viz_acima	1	Custo gerado pela predição do bloco vizinho acima
IMV_viz_acima	5	Valor do IMV do bloco vizinho acima
affineVizAci	1	Indica se o bloco vizinho acima foi codificado pela predição *Affine*
custo_affine_acima	1	Custo gerado pela predição *Affine* do bloco vizinho acima
sum	1	Somatório das amostras do bloco atual
media	1	Média das amostras do bloco atual
vari	1	Variância das amostras do bloco atual
desvioPadrao	1	Desvio padrão das amostras do bloco atual
grad	2	Gradientes Horizontal e Vertical do bloco atual (Sobel)
razao_grad	1	Gradiente Horizontal divido pelo Gradiente Vertical
grad_razao_píxeis	1	Soma dos gradientes divido pelo número de píxeis
dist_uni_L0_L1	1	Distância entre os Vetores de Movimento resultantes da predição Unidirecional da Lista 0 e Lista 1
bloco_atual_affine	1	Indica se o bloco atual possui o melhor modo de predição como sendo *Affine*

de exemplos selecionados por vídeo, QP e tamanho de bloco está no Apêndice B.

9.3 Treinamento dos modelos

Com base nos 12 *datasets* definidos e balanceados, foi realizada uma série de experimentos, incluindo a busca pelos melhores hiperparâmetros e a seleção das *features* relevantes, antes do treinamento final e implementação dos modelos. Esses processos serão detalhados a seguir.

9.3.1 Busca de Hiperparâmetros

Nesta etapa da busca de hiperparâmetros cada um dos 12 *datasets* foi dividido em treino e teste, sendo que as proporções são de 75% e 25% dos registros, respectivamente. Dessa forma, cada *dataset* terá o total de 72.000 registros, sendo 54.000 utilizados para o treinamento do modelo e 18.000 utilizados para o teste.

No primeiro momento, a busca foi baseada no método *Random Search*, que treinou e testou 500 combinações aleatoriamente definidas a partir das faixas de valores determinadas para cada hiperparâmetro. Na Tabela 29, são apresentadas as faixas de valores estabelecidas para os hiperparâmetros ***criterion***, ***min_samples_split***, ***min_samples_leaf***, ***max_features***, ***max_depth*** e ***max_leaf_nodes***. Essas faixas

de valores são inspiradas nos valores dos trabalhos de Duarte (2021) e Andrades; Grellert; Fonseca (2019). A representação das faixas de valores na tabela considera o valor inicial, final e a iteração (ou "passo"). Por exemplo, o hiperparâmetro ***min_samples_split*** foi testado com o valor padrão 2 e a faixa de valores entre 25 e 500, variando a cada 25 (25, 50, 75, 100, etc).

Tabela 29 – Faixas de valores utilizadas no *Random Search*

Hiperparâmetro	Faixa de valores
criterion	[gini, entropy]
min_samples_split	[2] OU [25..500, 25]
min_samples_leaf	[1] OU [10..100, 10]
max_features	[sqrt] OU [1..total]
max_depth	[2..10] OU [20..100 10]
max_leaf_nodes	[2..10] OU [20..700, 10]

Na Tabela 30 são apresentados os valores resultantes do processo *Random Search*, cuja combinação dos hiperparâmetros gerou o melhor resultado para F1-score em cada modelo. Os hiperparâmetros destacados são os que obtiveram a maior correlação de seu valor com o F1. O número de *features* (**max_features**) se destaca em todos os modelos, seguido pela profundidade máxima da árvore (**max_depth**) que tem maior correlação em oito dos 12 modelos.

Tabela 30 – Melhor Resultado do *Random Search* por modelo

Hiperparâmetro / Modelo	*criterion*	*min_samples_split*	*min_samples_leaf*	*max_features*	*max_depth*	*max_leaf_nodes*	F1-score
16×16	entropy	150	20	49	100	420	**0,95**
16×32	entropy	150	90	45	8	440	**0,96**
32×16	gini	425	40	65	70	20	**0,96**
32×32	gini	75	70	66	40	510	**0,96**
16×64	entropy	100	20	71	100	80	**0,96**
64×16	entropy	150	20	57	30	510	**0,96**
32×64	entropy	25	10	54	30	80	**0,97**
64×32	entropy	50	90	67	20	110	**0,97**
64×64	entropy	25	10	72	10	230	**0,98**
64×128	gini	150	20	55	80	570	**0,98**
128×64	entropy	125	70	60	60	100	**0,98**
128×128	entropy	50	40	64	8	500	**0,98**

Com base nesses resultados, foi possível realizar a segunda etapa de busca de hiperparâmetros, usando o método *Grid Search*. De maneira semelhante à otimização anterior, os hiperparâmetros que retornaram maior correlação com o F1 no *Random Search* tiveram mais possibilidades exploradas no *Grid Search*, conforme apresenta

a Tabela 31. Os demais hiperparâmetros foram testados com os valores padrão do modelo `DecisionTree` da biblioteca `sklearn` e o valor retornado no *Random Search*. Essas definições têm o objetivo de encontrar a melhor combinação de valores possível para os hiperparâmetros que mais impactam no resultado dos modelos e foram realizadas de maneira semelhante ao indicado em Duarte (2021) e Andrades; Grellert; Fonseca (2019). É importante destacar que, ao utilizar apenas duas possibilidades para os demais hiperparâmetros (com menor correlação), a ideia é não impor um longo tempo de processamento, visto que o *Grid Search* realiza o treinamento e teste de todas as combinações de valores utilizando a estratégia de validação cruzada com cinco conjuntos. Nesse sentido, pode-se citar que os modelos 16×16 e 16×64 tiveram o maior e menor número de candidatos, respectivamente, com 112.000 e 23.040 testes.

Tabela 31 – Faixas de valores utilizadas no *Grid Search*

Hiperparâmetro / Modelo	***criterion***	***min_samples_split***	***min_samples_leaf***	***max_features***	***max_depth***	***max_leaf_nodes***
16×16	[entropy, gini]	[2, 150]	[1, 20]	[sqrt] OU [1..49, 2]	[1..9] OU [10..100, 5]	[none, 420]
16×32	[entropy, gini]	[2, 150]	[1, 90]	[sqrt] OU [1..45, 2]	[8]	[10..440, 10]
32×16	[gini]	[2, 425]	[1, 40]	[sqrt] OU [1..65, 2]	[1..9] OU [10..70, 5]	[none, 20]
32×32	[gini]	[2, 75]	[1, 70]	[sqrt] OU [1..66, 2]	[40]	[10..510, 10]
16×64	[entropy, gini]	[2, 100]	[1, 20]	[sqrt] OU [1..71, 2]	[100]	[10..80, 10]
64×16	[entropy, gini]	[2, 150]	[1, 20]	[sqrt] OU [1..57, 2]	[1..9] OU [10..30, 5]	[none, 510]
32×64	[entropy, gini]	[2, 25]	[1, 10]	[sqrt] OU [1..54, 2]	[1..9] OU [10..30, 5]	[none, 80]
64×32	[entropy, gini]	[2, 50]	[1, 90]	[sqrt] OU [1..67, 2]	[1..9] OU [10..20, 5]	[none, 110]
64×64	[entropy, gini]	[2, 25]	[1, 10]	[sqrt] OU [1..72, 2]	[1..10]	[none, 230]
64×128	[gini]	[2, 150]	[1, 20]	[sqrt] OU [1..55, 2]	[80]	[10..570, 10]
128×64	[entropy, gini]	[2, 125]	[1, 70]	[sqrt] OU [1..60, 2]	[1..9] OU [10..60, 5]	[none, 100]
128×128	[entropy, gini]	[2, 50]	[1, 40]	[sqrt] OU [1..64, 2]	[1..8]	[none, 500]

A partir da aplicação do método *Grid Search* em cada modelo, são obtidas as combinações dos hiperparâmetros que geram o maior F1-score. A Tabela 32 apresenta esses valores para cada modelo. É importante destacar que, ao contrário da solução

anterior, não foi necessário nenhum ajuste nos valores, visto que estão sendo usados modelos DT, com menor custo computacional. Os valores apresentados na Tabela 32 foram utilizados no treinamento final de cada modelo.

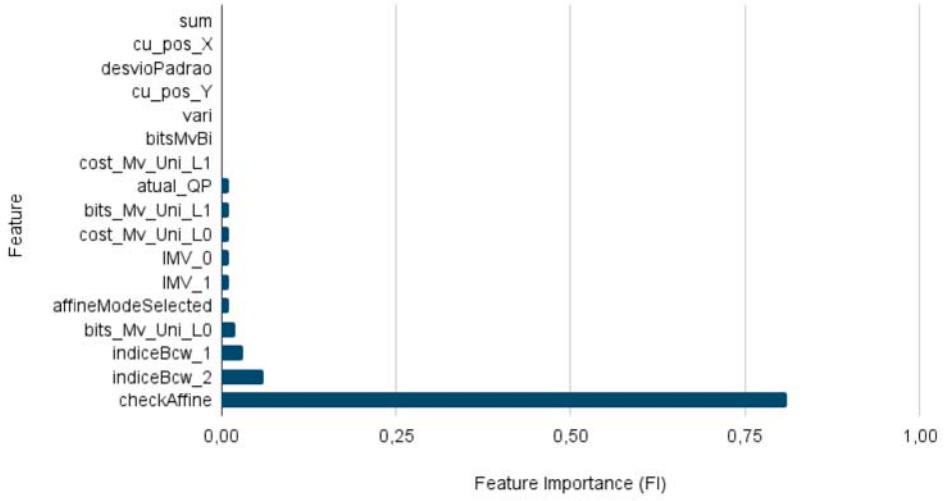
Tabela 32 – Hiperparâmetros utilizados para o treinamento final

Hiper-parâmetro / Modelo	*criterion*	*min_ samples _split*	*min_ samples _leaf*	*max_ features*	*max_ depth*	*max_ leaf_ nodes*	F1-score
16×16	entropy	150	20	38	40	420	0,95
16×32	gini	150	90	42	8	280	0,96
32×16	gini	2	1	63	35	20	0,96
32×32	gini	2	1	46	40	40	0,96
16×64	entropy	2	20	61	100	60	0,96
64×16	entropy	150	20	49	10	510	0,96
32×64	entropy	25	10	43	10	80	0,97
64×32	gini	50	90	49	20	110	0,97
64×64	entropy	25	10	63	10	230	0,98
64×128	gini	2	1	49	80	50	0,98
128×64	gini	2	1	52	55	100	0,98
128×128	gini	2	40	44	8	500	0,98

9.3.2 Seleção das *features*

Na sequência da seleção dos hiperparâmetros utilizados para o treinamento, foi realizada a seleção das *features*, a fim de simplificar os modelos finais. As Figuras de 53 a 64 ilustram em formato gráfico as 17 *features* mais relevantes para cada modelo. Essa quantidade de *features* foi definida a partir do resultado de FI para o modelo 64×64, que possui 17 features com valores de FI a partir de 0,1%. Dessa maneira, além da padronização, os gráficos permitem uma melhor visualização dos resultados.

Figura 53 – *Feature Importance* para o Modelo 16×16

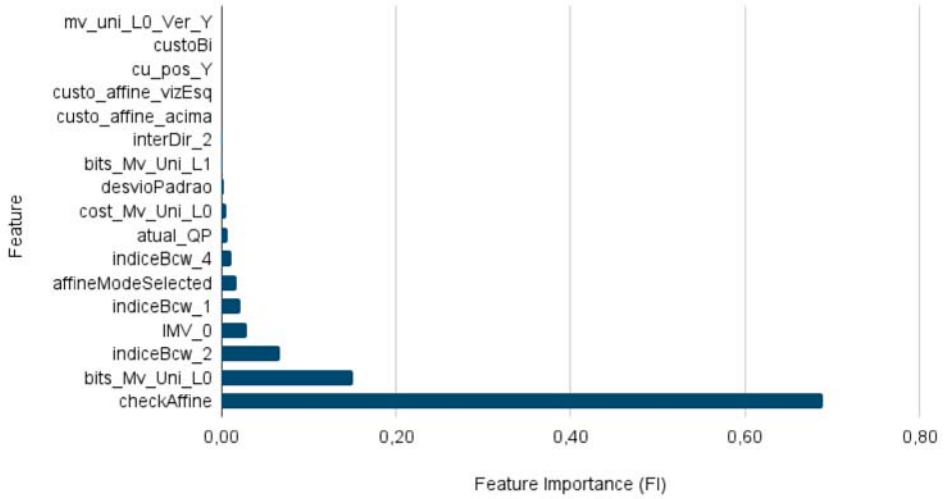


Fonte: Elaborada pelo autor.

Na Figura 53 é possível notar que `checkaffine` é a *feature* com maior influência nas decisões do modelo 16×16. Seu peso é significativamente alto, com 0,81, em contraste com as *features* seguintes, como `indiceBcw_2` e `indiceBcw_1`, que têm apenas 0,06 e 0,03 de FI.

No modelo 16×32, conforme apresentado na Figura 54, `checkAffine` também se destaca como a *feature* de maior importância para a decisão, com um valor de 0,69. Já as *features* `bits_Mv_Uni_L0` e `indiceBcw_2` apresentam valores significativamente menores, 0,15 e 0,07, respectivamente. Nota-se também que a maioria das demais *features* possuem valores próximos a 0,0, ou seja, baixa importância para as decisões do modelo.

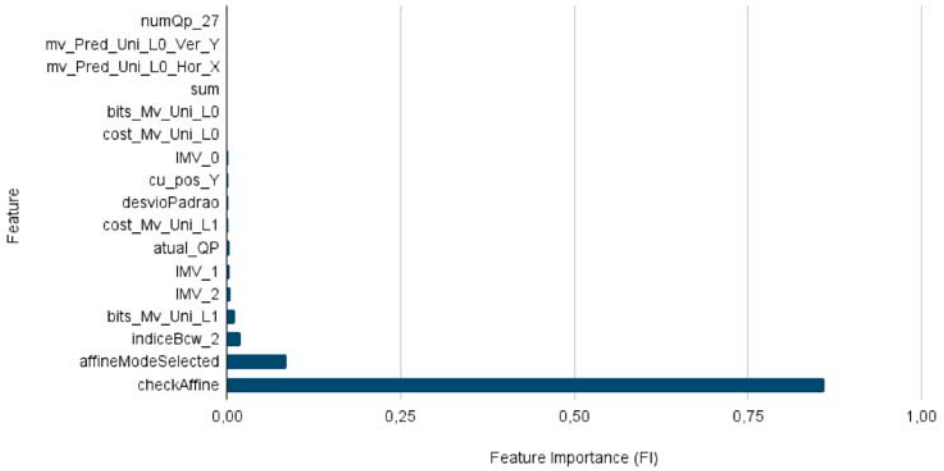
Figura 54 – *Feature Importance* para o Modelo 16×32



Fonte: Elaborada pelo autor.

A Figura 55 mostra que para o modelo 32×16 a *feature* `checkAffine` novamente se destaca com uma importância significativa de 0,86. A segunda *feature* mais relevante é `affineModeSelected`, porém com um peso de apenas 0,09. Já as demais *features* têm impacto praticamente nulo.

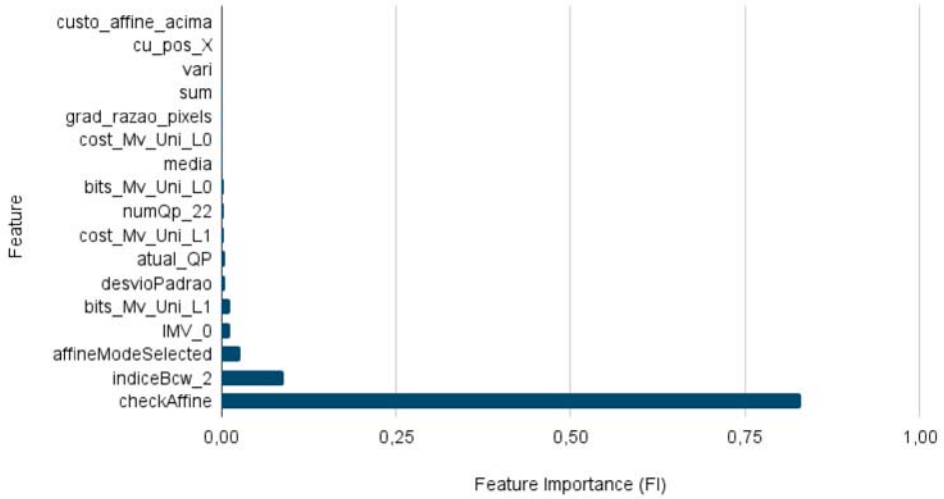
Figura 55 – *Feature Importance* para o Modelo 32×16



Fonte: Elaborada pelo autor.

De maneira semelhante, a Figura 56 demonstra que no modelo 32×32, a *feature* `checkAffine` mais uma vez domina a importância nas decisões do modelo, dessa vez com um impacto de 0,83. Na sequência, com baixa contribuição estão as *features* `indiceBcw_2` (0,09) e `affineModeSelected` (0,03).

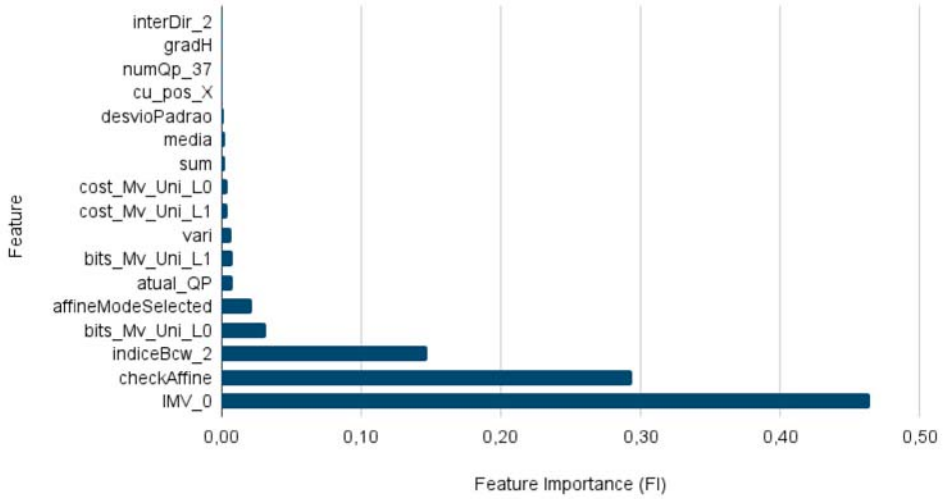
Figura 56 – *Feature Importance* para o Modelo 32×32



Fonte: Elaborada pelo autor.

Em contraste com os resultados de FI apresentados até então, o modelo 16×64 (Figura 57) possui a *feature* `checkAffine` como segunda mais relevante, com 0,29. Nesse modelo, a *feature* `IMV_0` se destaca com um maior impacto nas decisões, com 0,46. Também é possível notar que a *feature* `indiceBcw_2` é significativa para o modelo, apresentando um valor de 0,15. Esse comportamento revela um maior equilíbrio entre as *features* que mais influenciam o desempenho do modelo, diferentemente dos casos anteriores.

Figura 57 – *Feature Importance* para o Modelo 16×64

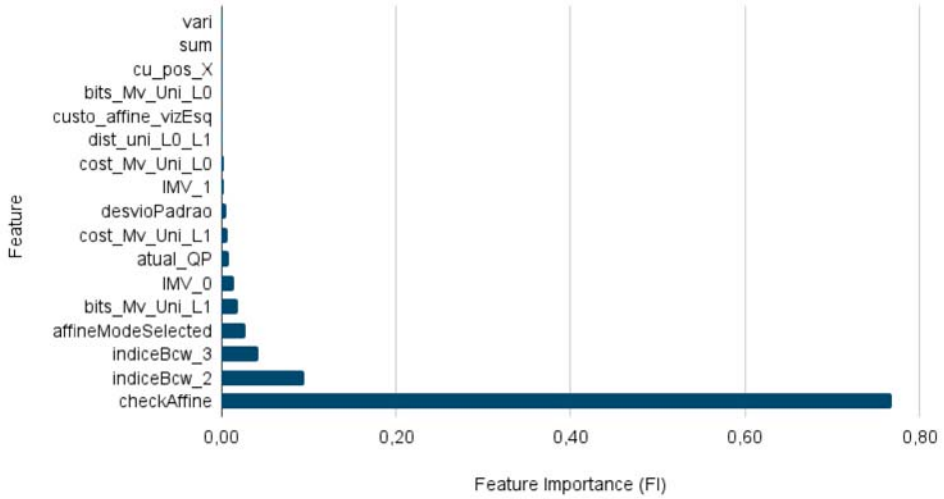


Fonte: Elaborada pelo autor.

A Figura 58 ilustra novamente o padrão de domínio da *feature* `checkAffine` (0,77),

desta vez no modelo 64×16. Com pesos significativamente menores, `indiceBcw_2` , com 0,09, e `affineModeSelected`, com 0,03, também impactam as decisões do modelo, mas moderadamente.

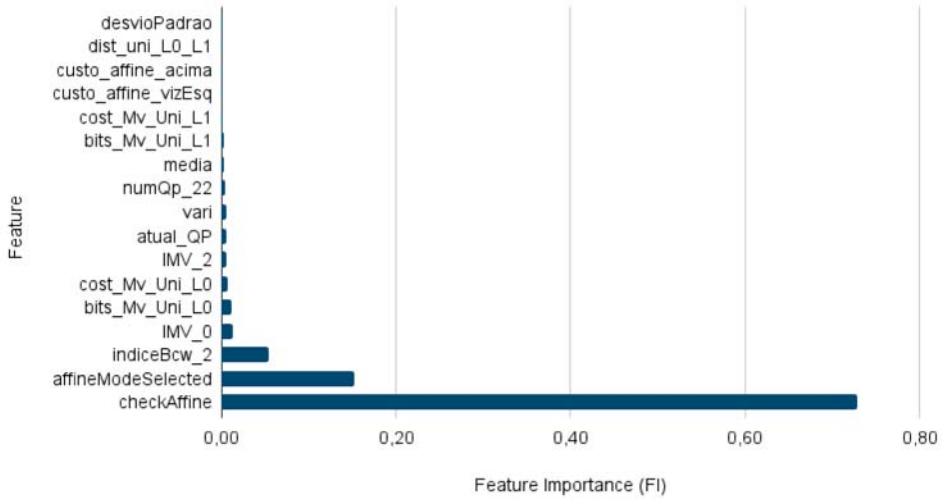
Figura 58 – *Feature Importance* para o Modelo 64×16



Fonte: Elaborada pelo autor.

Conforme mostra a Figura 59, o modelo 32×64 também é mais influenciado pela *feature* `checkAffine`, que obteve um peso de 0,73. De maneira secundária, as *features* `affineModeSelected` e `indiceBcw_2` aparecem com os valores de 0,15 e 0,05, respectivamente.

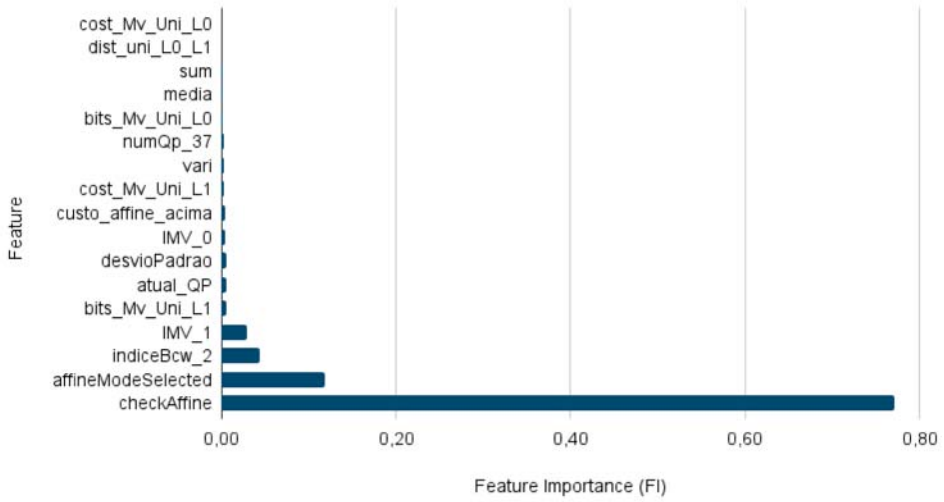
Figura 59 – *Feature Importance* para o Modelo 32×64



Fonte: Elaborada pelo autor.

É possível perceber na Figura 60 que o modelo 64×32 reflete um padrão semelhante ao de outros modelos, com `checkAffine` alcançando um peso de 0,77. Novamente, de maneira secundária, as *features* `affineModeSelected` (0,12) e `indiceBcw_2` (0,04) aparecem com os valores 0,12 e 0,04, respectivamente.

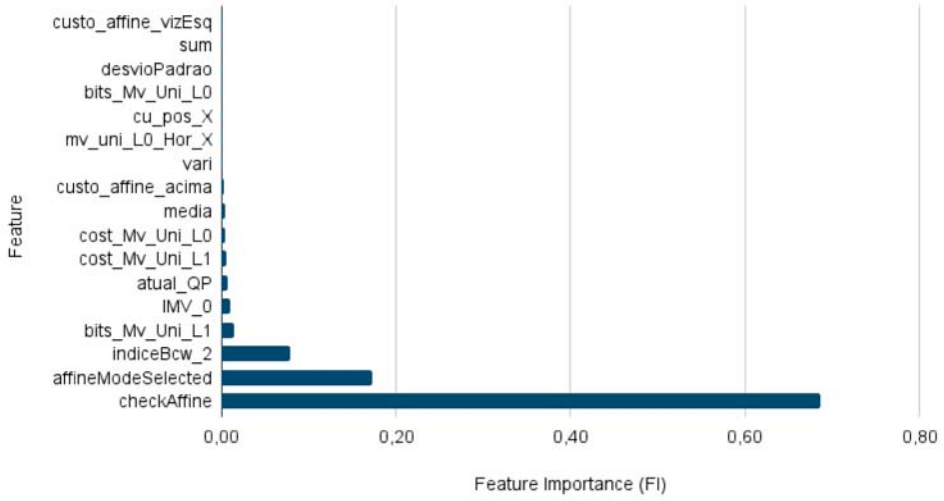
Figura 60 – *Feature Importance* para o Modelo 64×32



Fonte: Elaborada pelo autor.

O modelo 64×64, representado na Figura 61, mostra as mesmas três *features* descritas anteriormente, porém com pequenas variações nos pesos. A *feature* `checkAffine` novamente lidera, com 0,69, seguido de `affineModeSelected` (0,17) e `indiceBcw_2` (0,08).

Figura 61 – *Feature Importance* para o Modelo 64×64

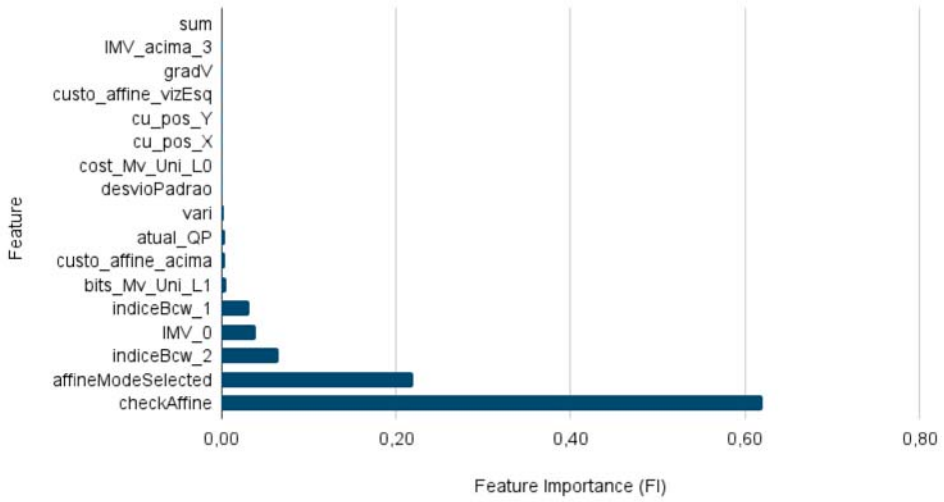


Fonte: Elaborada pelo autor.

Já para o modelo 64×128, a Figura 62 demonstra um destaque maior para as *features* `checkAffine` (0,62) e `affineModeSelected` (0,22). Mas também é possível visualizar um impacto, mesmo baixo, das *features* `indiceBcw_2`, com 0,07, e `IMV_0`, com 0,04. É possível considerar esta distribuição de FI levemente mais equilibrada em comparação aos demais modelos.

No modelo 128×64 (Figura 63), apesar de *checkAffine* continuar com a maior importância nas decisões, com 0,61, outras três *features* também contribuem. Os valores das *features* `indiceBcw_1` (0,14), `affineModeSelected` (0,11) e *indiceBcw_2*

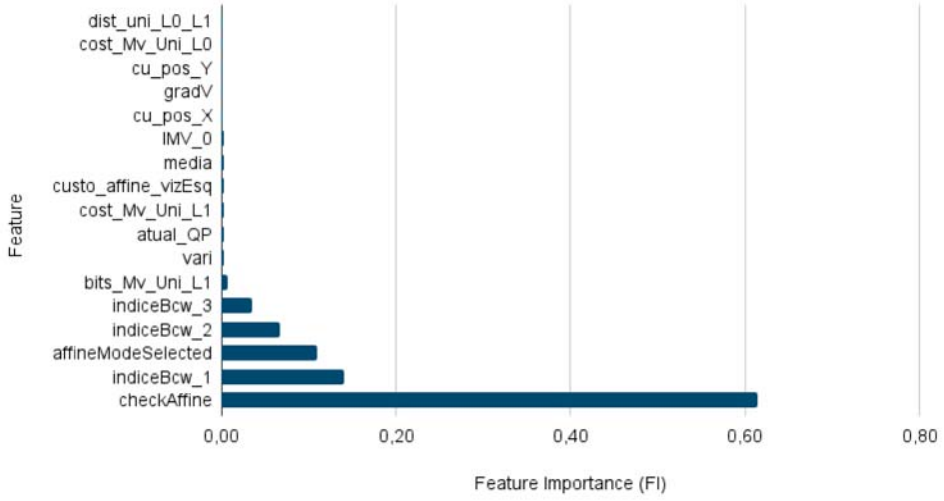
Figura 62 – *Feature Importance* para o Modelo 64×128



Fonte: Elaborada pelo autor.

(0,07) indicam que as mesmas equilibram entre si o peso secundário nas decisões do modelo.

Figura 63 – *Feature Importance* para o Modelo 128×64

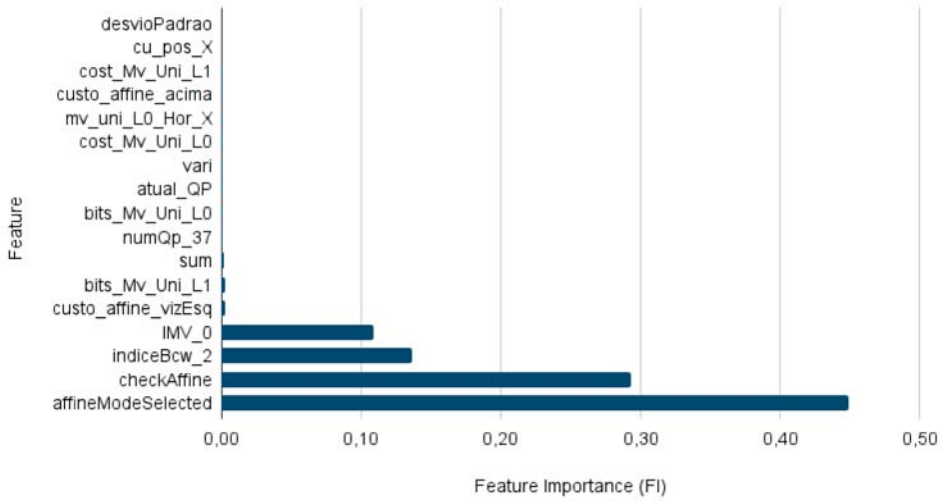


Fonte: Elaborada pelo autor.

Por fim, o modelo 128×128, representado pela Figura 64, possui uma distribuição mais equilibrada dos valores de FI entre quatro *features*. Em contraste com a maioria dos modelos apresentados, neste caso `affineModeSelected` é a *feature* mais relevante, com 0,45, seguida por `checkAffine` (0,29), *indiceBcw_2* (0,14) e *IMV_0* (0,11). Essa distribuição indica uma leve diferenciação no comportamento deste modelo, quando comparado aos demais.

Em geral, analisando os valores de FI para os 12 modelos, fica evidente a importância da *feature* `checkAffine`, que aparece com valores significativos na maior parte dos mesmos. Também é possível destacar a influência secundária das *features* `affineModeSelected` e `indiceBcw_2`, que frequentemente aparecem nos modelos,

Figura 64 – *Feature Importance* para o Modelo 128×128



Fonte: Elaborada pelo autor.

indicando sua importância de maneira geral. Há outras features que, de maneira variável, aparecem em destaque em alguns modelos, tais como `IMV_0` e `indiceBcw_1`. Nota-se que os modelos dependem de poucas *features* para as decisões, sendo que esse fator pode ser influenciado pelo fato de os modelos estarem inseridos em uma solução ampliada, visto que a maior parte das execuções da predição Bidirecional são ignoradas (conforme resultados do capítulo 8). Outra análise pode ser feita em relação à divisão da solução em 12 modelos como sendo positiva, pois, apesar de apresentarem FI muito semelhantes, alguns deles se destacam com um comportamento diferente. Nesse sentido, pode-se citar o caso do modelo 128×128, que se difere significativamente dos demais modelos e vai impactar na codificação do maior bloco com o maior tempo de codificação na etapa *Affine*.

Com base nos resultados de FI, optou-se por uma estratégia de selecionar as mesmas *features* para todos os modelos, por motivos de simplificação na implementação dos mesmos no VTM. Dessa maneira, buscou-se identificar e manter as *features* mais relevantes em termos de FI comuns em todos os modelos ou ao menos, na maior parte deles. Nesse sentido, foram mantidas as *features* presentes na Tabela 33 para o treinamento final. Conforme já descrito anteriormente, a maior parte dessas *features* apresentou um valor de FI considerável (maior que 0,01) em todos ou na maioria dos modelos.

Especificamente, no caso do índice BCW (indiceBcw), do IMV e da posição do bloco (cu_pos), os mesmos foram mantidos com todos os seus valores da binarização, visto que não implicam em maior custo ao serem processados. Com exceção do Desvio Padrão, nota-se que as demais *features* geradas através de cálculos sobre as amostras do bloco, tais como os gradientes, foram excluídas. Essa decisão foi baseada em função do considerável tempo de processamento que essas *featu-*

res demonstraram e seus valores de FI praticamente nulos na maioria dos modelos. Para os blocos vizinhos (pai, acima e à esquerda) ocorre a situação semelhante, sendo que os mesmos foram excluídos em função do impacto no tempo de processamento. Apesar de algumas *features* apresentarem valores de FI maiores que 0,1, tais como os custos da etapa *Affine* no bloco acima (custo_affine_acima) e à esquerda (custo_affine_vizEsq), os valores não chegam a 1,0 e ocorrem somente em alguns modelos de maneira isolada. Nesse sentido, analisando o custo de processamento e o seu impacto nas decisões dos modelos, optou-se por retirar as *features* dos blocos vizinhos do treinamento final.

Tabela 33 – Relação das *Features* utilizadas no treinamento final

Feature	**Qtd**	**Descrição**
cu_pos	2	Posição X e Y do bloco dentro do quadro
indiceBcw	5	Índice relacionado ao peso usado na BCW
imv	4	Precisão do Vetor de Movimento (AMVR)
atual_QP	1	Parâmetro de quantização utilizado no momento
mv_uni	4	Vetor de Movimento (X, Y) gerado pela predição Unidirecional para a Lista 0 e Lista 1
mv_pred_uni	4	Vetor de Movimento (X, Y) gerado pela predição AMVP Unidirecional para a Lista 0 e Lista 1
cost_mv_uni	2	Custo do Vetor de Movimento gerado pela predição Unidirecional para a Lista 0 e Lista 1
bits_mv_uni	2	Quantidade de bits do Vetor de Movimento gerado pela predição Unidirecional para a Lista 0 e Lista 1
checkAffine	1	Indica se *Affine* deve ou não ser testada, de acordo com o valor atual do imv
affineModeSelected	1	Indica se o modo *Affine* já foi selecionado
desvioPadrao	1	Desvio padrão das amostras do bloco atual
bloco_atual_affine	1	Indica se o bloco atual possui o melhor modo de predição como sendo *Affine*

9.3.3 Treino Final dos Modelos

A partir dos valores dos hiperparâmetros (Tabela 32) e das *features* selecionadas (Tabela 33), os 12 modelos passaram pela etapa de treinamento com os *datasets* parciais. Esses *datasets* representam 75% do *dataset* original e foram definidos desde a etapa de busca de hiperparâmetros. Para o teste final e geração das métricas, foram considerados os *datasets* com 25% restantes dos dados originais. A Tabela 34 apresenta as métricas obtidas para cada um dos 12 modelos nesse processo de treino e teste. Ao final, o *dataset* completo foi utilizado para o treinamento final de cada modelo a ser implementado no VTM.

A fim de analisar os resultados apresentados é possível agrupar os modelos de acordo com seus desempenhos. Os modelos 16×16, 16×32, 32×16, 32×32, 16×64,

Tabela 34 – Resultados do treinamento e teste finais

Métrica/ Modelo		Acurácia	Precisão	Recall	F1-score	Falsos Negativos	Falsos Positivos
16×16	0	95%	98%	92%	95%	0,9%	4,1%
	1		92%	98%	95%		
16×32	0	96%	99%	93%	96%	0,5%	3,6%
	1		93%	99%	96%		
32×16	0	96%	99%	93%	96%	0,5%	3,6%
	1		93%	99%	96%		
32×32	0	96%	99%	94%	96%	0,6%	3,2%
	1		94%	99%	96%		
16×64	0	96%	98%	93%	96%	0,9%	3,5%
	1		93%	98%	96%		
64×16	0	96%	99%	93%	96%	0,7%	3,3%
	1		94%	99%	96%		
32×64	0	97%	99%	95%	97%	0,4%	2,8%
	1		95%	99%	97%		
64×32	0	97%	99%	94%	97%	0,4%	2,8%
	1		95%	99%	97%		
64×64	0	98%	99%	96%	97%	0,6%	1,9%
	1		96%	99%	98%		
64×128	0	98%	99%	97%	98%	0,3%	1,7%
	1		97%	99%	98%		
128×64	0	98%	99%	96%	98%	0,6%	1,8%
	1		96%	99%	98%		
128×128	0	99%	100%	98%	99%	0,2%	1,2%
	1		98%	100%	99%		
Média	**0**	96,2%	98,2%	94,0%	96,0%	0,8%	3,0%
	1		94,3%	98,3%	96,2%		

64×16 apresentam acurácia entre 95% e 96%. Especificamente, ao analisar a classe 0 a precisão fica entre 98% e 99% e o *recall* de 92% a 94%, demonstrando um bom desempenho nas previsões de não executar a *Affine*. Já para a classe 1 os valores se invertem, com a precisão de 92% a 94% e o *recall* entre 98% e 99%, refletindo a alta capacidade de prever corretamente quando a *Affine* deve ser executada. Ao analisar as taxas de Falsos Negativos nota-se que as mesmas apresentam valores significativamente baixos, de 0,5% a 0,9%, indicando que poucos erros na decisão de não executar a *Affine* e, consequentemente, baixo impacto em perda de eficiência de codificação. Já as taxas de Falsos Positivos são levemente maiores, variando entre 3,2% e 4,1%, o que significa um possível impacto negativo no tempo de codificação.

Já os modelos 32×64 e 64×32 atingem 97% de acurácia, com melhorias também nas outras métricas. De maneira específica, a classe 0 apresenta precisão de 99% e *recall* de 95% e 94%, significando que os modelos têm alta capacidade de prever quando a *Affine* não deve ser executada. A classe 1, por sua vez, também apresenta

valores altos de precisão e *recall*, com 95% e 99%, respectivamente, indicando que os modelos acertam praticamente todas as situações em que a *Affine* deve ser executada. Quanto à taxa de Falsos Negativos, nota-se que é mais baixa do que a dos modelos anteriores, bem como a taxa de Falsos Positivos, que fica em 2,8%. Isso indica que possivelmente esses modelos apresentarão poucos erros, principalmente ao ignorar a etapa *Affine*, afetando minimamente o tempo de codificação e a eficiência de codificação.

Os modelos 64×64, 64×128 e 128×64 apresentam 98% de acurácia, indicando alta taxa de acertos nas previsões de maneira geral. A precisão para a classe 0 chega a 99% e o *recall* varia entre 96% e 97%, o que demonstra alta capacidade de acertos nas decisões de não executar a *Affine*. A classe 1 apresenta os resultados invertidos, com 96% e 97% de precisão e 99% de *recall*, indicando um bom desempenho ao prever quando a *Affine* deve ser executada. Sobre as taxas de Falsos Negativos e Falsos Positivos, é possível notar que ambas diminuíram para esses modelos, sugerindo impacto ainda menor dos mesmos tanto em termos de tempo de codificação quanto perdas na eficiência de codificação.

O modelo 128×128 obteve o maior desempenho com 99% de acurácia. Além disso, para as classes 0 e 1, respectivamente, a precisão ficou em 100% e 98%, e o *recall* com os valores inversos. Já as taxas de erros, ou seja, Falsos Negativos e Falsos Positivos, diminuíram, chegando a 0,2% e 1,2%, nessa ordem.

De modo geral, os modelos apresentaram desempenhos positivos, com altas taxas de acurácia, precisão e *recall*. Diante disso, é necessário ponderar sobre a possibilidade de sobreajuste dos modelos aos dados apresentados. Entretanto, a separação dos *datasets* entre treino e teste (com este último sendo utilizado somente para a geração das métricas finais), o número considerável de exemplos utilizados nos *datasets* e o balanceamento realizado demonstram que a ocorrência de sobreajuste é pouco provável. Outra possibilidade é que a decisão entre executar ou não a predição *Affine*, após a otimização da Bidirecional, é um processo simples. Sendo assim, os valores de *Feature Importance* dos 12 modelos apresentados anteriormente corroboram nesse sentido, visto que é notável que poucas *features* impactam na decisão de cada modelo.

9.4 Implementação no VTM

Em termos de implementação dos modelos no VTM, a solução de otimização ampliada, descrita neste capítulo, segue um procedimento semelhante ao já descrito na seção 8.5 do capítulo 8. Após o treinamento final, cada modelo passou por um processo de tradução para a linguagem C++, através da função `export_to_c`, da biblioteca `m2cgen` (Zeigerman, 2022). A versão do VTM com custo reduzido foi utilizada

para a implementação de cada modelo por meio de classes na pasta `CommonLib`. Após a execução da etapa Bidirecional já otimizada, as *features* necessárias são extraídas e utilizadas como entrada para um dos 12 modelos de otimização da *Affine*, de acordo com o tamanho do bloco atual. Seguindo a mesma lógica da otimização anterior, foi considerado o ponto de corte de 0,6 nas decisões dos modelos, conforme ilustrado anteriormente na Figura 52. Sendo assim, a etapa *Affine* será ignorada quando a pontuação retornada pelo modelo na classe 0 for maior que 0,4. Já quando a pontuação do modelo para a classe 1 é igual ou maior que 0,6 a predição *Affine* é executada normalmente.

É importante destacar que para avaliar os resultados da otimização ampliada, foram utilizadas três versões do VTM: original, VTM com custo reduzido e VTM sem as predições Bidirecional e *Affine*. Novamente, a versão original do VTM é utilizada como base para as análises. Já a versão do VTM com custo reduzido serve para avaliar a performance dos modelos treinados em uma situação real. E a versão do VTM sem as etapas Bidirecional e *Affine* serve como parâmetro para indicar os limites possíveis de serem alcançados em termos de redução do tempo de execução e perda na eficiência de codificação. Novamente, o ideal seria que a solução obtivesse uma redução de tempo de codificação próxima da obtida pela exclusão das duas ferramentas, porém com perda na eficiência de codificação próxima a zero.

A versão do VTM sem as predições Bidirecional e *Affine* foi modificada para desabilitar as duas ferramentas, mantendo as demais funcionalidades ativas, de acordo com o padrão. Adicionalmente, as três versões do VTM também foram alteradas para extrair as métricas relacionadas ao tempo de execução de algumas etapas (intraquadro e interquadros - com as respectivas predições Unidirecional, Bidirecional e Affine - e ferramenta Merge). Além disso, o VTM com custo reduzido também foi alterado para obter a contagem relacionada às decisões dos modelos.

9.5 Resultados no VTM

A seguir, serão descritos e analisados os resultados alcançados a partir dos 12 Modelos de *Decision Tree* para a redução do custo computacional da etapa *Affine* do VVC, implementados de maneira complementar à solução descrita no capítulo 8. Os experimentos utilizaram as mesmas 15 sequências de vídeos das CTCs, conforme a Tabela 23 já descrita anteriormente. Os vídeos foram codificados em seus primeiros 32 quadros, usando a configuração *Random Access* do VTM 22.0. Cada vídeo foi codificado uma vez para cada QP (22, 27, 32 e 37) nas três versões do VTM (original, com custo reduzido e sem a Bidirecional e *Affine*).

As métricas utilizadas para a análise do desempenho dos modelos foram o tempo de codificação e o BDBR, sendo que o VTM já disponibiliza o tempo total, a taxa de

bits e o YUV-PSNR. Os valores referentes ao tempo de codificação de cada etapa e decisões dos modelos foram extraídos através das modificações já citadas anteriormente. Para realizar os cálculos foram consideradas as mesmas equações já descritas na seção 8.6. Serão apresentados os valores médios de redução de tempo total (MRT), da etapa interquadros (MRT_{Inter}), da etapa intraquadro (MRT_{Intra}), da etapa Merge (MRT_{Merge}), das predições Unidirecional (MRT_{uni}), Bidirecional (MRT_{bi}) e *Affine* (MRT_{aff}). Também será analisado o overhead da solução, através das médias de tempo dos modelos (MT_{DT} e MT_{RF}) e da extração das *features* (MT_{feat}). Além disso, o $BDBR$ será apresentado juntamente com a relação entre economia de tempo e $BDBR$ ($MRT/BDBR$).

A seguir, serão analisadas três tabelas contendo os resultados comparativos da solução de redução de custo para as predições Bidirecional e *Affine* e o VTM original. A Tabela 35 contém os resultados referentes à redução de tempo total e para as predições Bidirecional e *Affine*, além do *overhead* da solução. Em seguida, a Tabela 36 amplia a análise, incluindo também os resultados de redução de tempo para a Unidirecional, além das etapas interquadros, intraquadro e Merge. Já a Tabela 37 apresenta a comparação entre a redução no tempo total e o impacto na eficiência de codificação. Além dessas análises, também é incluída a Tabela 38 uma comparação dos resultados da solução proposta com a versão do VTM sem as predições Bidirecional e *Affine*.

Ao analisar a Tabela 35, nota-se que a solução ampliada, usando RF e DT, manteve a alta redução no tempo de codificação da predição Bidirecional. Entretanto, percebe-se também que a predição *Affine* continua sendo impactada com aumento no tempo de codificação na maioria dos vídeos. Apenas nos vídeos `SlideEditing` e `SlideShow` a solução ampliada obteve redução no tempo da *Affine*, de 5,1 e 2,0%, respectivamente. Mesmo assim, ao comparar esses resultados com a solução do capítulo 8, a predição *Affine* teve um desempenho cerca de 9% melhor na solução ampliada. Sendo assim, não houve redução no tempo de codificação, mas redução no impacto gerado pela solução de otimização da Bidirecional (Capítulo 8).

Já a redução no tempo total de codificação continuou moderada, alcançando no máximo de 7,0 e 6,3%, também nos vídeos citados anteriormente. Em comparação com a solução anterior (Capítulo 8), a média geral de redução no tempo total atingiu praticamente o dobro do valor, ou seja, de 2% foi para 3,9%. Especificamente nas classes de vídeo A1 e A2, de alta resolução, essa diferença chegou a 2,5% e 2,2%, respectivamente. Outra análise importante se refere ao *overhead* da solução, onde fica evidente que o fato de adicionar DT não o impactou significativamente, visto que esses modelos tomam apenas 0,1% do tempo total e acrescentam no máximo essa proporção de tempo na extração das *features* em alguns vídeos. Esse fator demonstra que a escolha pelo modelo de DT foi importante para não acrescentar mais complexidade à solução e não impactar negativamente nas reduções obtidas.

Tabela 35 – Impacto dos modelos na redução de tempo de execução

Vídeo/ Classe	MRT_{bi} **(%)**	MRT_{aff} **(%)**	MRT_{Total} **(%)**	MT_{RF} **(%)**	MT_{DT} **(%)**	MT_{feat} **(%)**
ArenaOfValor	92,6	-12,8	3,9	0,4	0,1	1,8
BasketballDrillText	92,2	-24,3	1,8	0,4	0,1	1,8
SlideEditing	92,2	5,1	7,0	0,2	0,1	1,3
SlideShow	93,3	2,0	6,3	0,3	0,1	1,3
Média Classe F	**92,6**	**-7,5**	**4,7**	**0,3**	**0,1**	**1,5**
BasketballDrive	95,6	-21,2	2,9	0,3	0,1	1,4
BQTerrace	92,7	-37,5	3,3	0,3	0,1	1,5
Cactus	92,5	-19,1	3,9	0,4	0,1	1,7
MarketPlace	86,0	-35,1	2,1	0,2	0,0	1,3
RitualDance	91,1	-20,5	3,4	0,3	0,0	1,4
Média Classe B	**91,6**	**-26,7**	**3,1**	**0,3**	**0,1**	**1,5**
Campfire	91,9	-15,4	5,2	0,4	0,1	1,8
FoodMarket4	95,7	-20,8	3,4	0,2	0,1	1,0
Tango2	93,3	-21,4	3,1	0,3	0,1	1,3
Média Classe A1	**93,6**	**-19,2**	**3,9**	**0,3**	**0,1**	**1,4**
CatRobot	92,2	-17,1	3,7	0,3	0,1	1,4
DaylightRoad2	91,4	-12,6	4,9	0,3	0,1	1,4
ParkRunning3	90,7	-27,7	3,5	0,4	0,1	1,6
Média Classe A2	**91,4**	**-19,1**	**4,1**	**0,4**	**0,1**	**1,5**
Média Geral	**92,2**	**-18,6**	**3,9**	**0,3**	**0,1**	**1,5**

A seguir, a Tabela 36 aborda os resultados de redução no tempo de codificação, incluindo os valores para a predição Unidirecional e das etapas interquadros, intraquadro e Merge. Na predição Unidirecional, especificamente, nota-se uma redução singela no tempo de codificação, de 0,5% até 3,7% em média. Quando esses valores são comparados aos resultantes na otimização anterior (capítulo 8), é possível perceber um avanço de cerca de 1,5% em média na redução. Uma situação semelhante ocorre quando a redução no tempo da etapa interquadros é analisada: na solução ampliada alcança de 10,7% até 16,5% em média, sendo esta uma melhora de 2,3% até 4,5%, quando comparada à solução de otimização para a Bidirecional (capítulo 8). Isso demonstra, mais uma vez, o alinhamento dos resultados alcançados com o foco principal da tese. Mesmo que parcialmente e com espaços para melhorias, os valores de redução de tempo obtidos para a etapa interquadros, através das duas soluções de otimização, validam a hipótese deste trabalho.

A próxima análise, quanto à eficiência de codificação, se encontra na Tabela 37, que contém os resultados de redução de tempo total, $BDBR$ e a relação entre os dois. Sendo assim, pode-se considerar que valores maiores para a coluna $MRT_{Total}/BDBR$ indicam que a solução obteve uma redução significativa no tempo total com baixo impacto na eficiência de codificação. Ao analisar as classes de vídeo, percebe-se que a classe F obteve o melhor desempenho, com a maior redução no tempo de codificação

Tabela 36 – Impacto dos modelos no tempo total e das principais ferramentas do codificador

Vídeo/ Classe	MRT						
	Bi **(%)**	Uni **(%)**	Aff **(%)**	$Inter$ **(%)**	$Intra$ **(%)**	$Merge$ **(%)**	$Total$ **(%)**
ArenaOfValor	92,6	1,1	-12,8	12,0	-1,6	0,1	3,9
BasketballDrillText	92,2	0,9	-24,3	8,2	-3,1	0,3	1,8
SlideEditing	92,2	-0,1	5,1	26,0	0,2	1,4	7,0
SlideShow	93,3	0,0	2,0	20,0	-1,6	0,9	6,3
Média Classe F	**92,6**	**0,5**	**-7,5**	**16,5**	**-1,5**	**0,7**	**4,7**
BasketballDrive	95,6	1,6	-21,2	10,6	-2,9	-1,2	2,9
BQTerrace	92,7	7,5	-37,5	10,8	-1,3	0,7	3,3
Cactus	92,5	4,1	-19,1	11,0	-0,8	0,7	3,9
MarketPlace	86,0	1,2	-35,1	8,5	-0,3	0,5	2,1
RitualDance	91,1	1,2	-20,5	12,4	-0,1	1,2	3,4
Média Classe B	**91,6**	**3,1**	**-26,7**	**10,7**	**-1,1**	**0,4**	**3,1**
Campfire	91,9	0,7	-15,4	17,5	-1,1	0,6	5,2
FoodMarket4	95,7	1,8	-20,8	11,3	-1,1	-2,5	3,4
Tango2	93,3	0,9	-21,4	11,1	-3,4	-1,9	3,1
Média Classe A1	**93,6**	**1,1**	**-19,2**	**13,3**	**-1,9**	**-1,3**	**3,9**
CatRobot	92,2	3,5	-17,1	10,7	-2,2	-0,9	3,7
DaylightRoad2	91,4	2,6	-12,6	12,2	-1,7	0,0	4,9
ParkRunning3	90,7	5,1	-27,7	10,5	-2,1	0,7	3,5
Média Classe A2	**91,4**	**3,7**	**-19,1**	**11,1**	**-2,0**	**-0,1**	**4,1**
Média Geral	**92,2**	**2,1**	**-18,6**	**12,8**	**-1,5**	**0,0**	**3,9**

(média de 4,7%) ao custo do menor valor de BDBR (média de 0,444%). Pode-se destacar também que o vídeo `SlideEditing` obteve o melhor desempenho dentre todos os vídeos, com redução total de 7% no tempo e melhora na eficiência de codificação, indicada pelo BDBR -0,055%. Já as classes B e A2 apresentaram desempenhos semelhantes, com redução moderada no tempo de codificação, mantendo um valor de BDBR abaixo de 1% em média. A classe A1, por sua vez, apresentou um desempenho levemente melhor, analisando a relação $MRT_{Total}/BDBR$, principalmente pelo resultado do vídeo `Campfire`, que atingiu o segundo melhor valor de todos.

A Tabela 38, por sua vez, compara os resultados do VTM com custo reduzido ("Com RF e DT" na Tabela) e da versão do VTM sem as ferramentas Bidirecional e *Affine* ("Sem Bi e Affine" na Tabela). O objetivo de abordar esses dados é estabelecer um limite superior máximo quanto à redução do tempo de execução e o impacto na eficiência de codificação. Conforme o esperado, ao desabilitar as ferramentas Bidirecional e *Affine*, foi obtida uma redução significativa no tempo de execução total, porém, ao custo também significativo de perda na eficiência de codificação.

Quanto aos resultados comparativos de redução de tempo de codificação, percebe-se que a solução "Com RF e DT" não atingiu valores próximos aos obtidos pela versão sem as ferramentas Bidirecional e *Affine*. Um fator importante que contribuiu para isso

Tabela 37 – Impacto dos modelos na eficiência de codificação

Vídeo/ Classe	MRT_{Total} **(%)**	$BDBR$ **(%)**	$MRT_{Total}/BDBR$
ArenaOfValor	3,9	0,268	14,5
BasketballDrillText	1,8	0,893	2,0
SlideEditing	7,0	-0,055	-125,7
SlideShow	6,3	0,670	9,4
Média Classe F	**4,7**	**0,444**	**-24,9**
BasketballDrive	2,9	0,883	3,3
BQTerrace	3,3	1,016	3,3
Cactus	3,9	0,546	7,1
MarketPlace	2,1	0,332	6,3
RitualDance	3,4	0,776	4,4
Média Classe B	**3,1**	**0,711**	**4,9**
Campfire	5,2	0,347	14,9
FoodMarket4	3,4	1,205	2,8
Tango2	3,1	1,362	2,3
Média Classe A1	**3,9**	**0,971**	**6,7**
CatRobot	3,7	0,682	5,5
DaylightRoad2	4,9	1,080	4,6
ParkRunning3	3,5	0,918	3,9
Média Classe A2	**4,1**	**0,893**	**4,6**
Média Geral	**3,9**	**0,728**	**-2,8**

é que os modelos DT focados em otimizar a *Affine* conseguiram apenas diminuir o impacto do aumento de tempo nessa ferramenta, que havia ocorrido na solução de otimização com RF para a Bidirecional. Dessa forma, esse fator, somado ao *overhead* da solução e do aumento no uso de outras ferramentas de codificação, é responsável pelos baixos valores de redução no tempo total do codificador.

Já os resultados comparativos de BDBR presentes na Tabela 38 demonstram que, em todos os vídeos, a solução de otimização obteve perdas significativamente inferiores aos valores do VTM sem as ferramentas. Estes resultados demonstram que a solução de otimização ampliada obteve um bom desempenho, garantindo baixa perda de eficiência de codificação. Nota-se que, ao retirar as ferramentas Bidirecional e *Affine* completamente, o impacto na eficiência é significativo, chegando ao valor máximo de 9,908%, para o vídeo `DaylightRoad2`. Por outro lado, a solução usando RF e DT teve o seu pior desempenho com o valor de 1,362%, para o vídeo `Tango2`.

Nesse sentido, pode-se concluir que a solução ampliada, de otimização para a Bidirecional e *Affine* baseada em modelos de aprendizado de máquina, gera uma redução no tempo total de codificação não tão significativa, porém mantendo as perdas na eficiência de compressão abaixo de 1%. Também é importante ressaltar que a solução ampliada permaneceu dentro dos limites apresentados na versão sem as duas ferramentas, tanto em termos de tempo de codificação, quanto de BDBR.

Tabela 38 – Comparação do VTM com redução de custo e do VTM sem a predição Bidirecional e *Affine*

Vídeo/ Classe	MRT_{Total}		$BDBR$	
	Com RF e DT (%)	Sem BI e Affine (%)	Com RF e DT (%)	Sem BI e Affine (%)
ArenaOfValor	3,9	24,7	0,268	2,511
BasketballDrillText	1,8	22,0	0,893	2,927
SlideEditing	7,0	16,9	-0,055	1,134
SlideShow	6,3	18,5	0,670	2,755
Média Classe F	**4,7**	**20,5**	**0,444**	**2,332**
BasketballDrive	2,9	24,2	0,883	5,581
BQTerrace	3,3	24,5	1,016	4,229
Cactus	3,9	25,8	0,546	8,168
MarketPlace	2,1	9,1	0,332	0,604
RitualDance	3,4	10,2	0,776	1,211
Média Classe B	**3,1**	**18,8**	**0,711**	**3,959**
Campfire	5,2	17,3	0,347	0,705
FoodMarket4	3,4	27,9	1,205	2,842
Tango2	3,1	27,6	1,362	4,879
Média Classe A1	**3,9**	**24,2**	**0,971**	**2,808**
CatRobot	3,7	28,7	0,682	9,693
DaylightRoad2	4,9	31,6	1,080	9,908
ParkRunning3	3,5	23,7	0,918	5,673
Média Classe A2	**4,1**	**28,0**	**0,893**	**8,424**
Média Geral	**3,9**	**22,2**	**0,728**	**4,188**

Por fim, a Figura 65 apresenta uma análise do percentual de decisões dos modelos DT, por classe de vídeo, para cada tamanho de bloco suportado pela *Affine*. Na legenda, a “Decisão 0” e “Decisão 1” se referem ao percentual de vezes em que a ferramenta *Affine* não foi executada ou foi executada, respectivamente. No geral, os modelos apresentaram grande predominância na decisão de ignorar a *Affine*, conforme era esperado. Entretanto, pode-se notar que apenas para as classes F e A1 esse percentual foi igual ou superior a 75%. Já nas classes B e A2, alguns blocos apresentam valores mais baixos, que chegam a 70%. Apesar desses percentuais ainda representarem que a maior parte das vezes a *Affine* é ignorada, é importante notar que a diferença ocorre nos blocos maiores, justamente os blocos com maior tempo de execução na *Affine*. Esse fator pode auxiliar na explicação quanto ao fato de a solução ampliada não ter reduzido o tempo da Affine, mas apenas diminuído o impacto causado pela otimização da Bidirecional. Dessa forma, uma possibilidade emergente dessa discussão é a de que desenvolver uma solução, utilizando aprendizado de máquina, focada apenas na otimização da etapa *Affine*, poderá atingir resultados mais promissores. Essa solução foi desenvolvida e será apresentada no próximo capítulo.

A partir dos resultados da proposta ampliada, focada nas etapas Bidirecional e *Af-*

Figura 65 – Percentual das decisões para *Affine* por tamanho de bloco e classe

(a) Classe F

(b) Classe B

(c) Classe A1

(d) Classe A2

Fonte: Elaborada pelo autor.

fine, é possível realizar a comparação com outros trabalhos, considerando as pesquisas abordadas no capítulo 6. A Tabela 39 apresenta, resumidamente, a comparação entre a solução ampliada e outros quatro trabalhos relacionados, analisando a versão do VTM, foco da aplicação, técnica utilizada, além da redução no tempo total de codificação e o impacto de BDBR. Em uma primeira análise, fica evidente que a comparação direta não é possível, devido as diferenças entre as versões do VTM utilizadas e o foco de cada aplicação, motivo pelo qual os valores de redução de tempo das etapas não foi incluído na comparação. Mesmo assim, os quatro trabalhos apresentados na Tabela 39 são os que mais se aproximam em termos de ferramenta foco da otimização. A partir disso, é possível perceber que, em termos de redução no tempo total de codificação, a solução ampliada perde para todos os três trabalhos relacionados, ficando mais próxima do trabalho de Xie et al. (2022), que utiliza o VVenc, um software baseado no VTM porém com otimizações já implementadas. Já em relação ao BDBR, a solução ampliada apresenta menor impacto na eficiência de compressão, comparada aos trabalhos de Xie et al. (2022) e Chan; Im 2023. É importante destacar que, os demais trabalhos utilizam versões anteriores do VTM, em que várias otimizações não haviam sido implementadas, o que pode indicar a diferença nos resultados. Mesmo assim, a solução ampliada proposta nesse capítulo demonstra ser competitiva com os demais trabalhos relacionados, apesar de não ser possível um comparação direta, principalmente em relação a versão do VTM.

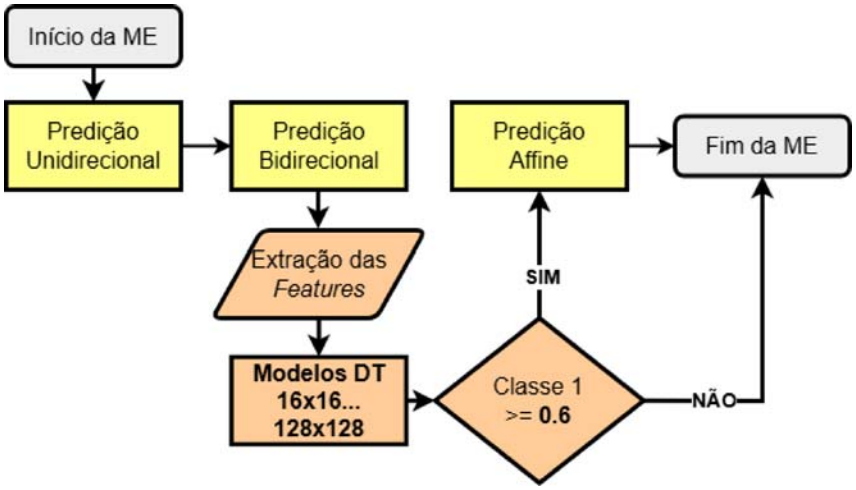
Tabela 39 – Comparação da solução proposta com trabalhos relacionados

Trabalho	**Versão VTM**	**Foco**	**Técnica**	MRT_{Total} **(%)**	$BDBR$ **(%)**
Pan et al. (2019)	1.0	ME	AE	34,3	0,49
Xie et al. (2022)	VVenc 1.0.0	interquadros	RF	7,7	1,48
Chan; Im (2023)	13.0	Quadro de referência	RNN	11,4	2,86
Huang et al. (2023)	10.0	Unidirecional, Bidirecional e Affine	AE e DT	10,3	0,14
Solução Proposta (Capítulo 9)	**22.0**	**Bidirecional e Affine**	**RF e DT**	**3,9**	**0,73**

10 REDUÇÃO DE CUSTO COMPUTACIONAL DA PREDIÇÃO *AFFINE* USANDO APRENDIZADO DE MÁQUINA

Neste capítulo será descrita a solução de otimização para a predição *Affine* do codificador VVC. Essa solução visa reduzir o tempo de codificação com baixo impacto na eficiência de codificação. Conforme ilustrado na Figura 66, essa otimização utiliza 12 modelos de Árvores de Decisão especializados para cada um dos tamanhos de bloco suportados pela predição *Affine* (de 16×16 até 128×128). A extração das *features* ocorre após a execução das predições Unidirecional e Bidirecional, sendo que as mesmas são utilizadas como entrada para um dos modelos, conforme o tamanho do bloco atual, e este retorna 1 ou 0. A decisão 1 indica que a predição *Affine* será executada conforme o padrão VVC, já a decisão 0 indica que toda a predição *Affine* será ignorada e a ME será finalizada para o bloco atual. É importante ressaltar que, conforme o padrão VVC, a *Affine* primeiramente testa o modelo de 4-parâmetros e, apenas se o seu respectivo custo for menor que 1.05 vezes o menor custo obtido nas predições Unidirecional e Bidirecional, o modelo *Affine* de 6-parâmetros é testado. No caso desta solução de otimização, esta lógica não foi alterada quando a decisão do modelo DT é 1.

Figura 66 – Solução de otimização aplicada à predição *Affine*



Fonte: Elaborada pelo autor.

Especificamente para esta solução, os experimentos, desde a extração até os testes, foram realizados em um servidor com processador Intel Xeon E5-2640v3 2.60 GHz com 32 GB de RAM. O treinamento dos modelos DT foi realizado a partir da linguagem Python, com a biblioteca `scikit-learn` (PEDREGOSA et al., 2011). Assim como as duas soluções anteriores, a biblioteca `m2cgen` (Zeigerman, 2022) foi utilizada para exportar os modelos DT para a linguagem C++, a fim de serem implementados no VTM 22.0.

10.1 Extração de Dados

Na fase de extração das *features* foram utilizadas as mesmas nove sequências selecionadas no capítulo 9, das bases (Mercat; Viitanen; Vanne, 2020) e NETVC (Daede; Norkin; Brailovskiy, 2019): *Dark*, *Netflix DrivingPOV* e *Vidyo4* (HD, resolução 1280×720), *Netflix TunnelFlags*, *Jockey* e *Touchdown Pass* (FHD, resolução 1920×1080), *ToddlerFountain*, *SunBath* e *Lips* (4K UHD, resolução 3840×2160). Essas sequências foram codificadas quatro vezes, uma vez para cada QP (22, 27, 32 e 37), nos seus primeiros 16 quadros, usando a versão 22.0 do VTM na configuração *Random Access*.

A Tabela 40 descreve as *features* extraídas após a execução da etapa Bidirecional, que ocorre conforme o codificador VTM. Essas *features* são as mesmas da solução anterior, incluindo os dados básicos sobre o bloco que está sendo testado no momento da extração, dados relacionados à profundidade de particionamento do bloco (depth, qtDepth e mtDepth) (conforme (Gonçalves, 2021)), dados sobre blocos vizinhos, os valores específicos extraídos da etapa Bidirecional, tais como vetores de movimento (mv_bi e mv_pred_bi), custo (cost_mv_bi), número de bits (bits_mv_bi), uso da ferramenta SMVD e direção do movimento (interDir) (de acordo com (Sagrilo; Loose; Viana; Sanchez; Corrêa; Agostini, 2023)), e as *features* checkAffine e affineModeSelected, que estão presentes no código do VTM, mais especificamente, no teste original que decide se a etapa *Affine* será ou não executada.

10.2 Elaboração dos *Datasets*

Assim como definido na solução ampliada (Capítulo 9), nesta otimização voltada para a *Affine* também foram utilizados 12 *datasets* separados, de acordo com os tamanhos de bloco suportados pela predição *Affine*. Dessa forma, os 12 *datasets* foram organizados a partir dos dados extraídos para os blocos de 16×16 até 128×128.

Para a organização dos *datasets*, foi realizado o balanceamento dos dados extraídos, semelhantemente ao processo realizado nas soluções anteriores (capítulos 8 e 9). Esse balanceamento considerou 1.000 exemplos para cada combinação de vídeo,

Tabela 40 – Relação das *Features* extraídas após a etapa Bidirecional

Feature	**Qtd**	**Descrição**
tamBlocoWidth	1	Largura do bloco em píxeis
tamBlocoHeight	1	Altura do bloco em píxeis
numQP	4	Parâmetro de quantização de entrada
widthVideo	1	Largura do vídeo em píxeis
heightVideo	1	Altura do vídeo em píxeis
cu_pos	2	Posição X e Y do bloco dentro do quadro
depth	1	Profundidade do bloco no particionamento QTMTT
qtDepth	1	Profundidade do bloco no particionamento QT
mtDepth	1	Profundidade do bloco no particionamento MTT
indiceBcw	5	Índice relacionado ao peso usado na BCW
imv	4	Precisão do Vetor de Movimento (AMVR)
atual_QP	1	Parâmetro de quantização utilizado no momento
mv_uni	4	Vetor de Movimento (X, Y) gerado pela predição Unidirecional para a Lista 0 e Lista 1
mv_pred_uni	4	Vetor de Movimento (X, Y) gerado pela predição AMVP Unidirecional para a Lista 0 e Lista 1
cost_mv_uni	2	Custo do Vetor de Movimento gerado pela predição Unidirecional para a Lista 0 e Lista 1
bits_mv_uni	2	Quantidade de bits do Vetor de Movimento gerado pela predição Unidirecional para a Lista 0 e Lista 1
mv_bi	4	Vetor de Movimento (X, Y) gerado pela predição Bidirecional para a Lista 0 e Lista 1
mv_pred_bi	4	Vetor de Movimento (X, Y) gerado pela predição AMVP Bidirecional para a Lista 0 e Lista 1
cost_mv_bi	1	Custo do Vetor de Movimento gerado pela predição Bidirecional
bits_mv_bi	1	Quantidade de bits do Vetor de Movimento gerado pela predição Bidirecional
SMVD	1	Indica se o SMVD está sendo testado ou não
interDir	1	Indica a direção da predição
checkAffine	1	Indica se *Affine* deve ou não ser testada, de acordo com o valor atual do imv
affineModeSelected	1	Indica se o modo *Affine* já foi selecionado
bidirecional_pai	1	Indica se o bloco pai foi codificado pela predição Bidirecional
custo_pai	1	Custo gerado pela predição Bidirecional do bloco pai
IMV_pai	5	Valor do IMV do bloco pai
affine_pai	1	Indica se o bloco pai foi codificado pela predição *Affine*
custo_affine_pai	1	Custo gerado pela predição *Affine* do bloco pai
bidirecional_viz_esq	1	Indica se o bloco vizinho a esquerda foi codificado pela predição Bidirecional
custo_viz_esq	1	Custo gerado pela predição do bloco vizinho a esquerda
IMV_viz_esq	5	Valor do IMV do bloco vizinho a esquerda

affineVizEsq	1	Indica se o bloco vizinho à esquerda foi codificado pela predição *Affine*
custo_affine_VizEsq	1	Custo gerado pela predição *Affine* do bloco vizinho à esquerda
bidirecional_viz_acima	1	Indica se o bloco vizinho acima foi codificado pela predição Bidirecional
custo_viz_acima	1	Custo gerado pela predição do bloco vizinho acima
IMV_viz_acima	5	Valor do IMV do bloco vizinho acima
affineVizAci	1	Indica se o bloco vizinho acima foi codificado pela predição *Affine*
custo_affine_acima	1	Custo gerado pela predição *Affine* do bloco vizinho acima
sum	1	Somatório das amostras do bloco atual
media	1	Média das amostras do bloco atual
vari	1	Variância das amostras do bloco atual
desvioPadrao	1	Desvio padrão das amostras do bloco atual
grad	2	Gradientes Horizontal e Vertical do bloco atual (Sobel)
razao_grad	1	Gradiente Horizontal divido pelo Gradiente Vertical
grad_razao_píxeis	1	Soma dos gradientes divido pelo número de píxeis
dist_uni_L0_L1	1	Distância entre os Vetores de Movimento resultantes da predição Unidirecional da Lista 0 e Lista 1
bloco_atual_affine	1	Indica se o bloco atual possui o melhor modo de predição como sendo *Affine*

QP, tamanho de bloco e decisão. Para cada um dos nove vídeos foram selecionados 1.000 exemplos da decisão 0 (não executar a *Affine*) e 1.000 exemplos da decisão 1 (executar a *Affine*), considerando os quatro QPs e os 12 tamanhos de bloco. Sendo assim, cada um dos 12 *datasets* possui 72.000 exemplos, totalizando então 864.000 exemplos.

Entretanto, para alguns vídeos e QPs em específico, ocorreu novamente o problema de não existirem 1.000 exemplos. Esse problema apresentou-se em 10 dos 12 tamanhos de bloco, com prevalência em vídeos da resolução HD e na decisão 1 (executar a *Affine*). Novamente, houve maior frequência dessa falta de registros para os blocos 128×128, 64×128 e 128×64, em que o problema ocorre em todas as resoluções. Já para os blocos 64×64, 16×64, 64×16, 32×32, 16×32 e 32×16, o problema ocorre apenas no QP 37 da sequência "Vidyo4"da resolução HD. Apesar disso, o balanceamento foi realizado a partir da seleção de outros exemplos, seguindo os critérios já descritos anteriormente: $i)$ são selecionados exemplos extras de outros vídeos da mesma resolução e QP; e, se necessário, $ii)$ os exemplos extras serão obtidos de vídeos de outra resolução, porém do mesmo QP.

10.3 Treinamento dos Modelos

Após o balanceamento dos registros nos *datasets* foi iniciado o processo de treinamento dos 12 modelos. As etapas seguintes englobam a busca de hiperparâmetros, utilizando os algoritmos *Random Search* e *Grid Search*, a seleção das *features* e o treinamento final. É importante salientar que, nessa solução em específico, os *datasets* utilizados no processo de busca de hiperparâmetros já consideraram as *features* selecionadas. Essa ação foi necessária, visto que ao realizar-se os primeiros testes no VTM, os modelos demonstraram um sobreajuste aos dados de treinamento. Sendo assim, houve dois processos de busca de hiperparâmetros, porém somente o último e definitivo é descrito a seguir.

10.3.1 Busca de Hiperparâmetros

A partir da etapa de busca de hiperparâmetros, cada *dataset* foi dividido em conjuntos de treino e teste. Seguindo as proporções já utilizadas anteriormente, de 75% dos registros para treino e 25% para teste, serão 54.000 registros para treinamento e 18.000 para teste, em cada *dataset*.

Assim como nas soluções anteriores, na primeira etapa da busca dos melhores hiperparâmetros foi utilizado o método *Random Search*, que treina e testa 500 combinações aleatórias de valores. A Tabela 41 contém as faixas de valores estipuladas para cada hiperparâmetro. Esses valores foram definidos a partir de trabalhos anteriores, como Duarte (2021) e Andrades; Grellert; Fonseca (2019).

Tabela 41 – Faixas de valores utilizadas no *Random Search*

Hiperparâmetro	Faixa de valores
criterion	[gini, entropy]
min_samples_split	[2] OU [25..500, 25]
min_samples_leaf	[1] OU [10..100, 10]
max_features	[sqrt] OU [1..total]
max_depth	[2..10] OU [20..100, 10]
max_leaf_nodes	[2..10] OU [20..700, 10]

A Tabela 42 apresenta o resultado do processo de busca a partir do método *Random Search*. Para cada *dataset* são mostrados os valores cuja combinação gerou o maior F1-score, sendo que em destaque estão os hiperparâmetros com a maior correlação com o mesmo. É notável que o hiperparâmetro **max_features** aparece para todos os *datasets*. Apenas no caso dos datasets 32×16 e 128×128 o hiperparâmetro **max_leaf_nodes** tem maior correlação com o F1, nos demais casos, é **max_depth** que se destaca.

Tabela 42 – Melhor resultado do *Random Search* por modelo

Hiper-parâmetro / Modelo	***criterion***	***min_ samples _split***	***min_ samples _leaf***	***max_ features***	***max_ depth***	***max_ leaf_ nodes***	**F1-score**
16×16	entropy	300	60	45	70	300	**0,88**
16×32	gini	25	70	40	50	640	**0,87**
32×16	entropy	325	30	42	10	300	**0,88**
32×32	gini	50	40	46	70	340	**0,88**
16×64	gini	25	30	39	70	350	**0,88**
64×16	gini	75	10	44	50	60	**0,88**
32×64	gini	100	20	45	90	80	**0,88**
64×32	gini	100	10	42	100	50	**0,88**
64×64	entropy	100	50	40	100	610	**0,86**
64×128	gini	2	1	42	30	330	**0,87**
128×64	gini	75	40	51	100	280	**0,87**
128×128	entropy	2	1	35	70	200	**0,87**

A partir dos resultados do *Random Search* é realizada a segunda parte da busca de hiperparâmetros, com o método *Grid Search*. A definição das faixas de valores seguiu a mesma lógica já descrita anteriormente, ou seja, os hiperparâmetros com maior correlação no *Random Search* são definidos com mais possibilidades a serem testadas no *Grid Search*. Conforme apresentado na Tabela 43, os demais hiperparâmetros foram testados com o seu respectivo valor padrão para o modelo `DecisionTree` da biblioteca `sklearn` e, quando fosse o caso, o valor retornado no *Random Search*. Essas especificações foram realizadas com base nos trabalhos de Duarte (2021) e Andrades; Grellert; Fonseca (2019).

Conforme mostrado na Tabela 44, o método *Grid Search* resultou na combinação de valores para os hiperparâmetros com maior F1-score. Esses valores foram utilizados no treinamento final de cada modelo.

Tabela 43 – Faixas de valores utilizadas no *Grid Search*

Hiper-parâmetro / Modelo	*criterion*	*min_ samples _split*	*min_ samples _leaf*	*max_ features*	*max_ depth*	*max_ leaf_ nodes*
16×16	[entropy, gini]	[2, 300]	[1, 60]	[sqrt] OU [1..45, 2]	[1..9] OU [10..70, 5]	[none, 300]
16×32	[gini]	[2, 25]	[1, 70]	[sqrt] OU [1..40, 2]	[1..9] OU [10..50, 5]	[none, 640]
32×16	[entropy, gini]	[2, 325]	[1, 30]	[sqrt] OU [1..42, 2]	[10]	[10..300, 10]
32×32	[gini]	[2, 50]	[1, 40]	[sqrt] OU [1..46, 2]	[1..9] OU [10..70, 5]	[none, 340]
16×64	[gini]	[2, 25]	[1, 30]	[sqrt] OU [1..39, 2]	[1..9] OU [10..70, 5]	[none, 350]
64×16	[gini]	[2, 75]	[1, 10]	[sqrt] OU [1..44, 2]	[1..9] OU [10..50, 5]	[none, 60]
32×64	[gini]	[2, 100]	[1, 20]	[sqrt] OU [1..45, 2]	[1..9] OU [10..90, 5]	[none, 80]
64×32	[gini]	[2, 100]	[1, 10]	[sqrt] OU [1..42, 2]	[1..9] OU [10..100, 5]	[none, 50]
64×64	[entropy, gini]	[2, 100]	[1, 50]	[sqrt] OU [1..40, 2]	[1..9] OU [10..100, 5]	[none, 610]
64×128	[gini]	[2]	[1]	[sqrt] OU [1..42, 2]	[1..9] OU [10..30, 5]	[none, 330]
128×64	[gini]	[2, 75]	[1, 40]	[sqrt] OU [1..51, 2]	[1..9] OU [10..100, 5]	[none, 280]
128×128	[entropy, gini]	[2]	[1]	[sqrt] OU [1..35, 2]	[70]	[10..200, 10]

Tabela 44 – Hiperparâmetros utilizados para o treinamento final

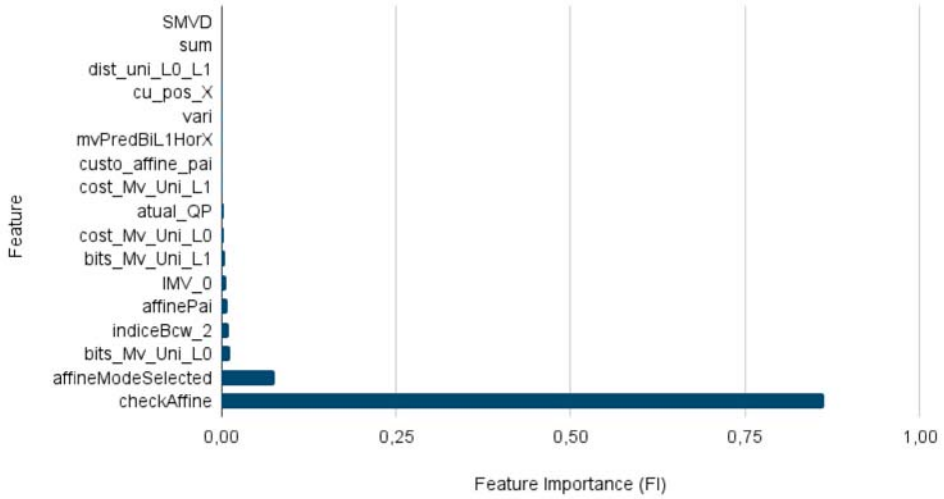
Hiper-parâmetro / Modelo	*criterion*	*min_ samples _split*	*min_ samples _leaf*	*max_ features*	*max_ depth*	*max_ leaf_ nodes*	F1-score
16×16	entropy	2	60	31	10	300	0,89
16×32	gini	2	70	39	30	None	0,88
32×16	gini	2	30	41	10	40	0,89
32×32	gini	50	40	43	50	None	0,89
16×64	gini	25	30	27	60	350	0,89
64×16	gini	2	1	37	50	60	0,89
32×64	gini	2	1	43	10	80	0,89
64×32	gini	2	10	39	25	50	0,89
64×64	entropy	100	1	27	10	None	0,87
64×128	gini	2	1	41	10	None	0,88
128×64	gini	2	1	45	8	280	0,88
128×128	entropy	2	1	35	70	90	0,88

10.3.2 Seleção das *Features*

Para a etapa de seleção das *features* foram gerados os gráficos com os respectivos valores de *feature importance* (FI) em cada modelo. São incluídas nos gráficos as 17 *features* com maior relevância, seguindo o mesmo padrão utilizado na solução anterior. A seguir, serão apresentados os valores de FI para os 12 modelos DT para a *Affine*.

Na Figura 67 são ilustrados os valores de FI para o modelo 16×16. É possível notar que a *feature* `checkaffine` desponta com um alto valor de FI (0,86). Na sequência aparece a `affineModeSelected`, além de outras *features* relacionadas com o resultado da Unidirecional e também do bloco pai.

Figura 67 – *Feature Importance* para o Modelo 16×16



Fonte: Elaborada pelo autor.

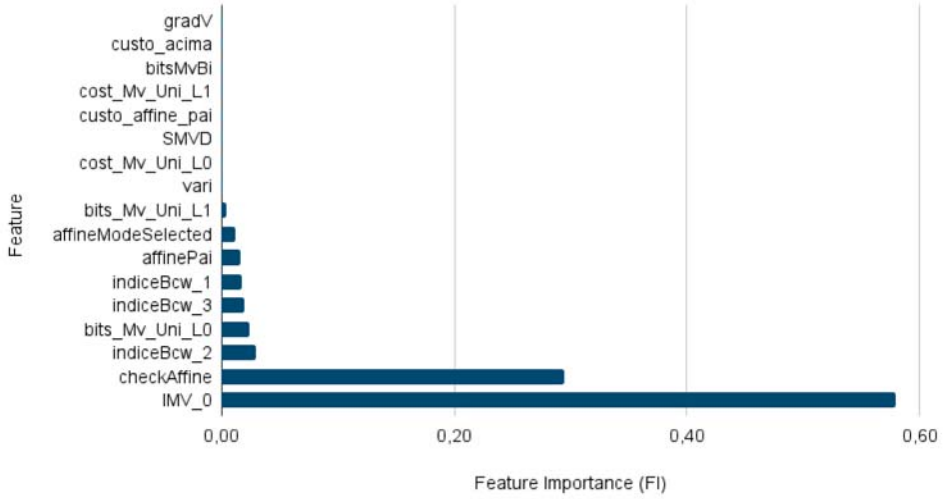
Já na Figura 68, os valores de FI para o modelo 16×32 apresentam um comportamento diferente. A *feature* `IMV_0` é a mais relevante com 0,58 de FI, seguida pela `checkaffine`, com 0,29. As demais *features* apresentam um equilíbrio entre seus valores e estão relacionadas ao índice da ferramenta BCW, bem como resultados da predição Unidirecional e do bloco pai.

A Figura 69, referente aos valores de FI para o modelo 32×16, volta a ter a *feature* `checkaffine` como destaque, com o valor de 0,85. As demais *features* apresentam valores bem menores e, fora a `affineModeSelected`, estão relacionadas novamente com o índice BCW, bloco pai e vetores de movimento da predição Unidirecional.

Para o modelo 32×32, como demonstra a Figura 70, os valores são muito semelhantes. Novamente a *feature* `checkaffine` se destaca, seguida pela `affineModeSelected`. Além disso, as demais *features* são praticamente as mesmas dos casos anteriores, alterando a ordem em que aparecem.

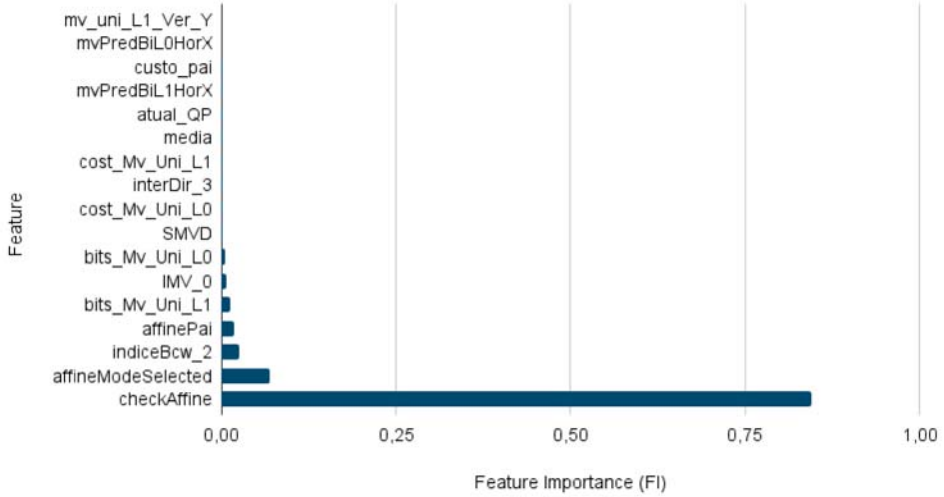
Seguindo um comportamento semelhante, a Figura 71 do modelo 16×64 demons-

Figura 68 – *Feature Importance* para o Modelo 16×32



Fonte: Elaborada pelo autor.

Figura 69 – *Feature Importance* para o Modelo 32×16



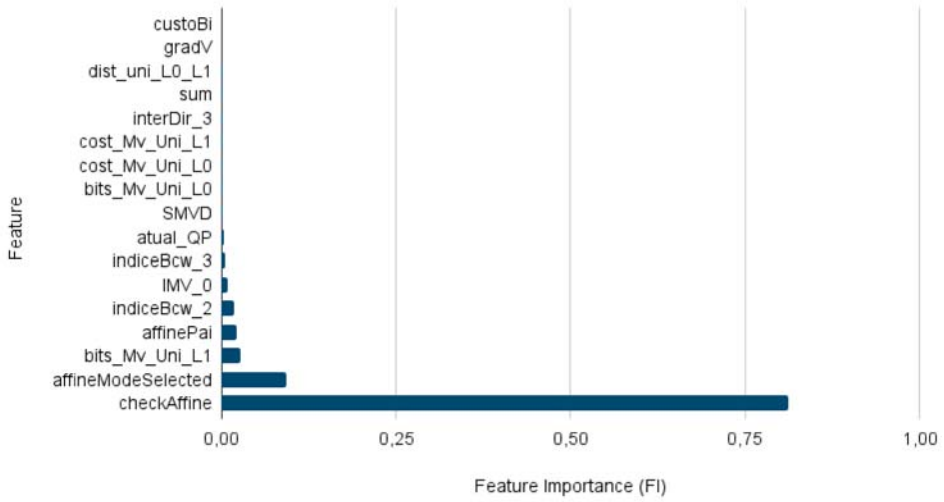
Fonte: Elaborada pelo autor.

tra que novamente a *feature* `checkaffine` possui maior relevância, com valor de FI 0,72. Nesse caso, as demais *features* na sequência são semelhantes aos demais casos, com exceção da `IMV_1`, que aparece pela primeira vez em destaque. Além disso, apesar de apresentarem valores muito baixos, algumas *features* relacionadas às amostras do bloco também aparecem, tais como média, variância e desvio padrão.

A Figura 72, referente ao modelo 64×16, apresenta um comportamento semelhante ao anterior, porém com *features* diferentes. Após as *features* que geralmente aparecem, estão a `SMVD` e `atual_QP`, além da distância entre os vetores de movimento da Unidirecional e também o custo dos mesmos, porém com valores de FI baixos.

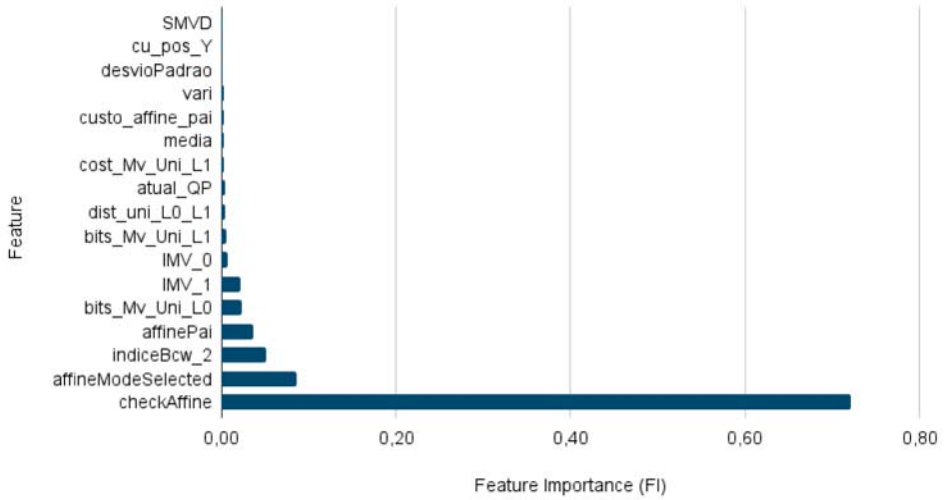
Já o modelo 32×64, representado pela Figura 73, apesar de manter o destaque para a *feature* `checkaffine`, também apresenta um valor relativamente maior

Figura 70 – *Feature Importance* para o Modelo 32×32



Fonte: Elaborada pelo autor.

Figura 71 – *Feature Importance* para o Modelo 16×64



Fonte: Elaborada pelo autor.

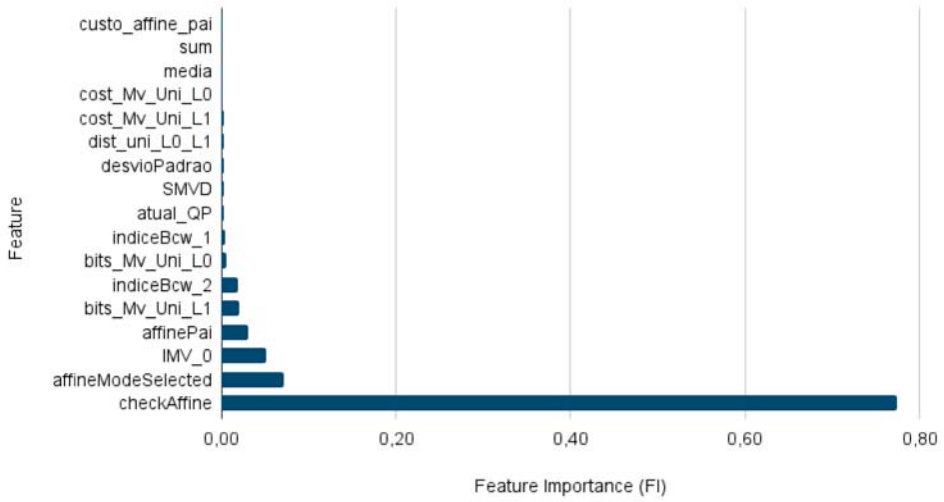
(quando comparado aos demais casos), para as outras três *features* da sequência, `affineModeSelected`, `indiceBcw_2` e `affinePai`.

A Figura 74 por sua vez, demonstra uma semelhança do modelo 64×32 com o anterior. O destaque continua sendo a *feature* `checkaffine`, com valor 0,71. Mesmo assim, as outras quatro *features* que aparecem na sequência têm valores relativamente maiores, sendo elas `affineModeSelected`, `indiceBcw_2` e `affinePai`.

Também é possível notar um comportamento semelhante na Figura 75, do modelo 64×64. A ordem das 5 primeiras *features* é a mesma que a anterior. A *feature* `checkaffine` aparece em destaque com 0,72, seguida por `affineModeSelected` com 0,18, `indiceBcw_2` e `affinePai`, com os valores de 0,05 e 0,02, respectivamente.

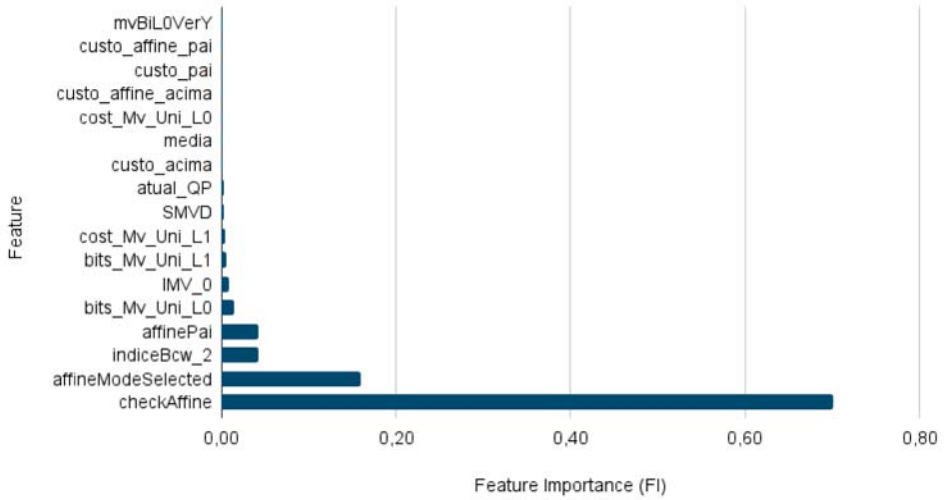
Já o modelo 64×128, ilustrado na Figura 76, apesar de manter o destaque para

Figura 72 – *Feature Importance* para o Modelo 64×16



Fonte: Elaborada pelo autor.

Figura 73 – *Feature Importance* para o Modelo 32×64



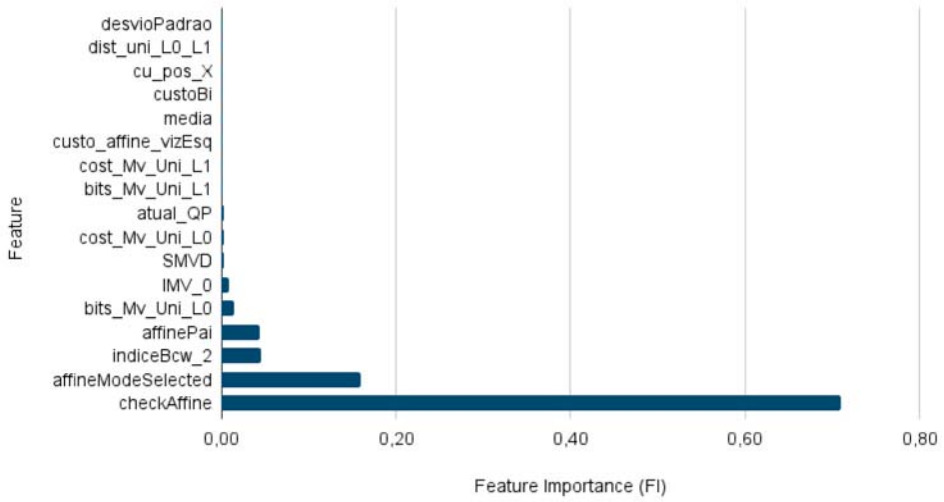
Fonte: Elaborada pelo autor.

a *feature* `checkaffine` (0,56), apresenta um maior equilíbrio entre os demais valores. São outras 8 *features* visíveis no gráfico, com valores entre 0,01 até 0,19, com um destaque para três diferentes valores relacionados ao `IMV`.

A Figura 77, que se refere ao modelo 128×64, já apresenta um comportamento diferente. Apenas quatro *features* aparecem claramente no gráfico, sendo elas `checkaffine`(com FI de 0,6), `affineModeSelected` (0,28), seguidas por `indiceBcw_2` e `affinePai`, com 0,07 e 0,03, respectivamente.

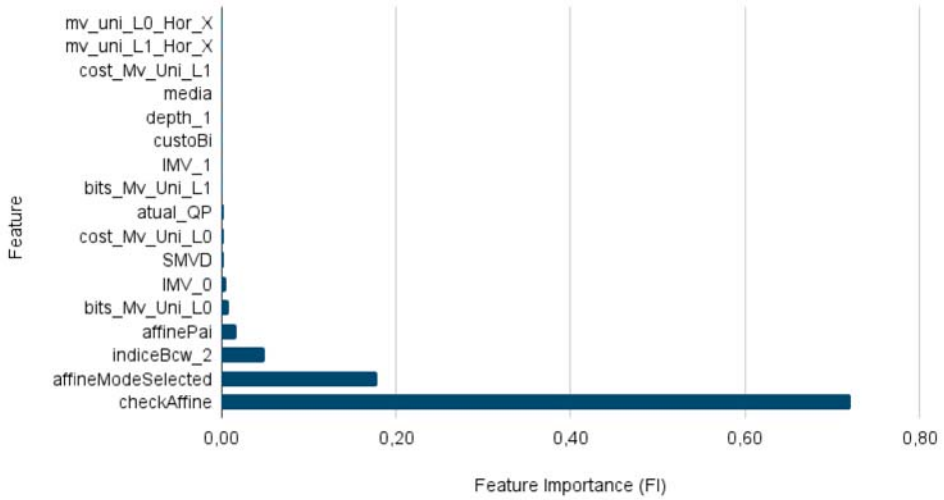
Por fim, na Figura 78, o modelo 128×128 se apresenta com três *features* em destaque: `checkaffine`, `affineModeSelected` e `indiceBcw_2`, com os respectivos valores de FI de 0,49, 0,34 e 0,11. Além disso, também apresenta várias outras *features* com valores menores, sendo que dentre elas estão `custoBi` e `custoVizEsq`, por exem-

Figura 74 – *Feature Importance* para o Modelo 64×32



Fonte: Elaborada pelo autor.

Figura 75 – *Feature Importance* para o Modelo 64×64



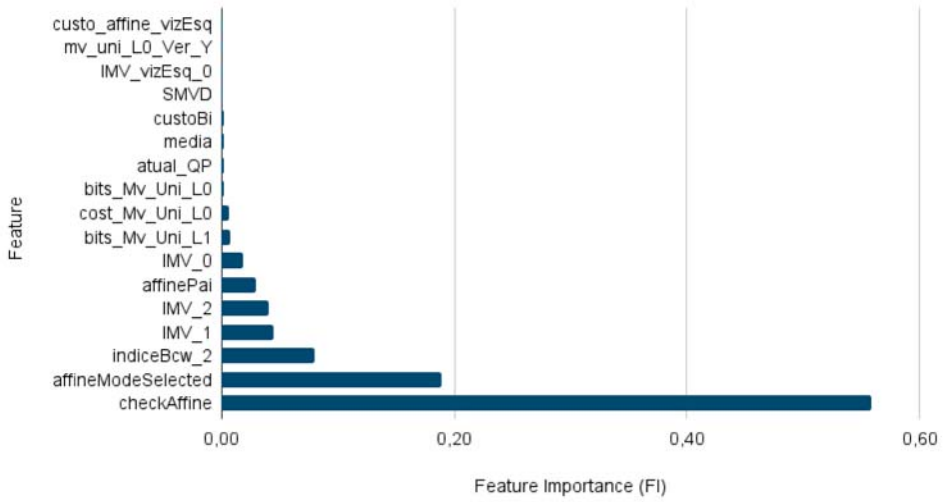
Fonte: Elaborada pelo autor.

plo, que não haviam aparecido anteriormente. Nota-se ainda que, pelo fato do bloco 128×128 ser o primeiro da árvore de particionamento, o mesmo não possui *features* relacionadas aos valores do bloco pai.

De maneira geral, é notável que na maior parte dos modelos, as *features* `checkaffine` e `affineModeSelected` se destacam das demais, com valores maiores de FI. Entretanto, outras *features* também aparecem como predominantes, mesmo que de maneira secundária, sendo elas relacionadas ao índice BCW, ao IMV, aos vetores de movimento da Unidirecional e até aos valores calculados a partir das amostras do bloco.

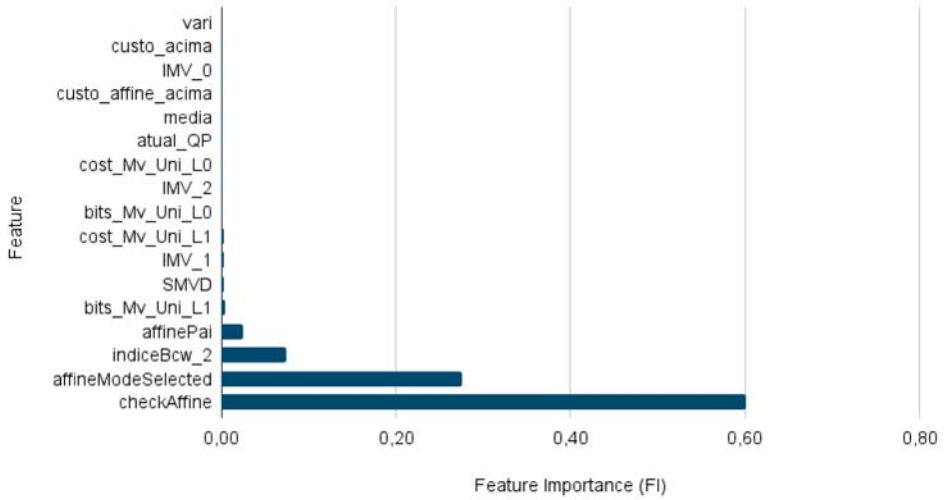
Dessa forma, o processo de seleção das *features* levou em consideração alguns critérios, já utilizados nas soluções anteriores, tais como: selecionar *features* que apa-

Figura 76 – *Feature Importance* para o Modelo 64×128



Fonte: Elaborada pelo autor.

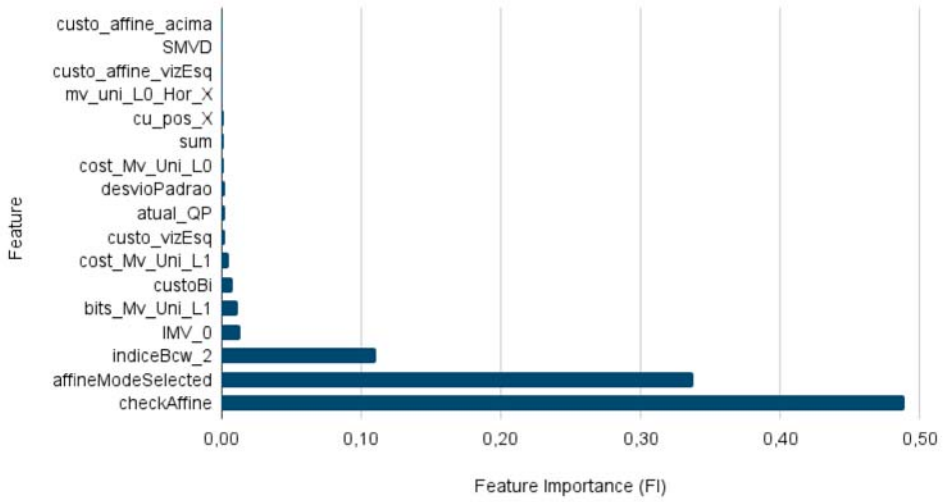
Figura 77 – *Feature Importance* para o Modelo 128×64



Fonte: Elaborada pelo autor.

recem com relevância em pelo menos dois modelos e manter as mesmas *features* para todos os modelos (com exceção do 128×128 que não possui dados para bloco pai). Além disso, também foi considerado o desempenho dos modelos nos primeiros testes. Esse desempenho demonstrou um provável sobreajuste dos modelos aos dados de treinamento, visto que na prática, não atingiram resultados aceitáveis. Por esse motivo, decidiu-se excluir as *features* `checkaffine` e `affineModeSelected` do processo de busca de hiperparâmetros e treinamento final. Sendo assim, as *features* selecionadas são apresentadas na Tabela 45.

Figura 78 – *Feature Importance* para o Modelo 128×128



Fonte: Elaborada pelo autor.

10.3.3 Treino Final dos Modelos

O processo do treinamento final ocorre primeiramente com base nos *datasets* parciais (com 75% dos registros) e teste (com os 25% dos registros) para fim de verificação das métricas de desempenho. Após isso, cada um dos 12 *datasets* é unificado, sendo realizado então o treinamento final de cada modelo para fins de implementação no VTM. É importante salientar que esse treinamento final considera os hiperparâmetros (Tabela 44) e as *features* (Tabela 45) selecionadas anteriormente. A seguir, na Tabela 46 são apresentados os resultados deste teste final.

De maneira geral, é possível notar que a acurácia dos modelos se manteve equilibrada e com média de 87,8%, o que demonstra uma capacidade de acerto elevada e aceitável. Nesse sentido, o modelo 64×16 se destaca com a maior acurácia, de 89%. Por outro lado, o modelo 64×64 atingiu 85%, sendo este o menor valor.

Em termos de precisão, a Classe 0 (“não executa *Affine*”) obteve um desempenho relativamente melhor em relação à Classe 1 (“executar *Affine*”), ou seja, das predições realizadas pelos modelos houve maior acerto quando se trata da Classe 0. Sendo assim, a precisão para a Classe 0 ficou entre 86%, para o modelo 64×64, e 93%, obtida para os modelos 16×16 e 128×128. Já a precisão para a Classe 1 foi de 83%, obtida no modelo 64×128, até 86%.

Em termos de *recall* os desempenhos dos modelos para as classes se invertem. Na Classe 0, os valores de *recall* ficam entre 81% e 86%, para os modelos 64×128 e 16×64, respectivamente. Já para a Classe 1, os valores estão entre 86% para o modelo 64×64 e 94% para os modelos 16×16 e 128×128. Esses desempenhos demonstram que, dos registros de cada classe, os modelos apresentam maior capacidade de classificar corretamente os registros da Classe 1.

Tabela 45 – Relação das *Features* utilizadas no treinamento final

Feature	**Qtd**	**Descrição**
indiceBcw	5	Índice relacionado ao peso usado na BCW
imv	4	Precisão do Vetor de Movimento (AMVR)
atual_QP	1	Parâmetro de quantização utilizado no momento
mv_uni	4	Vetor de Movimento (X, Y) gerado pela predição Unidirecional para a Lista 0 e Lista 1
cost_mv_uni	2	Custo do Vetor de Movimento gerado pela predição Unidirecional para a Lista 0 e Lista 1
bits_mv_uni	2	Quantidade de bits do Vetor de Movimento gerado pela predição Unidirecional para a Lista 0 e Lista 1
cost_mv_bi	1	Custo do Vetor de Movimento gerado pela predição Bidirecional
SMVD	1	Indica se o SMVD está sendo testado ou não
bidirecional_pai	1	Indica se o bloco pai foi codificado pela predição Bidirecional
custo_pai	1	Custo gerado pela predição Bidirecional do bloco pai
IMV_pai	5	Valor do IMV do bloco pai
affine_pai	1	Indica se o bloco pai foi codificado pela predição *Affine*
custo_affine_pai	1	Custo gerado pela predição *Affine* do bloco pai
bidirecional_viz_esq	1	Indica se o bloco vizinho a esquerda foi codificado pela predição Bidirecional
custo_viz_esq	1	Custo gerado pela predição do bloco vizinho a esquerda
IMV_viz_esq	5	Valor do IMV do bloco vizinho a esquerda
affineVizEsq	1	Indica se o bloco vizinho à esquerda foi codificado pela predição *Affine*
custo_affine_VizEsq	1	Custo gerado pela predição *Affine* do bloco vizinho à esquerda
bidirecional_viz_acima	1	Indica se o bloco vizinho acima foi codificado pela predição Bidirecional
custo_viz_acima	1	Custo gerado pela predição do bloco vizinho acima
IMV_viz_acima	5	Valor do IMV do bloco vizinho acima
affineVizAci	1	Indica se o bloco vizinho acima foi codificado pela predição *Affine*
custo_affine_acima	1	Custo gerado pela predição *Affine* do bloco vizinho acima
sum	1	Somatório das amostras do bloco atual
media	1	Média das amostras do bloco atual
vari	1	Variância das amostras do bloco atual
desvioPadrao	1	Desvio padrão das amostras do bloco atual
dist_uni_L0_L1	1	Distância entre os Vetores de Movimento resultantes da predição Unidirecional da Lista 0 e Lista 1
bloco_atual_affine	1	Indica se o bloco atual possui o melhor modo de predição como sendo *Affine*

Tabela 46 – Resultados do treinamento e teste finais

Métrica/ Modelo		Acurácia	Precisão	Recall	F1-score	Falsos Negativos	Falsos Positivos
16×16	0	88%	93%	83%	88%	2,9%	8,7%
	1		84%	94%	89%		
16×32	0	88%	89%	85%	87%	5,0%	7,5%
	1		86%	90%	88%		
32×16	0	88%	92%	83%	87%	3,6%	8,6%
	1		84%	93%	88%		
32×32	0	88%	90%	85%	88%	4,5%	7,3%
	1		86%	91%	88%		
16×64	0	88%	90%	86%	88%	5,0%	7,1%
	1		86%	90%	88%		
64×16	0	89%	91%	85%	88%	4,2%	7,9%
	1		86%	92%	89%		
32×64	0	88%	91%	84%	87%	4,1%	8,1%
	1		85%	92%	88%		
64×32	0	88%	91%	84%	87%	4,0%	8,1%
	1		85%	92%	88%		
64×64	0	85%	86%	85%	85%	7,0%	7,5%
	1		85%	86%	86%		
64×128	0	88%	92%	81%	86%	3,5%	9,3%
	1		83%	93%	88%		
128×64	0	88%	92%	82%	87%	3,7%	8,9%
	1		84%	93%	88%		
128×128	0	88%	93%	82%	87%	3,2%	9,2%
	1		84%	94%	88%		
Média	**0**	87,8%	90,8%	83,8%	87,1%	4,2%	8,1%
	1		84,8%	91,7%	88,0%		

Já a *F1-score*, sendo a média harmônica entre as métricas precisão e *recall*, representa o desempenho geral dos modelos, que demonstram estar equilibrados entre as duas classes. Em geral, a F1 fica entre 85% e 88% para a Classe 0 e 86% e 88% para a Classe 1. Em se tratando da taxa de Falsos Negativos, que se refere ao número de registros preditos como sendo da Classe 0, porém o correto seria da Classe 1, os modelos tiveram desempenhos distintos. O melhor desempenho foi do modelo 16×16, com 2,9%; já o pior foi do modelo 64×64, com 7,0%. Esses valores podem representar um possível impacto em BDBR, no momento da implementação e teste dos modelos no VTM. Quanto à taxa de Falsos Positivos, em que os modelos predizem registros como sendo da Classe 1, porém o correto seria a Classe 0, nota-se que os percentuais são maiores. O melhor desempenho foi do modelo 16×64, com 7,1% e o pior foi do modelo 64×128, com 9,3%. Esses percentuais indicam um possível impacto na redução do tempo de codificação, visto que os modelos podem errar as predições, indicando a necessidade de executar a predição *Affine* sendo que o correto

seria não executá-la.

10.4 Implementação no VTM

No que diz respeito à implementação dos modelos no VTM, a solução de otimização para a *Affine* apresentada neste capítulo segue um procedimento semelhante ao descrito nos capítulos anteriores.

Após a finalização do treinamento, cada modelo foi convertido para a linguagem C++ utilizando a função `export_to_c`, disponibilizada pela biblioteca `m2cgen` (Zeigerman, 2022). A implementação de cada modelo foi realizada na versão do VTM com custo reduzido, por meio da criação de classes na pasta `CommonLib`. Depois de concluída a etapa Bidirecional padrão, as *features* extraídas são utilizadas como entrada para um dos 12 modelos de otimização da *Affine*, conforme o tamanho do bloco processado. Mantendo a lógica das otimizações anteriores, foi adotado o ponto de corte de 0,6 para as decisões dos modelos, como ilustrado na Figura 66. Assim, a etapa *Affine* é ignorada quando a pontuação retornada para a classe 0 ultrapassa 0,4. Por outro lado, se a pontuação da classe 1 for igual ou superior a 0,6, a predição *Affine* é realizada normalmente.

Para a avaliação dos resultados da otimização da *Affine*, foram utilizadas três versões do VTM: original, com custo reduzido e sem a predição *Affine*. A versão original do VTM foi adotada como referência para as análises, enquanto a versão com custo reduzido permitiu avaliar o desempenho dos modelos treinados em cenários reais. Já a versão sem a etapa *Affine* serviu como parâmetro para definir os limites teóricos de redução do tempo de execução e de perda na eficiência de codificação. Novamente, o cenário ideal seria que a solução alcançasse uma redução de tempo próxima à obtida pela remoção da ferramenta *Affine*, mas com perda de eficiência de codificação mínima ou inexistente.

A versão do VTM sem a predição *Affine* foi ajustada para desativar exclusivamente essa ferramenta, mantendo as demais funcionalidades ativas conforme o padrão. Assim como nos capítulos anteriores, as três versões do VTM foram modificadas para extrair métricas relacionadas ao tempo de execução de diferentes etapas (intraquadro e interquadros, com predições Unidirecional, Bidirecional e *Affine*, além da ferramenta Merge). Por fim, a versão com custo reduzido também foi adaptada para registrar a contagem de decisões gerais tomadas pelos modelos.

10.5 Resultados no VTM

Nesta seção serão apresentados e descritos os resultados obtidos a partir da implementação dos 12 modelos de *Decision Tree* para a redução do custo computacional

da etapa *Affine* do VVC. Para os experimentos de teste foram utilizados os mesmos 15 vídeos das CTCs, conforme já apresentado anteriormente na Tabela 23. A codificação dos vídeos foi realizada no VTM 22.0, utilizando a configuração *Random Access*. Foram codificados os primeiros 32 quadros de cada vídeo, com quatro repetições para cada QP (22, 27, 32 e 37), em três versões do VTM (original, com custo reduzido e sem a predição *Affine*).

As métricas utilizadas para avaliar o desempenho dos modelos foram o tempo de codificação e o BDBR. O VTM já fornece informações sobre o tempo total, a taxa de bits e o YUV-PSNR. Os valores correspondentes ao tempo de codificação de cada etapa e das decisões dos modelos foram obtidos a partir das modificações mencionadas anteriormente. Os cálculos seguiram as mesmas equações descritas na Seção 8.6. Serão apresentados os valores médios de redução de tempo total (MRT), bem como das etapas interquadros (MRT_{Inter}), intraquadro (MRT_{Intra}), Merge (MRT_{Merge}), e das predições Unidirecional (MRT_{uni}), Bidirecional (MRT_{bi}) e *Affine* (MRT_{aff}). Além disso, será avaliado o *overhead* da solução, considerando as médias de tempo dos modelos (MT_{DT}) e da extração das *features* (MT_{feat}). Por fim, o $BDBR$ será apresentado juntamente com a relação entre a economia de tempo e o $BDBR$ ($MRT/BDBR$).

Na sequência são apresentados os resultados comparativos da solução de redução de custo para a predição *Affine* e o VTM original. Na Tabela 47 constam os valores de redução de tempo total e da predição *Affine*, além do *overhead* da solução. A seguir, os resultados de redução de tempo para a Unidirecional, Bidirecional, além das etapas interquadros, intraquadro e Merge, são apresentados de maneira adicional na Tabela 48. A Tabela 49, por sua vez, apresenta a comparação entre a redução no tempo total e o impacto na eficiência de codificação. Por fim, na Tabela 50 também é apresentada a comparação entre os resultados da solução proposta com a versão do VTM sem a predição *Affine*.

Os resultados apresentados na Tabela 47 demonstram que a solução de otimização utilizando DTs, alcançou valores moderados de redução no tempo de codificação da predição *Affine*. Em geral, a média de redução foi de 42,1%, sendo que para as classes a média ficou entre 36,7%, para a classe A1, até 52,7%, para a classe F. É possível perceber também que, conforme a resolução dos vídeos aumenta, o desempenho da solução acaba diminuindo. Além disso, em alguns vídeos a redução chega a ser próxima dos 20%, o que pode representar que a solução não foi eficiente devido a esses vídeos terem características específicas de menor movimento. Quanto à redução de tempo total, os resultados da solução se apresentam em uma faixa de valores próxima aos valores obtidos nas soluções dos capítulos 8 e 9. Em 8 dos 15 vídeos, a redução no tempo total ultrapassou o resultado obtido na solução ampliada (capítulo 9). Quanto ao *overhead*, é importante salientar que, nessa solução, foram consideradas *features* dos blocos vizinhos e *features* calculadas, que geralmente ocu-

Tabela 47 – Impacto dos modelos na redução de tempo de execução

Vídeo/ Classe	MRT_{aff} **(%)**	MRT_{Total} **(%)**	MT_{DT} **(%)**	MT_{feat} **(%)**
ArenaOfValor	45,8	3,9	0,1	0,9
BasketballDrillText	44,4	2,7	0,1	1,0
SlideEditing	66,6	3,9	0,1	0,8
SlideShow	54,1	4,9	0,0	0,8
Média Classe F	**52,7**	**3,8**	**0,1**	**0,9**
BasketballDrive	45,8	4,4	0,1	0,8
BQTerrace	52,3	4,4	0,1	0,9
Cactus	46,4	4,6	0,1	0,9
MarketPlace	21,8	-1,2	0,0	0,6
RitualDance	29,0	-1,1	0,0	0,6
Média Classe B	**39,1**	**2,2**	**0,0**	**0,7**
Campfire	24,5	0,0	0,0	0,8
FoodMarket4	48,1	6,8	0,1	0,7
Tango2	37,6	4,2	0,1	0,8
Média Classe A1	**36,7**	**3,7**	**0,1**	**0,8**
CatRobot	41,6	5,7	0,1	0,9
DaylightRoad2	39,4	6,5	0,1	0,8
ParkRunning3	34,6	1,9	0,0	0,8
Média Classe A2	**38,6**	**4,7**	**0,1**	**0,8**
Média Geral	**42,1**	**3,4**	**0,1**	**0,8**

pam um tempo maior na extração. Mesmo assim, os valores relativos à extração das *features* e do processamento das DTs são baixos, não impactando significativamente os demais resultados.

Já a Tabela 48 apresenta os resultados referentes à redução do tempo de codificação nas demais ferramentas do codificador. É possível notar que, para a predição Bidirecional, houve um impacto negativo, ou seja, um aumento no tempo de codificação, embora tenha sido de no máximo 1,1%. Analisando os resultados para a predição Unidirecional, nota-se que não houve impactos significativos, resultando em uma média geral de apenas 0,4% de redução. Já para a etapa interquadros, percebe-se que houve redução, mesmo que moderada, na maioria dos vídeos. Por fim, ao analisar o efeito da solução nas etapas intraquadro e Merge, observa-se que ambas, em média, tiveram aumento no tempo de codificação. Apesar de serem baixos, -1,3% no máximo, esses impactos se apresentam em 13 dos 15 vídeos, no caso da ferramenta intraquadro.

Tabela 48 – Impacto dos modelos no tempo total e das principais ferramentas do codificador

Vídeo/ Classe	MRT						
	Bi **(%)**	Uni **(%)**	Aff **(%)**	$Inter$ **(%)**	$Intra$ **(%)**	$Merge$ **(%)**	$Total$ **(%)**
ArenaOfValor	0,0	0,4	45,8	11,8	-0,5	-0,4	3,9
BasketballDrillText	-0,7	0,5	44,4	9,4	-0,7	-0,5	2,7
SlideEditing	3,4	3,1	66,6	16,3	-1,0	-0,6	3,9
SlideShow	0,8	1,7	54,1	16,8	-0,1	0,1	4,9
Média Classe F	**0,9**	**1,4**	**52,7**	**13,6**	**-0,6**	**-0,3**	**3,8**
BasketballDrive	-0,9	-0,4	45,8	12,4	-0,9	-0,3	4,4
BQTerrace	0,3	0,7	52,3	12,8	-0,5	-0,1	4,4
Cactus	0,3	0,6	46,4	13,5	-0,7	0,1	4,6
MarketPlace	-0,8	0,2	21,8	-3,1	-0,2	-0,4	-1,2
RitualDance	-0,5	0,3	29,0	-2,0	-0,8	-0,3	-1,1
Média Classe B	**-0,3**	**0,3**	**39,1**	**6,7**	**-0,6**	**-0,2**	**2,2**
Campfire	-0,7	0,0	24,5	0,1	0,4	0,1	0,0
FoodMarket4	-0,6	0,3	48,1	16,9	0,7	0,1	6,8
Tango2	-1,1	-0,2	37,6	10,7	-0,2	-0,5	4,2
Média Classe A1	**-0,8**	**0,1**	**36,7**	**9,2**	**0,3**	**-0,1**	**3,7**
CatRobot	-1,1	-0,6	41,6	14,2	-1,3	-0,3	5,7
DaylightRoad2	-0,6	0,3	39,4	14,3	-0,7	0,2	6,5
ParkRunning3	-1,0	-0,2	34,6	6,6	-1,2	-0,5	1,9
Média Classe A2	**-0,9**	**-0,2**	**38,6**	**11,7**	**-1,0**	**-0,2**	**4,7**
Média Geral	**-0,2**	**0,4**	**42,1**	**10,0**	**-0,5**	**-0,2**	**3,4**

A seguir, os resultados referentes à redução no tempo total de codificação, bem como o impacto na eficiência de codificação, são apresentados na Tabela 49. Além disso, também é apresentada a relação entre a redução de tempo e o impacto em BDBR, sendo que valores mais altos indicam situações em que houve maior redução no tempo ao custo de um baixo impacto na eficiência de codificação. Nesse caso, é possível notar que para a maioria dos vídeos essa relação ficou entre 7 e 15, ou seja, redução moderada no tempo de codificação, com impacto também moderado em BDBR. Entretanto, alguns vídeos se destacam com valores altos, entre 44 e 50, indicando redução moderada no tempo de codificação com impacto mínimo em BDBR, como, por exemplo, os vídeos ArenaOfValor, SlideEditing e BQTerrace. Já o vídeo FoodMarket4 se destaca com o melhor resultado, obtendo 360,6, visto que atingiu a maior redução no tempo total ao custo mais baixo de BDBR.

Por fim, a Tabela 50 apresenta uma comparação da redução do tempo total de codificação e do impacto em BDBR entre a versão do VTM otimizada usando DTs e a versão sem a ferramenta *Affine*. Em uma primeira análise, fica evidente que a versão do VTM com DT obteve uma redução de tempo total menos significativa do que a versão sem *Affine*, o que já era esperado. Ao analisar a diferença da média de redução de tempo entre as versões do VTM, observa-se que a versão com DTs

Tabela 49 – Impacto dos modelos na eficiência de codificação

Vídeo/ Classe	MRT **(%)**	$BDBR$ **(%)**	$MRT/BDBR$
ArenaOfValor	3,9	0,077	50,3
BasketballDrillText	2,7	0,354	7,6
SlideEditing	3,9	0,088	44,0
SlideShow	4,9	0,616	7,9
Média Classe F	**3,8**	**0,284**	**27,5**
BasketballDrive	4,4	0,342	12,8
BQTerrace	4,4	0,092	47,7
Cactus	4,6	0,342	13,4
MarketPlace	-1,2	0,095	-12,9
RitualDance	-1,1	0,127	-8,9
Média Classe B	**2,2**	**0,199**	**10,4**
Campfire	0,0	0,069	-0,6
FoodMarket4	6,8	0,019	360,6
Tango2	4,2	0,276	15,4
Média Classe A1	**3,7**	**0,121**	**125,1**
CatRobot	5,7	0,453	12,6
DaylightRoad2	6,5	0,640	10,2
ParkRunning3	1,9	0,132	14,2
Média Classe A2	**4,7**	**0,408**	**11,5**
Média Geral	**3,4**	**0,248**	**38,3**

é cerca de três vezes menor do que a atingida com a completa exclusão da *Affine*, valor semelhante à diferença apresentada no capítulo 8. Por outro lado, ao analisar o impacto de BDBR entre as duas versões do VTM, nota-se que a versão com DTs tem uma melhora significativa em BDBR, que é próxima de 10 vezes melhor do que na versão com a exclusão da *Affine*. Apenas como comparativo, a solução do capítulo 8, apresentou uma diferença mínima nessa comparação. Portanto, pode-se afirmar que, em comparação com a solução anterior voltada para a Bidirecional, a solução atual, focada na Affine, obteve resultados significativos, principalmente em termos de baixo impacto na eficiência de codificação.

A Tabela 51 apresenta a comparação entre a solução de otimização para a *Affine* e os trabalhos relacionados, que foram apresentados no capítulo 6. Os trabalhos são comparados considerando a versão do VTM, a técnica utilizada (AE - Análise Estatística, RF - *Random Forests*; DT - *Decision Trees*), as médias de redução de tempo da etapa *Affine* (MRT_{aff}) e total (MRT_{Total}), além do BDBR. Nota-se que, dentre as versões do VTM utilizadas, a solução proposta é a que utiliza a versão mais recente. Em termos de redução de tempo da etapa Affine, a solução deste capítulo alcançou 42,1%, ficando atrás apenas do trabalho de Duarte et al. (2022) (46,9%) que utiliza RF. Ao analisar a redução de tempo total, a solução proposta atingiu 3,4% em média, valor menor que a maioria dos trabalhos. Entretanto, é necessário considerar que os

Tabela 50 – Comparação do VTM com redução de custo e do VTM sem a predição *Affine*

Vídeo/ Classe	MRT_{Total}		$BDBR$	
	Com DT (%)	Sem Affine (%)	Com DT (%)	Sem Affine (%)
ArenaOfValor	3,9	13,2	0,077	1,532
BasketballDrillText	2,7	10,8	0,354	0,470
SlideEditing	3,9	8,2	0,088	-0,277
SlideShow	4,9	7,8	0,616	2,230
Média Classe F	**3,8**	**10,0**	**0,284**	**0,989**
BasketballDrive	4,4	13,0	0,342	2,028
BQTerrace	4,4	11,0	0,092	0,097
Cactus	4,6	13,0	0,342	5,700
MarketPlace	-1,2	1,5	0,095	0,269
RitualDance	-1,1	1,9	0,127	0,329
Média Classe B	**2,2**	**8,1**	**0,199**	**1,685**
Campfire	0,0	5,3	0,069	0,193
FoodMarket4	6,8	16,2	0,019	0,175
Tango2	4,2	13,6	0,276	1,028
Média Classe A1	**3,7**	**11,7**	**0,121**	**0,465**
CatRobot	5,7	17,3	0,453	6,030
DaylightRoad2	6,5	18,2	0,640	9,908
ParkRunning3	1,9	10,4	0,132	5,673
Média Classe A2	**4,7**	**15,3**	**0,408**	**7,204**
Média Geral	**3,4**	**10,8**	**0,248**	**2,359**

demais trabalhos utilizam versões iniciais do VTM, ou seja, anteriores às otimizações da ferramenta que estão presentes na versão 22.0. Ao comparar os resultados com o trabalho de Sagrilo et al. (2023), que utiliza a versão mais próxima do VTM, a solução deste capítulo é superior em termos de redução de tempo de codificação da etapa e total. Analisando o impacto na eficiência de compressão, o BDBR, a solução proposta tem melhor resultado do que os trabalhos Hong et al. (2023) e Pejman et al. (2023). Porém, é importante ressaltar que o resultado de 0,25% em BDBR da solução proposta fica próximo dos demais trabalhos, sendo aceitável mediante a redução de tempo alcançada. Dessa forma, é possível afirmar que a abordagem proposta demonstra-se competitiva em relação aos trabalhos analisados, oferecendo uma alternativa eficaz que reduz o tempo de codificação sem comprometer excessivamente a eficiência de compressão.

Tabela 51 – Comparação da solução proposta com trabalhos relacionados

Trabalho	**Versão VTM**	**Técnica**	MRT_{aff} **(%)**	MRT_{Total} **(%)**	$BDBR$ **(%)**
Park; Kang (2019)	3.0	AE	37,0	5,0	0,10
Ren; He; Cui (2020)	7.0	AE	30,0	6,0	0,21
Jung; Jun (2021)	10.0	AE	33,0	5,0	0,04
Li et al. (2022)	7.0	AE	40,8	10,1	0,16
Hong et al. (2023)	10.0	AE	24,8	6,2	0,76
Pejman et al. (2023)	14.0	AE	-	10,6	0,88
Duarte et al. (2022)	9.0	RF	46,9	8,5	0,18
Sagrilo et al. (2023)	16.2	RF	19,6	2,9	0,07
Solução Proposta (Capítulo 10)	**22.0**	**DT**	**42,1**	**3,4**	**0,25**

11 CONCLUSÃO

Esta tese apresentou quatro soluções voltadas para a redução do tempo de codificação da etapa interquadros do padrão de codificação de vídeo VVC. Também foram descritos os principais conceitos envolvendo o padrão VVC, as ferramentas específicas da predição interquadros, além dos conceitos básicos de aprendizado de máquina relacionados às soluções. Além disso, foram apresentados resultados de uma análise experimental sobre a predição interquadros do VVC, bem como a revisão sistemática da literatura realizada sobre o tema.

No capítulo 7 foi apresentada uma heurística configurável para redução do custo computacional das predições Unidirecional, Bidirecional e *Affine*. A heurística foi definida usando três pontos de operação e foi especializada para três resoluções de vídeo (HD, FHD e UHD). Apesar de não ser o foco principal desta tese, essa heurística foi apresentada com os resultados de estimativa de redução de energia, visando uma possível implementação em hardware. Os resultados experimentais mostraram reduções no consumo de energia entre 7,69% e 30,77%, dependendo do ponto de operação e da resolução do vídeo. Esses resultados são expressivos, considerando que atualmente a maioria dos aplicativos de vídeo está sendo executada em dispositivos alimentados por bateria. Os experimentos também mostraram reduções de tempo variando consistentemente com os pontos de operação. Especificamente, nas etapas de ME Unidirecional e Bidirecional, as reduções de tempo de codificação variaram de 3,57% a 26,5%, dependendo da resolução do vídeo e do ponto de operação selecionado. Considerando a *Affine*, a redução de tempo variou de 4,7% a 22,71%. Em termos de perda de eficiência de codificação, o BDBR variou de 0,04% a 1%. Embora alguns trabalhos relacionados se concentrem em reduções de tempo de codificação para as mesmas ferramentas individualmente, eles são baseados apenas em otimizações de software, que não são aplicadas diretamente a projetos de hardware. É importante ressaltar que a heurística proposta também pode ser configurada para incluir pontos de operação mais agressivos se uma maior redução no consumo de energia for necessária. Naturalmente, quanto maior for a economia de energia, maiores tendem a ser as perdas de eficiência de codificação. Este também é o primeiro

trabalho na literatura com foco em soluções otimizadas para predições Unidirecional, Bidirecional e *Affine* do VVC em conjunto.

O capítulo 8 apresentou uma solução para redução de custo computacional da predição Bidirecional utilizando o modelo de aprendizado de máquina *Random Forests*. Em geral, é possível concluir que a abordagem proposta traz reduções significativas no tempo de codificação da etapa Bidirecional, chegando a atingir a média de 92% de redução. Porém, a solução também demonstrou um impacto significativo na etapa *Affine*, que sofreu um aumento no tempo de codificação de quase 30% em média. Já em termos de impacto na eficiência de compressão, a solução demonstrou um bom desempenho, porém muito próximo ao obtido na versão sem a ferramenta Bidirecional. Considerando o tempo total de codificação, a solução obteve resultados modestos, com ganhos médios gerais de 2,0%. Quanto à implementação da solução, notou-se algumas dificuldades iniciais com o tamanho dos modelos, sendo necessária uma seleção criteriosa dos hiperparâmetros, além da redução do número de *features*, conforme a relevância em cada modelo.

Já o capítulo 9 definiu uma solução ampliada para otimização das etapas Bidirecional e *Affine* do padrão de codificação de vídeo VVC. Para essa otimização, foi considerada a solução do capítulo 8, utilizando RFs, juntamente com a solução de otimização para a *Affine*, que utiliza DTs. Os resultados obtidos demonstram que foi possível reduzir o impacto da otimização da etapa Bidirecional na *Affine*, porém sem obter redução nessa etapa em específico. Em média, foi possível reduzir esse impacto negativo em cerca de 10%. Com essa estratégia ampliada, também notou-se uma redução no tempo total de codificação mais significativa, em comparação com a solução do capítulo 8. É importante ressaltar que, em termos de impacto na eficiência de codificação, a solução ampliada manteve praticamente os mesmos valores da solução do capítulo 8. Isso demonstra que a solução ampliada atinge resultados superiores em termos de redução do tempo total de codificação, com o mesmo impacto na eficiência de codificação, quando comparada à solução de RFs para otimização da Bidirecional.

Por fim, o capítulo 10 detalhou a solução de otimização voltada para a etapa *Affine*, utilizando modelos de DTs. Apesar de ter obtido resultados mais modestos em termos de redução do tempo de codificação da etapa específica, a solução apresentou resultados próximos aos da solução ampliada ao analisar a redução no tempo total. A otimização atingiu a média de 42,1% de redução no tempo de codificação da etapa *Affine* e 3,4% no tempo total. Já em relação ao impacto na eficiência de codificação, a proposta voltada para a *Affine* obteve os melhores resultados, quando comparada com as duas soluções anteriores propostas nesta tese. O valor de BDBR, em média, atingiu 0,25%, o que indica que a solução é competitiva com os demais trabalhos relacionados.

Em síntese, os resultados das três soluções apresentadas nesta tese, que utilizam

aprendizado de máquina, podem ser considerados em termos de valores máximo e mínimo atingidos. Primeiramente, a solução descrita no capítulo 9, voltada para a otimização da etapa Bidirecional com o uso de RFs, atingiu o máximo de 95,6% (FoodMarket4, classe A1) e o mínimo de 85,7% (MarketPlace, classe B) de redução de tempo de codificação da etapa Bidirecional. Já em termos de redução no tempo total de codificação, a solução ficou entre 0,5% (MarketPlace, classe B) e 3,9% (SlideEditing, classe F). O impacto de BDBR variou de -0,06% (SlideEditing, classe F) até 1,47% (Tango2, classe A1). Já a solução ampliada, focada na otimização das etapas Bidirecional e *Affine* (capítulo 9), alcançou médias de redução no tempo de codificação para a Bidirecional entre 86,0% (MarketPlace, classe B) e 95,7% (FoodMarket4, classe A1). Para a etapa *Affine*, a solução diminuiu o impacto, obtendo de -37,5% (BQTerrace, classe B) até 5,1% (SlideEditing, classe F). Já a redução no tempo total de codificação ficou entre 1,8% (BasketballDrillText, classe F) e 7,0% (SlideEditing, classe F). O impacto de BDBR da solução ampliada variou entre -0,05% (SlideEditing, classe F) e 1,36% (Tango2, classe A1). Por fim, a solução de otimização voltada para a *Affine*, apresentada no capítulo 10, obteve médias de redução no tempo de codificação da etapa *Affine* de 21,8% (MarketPlace, classe B) até 66,6% (SlideEditing, classe F). Para o tempo total de codificação, a solução atingiu de -1,2% (MarketPlace, classe B) até 6,8% (FoodMarket, classe A1). Já o impacto na eficiência de codificação, BDBR, variou entre 0,02% (FoodMarket4, classe A1) e 0,64% (DaylightRoad2, classe A2).

De maneira geral, é possível comparar essas três soluções com alguns trabalhos relacionados, descritos no capítulo 6. Apesar das diferenças significativas, a solução do capítulo 8, com foco na etapa Bidirecional, pode ser comparada com o trabalho de Lee e Jun (2023), cujo foco é a ferramenta BCW, contida na Bidirecional. O trabalho de Lee e Jun (2023), atinge 33% de redução na etapa específica, com impacto de 0,26% em BDBR. A solução proposta nesta tese alcança 92% de redução no tempo de codificação da etapa Bidirecional, com maior impacto em BDBR (0,75%).

A solução ampliada, descrita no capítulo 9, que visa otimizar as etapas Bidirecional e *Affine*, pode ser comparada, de maneira limitada, com quatro trabalhos relacionados. Apesar de atingir uma redução menor em termos de tempo total, a solução ampliada possui resultados melhores para BDBR, quando comparada com os trabalhos de Xie et al. (2022) e Chan; Im 2023. Além disso, é importante ressaltar que os quatro trabalhos relacionados utilizam versões anteriores ao VTM 22.0 e possuem focos diversificados, o que impede uma comparação direta.

Por fim, a solução presente no capítulo 10, focada na etapa *Affine*, pode ser comparada parcialmente com outros oito trabalhos, considerando as diferenças entre a versão do VTM e técnicas utilizadas. Ao analisar a redução de tempo de codificação da etapa *Affine*, a solução proposta apresenta resultados superiores aos de seis tra-

balhos. Quanto à redução no tempo de codificação total, apenas o trabalho de Sagrilo et al. (2023) possui resultado inferior ao da solução proposta. De maneira semelhante, em termos de BDBR, apenas os trabalhos de Hong et al. (2023) e Pejman et al. (2023) atingem valores superiores ao da solução apresentada no capítulo 10 desta tese. Mesmo assim, é possível afirmar que a solução proposta é competitiva com os trabalhos relacionados, visto que há diferenças significativas entre as versões do VTM utilizadas.

Com base nos resultados obtidos, ao retomar a questão de pesquisa desta tese (*"Como reduzir o custo computacional da predição interquadros do VVC, com impactos mínimos na eficiência de codificação?"*), é possível afirmar que a hipótese *"o uso de heurísticas baseadas em aprendizado de máquina seriam os caminhos mais promissores para responder à questão de pesquisa"* foi comprovada verdadeira, apesar de possuir fragilidades que ainda precisam ser exploradas. Diante disso, podem ser elencados alguns trabalhos futuros. É possível realizar uma busca mais detalhada por *features* que representem melhor as características para a tomada de decisão dos modelos. Outra alternativa consiste em utilizar diferentes tipos de decisão, tais como o desligamento de vários tamanhos de bloco em conjunto para determinada ferramenta, ou então, a poda na árvore de particionamento, apontando para o tamanho máximo do bloco a ser testado para determinada ferramenta. Também é possível realizar uma busca por hiperparâmetros mais relevantes para esse tipo de aplicação, conduzindo uma busca mais detalhada para os mesmos. Outra alternativa consiste na utilização de modelos de aprendizado de máquina distintos dos utilizados neste trabalho para redução do tempo total e das etapas de codificação. Além disso, a análise experimental apresentada nesta tese demonstrou que o modo *Merge*, além da Estimação de Movimento, apresenta alto tempo de codificação, podendo esta ser uma ferramenta alvo de otimização no futuro.

REFERÊNCIAS

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

ANDRADES, R. Soares de; GRELLERT, M.; FONSECA, M. B. Hyperparameter Tuning and its Effects on Cardiac Arrhythmia Prediction. In: BRAZILIAN CONFERENCE ON INTELLIGENT SYSTEMS (BRACIS), 2019., 2019. **Anais...** [S.l.: s.n.], 2019. p.562–567.

BJØNTEGAARD, G. Calculation of Average PSNR Differences between RD-curves. **ITU-T VCEG-M33**, [S.l.], 2001.

BOISBERRANGER, J. d. et al. **K-Fold cross-validator**. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html>. Acesso em: 2025-01-10.

BOISBERRANGER, J. d. et al. **Randomized search on hyper parameters**. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html>. Acesso em: 2025-01-10.

BOISBERRANGER, J. d. et al. **Exhaustive search over specified parameter values for an estimator**. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html>. Acesso em: 2025-01-10.

BOSSEN, F. et al. VTM common test conditions and software reference configurations for SDR video. , [S.l.], 10 2020.

BOUAAFIA, S.; KHEMIRI, R.; SAYADI, F. E.; ATRI, M. Fast CU partition-based machine learning approach for reducing HEVC complexity. **Journal of Real-Time Image Processing**, [S.l.], v.17, n.1, p.185–196, 2020.

BRASIL, E.-C. **Internet durante a pandemia**: 97% dos entrevistados a usam todos os dias, diz pesquisa. Disponível em:

<https://www.ecommercebrasil.com.br/noticias/brasileiros-acessam-a-internet-todos-os-dias/>. Acesso em: 2022-03-15.

BROSS, B.; CHEN, J.; OHM, J.-R.; SULLIVAN, G. J.; WANG, Y.-K. Developments in International Video Coding Standardization After AVC, With an Overview of Versatile Video Coding (VVC). **Proceedings of the IEEE**, [S.l.], p.1–31, 2021.

BROSS, B.; WANG, Y.-K.; YE, Y.; LIU, S.; CHEN, J.; SULLIVAN, G. J.; OHM, J.-R. Overview of the Versatile Video Coding (VVC) Standard and its Applications. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.31, n.10, p.3736–3764, 2021.

BROWNE, A.; YE, Y.; KIM, S. H. Algorithm description for Versatile Video Coding and Test Model 17 (VTM 17). **Jt. Video Expert. Team ITU-T SG 16 WP 3 ISO/IEC JTC 1/SC 29, 26th Meet. by teleconference**, [S.l.], April 2022.

CEBRIÁN-MÁRQUEZ, G.; MARTÍNEZ, J. L.; CUENCA, P. Inter and intra pre-analysis algorithm for HEVC. **The Journal of Supercomputing**, [S.l.], v.73, n.1, p.414–432, 2017.

CHAN, K.-H.; IM, S.-K. Faster Inter Prediction by NR-Frame in VVC. In: INTERNATIONAL CONFERENCE ON GRAPHICS AND SIGNAL PROCESSING, 2023., 2023, New York, NY, USA. **Proceedings...** Association for Computing Machinery, 2023. p.24–28. (ICGSP '23).

CHANG, Y.-J. et al. Multiple Reference Line Coding for Most Probable Modes in Intra Prediction. In: DATA COMPRESSION CONFERENCE (DCC), 2019., 2019. **Anais...** [S.l.: s.n.], 2019. p.559–559.

CHEN, Y. et al. An Overview of Core Coding Tools in the AV1 Video Codec. In: PICTURE CODING SYMPOSIUM (PCS), 2018., 2018. **Anais...** [S.l.: s.n.], 2018. p.41–45.

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

DAEDE, T.; NORKIN, A.; BRAILOVSKIY, I. **Video Codec Testing and Quality Measurement**. [S.l.]: Internet Engineering Task Force, 2019. Internet-Draft, Work in Progress. (draft-ietf-netvc-testing-08).

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

DOOLEY, J. **Americans spent over 123 billion minutes streaming video content in just one week**. Disponível em: <https://www.clickz.com/americans-spent-over-123-billion-minutes-streaming-video-content-in-just-one-week/263646/>. Acesso em: 2022-03-15.

DUARTE, A.; GONçALVES, P.; AGOSTINI, L.; ZATT, B.; CORREA, G.; PORTO, M.; PALOMINO, D. Fast Affine Motion Estimation for VVC using Machine-Learning-Based Early Search Termination. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS (ISCAS), 2022., 2022. **Anais...** [S.l.: s.n.], 2022. p.1–5.

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

FREITAS SALDANHA, M. R. de. **Exploration of Encoding Time Reduction Solutions for Intra-Frame Prediction of VVC Encoders**. 2021. Tese (Doutorado em Ciência da Computação) — Universidade Federal de Pelotas.

GÉRON, A. **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow**. 2nd.ed. Sebastopol, CA: O'Reilly Media, 2019.

GONçALVES, P. H. R. **Um esquema rápido baseado em aprendizado de máquina para a predição interquadros do codificador de vídeo VVC**. 2021. 90p. Dissertação (Mestrado em Ciência da Computação) — Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas.

GUAN, X.; SUN, X. VVC Fast ME Algorithm Based on Spatial Texture Features and Time Correlation. In: INTERNATIONAL CONFERENCE ON DIGITAL SOCIETY AND INTELLIGENT SYSTEMS (DSINS), 2021., 2021. **Anais...** [S.l.: s.n.], 2021. p.371–377.

HELLE, P.; OUDIN, S.; BROSS, B.; MARPE, D.; BICI, M. O.; UGUR, K.; JUNG, J.; CLARE, G.; WIEGAND, T. Block Merging for Quadtree-Based Partitioning in HEVC. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.22, n.12, p.1720–1731, 2012.

HONG, J.; DONG, Z.; ZHANG, X.; SONG, N.; CAO, P. A Fast Gradient Iterative Affine Motion Estimation Algorithm Based on Edge Detection for Versatile Video Coding. **Electronics**, [S.l.], v.12, n.16, 2023.

HUANG, X.; ZHOU, F.; NIU, W.; LI, T.; LU, Y.; ZHOU, Y.; YIN, H.; YAN, C. Multistage affine motion estimation fast algorithm for versatile video coding using deci-

sion tree. **Journal of Visual Communication and Image Representation**, [S.l.], v.96, p.103910, 2023.

ITU-T. **Subjective video quality assessment methods for multimedia applications**. https://www.itu.int/rec/T-REC-P.910.

JUNG, S.; JUN, D. Context-Based Inter Mode Decision Method for Fast Affine Prediction in Versatile Video Coding. **Electronics**, [S.l.], v.10, n.11, 2021.

KE, G.; MENG, Q.; FINLEY, T.; WANG, T.; CHEN, W.; MA, W.; YE, Q.; LIU, T.-Y. LightGBM: a highly efficient gradient boosting decision tree. In: INTERNATIONAL CONFERENCE ON NEURAL INFORMATION PROCESSING SYSTEMS, 31., 2017, Red Hook, NY, USA. **Proceedings...** Curran Associates Inc., 2017. p.3149–3157. (NIPS'17).

KUANG, W.; LI, X.; ZHAO, X.; LIU, S. Unified Fast Partitioning Algorithm for Intra and Inter Predictions in Versatile Video Coding. In: PICTURE CODING SYMPOSIUM (PCS), 2022., 2022. **Anais...** [S.l.: s.n.], 2022. p.271–275.

KULUPANA, G.; KUMAR M, V. P.; BLASI, S. Fast Versatile Video Coding using Specialised Decision Trees. In: PICTURE CODING SYMPOSIUM (PCS), 2021., 2021. **Anais...** [S.l.: s.n.], 2021. p.1–5.

LEE, T.; JUN, D. Fast Mode Decision Method of Multiple Weighted Bi-Predictions Using Lightweight Multilayer Perceptron in Versatile Video Coding. **Electronics**, [S.l.], v.12, n.12, 2023.

LI, J.; ZHANG, S.; YANG, F. Random Forest Accelerated CU Partition for Inter Prediction in H.266/VVC. In: IEEE INTERNATIONAL CONFERENCE ON MULTIMEDIA AND EXPO (ICME), 2022., 2022. **Anais...** [S.l.: s.n.], 2022. p.01–06.

LI, L.; WU, K.; WEI, H.; LIU, J.; FANG, Y.; ZHENG, H. Deep Motion Vector Prediction for Versatile Video Coding. In: INTERNATIONAL CONFERENCE ON COMPUTER AND COMMUNICATIONS (ICCC), 2021., 2021. **Anais...** [S.l.: s.n.], 2021. p.874–878.

LI, X.; HE, J.; LI, Q.; CHEN, X. An Adjacency Encoding Information-Based Fast Affine Motion Estimation Method for Versatile Video Coding. **Electronics**, [S.l.], v.11, n.21, 2022.

LI, Y.; LIU, Z.; JI, X.; WANG, D. CNN based CU partition mode decision algorithm for HEVC inter coding. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 2018., 2018. **Anais...** [S.l.: s.n.], 2018. p.993–997.

LI, Y.; LUO, F.; ZHU, Y. Temporal Prediction Model-Based Fast Inter CU Partition for Versatile Video Coding. **Sensors**, [S.l.], v.22, n.20, 2022.

LI, Y.; ZHAO, Z.; LIU, Q.; DU, S.; IKENAGA, T. Bottom-up check and temporal rate proportion based fast InterIMV algorithm in versatile video coding. In: INTERNATIONAL CONFERENCE ON IMAGE, VIDEO PROCESSING AND ARTIFICIAL INTELLIGENCE, 2020., 2020. **Anais...** SPIE, 2020. v.11584, p.115840W.

LINDINO, M.; ZATT, B.; GRELLERT, M.; CORREA, G. Low-Complexity Multi-Type Tree Partitioning for Versatile Video Coding Based on Machine Learning. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 2022., 2022. **Anais...** [S.l.: s.n.], 2022. p.3616–3620.

LIU, J.-K.; LIN, Y. Efficient HEVC Inter Prediction using SVM. In: IEEE 8TH GLOBAL CONFERENCE ON CONSUMER ELECTRONICS (GCCE), 2019., 2019. **Anais...** [S.l.: s.n.], 2019. p.354–358.

LIU, Y.; ABDOLI, M.; GUIONNET, T.; GUILLEMOT, C.; ROUMY, A. Light-Weight CNN-Based VVC Inter Partitioning Acceleration. In: IEEE 14TH IMAGE, VIDEO, AND MULTIDIMENSIONAL SIGNAL PROCESSING WORKSHOP (IVMSP), 2022., 2022. **Anais...** [S.l.: s.n.], 2022. p.1–5.

LOOSE, M.; VIANA, R.; SAGRILO, F.; SANCHEZ, G.; CORRêA, G.; AGOSTINI, L. A Hardware-Friendly and Configurable Heuristic Targeting VVC Inter-Frame Prediction. In: IEEE INTERNATIONAL CONFERENCE ON ELECTRONICS, CIRCUITS AND SYSTEMS (ICECS), 2022., 2022. **Anais...** [S.l.: s.n.], 2022. p.1–4.

MERCAT, A.; VIITANEN, M.; VANNE, J. UVG dataset. In: ACM MULTIMEDIA SYSTEMS CONFERENCE, 11., 2020. **Proceedings...** ACM, 2020.

NORVIG, P.; RUSSELL, S. **Inteligência Artificial**: Tradução da 3ª Edição. 3ª.ed. Rio de Janeiro, Brasil: Elsevier Editora Ltda., 2014.

PAKDAMAN, F. et al. Complexity analysis of next-generation VVC encoding and decoding. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 2020., 2020. **Anais...** [S.l.: s.n.], 2020. p.3134–3138.

PALAU, R. d. C. N. et al. Modern Video Coding: Methods, Challenges and Systems. **Journal of Integrated Circuits and Systems**, [S.l.], v.16, n.2, p.1–12, 2021.

PAN, Z.; LEI, J.; ZHANG, Y.; SUN, X.; KWONG, S. Fast Motion Estimation Based on Content Property for Low-Complexity H.265/HEVC Encoder. **IEEE Transactions on Broadcasting**, [S.l.], v.62, n.3, p.675–684, 2016.

PAN, Z.; LEI, J.; ZHANG, Y.; WANG, F. L. Adaptive fractional-pixel motion estimation skipped algorithm for efficient HEVC motion estimation. **ACM Transactions on Mul-**

timedia Computing, Communications, and Applications (TOMM), [S.l.], v.14, n.1, p.1–19, 2018.

PAN, Z.; QIN, H.; YI, X.; ZHENG, Y.; KHAN, A. Low complexity versatile video coding for traffic surveillance system. **International Journal of Sensor Networks**, [S.l.], v.30, n.2, p.116–125, 2019.

PAN, Z.; ZHANG, P.; PENG, B.; LING, N.; LEI, J. A CNN-Based Fast Inter Coding Method for VVC. **IEEE Signal Processing Letters**, [S.l.], v.28, p.1260–1264, 2021.

PARK, S.-H.; KANG, J.-W. Fast Affine Motion Estimation for Versatile Video Coding (VVC) Encoding. **IEEE Access**, [S.l.], v.7, p.158075–158084, 2019.

PEDREGOSA, F. et al. Scikit-learn: Machine Learning in Python. **Journal of Machine Learning Research**, [S.l.], v.12, p.2825–2830, 2011.

PEJMAN, H.; COULOMBE, S.; VAZQUEZ, C.; JAMALI, M.; VAKILI, A. An Adjustable Fast Decision Method for Affine Motion Estimation in VVC. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 2023., 2023. **Anais...** [S.l.: s.n.], 2023. p.2695–2699.

PENG, Z.; SHEN, L. A classification-prediction joint framework to accelerate QTMT-based CU partition of inter-mode VVC. **Electronics Letters**, [S.l.], v.59, n.7, p.e12770, 2023.

PETERSEN, K.; FELDT, R.; MUJTABA, S.; MATTSSON, M. Systematic Mapping Studies in Software Engineering. In: INTERNATIONAL CONFERENCE ON EVALUATION AND ASSESSMENT IN SOFTWARE ENGINEERING, 12., 2008, Swindon, UK. **Proceedings...** BCS Learning & Development Ltd., 2008. p.68–77. (EASE'08).

RASCHKA, S. **Python Machine Learning**. 1st.ed. Birmingham, UK: Packt Publishing, 2015.

REN, W.; HE, W.; CUI, Y. An Improved Fast Affine Motion Estimation Based on Edge Detection Algorithm for VVC. **Symmetry**, [S.l.], v.12, n.7, 2020.

SAGRILO, F.; LOOSE, M.; VIANA, R.; SANCHEZ, G.; CORRêA, G.; AGOSTINI, L. Learning-Based Fast VVC Affine Motion Estimation. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS (ISCAS), 2023., 2023. **Anais...** [S.l.: s.n.], 2023. p.1–5.

SALDANHA, M.; CORRÊA, M.; CORRÊA, G.; PALOMINO, D.; PORTO, M.; ZATT, B.; AGOSTINI, L. An Overview of Dedicated Hardware Designs for State-of-the-Art AV1

and H. 266/VVC Video Codecs. In: IEEE INTERNATIONAL CONFERENCE ON ELECTRONICS, CIRCUITS AND SYSTEMS (ICECS), 2020., 2020. **Anais...** [S.l.: s.n.], 2020. p.1–4.

SAURTY, K.; CATHERINE, P. C.; SOYJAUDAH, K. M. Inter prediction complexity reduction for HEVC based on residuals characteristics. **International Journal of Advanced Computer Science and Applications**, [S.l.], v.7, n.10, p.1649–1668, 2016.

SCHOBER, P.; BOER, C.; SCHWARTE, L. A. Correlation Coefficients: Appropriate Use and Interpretation. **Anesthesia & Analgesia**, [S.l.], v.126, n.5, 2018.

SCHäFER, M. et al. An Affine-Linear Intra Prediction With Complexity Constraints. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 2019., 2019. **Anais...** [S.l.: s.n.], 2019. p.1089–1093.

SHANG, X.; LI, G.; ZHAO, X.; ZUO, Y. Low complexity inter coding scheme for Versatile Video Coding (VVC). **Journal of Visual Communication and Image Representation**, [S.l.], v.90, p.103683, 2023.

SHEN, L.; YANG, H.; WANG, S. Effective QTMT Partition Decision Algorithm for VVC Intercoding. In: IEEE 24TH INTERNATIONAL WORKSHOP ON MULTIMEDIA SIGNAL PROCESSING (MMSP), 2022., 2022. **Anais...** [S.l.: s.n.], 2022. p.1–6.

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

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

SULLIVAN, G.; WIEGAND, T. Video Compression - From Concepts to the H.264/AVC Standard. **Proceedings of the IEEE**, [S.l.], v.93, n.1, p.18–31, 2005.

TANG, N.; CAO, J.; LIANG, F.; WANG, J.; LIU, H.; WANG, X.; DU, X. Fast CTU Partition Decision Algorithm for VVC Intra and Inter Coding. In: IEEE ASIA PACIFIC CONFERENCE ON CIRCUITS AND SYSTEMS (APCCAS), 2019., 2019. **Anais...** [S.l.: s.n.], 2019. p.361–364.

THANH, H. B.; QUANG, S. N.; HUU, T. V.; HOANGVAN, X. Learning adaptive motion search for fast versatile video coding in visual surveillance systems. **IET Image Processing**, [S.l.], v.n/a, n.n/a, 2023.

TISSIER, A.; HAMIDOUCHE, W.; VANNE, J.; MENARD, D. Machine Learning Based Efficient Qt-Mtt Partitioning for VVC Inter Coding. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 2022., 2022. **Anais...** [S.l.: s.n.], 2022. p.1401–1405.

WANG, J.; YANG, J. H. 266/VVC. **Journal of Innovation and Social Science Research ISSN**, [S.l.], v.9, p.193–200, 2022.

XIE, K.; ZHOU, J.; ZHANG, S.; YANG, F. Fast Inter Prediction Mode Decision Method Based On Random Forest For H.266/VVC. In: IEEE INTERNATIONAL CONFERENCE ON VISUAL COMMUNICATIONS AND IMAGE PROCESSING (VCIP), 2022., 2022. **Anais...** [S.l.: s.n.], 2022. p.1–5.

XU, J.; HE, S. Optimization of Inter-Frame Coding Algorithm Based on Random Forest. In: ASIAN CONFERENCE ON ARTIFICIAL INTELLIGENCE TECHNOLOGY (ACAIT), 2022., 2022. **Anais...** [S.l.: s.n.], 2022. p.1–4.

YANG, X.; LI, W.; CHEN, S.; HUANG, L.; FAN, Y. A Dynamic-Texture-Guided Fast Algorithm for Geometric Partitioning Mode of VVC. In: IEEE 15TH INTERNATIONAL CONFERENCE ON ASIC (ASICON), 2023., 2023. **Anais...** [S.l.: s.n.], 2023. p.1–4.

YEO, W.-H.; KIM, B.-G. CNN-based Fast Split Mode Decision Algorithm for Versatile Video Coding (VVC) Inter Prediction. **Journal of Multimedia Information System**, [S.l.], v.8, n.3, p.147–158, 2021.

YU, X.; YANG, F. The Optimization of TZSearch for VVC Motion Estimation. In: INTERNATIONAL CONFERENCE ON UBIQUITOUS COMMUNICATION (UCOM), 2023., 2023. **Anais...** [S.l.: s.n.], 2023. p.156–159.

ZEIGERMAN, I. **Model 2 Code Generator**. Acessado em 15 de julho de 2024. Disponível em: <https://github.com/BayesWitnesses/m2cgen>.

ZHANG, T.; MAO, S. An Overview of Emerging Video Coding Standards. **GetMobile: Mobile Comp. and Comm.**, New York, NY, USA, v.22, n.4, p.13–20, May 2019.

Apêndices

APÊNDICE A – Balanceamento - Solução Bidirecional

Legenda das Tabelas:

valor Número de exemplos menor que 1000

valor Critério $i)$

valor Critério $ii)$

Res.	Vídeo	QP	128x128 (0)	128x64 (0)	128x64 (1)	64x128 (0)	64x128 (1)	4x64 (0)	4x64 (1)
HD	*Dark*	22	1051	695	1000	753	1000	1000	1000
		27	1071	1000	1000	1000	1000	1000	1000
		32	936	1000	1000	1000	1000	1077	1063
		37	737	1000	1000	1000	1000	1312	1322
	Netflix Driving POV	22	1051	1077	1000	1062	1000	1000	1000
		27	1071	1000	1000	1000	1000	1000	1000
		32	1122	1000	1000	1000	1000	1077	1063
		37	1066	1000	1000	1000	1000	1312	1322
	Vidyo4	22	1051	1077	1000	1062	1000	1000	1000
		27	1071	1000	1000	1000	1000	1000	1000
		32	1122	1000	1000	1000	1000	1077	1063
		37	1066	1000	1000	1000	1000	879	887
	Netflix DinnerScene	22	796	1077	1000	1062	1000	1000	1000
		27	726	1000	1000	1000	1000	1000	1000
		32	839	1000	1000	1000	1000	693	749
		37	1066	1000	1000	1000	1000	185	148
	Kristen And Sara	22	1051	1077	1000	1062	1000	1000	1000
		27	1061	1000	1000	1000	1000	1000	1000
		32	982	1000	1000	1000	1000	1077	1063
		37	1066	1000	1000	1000	1000	1312	1322
FHD	*Netflix Tunnel Flags*	22	1000	1284	1213	1307	1214	1000	1000
		27	1000	1219	1000	1216	1000	1000	1000
		32	1000	1000	1000	1000	1000	1000	1000
		37	1000	1000	1000	1000	1000	1000	1000
	Jockey	22	1000	2572	1213	2668	1214	1000	1000
		27	1000	1219	1000	1216	1000	1000	1000
		32	1000	1000	1000	1000	1000	1000	1000
		37	1000	1000	1000	1000	1000	1000	1000
	Beauty	22	1000	547	1130	601	1113	1000	1000
		27	1000	1219	1000	1216	1000	1000	1000
		32	1000	1000	1000	1000	1000	1000	1000
		37	1000	1000	1000	1000	1000	1000	1000
	Touchdown Pass	22	1000	558	1213	396	1214	1000	1000
		27	1000	1219	1000	1216	1000	1000	1000
		32	1000	1000	1000	1000	1000	1000	1000
		37	1000	1000	1000	1000	1000	1000	1000
	Rush Field Cuts	22	1000	39	233	28	246	1000	1000
		27	1000	127	1000	138	1000	1000	1000
		32	1000	1000	1000	1000	1000	1000	1000
		37	1000	1000	1000	1000	1000	1000	1000

Res.	Vídeo	QP	128x128 (0)	128x64 (0)	128x64 (1)	64x128 (0)	64x128 (1)	4x64 (0)	4x64 (1)
4K UHD	*Toddler Fountain*	22	1000	240	729	189	732	1000	1000
		27	1000	310	786	314	774	1000	1000
		32	1000	357	1000	392	1000	1000	1000
		37	1000	848	1000	816	1000	1000	1063
	SunBath	22	1000	1068	1068	1070	1068	1000	1000
		27	1000	1173	1054	1172	1057	1000	1000
		32	1000	1161	1000	1152	1000	1000	1000
		37	1000	1038	1000	1046	1000	1000	1063
	Lips	22	1000	1231	1068	1247	1068	1000	1000
		27	1000	1173	1054	1172	1057	1000	1000
		32	1000	1161	1000	1152	1000	1000	1000
		37	1000	1038	1000	1046	1000	1000	1063
	Building Hall2	22	1000	1231	1068	1247	1068	1000	1000
		27	1000	1173	1054	1172	1057	1000	1000
		32	1000	1161	1000	1152	1000	1000	1000
		37	1000	1038	1000	1046	1000	1000	1063
	Netflix Dancers	22	1000	1231	1068	1247	1068	1000	1000
		27	1000	1173	1054	1172	1057	1000	1000
		32	1000	1161	1000	1152	1000	1000	1000
		37	1000	1038	1000	1046	1000	1000	748

Res.	Vídeo	QP	64x4 (0)	64x4 (1)	8x16 (0)	8x16 (1)	16x8 (0)	16x8 (1)	4x32 (0)	4x32 (1)
HD	*Dark*	22	1000	1000	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000	1091
		32	1185	1171	1000	1164	1000	1176	1062	467
		37	1562	1256	1206	234	1224	248	699	45
	Netflix Driving POV	22	1000	1000	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000	1091
		32	1185	1171	1000	1164	1000	1176	1062	2838
		37	1562	1524	1206	3779	1224	3692	2245	2523
	Vidyo4	22	1000	1000	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000	1091
		32	1185	1171	1000	1164	1000	1176	1062	638
		37	1124	1127	1206	321	1224	390	659	98
	Netflix DinnerScene	22	1000	1000	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000	637
		32	263	318	1000	346	1000	296	752	127
		37	75	60	176	24	104	6	99	3
	Kristen And Sara	22	1000	1000	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000	1091
		32	1185	1171	1000	1164	1000	1176	1062	930
		37	677	1033	1206	642	1224	664	1299	191
FHD	*Netflix Tunnel Flags*	22	1000	1000	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000	1000
		32	1000	1000	1000	1000	1000	1000	1000	1000
		37	1000	1000	1000	1191	1000	1172	1000	1861
	Jockey	22	1000	1000	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000	1000
		32	1000	1000	1000	1000	1000	1000	1000	1000
		37	1000	1000	1000	1191	1000	1172	1000	762
	Beauty	22	1000	1000	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000	1000
		32	1000	1000	1000	1000	1000	1000	1000	1000
		37	1000	1000	1000	237	1000	314	1000	111
	Touchdown Pass	22	1000	1000	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000	1000
		32	1000	1000	1000	1000	1000	1000	1000	1000
		37	1000	1000	1000	1191	1000	1172	1000	1861
	Rush Field Cuts	22	1000	1000	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000	1000
		32	1000	1000	1000	1000	1000	1000	1000	1000
		37	1000	1000	1000	1191	1000	1172	1000	1861

Res.	Vídeo	QP	64x4 (0)	64x4 (1)	8x16 (0)	8x16 (1)	16x8 (0)	16x8 (1)	4x32 (0)	4x32 (1)
4K UHD	*Toddler Fountain*	22	1000	1000	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000	1000
		32	1000	1000	1000	1113	1000	1130	1000	1212
		37	1074	1143	1094	1532	1105	1550	1223	1921
	SunBath	22	1000	1000	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000	1000
		32	1000	1000	1000	1113	1000	1130	1000	1212
		37	1074	1143	1094	1532	1105	1550	1223	1741
	Lips	22	1000	1000	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000	1000
		32	1000	1000	1000	1113	1000	1130	1000	1175
		37	1074	1143	1094	384	1105	330	1133	100
	Building Hall2	22	1000	1000	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000	1000
		32	1000	1000	1000	1113	1000	1130	1000	1212
		37	1074	1143	1094	1532	1105	1550	1223	1921
	Netflix Dancers	22	1000	1000	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000	1000
		32	1000	1000	1000	549	1000	482	1000	191
		37	707	431	626	21	582	22	198	4

Res.	Vídeo	QP	32x4 (0)	32x4 (1)	8x8 (0)	8x8 (1)	4x16 (0)	4x16 (1)	16x4 (0)	16x4 (1)
HD	*Dark*	22	1000	1000	1000	1000	1000	1000	1000	1000
		27	1000	1123	1000	1066	1000	1116	1000	1151
		32	1077	747	1046	811	1079	497	1139	583
		37	869	91	830	64	580	26	683	48
	Netflix Driving POV	22	1000	1000	1000	1000	1000	1000	1000	1000
		27	1000	1123	1000	1066	1000	1116	1000	1151
		32	1077	2210	1046	1738	1079	2899	1139	2900
		37	1706	2294	1575	3123	2042	2172	2437	1555
	Vidyo4	22	1000	1000	1000	1000	1000	1000	1000	1000
		27	1000	1123	1000	1066	1000	1116	1000	1151
		32	1077	959	1046	1024	1079	599	1139	641
		37	1402	133	981	62	773	44	801	30
	Netflix DinnerScene	22	1000	1000	1000	1000	1000	1000	1000	1000
		27	1000	511	1000	738	1000	536	1000	396
		32	695	76	817	62	684	41	446	17
		37	63	0	40	1	25	0	6	0
	Kristen And Sara	22	1000	1000	1000	1000	1000	1000	1000	1000
		27	1000	1123	1000	1066	1000	1116	1000	1151
		32	1077	1008	1046	1365	1079	964	1139	859
		37	960	183	1575	234	1580	143	1073	135
FHD	*Netflix Tunnel Flags*	22	1000	1000	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000	1000
		32	1000	1000	1000	1000	1000	1000	1000	1000
		37	1000	1903	1134	2050	1210	2340	1202	2533
	Jockey	22	1000	1000	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000	1000
		32	1000	1000	1000	1000	1000	1000	1000	1000
		37	1000	643	1134	351	1210	302	1202	232
	Beauty	22	1000	1000	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000	1000
		32	1000	1000	1000	1000	1000	1000	1000	1000
		37	1000	103	467	15	161	9	193	9
	Touchdown Pass	22	1000	1000	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000	1000
		32	1000	1000	1000	1000	1000	1000	1000	1000
		37	1000	1903	1134	2050	1210	2340	1202	2533
	Rush Field Cuts	22	1000	1000	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000	1000
		32	1000	1000	1000	1000	1000	1000	1000	1000
		37	1000	1903	1134	2050	1210	2340	1202	2533

Res.	Vídeo	QP	32x4 (0)	32x4 (1)	8x8 (0)	8x8 (1)	4x16 (0)	4x16 (1)	16x4 (0)	16x4 (1)
4K UHD	*Toddler Fountain*	22	1000	1000	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000	1000
		32	1000	1249	1000	1382	1000	1439	1052	1491
		37	1280	2271	1314	2078	1383	2340	1475	2533
	SunBath	22	1000	1000	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000	1000
		32	1000	1249	1000	1382	1000	1439	1052	1491
		37	1280	1168	1314	781	1383	548	1475	296
	Lips	22	1000	1000	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000	1000
		32	1000	1072	1000	787	1000	625	1052	508
		37	951	131	954	60	828	56	563	34
	Building Hall2	22	1000	1000	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000	1000
		32	1000	1249	1000	1382	1000	1439	1052	1491
		37	1280	2271	1314	2078	1383	2340	1475	2533
	Netflix Dancers	22	1000	1000	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000	1000
		32	1000	183	1000	68	1000	60	795	21
		37	209	7	105	3	24	1	12	0

Res.	Vídeo	QP	64x8 (0)	64x8 (1)	8x64 (0)	8x64 (1)	16x64 (0)	16x64 (1)	16x32 (1)
HD	*Dark*	22	1000	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000
		32	1074	1000	1000	1000	1000	1000	1000
		37	1179	1185	1162	1150	1093	1044	1024
	Netflix Driving POV	22	1000	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000
		32	1074	1000	1000	1000	1000	1000	1000
		37	1179	1185	1162	1150	1093	1044	1024
	Vidyo4	22	1000	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000
		32	1074	1000	1000	1000	1000	1000	1000
		37	1179	1185	1162	1150	1093	1044	1024
	Netflix DinnerScene	22	1000	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000
		32	706	1000	1000	1000	1000	1000	1000
		37	286	261	355	402	631	827	904
	Kristen And Sara	22	1000	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000
		32	1074	1000	1000	1000	1000	1000	1000
		37	1179	1185	1162	1150	1093	1044	1024

Res.	Vídeo	QP	32x 16 (1)	16x 16 (0)	16x 16 (1)	8x 32 (0)	8x 32 (1)	32x 8 (0)	32x 8 (1)
HD	*Dark*	22	1000	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000
		32	1000	1000	1000	1000	1000	1000	1000
		37	1067	1087	1137	1121	1159	1155	1182
	Netflix Driving POV	22	1000	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000
		32	1000	1000	1000	1000	1000	1000	1000
		37	1067	1087	1137	1121	1159	1155	1182
	Vidyo4	22	1000	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000
		32	1000	1000	1000	1000	1000	1000	1000
		37	1067	1087	1137	1121	1159	1155	1182
	Netflix DinnerScene	22	1000	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000
		32	1000	1000	1000	1000	1000	1000	1000
		37	734	652	452	517	367	383	275
	Kristen And Sara	22	1000	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000
		32	1000	1000	1000	1000	1000	1000	1000
		37	1067	1087	1137	1121	1159	1155	1182

APÊNDICE B – Balanceamento - Solução Ampliada Bidirecional e *Affine*

Legenda da Tabela:

valor Número de exemplos menor que 1000

valor Critério $i)$

valor Critério $ii)$

Res.	Vídeo	QP	128x128 (1)	128x64 (1)	64x128 (1)	64x16 (1)	16x64 (1)	32x16 (1)	16x16 (1)
HD	*Dark*	22	1000	816	782	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000
		32	1000	1000	1000	1000	1000	1000	1000
		37	719	702	826	828	883	1001	422
	Netflix Driving POV	22	1000	1092	1109	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000
		32	1000	1000	1000	1000	1000	1000	1000
		37	1492	1537	1352	1401	1406	1001	2198
	Vidyo4	22	1000	1092	1109	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000
		32	1000	1000	1000	1000	1000	1000	1000
		37	656	761	822	771	711	998	380
FHD	*Netflix Tunnel Flags*	22	1000	1044	1111	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000
		32	1000	1000	1000	1000	1000	1000	1000
		37	1045	1000	1000	1000	1000	1000	1000
	Jockey	22	1000	1044	1111	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000
		32	1000	1000	1000	1000	1000	1000	1000
		37	1045	1000	1000	1000	1000	1000	1000
	Touchdown Pass	22	1000	912	779	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000	1000
		32	1000	1000	1000	1000	1000	1000	1000
		37	1045	1000	1000	1000	1000	1000	1000
4K UHD	*Toddler Fountain*	22	1000	673	692	1000	1000	1000	1000
		27	1000	858	807	1000	1000	1000	1000
		32	1000	1000	1000	1000	1000	1000	1000
		37	1060	1000	1000	1000	1000	1000	1189
	SunBath	22	1000	1164	1154	1000	1000	1000	1000
		27	1000	1071	1097	1000	1000	1000	1000
		32	1000	1000	1000	1000	1000	1000	1000
		37	1060	1000	1000	1000	1000	1000	1189
	Lips	22	1000	1164	1154	1000	1000	1000	1000
		27	1000	1071	1097	1000	1000	1000	1000
		32	1000	1000	1000	1000	1000	1000	1000
		37	881	1000	1000	1000	1000	1000	622

APÊNDICE C – Balanceamento - Solução *Affine*

Legenda das Tabelas:

valor Número de exemplos menor que 1000

valor Critério $i)$

valor Critério $ii)$

Res.	Vídeo	QP	128x128 (1)	128x64 (1)	64x128 (1)	16x16 (1)
HD	*Dark*	22	1000	440	512	1000
		27	1064	1000	1025	1000
		32	543	576	722	1000
		37	413	566	668	964
	Netflix Driving POV	22	1000	1099	1116	1000
		27	1064	1000	1025	1000
		32	1709	1647	1483	1000
		37	1188	1532	1569	1683
	Vidyo4	22	1000	1114	930	1000
		27	873	1000	950	1000
		32	673	777	795	1000
		37	371	490	518	353
FHD	*Netflix Tunnel Flags*	22	1000	1158	1189	1000
		27	1000	1000	1000	1000
		32	1025	1000	1000	1000
		37	1343	1138	1082	1000
	Jockey	22	1000	1278	1474	1000
		27	1000	1000	1000	1000
		32	1025	1000	1000	1000
		37	1343	1138	1082	1000
	Touchdown Pass	22	1000	912	779	1000
		27	1000	1000	1000	1000
		32	1025	1000	1000	1000
		37	1343	1138	1082	1000
4K UHD	*Toddler Fountain*	22	1000	684	669	1000
		27	1000	892	830	1000
		32	1000	1000	1000	1000
		37	1024	1000	1000	1151
	SunBath	22	1000	1158	1166	1000
		27	1000	1054	1085	1000
		32	1000	1000	1000	1000
		37	1024	1000	1000	1151
	Lips	22	1000	1158	1166	1000
		27	1000	1054	1085	1000
		32	1000	1000	1000	1000
		37	952	1000	1000	698

Res.	Vídeo	QP	64x 64 (1)	64x 16 (1)	16x 64 (1)	32x 32 (1)	16x 32 (1)	32x 16 (1)
HD	*Dark*	22	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000
		32	1000	1000	1000	1000	1000	1000
		37	1039	1160	1156	1041	1010	1100
	Netflix Driving POV	22	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000
		32	1000	1000	1000	1000	1000	1000
		37	1039	1160	1156	1041	1010	1100
	Vidyo4	22	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000
		32	1000	1000	1000	1000	1000	1000
		37	923	680	688	919	981	800
FHD	*Netflix Tunnel Flags*	22	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000
		32	1000	1000	1000	1000	1000	1000
		37	1000	1000	1000	1000	1000	1000
	Jockey	22	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000
		32	1000	1000	1000	1000	1000	1000
		37	1000	1000	1000	1000	1000	1000
	Touchdown Pass	22	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000
		32	1000	1000	1000	1000	1000	1000
		37	1000	1000	1000	1000	1000	1000
4K UHD	*Toddler Fountain*	22	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000
		32	1000	1000	1000	1000	1000	1000
		37	1000	1000	1000	1000	1000	1000
	SunBath	22	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000
		32	1000	1000	1000	1000	1000	1000
		37	1000	1000	1000	1000	1000	1000
	Lips	22	1000	1000	1000	1000	1000	1000
		27	1000	1000	1000	1000	1000	1000
		32	1000	1000	1000	1000	1000	1000
		37	1000	1000	1000	1000	1000	1000

APÊNDICE D – Lista das principais publicações durante o Doutorado

1. Marta Loose, Ramiro Viana, Fernando Sagrilo, Gustavo Sanchez, Guilherme Corrêa and Luciano Agostini, “A Hardware-Friendly and Configurable Heuristic Targeting VVC Inter-Frame Prediction”, 2022 29th IEEE International Conference on Electronics, Circuits and Systems (ICECS), Glasgow, United Kingdom, 2022, pp. 1-4.

2. Marta Loose, Ramiro Viana, Fernando Sagrilo, Gustavo Sanchez, Guilherme Corrêa and Luciano Agostini. “CAHUBA: A Configurable and Adaptive Heuristic Targeting Computational Effort Reduction of VVC Inter-frame Prediction”. In: 37th South Brazil Microelectronics School (EMICRO 2022), 2022.

3. Marta Loose, Ramiro Viana, Fernando Sagrilo, Gustavo Sanchez, Guilherme Corrêa and Luciano Agostini. “A Hardware Friendly and Configurable Heuristic Targeting VVC Inter Frame Prediction”. In: IEEE CASS/SPS Seasonal School On Digital Processing of Visual Signals And Applications, 2022. *Best poster

4. Fernando Sagrilo, Marta Loose, Ramiro Viana, Gustavo Sanchez, Guilherme Corrêa and Luciano Agostini, “Learning-Based Fast VVC Affine Motion Estimation”, 2023 IEEE International Symposium on Circuits and Systems (ISCAS), Monterey, CA, USA, 2023, pp. 1-5,

5. Ramiro Viana, Marta Loose, Rafael Ferreira, Marcelo Porto, Guilherme Corrêa and Luciano Agostini. “A Hardware-Friendly Acceleration of VVC Affine Motion Estimation Using Decision Trees”. In: 2024 37th SBC/SBMicro/IEEE Symposium on Integrated Circuits and Systems Design (SBCCI), 2024, João Pessoa. 2024 37th SBC/SBMicro/IEEE Symposium on Integrated Circuits and Systems Design (SBCCI), 2024. p. 1-5.

6. Ramiro Viana, Fernando Sagrilo, Rafael Ferreira, Marta Loose, Marcelo Porto, Guilherme Corrêa and Luciano Agostini. “A Hardware-Friendly Fast VVC Test Zone Search Algorithm Using Machine Learning”. In: 2024 IEEE 15th Latin America Symposium on Circuits and Systems (LASCAS), 2024, Punta del Este. 2024 IEEE 15th Latin America Symposium on Circuits and Systems (LASCAS), 2024. p. 1-5.

7. Ramiro Viana, Marta Loose, Ruhan Conceição, Marcelo Porto, Guilherme Corrêa and Luciano Agostini. “Fast VVC Test Zone Search and Affine Motion Estimation Using Machine Learning”. Artigo aceito para publicação em 25/10/2024. In: 2025 IEEE 16th Latin America Symposium on Circuits and Systems (LASCAS), 2025, Bento Gonçalves. 2025 IEEE 16th Latin America Symposium on Circuits and Systems (LASCAS), 2025. p. 1-5.

8. Marta Loose, Ramiro Viana, Bianca Coelho, Marcelo Porto, Guilherme Corrêa and Luciano Agostini. “Computational Effort Reduction Strategies for VVC Inter-Frame Prediction: A Literature Review”. In: 31th IBERCHIP, 2025, Bento Gonçalves. 31th IBERCHIP Proceedings, 2025. p. 28-32.