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

Dissertação

Decisão Rápida na Predição Intra-Quadro do Codificador AV1 Usando Aprendizado de Máquina

Pablo De Chiaro Rosa

Pelotas, 2022

Pablo De Chiaro Rosa

Decisão Rápida na Predição Intra-Quadro do Codificador AV1 Usando Aprendizado de Máquina

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

Orientador: Prof. Dr. Luciano Volcan Agostini
Coorientadores: Prof. Dr. Marcelo Schiavon Porto
Prof. Dr. Daniel Munari Palomino

Pelotas, 2022

Universidade Federal de Pelotas / Sistema de Bibliotecas
Catalogação na Publicação

R788d Rosa, Pablo de Chiaro

Decisão rápida na predição intra-quadro do codificador AV1 usando aprendizado de máquina / Pablo de Chiaro Rosa ; Luciano Volcan Agostini, orientador ; Marcelo Schiavon Porto, Daniel Munari Palomino, coorientadores. — Pelotas, 2022.

109 f. : il.

Dissertação (Mestrado) — Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, 2022.

1. AV1. 2. Predição intra-quadro. 3. Aprendizado de máquina. 4. Decisão rápida de modo. I. Agostini, Luciano Volcan, orient. II. Porto, Marcelo Schiavon, coorient. III. Palomino, Daniel Munari, coorient. IV. Título.

CDD : 005

Elaborada por Aline Herbstrith Batista CRB: 10/1737

AGRADECIMENTOS

Quero agradecer primeiramente, aos meus pais, Lusardo e Vera, pelas oportunidades que me proporcionaram ao longo de toda a vida, de estudar, trabalhar, explorar o mundo, os princípios que me ensinaram e moldam minhas atitudes até hoje e, apoio incondicional.

Ao meu Pai, por desde cedo, me incentivar o contato com a tecnologia, me proporcionando um computador próprio em tempos que isto era muito difícil e, liberdade de explora-lo, mesmo que por vezes isto resultasse em formata-lo. Mas acima de tudo, ser minha referência de homem, pai e, profissional.

À minha mãe, sendo incansável, a maior apoiadora em tudo que eu me interessava, me fez descobrir o interesse por blocos de montar que, sem dúvidas, desenvolveram meu pensamento lógico e, despertou junto a tecnologia, minha paixão pela informática. E também, me ensinar o que é um amor incondicional.

À minha companheira de todas as horas, Corintha Dias, pelo apoio incessante ao longo de toda está jornada do mestrado, que foi fundamental para que eu chegasse até este momento. Por ser meu suporte sempre que eu caísse, mesmo em meus momentos difíceis de suportar e, ainda assim, ao meu lado dando o melhor de si. Tornando fácil trilhar este caminho. Obrigado meu amor e, desculpa todo momento que estive distante, mesmo perto. O seu, logo mais chegará, quero estar junto a ti também.

Aos professores do PPGC, pela sua dedicação e excelência durante as aulas, em especial, aos meus orientadores, Luciano Agostini, Marcelo Porto e Daniel Palomino pela paciência e disposição durante o desenvolvimento deste trabalho.

Aos colegas do PPGC, por sempre estarem abertos a compartilhar seus conhecimentos, mesmo em tempos difíceis que todos passamos nestes últimos anos, em especial, aos colegas Alex Borges e Marcel Corrêa.

À todos meus amigos, que durante todo este processo de aprendizado do mestrado foram minha válvula de escape, sempre disponíveis e pacientes, prontos para descontrair, uns entre churrascos, jogatinas de canastra, cervejas e drinks; outros em longas partidas online com vitórias ou, derrotas com muitas risadas.

E, finalmente, aos meus cachorros Luke e Apollo, que indiretamente foram meus parceiros de escrita inseparáveis.

42

— DEEP THOUGHT

RESUMO

ROSA, Pablo De Chiaro. **Decisão Rápida na Predição Intra-Quadro do Codificador AV1 Usando Aprendizado de Máquina**. Orientador: Luciano Volcan Agostini. 2022. 109 f. Dissertação (Mestrado em Ciência da Computação) – Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2022.

A reprodução e transmissão de vídeos digitais de alta definição tem crescido em todo o mundo. Estes vídeos são usados tanto para fins de entretenimento, quanto profissionais e têm na internet a sua principal fonte de propagação. Considerando o volume de informações necessárias para representar um vídeo, o espaço massivo para armazenar as mídias digitais e o enorme tráfego gerado para sua transmissão, se torna imprescindível o uso de técnicas eficientes de compressão de vídeo. Assim, os codificadores de vídeo têm como foco eliminar redundâncias nos dados e reduzir informações que são pouco relevantes para o sistema visual humano. A utilização de codificadores de vídeo padronizados por órgãos de padronização internacionais envolvem, no geral, um custo elevado devido às despesas de direitos de uso. Na tentativa de evitar estes custos, foi fundado o AOMedia, um consórcio do setor da tecnologia que lançou, em março de 2018, a especificação do seu codificador livre destes custos, denominado AOMedia Video 1 (AV1). Contudo, o codificador AV1 alcança elevada eficiência de codificação mas com um aumento significativo no custo computacional, que reflete no tempo despendido para execução da codificação e no consumo de energia relacionado. Este trabalho apresenta uma solução para decisão rápida de modos no processo de predição intra-quadro do codificador AV1, reduzindo o custo computacional envolvido na tomada de decisão do modo preditor através do uso de aprendizado de máquina supervisionado. Modelos baseados em floresta aleatória foram treinados sobre bases de dados criadas a partir dos experimentos realizados com o software de referência do AV1, o libaom. O principal objetivo foi reduzir os modos avaliados na predição intra-quadro, acelerando seu processo de decisão. Com isso foi possível implementar os modelos treinados no codificador e avaliar seus impactos em termos de redução de tempo de execução e de perdas em eficiência de codificação. Como principais resultados, foi possível obter médias de redução de tempo de 44%, com um impacto de 4,6% no BD-Rate.

Palavras-chave: av1. predição intra-quadro. aprendizado de máquina. decisão rápida de modo.

ABSTRACT

ROSA, Pablo De Chiaro. **Fast-Decision in AV1 Encoder Intra-Frame Prediction Applying Machine Learning**. Advisor: Luciano Volcan Agostini. 2022. 109 f. Dissertation (Masters in Computer Science) – Technology Development Center, Federal University of Pelotas, Pelotas, 2022.

The playback and transmission of high-definition digital videos grows exponentially worldwide. These videos are used for both entertainment and professional purposes and have the internet as their main source of propagation. Considering the volume of information needed to represent a video, the massive space to store digital media and the huge data traffic generated, it is essential to use efficient video compression techniques. In this way, video encoders focus on eliminating data redundancies and reducing information that is of little relevance to the human visual system. The use of video encoders standarized by international standardization organizations, generally involve a high cost due to the expenses of royalties applied. In an attempt to avoid these costs, Aomedia was founded. Aomedia is a technology industry consortium that launches in March 2018 the specification of its royalty-free encoder, called Aomedia Video 1 (AV1). However, this developed format achieves high values of coding efficiency at the same time as it increases the computational costs that reflects the time taken to enconding and the related energy consumption. This work presents a solution for the fast mode decision in the intra-frame prediction process of the AV1 encoder, reducing the computational cost involved in decision-making of the predictor mode, using supervised machine learning. Random Forest-based models were trained on datasets created from the experiments performed in the AV1 reference software, libaom. The main goal was to reduce the modes evaluated in intra prediction, accelarating the decision process. Then, it was possible to implemented the models at the encoder and evaluate the impacts in terms of encoding time reduction and encoding efficiency losses. As a result, it was possible to achieve averages of time savings of 44% with and impact of 4.6% on BD-Rate.

Keywords: av1. intra-frame prediction. machine learning. fast mode decision.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

2D	Duas Dimensões
3D	Três Dimensões
AI	All Intra
AIC	Aprendizado Indutivo de Conceitos
AOMedia	Alliance for Open Media
AV1	AOMedia Video 1
BD-rate	Bjøntegaard Delta Rate
BL	Below Left
Cb	Chrominance Blue
CfL	Chroma from Luma
CQ	Constant Quality
Cr	Chrominance Red
CTC	Common Test Conditions
DT	Decision Tree - Árvore de Decisão
GB	Giga Bytes
HD	High Definition
HEVC	High Efficiency Video Coding
HVS	Human Vision System
IETF	Internet Engineering Task Force
intraBC	Intra Block Copy
Kbps	Kilobit per Second
MB	Macroblock
MV	Motion Vector
NETVC	Internet Video Codec
PSNR	Peak Signal-to-Noise Ratio
RD	Rate-Distortion

RF	Random Forest - Floresta Aleatória
RT	Redução de Tempo
RMSE	Root Mean Squared Error
ROC	Receiver Operating Characteristic
SB	SuperBlock
SC	Screen Content Video
TR	Top Right
TS	Time Saving
VCEG	Video Coding Experts Group
VVC	Versatile Video Coding
Y	Luminance
YCbCr	Luminance, Chrominance Blue, Chrominance Red

SUMÁRIO

1 INTRODUÇÃO

A captura, reprodução e transmissão de vídeos digitais de alta definição tem crescido em todo o mundo. Estes vídeos são usados tanto para fins de entretenimento, quanto profissionais e têm na internet sua principal fonte de propagação, com uso tanto nos computadores pessoais, quanto nos dispositivos móveis (e.g. *smartphones* e *tablets*).

Uma análise realizada pela CISCO sobre o tráfego global na internet (CISCO, 2018), destaca que em 2022 os vídeos transmitidos pela internet serão responsáveis por 82% de todo tráfego de dados global, onde, vídeos de alta definição (HD) e ultra definição (UHD) irão ocupar 78% deste consumo. Esses valores são fomentados pelo crescente número de dispositivos conectados à internet. A CISCO prevê, em seu relatório anual da internet, que até 2023 existirá um número de dispositivos conectados à internet superior a três vezes a população global, isto é, 29,3 bilhões de dispositivos (ou 3,6 dispositivos conectados por pessoa) (CISCO, 2020). Cabe mencionar que a alta demanda das aplicações nestes dispositivos exigem largura de banda ampla e baixa latência da rede (HECHT, 2016) e o crescente tráfego de vídeos aponta para um potencial problema de largura de banda da internet.

Ainda no início do ano de 2020, no mundo inteiro se espalhava a pandemia causada pelo coronavírus, agravando os desafios relacionados ao acesso à internet. Com o uso de tecnologias e internet, novos hábitos foram agregados à rotina das pessoas. Exemplos são vários, como: (i) o cenário do trabalho remoto, onde vídeo conferências aproximaram as equipes de trabalho; (ii) o uso de aplicativos para atendimento online ao cliente e serviços; e (iii) a utilização de *streaming* e *On demand* para entreterimetno, entre outros. Apenas no período observado de 28 dias em março de 2020, uma pesquisa realizada sobre o impacto do Covid-19 no *streaming* de vídeo (BITMOVINS, 2020) observou que o consumo aumentou dramaticamente, com um acréscimo de 380% nos dados baixados, enquanto o tempo assistido e a reprodução de vídeos aumentaram em 220% e 118%, respectivamente. Em conformidade com os dados apresentados, segundo NIELSEN (2020) em seu relatório geral de audiência, até agosto de 2020 a média de minutos semanais de *streaming* atingiu 142 bilhões de

minutos, enquanto no ano anterior era de 81 bilhões de minutos.

Considerando o volume de informações necessárias para representar um vídeo, o espaço massivo para armazenar as mídias digitais e o enorme tráfego gerado, se torna imprescindível o uso de técnicas eficientes de compressão de vídeos. Para isso, os codificadores de vídeo têm como foco eliminar redundâncias nos dados e reduzir informações que são pouco relevantes para o sistema visual humano. Um dado redundante é aquele que não contribui com novas informações relevantes para a representação da imagem (AGOSTINI, 2007). Segundo RICHARDSON (2010) as redundâncias de dados em um vídeo podem ser classificadas basicamente em três diferentes tipos: redundância espacial, redundância temporal e redundância entrópica. Assim, eliminar essas redundâncias e reduzir as informações pouco relevantes para o sistema visual humano, buscando otimizar a relação entre qualidade final do vídeo e o tamanho do arquivo gerado, é o principal desafio dos codificadores de vídeo atuais. Esse processo normalmente implica em perdas controladas de informação dos vídeos. Baseado neste princípio, foram desenvolvidos vários padrões de codificação, como o H.264 (padrão da ITU-T e da ISO/IEC usado em *blurays* e no Sistema Brasileiro de TV Digital), o H.265/HEVC (*High Efficiency Video Coding*) (ITU-T, 2019) e, por fim, o H.266/VVC (*Versatile Video Coding*) (ITU-T, 2020) atual estado da arte em padrões de codificação de vídeo da ITU-T e da ISO/IEC.

A utilização desses padrões, definidas por órgãos de padronização internacionais, envolvem um custo elevado devido aos *royalties* (OZER, 2015) envolvidos, onerando as empresas interessadas em desenvolver aplicações. A fim de desenvolver um codificador alternativo, aberto e livremente implementável para entrega de vídeo para um amplo conjunto de casos de uso da indústria (vídeo sob demanda, videoconferência, *streaming* e *video game streaming*), várias iniciativas foram realizadas (EGGE, 2018). Como vários atores da indústria tinham os mesmos objetivos e necessidades na área de codificação de vídeos, foi fundada, em 2017, a AOMedia (*Alliance for Open Media*). A AOMedia é um consórcio do setor da tecnologia, formado inicialmente pela Google, Cisco e Mozilla e que rapidamente passou a contar com a participação das maiores empresas de tecnologia. A AOMedia partiu do codificador VP9 da Google e dos codificadores Thor e Daala da Cisco e da Mozilla, respectivamente, para desenvolver um novo e eficiente codificador livre de *royalties*. A especificação deste codificador foi lançada em março de 2018 (AOMEDIA, 2019) e foi batizado de AV1 (*AOMedia Video 1*).

Para realizar a codificação de vídeo, compreendendo que ela deve identificar e reduzir ao máximo as redundâncias de dados, o AV1 segue o fluxo de codificadores de vídeo híbridos do estado-da-arte, dividindo o processo nas etapas de: predição intra-quadro, predição inter-quadros, transformadas, quantização, filtros e codificação de entropia, mas inserido novidades em cada uma destas etapas quando comparado

aos codificadores anteriores.

As análises recentemente apresentadas em artigos que comparam o codificador AV1, em sua implementação inicial, com outros codificadores já estabelecidos, demonstraram que o AV1, em certas condições, é superior ao HEVC se avaliada a eficiência de codificação (GROIS; NGUYEN; MARPE, 2016) e (AKYAZI; EBRAHIMI, 2018). Entretanto, se comparado ao HEVC com vídeos de mesma qualidade e resolução, é importante destacar que o tempo necessário para codificar os vídeos foi duplicado (LAYEK et al., 2017). Em outras palavras, para alcançar altas taxas de eficiência de codificação, o AV1 emprega um tempo de execução muito elevado e esta relação define o alto custo computacional envolvido na codificação neste codificador.

Além de buscar a ampliação da eficiência de codificação, os desenvolvedores do AV1 também tiveram que evitar técnicas anteriores cobertas por patentes, ampliando muito o desafio para gerar esse novo codificador e para torná-lo um codificador de vídeo competitivo, para que as empresas tenham interesse em adota-lo comercialmente. Nesse cenário, a importância da geração de implementações em software e hardware competitivas com outros codificadores do estado-da-arte se torna imprescindível para que as empresas migrem seus conteúdos para o formato AV1.

1.1 Objetivos

Este trabalho tem o objetivo principal de desenvolver uma solução para acelerar o processo de predição intra-quadro do codificador AV1, reduzindo o custo computacional envolvido na tomada de decisão do modo preditor intra-quadro.

São considerados objetivos específicos deste trabalho:

1. Compreender detalhadamente o fluxo da tomada de decisão na predição intra-quadro do AV1.
2. Investigar modelos de aprendizado de máquina que sejam adequados para aplicação no contexto da predição intra-quadro do AV1.
3. Propor um modelo de aprendizado de máquina eficiente para reduzir a complexidade do processo de predição intra-quadro do AV1.
4. Avaliar os resultados obtidos e encaminhar para publicação.

1.2 Organização

A estrutura do texto está organizada da seguinte forma: o Capítulo 2 apresenta uma visão geral do fluxo de dados, ferramentas e técnicas implementados no padrão de codificação de vídeo AV1. O Capítulo 3 aborda, mais detalhadamente, todos modos intra-quadro suportados pelo AV1. No Capítulo 4 são apresentados os conceitos

de aprendizado de máquina necessários para essa dissertação, as abordagens mais utilizadas e os algoritmos supervisionados, além de uma visão global das métricas de avaliação de modelos de classificação e técnicas de aprendizado por agrupamento. O capítulo 5 apresenta a metodologia e condições comuns de teste utilizadas nos experimentos realizados no codificador AV1 e como o problema de redução de complexidade na predição intra-quadro do AV1 foi abordado. No Capítulo 6 são introduzidas as soluções desenvolvidas com o uso de aprendizado de máquina. No Capítulo 7 são apresentados os resultados obtidos para as soluções desenvolvidas. Por fim, o último capítulo apresenta a conclusão desta dissertação.

2 FORMATO DE CODIFICAÇÃO DE VÍDEO AV1

Neste capítulo apresenta-se brevemente o fluxo de dados, as ferramentas e as técnicas implementadas na codificação de vídeo segundo o formato AV1, conforme o documento de especificação do bitstream do AV1 AOMEDIA (2019), no software de referência do AV1 [1] e referencial teórico.

2.1 Fluxo de Codificação

Conforme a Figura 1, é possível observar que o AV1 segue o fluxo de codificadores de vídeo híbridos do estado-da-arte. De forma geral, as etapas são: (i) **Particionamento**: que consiste em dividir o quadro em partes menores em tamanhos e formatos de blocos pré-definidos no formato, que facilitam, assim, a aplicação das técnicas de predição, tornando-as mais eficientes; (ii) **Predição**: que visa prever o conteúdo do bloco de vídeo com base em blocos já processados e que pode ser aplicada entre quadros vizinhos (predição inter-quadros) ou dentro do mesmo quadro (predição intra-quadro). A predição inter-quadros explora as redundâncias temporais, buscando encontrar em quadros vizinhos previamente codificados (ou quadros de referência) um bloco semelhante ao que esta sendo codificado, enquanto a predição intra-quadro tem como objetivo reduzir a redundâncias espaciais entre amostras vizinhas dentro de um mesmo quadro; (iii) **Transformada**: as informações residuais (diferença entre o bloco original e o bloco predito) são transformadas do domínio espacial para o domínio das frequências, com o objetivo de potencializar a etapa seguinte; (iv) **Quantização**: é basicamente uma divisão inteira aplicada para eliminar ou atenuar as frequências menos relevantes ao sistema visual humano, gerando, tipicamente, matrizes esparsas; (v) **Codificação de Entropia**: usa técnicas de compressão sem perdas para comprimir de forma muito eficiente as matrizes esparsas geradas pela quantização e também os dados laterais necessários ao codificador, gerando o *bitstream* final com o vídeo codificado. As novidades de cada uma destas etapas no AV1 serão brevemente descritas nas próximas subseções, onde os conceitos diretamente relacionados com

[1] AOM `https://aomedia.googlesource.com/aom/`

o trabalho desenvolvido serão mais detalhados, enquanto que os demais conceitos serão abordados de forma mais superficial.

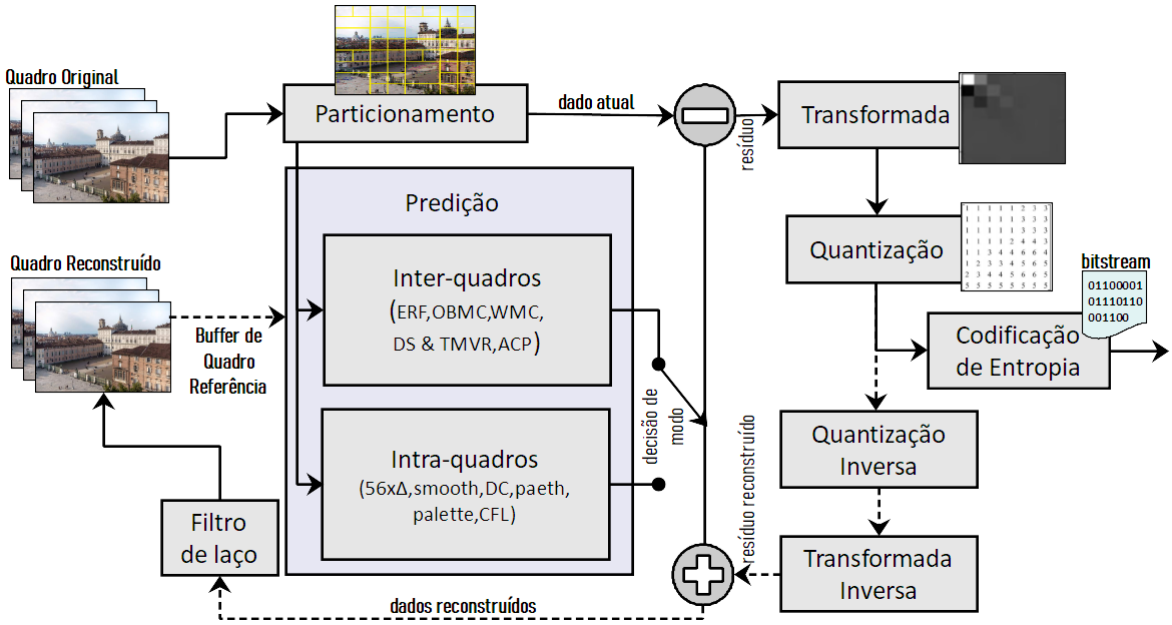

Figura 1 – Fluxograma do Codificador AV1

Além do fluxo de cinco etapas de codificação sintetizado na Figura 1, o AV1 implementa também o recurso de codificação de duas passagens (*two-pass*) que a alta eficiência é priorizada sobre o tempo de execução, este recurso permite ao codificador realizar a leitura de todo conteúdo do vídeo durante a primeira passagem neste fluxo, reunindo dados estatísticos do vídeo inteiro e armazena em um arquivo temporário (*log*) para melhor organizar os dados. Neste contexto a primeira passagem é relativamente rápida se comparada com a segunda, mas devido o requerimento de um arquivo de vídeo completo antes da codificação, o recurso de duas passagens não pode ser utilizado para transmissão ao vivo (*streaming*). A segunda passagem neste fluxo realiza a codificação de fato, alocando a taxa de *bits* de acordo com os dados estatísticos obtidos anteriormente, resultando em uma codificação mais eficiente e qualidade perceptual mais consistente no geral, ao passo de um tempo de execução elevado.

2.1.1 Particionamento

O primeiro passo da execução visto na Figura 1, é o *Coding Block Partition* ou Particionamento. Codificadores baseados em codificação em blocos apresentam um funcionamento semelhante nesta etapa, onde cada quadro de um vídeo é divido em particionamentos. Cada um destes particionamentos ainda pode ser dividido em partes menores, denominada bloco. O codificador executa as etapas de codificação sobre o bloco.

A unidade básica de particionamento do AV1 é denominada *SuperBlock* (SB) e possui 128x128 pixels ou 64x64 pixels. O *SuperBlock*, pode ser subdivido em partes menores e esta estrutura de subdivisão é denominada *partition-tree* ou árvore de particionamentos. Cada folha da árvore é um bloco e cada bloco possui seus próprios modos de predição e transformada. Na predição intra-quadro, a codificação do SB depende apenas de seus SB vizinhos acima e à esquerda (CHEN et al., 2018).

Há, na árvore de particionamento do AV1, três estruturas básicas, que definem os tipos de particionamento: (i) a árvore de predição define os blocos que serão utilizados pela etapa de predição, (ii) a árvore de transformadas, que os blocos a serem utilizados pela etapa de transformada e, por fim, (iii) a árvore de quantização, que é utilizada pela etapa de quantização no fluxo de codificação.

A árvore de predição pode atingir seis níveis de profundidade, partindo de blocos com 128x128 pixels, que pode ser subdividido até o tamanho de 4x4 pixels. Para cada profundidade, pode-se assumir uma das 10 formas de particionamento pré-definidas (AOMEDIA, 2019), representadas na Figura 2. Para melhorar a qualidade de predição de vídeos complexos, o tamanho mínimo do bloco de codificação é de 4×4 amostras de luminância (CHEN et al., 2018).

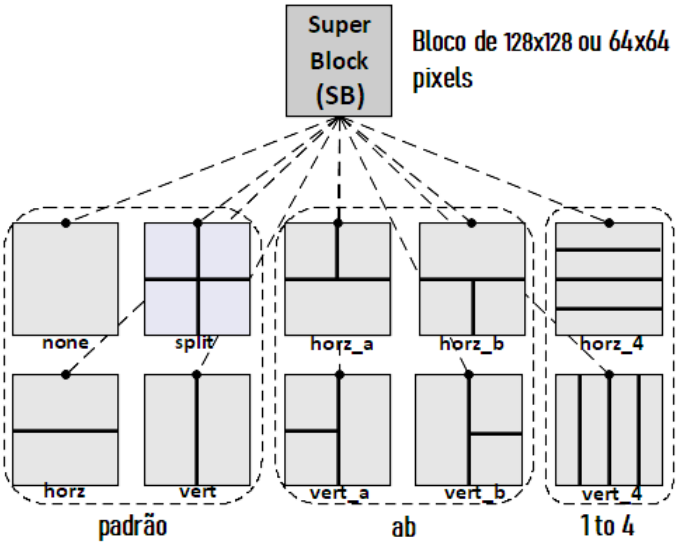

Figura 2 – Árvore de Particionamento do AV1

Diferentemente do codificador VP9, que usa uma árvore de particionamento com 4 formas, partindo do tamanho de bloco 64×64 até atingir o nível mais baixo de 4×4 (CHEN et al., 2018), o AV1 não apenas expande a árvore de particionamento para uma estrutura com 10 tipos de particionamento como mostra Figura 2, mas também aumenta o tamanho do maior bloco, se comparado com VP9. Como resultado, existem muito mais tamanhos de bloco para serem avaliados no AV1. Devido ao processo de particionamento de blocos da Figura 2, os algoritmos de predição no AV1 podem suportar uma ampla variedade de tamanhos de blocos: 64×64, 64×32, 32×64, 64×16,

16×64, 32×32, 32×16, 16×32, 32×8, 8×32, 16×16, 16×8, 8×16, 16×4, 4×16, 8×8, 8×4, 4×8 e 4×4 (HAN et al., 2020).

Como é possível notar na Figura 2, cada forma pode ser agrupada em classes: (i) **padrão**, (ii) **ab**, que correspondem as partições em "T" e (iii) **1 to 4**, ou partições assimétricas. É importante observar que a forma *split* destacada na Figura, pode ser particionada novamente, de forma recursiva, podendo assumir outra vez um dos 10 tipos de partição.

Também é importante compreender que, ao assumir a forma *split* no particionamento, um novo nível de profundidade da árvore é atingido, sendo níveis permitidos de 0 até 5, como no exemplo da Figura 3. Contudo, não são todos níveis de profundidade da árvore que podem assumir todas as formas de particionamento (AOMEDIA, 2019). O AV1 possui regras explicitas quanto esta restrição, que se da pelas classes de particionamento, conforme apresentado na Tabela 1.

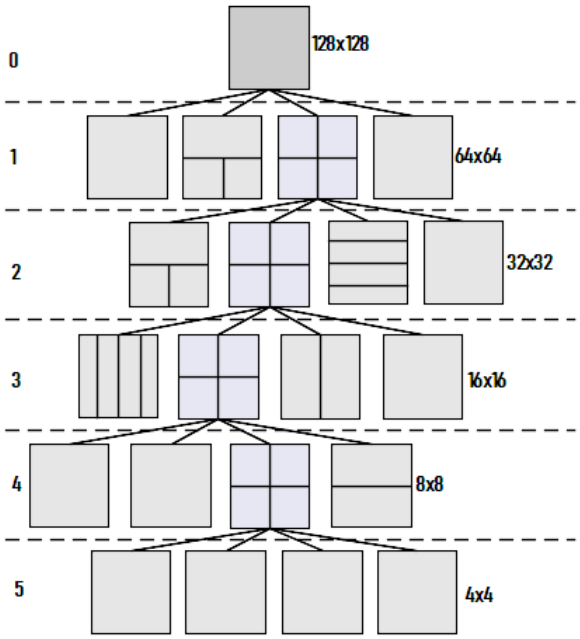

Figura 3 – Profundidade da Árvore de Particionamento

Tabela 1 – Nível de profundidade x Classes no AV1 (AOMEDIA, 2019)

Profundidade	Classes
Nível 0	Padrão
Nível 1	Padrão, ab, 1 to 4
Nível 2	Padrão, ab, 1 to 4
Nível 3	Padrão, ab, 1 to 4
Nível 4	Padrão, ab
Nível 5	apenas NONE

2.1.2 Predição

O particionamento de blocos proporciona uma flexibilidade que é essencial para permitir que o processo de predição se ajuste a uma ampla variedade de conteúdos. Assim, o processo de codificação de vídeo, visto na Figura 1, executa a predição dos blocos buscando diminuir os dados redundantes de duas formas: intra-quadro ou inter-quadros.

A predição intra-quadro explora a redundância espacial, explorando as semelhanças entre blocos vizinhos já preditos de um mesmo quadro (CORREA et al., 2010). Assim como em outros codificadores, a predição intra-quadro do AV1 busca diminuir as redundâncias espaciais do quadro, através de predições direcionais ou não-direcionais. Em relação ao VP9, foram incluídas algumas melhorias, como o aumento dos modos direcionais para 56 modos, a inclusão do modo *Chroma from Luma* (CfL), a inclusão filtros recursivos denominados intrafiltros, entre outros. Essas novidades aumentaram a complexidade e tornaram mais eficiente este processo (CHEN et al., 2018). No capítulo 3 os modos e as respectivas técnicas aplicadas na predição intra-quadro serão abordados mais profundamente, dado que este é o foco deste trabalho.

Enquanto o processo de predição intra-quadro visa diminuir as redundâncias espaciais, a predição inter-quadros destina-se à reduzir as redundâncias temporais. Conforme explica AGOSTINI (2007) “A predição inter-quadros utiliza amostras reconstruídas de quadros vizinhos temporalmente, com o objetivo em identificar a redundância temporal que ocorre em blocos temporalmente vizinhos”. A predição inter-quadros utiliza vetores de movimento para identificar regiões em quadros vizinhos buscando encontrar o bloco candidato que seja mais similar com o bloco original que está sendo predito (HAN et al., 2020).

A predição inter-quadros do AV1 suporta uma variedade de ferramentas para explorar as redundâncias temporais, como detalhado em CHEN et al. (2020). Algumas das ferramentas da predição inter-quadros do codificador AV1, que são novidades em relação ao VP9, são descritas a seguir.

1. **Quadros de Referência Estendidos**: O AV1 estende o número de referências para cada quadro. No VP9 são três referências: *LAST-frame* (passado mais próximo), *GOLDEN-frame* (passado distante) e *ALTREF-frame* (temporal futuro filtrado). O AV1 adiciona mais dois quadros passados (*LAST2* e *LAST3*) e mais dois quadros futuros (*BWDREF* e *ALTREF2*), totalizando sete quadros de referência. Quadros do tipo *ALTREF2* também são quadros temporais futuros filtrados. Quadros do tipo *BWDREF* são quadros futuros codificados de forma direta sem aplicar filtragem temporal, portanto, esse tipo de quadro é mais aplicável para referências mais próximas CHEN et al. (2020).

2. **Referência Dinâmica Espacial e Temporal do Vetor de Movimento**: Codifi-

car de forma eficiente os vetores de movimento é crucial para a eficiência de codificação global. No codificador AV1 vetores de blocos vizinhos são reaproveitados para ampliar a eficiência de codificação através desta técnica, detalhada em CHEN et al. (2020)

3. **Compensação de Movimento com Blocos Sobrepostos (OBMC)**: A OBMC atinge reduções substanciais no erro de predição através de uma média ponderada de segmentos de bloco sobrepostos durante a predição de movimento (FURHT, 2006). A OBMC só está habilitada para blocos usando um único quadro de referência e só funciona com o primeiro preditor de qualquer vizinho com dois quadros de referência (CHEN et al., 2020) (CHEN; MUKHERJEE, 2017).

4. **Compensação de Movimento Deformada (WMC)**: O AV1 explora modelos de compensação de movimento afim, permitindo dois modos: global e local (PARKER et al., 2017). A WMC busca predizer padrões de movimento em 3D, prevendo trajetórias de movimento espacial dentro de vídeos e, assim, ampliando a eficiência em relação à estimação de movimento 2D tradicional. Ambas as ferramentas de codificação do WMC (*Global Motion* e *Local Motion*) competem com modos translacionais tradicionais em nível de bloco, e são selecionados apenas se houver uma vantagem na eficiência de codificação (CHEN et al., 2020).

5. **Advanced Compound Prediction (ACP)**: O AV1 implementa um conjunto amplo de ferramentas de predição composta e isto torna a predição inter-quadros muito mais versátil se comparada com outros codificadores de vídeo. A ACP do AV1 inclui as ferramentas: *Compound wedge prediction*, *Difference-modulated masked prediction*, *Frame distance based compound prediction* e *Compound inter-intra prediction*, detalhadas em (JOSHI et al., 2017) (CHEN et al., 2018).

2.1.3 Transformadas

O módulo de transformada na Figura 1, como já mencionado, visa transformar os resíduos para o domínio das frequências.

O particionamento de bloco da transformada no AV1 suporta um grande número de tamanhos de blocos quadrados e não-quadrados, que podem ser representados nas relações 1:2/2:1 ou 1:4/4:1. São suportados blocos de tamanho 4x4 até 64x64, em um total de 19 tamanhos: 4x4, 4x8, 8x4, 8x8, 8x16, 16x8, 16x16, 16x32, 32x16, 32x32, 32x64, 64x32, 64x64, 4x16, 16x4, 8x32, 32x8, 16x64 e 64x16.

O AV1 define um conjunto de quatro transformadas 2D que podem ser aplicadas de forma independente na horizontal e na vertical, permitindo 16 diferentes combinações. Em complemento à transformada discreta dos cossenos (DCT) e à transformada discreta dos senos assimétrica (ADST), definidas no VP9, o AV1 também define o uso da flipADST (que aplica a ADST em ordem inversa) e da IDTX (que é a transformada

identidade e, basicamente, não aplica nenhuma transformada) (PARKER et al., 2016) (MUKHERJEE et al., 2015).

2.1.4 Quantização

Ao fim do processo de transformadas, as informações residuais do quadro encontram-se no domínio das frequências onde, então é aplicada a quantização, buscando eliminar ou reduzir a amplitude dos coeficientes transformados menos relevantes para o sistema visual humano. Essa operação de quantização é irreversível, pois é realizada uma divisão inteira dos coeficientes gerados pela transformada, reduzindo grande parte dos coeficientes à zero e o resto da divisão não é armazenado, logo, ao fim deste processo, perdas de informação ocorrerão entre as amostras originais e as reconstruídas na codificação (AGOSTINI, 2007).

No codificador AV1, a quantização suporta, a cada novo SB, o ajuste dos parâmetros de quantização sinalizando o *offset*, que é um parâmetro ligado à qualidade do vídeo. Este controle é realizado através do *Constant Quality* (CQ), que varia entre zero e 63. Um CQ zero indica que não há perda de qualidade, mas que a compressão será menor, enquanto que um CQ 63 indica uma perda de qualidade maior, mas com taxa de compressão máxima(RIVAZ, 2020).

2.1.5 Codificação de Entropia

Como já apresentado, existem três tipos de redundâncias de dados exploradas em codificadores de vídeos, e uma delas é a redundância entrópica. Essa redundância é relacionada à forma como os dados são codificados. Para isso, aplica-se à codificação de entropia, que tem como objetivo representar mais informações com um número menor de bits para a representação de cada símbolo codificado, utilizando diferentes técnicas e algoritmos de compressão sem perdas (AGOSTINI, 2007). Neste processo, o AV1, explora as redundâncias entrópicas e utiliza-se da codificação de entropia para reunir todas informações dos blocos transformados e quantizados, além dos dados laterais provenientes dos processos de predição, transformada e quantização vistos na Figura 1 em uma sequência de dados conforme a especificação do *bitstream* (RIVAZ, 2020).

O AV1 usa um codificador aritmético adaptativo com múltiplos símbolos. O uso de múltiplos símbolos reduz o custo computacional envolvido na codificação e decodificação, quando comparado a um codificador aritmético adaptativo binário (CHEN et al., 2020). Como o codificador de entropia não é foco deste trabalho, ele não será detalhado nesta seção.

2.1.6 Golden-Frame Group

O último conceito apresentado neste capítulo é o de *golden-frame groug* (GF). Um GF é um conjunto de quadros que é processado de forma totalmente independente de outros quadros. A Figura 4 exemplifica uma estrutura de *golden-frame groug* (GF), onde podem ser observados os diferentes tipos de quadros permitidos pelo AV1. Nesta figura estão apresentadas a ordem temporal de apresentação dos quadros (*Display Order*) e a ordem de processamento destes quadros (seus números). Todos os quadros dentro do GF compartilham os quadros GOLDEN e ALTREF. O quadro GOLDEN é codificado apenas com predição intra-quadro, pois não usa nenhuma referência temporal. ALTREF e todos os demais quadros podem usar, também, a predição inter-quadros. Sempre que dois quadros, um futuro e um passado, estejam disponíveis, então a bi-predição pode ser aplicada, usando ambos os quadros como referência. (Chen, Di and Liu et al., 2018). As setas na Figura 4 indicam quais quadros são usados como referência no processo de predição de cada quadro. Por exemplo, o quadro 8 pode usar os quadros 1, 2, 3 e 7 como referência para a predição de seus blocos. Por sua vez, o quadro 8 pode ser usado como referência do processo de predição dos quadros 9 e 10.

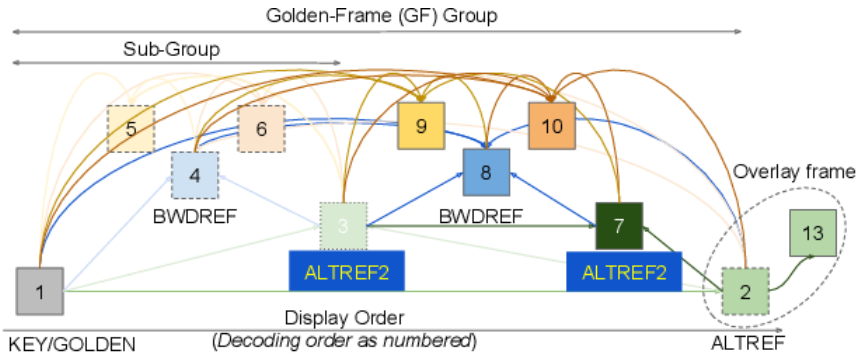

Figura 4 – Estrutura de Várias Camadas de um Grupo golden-frame (CHEN et al., 2020)

3 PREDIÇÃO INTRA-QUADRO NO CODIFICADOR AV1

Como já mencionado, o particionamento de blocos do AV1 permite blocos entre 128x128 até 4x4 amostras, mas a predição intra-quadro não pode usar todos estes tamanhos de bloco. Os modos de predição intra-quadro suportam tamanhos de bloco até 64x64 amostras, no máximo. Assim, a predição intra-quadro suporta os mesmos 19 tamanhos de bloco das transformadas, ou seja: 4x4, 4x8, 8x4, 8x8, 8x16, 16x8, 16x16, 16x32, 32x16, 32x32, 32x64, 64x32, 64x64, 4x16, 16x4, 8x32, 32x8, 16x64 e 64x16.

O AV1 trouxe muitas novidades em relação ao VP9 e a outros codificadores atuais, conforme apresentado no Capítulo 2, e muitas destas novidades encontram-se na etapa de predição intra-quadro. Como a predição intra-quadro é o foco deste trabalho, nesse capítulo serão detalhados os modos de predição intra-quadro definidos no codificador AV1. A ampliação no número de modos de predição suportados na predição intra-quadro causam uma ampliação expressiva no custo computacional desta etapa de predição, mas estes modos foram incluídos para ampliar a eficiência de codificação. Porém, dependendo das características específicas do vídeo a ser processado, o uso de alguns destes modos de predição nem sempre representam um ganho significativo em eficiência de codificação e seguem causando um aumento expressivo no custo computacional. Assim, são importantes soluções que reduzam o custo computacional envolvido nesta etapa.

3.1 Modos de Predição Intra-Quadro

A etapa de predição intra-quadro do VP9 suporta 10 modos, incluindo oito modos direcionais correspondentes a ângulos de 45 a 207 graus, e dois preditores não direcionais: DC e *True Motion* (TM) (MUKHERJEE et al., 2015). Como o VP9 e outros demais codificadores de vídeo, no geral, o AV1 também realiza a predição intra-quadro por extrapolação ou interpolação direcional utilizando como referência as informações de blocos já preditos de um mesmo quadro. No entanto, o desenvolvimento do codificador AV1 buscou atingir maiores ganhos da taxa de compressão nesta etapa,

utilizando preditores aprimorados (alguns já vistos no VP9), e definindo novas ferramentas e modos de predição.

Os modos de predição implementados pelo AV1 são: preditores direcionais correspondendo a ângulos entre 36 e 212 graus (com 56 modos), o modos para suavizar superfícies (*Smooth, Smooth Vertical, Smooth Horizontal*), o modo DC, o modo *Paeth* (substituto do TM), o modo *Recursive-based-filtering* ou Intra Filtros, o modo *Chroma-from-Luma* (CfL) e os modos cópia de blocos intra e paleta (estes dois últimos direcionados para vídeos de conteúdo de tela) (CHEN et al., 2020).

A predição intra-quadro faz uso das amostras já reconstruídas do bloco atual, localizados à esquerda e acima deste bloco, para formar uma predição para o bloco atual. As saídas deste processo são amostras preditas. As referências são definidas no processo de predição como a linha de amostras acima do bloco atual e a coluna de amostras à esquerda do bloco atual. Estas referências podem ser representadas em uma matriz 2D e, assim, cada modo de predição pode utilizar as amostras de acordo com o modo aplicado.

Importante ressaltar que o software de referência do codificador AV1, o libaom (AOM, 2020) em sua implementação de versão 3.0 [1] investigada neste trabalho, apresenta ao menos uma heurística destinada à como os modos de predição intra são avaliados, está define a ordem dos quais os modos deverão ser aplicados durante a predição intra-quadro, os modos são listados respectivamente por: Modo DC, Modo Direcional Horizontal (*H_Pred*), Modo Direcional Vertical (*V_Pred*), Modo *Smooth*, Modo *Paeth*, Modo *Smooth* Vertical, Modo *Smooth* Horizontal e seguido pelos Modos Direcionais dos ângulos nominais de 135, 203, 157, 67, 113 e 45 graus. Desta heurística o modo *Chroma-from-Luma* (CfL), modos dedicados para conteúdo de tela e filtros intra são omitidos.

As subseções desta Seção 3.1 irão detalhar cada um dos modos de predição intra suportados pelo codificador AV1.

3.1.1 Modos Direcionais

Para explorar mais amplamente a redundância espacial em texturas direcionais, o codificador AV1 implementa 56 ângulos de predição para os modos direcionais. Este maior número de ângulos é alcançado a partir de 8 ângulos nominais: 45, 67, 90, 113, 135, 157, 180, 203 graus; e duas variáveis: passo e ângulo delta, que assumem o valor de 3 graus e valores inteiros entre -3 e +3 respectivamente. O ângulo de predição é calculado somando o ângulo nominal com produto da multiplicação do ângulo delta com valor do passo (XU; JEON, 2021), este cálculo é apresentado na equação 1.

$$pAngle = Angulo_{nominal} + Angulo_{\Delta} \times Passo \quad (1)$$

[1] hash e3cad234fe5fd69d7f1919579ff442056c9895ed/

Os ângulos nominais (em negrito) e cada variação de ângulo possível (*pAngle*) são listados na Tabela 2, que também lista o nome dado para cada conjunto de modos intra. Todos os 56 ângulos suportados estão ilustrados na Figura 5, onde os ângulos nominais estão destacados em azul.

Tabela 2 – Ângulos Suportados pela Predição Intra Direcional do AV1.

Modos Intra	Variações de Ângulo Suportados (pAngle)						
D45_PRED	36	39	42	**45**	48	51	54
D67_PRED	58	61	64	**67**	70	73	76
V_PRED	81	84	87	**90**	93	96	99
D113_PRED	104	107	110	**113**	116	119	121
D135_PRED	126	129	132	**135**	138	141	144
D157_PRED	148	151	154	**157**	160	163	166
H_PRED	171	174	177	**180**	183	186	189
D203_PRED	194	197	200	**203**	206	209	211

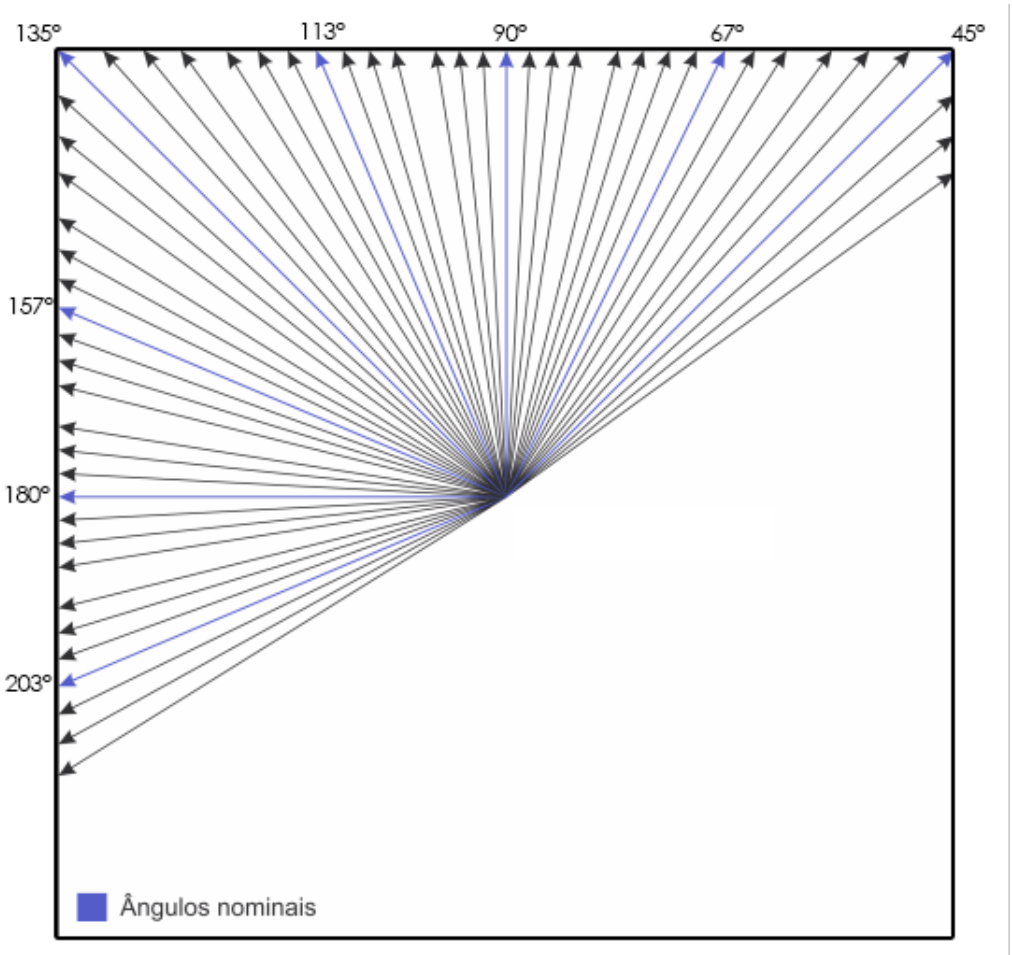

Figura 5 – Ângulos nominais (setas azuis) e variações de ângulo fino (setas pretas) na predição intra do AV1.

Os oito modos direcionais originais do VP9 são usados como uma base no codificador AV1, com um sinal suplementar para refinar o ângulo de predição. Isto compreende até três etapas no sentido horário ou no sentido anti-horário, cada um de três graus, como mostrado na Figura 5. Um filtro bilinear de dois *taps* é usado para interpolar as amostras de referência. Então um processo de filtragem e *upscaling* pode ser aplicado às amostras de referência da linha superior e da coluna à esquerda, que precedem o processo de predição (CHEN et al., 2018).

Os ângulos de predição não são aplicados de forma direta para o cálculo das amostras de referência. Eles devem ser mapeados conforme a Tabela 3 e 4, através dos valores no eixo de coordenadas cartesiano dx, dy e a derivada direcional intra do AV1 (D). O processo começa definindo os valores, como descrito na Tabela 3, para calcular o valor para dx correspondente ao Eixo X e para dy ao Eixo Y, dependendo da variação de ângulo de *pAngle*, exceto para os ângulos de 90 e 180 graus, que seguem uma regra específica.

Tabela 3 – Condicionais para variáveis Abscissa e Ordenada no Modo Direcional

	Expressão
dx	D[pAngle] se pAngle < 90 OU D[180-pAngle] se pAngle > 90 e pAngle < 180 SE NÃO é 1
dy	D[pAngle-90] se pAngle > 90 e pAngle < 180 OU D[270-pAngle] se pAngle > 180 SE NÃO é 1

Tabela 4 – Derivação das variáveis dx e dy

	Ângulos
pAngle	3, 6, 9, 14, 17, 20, 23, 26, 29, 32, 36, 39, 42, 45, 48, 51, 54, 58, 61, 64, 67, 70, 73, 76, 81, 84, 87
Derivada Direcional (D)	0, 0, 0, 1023, 0, 0, 547, 0, 0, 372, 0, 0, 0, 0, 273, 0, 0, 215, 0, 0, 178, 0, 0, 151, 0, 0, 32, 0, 0, 116, 0, 0, 102, 0, 0, 0, 90, 0, 0, 80, 0, 0, 71, 0, 0, 64, 0, 0, 57, 0, 0, 51, 0, 0, 5, 0, 0, 0, 40, 0, 0, 35, 0, 0, 31, 0, 0, 27, 0, 0, 23, 0, 0, 19, 0, 0, 15, 0, 0, 0, 0, 11, 0, , 7, 0, 0, 3, 0, 0

Na Tabela 3, é importante observar que os valores de dx e dy determinam se a predição necessitará de uma ou ambas as matrizes de amostras de referências (linha acima e coluna à esquerda). Isto é definido se o valor um for atribuído para dx ou dy.

Como mencionado, todos ângulos dos modos de predição são alcançados utilizando a definição das Tabelas 3 e 4, entretanto, os ângulos 90 e 180 graus apresen-

tam seus cálculos de uma forma diferente, pois eles apenas realizam uma cópia das amostras de referência, preenchendo as amostras preditas utilizando as amostras de referência no sentido do ângulo. A Figura 6, representa a geração do bloco predito para (i) 90 graus que utiliza amostras da linha superior e, (ii) 180 graus com amostras da coluna à esquerda.

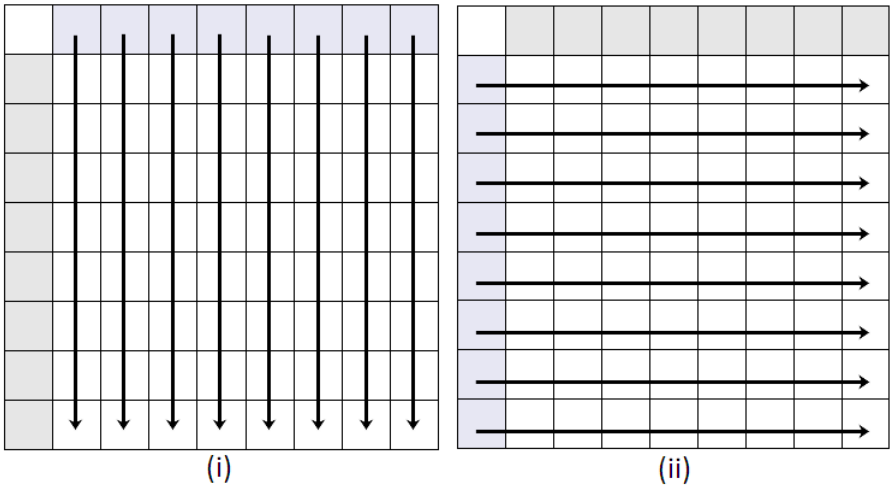

Figura 6 – Predição dos Ângulos de 180 e 90 graus 8x8 no AV1.

Para blocos de tamanho inferior a 8x8, apenas os ângulos nominais são permitidos, uma vez que o pequeno número de amostras a serem preditas não justifica o custo adicional da granularidade mais fina (HAN et al., 2020).

Após as variáveis dx e dy serem derivadas, estas são utilizadas no algoritmo apresentado na documentação do software de referência do codificador AV1, que pode ser visto em (AOMEDIA, 2019) seção 7.11.2.4 que, com base no valor de *pAngle*, são possíveis cinco condições de realizar a predição em si, assim, as saídas deste processo são amostras preditas.

3.1.2 Modo Smooth

O codificador AV1 traz três novos modos de predição *smooth*, que são indicados para a predição de regiões com textura em gradiente. Neste caso, é utilizada uma interpolação linear para gerar superfícies muito lisas compostas por amostras filtradas a partir das amostras de referência da coluna à esquerda e da linha acima. Os três modos são o *Smooth* Vertical, o *Smooth* Horizontal e o *Smooth*, que é uma combinação dos dois primeiros. A especificação do codificador denomina esses três modos como *SMOOTH_V_PRED*, *SMOOTH_H_PRED* e *SMOOTH_PRED* respectivamente (HAN et al., 2020).

Os preditores *smooth* não direcionais realizam a predição do bloco por interpolação quadrática em direções verticais ou horizontais, ou a sua média. Após a aproximação das amostras de referência inferior esquerda (BL) e superior direita (TR), preenchendo

a coluna mais à direita (T) e a linha mais abaixo (L) com intuito de selecionar a amostra predita (P) com o gradiente mais baixo. O modo *Smooth* Vertical realiza o processo de interpolação no sentido vertical, *SMOOTH_V_PRED* e utiliza amostras superiores e a última amostra de referência à esquerda *LeftCol[h-1]*, este cálculo é apresentado na equação 2. O modo *Smooth* Horizontal também realiza uma interpolação, mas no sentido horizontal, *SMOOTH_H_PRED*, em um processo similar ao vertical, utilizando amostras de referência à esquerda e a última amostra de referência superior *AboveRow[w-1]*, este cálculo por sua vez é apresentado na equação 3. Ambos modos demonstrados na Figura 7.

$$PV = W(y) \times T + (256 - W(y)) \times BL \tag{2}$$

$$PH = W(x) \times L + (256 - W(x)) \times TR \tag{3}$$

$$P = \frac{(PH + PV)}{2} \tag{4}$$

Por fim, o modo *Smooth* consiste em realizar a média entre as amostras preditas na interpolação, *SMOOTH_PRED*, visto na equação 4. As equações citadas utilizam a notação “W(x) e W(y)” que consistem ao coeficiente *Smooth*, ou peso (W), que é um conjunto constante para os coeficientes de interpolação referentes às distâncias (x,y) entre as amostras de referência, amostra predita e o tamanho do bloco, onde cada amostra de referência (*AboveRow*,*LeftCol*) são multiplicadas por um valor atribuído à sua posição de interpolação. O filtro de interpolação tem uma precisão de 1/256 pixel (RIVAZ, 2020).

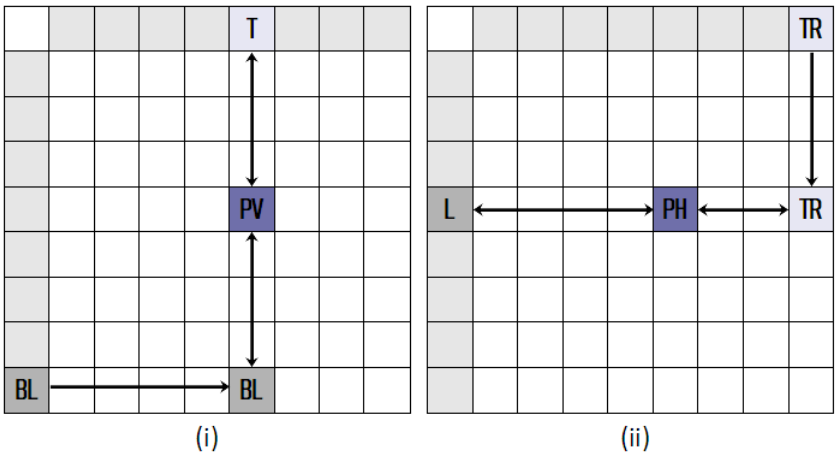

Figura 7 – Interpolação do (i) *Smooth* Vertical e do (ii) *Smooth* Horizontal .

Na Tabela 5 são encontrados todos os pesos de multiplicação de altura (h) do bloco para o modo vertical e de largura (w) do bloco para o modo horizontal, de acordo com o tamanho do bloco.

Tabela 5 – Tamanho do Bloco x Conjunto de Pesos (W) para os Modos de Predição *Smooth* Vertical e Horizontal no AV1.

Tamanho do Bloco	Pesos (W)
4	{255, 149, 85, 64}
8	{255, 197, 146, 105, 73, 50, 37, 32}
16	{255, 225, 196, 170, 145, 123, 102, 84, 68, 54, 43, 33, 26, 20, 17, 16}
32	{255, 240, 225, 210, 196, 182, 169, 157, 145, 133, 122, 111, 101, 92, 83, 74, 66, 59, 52, 45, 39, 34, 29, 25, 21, 17, 14, 12, 10, 9, 8, 8}
64	{255, 248, 240, 233, 225, 218, 210, 203, 196, 189, 182, 176, 169, 163, 156, 150, 144, 138, 133, 127, 121, 116, 111, 106, 101, 96, 91, 86, 82, 77, 73, 69, 65, 61, 57, 54, 50, 47, 44, 41, 38, 35, 32, 29, 27, 25, 22, 20, 18, 16, 15, 13, 12, 10, 9, 8, 7, 6, 6, 5, 5, 4, 4, 4}

3.1.3 Modo DC

Embora o codificador AV1 implemente o modo DC com base na versão encontrada no VP9 (MUKHERJEE et al., 2015), este preditor é bem conhecido e presente em diversos outros codificadores de vídeo anteriores. Este modo consiste em gerar um bloco predito a partir da média das amostras de referência, conforme sua disponibilidade. Assim, quatro casos distintos podem ocorrer: (i) quando todas as amostras de referência estão disponíveis, (ii) quando apenas as amostras superiores estão disponíveis, (iii) quando apenas as amostras à esquerda estão disponíveis, ou (iv) quando nenhuma amostra de referência está disponível. Quando as amostras de referência superior e esquerda estão disponíveis, o preditor gera uma superfície sólida, representada pela média entre essas referências. Se um conjunto de amostras está disponível e não há o outro, apenas as referências do conjunto disponível serão utilizadas para calcular a média e é realizado um arredondamento nos resultados. Mas se nenhuma amostra de referência está disponível, então será utilizado o valor médio da representação das amostas de referência, que é definido pelo número de bits por amostra do vídeo. Este número pode ser 8, 10 ou 12 bits por amostra. Na Figura 8, são demonstrados exemplos para cada um dos quatro casos, considerando um bloco 4x4. A Figura 8 (iv) mostra um exemplo onde as amostras de referência não estão disponíveis e com amostras de 8 bits. Nesse caso, o valor médio de valores representados por 8 bits é 128. Este preditor DC é extremamente eficiente para regiões com textura

homogênea.

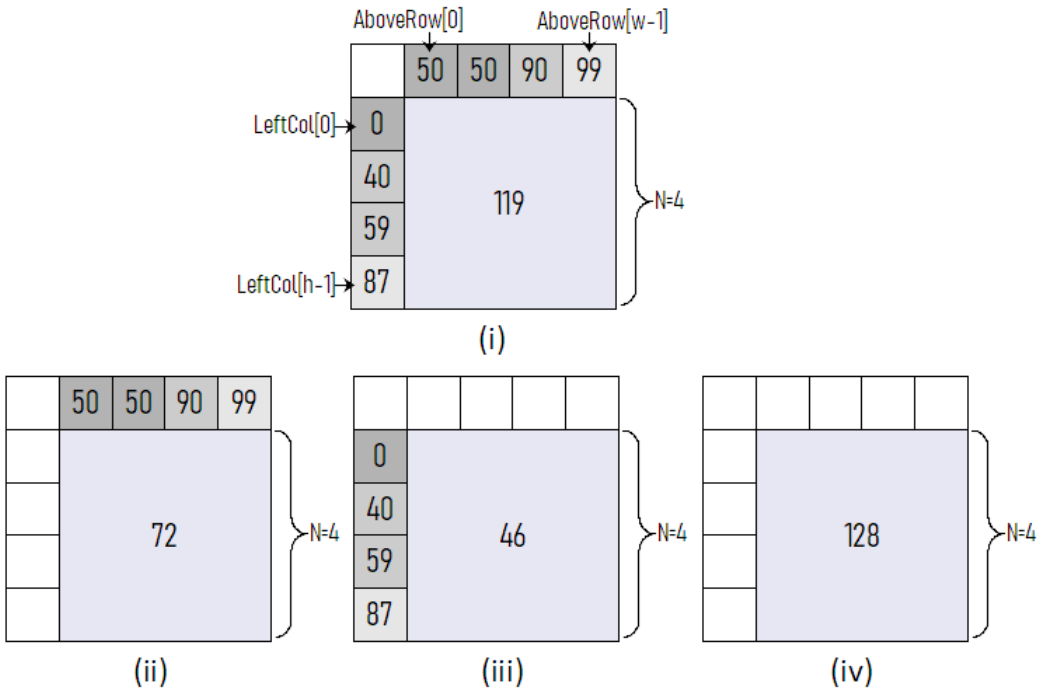

Figura 8 – Exemplos para casos de predição DC no AV1.

3.1.4 Modo *Paeth*

O modo *Paeth* é introduzido no codificador AV1 para substituir o *True Motion* (TM) presente no VP9. Para cada bloco predito, é feita uma interpolação das amostras, com base em três referências: *top*, *left*, *topleft*. Esse modo consiste em calcular uma base com as três amostras de referência e, em seguida, gerar três amostras a partir do valor absoluto da subtração da base e cada referência, resultando as amostras: *pTop*, *pLeft* e *pTopLeft*, este cálculo é apresentado nas equações 5 a 8. Na Tabela 6 é apresentada, a comparação entres essas três novas referências que, visa adotar a amostra de referência a partir desta posição (p) com o menor gradiente.

$$base = (left + top - topleft) \tag{5}$$

$$pTop = |(base - top)| \tag{6}$$

$$pLeft = |(base - left)| \tag{7}$$

$$pTopLeft = |(base - topleft)| \tag{8}$$

Tabela 6 – Condicional para escolha da amostra no modo Paeth no AV1.

	Expressão
p	left se pleft <= ptop e pleft <= ptopleft OU top se ptop <= ptopleft SE NÃO é topleft

3.1.5 Modo *Recursive-based-filtering*

Para capturar a correlação espacial do coeficiente das amostras de referência, um conjunto de filtros lineares são projetados unicamente para componentes de amostras de luminância, os quais compõem uma distribuição de amostras 4x2 usando os sete vizinhos adjacentes. As amostras preditas servem como referência para as próximas amostras 4x2 no bloco atual. Um total de cinco conjuntos diferentes de preditores lineares são definidos na especificação: *DC, V, H, D135 e Paeth* (RIVAZ, 2020). Cada um representa um padrão de coeficiente diferente da correlação espacial. A Tabela 7 lista todos coeficientes para cada modo. Basicamente, esse modo consiste na aplicação direta de filtros nas amostras preditas (p) para compor o bloco predito. A Figura 9 ilustra um exemplo com duas iterações de aplicação dos filtros, (i) representa posição 0, enquanto em (ii) temos a posição 1.

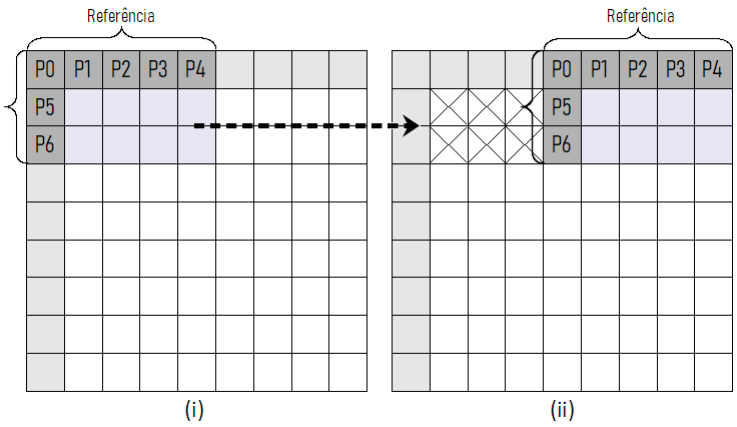

Figura 9 – Exemplo de aplicação dos Filtros Recursivos para um bloco 8x8.

3.1.6 Modo CfL

Modo *Chroma-from-Luma* ou CfL, consiste em realizar uma predição para os blocos de crominância (*Chroma*) com uma função linear, a partir das amostras reconstruídas do bloco de luminância (*Luma*). O CfL do AV1 é uma melhoria das variantes encontradas nos codificadores Thor e Daala, apesar do conceito de prever amostras de crominância a partir de amostras luminância reconstruídas ter sido proposto pela primeira vez no codificador HEVC CHEN (2011), mas rejeitada com a justificativa de aumentar consideravelmente a complexidade do decodificador.

Tabela 7 – Coeficientes dos conjuntos de Filtros Recursivos no AV1.

Modo	Posição	Coeficiente
0 (DC)	0	{-6, 10, 0, 0, 0, 12, 0}
	1	{-5, 2, 10, 0, 0, 9, 0}
	2	{-3, 1, 1, 10, 0, 7, 0}
	3	{-3, 1, 1, 2, 10, 5, 0}
	4	{-4, 6, 0, 0, 0, 2, 12}
	5	{-3, 2, 6, 0, 0, 2, 9}
	6	{-3, 2, 2, 6, 0, 2, 7}
	7	{-3, 1, 2, 2, 6, 3, 5}
1 (V)	0	{-10, 16, 0, 0, 0, 10, 0}
	1	{-6, 0, 16, 0, 0, 6, 0}
	2	{-4, 0, 0, 16, 0, 4, 0}
	3	{-2, 0, 0, 0, 16, 2, 0}
	4	{-10, 16, 0, 0, 0, 0, 10}
	5	{-6, 0, 16, 0, 0, 0, 6}
	6	{-4, 0, 0, 16, 0, 0, 4}
	7	{-2, 0, 0, 0, 16, 0, 2}
2 (H)	0	{-8, 8, 0, 0, 0, 16, 0}
	1	{-8, 0, 8, 0, 0, 16, 0}
	2	{-8, 0, 0, 8, 0, 16, 0}
	3	{-8, 0, 0, 0, 8, 16, 0}
	4	{-4, 4, 0, 0, 0, 0, 16}
	5	{-4, 0, 4, 0, 0, 0, 16}
	6	{-4, 0, 0, 4, 0, 0, 16}
	7	{-4, 0, 0, 0, 4, 0, 16}
3 (D135)	0	{-2, 8, 0, 0, 0, 10, 0}
	1	{-1, 3, 8, 0, 0, 6, 0}
	2	{-1, 2, 3, 8, 0, 4, 0}
	3	{0, 1, 2, 3, 8, 2, 0}
	4	{-1, 4, 0, 0, 0, 3, 10}
	5	{-1, 3, 4, 0, 0, 4, 6}
	6	{-1, 2, 3, 4, 0, 4, 4}
	7	{-1, 2, 2, 3, 4, 3, 3}
4 (Paeth)	0	{-12, 14, 0, 0, 0, 14, 0}
	1	{-10, 0, 14, 0, 0, 12, 0}
	2	{-9, 0, 0, 14, 0, 11, 0}
	3	{-8, 0, 0, 0, 14, 10, 0}
	4	{-10, 12, 0, 0, 0, 0, 14}
	5	{-9, 1, 12, 0, 0, 0, 12}
	6	{-8, 0, 0, 12, 0, 1, 11}
	7	{-7, 0, 0, 1, 12, 1, 9}

A Figura 10 demonstra o processo da predição CfL do AV1, onde as amostras de luminância reconstruídas são amostras que já foram somadas com os seus respectivos resíduos. Primeiramente, para fazer a predição de dois blocos de amostras de

crominância (Cb e Cr), o bloco reconstruído de amostras de luminância (Y) deve ser subamostrado para o mesmo tamanho dos blocos de crominância, dependendo da taxa de subamostragem de cores utilizada para representar o vídeo. Em seguida, é feita a média de todas as amostras de luminância e essa média é subtraída de cada uma das amostas de luminância de entrada. Esse resultado é multiplicado por um fator de escala (α), que são função dos valores originais de Cb e Cr. Por fim, esse resultado é adicionado às amostras de crominância para gerar a predição final. O fator de escala pode variar entre [-2,2] para cada canal de crominância e ambos devem ser sinalizados no *bitstream* (TRUDEAU; EGGE; BARR, 2017).

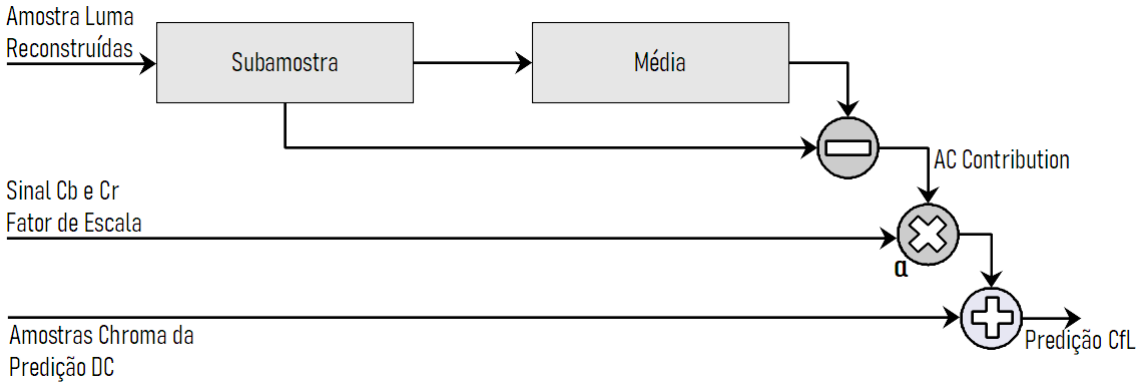

Figura 10 – Ilustração das Etapas do Preditor CfL no AV1.

3.1.7 Modos Dedicados para Conteúdo de Tela

O AV1 define dois modos da predição intra-quadro dedicados para vídeos de conteúdo de tela.

Uma das características de vídeos de conteúdo de tela que é explorada é o fato de que, em muitos casos, os blocos podem ser preditos por um pequeno número de amostras de referência de cores. Assim, o AV1 introduz o novo preditor Paleta de Cores ao codificador intra-quadro. Para tanto, o bloco predito tem disponível uma paleta de cores que varia entre 2 a 8 cores de base e um conjunto de paletas para cada canal (Y, Cb e Cr) das amostras. O número de cores base é uma atribuição do codificador que determina a relação entre qualidade e compressão do vídeo (CHEN et al., 2018).

Estas paletas são processadas pelo codificador e enviadas para o decodificador juntamente com a posição da amostra da paleta, para que ele possa gerar as amostras reconstruídas a partir da paleta. É importante destacar que este modo é permitido apenas para blocos de tamanho 8x8 ou maior.

Outro modo novo definido pelo AV1 para vídeos de conteúdo de tela é o *Intra Block Copy* (IntraBC). Esse modo permite ao preditor se referir aos blocos previamente reconstruídos no mesmo quadro, de forma semelhante à forma como preditor

inter-quadros se refere a blocos de quadros anteriores. Este modo é especialmente eficiente para quadros de vídeos de conteúdo de tela, assim como o *Color Pallete*, onde muitas texturas e padrões repetidos estão presentes (CHEN et al., 2018).

O modo IntraBC se assemelha muito à predição inter-quadros e, inclusive, gera um vetor de movimento associado ao bloco predito. Esse modo só é permitido para quadros onde a predição inter-quadros convencional não foi realizada.

4 APRENDIZADO DE MÁQUINA

Este capítulo apresenta os conceitos básicos sobre aprendizado de máquina relacionados a este trabalho, apresentando brevemente as abordagens mais utilizadas e descrevendo seus algoritmos com suas características e especificidades. Uma visão geral das métricas de avaliação de modelos de classificação e, por fim, a técnica do aprendizado por agrupamento também serão apresentadas neste capítulo.

4.1 Conceitos Básicos

É notório que o homem sempre buscou artifícios que auxiliassem o seu trabalho, reduzindo o seu esforço, agilizando tarefas e simplificando de diversas maneiras sua execução, construindo máquinas e fabricando ferramentas para tais atividades. Isto não é recente, pois temos registros de ferramentas de pedra com milhares de anos utilizadas como pontas de seta (ANDRADE; LOPES; VILELA, 2014), para facilitar a caça, ou, ainda, a conceituação de Arquimedes sobre Alavancas (ASSIS, 2008), utilizada para multiplicar a força mecânica que pode ser aplicada a outro objeto (ou vantagem mecânica).

Este aprendizado do homem, por vezes, parte de observações do seu meio. Observando como algo é feito enquanto busca entender seu funcionamento, depois somando todas ideias e informações captadas, tenta construir um objeto sob algumas leis. Essas restrições são aplicadas ora intencionalmente outrora sem pensar, o que leva de forma pragmática o homem aprimorar este objeto, tornando-o mais eficiente e avançando tecnologicamente.

Portanto, o aprendizado de máquina (*machine learning*) pode aplicar o mesmo conceito em estudar dados relacionados a um contexto já conhecido, com principal objetivo de criar um modelo matemático. Este modelo pode ser treinado para inferir, com base nas informações que receber no futuro, a melhor ação para atingir um objetivo. Essa decisão é tomada observando seus resultados e por meio de uma abordagem estatística. Assim, as probabilidades daquela ação podem ser inferidas e, então é possível escolher a ação mais provável de ser a melhor (com menor erro

possível) naquele contexto (BONACCORSO, 2017).

De forma geral, a base do aprendizado de máquina é o aprendizado indutivo de conceitos (AIC), como explicam MONARD; BARANAUSKAS (2003): "o aprendizado indutivo é efetuado a partir de raciocínio sobre exemplos fornecidos por um processo externo ao sistema de aprendizado", e (CARBONELL; MICHALSKI; MITCHELL, 1983): "dado um conjunto de exemplos e contra-exemplos de um conceito, o preditor induz uma descrição geral do conceito que descreve todos os exemplos positivos e nenhum dos contra-exemplos".

Este aprendizado pode ser divido em duas abordagens básicas: **supervisionado** e **não supervisionado**. Mas também existem outras abordagens em aprendizado de máquina como **por reforço**, que serão brevemente apresentados nas seções seguintes.

4.1.1 Aprendizado Supervisionado

Aprendizado supervisionado é a abordagem mais utilizada na resolução de problemas em aprendizado de máquina. Essa abordagem tem como característica principal o fato de que os dados utilizados (ou amostras) para treinar o modelo contém a resposta à ser prevista (denominada de classe - *target*), ou contém dados rotulados, isto é, o valor resultante para aquele grupo de variáveis (ou *features*) observadas. Ainda como menciona BONACCORSO (2017), a abordagem de aprendizado supervisionado é caracterizada também pelo conceito de um observador, cujo principal propósito é fornecer ao algoritmo preditor uma medida precisa do seu erro.

Sendo assim, a abordagem do aprendizado supervisionado aprende a partir de variáveis conhecidas, previamente definidas com uma saída, onde estes mesmos valores são base para a supervisão do modelo, permitindo o agente ou algoritmo ajustar as previsões com base nos erros, podendo sempre consultar o valor de saída previsto com o esperado.

Dentre os algoritmos mais comuns em problemas de aprendizado supervisionado, como menciona SHOBHA; RANGASWAMY (2018), estão: Regressão Linear, Árvores de Decisão (*Decision Trees* - DT), Redes Bayesianas e Máquina de vetores de suporte (*Support Vector Machine* - SVM).

Para os problemas de aprendizado supervisionado há duas tarefas principais de modelagem: classificação e regressão. Em um problema de regressão, o modelo matemático possui a tarefa de mapear variáveis de entrada para uma função de saída contínua. Enquanto um problema de classificação busca mapear variáveis de entrada em classes distintas (saída discreta).

O modelo preditivo de classificação exige que às variáveis de entrada (**X**), geralmente chamadas de amostras (*samples*), estejam rotuladas em classes distintas (**y**). Podem ser atribuído às variáveis **X** ou valores categóricos (que representam um nú-

mero finito de categorias, que podem não ter uma ordem lógica) ou valores discretos (sendo sempre numéricos, que têm um número contável finito ou infinito e inteiro). Somente fazem sentido valores discretos para variáveis **y**, que geralmente são binários, ou seja, a classificação é representada em duas classes apenas (0 ou 1) (DUDA et al., 2001). Embora também se possa ter problemas com mais de duas classes, denominando-os assim de multiclasse.

Problemas binários tratam regularmente cenários de Sim ou Não, Positivo ou Negativo, Verdadeiro ou Falso, isto é, o processo de tomar as variáveis **X** e atribuir a probabilidade prevista de uma amostra pertencer ou não à classe. Por exemplo: “um e-mail é ou não é um Spam ” ou “o aluno aprovou ou reprovou”. Em contraponto, problemas multiclasse são cenários onde existem ene classes a serem inferidas. Dependendo da técnica utilizada, o modelo terá sua resposta prevista em escala distribuída entre as classes possíveis, caberá as métricas escolhidas para determinar o rótulo **y** da amostra **X** que está sendo analisada. A resposta pode ser interpretada como as probabilidades para cada classe.

As classificações binárias e multiclasse possuem, tipicamente, as mesmas métricas de avaliação do modelo. A única diferença é que, para problemas multiclasse, usualmente trata-se cada classe individualmente, agrupando todas as demais como pertencentes a uma classe agrupada, para usar uma métrica binária e obter sua média macro. As métricas de avaliação de modelos serão detalhadas na Seção 4.3.

O modelo preditivo de regressão pode possuir variáveis de entrada (**X**) discretas assim como na classificação, mas não utiliza classes. Em vez disto, possui uma variável de saída (**y**) contínua, que é um valor real (número inteiro ou valor de ponto flutuante) (BONACCORSO, 2017). Usualmente, representam um instrumento de medida de quantidades: peso, tempo, altura e etc. Podemos, então, caracterizar a regressão como um modelo estatístico empregado para prever valores numéricos na reta real ao invés de dados rotulados, buscando encontrar a correlação entre variáveis.

Assim como a classificação, o modelo de regressão possui métricas próprias de avaliação e dentro de todas elas, o mais habitual é o calculo do erro quadrático médio entre valores de amostra e predição. Cabem exemplos de modelos comuns de regressão: previsões do tempo e análise preditiva de mercado imobiliário ou bolsa de valores.

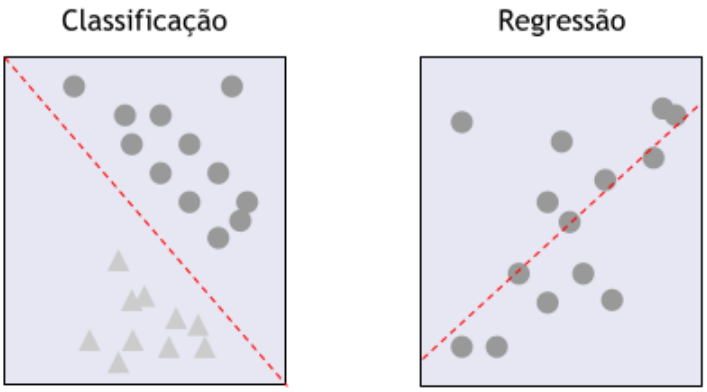

Figura 11 – Diferença entre Classificação e Regressão

A diferença entre estas abordagens de aprendizado supervisionado se dá, sobretudo, pelo método pelo qual a saída do modelo é obtida e a forma de saída. Isto é, enquanto a classificação envolve um processo de descoberta de um modelo ou função através da qual as amostras são previstas em forma de dados rotulados (classe discreta) não ordenados, na regressão busca-se diferir os dados em valores reais contínuos ordenados e não rótulos.

Ainda quando se avalia modelos de aprendizado supervisionado três aspectos devem ser considerados: viés, variância e relação entre viés-variância. Estes influenciam ou evitam o ajuste excessivo (*overfitting*) ou ajuste insuficiente (*underfitting*) e generalização do modelo ao problema e conjuntos de dados. Os resultados de um dado modelo de aprendizado são reflexo de um *underfitting* quando o modelo é incapaz de aprender o suficiente com os dados de treinamento e generalizar dados não vistos, gerando uma alta taxa de erro tanto no conjunto de treinamento quanto no conjunto de teste, produzindo previsões não confiáveis. Enquanto de *overfitting* quando o modelo tem previsões otimizadas no conjunto de treinamento se ajustando muito bem e não consegue prever com precisão dados não vistos no conjunto de teste.

O viés (ou *bias*, do inglês) termo introduzido no aprendizado de máquina em MITCHELL (1980) e, mais recentemente em GOODFELLOW; BENGIO; COURVILLE (2016), refere-se a diferença média da distância entre os valores estimados pelo modelo e os valores esperados no conjunto de treinamento. A variância (ou *variance*, do inglês), em contrapartida mede a variação dos erros quando se alterna entre o conjunto de treinamento e teste. Modelos de aprendizado de máquina desejáveis possuem baixo viés e baixa variância, entretanto, a medida que estes dois aspectos aumentam, as previsões perdem precisão. Assim, um dado modelo com baixo viés mas alta variância indica um *overfitting* nos dados de treinamento e não generaliza bem com novos dados. Por outro lado, um modelo com alto viés mas baixa variância indica um *underfitting*. Por fim, o modelo com alto viés e alta variância é em média impreciso.

4.1.2 Aprendizado Não-Supervisionado

Embora a abordagem anterior possua uma capacidade de inferir resultados de forma bem sucedida, nem todos problemas podem ser solucionados por meio dela. Em alguns casos, conseguir as variáveis bem definidas com a sua saída correspondente é extremamente custoso ou até mesmo inviável. Então, para cenários onde não existem resultados pré-definidos para o modelo utilizar como referência para aprender, é utilizado o aprendizado não-supervisionado.

Nesta abordagem não há uma referência ou saída para o modelo seguir. A forma mais comum é apresentar ao modelo um conjunto de variáveis com informações diversas sobre o objeto alvo, por exemplo sobre um vídeo (taxa de quadros, proporção, resolução, taxa de bits, formato do arquivo e etc), e pedir ao algoritmo que separe esses dados em uma certa quantidade de categorias. Essa técnica é chamada de *clustering*. Para BONACCORSO (2017), "o modelo aprende diretamente dos dados, tentando identificar todas as variáveis em comum para uma categoria (*cluster*) que seja capaz de associar novas amostras ao *cluster* correto".

Todas as técnicas dessa abordagem buscam encontrar uma representação mais definida dos dados que são apresentados, geralmente sendo essa representação mais simples, condensando as variáveis mais relevantes. É importante ressaltar que, neste caso, por não existir uma referência para o modelo avaliar seu desempenho (ou erro), cabe ao pesquisador (responsável) por analisar os dados resultantes. No caso de *clustering* o pesquisador deve definir se as categorias criadas pelo algoritmo fazem sentido, dados os grupos de variáveis agrupadas, e, por fim, avaliar se o modelo é satisfatório. Outros exemplos de algoritmos de aprendizado não-supervisionos, bem comuns, são: Rede de Crenças Profundas (*Deep belief network*) (HUA; GUO; ZHAO, 2015) e Rede Adversária Generativa (GANs) (CRESWELL et al., 2018).

4.1.3 Aprendizado Por Reforço

As abordagens supervisionadas e o aprendizado por reforço, possuem um ciclo de aprendizagem apresentando certa similaridade. Enquanto a primeira possui um *feedback* sobre a medida de erro através da classe atribuída àquela amostra, o aprendizado por reforço também se baseia em *feedback*, mas este é fornecido pelo ambiente. Nesta abordagem, existem componentes responsáveis por criar este ciclo de aprendizagem, que são eles: agente, ambiente, ação, recompensa e estado.

De forma geral, um agente procura atingir um objetivo determinado usando, a princípio, uma técnica de tentativa e erro. A seguir, os dados obtidos, independente de seu resultado positivo ou negativo, são a entrada para treinar o modelo por meio de *feedback* ou recompensa. Então as ações são julgadas pelos resultados que elas produziram. O aprendizado por reforço se caracteriza por ser uma abordagem que evolui com base em experiências anteriores, onde as ações de um agente para o ambiente

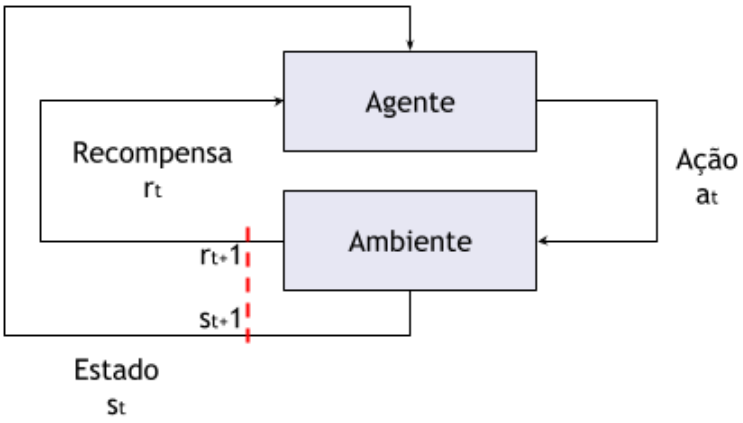

Figura 12 – Aprendizado por Reforço

dada uma situação ou estado especifico, retornam seu novo estado com base na recompensa resultante da ação no estado anterior, inferindo seu sucesso ou fracasso (WIERING; OTTERLO, 2012).

Essa abordagem é comumente utilizada para desenvolvimento de aplicações autônomas (drones, automóveis, etc) ou *players* (máquinas que jogam determinados jogos, por exemplo Xadrez) e, dentre os algoritmos mais conhecidos, estão o Monte Carlo (HOROWITZ, 1991) e *Q-learning* (WIERING; VAN OTTERLO, 2012).

4.2 Modelos de Aprendizado Supervisionado

Como o Aprendizado Supervisionado será o foco deste trabalho, seus principais modelos serão apresentados em mais detalhes nessa seção. Diferentes modelos analisam os dados de formas diversas, mas no geral o objetivo é sempre o mesmo.

4.2.1 Árvores de Decisão

Para compreender o funcionamento de um modelo de árvore de decisão (DT - *Decision Tree*) deve-se, primeiramente, entender a estrutura de dados de árvores binárias, que é o formato base das DT.

Na estrutura de dados de uma árvore binária, os dados são separados em nós que compõem os níveis da árvore ou profundidade. O nível zero representa a raiz onde toda a árvore será apoiada, isto é, toda a árvore vai se estruturar a partir dele. Os nós que partem da raiz são sempre divididos em dois e são nomeados de nós filhos. Eventualmente, estes se tornam nós-pai quando novos nós partem deles. Essa divisão acontece sucessivamente, até que os nós que não possuem nós-filhos são chamados de nós-folha.

Árvores de decisão é um dos métodos de aprendizado de máquina simbólico mais amplamente utilizado para inferências indutivas, ou seja, supervisionado e não paramétricos e, a priori, utilizados para problemas de classificação, embora possam ser

ajustados para funcionarem também em regressão. Dentre suas implementações, o algoritmo mais extensamente estudado é o ID3 (QUINLAN, 1983), que possui um viés indutivo de dar preferência por árvores menores. Quanto menor for a árvore, melhor será a indução do algoritmo, do contrário, é presumível que ocorra um *overfitting* no modelo (um modelo super especializado e não generalista). Ainda vale ressaltar outros algoritmos muito relevantes como C4.5 (QUINLAN, 2014) e o CART (BREIMAN et al., 2017).

Uma característica que simplifica bastante a representação do modelo é a capacidade das árvores de decisão serem representadas como conjuntos de regras SE-ENTÃO, não sendo obrigatória a construção de uma estrutura de dados.

Um exemplo de árvore de decisão está apresentado na Figura 13. A representação SE-ENTÃO para que a "Decisão 1" seja **Positiva** consiste no seguinte conjunto de regras: **SE** ((Atributo 1 = a **E** Atributo 2 = b) **OU** (Atributo 1 = b **E** Atributo 3 = b)) **ENTÃO** Decisão 1 = Positiva.

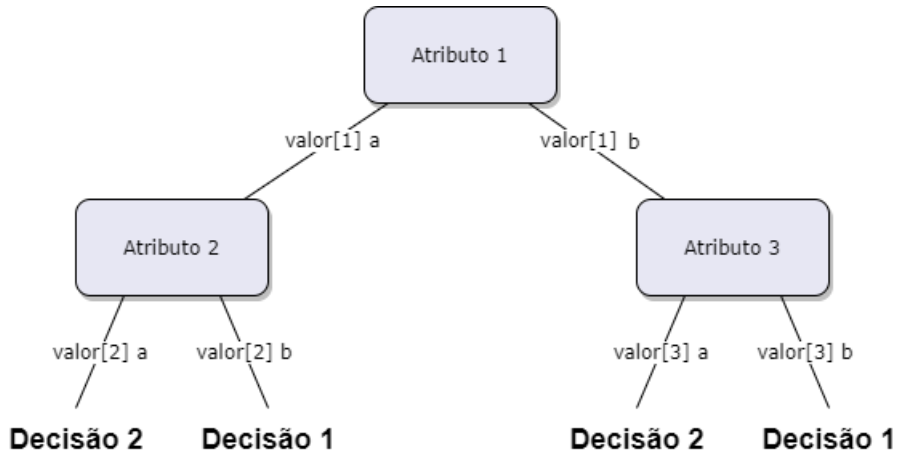

Figura 13 – Representação genérica de Árvore de Decisão Abstrata

Os atributos que incorporam as árvores de decisão, não necessariamente carecem de valores qualitativos, pois também são aceitos valores quantitativos e, na prática, a estrutura de dados continuará a mesma, mas com a adição de operadores relacionais: *menor que, maior que, igual a, diferente de, menor e igual a, maior e igual a*. A Figura 14 apresenta a representação de uma DT com atributos quantitativos.

A fim de compreender um algoritmo que implemente o modelo de árvore de decisão, será brevemente apresentado o já mencionado ID3 por se tratar de um algoritmo bastante difundindo nesta área e, de fácil compreensão.

4.2.1.1 Algoritmo ID3

O algoritmo ID3 foi desenvolvido por Ross Quinlan em 1983 (QUINLAN, 1983). A ideia fundamental é construir a DT empregando uma estratégia *top-down*, de cima

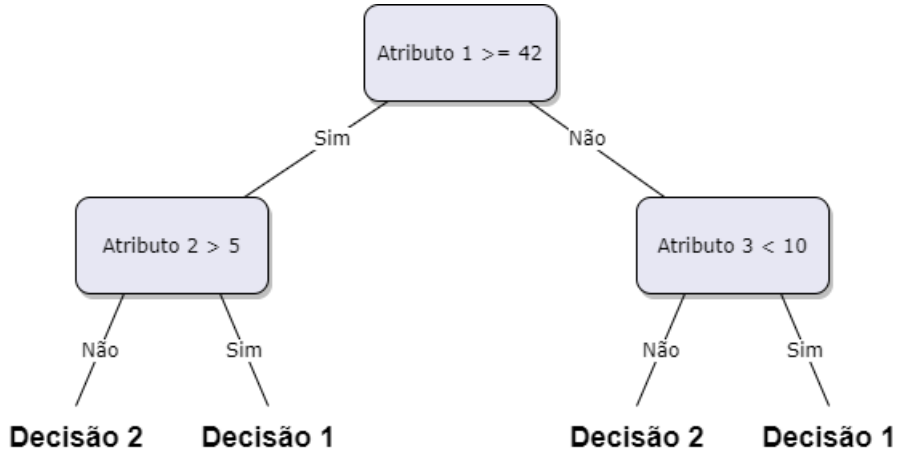

Figura 14 – Representação genérica de Árvore de Decisão Quantitativa

para baixo. Iniciando pela raiz, é feita uma busca entre os atributos do conjunto indicado para teste buscando encontrar qual atributo ocupará este nó com base no ganho de informação, que será apresentado a seguir.

Mas antes é necessário compreender o conceito de **Entropia** neste cenário. Entropia, por definição, caracteriza a medida de aleatoriedade de uma coleção de variáveis. Assim, a entropia E dada uma coleção S composta por amostras positivas e negativas de algum problema é calculada de acordo com a Equação 9. A função de entropia varia de 0 a 1 e, em árvores de decisão, sendo p é compreendido como a probabilidade da classe. Nesse caso, p_+ e p_- indicam a proporção de exemplos positivos e negativos, respectivamente, na coleção S. Esta função compreende uma classificação binária, mas ainda é possível generalizar esta definição para uma classificação multiclasses como sendo a somatória da proporção de S pertencendo a classe i, conforme a Equação 10.

$$E(S) = -p_+ \times log_2 p_+ - p_- \times log_2 p_- \tag{9}$$

$$E(S) \equiv \sum_{i=1}^{c} -p_i log_2 p_i \tag{10}$$

Dessa forma, árvores de decisão são tanto capazes de serem utilizadas com problemas de classes binárias, bem como com problemas multiclasses (n-classe). Com o valor da entropia E para o modelo, é possível calcular o ganho de informação, que é essencial para a definição dos atributos mais importantes no conjunto.

O ganho de informação é usado para construir a DT buscando sempre diminuir a entropia, diminuindo a aleatoriedade e evitando que o modelo fique enviesado à alguma classe. Assim, utiliza-se o calculo de ganho de informação para avaliar a redução na entropia causada pela partição das amostras do conjunto de acordo com

um atributo alvo. Assim, deve-se calcular o ganho de informação levando em consideração a entropia geral do conjunto S e a entropia do atributo alvo A, conforme a Equação 11.

$$Ganho(S, A) \equiv E(S) - \sum_{v \epsilon Valores(A)} \frac{|S_v|}{|S|} E(S_v) \tag{11}$$

A Tabela 8 será usada para exemplificar o cálculo do ganho de informação sobre um atributo alvo. Nesse caso, o conjunto **S** possui nove amostras no total, com cinco exemplos que fazem parte da classe ("Sim"na coluna Classe) e quatro que não fazem ("Não"na coluna Classe), ou seja, $S[5+, 4-]$. Considerando o **atributo 4** como exemplo, é possível perceber que quando seu valor é igual a **a**, então quatro amostras fazem parte da classe e duas não fazem, ou seja, $S_{4a}[4+, 2-]$. Por outro lado, quando o valor do **atributo 4** é igual a **b**, então uma amostra faz parte da classe e duas não fazem parte, ou seja $S_{4b}[1+, 2-]$. Com estas informações, é possível calcular o ganho de informação do **atributo 4** usando a Equação 11 e a Equação 10. Nesse caso, $E(S) = E(5+, 4-) = -5/9 \times log_2(5/9) - 4/9 \times log_2(5/9) = 0,991$. Para o valor **a** do **atributo 4**, $|S_v| = 6$ e $|S| = 9$, enquanto que $E(S_v) = E(4+, 2-) = -4/6 \times log_2(4/6) - 2/6 \times log_2(2/6) = 0,918$. Para o valor **b** do **atributo 4**, $|S_v| = 3$, e $|S| = 9$, enquanto que $E(S_v) = E(1+, 2-) = -1/3 \times log_2(1/3) - 2/3 \times log_2(2/3) = 0,918$. Por coincidência, neste exemplo $E(4+, 2-) = E(1+, 2-) = 0,918$. Finalizando o cálculo, $Ganho = 0,991 - (6/9) \times 0,918 - (3/9) \times 0,918 = 0,073$.

Tabela 8 – Árvore de Decisão - conjunto de exemplo

id	atributo 1	atributo 2	atributo 3	**atributo 4**	classe
01	a	a	a	a	Não
02	a	a	a	b	Não
03	b	a	a	a	Sim
04	c	b	a	a	Sim
05	c	c	b	a	Sim
06	c	c	b	b	Não
07	b	c	b	b	Sim
08	a	b	a	a	Não
09	a	b	a	a	Sim

Como a raiz da DT ainda não foi atribuída, este cálculo deverá ser realizado para todos os atributos do conjunto e, então, deve ser escolhido o atributo que apresentar o maior ganho de informação para ser a raiz da árvore. A partir do momento que a raiz da árvore estiver atribuída, deve-se distribuir seus ramos com base nos valores do atributo raiz e criar um novo nó com subconjunto de amostras que contém apenas

o valor correspondente à este ramo.

Assim, para cada novo nó considerando apenas o seu subconjunto, o cálculo de ganho de informação é repetido para todos atributos remanescentes escolhendo o melhor atributo para este nó.

Ao atribuir à um nó da arvore um atributo, isto é, incorporar este atributo na estrutura da árvore, este atributo não estará mais disponível para ser usado nos demais nós e não terá o ganho de informação calculado. Assim, todo e qualquer atributo incorporado aparece somente uma vez ao longo de qualquer caminho na árvore. A árvore é construída de forma exaustiva até que uma de duas condições seja satisfeita: (i) todos os atributos já estejam incorporados na árvore (ii) as amostras de treino associadas com este nó-folha tenham todos a mesma classe (DECISION TREES, 2005).

4.3 Métricas de Avaliação

BONACCORSO (2017) comenta que durante o processo de criação de um modelo de aprendizado de máquina, é essencial a utilização de métricas apropriadas para medir sua qualidade para cada problema. CARBONELL; MICHALSKI; MITCHELL (1983) comenta que a definição errônea das métricas será prejudicial para julgar se o modelo de fato está atendendo a seu propósito. Ao escolher as métricas, é crucial considerar a proporção de dados do conjunto e o objetivo desejado da previsão, seja ela binário, probabilidade, *ranking*, multiclasse e etc. Portanto, tão importante quanto escolher um modelo adequado, é compreender bem a métrica que será utilizada.

4.3.1 Classificação

Para modelos preditivos de classificação o objetivo é decidir, para uma nova observação, a qual classe dentre duas possíveis essa nova entrada pertence. As métricas mais utilizadas para avaliar a correção da decisão são: Matriz de Confusão, Acurácia, Precisão, Sensibilidade, *Fscore* e Curva AUC-ROC. Vale ressaltar que estas métricas são usadas, no geral, para classes binárias, mas com nenhum ou pouco ajuste, todas podem ser igualmente válidas para problemas multiclasse.

Classes de dados preditivos são valores conceituais utilizados praticamente em todas as métricas de avaliação de problemas de classificação. Os valores possíveis são: (i) Verdadeiros Positivos (**VP**), quando os exemplos são positivos e o modelo previu como positivos; (ii) Verdadeiros Negativos (**VN**), quando os exemplos são negativos e o modelo previu como negativos; (iii) Falso Positivos (**FP**), quando os exemplos são negativos e o modelo previu como positivos; e (iv) Falso Negativo (**FN**), quando os exemplos são positivos e o modelo previu como negativos.

4.3.1.1 *Matriz de Confusão*

Matriz de Confusão é uma ferramenta muito utilizada para visualizar as previsões realizadas pelo modelo, pois através dela é possível extrair as classes de dados preditivos VP, VN, FP, FN resultantes do modelo treinado, conforme ilustrado na Figura 15 A (SHOBHA; RANGASWAMY, 2018).

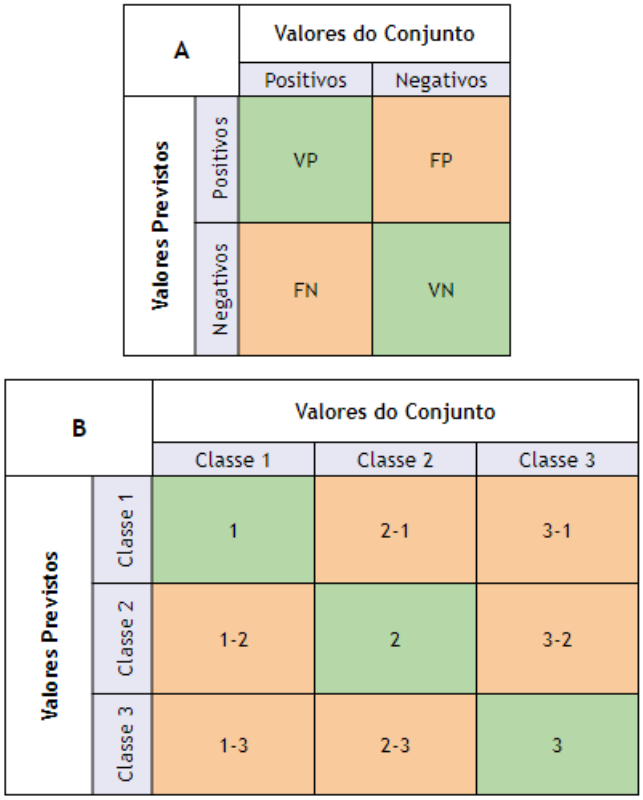

Figura 15 – Matriz de Confusão Genérica (Binária e Multiclasse)

A matriz de confusão também pode ser utilizada em problemas multiclasse, onde, ao contrário de cruzar os dados Positivos e Negativos, serão alinhadas as N-classes deste problema, como apresentado na Figura 15 B.

4.3.1.2 *Acurácia*

A métrica acurácia é a média comum de acertos do modelo ao classificar as classes. Também pode ser considerada a fração de predições corretas do modelo podendo ser calculada como definido na Equação 12, através do calculo da razão entre predições Verdadeiras Positivas e Negativas somadas, e todas as classes de dados preditivos somadas.

$$acuracia = \frac{VP + VN}{\sum classes} \tag{12}$$

Quando essa métrica é utilizada, é importante considerar o balanceamento dos dados, isto é, se cada classe do modelo possui o mesmo percentual de exemplos no conjunto. Se os dados estiverem desbalanceados, uma classe majoritária poderá atingir um resultado de acurácia alto por representar a maioria dos exemplos previstos. Por outro lado, os exemplos de classes minoritárias por mais que tenham sido classificados de forma errada, não afetarão a acurácia do modelo, pois seus valores tendem a ser suprimidos pela classe majoritária.

Quando os dados estão desbalanceados, então o modelo possui um viés para uma classe. Isto pode não ser desejado, dependendo do problema alvo. Nesse caso, será necessário analisar outras métricas ou aplicar técnicas de balanceamento de classes ou pesos (SHOBHA; RANGASWAMY, 2018).

4.3.1.3 Precisão e Sensibilidade

Enquanto a acurácia trata os valores globais, a precisão é dedicada a medir quanto o modelo é preciso em classificar os verdadeiros positivos, ou seja, de todas amostras do conjunto que o modelo fez a previsão como Positivos, quantos, de fato, estavam certos. A Equação 13 define o cálculo desta métrica e a Figura 16 a apresenta os elementos da matriz de confusão usados neste cálculo.

$$precisao = \frac{VP}{VP + FP} \tag{13}$$

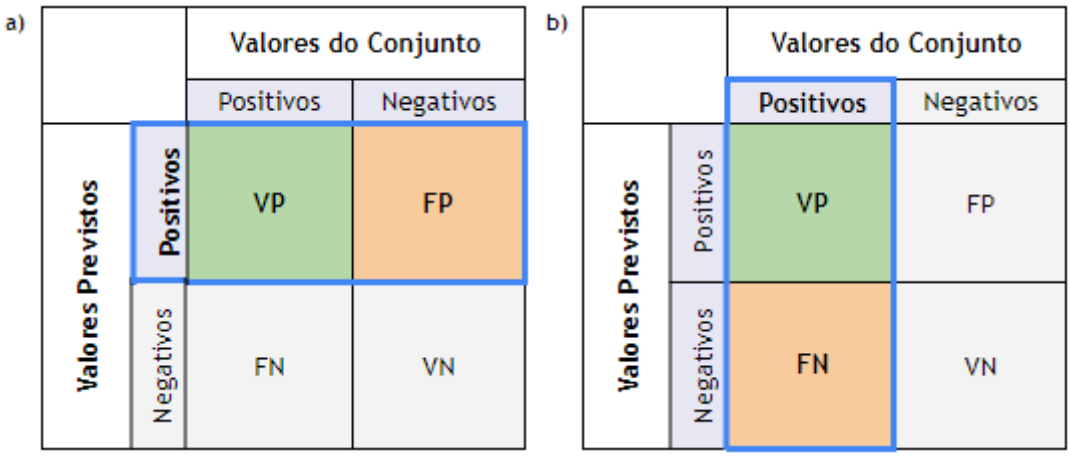

Figura 16 – Matriz de Confusão para o cálculo de Precisão e Sensibilidade

A sensibilidade é uma métrica utilizada para problemas onde diminuir os Falsos Negativos é o mais importante e avalia a proporção de Positivos identificados corretamente pelo modelo. A Equação 14 define o cálculo da sensibilidade e a Figura 16 b

apresenta os elementos da matriz de confusão usados neste cálculo.

$$sensibilidade = \frac{VP}{VP + FN} \tag{14}$$

É importante notar que, as métricas precisão e sensibilidade, não fornecem qualquer informação sobre o número de verdadeiros negativos (VN), ou seja, pode o modelo de aprendizado de máquina ter uma precisão ou sensibilidade baixos e, ainda, ter um alto número de acertos (SHOBHA; RANGASWAMY, 2018).

4.3.1.4 Fscore ou F1

A métrica Fscore ou F1 é utilizada quando o mais importante para o problema em questão é identificar o equilíbrio entre a precisão e a sensibilidade. O F1 é definido na Equação 15 e é calculado como a média harmônica ponderada destas medidas (SHOBHA; RANGASWAMY, 2018).

$$Fscore = 2/(\frac{1}{sensibilidade}) + (\frac{1}{precisao}) \tag{15}$$

4.3.1.5 Curva AUC-ROC

A curva AUC-ROC, ou simplesmente AUROC (Área sob as Características de Operação do Receptor), tem a capacidade de mostrar o quanto o modelo é capaz de distinguir as diferentes classes do conjunto. Onde ROC é uma curva da Taxa de Verdadeiros Positivos (TPR) em função da Taxa de Falsos Positivos (FPR), e AUC representa a medida de separação dos valores ou a área formada abaixo da curva responsável por quantificar o desempenho da curva ROC (KHAN; ALI RANA, 2019). O gráfico da curva ROC apresenta os valores no plano cartesiano de FPR no eixo X, e valores de TPR no eixo Y. De forma geral, à medida que o valor de AUC aumenta, melhor o modelo é em classificar cada classe distinta. Os valores da curva AUC-ROC estão sempre entre 0 e 1 (PRATI; BATISTA; MONARD, 2008). A Figura 17 demonstra como seria a representação gráfica desta curva para um problema binário onde é apresentado um classificador perfeito (linha vermelha), ou seja, quando o modelo treinado é capaz de identificar todas novas observações com máximo de acurácia (entretanto é improvável alcançar estes valores). A Figura 17 também mostra um classificador aleatório (ou linha base, que está tracejada). Por fim, a Figura 17 mostra a curva para um classificador ideal (linha azul), partindo de 0 previsões e convergindo até 1 das taxas de TPR e FPR. Esse tipo de resultado também improvável de ocorrer, embora seja o desejado na maioria dos casos quando um modelo esta sendo treinado.

Na prática, o modelo correto deveria imprimir uma curva semelhante à curva do classificador ideal da Figura 17. A área abaixo desta curva, ou AUC, determina quão eficiente o modelo está sendo em distinguir as classes. De forma geral, é esperado

que a Curva ROC esteja acima da linha base, pois equivale a um modelo melhor do que a aleatoriedade das previsões.

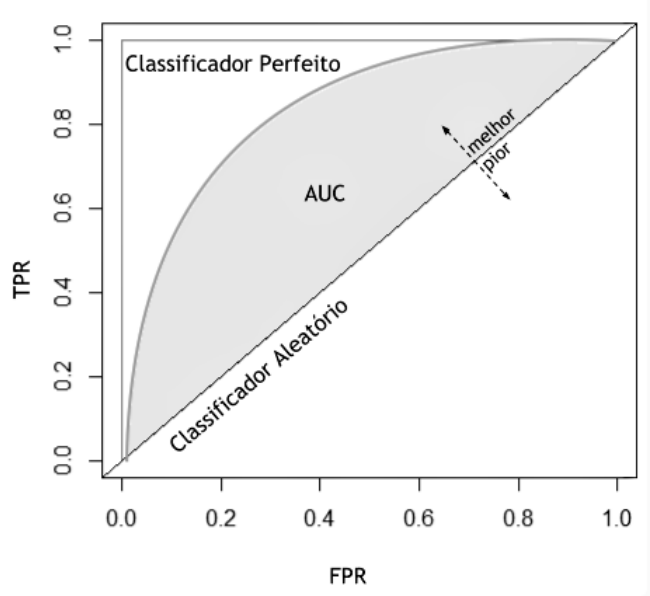

Figura 17 – Representação da Curva AUC-ROC Perfeita, Ideal e Aleatória

4.3.2 Regressão Linear

Antes de abordar o algoritmo de regressão linear, é fundamental compreender os conceitos de correlação. Correlação é o estudo do grau de associação entre variáveis e indica a variação simultânea ou independente de duas ou mais variáveis.

A Correlação Simples é o estudo do grau de relação entre duas variáveis, uma dependente, representada por Yi, e outra variável independente Xi. Por outro lado, a Correlação Múltipla é o estudo do grau de relação simultânea entre a variável dependente e duas ou mais variáveis independentes (OLIVEIRA RIBEIRO, 2009).

O coeficiente de correlação de Pearson (r) mede a associação entre duas variáveis contínuas, isto é, o grau de relacionamento entre Yi e Xi, indicando a força e direção do relacionamento linear (não significa, a priori, que uma é causa da outra). O intervalo de valores pode variar de -1 a 1 e existem três tipos de correlação de Pearson, conforme apresentado na Figura 18. Um valor de $r = 0$, ou correlação nula, indica que não existe associação entre as duas variáveis. Um valor de $r > 0$ indica a correlação positiva, ou seja, existe um relacionamento linear e direto entre as variáveis. Nesse caso, na medida que o valor de uma variável aumenta ou diminui, o mesmo ocorrerá com o valor da outra variável. Por fim, um valor de $r < 0$ indica a correlação negativa, isto é, quando o valor de uma variável aumenta a outra tende a diminuir e vice e versa, indicando que as variáveis estão inversamente relacionadas (OLIVEIRA RIBEIRO, 2009).

Uma correlação negativa perfeita ocorre quando valor de $r = -1$. Do mesmo

Figura 18 – Tipos de Correlação de Pearson

modo, se o valor de $r = 1$, então ocorre uma correlação positiva perfeita. Em ambos os casos, há um relacionamento linear perfeito. Entretanto, quando estes valores não são atingidos, é preciso analisar a força da correlação com base em uma escala, com r variando entre 0 e 1 para a correlação positiva e r variando entre 0 e -1 para a correlação negativa. Em ambos os casos, quanto mais distantes os valores estão de 0, mais forte é a correlação.

Modelos lineares em aprendizado de máquina são os métodos paramétricos mais simples, servindo de base para diversos outros modelos. Contudo, muitos problemas até mesmo a priori os não-lineares, podem ter facilmente uma solução com esses modelos. Possivelmente, o algoritmo mais utilizado deste modelo é a Regressão Linear.

A equação de regressão que melhor se ajusta aos dados estima o valor do coeficiente m da variável independente (isto é, a inclinação da reta em relação ao eixo horizontal) e o valor de b que representa o coeficiente de interceptação no eixo vertical para aproximar o valor da variável dependente. Assim, a regressão linear simples pode ser definida então pela Equação 16, onde x é a variável independente e y é a variável dependente.

$$y = mx + b \tag{16}$$

Como o propósito é encontrar as melhores estimativas para m e b, que reduzam os erros na predição da variável dependente a partir da variável independente, o algoritmo deverá calcular o erro padrão sobre a linha de regressão e analisar o coeficiente de determinação. Um exemplo de linha de regressão está apresentado na Figura 19.

Assim a Equação 17 inclui esse melhoramento, inserindo o erro padrão e, que representa a medida do valor médio em que a função superestimou ou subestimou o resultado. Enquanto, o coeficiente de determinação é uma medida estatística entre 0 e 1 que demonstra que distância os dados estão da linha de regressão, ou seja, quanto mais próximo de 1, mais ajustados os dados estão à linha de regressão e menor é o erro e, de acordo com (RODRIGUES, 2012) é igual ao quadrado do coeficiente de

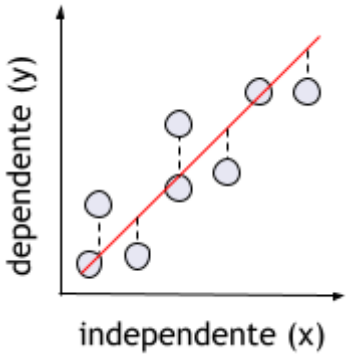

Figura 19 – Exemplo de linha de regressão para avaliar o erro padrão estimado

correlação de Pearson.

$$y = mx + b + e \tag{17}$$

Uma vez compreendendo as especificidades deste modelo, se torna clara a forma de utilizá-lo em problemas de aprendizado supervisionado de regressão onde o foco são variáveis contínuas.

4.4 Aprendizado por Agrupamento

O aprendizado por agrupamento (HUANG; XIE; XIAO, 2009) é baseado na técnica de combinar diversos modelos de aprendizagem. Assim, os vários modelos são capazes de aprender múltiplas suposições para o mesmo problema, usando conjuntos de treinamento diferentes ou múltiplos algoritmos mais simples (*weak learners*). Como resultado, é possível construir uma coleção de resultados que produza um modelo mais complexo que, empiricamente tende a produzir uma predição melhor do que cada um dos modelos separadamente. Isso é possível pois ao juntar algoritmos mais simples, as fraquezas de cada modelo são compensadas, diminuindo o viés e a variabilidade do modelo para um dado especifico (HUANG; XIE; XIAO, 2009). Isto é algo desejado, pois um modelo com viés e variância baixos e bem equilibrados tendem a apresentar resultados melhores e constantes.

Existem diversos métodos de aplicação de aprendizado por agrupamento, que, em sua maioria, têm o objetivo de construir um modelo robusto a partir de um único tipo de modelo simples agrupado várias vezes. Estes casos são chamados de modelos homogêneos. Por outro lado, se o agrupamento for formado por diferentes tipos de modelos, então o modelo resultante é chamado de heterogêneo (GASHLER; GIRAUD-CARRIER; MARTINEZ, 2008). Usualmente ambos os tipos de agrupamento são treinados sobre diferentes partes de um mesmo conjunto. Um exemplo de aplicação de agrupamento seria treinar separadamente diversas DT, que são modelos simples, de

modo independente para partes do conjunto de exemplos, de forma a diminuir sua suscetibilidade à variação nos dados e a variância. Assim, os erros aleatórios de cada modelo tendem a se cancelar, enquanto previsões corretas são reforçadas. Os dois principais métodos de agrupamento são *Bagging* e *Boosting* (OPITZ; MACLIN, 1999).

4.4.1 Método Bagging

Bootstrap aggregating, ou simplesmente *bagging* (BREIMAN, 1996), tem como objetivo produzir um modelo final que apresente uma maior resistência à variabilidade de dados. A Figura 20 exemplifica esta abordagem, onde modelos homogêneos de aprendizado simples e com alta variância são treinados em subconjuntos formados de forma aleatória a partir do conjunto de exemplos inicial (EFRON; TIBSHIRANI, 1993). Em seguida, os resultados são combinados por meio de algum processo de média das previsões individuais ou por votação, formando assim a predição do modelo final.

Uma característica positiva deste método é a geração de modelos menos propensos ao *overfitting*, mas que, em alguns casos, aumenta levemente o viés do modelo. Esta desvantagem se dá porque a técnica de *bagging* pode acabar selecionando múltiplas vezes a mesma amostra do conjunto durante o processo de construir o subconjunto de treino. Um exemplo bastante popular de modelo de agrupamento que utiliza o método de *bagging* é o modelo de floresta aleatória (*random forest* - RF). Nas RF, várias DT (denominadas de *shallow decision trees*) com alta variância, são treinadas sob os subconjuntos gerados pela técnica de *bagging* e combinadas que, por sua vez, reduzem a variância e constroem um modelo robusto (MOHANDOSS; SHI; SUO, 2021).

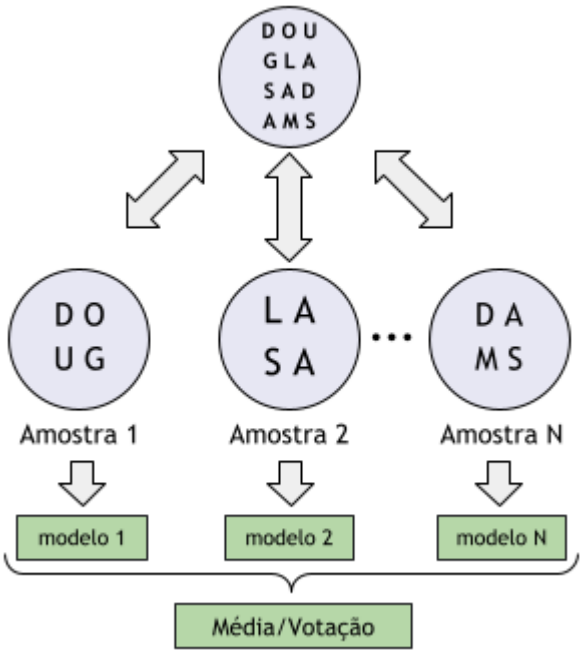

Figura 20 – Exemplo de Método *Bagging*

4.4.2 Método Boosting

O *boosting* segue o mesmo principio do método de *bagging*, combinando múltiplos modelos simples, para obter um modelo que tenha resultados melhores do que qualquer modelo simples individual. Porém, diferentemente do método anterior, que tem como objetivo a redução da variância, o *boosting* (SCHAPIRE, 2003) visa reduzir o viés do modelo. Não é uma regra, mas é mais apropriado utilizar modelos simples que possuam uma baixa variância e um alto viés neste método de agrupamento.

Enquanto no método *bagging* os modelos são treinados separadamente um dos outros, no método *boosting* cada modelo é treinado sequencialmente. Um exemplo de algoritmo que implementa este método, é o *Adaboost* (*Adaptive Boosting*) que, de acordo com SCHAPIRE (2003), consiste em iniciar os modelos todos com peso iguais, conforme são treinados seus pesos são atualizados, para as amostras classificadas incorretamente são aumentados os pesos, para que, o próximo *weak learner* se concentre nas amostras mais difíceis na próxima iteração, este processo de iteração que gera cada novo modelo simples será repetido até que um critério estabelecido no algoritmo seja alcançado (número de iterações, por exemplo).

De forma geral, o treino realizado de forma sequencial do método *boosting*, exemplificado na Figura 21, gera cada novo modelo sempre dando mais importância às amostras no conjunto de teste observados com maior dificuldade pelos modelos anteriores. Assim, o aprendizado é baseado nos erros passados, de modo a tornar a próxima predição mais resistente ao viés. Então, os pesos das amostras no conjunto de treino são atualizadas com base nas observações do modelo atual, para eleger predições com maior precisão.

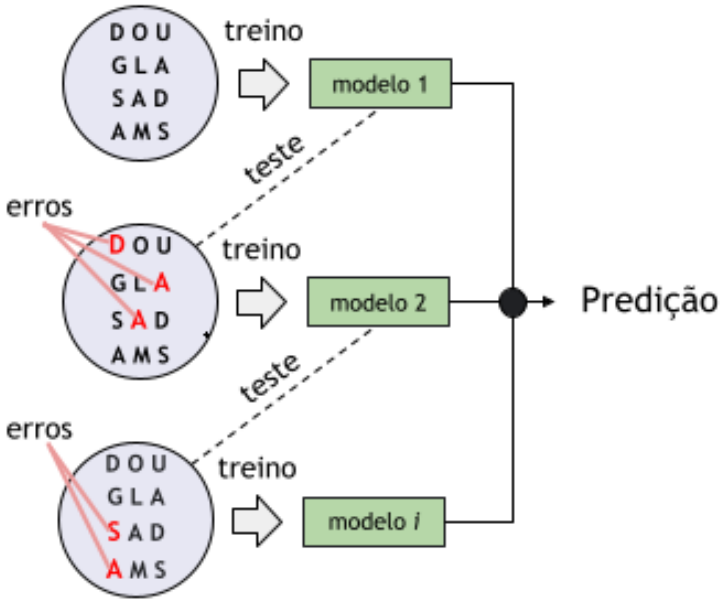

Figura 21 – Exemplo de Método *Boosting*

4.5 Modelos de Aprendizado de Máquina e a Decisão do Modo de Predição Intra-Quadro no AV1

Como visto no Capítulo 3, o maior desafio da predição intra-quadro do AV1 é a alta complexidade envolvida na tomada de decisão do modo preditor ideal, que consiste em avaliar todos modos preditores disponíveis e escolher um para cada bloco. Logo, dentro da tomada de decisão do modo preditor, os modelos de aprendizado de máquina podem, com base nas variáveis disponíveis neste processo, classificar quais modos preditores devem ser testados de forma prioritária, reduzindo o número de modos disponíveis para cada bloco. Diante disso, é possível traçar uma relação entre um problema de classificação do aprendizado de máquina com o problema existente na tomada de decisão do modo de predição intra-quadro no AV1.

Para explorar essa abordagem optou-se, neste trabalho, por avaliar os modelos de classificação árvores de decisão (Seção 4.2.1), pela sua construção simplificada, fácil compreensão dos resultados e por requerer o mínimo de preparação dos dados de entrada. Também foram avaliados modelos de floresta aleatórias (Seção 4.4), que compreendem uma coleção de árvores de decisão mas a natureza aleatória de sua construção minimiza o *overfitting* do modelo. Outros modelos mais complexos de aprendizado de máquina não foram avaliados, pois o seu uso poderia acarretar em um aumento do custo computacional do codificador AV1, que é exatamente o oposto do que se espera com esse trabalho.

5 METODOLOGIA

Neste capítulo serão apresentadas as condições comuns de teste utilizadas nos experimentos realizados neste trabalho com o codificador AV1, com base nas informações detalhadas que estão apresentadas no Apêndice B. Este Apêndice foi construído de acordo com as recomendações do documento mais recente do grupo de pesquisa *Internet Video Codec* (NETVC) da *Internet Engineering Task Force* (IETF).

Também são apresentadas, neste capítulo, as estratégias adotadas para o desenvolvimento dos modelos de aprendizado de máquina supervisionado que são foco desta tese.

5.1 Condições de Teste Utilizadas

Levando em consideração as recomendações apresentadas no Apêndice B e no documento (DAEDE; NORKIN; BRAILOVSKIY, 2020), foram usadas, neste trabalho, as seguintes configurações de teste para os experimentos realizados com o software de referência do AV1, o libaom (AOM, 2020):

1. A versão do libaom utilizada nos experimentos foi a 3.0 [1].

   O computador usado para realizar os experimentos com o software de referência tem um processador Ivy Bridge de 3.90 GHz com 4 núcleos e 8 *threads* e 16GB de memória DIMM CL9 de 1600MHz.

2. Os modos de predição intra-quadro direcionados à vídeos de conteúdo de tela (Pallete e IntraBC) não foram habilitados, pois as sequências de vídeo utilizadas não possuem esse tipo de informação.

3. Foram utilizadas apenas métricas de avaliação objetivas, justamente pela facilidade e pela confiabilidade de comparação entre resultados atingidos. As métricas objetivas primárias foram PSNR da camada de luminância (PSNR-Y) e taxa de bits (em bits por segundo - bps), além do tempo de codificação para

[1] hash e3cad234fe5fd69d7f1919579ff442056c9895ed/

cada configuração (em milissegundos). Foram comparadas a implementação de referência sem alterações e o codificador com o algoritmo desenvolvido.

4. A comparação e a interpretação dos resultados objetivos de qualidade foram realizadas com base na redução do tempo de codificação (RT) e BD-Rate, bem como, na relação entre estas medidas BD-Rate/RT, que mede para cada um por cento de redução de tempo de execução, o quanto se perde em eficiência de codificação.

5. Foram utilizadas oito sequências de vídeo por resolução, sendo duas resoluções: HD720 e HD1080. Estas resoluções foram definidas em função dos modelos de aprendizado de máquina construídos, como ficará claro na próxima seção. As sequências usadas para todo processo de aprendizado de máquina são do grupo de teste *objective-2-fast*, enquanto, os experimentos finais deste trabalho adiciona sequências do grupo de teste *objective-2-slow*, estes grupos estão definidos no Apêndice B. Estas sequências têm 8 bits por amostra e subamostragem de cor de 4:2:0.

6. Cada sequência foi codificada sob quatro valores de quantização (CQ): 20, 32, 43 e 55 (parâmetro <cq>), conforme definido nas condições comuns de teste apresentadas no Apêndice B.

7. Os parâmetros da configuração comum foram usados para os experimentos no software referência. Além disso, por conta dos objetivos deste trabalho, durante os experimentos o modo All-intra do libaom foi ativado, para tanto, como recomendado, o intervalo de *KeyFrame* foi definido como *kf-min-dist=1* e *kf-max-dist=1*. Estes parâmetros indicam que todos os quadros foram definidos como quadros intra. A seguir estão detalhados os parâmetros utilizados:

```
-codec=av1 -ivf  --verbose --psnr --frame-parallel=0
--tile-columns=0 --cpu-used=0 --threads=1 --end-usage=q
--cq-level=<cq> --kf-min-dist=1 --kf-max-dist=1
```

   A descrição de cada parâmetro utilizado, pode ser vista na Tabela 24 do Apêndice B.

5.2 Construção dos Modelos de Aprendizado de Máquina

Para diminuir o tempo de decisão na predição intra-quadro do codificador AV1, este trabalhou partiu de algumas suposições, onde a premissa foi a utilização de um modelo de aprendizado de máquina para acelerar a tomada de decisão do codificador AV1 na predição intra-quadro.

O modelo de aprendizado utilizado é um importante aspecto considerado, devendo necessariamente estar alinhado ao tipo de problema. Para tanto, o software referência do AV1 foi investigado e foi possível observar que a decisão na predição intra-quadro retorna, geralmente, um modo preditor (Capítulo 3), ou seja, um valor discreto (categórico). Assim, foi possível traçar uma relação da tomada de decisão da predição intra-quadro do AV1 com um problema de Classificação (Capítulo 4), onde as variáveis de entrada (X) deste modelo foram os dados extraídos dos experimentos a partir das condições de teste definidas anteriormente e o rótulo (y) é o modo escolhido.

Também foi definida uma linha de base (*baseline*) para avaliar o modelo, afim de determinar qual algoritmo seria a melhor escolha para o problema em questão. Para tanto, optou-se pelo algoritmo DT (visto na Seção 4.2) para *baseline* deste trabalho, por atender os seguintes critérios:

- Fácil interpretação, diminuindo a complexidade de sua implementação em outras ferramentas (tendo em vista a sua utilização no codificador AV1);
- Não é necessário valorar as variáveis manualmente, sendo possível escolher automaticamente as variáveis mais relevantes (priorizando que não ocorra nenhuma influência externa);
- Capacidade de trabalhar com dados numéricos e requerer o mínimo de preparação dos dados (considerando a possibilidade do vasto volume de dados extraídos dos experimentos);

Durante o desenvolvimento do modelo de aprendizado de máquina, uma série de etapas foram executadas: Aquisição de Dados, Processamento dos dados, Treino e Validação e Teste. Estas etapas serão detalhadas a seguir.

5.2.1 Aquisição de Dados

A etapa de aquisição de dados é a primeira etapa do aprendizado de máquina e é um momento fundamental para o resultado final, onde a quantidade e a qualidade das informações adquiridas determina a eficiência do preditor.

Usualmente, o ideal para construção de um modelo de aprendizado de máquina é utilizar uma base de dados (conjunto de amostras) que defina bem as classes do problema (rótulos), baseada nas variáveis extraídas de dados brutos e onde estes conjuntos sejam de domínio público e extensamente utilizados em outras pesquisas. Como mencionado na Capítulo 4, nem todo problema abordado possui um conjunto de amostras disponível de forma pública e este é o caso deste trabalho. Portanto, a primeira estratégia foi explorar o software de referência (AOM, 2020), no que compreende o fluxo da tomada de decisão do modo preditor intra-quadro no código-fonte, buscando uma função para extrair as amostras. Assim, identificou-se que a função

av1_rd_pick_intra_sby_mode é a responsável pela avaliação dos modos preditores intra. Este fluxo é apresentado em detalhes no Apêndice A.

Em seguida, investigou-se o algoritmo desta função do libaom, buscando identificar as variáveis ali utilizadas como entrada para a tomada de decisão do modo preditor, isto é, excluindo, assim, todas outras variáveis que não eram diretamente relacionas à decisão neste fluxo. Com isto, foram extraídas as variáveis relevantes, que serão tratadas como **variáveis** (ou *features*) nesta dissertação.

Foram identificadas 13 variáveis, seis destas variáveis foram denominados de entrada pelo sufixo I, referente a entrada de valores (*input*) na função, foram eles: I_bsize, I_best_rd, I_A, I_L, I_above_ctx e I_left_ctx. Além destas, também foram consideradas outras três variáveis auxiliares, que receberam o sufixo S: S_partition, S_uv_mode e S_mbmi_mode. A variável de saída (*output*) da função e recebeu o sufixo O: O_best_mode. Por fim, foi criada uma variável de marcação para indicar o CQ usado na codificação. A Tabela 9 descreve estas variáveis extraídas do código-fonte, com seu tipo e com uma breve descrição de sua funcionalidade.

Tabela 9 – Variáveis extraídas do fluxo de tomada de decisão do modo preditor intra-quadro do software libaom

Variáveis	Tipo	Descrição
cq	int	Identifica qual CQ está em uso
I_bsize	enum int	Número do tamanho de bloco atual
I_best_rd	int64	Custo RD passado para função
I_A	int	Número do modo do bloco logo acima
I_L	int	Número do modo do bloco logo à esquerda
I_above_ctx	const int	Valor de contexto acima
I_left_ctx	const int	Valor de contexto à esquerda
S_partition	int	Número do tipo de partição do bloco atual
S_uv_mode	int	Modo UV quando a predição intra-quadro é utilizada
S_mbmi_mode	enum int	Número do modo do bloco anterior
O_best_mode	enum int	Número do modo escolhido

5.2.2 Processamento dos Dados

Todos os procedimentos a seguir foram executados no ambiente online do Google *Colaboratory* (Google Colab)[2], que faz parte dos produtos do Google *Research* voltado para área de pesquisas cientificas. Este ambiente utiliza a linguagem de programação *Python*. Deste modo, a biblioteca de aprendizado de máquina de código aberto *scikit-learn* (sklearn) (PEDREGOSA et al., 2011) foi adotada, por ser implementada em *Python* e por ser extensamente difundida no meio de aprendizado de

[2]https://colab.research.google.com/

máquina. Além disso, essa biblioteca proporciona a maioria das ferramentas necessárias para o presente trabalho, como os principais algoritmos de classificação, métricas de avaliação e recursos para tratamento de dados.

Inicialmente, foram carregadas, no ambiente Google Colab, as amostras adquiridas na etapa anterior. Em seguida, estas amostras foram separadas em dois conjuntos, um conjunto para as sequências de resolução **HD1080** e outro para as sequências de resolução **HD720**, pois os modelos de aprendizado de máquina foram construídos de forma especializada por resolução. Cada um dos subconjuntos foram subdivididos novamente em quatro subconjuntos adicionais relacionado ao **CQ** utilizado.

O resumo dos conjuntos resultantes desta primeira etapa de pré-processamento estão apresentados na Tabela 10, onde o primeiro grupo de sequências são **HD1080** e o segundo são sequências **HD720**. Na coluna uso, *T_V* indica que a sequência foi utilizada para treino e validação do modelo, *Teste* indica que a sequência foi utilizada para o teste do modelo e, *R* indica que a sequência não foi utilizada na construção do modelo e, assim, foi reservada para os experimentos no codificador AV1 para avaliação final dos modelos desenvolvidos.

Tabela 10 – Número de amostras dos grupos de dados HD1080 e HD720 da primeira etapa de pré-processamento

Sequência	CQ_20 ($x10^6$)	CQ_32 ($x10^6$)	CQ_43 ($x10^6$)	CQ_55 ($x10^6$)	Uso
aspen	2,8	1,8	1,2	0,8	T_V
ducks_take_off	4,9	4,2	3,2	2,1	T_V
life	3,4	2,8	2,1	1,3	T_V
Netflix_Boat	4	3,6	3,1	2,3	T_V
Netflix_PierSeaside	2,4	1,89	1,3	0,92	T_V
Netflix_SquareAndTimelapse	3,9	3,3	2,6	1,8	T_V
rush_hour	2,8	1,6	0,95	0,6	T_V
seaplane_hdr_amazon	3,6	2,8	2	1,2	Teste
touchdown_pass	4,1	3,2	2,1	1,3	R
Netflix_Aerial	4,2	3,7	2,9	1,8	R
Netflix_FoodMarket	4,1	2,7	1,7	1	R
Netflix_TunnelFlag	3,2	2,5	1,9	1,4	R
boat_hdr_amazon	1,3	1	0, 73	0,47	T_V
dark	1,4	0,97	0,56	0,34	T_V
gipsrestat	1,3	1	0,8	0,58	T_V
Netflix_DrivingPOV	1,6	1,3	1,1	0,788	T_V
Netflix_RollerCoaster	1,4	1,1	0,813	0,537	T_V
vidyo1	1,2	0,880	0,612	0,414	T_V
vidyo4	1,2	0,844	0,57	0,384	T_V
KristenAndSara	1,1	0,81	0,6	0,43	Teste

A segunda etapa de pré-processamento foi a criação dos grupos de dados específicos, agregando e relacionando as sequências de uma determinada resolução com apenas um CQ nas amostras. Contudo, duas restrições foram aplicadas durante esta etapa: (i) foi definido o número de oito sequências para cada resolução (HD1080 e HD720) e (ii) uma sequência por resolução foi reservada para testar os modelos e não foram incluídas na fase de treino e validação (sequências *Teste* na Tabela 10). A primeira restrição é em função de existirem 12 sequências HD1080 e oito sequências HD720 no conjunto *objetive-2-fast*, então foram selecionadas aleatoriamente oito sequências do conjunto HD1080 e a sequências *R* na Tabela 10 não foram utilizadas nesta etapa de construção dos modelos de aprendizado de máquina. A segunda restrição é para separar as sequências usadas na construção do modelo daquelas usadas no teste deste modelo, o que é uma boa prática em aprendizado de máquina e evita *overfiting*. Assim, esta etapa resultou em oito conjunto de dados para treinamento, validação e teste. O número de amostras de cada um destes conjuntos de

dados, somando todas as sequências estão apresentados na Tabela 11, onde o título de cada grupo identifica a resolução e o CQ usado.

Tabela 11 – Número de amostras dos grupos de dados de treino, validação e teste da segunda etapa de pré-processamento

Título	Número de Amostras Extraídas
hd1080_cq20	24.555.701
hd1080_cq32	19.472.968
hd1080_cq43	14.710.369
hd1080_cq55	10.060.093
hd720_cq20	9.774.500
hd720_cq32	7.260.914
hd720_cq43	5.202.650
hd720_cq55	3.527.272

A terceira etapa foi caracterizada pelo tratamento dos dados das amostras de cada grupo de dados criado na etapa anterior. Todos os valores das variáveis são numéricos, não sendo necessário nenhum procedimento de transformação dos dados. Amostras duplicadas (ou ruído) aparecem em situações em que, os dados vêm de diferentes sequências de vídeo e, as variáveis extraídas destes são observadas mais de uma vez quando reunidos nos conjuntos de dados, sendo necessário excluir das duplicidades.

A variável que representa o modo preditor escolhido nas amostras extraídas é *O_best_mode*. Para facilitar a compreensão, os valores numéricos de *O_best_mode* associados a cada modo de predição estão apresentados na Tabela 12. Na Tabela 12 é possível identificar os modos DC, os modos direcionais vertical e horizontal (V e H), os demais modos direcionais (D45, D135, D113, D207 e D67), os três modos *Smooth* (vertical - V, horizontal - H e a média de ambos) e o modo *Paeth*.

Tabela 12 – Listagem de modos de predição intra-quadro de acordo com a representação numérica no libaom

Número	Modo de Predição
0	DC_PRED
1	V_PRED
2	H_PRED
3	D45_PRED
4	D135_PRED
5	D113_PRED
6	D157_PRED
7	D203_PRED
8	D67_PRED
9	SMOOTH_PRED
10	SMOOTH_V_PRED
11	SMOOTH_H_PRED
12	PAETH_PRED

Os modos IntraBC, *Palette*, CfL e *Recursive-based-filtering* não estavam presentes nas amostras. Os dois primeiros modos não estavam habilitados, como já discutido na seção anterior, pois as sequências de teste não eram de conteúdo de tela. Como apenas amostras de luminância foram usadas, é natural que o modo CfL não estivesse presente nas amostas. Por fim, o modo *Recursive-based-filtering* não esta presente nas amostras porque este modo é avaliado, no software referência, são tratados pelo codificado AV1 separadamente, por outras funções que não a função principal da definição dos modos da predição intra-quadro. Então os modelos desenvolvidos neste trabalho foram treinados sem considerar estes quatro modos.

A partir do tratamento dos dados das amostras, realizou-se o processo de **rotulagem das amostras** (isto é, definir a variável categórica y ou a classe) e uma estratégia foi abordada: agrupar os modos preditores intra-quadro do AV1 em grupos que categorizam as amostras a partir dos rótulos atribuídos a cada um dos grupos;

Três agrupamentos de modos foram criados: Grupo 1 (os modos direcionais), Grupo 2 (o modo DC) e Grupo 3 (modos *smooth* e o modo *paeth*). Dentre as diversas formas possíveis de agrupamento, decidiu-se por essa maneira levando em consideração a similaridade dos modos preditores intra-quadro no aspecto de como as amostras são tratadas no AV1, logo, o Grupo 1 é formado pelos oito modos direcionais pela natural semelhança entre eles. O Grupo 2 contêm exclusivamente o modo DC por diferir bastante de todos os demais modos e por ser muito usado. O grupo 3 é formado pelos modos restantes, que são os modos *smooth* e *paeth*. Por fim o grupo de dados foi alterado para incluir a coluna da variável rótulo com os valores dos grupos

criados, conforme apresentado na Tabela 13. Essa coluna foi nomeada de *O_group* e, por não mais ser necessária, a coluna da variável *O_best_mode* foi removida.

Tabela 13 – Rotulagem por Agrupamento de Modos Preditores

Classe	Agrupamento	Classes de modos
0	Grupo 1	1, 2, 3, 4, 5, 6, 7, 8
1	Grupo 2	0
2	Grupo 3	9, 10, 11, 12

Outra característica identificada nas amostras ao se analisar os grupos de dados, foi que os dados estavam desbalanceados, isto é, a presença de mais amostras de uma classe (classe majoritária) do que de outras (classe minoritária), o que poderia acarretar um viés no modelo treinado.

A fim de lidar com o desbalanceamento nas amostras, foi utilizada a biblioteca *Imbalanced-learn* (imblearn)[3], que é uma biblioteca em *Python* e de código aberto. Essa biblioteca depende do *scikit-learn* e fornece ferramentas para lidar com classes desbalanceadas. Dentre as ferramentas presentes, foram utilizadas aquelas voltadas para a reestruturação dos dados. Essas ferramentas manipulam a frequência de classes das amostras que são efetivamente apresentadas ao modelo, tentando igualar o número de amostras entre as classes. Duas abordagens são utilizadas: (i) *oversampling*: que cria novas amostras sintéticas da classe minoritária, objetivando igualar a proporção de classes; e (ii) *undersampling*: que consiste em reduzir as amostras da classe majoritária, também buscando igualar as amostras entre as classes no grupo de dados.

Tendo em vista o grande volume de amostras presentes em cada grupo de dados, conforme apresentado na Tabela 11, este trabalho usou a abordagem de *undersampling*, onde a biblioteca *imblearn* proporciona mais de 10 métodos para lidar com grupo de dados desbalanceados. Dentre esses, foram avaliados dois métodos: (i) *Random Under-sampling*: que é bastante simples e seleciona de forma aleatória as amostras da classe majoritária que devem ser excluídas do grupo de dados, até que uma frequência de classes mais equilibrada seja alcançada; e (ii) *NearMiss*: que também reduz as amostras da classe majoritária, porém, selecionando amostras com base na distância entre dados vizinhos ZHANG; MANI (2003).

5.2.3 Treinamento, Validação e Teste

Após a obtenção dos grupos de dados balanceados com distribuição de classes de igual tamanho, teve início as etapas de **Treinamento** e de **Validação**, onde foi realizada a busca por hiperparâmetros aplicando a técnica de validação cruzada, onde

[3] https://imbalanced-learn.org

o treino do modelo foi realizado com uma parte do grupo de dados e a validação foi realizadas com uma outra parte do grupo de dados, usando a métrica de avaliação *Fscore*.

Então foi realizada a etapa de **Teste**, onde um novo conjunto de amostras balanceadas, formado por amostras extraídas de sequências que não formaram o grupo de dados da etapa anterior, é submetido ao modelo validado. Deste teste são obtidas as métricas de avaliação finais do modelo em questão.

Este método de três etapas foi adotado para preservar um grupo de dados exclusivo para os testes finais do modelo e, assim, evitando o sobre ajuste dos modelos treinados. Então os modelos foram definidos com base no *Fscore* de um grupo de dados, e foram testados em um outro grupo de dados.

Na etapa de **Treinamento** cada um dos grupos de dados apresentados na Tabela 11 foi dividido em 80% para treinamento e os 20% restantes para teste do modelo. Destes 80%, ainda 20% foi utilizado para validação cruzada. É importante reafirmar que o grupo de dados reservado para teste não foi empregado nesta etapa.

Em seguida, para cada modelo avaliado, as amostras selecionadas (80% das presentes na Tabela 11) foram utilizadas para busca de hiperparâmetros usando o método de busca aleatória (*Random Search*)[4], baseado no estudo conduzido por BERGSTRA; BENGIO (2012). O método de busca aleatória busca algumas combinações de parâmetros sobre um conjunto maior definido em um espaço de busca limitado, onde o *Fscore* é obtido através de validação cruzada (*Cross-Validation*, CV), conforme definido na equação 18.

A técnica CV utilizada neste trabalho foi *K-folds* igual a cinco, uma vez que é fácil de interpretar seus resultados e essa técnica geralmente resulta em estimativas com menor viés ou estimativas menos otimistas do desempenho do modelo. A Figura 22 apresenta a técnica *K-folds* usada neste trabalho.

$$Fscore_{CV} = \frac{1}{5}\sum_{i=1}^{5} Fscore_i \tag{18}$$

[4]https://scikit-learn.org/stable/modules/generated/sklearn.model$_{s}election.RandomizedSearchCV.html$

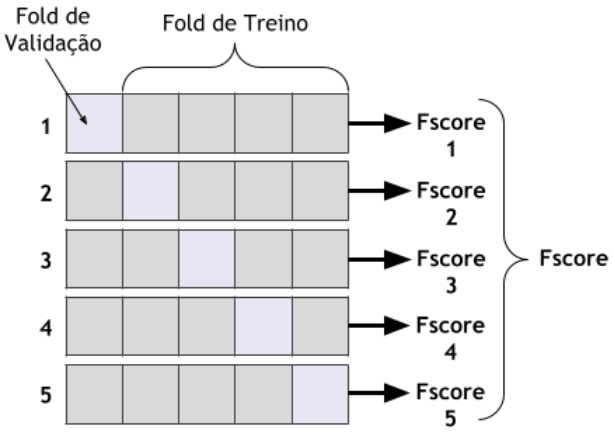

Figura 22 – Representação da técnica K-Fold aplicada

Com a definição dos melhores valores para os hiperparâmetros no espaço de busca determinado, foi treinado o modelo com os 80% das amostras do grupo de dados, reservando os 20% restantes para a validação, conforme já destacado. Os resultados foram avaliados com base no *Fscore*.

Por fim, a última etapa caracterizou-se pelo **Teste** do modelo validado na etapa anterior, usando o grupo de dados reservado para teste (não usado em nenhum momento nas etapas anteriores) para obter os resultados finais dos modelos.

6 SOLUÇÕES PARA REDUÇÃO DE COMPLEXIDADE NA DECISÃO DE MODO DA PREDIÇÃO INTRA-QUADRO DO AV1

O capítulo anterior demonstrou as metodologias que nortearam o desenvolvimento deste trabalho. Este capítulo irá apresentar os algoritmos propostos neste trabalho para acelerar o funcionamento do processo de tomada de decisão do modo de predição intra-quadro do codificador AV1.

Os algoritmos propostos buscaram alcançar maior rapidez na tomada de decisão dos modos preditores intra-quadro do codificador, atingindo um tempo de codificação inferior em comparação à implementação original do libaom e com a menor perda possível na eficiência de codificação.

Como já mencionado no capítulo anterior, os modos de predição intra-quadro tratados por esse trabalho são os modos *0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12*, ou seja, o modo DC, todos os modos direcionais, os três modos *Smooth* e o modo *Paeth*. Os demais modos intra (IntraBC, *Palette*, CfL e *Recursive-based-filtering*) não foram incluídos na solução proposta.

A ideia básica explorada nesse trabalho é reduzir o número de modos testados na predição intra-quadro com o intuito de reduzir o tempo de execução total. Como a grande maioria dos modos de predição encontram-se reunidos no mesmo fluxo de tomada de decisão, é razoável inferir que, diminuindo o número de modos avaliados irá acelerar esse processo.

Duas abordagens foram desenvolvidas: (i) redução de complexidade por agrupamento de modos e CQ específico (AME) e (ii), redução de complexidade por agrupamento de modos e CQ agrupados (AMA). A primeira abordagem tem menor redução de tempo de execução e menor impacto na eficiência de codificação e, enquanto a segunda é menos conservadora, visando maior redução de tempo de execução, mas, em contrapartida trazendo um maior impacto na eficiência de codificação. Estas duas abordagens utilizaram aprendizado de máquina supervisionado, onde foram desenvolvidos modelos preditivos de classificação.

O objetivo final desejado é que estes modelos sejam capazes de determinar, durante o processo de codificação e com o conjunto de variáveis disponíveis, a qual

rótulo, dentre os diversos possíveis, pertence cada entrada. O rótulo representa, no fluxo da tomada de decisão do codificador, qual modo de predição intra-quadro está disponível para busca. Então, se a classificação for bem sucedida, um número importante de modos deixarão de ser avaliados na predição intra-quadro, com impacto baixo na eficiência de codificação e com redução significativa no tempo de execução deste processo.

6.1 Redução de Complexidade por Agrupamento de Modos e CQ Específico (AME)

A primeira alternativa explorada para reduzir o número de modos preditores intra-quadro considera o rótulo atribuído como um agrupamento de modos de predição intra-quadro, como discutido no capítulo anterior. Assim, foram criados três grupos de modos de preditores intra-quadro (apresentados na Tabela 13). Nesta seção, serão apresentados os modelos de aprendizado de máquina treinados para os grupo de dados HD720 e HD1080 e para os quatro CQs (20, 32, 43 e 55). Assim, para cada técnica de aprendizado de máquina explorada usando a abordagem AME, um total de oito modelos foram treinados, validados e testados. A avaliação dos modelos foi feita a partir das métricas recomendadas na Seção 4.3. Por fim, é apresentada a comparação entre os modelos treinados com a abordagem AME, em comparação com um modelo *baseline* que também foi desenvolvido.

6.1.1 Avaliação do Modelo Preditivo Base

Este modelo preditivo de classificação serviu para determinar as métricas base (*baseline*) para o desenvolvimento desta proposta e é baseado em modelos de Árvores de Decisão (Seção 4.2). Sua implementação foi feita a partir do classificador Árvore de Decisão[1] da biblioteca *sklearn* (PEDREGOSA et al., 2011) sem nenhum ajuste de hiperparâmetros neste primeiro momento. Apenas foi definida a profundidade máxima da árvore (10) e especificado o parâmetro de *random_state* para possibilitar a reprodução dos resultados.

O modelo *baseline* serviu, também, para a compreensão da melhor abordagem para balanceamento de classes no grupo de dados, através dos métodos de *Random Under-sampling* e *NearMiss* da biblioteca *imblearn*[2], conforme já apresentado no capítulo anterior.

É notório que todo método para reduzir a frequência de amostras de classes acarreta em perda de informação. Assim, o que deve ser avaliado é o impacto dos diferentes métodos empregados. Então o modelo preditivo *baseline* foi treinado com

[1] https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
[2] https://imbalanced-learn.org/stable/references/under_sampling.html

um grupo de dados balanceado pelo método *Random Under-sampling* e, em seguida, treinado novamente usando o balanceamento *NearMiss*. Em ambos os casos, o rótulo (y) estava criado, representado por *O_group*. Ambos os métodos resultaram em um grupo de dados de igual tamanho e proporção de classes. A Figura 23(a) representa a distribuição original dos dados, enquanto que a Figura 23(b) apresenta o gráfico de classes (isto é, amostras por rótulo) para os grupo de dados HD720 e HD1080 resultante deste processo.

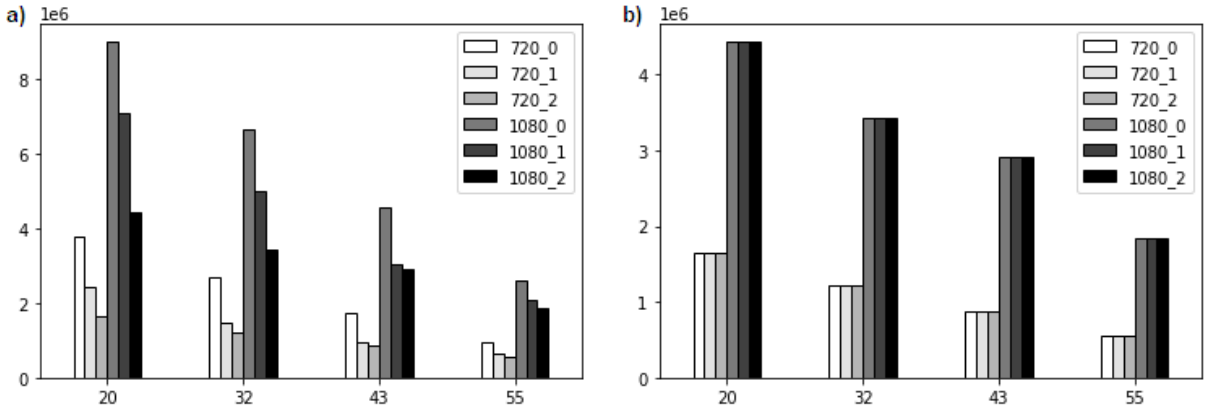

Figura 23 – Distribuição de Classes Original e Balanceadas - HD720 e HD1080

Foram então treinados os dois modelos *baseline* de árvore de decisão usando os dois métodos de balanceamento. As árvores tiveram uma profundidade máxima igual a 10 e nenhum ajuste de hiperparâmetros foi realizado, como já mencionado. Os modelos foram avaliados através das métricas de acurácia e Fscore, conforme apresentado na Tabela 14. Os resultados apresentados consideram a média dos quatro CQs investigados (20, 32,43 e 55).

Tabela 14 – Proposta AME - Avaliação de métricas dos modelos *baseline*

Dataset	Método	Acurácia	Fscore
HD720	*Random Under-sampling*	41,54	41,54
HD720	*NearMiss*	47,93	47,95
HD1080	*Random Under-sampling*	39,75	39,81
HD1080	*NearMiss*	45,02	45,00

Os resultados da Tabela 14 indicam que os desempenhos do método de *NearMiss* foram melhores em ambas as resoluções testadas. A acurácia do modelo foi ampliada em aproximadamente 6% (isto é, a média geral de acertos do modelo ao classificar as classes) com o uso de *NearMiss*. Além disso, o Fscore aumentou na mesma proporção. Lembrando que o Fscore é calculado através da média harmônica entre as métricas Precisão e Sensibilidade do modelo, onde tanto os falsos positivos e quanto os falsos negativos podem degradar o desempenho do modelo. Vale ressaltar que

os modelos preditivos de classificação treinados estão lidando com um problema de multiclasses, e não binário, onde é possível definir um limiar de total aleatoriedade aproximado de 33% considerando três possíveis classes (os agrupamentos de modos).

As Figuras 24 e 25 apresentam as matrizes de confusão dos modelos construídos usando os dois métodos de balanceamento avaliados para o conjunto de dados HD720 e HD1080, considerando o melhor valor de quantização recomendado CQ=20. Na Figura 24 é possível observar que o modelo de *baseline*, quando treinado com grupo de dados balanceado pelo método *NearMiss* obteve um aumento no acerto se comparado aos dados balanceados pelo método *Random Under-sampling*.

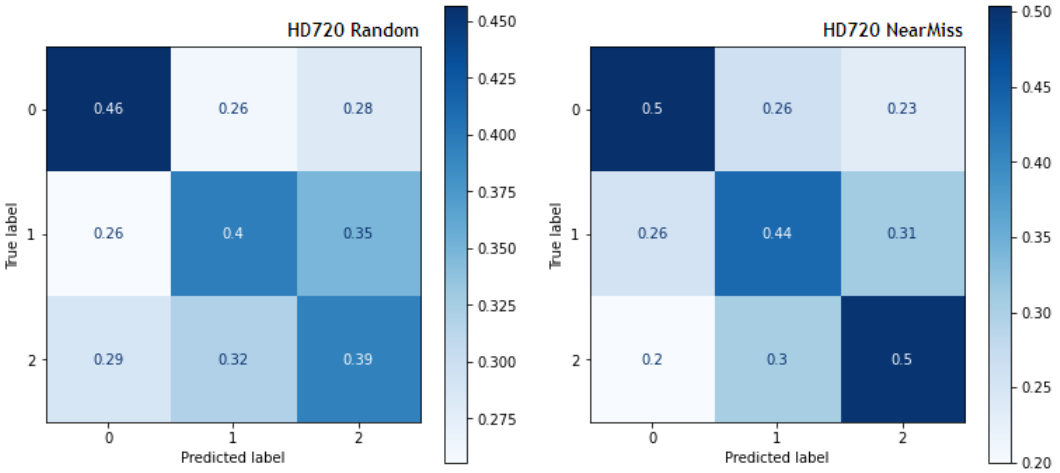

Figura 24 – Matriz de Confusão - modelo *baseline* HD720 com CQ=20

Na Figura 25, com as matrizes de confusão do grupo de dados **HD1080**, é possível observar que ocorre algo semelhante ao observado anteriormente no grupo de dados **HD720**, ou seja, o método *NearMiss* apresentou, para todas as classes, um aumento no acerto (embora mais discreto nesta resolução).

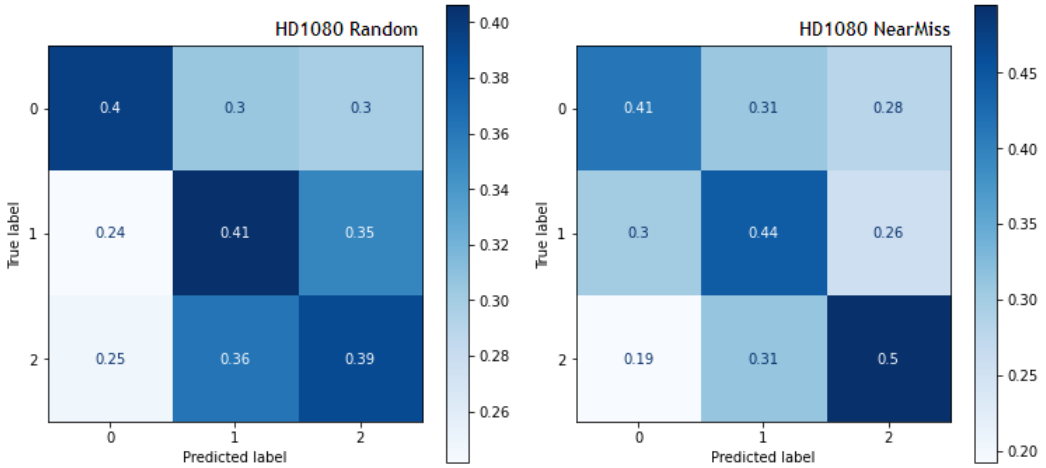

Figura 25 – Matriz de Confusão - modelo *baseline* HD1080 com CQ=20

Considerando os resultados apresentados, foi definido o *NearMiss* como método de balanceamento usado neste trabalho. Além disso, foram definidos os valores da métrica Fscore do modelo *baseline* que irão servir de base comparativa para os modelos preditivos de classificação definidos a seguir seguindo a abordagem AME.

6.1.2 Árvore de Decisão - Agrupamento de Modos com Ajuste de Hiperparâmetros

O primeiro modelo explorado em busca de superar os valores das métricas definidas como base foi o próprio modelo de árvore de decisão (DT), realizando nele o ajuste de hiperparâmetros (isto é, modificando os valores dos parâmetros que permitem controlar o processo de treinamento do modelo). Para tanto, foram usados os seguintes hiperparâmetros:

- **criterion**: Define a função para medir a qualidade de um particionamento em cada nó da árvore. Os critérios disponíveis são "gini"e "entropy";
- **splitter**: Define a estratégia utilizada para dividir o nó. Os valores possíveis são "best"e "random";
- **max_features**: Define o número máximo de variáveis a serem consideradas ao procurar qual variável gera o melhor particionamento de um nó de acordo com função de **criterion**. Este parâmetro aceita valores inteiros diretamente ou funções que são "sqrt", "log2"e "None"(este último representa o número total de variáveis presentes no grupo de dados);
- **max_leaf_nodes**: Define a quantidade máxima de nós folha que a árvore pode ter, aceitando valores inteiros;
- **max_depth**: Define a profundidade máxima da árvore. Por padrão não há uma limitação na profundidade. Este parâmetro aceita valores inteiros.

O espaço de busca para os hiperparâmetros é apresentado na Tabela 15.

Tabela 15 – Proposta AME - Espaço de Busca de Hiperparâmetros Árvore de Decisão com *Random Search*

Hiperparâmetro	Intervalo	Combinações
criterion	[entropy, gini]	2
splitter	[best, random]	2
max_features	[1 a 9 com passo 1] U [sqrt, log2, None]	12
max_leaf_nodes	[10 a 300 com passo 10]	30
max_depth	[3 a 9 com passo 1] U [10 a 50 com passo 10]	12

Neste espaço de busca de hiperparâmetros são possíveis 17.280 combinações distintas, o que torna o treinamento muito custoso. Deste modo, optou-se por aplicar o método de *Random Search* com limite de iterações igual a 200 e validação cruzada igual a cinco *folds* (CV=5), para limitar o número de modelos gerados e buscando prevenir erros de variância e sobreajuste no modelo treinado. Antes da execução do *Random Search*, o conjunto de amostras para o treinamento foi particionado pela técnica de *split* (seção 5.2.3), onde somente o subconjunto composto de 80% das amostras foi utilizado nesta etapa.

Definidos o espaço de busca de hiperparâmetros e aplicado o *split*, foi realizada a execução do *Random Search*, onde 1.000 modelos foram treinados com diferentes combinações de hiperparâmetros. A partir dessa execução, através do método de treino e teste de validação cruzada (CV) definido, a métrica *Fscore* foi calculada para cada combinação avaliada pelo *Random Search*, extraindo a melhor combinação dentro deste espaço de busca.

Ao final, cada modelo ajustado com hiperparâmetros selecionados foi validado no subconjunto composto por 20% das amostras de validação (que não constituíram o subconjunto de treino). Os resultados de Fscore dos modelos gerados com os melhores valores dos hiperparâmetros após a fase de busca estão apresentados na Tabela 16.

Tabela 16 – Proposta AME - Precisão do modelo Árvore de Decisão de Três Classes por CQ

Resolução	CQ	Fscore CV=5	Fscore Validação
HD720	20	60,61	60,01
HD720	32	59,23	58,64
HD720	43	57,43	56,85
HD720	55	57,86	57,28
Média		**58,79**	**58,20**
HD1080	20	58,02	57,44
HD1080	32	58,46	57,88
HD1080	43	55,66	55,10
HD1080	55	55,63	55,07
Média		**56,94**	**56,37**

Estes resultados serão discutidos, de forma comparativa, na Seção 6.1.4.

6.1.3 Floresta Aleatória - Agrupamento de Modos com Ajuste de Hiperparâmetros

O segundo modelo explorado seguiu conceito de aprendizado por agrupamento, que se baseia na ideia de combinar diversos modelos de predição mais simples, con-

forme discutido na Seção 4.4. Para tanto, foi utilizado o modelo de Floresta Aleatória (RF) que, como já discutido, integra uma combinação de diversas árvores de decisão. O modelo RF foi treinado com o método de *bagging*, cuja essência é a combinação dos modelos de aprendizado para otimizar o resultado final.

Nesta etapa, o ajuste de hiperparâmetros do modelo Floresta Aleatória foi realizado da mesma forma que na solução usando DT, ou seja, usando *Random Search* e validação cruzada. Mas como o modelo de Floresta Aleatória gera N árvores simples (de acordo pelo hiperparâmetro *n_estimators*), a busca pelos melhores hiperparâmetros é muito mais custosa do que no modelo de árvores de decisão. Assim, o número de iterações usado no *Random Search* foi reduzido para 100.

Foram considerados, nesta busca, os seguintes hiperparâmetros: *criterion*, *max_features*, *max_leaf_nodes* e, *max_depth*. Estes hiperparâmetros também estão presentes no modelo de Floresta Aleatória, pois fazem parte da construção das árvores de decisão e possuem igual função ao apresentado no modelo anterior. Além destes, foi considerado o hiperparâmetro *n_estimators*, que define quantas árvores são geradas dentro do modelo.

A Tabela 17 define o espaço de busca de hiperparâmetros para este modelo, onde é possível perceber que até 80.640 combinações distintas são possíveis. Entretanto, com o limite de iterações do *Random Search* definido como 100 e com validação cruzada igual a cinco, foram testados 500 modelos no subconjunto composto por 80% das amostras (isto é, o particionamento reservado para treino através da técnica de *split* conforme apresentado anteriormente).

Tabela 17 – Proposta AME - Espaço de Busca de Hiperparâmetros Floresta Aleatória com *Random Search*

Hiperparâmetro	Intervalo	Combinações
n_estimators	[10 a 110 com passo 10]	10
criterion	[entropy, gini]	2
max_features	[1 a 9 com passo 1] U [sqrt, log2, None]	12
max_leaf_nodes	[10 a 120 com passo 10]	12
max_depth	[3 a 30 com passo 1]	28

Desta execução, assim como realizado no primeiro modelo, avaliou-se cada combinação por meio da métrica de Fscore calculada a partir do método de validação cruzada definido. Por fim, extraiu-se o melhor ajuste de hiperparâmetros neste espaço de busca e validou-se cada modelo no subconjunto de validação (ou seja, usando os 20% de amostras que não estiveram presentes durante o treino). Os resultados dos melhores modelos após esta avaliação são apresentados na Tabela 18.

Tabela 18 – Proposta AME - Precisão do modelo de Floresta Aleatória de Três Classes por CQ

Resolução	CQ	Fscore CV=5	Fscore Validação
HD720	20	60,69	60,75
HD720	32	59,15	59,21
HD720	43	57,41	57,47
HD720	55	57,56	57,62
Média		**58,70**	**58,76**
HD1080	20	57,93	57,99
HD1080	32	58,42	58,48
HD1080	43	55,34	55,39
HD1080	55	55,38	55,43
Média		**56,76**	**56,82**

A próxima seção discute esses resultados de forma comparativa com o modelo usando DT com hiperparâmetros ajustados e também com o modelo *baseline*.

6.1.4 Comparação dos modelos treinados na abordagem AME

Como já apresentado, foram desenvolvidos três grupos de modelos usando a abordagem AME: (i) DT *baseline*, (ii) DT com ajuste de hiperparâmetros e (iii) RF com ajuste de hiperparâmetros. Cada grupo de modelos foi construído considerando oito grupo de dados (duas resoluções e quatro CQs). Assim, foram gerados 24 modelos finais diferentes nesta etapa.

Essa seção apresenta os resultados da fase de **Teste** destes modelos, usando as sequência indicadas como "*Teste*"na Tabela 10. Ou seja, os dados apresentados nesta seção foram gerados usando sequências que não foram usadas nem na etapa de treinamento e nem na etapa de validação. Assim esse conjunto de amostras é inédito para os modelos.

A fim de apresentar de uma forma objetiva a comparação do desempenho de cada um destes 24 modelos, a Tabela 19 apresenta os valores da precisão destes modelos por meio da métrica Fscore.

Conforme apresentado na Tabela 19, os resultados médios de Fscore dos modelos DT foram de 56,41 e de 53,23 para os conjuntos de dados HD720 e HD1080, respectivamente. Isso implica em uma melhoria média de 12,75 e 9,78 no Fscore para ambos os grupos de dados, se comparados com os resultados da versão *baseline*. Ou seja, o ajuste dos hiperparâmetros permitiu uma ampliação importante na qualidade dos resultados gerados em ambos os modelos.

Se comparados os resultados de Fscore do modelo *baseline* com os do modelo RF, conforme apresentado na Tabela 19, é possível perceber uma melhora ainda mais

expressiva. Os resultados médios de Fscore dos modelos RF foram de 58,94 e de 57 para os conjuntos de dados HD720 e HD1080, respectivamente. Assim, a melhoria foi de 15,28 e 13,55 no Fscore em comparação com os resultados da versão *baseline*, isso para os dois grupos de dados. Ou seja, o uso de RF possibilitou uma ampliação bastante expressiva na qualidade dos resultados gerados em ambos os modelos.

Quando comparados os resultados dos modelos DT e RF, é possível perceber, observando os resultados da Tabela 19, que os modelos de RF atingiram melhores resultados de Fscore na maioria dos casos, quando comparado com os modelos DT. Estes ganhos podem chegar a 7,28, no caso do modelo para resolução HD1080 e CQ 32. Apenas para a resolução HD 1080 e CQ 55 o modelo RF tem um Fscore pior do que o modelo DT, mas apenas com 0,08 de diferença. Na média, os ganhos de Fscore dos modelos RF em relação aos modelos DT são de 2,53 e 3,52 para os conjuntos de dados HD720 e HD1080, respectivamente.

Também é possível observar que os modelos apresentaram melhor precisão em resolução de HD720 do que em resolução HD1080 e isso é especialmente válido para os modelos DT, onde a diferença máxima no Fscore chegou a 4,66 para o CQ 32 e 2,93 na média. Para os modelos RF, as diferenças máxima e média no Fscore foram de 2,78 e 1,94, respectivamente.

Outra observação interessante é que os ganhos dos modelos RF em relação aos modelos DT são maiores para a resolução HD1080 do que na resolução HD720, onde, na média, o ganho no Fscore é 0,99 maior para resoluções HD1080. Por outro lado, a variação dos valores de CQ dentro de cada resolução não gera nenhuma tendência observável.

Tabela 19 – Proposta AME - Comparação da precisão dos modelos na fase de **Teste**

Resolução	CQ	Fscore *Baseline*	Fscore DT	Fscore RF
HD720	20	45,92	57,76	60,94
HD720	32	43,28	56,04	59,38
HD720	43	40,14	54,49	57,64
HD720	55	45,31	57,35	57,79
Média		**43,66**	**56,41**	**58,94**
HD1080	20	44,90	54,91	58,16
HD1080	32	42,63	51,38	58,66
HD1080	43	42,00	50,93	55,56
HD1080	55	44,30	55,68	55,60
Média		**43,45**	**53,23**	**57,00**

No geral, os resultados apresentados na Tabela 19 demonstram claramente a im-

portância de se realizar o ajuste de hiperparâmetros, bem como a importância do uso de modelos de aprendizagem por agrupamento.

6.2 Redução de Complexidade por Agrupamento de Modos e CQ Agrupados (AMA)

Uma abordagem alternativa às soluções apresentadas para a abordagem AME também foi avaliada. A abordagem anterior particionou os conjuntos de dados por resolução e por CQ, mas os resultados apresentados indicam que não houve uma diferença significativa a cada CQ testado, se comparadas as precisões obtidas entre os modelos de mesma resolução.

A abordagem Redução de Complexidade por Agrupamento de Modos e CQ Agrupados (AMA) visa explorar a mesma divisão de classes da abordagem AME (isto é, em três grupos de modos preditores intra-quadro do AV1, que representam as classes do aprendizado de máquina). Além disso, foi usando o modelo com melhores resultados dentre os apresentados na seção anterior, ou seja, Florestas Aleatórias. A diferença foi no particionamento dos dados, onde o CQ deixou de ser usado para definir grupos de dados específicos, e passou a ser usado como mais um atributo para a construção do modelo. Assim, ao invés de oito modelos (um para cada resolução e para cada CQ), a solução AMA gerou apenas dois modelos, um para o conjunto de dados da resolução HD720 e outro para a resolução HD1080.

Para definir se é mais vantajoso treinar o modelo sobre um único grupo de dados geral para cada resolução com todos os CQs, ao invés de particionar os grupos de dados e especializar os modelos por CQ, foram treinados dois modelos *baseline*baseados em *Bagging*, através do modelo de Floresta Aleatória. Os grupos de dados usados nesta abordagem são a agregação do conjunto de amostras de treino de cada resolução resultantes da etapa de processamento dos dados, que estavam separados por CQ e resolução na solução AME. Como realizado anteriormente, os grupos de dados foram balanceados por meio do método de *NearMiss* e, em seguida, foi aplicado o método *split*, separando 80% das amostras no subconjunto de treino e os 20% restantes para o subconjunto de validação do modelo. A Figura 26 apresenta o resultado destas operações.

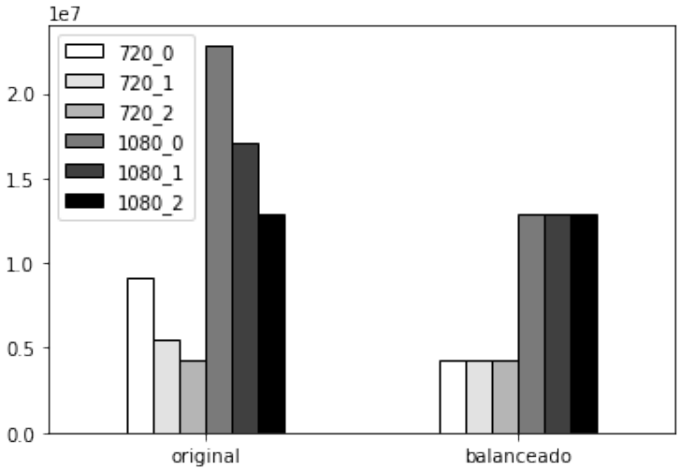

Figura 26 – Proposta AMA - Distribuição de Classes HD720 e HD1080

Na distribuição de classes resultante na Figura 26, é possível visualizar que ambos os grupos de dados balanceados são consideravelmente maiores, se comparados com cada um dos grupos de dados utilizados na abordagem AME. Esse é um aspecto esperado, pois as amostras contidas nos grupos de dados são de todos os CQs de cada resolução. Por outro lado, como esperado, o treino do modelo se torna mais demorado.

A seguir, novamente foram avaliadas as combinação de hiperparâmetros dentro do espaço de busca definido por meio da métrica de avaliação Fscore, a partir do método de teste de validação cruzada igual a cinco. Do mesmo modo que na solução RF da abordagem AME, foi usado o método *Random Search*, limitado a 100 iterações e CV=5, resultando em 500 modelos testados. Da execução do *Random Search* sobre 80% das amostras, extraiu-se o melhor ajuste de hiperparâmetros. Em seguida, cada modelo foi validado no subconjunto de validação.

Por fim, o modelo foi avaliado sobre o grupo de dados de Teste, assim como realizado na proposta AME. Novamente é importante destacar que o grupo de dados de Teste não foi usado em nenhuma das etapas anteriores. Os resultados desta avaliação estão apresentados na Tabela 20, incluindo o Fscore do treinamento, da validação e do teste para os dois modelos AMA construídos, um para HD720 e outro para HD1080. Além disso, para facilitar a comparação, a Tabela 20 também traz ainda os resultados obtidos pela abordagem AME, considerando as médias da métrica Fscore para os diferentes CQs na mesma resolução.

Tabela 20 – Proposta AMA - Precisão do modelo Floresta Aleatória de Três Classes para todos CQs

Proposta	Resolução	Fscore CV=5	Fscore Validação	Fscore Teste
AME	HD720	58,70	58,76	58,94
AMA	HD720	59,97	60,02	58,72
AME	HD1080	56,77	56,82	57,00
AMA	HD1080	58,03	58,15	58,32

O que pode ser observado através da comparação na Tabela 20 é que os modelos construídos com a abordagem AME, mesmo com modelos mais especializados (oito no total), não tiveram resultados superiores aos da abordagem AMA, que usa apenas dois modelos. Considerando as médias dos resultados AME, para o grupo de dados HD720, a abordagem AME atingiu um Fscore médio 0,22 melhor do que a abordagem AMA. Por outro lado, para o grupo de dados HD1080 a situação se inverte, onde a abordagem AMA obteve um Fscore médio 1,32 melhor do que a abordagem AME.

Estes resultados são interessantes, pois indicam que o uso de uma solução de menor complexidade, usando apenas dois modelos, atingiu resultados similares à uma solução mais especializada, usando oito modelos. Assim, a solução AMA foi considerada prioritária para ser utilizada, embora resultados experimentais de eficiência de codificação e tempo de execução são apresentadas para ambas as soluções em 7.

6.3 Implementação no Processo de Codificação no AV1

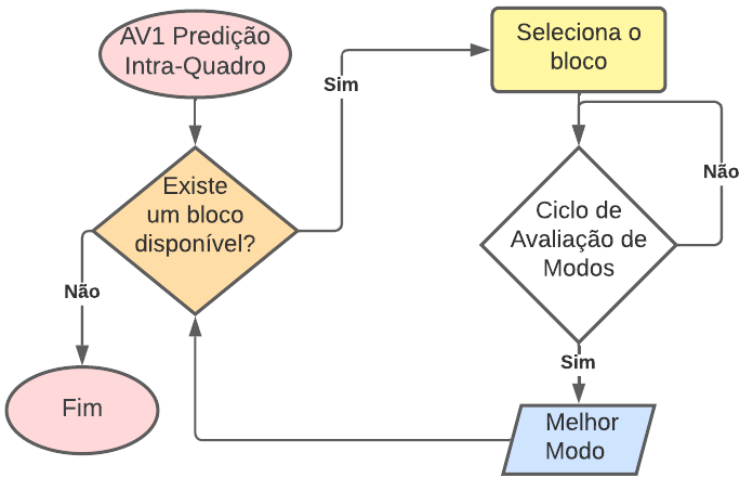

Figura 27 – Fluxograma da Decisão Rápida do Modo Preditor Intra-quadro Original

Na Figura 27 é apresentado um fluxograma de alto nível da implementação original do codificador AV1 da decisão de modos intra, onde o alvo deste estudo é o ciclo de

avaliação de modos.

Os modelos com melhores resultados da abordagem AME e os modelos da abordagem AMA foram implementados no software de referência do AV1 para avaliar os impactos em tempo de execução e eficiência de codificação. Na Figura 28 é apresentado um fluxograma de alto nível do algoritmo desenvolvido para decisão rápida na predição intra-quadro do codificador AV1. Os modelos de aprendizado de máquina supervisionados desenvolvidos substituem o ciclo de avaliação original. Nesta implementação, optou-se por alterar o mínimo possível do fluxo do processo de codificação presente no AV1, logo, os modelos treinados foram exportados na linguagem C em formato de pacote fonte (*source package*) para possibilitar compilar junto ao software referência, sem a necessidade de escrever todo código dentro do fluxo original. Assim, o algoritmo parte do código de predição intra-quadro inalterado.

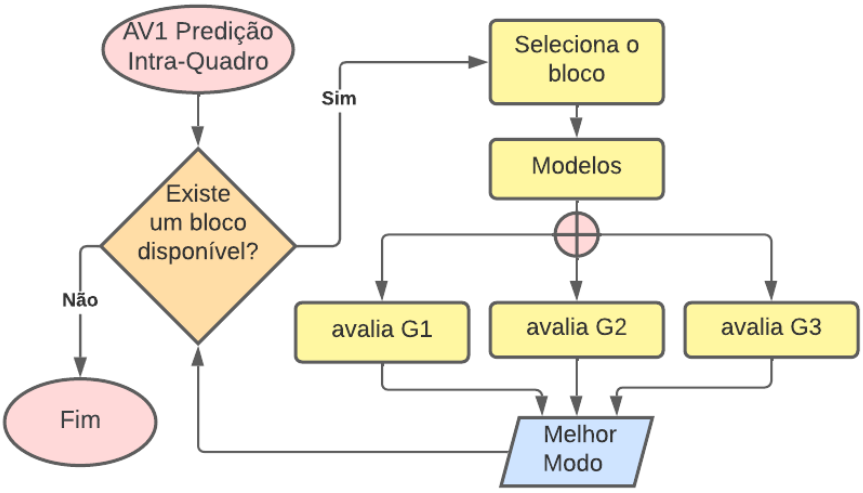

Figura 28 – Fluxograma da Decisão Rápida do Modo Preditor Intra-quadro Proposto

As variáveis disponíveis necessárias para os modelos são armazenadas com base no bloco atual e são entradas para os modelos de aprendizado de máquina supervisionados. A saída do modelo é um entre os três grupos de modos preditores intra-quadro disponíveis, conforme apresentado na Tabela 13 (os modos nos outros dois grupos não são avaliados).

O algoritmo segue após a execução do modelo para o laço de avaliação de modos. Diferentemente da implementação original do AV1, que testa todos os modos preditores intra-quadro para o bloco predito, no algoritmo proposto serão avaliados (ainda segundo os mesmos procedimentos e critérios do software referência) exclusivamente os modos disponibilizados na saída do modelo de aprendizado de máquina.

O laço termina de duas formas: (i) quando o melhor modo foi encontrado durante a busca considerando o *RD-cost* calculado pelo software de referência, ou (ii) quando nenhum dos modos avaliados é adequado. Nesse último caso, outros modos que não

são tratados neste fluxo do software de referência e que não estão presentes nos modelos de aprendizado de máquina desenvolvidos (IntraBC, *Palette*, CfL e *Recursive-based-filtering*) são avaliados. Além disso, o caso (ii) também pode representar uma exceção no software de referência, que possui uma condição que trata tal exceção.

7 RESULTADOS DOS EXPERIMENTOS

O capítulo anterior apresentou as soluções propostas e desenvolvidas com modelos de aprendizado de máquina. Este capítulo irá apresentar os resultados dos experimentos realizados para verificar os impactos no tempo e na eficiência de codificação dos algoritmos propostos implementados no software de referência (AOM, 2020) do codificador AV1. Todos os experimentos foram realizados considerando a metodologia apresentada na Seção 5.1.

A partir destes experimentos, foram obtidas as informações de PSNR (Y/U/V), taxa de bits e tempo de codificação, utilizados na análise objetiva. Também foram calculados o BD-Rate da camada de luminância, a redução de tempo de codificação (RT) e a relação entre BD-Rate e o RT (isto é, para cada 1% de redução de tempo, o quanto se perde em eficiência de codificação). O cálculo de redução de tempo de codificação está definido na equação (19), onde RT é a redução percentual do tempo, TR_{CQ} é o tempo de codificação sem o uso dos modelos e TP_{CQ} é o tempo de codificação com o uso dos modelos propostos. No somatório, CQ pode ter os quatro valores base, ou seja, 20, 32, 43 e 55. O cálculo do desvio padrão neste cenário está apresentado na equação (20), onde M_a representa a média aritmética dos dados e, X_i o valor na posição *i* no conjunto de dados.

$$RT = \frac{1}{4}\left(\sum_{CQ=20}^{55} \frac{TR_{CQ} - TP_{CQ}}{TR_{CQ}}\right) * 100 \tag{19}$$

$$DesvioPadrao = \sqrt{\frac{\sum_{i=1}^{8}(X_i - M_a)^2}{8}} \tag{20}$$

Conforme discutido no capítulo anterior, a abordagem AMA pareceu a mais promissora, por ter resultados similares à abordagem AME e por usar apenas dois modelos ao invés de oito. Ainda assim, foram gerados resultados experimentais com a abordagem AME, que são apresentados na Tabela 21. Nesse caso, foram feitos experimentos mais simples, usando apenas uma sequência de cada resolução: *KristenAndSara*, de resolução HD720, e *Seaplane_hdr_amazon*, de resolução HD1080. Estas sequências

não foram usadas no processo de treinamento e validação dos modelos, conforme apresentado na Tabela 10.

Na Tabela 21 é possível observar que os resultados para a sequência de resolução maior foram muito melhores do que os resultados para a sequência de resolução menor. Na sequência HD1080, o impacto em tempo de execução foi superior a 60%, com impacto em eficiência de codificação inferior a 6%.

Tabela 21 – BD-Rates e RT para solução AME de resolução HD720 e HD1080

Resolução	Sequência	BD-Rate (%)	RT (%)	BD-Rate/RT
HD720	KristenAndSara	14,76	45,26	0,3261
HD1080	Seaplane_hdr_amazon	5,94	62,33	0,0953

Os resultados obtidos com a abordagem AMA foram mais detalhados, já que esta solução apresentou melhor relação entre o Fscore obtido e o número de modelos utilizados, como já discutido no capítulo anterior. Nesse caso, cada um dos dois modelos desenvolvidos foram avaliados com oito sequências de vídeo. Novamente, estas sequências não foram usadas no processo de treinamento e validação dos modelos, conforme apresentado na Tabela 10.

A Tabela 22 apresenta os resultados obtidos com o modelo construído para sequências de vídeo HD720. o Apêndice C apresenta os dados detalhados dos experimentos que geraram esta tabela. Ao observar os valores apresentados na Tabela 22, é possível perceber 55,69% de RT médio (com um desvio padrão de 6,4%) e um acréscimo médio de BD-Rate de 10% (com desvio padrão de 3,6%).

Considerando unicamente o contexto em que se obteve a maior redução de tempo, sem avaliar o aumento de BD-Rate, o melhor resultado é da sequência de vídeo *Netflix_Tango*, com aproximadamente 65% de RT. Por outro lado, avaliando exclusivamente o menor BD-Rate obtido, o melhor resultado foi para sequência de vídeo *Rain_hdr_amazon*, com 3,33%.

Tabela 22 – BD-Rate e RT para a solução AMA de resolução HD720

Sequência	BD-Rate (%)	RT (%)	BD-Rate/RT
KristenAndSara	14,16	44,34	0,3195
FourPeople	10,65	60,23	0,1770
Johnny	14,49	47,43	0,3056
Netflix_DinnerScene	10,01	55,41	0,1807
Netflix_FoodMarket2	5,85	58,31	0,1004
Netflix_Tango	11,32	65,51	0,1728
Rain_hdr_amazon	3,33	56,21	0,0592
Vidyo3	11,44	58,15	0,1969
Média	**10,16**	**55,69**	**0,1890**

A relação entre BD-Rate e RT (BD-Rate/RT) tem seu melhor resultado quando essa relação tem seu menor valor (valor mais próximo de zero). Voltando à Tabela 22, é possível perceber que, neste caso, o melhor cenário é para a sequência de vídeo *Rain_hdr_amazon*, onde para cada 1% de redução de tempo é acrescido apenas 0,0592% em BD-Rate. Por outro lado, a sequência *KristenAndSara* teve o pior resultado nesse cenário, pois a cada 1% de redução de tempo é acrescido 0,3195% em BD-Rate.

A Tabela 23 apresenta os resultados obtidos com o modelo construído para as sequências de vídeo HD1080. Toda relação detalhada dos dados que geraram esta tabela estão apresentados no Apêndice C. Na resolução HD1080, é observada uma redução de tempo médio de 44,69% (com um desvio padrão de 8,9%) e um acréscimo médio de BD-Rate de 4,7% (desvio padrão de 3,3%).

Tabela 23 – BD-Rate e RT para solução AMA de resolução HD1080

Sequência	BD-Rate (%)	RT (%)	BD-Rate/RT
Seaplane_hdr_amazon	3,92	56,96	0,0689
Crowd_run	2,17	54,82	0,0397
Guitar_hdr_amazon	8,09	42,09	0,1924
Netflix_Aerial	2,49	46,09	0,0542
Station2	11,60	49,70	0,2335
Pan_hdr_amazon	4,55	27,31	0,1668
Old_town_cross	3,48	42,85	0,0814
Park_joy	1,04	37,77	0,0277
Média	**4,67**	**44,69**	**0,1080**

Considerando apenas a sequência com maior RT, a sequência com melhor resultado é a *Seaplane_hdr_amazon*, com 56,96% de RT. Do mesmo modo, ao considerar

apenas o menor BD-Rate obtido, o melhor resultado é na sequência *Park_joy* com um aumento de 1,04% do BD-Rate. Entretanto, como destacado anteriormente, o melhor cenário neste trabalho é aquele onde observou-se o menor valor para a relação BD-Rate/RT. Assim, o melhor resultado neste cenário é para a sequência*Park_joy*, onde para cada 1% de redução de tempo é acrescido 0,0277% em BD-Rate. O pior caso fica para sequência de *Station2* onde para cada 1% de RT ocorre um acréscimo de 0,2335% em BD-Rate.

Outra observação importante é que os resultados foram melhores para o modelo especializado em vídeos HD1080 do que para o modelo especializado em vídeos HD720. Esse fato é constatado, principalmente, no que diz respeito aos impactos em eficiência de codificação, que foram muito menores para o modelo HD1080, onde a perda média de eficiência de codificação foi menos que a metade daquela atingida para o modelo HD720. Por outro lado, a redução de tempo foi cerca de 10% menor para o modelo HD1080 em comparação ao modelo HD720, mas ainda assim com resultados expressivos. A relação BD-Rate/RT também foi melhor no modelo especializado em vídeo HD1080.

A solução com a abordagem AME serviu para validar essencialmente dois conceitos. O primeiro de que a abordagem de reduzir de alguma forma os modos preditores disponíveis durante a tomada de decisão na predição intra-quadro reverteria em uma diminuição no custo computacional. Neste sentido, era esperado um RT com mínimo possível de acréscimo no BD-Rate. O segundo conceito validado foi o método para reduzir os modos, ou seja, como seriam determinados os modos disponíveis durante o processo de predição intra. Para tanto, os modos com certa relação foram agrupados. Assim, os dois conceitos iniciais foram validados com a abordagem AME, onde se demonstrou ser possível reduzir a complexidade deste processo diminuindo os modos intra, obtendo uma média geral de 53,79% de RT.

A solução com a abordagem AMA, por sua vez, apresentou melhorias em relação à abordagem AME, tornando o processo de treinamento do aprendizado de máquina e a implementação no libaom mais simplificados. Foram diminuídos os modelos de oito para dois, um por resolução testada. Para as sequências de vídeo como melhores resultados se avaliado o RT, foram obtidos até 65,51% com média de 55,69% para HD720 e, até 56,96% com média de 44,69% de RT para HD1080.

Existem alguns trabalhos na literatura que apresentam soluções eficientes visando a predição intra-quadro do codificador AV1, como JIN et al. (2021), XU; JEON (2020) e JEONG; GANKHUYAG; KIM (2019). A solução em JIN et al. (2021) concentra-se na melhoria da eficiência de codificação intra-quadro do AV1. Os trabalhos XU; JEON (2020) e JEONG; GANKHUYAG; KIM (2019) centram-se na redução do custo computacional para a predição intra-quadro, utilizando heurísticas. Então, nenhum desses trabalhos explora soluções baseadas em aprendizado de máquina. O trabalho

em HE et al. (2021) propôs uma solução baseada na aprendizagem de máquina para a predição intra-quadro, mas voltado para o codificador *Versatile Video Coding* (VVC) e não o AV1.

Apenas os trabalhos XU; JEON (2020) e JEONG; GANKHUYAG; KIM (2019) estão focados na redução do custo computacional, permitindo uma comparação com os resultado obtidos deste trabalho. Em XU; JEON (2020) é proposta uma seleção eficiente de ferramentas de predição intra-quadro para AV1, que fornece cerca de 5% e 2% de RT com perda de eficiência de codificação insignificante, segundo o autor, quando avaliada usando o *libaom* versão 1.0. O trabalho em JEONG; GANKHUYAG; KIM (2019) propôs um algoritmo de decisão rápida do modo intra-quadro que atinge 8,67% da RT com acréscimo no BD-Rate de 0,04%, mas a versão *libaom* não é apresentada.

Comparando os resultados da solução com a abordagem AMA, com os resultados em XU; JEON (2020) e JEONG; GANKHUYAG; KIM (2019), pode-se concluir que o presente trabalho atingiu RT expressivamente maior (10 vezes maior que XU; JEON (2020) e 5,8 vezes maior que JEONG; GANKHUYAG; KIM (2019), com base nos resultados médios). Por outro lado, a abordagem agressiva usada nesta solução para reduzir o custo computacional necessário também causou um maior impacto na eficiência de codificação.

8 CONCLUSÕES

Este trabalho apresentou soluções baseadas em aprendizado de máquina para a decisão rápida na predição intra-quadro do codificador de vídeo AV1.

Neste sentido, foram feitas diversas investigações que antecederam a escolha da melhor solução. Por conta da falta de uma base de dados foi realizada a aquisição de dados com base em experimentos realizados no software de referência do AV1 e processamento dos dados extraídos para construção de conjuntos, foram investigadas algumas maneiras de dividir as classes deste aprendizado de máquina, dentre elas: criação de um modelo específico por modo intra, um modelo por resolução com 13 classes (que representavam os modos intra explorados), um modelo por agrupamento de modos por CQ e resolução e, um modelo por agrupamento de modos agregando os CQ.

Por fim, foi escolhida a abordagem AMA como melhor opção, por ser mais simples, com apenas dois modelos, e atingir resultados similares (e melhores em alguns casos) à abordagem AME.

A redução de tempo foi possível devido à abordagem adotada de reduzir o número de modos disponíveis na busca do melhor modo intra no libaom, apresentando um subconjunto do total de modos intra definidos pelo AV1. Na solução AMA, os modelos preditivos foram divididos por resolução: HD720 e HD1080, e foram definidas três classes: (i) modos direcionais, (ii) modo DC e (iii) modos *Smooth* e *Paeth*.

Com os dois modelos treinados, validados e testados, ambos foram inseridos no software de referência do AV1, interagindo no laço de decisão do modo intra, sem alterar os demais processos do libaom. A seguir, foram codificadas oito sequências de video para cada modelo e foram coletados resultados de redução de tempo e de impacto em eficiência de codificação. Um experimento mais simples também foi realizada com os 16 modelos gerados com a abordagem AME.

Os resultados experimentais dos modelos construídos com a abordagem AMA, na melhor solução proposta, indicaram uma média de redução de tempo de execução de 44,69%, com uma perda de eficiência de codificação de 4,67% no BD-Rate. Considerando o caso com maior redução de complexidade, a solução foi capaz de alcançar

65,51% de redução de tempo de codificação.

Observando os resultados obtidos com a solução desenvolvida, quando comparados aos resultados originais do software de referência, é possível concluir que foi possível atingir uma elevada redução de tempo de codificação. A relação entre a redução de tempo e as perdas de eficiência de codificação foram bastante favoráveis, além disso, tanto quanto é do conhecimento do autor, a solução com abordagem AMA é a primeira na literatura a explorar modelos de aprendizado de máquina supervisionado para reduzir o custo computacional necessário da predição intra-quadro do codificador AV1.

Como trabalho futuro, pretende-se investigar uma abordagem de modelos binários sequenciais que compõem os mesmos subconjuntos propostos, onde cada modelo será especializado em um dos três grupos, calculando as probabilidades de um determinado bloco intra utilizar aquele subconjunto ou não. Serão testados arranjos diferentes dos três grupos, a fim de descobrir a melhor ordem para realizar os testes durante a decisão do modo intra-quadro. Também será avaliada a inclusão de novos parâmetros, não nativos na implementação do libaom, para tentar melhorar ainda mais a acurácia dos modelos propostos. Entre os novos parâmetros possíveis está a média e a variância das amostras do bloco, os modos escolhidos nos blocos vizinhos, entre outros. Nesse caso, novas funções serão implementadas no software de referência para disponibilizar estes valores para os modelos.

REFERÊNCIAS

AGOSTINI, L. V. Desenvolvimento de Arquiteturas de Alto Desempenho Dedicadas à Compressão de Vídeo Segundo o Padrão H. 264/AVC. **Universidade Federal do Rio Grande do Sul. Instituto de Informática. Programa de Pós-Graduação em Computação**, [S.l.], p.31–173, 08 2007.

AKYAZI, P.; EBRAHIMI, T. Comparison of Compression Efficiency between HEVC/H.265, VP9 and AV1 based on Subjective Quality Assessments. , [S.l.], p.1–6, 05 2018.

ANDRADE, M. A.; LOPES, G.; VILELA, C. O sítio calcolítico do Cabeço dos Mouros: identificação de uma nova oficina de talhe de pontas de seta na área de Arruda dos Pisões (Rio Maior, Portugal). **Revista Portuguesa de Arqueologia**, [S.l.], v.17, p.113–126, 2014.

AOM. **Alliance for Open Media - Git at Google**. Disponível em: <https://aomedia.googlesource.com/aom/>. Acesso em: 2020-02-18.

AOMEDIA. **AV1 Bitstream & Decoding Process Specification**. Disponível em: <https://aomediacodec.github.io/av1-spec/av1-spec.pdf>. Acesso em: 2021-10-18.

ASSIS, A. K. T. **Arquimedes, o centro de gravidade ea lei da alavanca**. [S.l.]: Apeiron Montreal, 2008.

BERGSTRA, J.; BENGIO, Y. Random Search for Hyper-Parameter Optimization. **The Journal of Machine Learning Research**, [S.l.], v.13, p.281–305, 03 2012.

BITMOVINS. **Covid-19's impact on streaming video**. Disponível em: <https://go.bitmovin.com/hubfs/Covid-19%20OTT%20streaming_Bitmovin_Analytics_Infographic.pdf>. Acesso em: 2020-12-30.

BJØNTEGAARD, G. Calculation of Average PSNR Differences between RD-curves. In: 2001. **Anais...** [S.l.: s.n.], 2001.

BONACCORSO, G. **Machine Learning Algorithms**: A Reference Guide to Popular Algorithms for Data Science and Machine Learning. [S.l.]: Packt Publishing, 2017.

BOSSEN, F. et al. Common test conditions and software reference configurations. **JCTVC-L1100**, [S.l.], v.12, n.7, 2013.

BREIMAN, L. Bagging Predictors. **Machine Learning**, [S.l.], v.24, n.2, p.123–140, Aug 1996.

BREIMAN, L.; FRIEDMAN, J. H.; OLSHEN, R. A.; STONE, C. J. **Classification and regression trees**. [S.l.]: Routledge, 2017.

CARBONELL, J. G.; MICHALSKI, R. S.; MITCHELL, T. M. 1 - AN OVERVIEW OF MACHINE LEARNING. In: MICHALSKI, R. S.; CARBONELL, J. G.; MITCHELL, T. M. (Ed.). **Machine Learning**. San Francisco (CA): Morgan Kaufmann, 1983. p.23,39–40.

Chen, Di and Liu, Z.; XU, Y.; ZHU, F.; DELP, E. Multi-Reference Video Coding Using Stillness Detection. **Electronic Imaging**, [S.l.], v.2018, 03 2018.

CHEN, J. CE6.a.4 : Chroma intra prediction by reconstructed luma samples. **JCTVC-E266**, [S.l.], 2011.

CHEN, Y. et al. An Overview of Core Coding Tools in the AV1 Video Codec. , [S.l.], p.41–45, 06 2018.

CHEN, Y. et al. An Overview of Coding Tools in AV1: the First Video Codec from the Alliance for Open Media. **APSIPA Transactions on Signal and Information Processing**, [S.l.], v.9, p.e6, 2020.

CHEN, Y.; MUKHERJEE, D. Variable block-size overlapped block motion compensation in the next generation open-source video codec. In: 2017. **Anais...** [S.l.: s.n.], 2017. p.938–942.

CISCO. Visual Networking Index (VNI) Global 2022 Forecast Highlights. , [S.l.], 2018.

CISCO, W. P. Cisco Annual Internet Report (2018–2023) White Paper. , [S.l.], 2020.

CORREA, G. et al. Homogeneity and distortion-based intra mode decision architecture for H.264/AVC. In: IEEE INTERNATIONAL CONFERENCE ON ELECTRONICS, CIRCUITS AND SYSTEMS, 2010., 2010. **Anais...** [S.l.: s.n.], 2010. p.591–594.

CRESWELL, A. et al. Generative Adversarial Networks: An Overview. **IEEE Signal Processing Magazine**, [S.l.], v.35, n.1, p.53–65, 2018.

DAEDE, T.; NORKIN, A.; BRAILOVSKIY, I. **Video Codec Testing and Quality Measurement**. [S.l.]: Internet Engineering Task Force, 2020. Internet-Draft, Work in Progress. (draft-ietf-netvc-testing-09).

MAIMON, O.; ROKACH, L. (Ed.). **Decision Trees**. Boston, MA: Springer US, 2005. p.165–192.

DUDA, R. O. et al. **Pattern Classification, 2nd Ed**.

EFRON, B.; TIBSHIRANI, R. J. **An Introduction to the Bootstrap**. Boca Raton, Florida, USA: Chapman & Hall/CRC, 1993. n.57. (Monographs on Statistics and Applied Probability).

EGGE, N. **Into the Depths**: The Technical Details Behind AV1. Disponível em: <http://mile-high.video/files/mhv2018/pdf/day1/1_02_Egge.pdf>. Acesso em: 2019-01-10.

Overlapped Block Motion Compensation. In: ENCYCLOPEDIA OF MULTIMEDIA, 2006, Boston, MA. **Anais...** Springer US, 2006. p.681–683.

GASHLER, M.; GIRAUD-CARRIER, C.; MARTINEZ, T. Decision Tree Ensemble: Small Heterogeneous Is Better Than Large Homogeneous. In: SEVENTH INTERNATIONAL CONFERENCE ON MACHINE LEARNING AND APPLICATIONS, 2008., 2008. **Anais...** [S.l.: s.n.], 2008. p.900–905.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep learning**. [S.l.]: MIT press, 2016.

GROIS, D.; NGUYEN, T.; MARPE, D. Coding efficiency comparison of AV1/VP9, H.265/MPEG-HEVC, and H.264/MPEG-AVC encoders. , [S.l.], p.1–5, 01 2016.

HAN, J. et al. **A Technical Overview of AV1**.

HE, Q. et al. Random Forest Based Fast CU Partition for VVC Intra Coding. In: IEEE INTERNATIONAL SYMPOSIUM ON BROADBAND MULTIMEDIA SYSTEMS AND BROADCASTING (BMSB), 2021., 2021. **Anais...** [S.l.: s.n.], 2021. p.1–4.

HECHT, J. The bandwidth bottleneck that is throttling the Internet. **Nature**, [S.l.], v.536, p.139–142, 2016.

HOROWITZ, A. M. A generalized guided Monte Carlo algorithm. **Physics Letters B**, [S.l.], v.268, n.2, p.247–252, 1991.

HUA, Y.; GUO, J.; ZHAO, H. Deep Belief Networks and deep learning. In: INTERNATIONAL CONFERENCE ON INTELLIGENT COMPUTING AND INTERNET OF THINGS, 2015., 2015. **Proceedings...** [S.l.: s.n.], 2015. p.1–4.

HUANG, F.; XIE, G.; XIAO, R. Research on Ensemble Learning. In: INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE AND COMPUTATIONAL INTELLIGENCE, 2009., 2009. **Anais...** [S.l.: s.n.], 2009. v.3, p.249–252.

ITU-T, I. .-. "High efficiency video coding". **ITU-T Recommendation H.265 and ISO/IEC 23008-2**, [S.l.], 2019.

ITU-T, I. .-. "Versatile Video Coding". **ITU-T Recommendation H.266 and ISO/IEC 23090-3**, [S.l.], 2020.

JEONG, J.; GANKHUYAG, G.; KIM, Y.-H. A Fast Intra Mode Decision Based on Accuracy of Rate Distortion Model for AV1 Intra Encoding. In: INTERNATIONAL TECHNICAL CONFERENCE ON CIRCUITS/SYSTEMS, COMPUTERS AND COMMUNICATIONS (ITC-CSCC), 2019., 2019. **Anais...** [S.l.: s.n.], 2019. p.1–3.

JIN, Y. et al. Improved Intra Mode Coding Beyond Av1. In: ICASSP 2021 - 2021 IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING (ICASSP), 2021. **Anais...** [S.l.: s.n.], 2021. p.1580–1584.

JOSHI, U. et al. Novel inter and intra prediction tools under consideration for the emerging AV1 video codec. In: APPLICATIONS OF DIGITAL IMAGE PROCESSING XL, 2017. **Anais...** SPIE, 2017. v.10396, p.54 – 66.

KHAN, S. A.; ALI RANA, Z. Evaluating Performance of Software Defect Prediction Models Using Area Under Precision-Recall Curve (AUC-PR). In: INTERNATIONAL CONFERENCE ON ADVANCEMENTS IN COMPUTATIONAL SCIENCES (ICACS), 2019., 2019. **Anais...** [S.l.: s.n.], 2019. p.1–6.

LAYEK, A. et al. Performance analysis of H.264, H.265, VP9 and AV1 video encoders. , [S.l.], p.322–325, 09 2017.

MITCHELL, T. M. **The need for biases in learning generalizations**. [S.l.]: Department of Computer Science, Laboratory for Computer Science Research . . . , 1980.

MOHANDOSS, D. P.; SHI, Y.; SUO, K. Outlier Prediction Using Random Forest Classifier. In: IEEE 11TH ANNUAL COMPUTING AND COMMUNICATION WORKSHOP AND CONFERENCE (CCWC), 2021., 2021. **Anais...** [S.l.: s.n.], 2021. p.0027–0033.

MONARD, M. C.; BARANAUSKAS, J. A. Conceitos Sobre Aprendizado de Máquina. In: **Sistemas Inteligentes Fundamentos e Aplicações**. 1.ed. Barueri-SP: Manole Ltda, 2003. p.89–114.

MUKHERJEE, D. et al. An overview of new video coding tools under consideration for VP10: the successor to VP9. In: SPIE OPTICAL ENGINEERING + APPLICATIONS, 2015. **Anais...** [S.l.: s.n.], 2015.

MUKHERJEE, D. et al. A Technical Overview of VP9–the Latest Open-Source Video Codec. **SMPTE Motion Imaging Journal**, [S.l.], v.124, p.44–54, 02 2015.

NIELSEN. **The Nielsen Total Audience Report**: August 2020. Disponível em: <https://www.nielsen.com/us/en/insights/report/2020/the-nielsen-total-audience-report-august-2020>. Acesso em: 2020-08-13.

OLIVEIRA RIBEIRO, S. de. Bioestatística. In: . [S.l.: s.n.], 2009. p.151.

OPITZ, D.; MACLIN, R. Popular Ensemble Methods: An Empirical Study. , [S.l.], v.11, 12 1999.

OZER, J. **Current Status of HEVC Royalties**. Disponível em: <https://streaminglearningcenter.com/blogs/current-status-of-hevc-royalties.html>. Acesso em: 2020-02-17.

PARKER, S. et al. On transform coding tools under development for VP10. In: 2016. **Anais...** [S.l.: s.n.], 2016. p.997119.

PARKER, S. et al. Global and locally adaptive warped motion compensation in video compression. In: 2017. **Anais...** [S.l.: s.n.], 2017. p.275–279.

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

PRATI, R. C.; BATISTA, G. E. A. P. A.; MONARD, M. C. Evaluating Classifiers Using ROC Curves. **IEEE Latin America Transactions**, [S.l.], v.6, n.2, p.215–222, 2008.

QUINLAN, J. R. Learning efficient classification procedures and their application to chess end games. In: **Machine learning**. [S.l.]: Springer, 1983. p.463–482.

QUINLAN, J. R. **C4. 5**: programs for machine learning. [S.l.]: Elsevier, 2014.

RICHARDSON, I. The H.264 Advanced Video Compression Standard: Second Edition. , [S.l.], 04 2010.

RIVAZ, J. H. Peter de. **AV1 Bitstream Decoding Process Specification**. Disponível em: <http://aomedia.org/av1/specification/>. Acesso em: 2020-05-01.

RODRIGUES, S. C. A. **Modelo de regressão linear e suas aplicações**. 2012. Tese (Doutorado em Ciência da Computação) — Universidade da Beira Interior.

SAMAJDAR, T.; QURAISHI, M. **Analysis and Evaluation of Image Quality Metrics**. [S.l.: s.n.], 2015. v.340, p.369–378.

SCHAPIRE, R. E. The Boosting Approach to Machine Learning An Overview. In: 2003. **Anais...** [S.l.: s.n.], 2003.

SHOBHA, G.; RANGASWAMY, S. Chapter 8 - Machine Learning. In: GUDIVADA, V. N.; RAO, C. (Ed.). **Computational Analysis and Understanding of Natural Languages**: Principles, Methods and Applications. [S.l.]: Elsevier, 2018. p.197–228. (Handbook of Statistics, v.38).

TRUDEAU, L.; EGGE, N.; BARR, D. Predicting Chroma from Luma in AV1. , [S.l.], 11 2017.

WIERING, M. A.; VAN OTTERLO, M. Reinforcement learning. **Adaptation, learning, and optimization**, [S.l.], v.12, n.3, 2012.

WIERING, M.; OTTERLO, M. **Reinforcement Learning**: State-Of-The-Art. [S.l.: s.n.], 2012. v.12.

XU, M.; JEON, B. Selection of Intra Prediction Tools for Fast AV1 Encoding. In: IEEE INTERNATIONAL SYMPOSIUM ON BROADBAND MULTIMEDIA SYSTEMS AND BROADCASTING (BMSB), 2020., 2020. **Anais...** [S.l.: s.n.], 2020. p.1–4.

XU, M.; JEON, B. User-Priority Based AV1 Coding Tool Selection. **IEEE Transactions on Broadcasting**, [S.l.], v.67, n.3, p.736–745, 2021.

ZHANG, J.; MANI, I. KNN Approach to Unbalanced Data Distributions: A Case Study Involving Information Extraction. In: Proceedings of the ICML'2003 Workshop on Learning from Imbalanced Datasets, 2003. **Anais...** [S.l.: s.n.], 2003.

Apêndices

APÊNDICE A – Síntese do fluxo de tomada de decisão do modo intra do software de referência do AV1

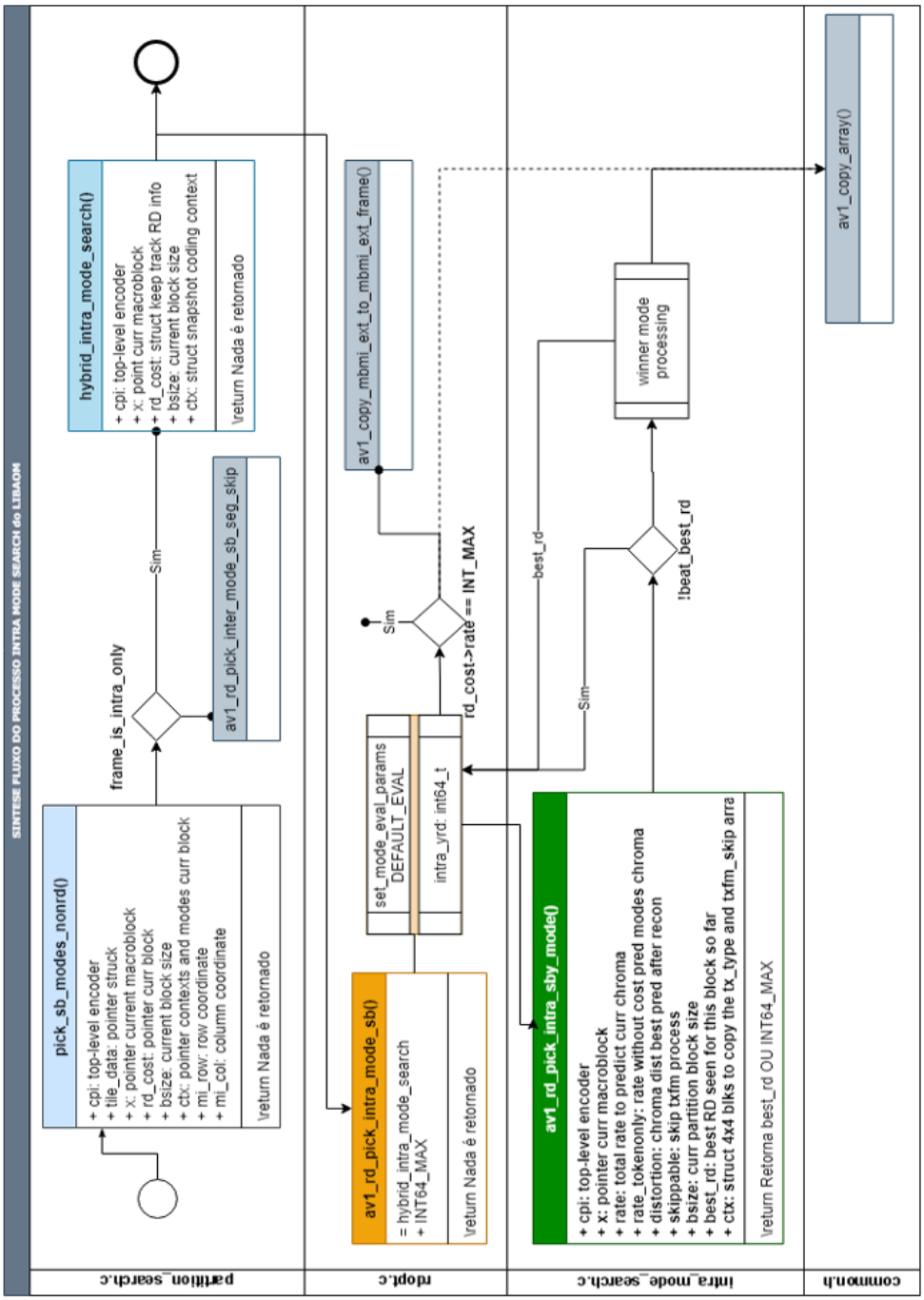

Figura 29 – Síntese do fluxo de tomada de decisão do modo intra do software de referência do AV1

APÊNDICE B – Condições Comuns de Teste (CTC)

B.1 Condições Comuns de Teste

Com intuito de possibilitar uma comparação adequada entre várias implementações de codificadores distintos, ou entre um mesmo codificador com diferentes configurações, condições comuns de teste (CTC) são definidas por grupos ou órgãos voltados no desenvolvimento e promoção de padrões de codificação de vídeo. Cabe o exemplo do HEVC, onde o documento JCTVC-L1100 apresenta as condições comum de teste para aquele padrão BOSSEN et al. (2013). Para o AV1, o grupo de pesquisa *Internet Video Codec* da *Internet Engineering Task Force* (IETF) foi responsável por desenvolver as condições comum de teste do AV1 AOMEDIA (2019). O objetivo foi permitir dados confiáveis em comparações entre diferentes codificadores de código aberto ou a diferentes configurações e versões de um mesmo codificador. Periodicamente o NETVC atualiza este documento e, até o término do desenvolvimento deste trabalho, a versão mais atual do documento é a *NETVC-Testing-09* e está disponível em (DAEDE; NORKIN; BRAILOVSKIY, 2020).

O documento *NETVC-Testing-09* está divido em nove seções, dentre elas, são relevantes para este trabalho as seções 3, 4 e 5, que tratam das recomendações de métricas objetivas, as formas de comparação e a interpretação dos resultados e, por fim, as sequências de testes recomendadas.

B.1.1 Métricas Objetivas

No documento DAEDE; NORKIN; BRAILOVSKIY (2020), seção 3, encontram-se as métricas objetivas recomendadas, que são utilizadas no lugar de testes subjetivos para tornar os experimentos mais fáceis e possíveis de serem repetidos. Os testes subjetivos usam opiniões de espectadores humanos, aplicando certas metodologias para avaliar e comparar a qualidade de um vídeo. As métricas objetivas são cálculos matemáticos comparando os vídeos reconstruídos após a compressão com os vídeos originais. A maioria das métricas objetivas são projetadas para se correlacionar com as medidas subjetivas, embora todas elas possuam limitações. As métricas objetivas expostas em DAEDE; NORKIN; BRAILOVSKIY (2020) são: *Peak Signal-to-Noise Ratio* (PSNR), *Frame-averaged* PSNR, PSNR *Human Visual System* (PSNR-HVS-M), *structural similarity index measure* (SSIM), *Multi-Scale* SSIM, CIEDE2000 e *Video Mul-*

timethod Assessment Fusion (VMAF).

Dentre as métricas recomendadas, a mais extensamente utilizada é a PSNR (SAMAJDAR; QURAISHI, 2015), que gera um resultado em uma escala logarítmica e depende do erro médio quadrático (*mean squared error* - MSE), conforme definido na equação e na equação 22. O MSE é calculado entre quadros ou blocos do vídeo original e do video reconstruído, onde o erro é calculado sobre todas as amostras do vídeo. A unidade de medida do PSNR é decibéis (dB).

$$PSNR = 10 * log_{10}\left(\frac{MAX^2}{MSE}\right) \tag{21}$$

$$MSE = \frac{1}{M * N}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}[f(x,y) - g(x,y)]^2 \tag{22}$$

Na Equação 22, M e N representam as dimensões do quadro ou do bloco do vídeo e $f(x,y)$ e $g(x,y)$ são as amostras na posição x e y do quadro ou do bloco original e reconstruído, respectivamente. Enquanto na equação 21, MAX é o mais alto valor de sinal possível do quadro original, onde, para vídeos de 8 bits por amostra, o valor é de 256. Conforme (DAEDE; NORKIN; BRAILOVSKIY, 2020) esta métrica de qualidade pode ser aplicada a ambas as camadas de cor: luminância e crominância separadamente, embora o mais comum é utilizar apenas a informação de luminância (Y), identificada por PSNR-Y.

B.1.2 Comparando e Interpretando Resultados

No documento DAEDE; NORKIN; BRAILOVSKIY (2020) também estão as recomendações de como apresentar os resultados das comparações, sejam elas entre codificadores distintos ou de diferentes configurações de um mesmo codificador. Na seção 4.1 do documento DAEDE; NORKIN; BRAILOVSKIY (2020) estão as recomendações de como usar gráficos para explorar a relação de taxa de bits (*bitrate*) no eixo X e a métrica de qualidade no eixo Y.

Na seção 4.2 é apresentada a recomendação de como usar a métrica de Bjontegaard Rate Difference (BD-Rate) (BJØNTEGAARD, 2001), que possibilita representar nitidamente a diferença média no *bitrate* entre os dois vídeos comparados para a mesma qualidade de imagem. Os resultados de BD-Rate são uma forma convincente de comparar dois codificadores diferentes ou duas configurações diferentes do mesmo codificador. O cálculo é realizado com as informações entre as curvas de *bitrate* e PSNR. Quanto mais pontos são observados, maior é a precisão, porém, ao menos três pontos no gráfico precisam ser avaliados para possibilitar o cálculo dos valores. Uma forma de definir estes pontos é através de diferentes configurações do experimento variando o CQ (Capítulo 2), que controla a quantização do AV1 (vari-

ando a qualidade de codificação). A recomendação mínima apresentada em DAEDE; NORKIN; BRAILOVSKIY (2020) é utilizar quatro CQs: 20, 32, 43 e 53.

Na Figura 30 é possível observar a exemplificação do gráfico de BD-Rate para o mesmo codificador fictício com configurações distintas do experimento, onde a área destacada entre as curvas demonstra a diferença calculada pelo BD-Rate.

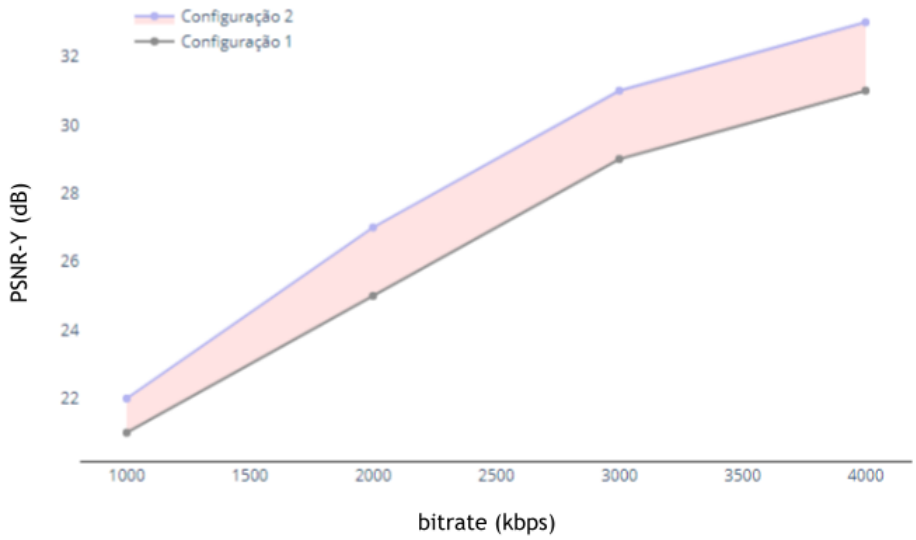

Figura 30 – Exemplo de gráfico de *bitrate* x PSNR e a área do *Bjontegaard Rate Difference* (BD-Rate)

No exemplo da Figura 30, a Configuração 2 é a que tem maior eficiência de codificação (maior PSNR para o mesmo *bitrate* em todos os pontos observados). O cálculo de BD-Rate é detalhado em (BJØNTEGAARD, 2001).

B.1.3 Sequências de Teste

Na seção 5 do documento DAEDE; NORKIN; BRAILOVSKIY (2020) são feitas recomendações de sequências de vídeo para a realização dos experimentos. São recomendadas três categorias distintas, denominadas grupos de teste. Cada grupo de teste visa avaliar diferentes cenários em que o codificador poderá ser usado. Os grupo de teste estão organizados por vídeos rotulados, respectivamente com resolução, profundidade de bits, taxa de subamostragem de cor e número de quadros disponíveis. Os grupos de teste com suas respectivas sequências de vídeo, são apresentados a seguir.

1. **Grupo de teste *regression-1***: É apresentado no item 5.2.1 do documento e é um conjunto de testes básicos para testes de regressão. Ele contém um número pequeno vídeos:

   (a) Vídeo *kirlandvga*: **(640x360, 8 bits, 4:2:0, 300 quadros)**;

(b) Vídeo *FourPeople*: **(1280x720, 8 bits, 4:2:0, 60 quadros)**;

(c) Vídeo *Narrarator*: **(4096x2160, 10 bits, 4:2:0, 15 quadros)**;

(d) Vídeo *CSGO*: **(1920x1080, 8 bits, 4:4:4, 60 quadros)**;

2. **Grupo de teste*objective-2-slow***: É apresentado no item 5.2.2 do documento e é um conjunto de testes mais abrangente, agrupado por resolução. As sequências foram escaladas e cortadas para coincidir com a resolução de sua categoria com 8 e 10 bits de profundidade:

   (a) Vídeos *Netflix_BarScene, Netflix_BoxingPractice, Netflix_Dancers, Netflix_Narrator, Netflix_RitualDance, Netflix_ToddlerFountain, Netflix_WindAndNature, street_hdr_amazon*: **(4096x2160, 10 bits, 4:2:0, 60 quadros)**;

   (b) Vídeos *aspen, crowd_run, ducks_take_off, guitar_hdr_amazon, life, Netflix_Aerial, Netflix_Boat, Netflix_Crosswalk, Netflix_FoodMarket, Netflix_PierSeaside, Netflix_SquareAndTimelapse, Netflix_TunnelFlag, old_town_cross, pan_hdr_amazon, park_joy, pedestrian_area, rush_field_cuts, rush_hour, seaplane_hdr_amazon, station2, touchdown_pass*: **(1920x1080, 8 bits, 4:2:0, 60 quadros)**;

   (c) Vídeos *boat_hdr_amazon, dark, FourPeople, gipsrestat, Johnny, KristenAndSara, Netflix_DinnerScene, Netflix_DrivingPOV, Netflix_FoodMarket2, Netflix_RollerCoaster, Netflix_Tango, rain_hdr_amazon, vidyo1, vidyo3, vidyo4*: **(1280x720, 8 bits, 4:2:0, 120 quadros)**;

   (d) Vídeos *blue_sky, controlled_burn, desktop2, kirland, mmstationary, niklas, rain2_hdr_amazon, red_kayak, riverbed, shields2, snow_mnt, speed_bag, stockholm, tacomanarrows, thaloundeskmtg, water_hdr_amazon*: **(640x360, 8 bits, 4:2:0, 120 quadros)**;

   (e) Vídeos *bqfree, bqhighway, bqzoom, chairlift, dirtbike, mozzoom*: **(426x240, 8 bits, 4:2:0, 120 quadros)**;

   (f) Vídeos *CSGO, DOTA2, MINECRAFT, STARCRAFT, EuroTruckSimulator2, Hearthstone, wikipedia, pvq_slideshow*: **(1920x1080, 8 bits, 4:4:4 ou 4:2:0, 60 quadros)**;

3. **Grupo de teste *objective-2-fast***: É apresentado no item 5.2.3 do documento e é um subconjunto do grupo *objetivo-2-slow* e tem o objetivo de reduzir o tempo de execução dos testes.

   (a) Vídeos *aspen, ducks_take_off, life, Netflix_Aerial, Netflix_Boat, Netflix_FoodMarket, Netflix_PierSeaside, Netflix_SquareAndTimelapse, Net-*

flix_TunnelFlag, rush_hour, seaplane_hdr_amazon, touchdown_pass: **(1920x1080, 8 bits, 4:2:0, 60 quadros)**;

(b) Vídeos *boat_hdr_amazon, dark, gipsrestat, KristenAndSara, Netflix_DrivingPOV, Netflix_RollerCoaster, vidyo1, vidyo4*: **(1280x720, 8 bits, 4:2:0, 120 quadros)**;

(c) Vídeos *blue_sky, controlled_burn, kirland, niklas, rain2_hdr_amazon, red_kayak, riverbed, shields2, speed_bag, thaloundeskmtg*: **(640x360, 8 bits, 4:2:0, 120 quadros)**;

(d) Vídeos *bqfree, bqzoom, dirtbike*: **(426x240, 8 bits, 4:2:0, 120 quadros)**;

(e) Vídeos *DOTA2, MINECRAFT, STARCRAFT, wikipedia*: **(1920x1080, 8 bits, 4:2:0, 60 quadros)**;

Ainda na seção 5 do documento DAEDE; NORKIN; BRAILOVSKIY (2020) é recomendada a configuração comum para testes do software referência libaom (AOM, 2020), além de serem definidos quatro modos de operação:

1. **Configuração Comum**: Os codificadores devem ser configurados para suas melhores configurações ao serem comparados uns com os outros e, o documento recomenda os seguintes parâmetros:

```
-codec=av1 -ivf -frame-parallel=0 -tile-columns=0
-cpu-used=0 -threads=1
```

   Para os seguintes modos de operação recomendados na CTC, são parâmetros adicionais a configuração comum.

2. **CQP de Alta Elevada**: É usado para avaliar modificações incrementais do codificador e é adequado para comparar codificadores com ferramentas semelhantes. Ele permite recursos de codificação com atraso intrínseco de quadros DAEDE; NORKIN; BRAILOVSKIY (2020). Os parâmetros a serem utilizados adicionais neste modo são:

```
-end-usage=q -cq-level=<cq> -auto-alt-ref=2
```

3. **CQP de Baixa Latência** : CQP de baixa latência, igualmente ao modo alta latência, é usado para avaliar modificações incrementais do codificador e é adequado para comparar codificadores com ferramentas de codificação semelhantes. Mas, neste caso, o atraso intrínseco de quadros deve ser definido como zero DAEDE; NORKIN; BRAILOVSKIY (2020). Os parâmetros á serem utilizados adicionais neste modo são:

```
-end-usage=q -cq-level=<cq> -lag-in-frames=0
```

4. **Alta Latência Irrestrita**: O modo de alta latência irrestrita é usado para executar o codificador no melhor modo de qualidade disponível, escolhendo o controle de qualidade por *bitrate* (CQ). Além de permitir codificação em duas passadas (*–passes=2*) quando suportado pelo codificador e, definir de forma opcional a taxa de bits alvo (bits por segundo, *–target-bitrate=<b>*) (DAEDE; NORKIN; BRAILOVSKIY, 2020). Parâmetros adicionais para uso deste modo são:

```
-end-usage=<q> -cq-level=<cq> -lag-in-frames=25 -auto-alt-ref=2
-target-bitrate=<b> -passes=2
```

5. **Baixa Latência Irrestrita**: O modo de baixa latência irrestrita é semelhante ao anterior, mas diferente do que o nome sugere, este modo possui algumas restrições no uso, não sendo permitido o recurso de atraso intrínseco de quadros e *buffering* de quadros (DAEDE; NORKIN; BRAILOVSKIY, 2020). Abaixo estão os parâmetros adicionais usados neste modo.

```
-end-usage=<q> -cq-level=<cq> -lag-in-frames=0
```

Na Tabela 24, estão descritos os parâmetros citados para a configuração do software de referência do AV1 na configuração comum, modos de operações e outros mais utilizados:

Tabela 24 – Descrição dos parâmetros de configuração do software de referência do AV1

Parâmetro	Descrição
codec	Define qual codificador deve ser utilizado
ivf	Habilita a exportação do arquivo com as características do vídeo
frame-parallel	Define quantos quadros são processados paralelamente
tile-columns	Define quantas colunas são processadas paralelamente
cpu-used	Ajuste de preferência da relação Qualidade-Velocidade da codificação, sendo zero melhor qualidade e, até oito maior velocidade
threads	Número de execução simultâneas na CPU
end-usage	Define o tipo de controle de quantização, no AV1 apenas é permitido 'q' (CQ)
cq-level	Define qual CQ será utilizado na codificação
auto-alt-ref	Define quantos quadros do tipo 'ALTREF' são permitidos
lag-in-frames	Define quantos quadros antes da codificação iniciar podem ser armazenados (*buffer*)
target-bitrate	Define taxa de bits alvo esperada em bits por segundo
passes	Define a estratégia de controle de taxa qualidade, número de vezes que o fluxo de dados é analisado (usualmente 1 ou 2)
kf	Compõem dois parâmetros kf-min-dist e kf-max-dist, que definem o intervalo de *KeyFrame*
verbose	Fornece detalhes adicionais da codificação
psnr	Habilita a exportação das métricas de PSNR

APÊNDICE C – Relação Detalhada Dos Resultados Obtidos

ORIGINAL	MODELO	TS %	TS Médio %	BD-Rate %	BD-Rate/TS
KristenAndSara_1280x720_60_120f					
Stream 0 PSNR (Overall/Avg/Y/U/V) 44.708 44.708 43.748 47.262 48.125 17126744 bps 4957901 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 44.664 44.664 43.694 47.273 48.129 18687064 bps 2388221 ms	51,82999822	44,33866379	14,16579101	0,3194907064
Stream 0 PSNR (Overall/Avg/Y/U/V) 42.079 42.080 41.033 45.020 46.042 9677224 bps 3882429 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 41.985 41.986 40.916 45.042 46.101 10709600 bps 1931033 ms	50,26224562			
Stream 0 PSNR (Overall/Avg/Y/U/V) 38.992 38.993 37.843 42.561 43.582 5311928 bps 3791373 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 38.820 38.821 37.631 42.668 43.692 5886440 bps 2422722 ms	36,099086			
Stream 0 PSNR (Overall/Avg/Y/U/V) 35.310 35.311 34.070 39.750 40.305 2740880 bps 2939197 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 34.995 34.997 33.706 39.809 40.446 2999800 bps 1924995 ms	34,50609129			
FourPeople_1280x720_60_120f					
Stream 0 PSNR (Overall/Avg/Y/U/V) 44.047 44.047 43.073 46.534 47.715 23181176 bps 7161343 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 43.976 43.976 42.988 46.529 47.716 24825552 bps 2688836 ms	62,45346718	60,23064914	10,65983358	0,1769835413
Stream 0 PSNR (Overall/Avg/Y/U/V) 41.209 41.209 40.147 44.110 45.410 13465280 bps 5936796 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 41.073 41.073 39.975 44.172 45.521 14486992 bps 2277917 ms	61,6305327			
Stream 0 PSNR (Overall/Avg/Y/U/V) 37.867 37.867 36.676 41.676 42.788 7550312 bps 6140958 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 37.642 37.642 36.405 41.750 42.957 8026256 bps 2507338 ms	59,17024673			
Stream 0 PSNR (Overall/Avg/Y/U/V) 33.828 33.828 32.508 38.734 39.747 3772312 bps 5271205 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 33.542 33.542 32.194 38.739 39.712 3929760 bps 2273497 ms	56,86950138			
Johnny_1280x720_60_120f					
Stream 0 PSNR (Overall/Avg/Y/U/V) 44.415 44.415 43.310 47.886 48.491 14491056 bps 5065079 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 44.378 44.378 43.267 47.895 48.484 15737760 bps 2425134 ms	52,12050987	47,42718937	14,49213673	0,3055660038
Stream 0 PSNR (Overall/Avg/Y/U/V) 42.087 42.087 40.931 45.914 46.460 7600952 bps 3659541 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 42.001 42.001 40.838 45.876 46.421 8410112 bps 1898044 ms	48,13436986			
Stream 0 PSNR (Overall/Avg/Y/U/V) 39.412 39.412 38.216 43.460 44.130 4009464 bps 3422467 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 39.243 39.244 38.017 43.522 44.182 4444312 bps 1853561 ms	45,841377			
Stream 0 PSNR (Overall/Avg/Y/U/V) 36.080 36.080 34.813 40.757 41.257 1949312 bps 2715200 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 35.828 35.828 34.535 40.691 41.264 2151616 bps 1636783 ms	39,71777401			
Netflix_DinnerScene_1280x720_60fps_8bit_420_120f					
Stream 0 PSNR (Overall/Avg/Y/U/V) 46.117 46.117 45.737 45.702 48.850 4719720 bps 6178568 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 46.085 46.085 45.689 45.705 48.864 5107288 bps 2790188 ms	54,8408628	55,40813596	10,01466916	0,1807436577
Stream 0 PSNR (Overall/Avg/Y/U/V) 44.851 44.851 44.278 45.131 47.878 2484776 bps 4212776 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 44.755 44.755 44.151 45.138 47.881 2630968 bps 1795999 ms	57,36780213			
Stream 0 PSNR (Overall/Avg/Y/U/V) 42.939 42.939 42.136 44.151 46.591 1268144 bps 3516779 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 42.788 42.788 41.954 44.131 46.568 1322152 bps 1532778 ms	56,41528797			
Stream 0 PSNR (Overall/Avg/Y/U/V) 40.234 40.235 39.212 42.581 44.737 596320 bps 2540844 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 40.032 40.033 38.982 42.534 44.686 613536 bps 1215936 ms	52,14440556			
Netflix_FoodMarket2_1280x720_60fps_8bit_420_120f					
Stream 0 PSNR (Overall/Avg/Y/U/V) 42.543 42.563 41.833 44.317 44.776 42404920 bps 8707410 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 42.452 42.470 41.716 44.317 44.779 43709776 bps 3369708 ms	61,30068528	58,3122702	5,853362753	0,1003796068
Stream 0 PSNR (Overall/Avg/Y/U/V) 38.842 38.867 37.886 41.745 42.138 22813304 bps 7338244 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 38.704 38.730 37.721 41.743 42.160 23482176 bps 2921605 ms	60,18659232			
Stream 0 PSNR (Overall/Avg/Y/U/V) 35.198 35.223 34.038 39.436 39.590 11352176 bps 7759707 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 35.049 35.076 33.858 39.538 39.680 11606296 bps 3293859 ms	57,55176065			
Stream 0 PSNR (Overall/Avg/Y/U/V) 31.522 31.549 30.259 36.692 36.619 4947680 bps 6132808 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 31.316 31.345 30.024 36.785 36.691 4986208 bps 2895371 ms	52,78882039			
Netflix_Tango_1280x720_60fps_8bit_420_120f					
Stream 0 PSNR (Overall/Avg/Y/U/V) 44.367 44.367 43.461 48.094 46.259 20117120 bps 8105023 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 44.296 44.296 43.375 48.098 46.250 21568376 bps 2676120 ms	66,98195674	65,50860271	11,32092871	0,1728159087
Stream 0 PSNR (Overall/Avg/Y/U/V) 41.524 41.525 40.487 46.214 43.888 10916752 bps 6676758 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 41.393 41.394 40.331 46.293 43.857 11724232 bps 2191798 ms	67,17272065			
Stream 0 PSNR (Overall/Avg/Y/U/V) 38.399 38.400 37.263 43.958 41.200 5681272 bps 6809139 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 38.226 38.226 37.047 44.181 41.249 6073872 bps 2332992 ms	65,73734212			
Stream 0 PSNR (Overall/Avg/Y/U/V) 34.914 34.914 33.724 40.951 38.024 2736560 bps 5480613 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 34.635 34.636 33.400 41.091 38.039 2905128 bps 2136440 ms	61,01822917			
rain_hdr_amazon_720p					
Stream 0 PSNR (Overall/Avg/Y/U/V) 42.342 42.342 41.327 45.467 45.710 15191982 bps 9243398 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 42.287 42.288 41.260 45.484 45.729 15357084 bps 3621996 ms	60,81531921	56,21441507	3,330255797	0,05924202525
Stream 0 PSNR (Overall/Avg/Y/U/V) 38.845 38.846 37.568 43.447 44.296 7318511 bps 6742395 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 38.751 38.752 37.447 43.638 44.383 7381399 bps 2909175 ms	56,85249826			
Stream 0 PSNR (Overall/Avg/Y/U/V) 35.543 35.545 34.159 40.844 42.503 3247654 bps 6733317 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 35.439 35.440 34.030 41.048 42.651 3251033 bps 3089636 ms	54,11420552			
Stream 0 PSNR (Overall/Avg/Y/U/V) 32.350 32.351 30.878 38.478 40.886 1240624 bps 4774443 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 32.236 32.237 30.759 38.458 40.795 1230347 bps 2417406 ms	49,36779013			
vidyo3_720p_60fps_120f					
Stream 0 PSNR (Overall/Avg/Y/U/V) 44.809 44.819 43.771 48.432 48.090 16998248 bps 5591527 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 44.757 44.767 43.714 48.412 48.057 18201504 bps 2237534 ms	59,9834893	58,15423911	11,44901736	0,1968733069
Stream 0 PSNR (Overall/Avg/Y/U/V) 42.135 42.138 40.990 46.746 45.672 9687888 bps 4586414 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 42.041 42.044 40.880 46.800 45.656 10539360 bps 1852931 ms	59,59956951			
Stream 0 PSNR (Overall/Avg/Y/U/V) 38.791 38.791 37.522 44.794 42.877 5379496 bps 4671602 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 38.584 38.585 37.276 44.939 42.995 5804360 bps 1985763 ms	57,49289002			
Stream 0 PSNR (Overall/Avg/Y/U/V) 34.696 34.697 33.293 42.152 40.013 2591696 bps 3981466 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 34.362 34.363 32.923 42.311 40.086 2701584 bps 1803751 ms	54,69631035			
crowd_run_1080p50_60f					
Stream 0 PSNR (Overall/Avg/Y/U/V) 39.895 39.896 39.536 40.536 40.908 152967966 bps 30750459 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 39.877 39.877 39.510 40.537 40.911 154574633 bps 13162267 ms	57,1965186	54,8246225	2,17464109	0,03966540928
Stream 0 PSNR (Overall/Avg/Y/U/V) 36.152 36.153 35.526 37.643 37.895 81208380 bps 22627139 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 36.120 36.120 35.483 37.642 37.903 82201206 bps 11162857 ms	50,66606963			
Stream 0 PSNR (Overall/Avg/Y/U/V) 32.542 32.543 31.608 35.265 35.510 39971033 bps 24693786 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 32.467 32.468 31.519 35.256 35.499 40229126 bps 10821568 ms	56,17695885			
Stream 0 PSNR (Overall/Avg/Y/U/V) 28.881 28.882 27.751 32.565 33.067 16985746 bps 18889342 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 28.745 28.746 27.596 32.545 33.064 16816166 bps 8655682 ms	54,17690039			
guitar_hdr_amazon_1080p					

Stream 0 PSNR (Overall/Avg/Y/U/V) 47.224 47.225 46.329 48.956 51.085 6071151 bps 10860195 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 47.198 47.199 46.296 48.952 51.107 6371532 bps 5849710 ms	46,1362342	42,0889327	8,096482686	0,1923660727
Stream 0 PSNR (Overall/Avg/Y/U/V) 45.129 45.131 44.113 47.490 49.538 3109319 bps 8130366 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 45.064 45.066 44.034 47.479 49.578 3275652 bps 4672133 ms	42,53477642			
Stream 0 PSNR (Overall/Avg/Y/U/V) 42.582 42.585 41.457 45.614 47.581 1578920 bps 7502282 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 42.446 42.449 41.302 45.594 47.597 1651935 bps 4275490 ms	43,01080658			
Stream 0 PSNR (Overall/Avg/Y/U/V) 39.525 39.530 38.336 43.073 44.899 773424 bps 5267309 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 39.286 39.292 38.074 43.011 44.846 797643 bps 3595310 ms	31,74294502			
Netflix_Aerial_1920x1080_60fps_8bit_420_60f					
Stream 0 PSNR (Overall/Avg/Y/U/V) 41.436 41.437 40.684 43.344 43.675 124599040 bps 26999118 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 41.411 41.412 40.655 43.336 43.670 125741864 bps 14412410 ms	46,61895992	46,08979804	2,499127292	0,05422300376
Stream 0 PSNR (Overall/Avg/Y/U/V) 37.373 37.375 36.241 40.841 41.870 60174128 bps 23369867 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 37.329 37.331 36.192 40.823 41.870 60775208 bps 12731383 ms	45,52222741			
Stream 0 PSNR (Overall/Avg/Y/U/V) 33.739 33.742 32.371 38.841 40.620 23895328 bps 23451877 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 33.672 33.675 32.297 38.839 40.643 24103264 bps 12533843 ms	46,55505399			
Stream 0 PSNR (Overall/Avg/Y/U/V) 30.697 30.703 29.222 36.984 39.282 7196160 bps 17902124 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 30.617 30.623 29.139 36.956 39.263 7233616 bps 9770411 ms	45,42317437			
station2_1080p25_60f					
Stream 0 PSNR (Overall/Avg/Y/U/V) 42.985 42.987 42.075 45.645 45.803 17101133 bps 20981474 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 42.959 42.961 42.039 45.655 45.831 18118423 bps 10390687 ms	50,47684924	49,6965253	11,60452329	0,2335077396
Stream 0 PSNR (Overall/Avg/Y/U/V) 40.696 40.699 39.782 43.594 43.335 8363733 bps 16135692 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 40.645 40.648 39.712 43.616 43.380 9016486 bps 8166920 ms	49,38599473			
Stream 0 PSNR (Overall/Avg/Y/U/V) 38.053 38.057 36.997 41.891 41.242 3923300 bps 15042799 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 37.966 37.970 36.893 41.904 41.247 4300880 bps 7431950 ms	50,59463335			
Stream 0 PSNR (Overall/Avg/Y/U/V) 34.971 34.974 33.724 40.174 39.380 1636653 bps 11344505 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 34.778 34.781 33.513 40.107 39.358 1784603 bps 5955398 ms	47,50411763			
pan_hdr_amazon_1080p					
Stream 0 PSNR (Overall/Avg/Y/U/V) 46.772 46.814 46.114 48.310 49.085 10741284 bps 11177215 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 46.748 46.789 46.077 48.324 49.114 11032867 bps 7949388 ms	28,8786339	27,30704376	4,554434556	0,1667860716
Stream 0 PSNR (Overall/Avg/Y/U/V) 43.753 43.779 42.857 45.952 47.269 5691406 bps 9271448 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 43.690 43.717 42.776 45.976 47.304 5833775 bps 6807537 ms	26,57525556			
Stream 0 PSNR (Overall/Avg/Y/U/V) 40.407 40.428 39.349 43.057 45.312 2827460 bps 8982107 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 40.320 40.340 39.248 43.018 45.334 2903000 bps 6542630 ms	27,1592957			
Stream 0 PSNR (Overall/Avg/Y/U/V) 36.799 36.819 35.564 40.478 43.234 1201719 bps 7107462 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 36.665 36.685 35.421 40.420 43.177 1227259 bps 5261166 ms	25,97686769			
old_town_cross_1080p50_60f					
Stream 0 PSNR (Overall/Avg/Y/U/V) 39.124 39.124 38.663 39.319 41.367 91808366 bps 30170811 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 39.098 39.098 38.626 39.327 41.378 93414340 bps 17218360 ms	42,93040383	42,85339171	3,489819876	0,08143625829
Stream 0 PSNR (Overall/Avg/Y/U/V) 36.565 36.566 35.715 38.401 39.762 34420240 bps 22247458 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 36.541 36.541 35.681 38.417 39.777 35154426 bps 12586405 ms	43,42542415			
Stream 0 PSNR (Overall/Avg/Y/U/V) 34.430 34.430 33.384 37.499 38.226 16042706 bps 19200820 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 34.370 34.370 33.309 37.522 38.252 16278073 bps 10998845 ms	42,71679543			
Stream 0 PSNR (Overall/Avg/Y/U/V) 31.662 31.662 30.427 36.156 36.494 6408520 bps 14495572 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 31.560 31.560 30.314 36.136 36.489 6399526 bps 8407998 ms	41,9960937			
park_joy_1080p50_60f					
Stream 0 PSNR (Overall/Avg/Y/U/V) 40.505 40.528 40.193 41.026 41.587 140775753 bps 24958165 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 40.486 40.509 40.167 41.025 41.586 141199526 bps 15148451 ms	39,30462837	37,7732578	1,046933134	0,02771625206
Stream 0 PSNR (Overall/Avg/Y/U/V) 36.587 36.650 36.158 37.304 38.461 75231053 bps 21137118 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 36.556 36.619 36.118 37.298 38.460 75395120 bps 13523109 ms	36,02198275			
Stream 0 PSNR (Overall/Avg/Y/U/V) 32.834 32.921 32.016 34.776 36.686 35998840 bps 23325041 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 32.780 32.867 31.952 34.765 36.683 35947173 bps 13986193 ms	40,03786317			
Stream 0 PSNR (Overall/Avg/Y/U/V) 29.077 29.177 27.916 32.931 35.533 14389953 bps 16583534 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 28.981 29.081 27.810 32.915 35.532 14248720 bps 10859646 ms	34,51548988			
seaplane_hdr_amazon_1080p					
Stream 0 PSNR (Overall/Avg/Y/U/V) 43.871 43.874 43.013 45.551 47.406 25645797 bps 15332523 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 43.853 43.855 42.991 45.543 47.410 26203821 bps 7494445 ms	51,12060161	56,964556	3,922791132	0,0688637182
Stream 0 PSNR (Overall/Avg/Y/U/V) 40.178 40.182 39.033 43.159 45.653 12954338 bps 16164057 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 40.143 40.147 38.990 43.171 45.686 13311012 bps 6210954 ms	61,57552525			
Stream 0 PSNR (Overall/Avg/Y/U/V) 36.667 36.671 35.354 40.712 43.941 6042664 bps 16728994 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 36.596 36.600 35.273 40.740 43.940 6200484 bps 6190709 ms	62,99413461			
Stream 0 PSNR (Overall/Avg/Y/U/V) 33.274 33.277 31.841 38.489 42.112 2457024 bps 9128118 ms	Stream 0 PSNR (Overall/Avg/Y/U/V) 33.159 33.162 31.723 38.393 42.058 2491342 bps 4786308 ms	47,56522648			