Trabalho Final - Ferramentas de Inteligência Artificial

Caroline Souza Camargo[1], Yasmin Souza Camargo[1], Ulisses Brisolara Correa[1]

[1]Universidade Federal de Pelotas (UFPel) – Pelotas – RS – Brazil

{caroline.sc, yamin.sc, ulisses}@inf.ufpel.edu.br

Resumo. *Este trabalho aborda a otimização da seleção de transformadas na codificação de vídeo por meio do uso de aprendizado de máquina. O estudo envolve a construção e análise de um dataset extraído de vídeos codificados, a seleção e pré-processamento de dados, e a aplicação de métodos de aprendizado de máquina utilizando random forest. Os resultados indicam que a abordagem alcançou uma acurácia de 76%, evidenciando uma melhoria significativa na eficiência da compressão de vídeo.*

1. Introdução

Atualmente, há um grande aumento na demanda por conteúdo de vídeo, evidenciado pela popularidade de plataformas como YouTube, Facebook, Netflix, TikTok, entre outras. Estima-se que, até 2029, o tráfego de vídeo representará expressivos 76% do tráfego total de dados móveis (Jonsson et al., 2020).

Um vídeo pode ser visto como um grande número de imagens que são exibidas em sequência rápida e criam a ilusão de movimento. Cada uma dessas imagens é composta por milhares de pixels. Cada pixel armazena informações como a cor, intensidade luminosa e outros atributos visuais. Quando multiplicamos esses dados de todos esses pixels pelo número de imagens em um vídeo, a quantidade de dados se torna extremamente grande. Dessa maneira, se faz necessário ferramentas conhecidas como codificadores de vídeo, pois armazenar vídeos sem compressão seria inviável devido ao volume de dados gerado. Esses codificadores são responsáveis por comprimir os dados de vídeo de maneira eficiente, permitindo uma representação otimizada que facilita tanto a transmissão quanto o armazenamento. Em dispositivos móveis, a eficiência dos codecs de vídeo torna-se ainda mais crucial, impactando diretamente o tempo de codificação e a autonomia da bateria.

Entre os codificadores de vídeo mais recentes no mercado, que têm como alvo as resoluções de alta definição, destaca-se o formato Versatile Video Coding (H.266/VVC). Este codec oferece uma notável redução de 50% na taxa de compressão de bits em comparação com High Efficiency Video Coding (HEVC), sem comprometer a qualidade visual (Fraunhofer Institute for Telecommunications, 2020). No entanto, essa eficiência vem com um aumento significativo no custo computacional. De acordo com (Siqueira et al., 2020), a codificação VVC é de 10 a quase 16 vezes mais complexa que HEVC. Além disso, a etapa das transformadas e quantização é particularmente complexo, representando cerca de 28% da carga de processamento total do codificador para (Pakdaman et al., 2020). Em resoluções mais altas, a participação da etapa de transformação e quantização tende a aumentar, o que indica uma maior demanda por transformação e quantização em vídeos de alta resolução. Isto destaca que esta etapa específica representa uma demanda significativa de processamento na codificação de

vídeo. Neste contexto, compreender e otimizar essa etapa é fundamental para alcançar desempenhos satisfatórios em sistemas de compressão de vídeo.

Explorar técnicas de aprendizado de máquina torna-se uma opção promissora para reduzir a complexidade do processo, especialmente a respeito da etapa das transformadas. Esses métodos têm a capacidade de aprender padrões e tomar decisões otimizadas com base em grandes volumes de dados. Isso pode resultar em uma significativa economia de tempo e recursos, sem comprometer a qualidade da compressão e do vídeo final. Portanto, o desafio deste trabalho consiste em automatizar e aprimorar a seleção das transformadas, com o objetivo de reduzir significativamente o tempo de processamento e o consumo de recursos computacionais, como energia e memória.

2. Transformadas

As transformadas são estratégias que convertem a informação do domínio espacial para o domínio das frequências, com o objetivo de explorar o sistema visual humano e reduzir a quantidade de dados necessários para representar imagens. Essa etapa é crucial para o processo de compressão, uma vez que ela é responsável por preparar os dados para posteriormente eliminar ou atenuar as frequências menos relevantes durante a etapa de codificação de entropia (Lakshmanan, 2005). Isso possibilita alcançar elevadas taxas de compressão. Entre as novas técnicas empregadas pelo H.266/VVC na etapa das transformadas está a Multiple Transform Selection (MTS), que visa aprimorar o desempenho durante o processo de codificação. No MTS do H.266/VVC foram empregados três tipos de transformação: o DCT tipo II, o DST tipo VII e o DCT tipo VIII. Vale ressaltar, que esses tipos de transformadas podem ser aplicadas tanto na direção horizontal quanto vertical de um bloco, as combinações possíveis podem ser visualizadas na Tabela I.

Tabela I - Combinações de transformadas MTS Explicit

1° Transformada	2° Transformada
DCT-II	DCT-II
DST-VII	DST-VII
DCT-VIII	DST-VII
DST-VII	DCT-III
DCT-VIII	DCT-VIII
SKIP	SKIP

No VTM, são encontradas duas variantes distintas de MTS, conhecidas como MTS explícito e MTS implícito (Hamidouche et al., 2022). Para controlar o MTS explícito, certos sinalizadores são habilitados no conjunto de parâmetros de sequência de vídeo definido no VVC (Sequence Parameter Set – SPS) (JVET, 2020). Os sinalizadores são habilitados conforme especificado na Tabela II. Em análises anteriores desenvolvidas pelos próprios integrantes do grupo descobriu-se que o modo MTS que tem o maior consumo computacional e a melhor eficiência de codificação é o modo MTS = 3, sendo

o modo que oferece mais oportunidades de otimização. O esforço computacional neste modo específico nos permite explorar otimizações mais significativas no tempo de codificação.

Tabela II - Aplicação de MTS explicit e Implicit em modos MTS

Modo MTS	Intra	Inter
1	Explicit	-
2	Implicit	Explicit
3	Explicit	Explicit
4	Implicit	-

Quando o MTS explícito está ativado, o codificador realiza uma busca exaustiva para cada bloco de imagem, a fim de determinar a melhor combinação de transformadas a ser aplicada. Esse processo visa melhorar a compressão e a qualidade final do vídeo. No entanto, para cada bloco, podem existir até seis combinações possíveis de transformadas. Devido ao grande número de combinações e à quantidade de blocos que precisam ser processados, a escolha da melhor transformada pode ser extremamente exaustiva e custosa em termos de tempo e recursos.

Neste trabalho, propomos uma abordagem alternativa: em vez de testar todas as combinações possíveis para cada bloco, utilizamos modelos de aprendizado de máquina para prever a melhor combinação de transformadas com base nas características dos blocos.

3. Metodologia

3.1. Coleta de dados e construção do dataset

O dataset foi construído a partir de modificações no código do VVC na etapa de transformadas durante a codificação de vídeo. Essas alterações envolveram ajustes no processamento dos dados e na forma como as características eram extraídas e manipuladas pelo codificador. Os dados capturados durante a codificação foram armazenados em um arquivo CSV. Antes da implementação, realizamos um estudo detalhado para identificar o momento ideal de coleta dos dados e selecionar as features mais relevantes para o problema em questão.

O vídeo selecionado para a tarefa foi o YachtRide_1920x1080 de resolução Full HD, presente nas condições comuns de teste, considerando os QPs 22, 27, 32 e 37. A coleta das transformadas escolhidas foi realizada armazenando informações para cada bloco candidato no processo de codificação, resultando em uma quantidade massiva de informações, ultrapassando 100 GB de espaço ocupado pelos arquivos extraídos.

Esse volume de dados gerou problemas de memória ao tentar carregar o dataset, tornando necessário restringir a coleta. Para mitigar esse problema, delimitamos a coleta de dados especificamente para blocos de tamanho 32x32. Essa estratégia reduziu para um arquivo com tamanho de 2.9 GB. Apesar disso, foi necessário realizar um

balanceamento prévio e remover linhas duplicadas para garantir que o dataset final fosse manejável e não causasse estouro de memória. Para tentar lidar com problemas de falta de memória e otimizar o processo, começamos utilizando a biblioteca vaex, que é eficiente para manipulação de grandes volumes de dados.

O dataset construído para este trabalho contém um total de 3.831.516 linhas e 57 colunas, capturando diversas características cruciais durante a codificação de vídeo. O tamanho do arquivo após o balanceamento ficou 622 MB. As colunas incluem informações detalhadas sobre o tipo de predição, a localização e o tamanho dos blocos, a profundidade, o QP, modos específicos como MTS, LFNST, e ISP, além de informações sobre o movimento, como vetores de movimento e tipos de blocos. A coluna "MTSChosen" reflete a decisão final do modo de transformação que foi selecionado, que é feature central para o objetivo deste estudo.

3.2. Seleção e Pré-processamento dos Dados

Utilizamos inicialmente 50% dos dados disponíveis para os experimentos, selecionando aleatoriamente uma amostra representativa. Vale ressaltar que utilizamos o dataset inteiro apenas no treinamento final. Pois apesar de limitarmos e utilizarmos as estratégias comentadas, ainda enfrentamos problemas de tempo e memória. Após o pré-processamento inicial, convertemos o dataset para um DataFrame do Pandas para manipulação e análise mais detalhadas.

Realizamos um mapeamento das classes na coluna MTSChosen para reduzir a complexidade do problema. As classes foram agrupadas de maneira lóogica, de acordo com a sua ordem de ocorrência:

- [0, 1] foram agrupados na classe 0
- [2] foi mapeado para a classe 1
- [3, 4, 5] foram agrupados na classe 2

Dessa forma, por exemplo, quando o grupo zero for escolhido apenas a DCT2 e a SKIP serão testadas no codificador ao inves de todos os cinco tipos de MTS, esperando assim reduzir a complexidade desta etapa.

Para assegurar que nosso modelo não sofresse de viés de classe, realizamos um balanceamento por undersampling. Assim, garantimos que cada classe tivesse 638.586 amostras, totalizando um conjunto equilibrado.

Dividimos o dataset em conjuntos de treino (70%) e teste (30%). Removemos 3.144 linhas duplicadas e substituímos dados faltantes por um valor (999). Colunas com valores constantes foram identificadas e removidas, reduzindo a dimensionalidade do dataset. As colunas removidas foram aquelas que não apresentavam variação nos dados e, portanto, não contribuem para o modelo. Finalizando o tratamento dos dados, obtivemos um total de 35 colunas, o conjunto de treino com 1.337.886 linhas e o conjunto de teste com 574.728 linhas. Cada classe MTS obteve a seguinte distribuição: classe 2 com 446.774 linhas, classe 1 com 445.872 e classe 0 com 445.240 linhas.

Utilizamos o StandardScaler para normalizar os dados que ajusta os dados para média 0 e desvio padrão 1, garantindo que todas as características estivessem na mesma escala.

3.2. Seleção de colunas

Com 50% do conjunto de treino, utilizamos a seleção de características com base em modelos, primeiramente executamos o Selection from Model para termos uma noção do número de colunas escolhidas. Depois, aplicamos Recursive Feature Elimination para refinar as melhores características, passando como parâmetro o número de features para selecionar igual a 15 (valor encontrado pelo selection from model).

Como resultados obtivemos as seguintes colunas: 'frame', 'x', 'y', 'CUqp', 'mv_hor[0]', 'mv_ver[0]', 'mv_hor[1]', 'mv_ver[1]', 'AbsSumResidual', 'AbsSumUltimaLinha', 'AbsSumUltimaColuna', 'lefTopResidual', 'leftBottomResidual', 'rightTopResidual', 'rightBottomResidual'.'

3.3. Treinamento e Avaliação do Modelo

Primeiro treinamos o modelo Random Forest para n_estimators = 200 (Teste 1). Para otimizar o desempenho do modelo, realizamos uma busca aleatória de hiperparâmetros (Random Search) usando 30% do dataset. Os melhores parâmetros encontrados foram aplicados ao modelo, e o desempenho foi avaliado novamente (Teste 2).

Para tentar melhorar mais o desempenho, implementamos uma redução dimensional através da Análise Discriminante Linear (LDA) e realizamos um novo treinamento com os dados reduzidos (Teste 3).

No modelo final treinamos 100% do Random Forest com os melhores hiperparâmetros encontrados pelo Random Search (Final).

4. Resultados e Discussões

Durante nossas simulações realizamos três execuções, no qual testamos diferentes parâmetros. Os resultados são apresentados na tabela Tabela III e mostram uma análise detalhada do desempenho de diferentes testes realizados com o modelo Random Forest para a tarefa de classificação no dataset obtido a partir das transformadas do VVC.

Tabela III - Resultados de treinamento

	Precisão			**Recall**			**f1-score**			**Acurácia**
Classe MTS	**0**	**1**	**2**	**0**	**1**	**2**	**0**	**1**	**2**	
Teste 1	0.82	0.68	0.68	0.78	0.72	0.68	0.80	0.70	0.68	0.73
Teste 2	0.82	0.69	0.68	0.78	0.72	0.69	0.80	0.70	0.68	0.73
Teste 3	0.50	0.52	0.50	0.50	0.53	0.50	0.50	0.53	0.50	0.51
Final	**0.85**	**0.73**	**0.72**	**0.80**	**0.75**	**0.73**	**0.83**	**0.74**	**0.72**	**0.76**

Para analisar os resultados do treinamento do modelo, consideramos as métricas de precisão, recall, f1-score para cada classe, e a acurácia global do modelo.

No primeiro teste, utilizando o modelo Random Forest com 200 estimadores, os resultados foram variados. A precisão do modelo oscilou entre 0,68 e 0,82. A classe '0' apresentou a maior precisão, com 0,82, indicando que o modelo foi mais eficaz em evitar falsos positivos para essa classe. As classes '1' e '2' tiveram precisão mais baixa, em torno de 0,68, sugerindo uma tendência maior do modelo em gerar falsos positivos nessas categorias. O recall variou de 0,68 a 0,78, com a classe '0' se destacando novamente, enquanto as outras classes ficaram em torno de 0,72. O f1-score foi mais equilibrado, variando de 0,68 a 0,80, o que demonstra um desempenho razoável, especialmente na classe '0'. A acurácia global do modelo foi de 73%, o que é promissor, mas ainda deixa margem para melhorias.

No segundo teste, após a otimização dos hiperparâmetros via Random Search, observamos uma leve melhoria nos resultados. A precisão aumentou ligeiramente para as classes '1' (0,69), enquanto a classe '0' e '2' manteve-se nos resultados encontrados anteriormente. Isso indica uma melhoria leve. O recall das classe '2' também melhorou um pouco, chegando a 0,69. O f1-score manteve-se estável e a acurácia global permaneceu em 73%, mostrando que a otimização dos hiperparâmetros trouxe pequenas melhorias.

No terceiro teste, aplicamos o LDA para realizar uma redução dimensional antes de treinar o modelo. Os resultados mostraram uma queda significativa no desempenho do modelo. A precisão caiu consideravelmente para todas as classes, variando de 0,50 a 0,52, o que indica que o modelo teve dificuldades em manter a capacidade de classificação após a redução dimensional. O recall também diminuiu, variando de 0,50 a 0,53, com a classe '1' apresentando o maior valor, mas ainda muito inferior aos testes anteriores. O f1-score ficou entre 0,50 e 0,53, confirmando a perda de desempenho geral. A acurácia global caiu para 51%, mostrando que a redução dimensional não foi eficiente neste caso.

No último teste, aplicamos os melhores hiperparâmetros do Teste 2 ao conjunto completo de dados (100%). Os resultados mostraram a melhor performance entre todos os testes. A precisão aumentou para todas as classes, atingindo 0,85 para a classe '0', 0,73 para a classe '1' e 0,72 para a classe '2'. Isso reflete uma melhoria significativa na capacidade do modelo de evitar falsos positivos. O recall também melhorou, com a classe '0' chegando a 0,80, enquanto as classes '1' e '2' ficaram em 0,75 e 0,73, respectivamente. O f1-score variou entre 0,72 e 0,83, com a classe '0' apresentando o melhor desempenho geral. A acurácia global atingiu 76%, a maior entre todos os testes, demonstrando que o modelo conseguiu aproveitar plenamente o volume de dados e aos ajustes utilizados para otimizar o desempenho.

4.1. Análise da Curva ROC e Validação Cruzada

Na Figura 1 pode-se observar a área sob a curva ROC, no qual é possível avaliar o desempenho do modelo. Quanto mais a curva estiver próxima do canto superior esquerdo, melhor o desempenho. Foram obtidos AUCs de 0,87 para a classe '0', 0,81 para a classe '1' e 0,79 para a classe '2'. Esses valores indicam que o modelo é

particularmente eficaz em identificar a classe '0', enquanto o desempenho é um pouco menor para as outras classes.

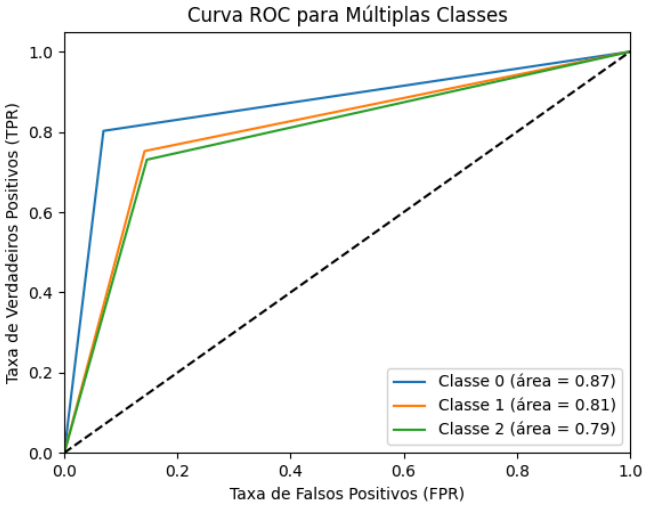

Figura I - Comparação de Desempenho por Classe - Curva ROC

Além disso, realizamos a validação cruzada K-Fold para avaliar a robustez e a capacidade de generalização do modelo. Os resultados mostraram uma precisão média de 0,7466, um recall médio de 0,7445 e um f1-score médio de 0,7453, refletindo um equilíbrio com os resultados encontrados anteriormente. Esses resultados confirmam que o modelo tem um desempenho consistente e eficiente, com boas capacidades de generalização.

Para resultados detalhados da saida dos experimentos pode ser encontrados neste documento:
https://docs.google.com/document/d/1igWK8oeoGroZiRb9KcqjnfEM3InbtgdgOq4lkLthXWk/edit?usp=sharing

5. Conclusão

Neste trabalho, exploramos a otimização da seleção de transformadas na codificação de vídeo por meio de aprendizado de máquina. Inicialmente, geramos um dataset, realizamos a análise dos dados e aplicamos modelos de Random Forest, com ajustes de hiperparâmetros e redução de dimensionalidade para maximizar o desempenho.

Os resultados obtidos foram promissores, com a abordagem proposta superando o desempenho esperado de 33% de acurácia aleatória por classe, alcançando uma acurácia global de 76%. Essa melhoria significativa demonstra a eficácia do modelo em otimizar a compressão de vídeo, equilibrando desempenho e eficiência. Além disso, esses resultados são apenas um ponto de partida, com potencial para futuras otimizações. Como trabalhos futuros, pretendemos validar o modelo utilizando validação cruzada e explorar novas estratégias para aprimorar ainda mais os resultados obtidos.

6. Referenciass

Fraunhofer Institute for Telecommunications. (2020) “Fraunhofer HHI is proud to present the new state-of-the-art in global video coding: H.266/VVC brings video transmission to new speed.” Fraunhofer Institute for Telecommunications. Available online: https://newsletter.fraunhofer.de/-viewonline2/17386/465/11/14SHcBTt/V44RELLZBp/1

Pakdaman, F., Adelimanesh, M. A., Gabbouj, M., and Hashemi, M. R. (2020) “Complexity analysis of next-generation VVC encoding and decoding.” In: 2020 IEEE International Conference on Image Processing (ICIP), pp. 3134–3138.

Peter Jonsson, Patrik Cerwall, Anette Lundvall, David von Koch, and Steven Davis, “Ericsson mobility report,” Nov 2023

Lakshmanan, S. (2005) “Statistical Methods for Image Segmentation.” In: A. Bovik (Ed.), Handbook of Image and Video Processing (2nd ed.), Academic Press, pp. 443-XV. ISBN: 9780121197926. DOI: 10.1016/B978-012119792-6/50090-5.

Hamidouche, W., Biatek, T., Abdoli, M., François, E., Pescador, F., Radosavljevic, M., Menard, D., and Raulet, M. (2022) “Versatile video coding standard: A review from coding tools to consumers deployment.” IEEE Consumer Electronics Magazine, vol. 11, no. 5, pp. 10–24.

JVET. (2020) “VVC software VTM.” Available online: https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftwareVTM

Icaro Siqueira, Guilherme Correa, and Mateus Grellert, “Rate-distortion and complexity comparison of hevc and vvc video encoders,” in 2020 IEEE 11th Latin American Symposium on Circuits & Systems (LASCAS), 2020, pp. 1–4