Accelerating VVC Inter-Frame Coding: A Lightweight CNN for Fast QTMT Partitioning

Jui-Chen Luo*, Jiann-Jone Chen*, Tien-Ying Kuo†, Yi-Fan Wu*, and Kai-Jie Zhang*
* National Taiwan University of Science and Technology
E-mail: jjchen@mail.ntust.edu.tw Tel/Fax: +886-2-27376669/27376699
† National Taipei University of Technology
E-mail: tykuo@ntut.edu.tw Tel: +886-2-27712171ext:2117

***Abstract*—The H.266/Versatile Video Coding (VVC) standard offers 30%−−50% higher compression efficiency than its predecessor, H.265/HEVC, while maintaining comparable visual quality. However, this improvement comes at the cost of substantially higher computational complexity, posing challenges for real-time encoding. In particular, VVC's Quad-Tree plus Multi-type Tree (QTMT) block partitioning, high-precision motion compensation, and multi-directional prediction contribute to encoding times that are 8 to 10 times longer than those of HEVC. To address this issue, we propose a lightweight, adaptive information (LAI-CNN), a compact CNN-based model featuring a three-stage prediction scheme that integrates VVC inter-coding structure constraints to make fast and accurate QTMT partitioning decisions, thereby reducing redundant RDO operations while maintaining a compact model architecture. Experimental results under the VTM-22.0 *Random Access* configuration show that LAI-CNN achieves a 36.35%–49.20% reduction in encoding time, with only a 1.36%–3.48% increase in Bjøntegaard Delta Bit Rate (BDBR). These results demonstrate that the proposed coding acceleration framework significantly reduces VVC encoding complexity, enabling practical deployment in real-time scenarios such as live streaming, interactive video services, and cloud-based processing.**

***Index Terms*—Video coding, Versatile Video Coding (VVC), QTMT, inter prediction, convolutional neural networks**

I. Introduction

While H.265/High-Efficiency Video Coding (HEVC) has been widely adopted, it falls short in efficiently handling ultra-high-definition (UHD) content such as 4K and 8K videos. To address these challenges, the Joint Video Exploration Team (JVET) introduced the H.266/Versatile Video Coding (VVC) standard, offering 30%–50% improved compression efficiency over HEVC while maintaining the same perceptual quality. VVC also supports advanced features like High Dynamic Range (HDR) and panoramic imaging, broadening its applicability. Video intra-coding exploits spatial redundancy within a frame, whereas inter-coding leverages temporal redundancy across frames. Since inter-coding plays a dominant role in compression efficiency, this work focuses on optimizing its computational cost. VVC employs exhaustive rate-distortion optimization (RDO), testing all possible splitting and prediction modes, which substantially increases encoding complexity. Compared to HEVC, VVC incorporates advanced features such as QTMT partitioning, high-precision motion estimation, and multi-directional prediction. While these techniques enhance compression, they also raise encoding times by up to 8–10×, posing challenges for real-time applications.

To address this, we propose LAI-CNN, a lightweight, adaptive CNN-based method for accelerating QTMT mode decisions in VVC inter-coding. Our approach analyzes VVC's structural constraints and integrates a compact CNN model to reduce unnecessary split evaluations. LAI-CNN significantly decreases computational load while preserving visual quality. Unlike heuristic-based methods, our model learns split patterns directly from the data, thereby enhancing its adaptability across diverse video content.

The rest of this paper is organized as follows: Section II reviews the VVC inter-coding algorithm and discusses related research. Section III presents the proposed LAI-CNN method in detail. Section IV provides experimental results and compares our method with state-of-the-art (SOTA) algorithms. Section V concludes the paper and discusses potential future research directions.

II. Related Research

VVC inter-coding evaluates each coding unit (CU) using Merge, Motion, or Intra prediction modes. Merge mode, often adopted in low-motion or simple-texture areas [1], reuses motion vectors (MVs) from neighboring CUs, thereby eliminating the need for additional motion estimation and improving efficiency. VVC enhances this with advanced Merge techniques [2], such as Combined Intra/Inter-Picture Prediction (CIIP), Affine Motion Estimation (AME), and Bi-Prediction with CU-Level Weighting (BCW). CIIP blends intra- and inter-predictions for complex textures; AME applies affine transformations to handle non-translational motion; and BCW optimizes bidirectional prediction via adaptive weighting. Motion prediction estimates MVs by matching blocks in reference frames. VVC introduces Adaptive Motion Vector Resolution (AMVR), which allows for variable MV precision (integer to eighth-pel), thereby balancing coding overhead and accuracy. Intra prediction, supporting 67 angular modes [3], is typically selected when motion-based modes fail to model complex content.

To adapt to diverse textures, VVC employs the QTMT structure that recursively splits CUs using quadtree (QT), binary tree (BT), and ternary tree (TT) strategies. QT and BT offer symmetric partitions, while TT uses a 1:2:1 split ratio. Fig. 1 illustrates the prediction constraints: (a) Type 1 predicts split or non-split (SP/NS); (b) Type 2 predicts the

979-8-3315-7206-8/25/$31.00

MTT split direction (horizontal or vertical); (c) Type 3 predicts among QT, HS, VS, and NS modes. This structure enforces VVC splitting rules while guiding fast mode decisions. Starting from a $CU_{128\times128}$, QT splitting yields four $CU_{64\times64}$ blocks, which can be further split recursively. BT splits yield two $CU_{128\times64}$ units. This recursive RDC-based process continues until the minimum CU size is met. VVC imposes constraints to manage this complexity—Table I summarizes these rules, such as limiting MTT to CUs smaller than 64×64 and capping the MTT depth at 3. Liu et al. [4] report that VVC inter-coding complexity is roughly 7x higher than HEVC, motivating the use of early termination and fast mode decision algorithms.

TABLE I: VVC Constraints on CU sizes.

Parameters	Intra Coding	Inter Coding
CTU size	128×128	128×128
MaxQTSize	128×128	128×128
MaxBTSize	32×32	128×128
MaxTTSize	32×32	64×64
MinQTSize	8×8	8×8
MinBTSize	4×4	4×4
MinTTSize	4×4	4×4
MaxMTTDepth	3	3

Intra-coding, which handles only spatial redundancy, has been a common focus for acceleration. Tang et al. [5] applied Laplacian filters to extract texture features for early CU split termination. Wu et al. [6] used SVMs to predict CU splits, while Li et al. [7] trained 19 CNNs, each for a specific CU size, and adopted early exit strategies. Tang et al. [8] introduced shape-adaptive CNNs, and Chen et al. [9] developed ACUCNN, integrating CU, neighbor pixels, QP, and CU size for accurate predictions. While effective, these efforts mostly target intra-coding and leave inter-coding less explored.

Recent inter-coding speedup techniques include handcrafted rules, classical machine learning, and deep learning. Handcrafted and ML-based methods rely on predefined features and lack adaptability. In contrast, deep learning enables data-driven feature extraction and flexible decision logic.

Deep learning approaches, such as MF-CNN[10] and MBMP-CNN[11], achieve substantial encoding time savings; however, they rely on complex, resource-intensive architectures that may not be feasible for real-time or low-power devices.

Similarly, Tissier et al.'s use of MobileNetV2 combined with LightGBM[12] requires a 14 MB model and manual feature engineering, complicating deployment and adaptation to unseen content [1]. Crucially, these methods often overlook VVC's structural constraints in their prediction logic, which can lead to invalid split mode selection and potential quality degradation. Our work addresses these limitations by proposing a compact, constraint-compliant CNN that relies on beyond handcrafted features, and is explicitly designed for lightweight, adaptive inter-coding acceleration.

III. Proposed Method

A. Compliant with QTMT Splitting Constraints

In VVC, the QTMT function evaluates CU splits in a fixed order: NS, QT, BH, BV, TH, and TV. Many fast inter-coding methods predict CU split types to skip low-probability MTT modes, but overlooking structural rules can cause problems. For example, a $CU_{64\times64}$ subject to MTT splitting only permits {NS, BH, BV, TH, TV}; if a model predicts QT, VTM overrides it with NS, leading to unnecessary computation and increased BDBR. Previous work [9] addressed this by boosting model accuracy or tuning thresholds, but these remain indirect solutions. We instead embed structural constraints into the prediction scheme to exclude invalid or low-probability modes while retaining VTM's flexibility. Based on VTM rules, three key constraints are applied:

1) CUs of size $\{128 \times 128, 128 \times 64, 64 \times 128\}$ permit only {QT, BH, BV}.
2) After an MTT split, sub-CUs cannot be further split by QT.
3) MTT-based splitting is limited to three consecutive depths.

Among these, Rule 1 applies specifically to inter-coding. Guided by these rules, our CNN ensures only valid split types are considered during inference.

B. Impact of Structural Constraints

The three structural rules in Section III-A are central to LAI-CNN's effectiveness. Without them, prediction models often generate invalid modes (e.g., QT after MTT), which VTM converts to NS–wasting computation and degrading R–D efficiency. By integrating the constraints at inference time, our approach suppresses invalid outputs, improves pruning precision, and stabilizes coding performance across sequences. As validated in Section IV, enforcing these rules reduced invalid QT-after-MTT predictions by about 15% and consistently improved normalized speedup $\Delta\tilde{T}$ by 2–4 points compared to unconstrained inference, while also lowering BDBR growth.

C. LAI-CNN Three-Stage Prediction Scheme

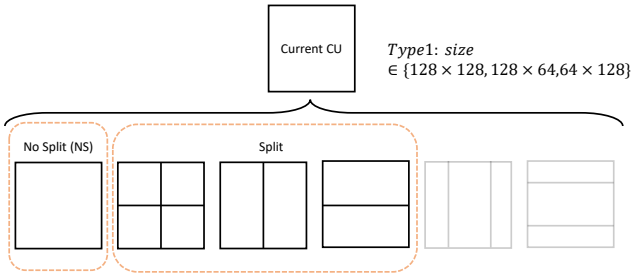
To reduce the complexity of VTM inter-coding, we propose a CNN-based model that predicts feasible CU split types while excluding those invalid under structural constraints. Our method adopts a three-stage prediction scheme that balances pruning efficiency with R–D performance. Fig. 1 shows the three stages and their prediction tasks.

To simplify decision-making, the original five splitting types {QT, BH, BV, TH, TV} are consolidated into three categories: {**QT**, **VS**={BV, TV}, **HS**={BH, TH}}. QT remains unchanged, while BV/TV and BH/TH are grouped by orientation. This reduces model complexity but preserves full decision space.

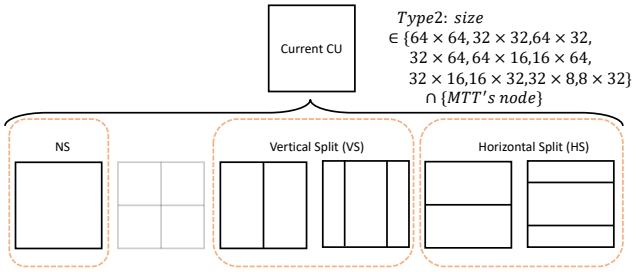
1) Type 1: Split or Non-Split Prediction: The first stage predicts whether a CU should be split (SP) or not (NS), enabling early termination. Since false NS predictions cause quality loss, we adopt a conservative strategy: a CU is labeled NS only with high prediction confidence. As shown in Fig. 1(a), texture differences between SP and NS regions provide effective cues.

2) Type 2: MTT Splitting Direction Prediction: If a CU is classified as SP, the second stage predicts the most likely MTT direction, **HS** or **VS**. This avoids testing both orientations, cutting complexity while preserving MTT flexibility (Fig. 1(b)).

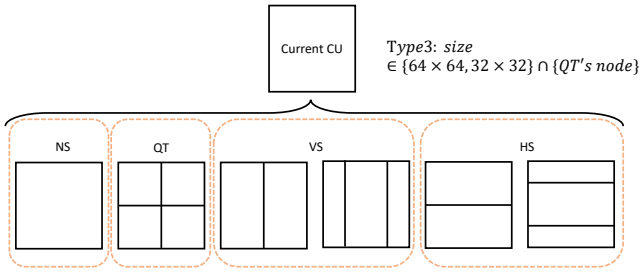
3) Type 3: Unrestricted CU Mode Prediction: When all split types are allowed, the third stage selects the best mode from {QT, HS, VS}. This case mainly arises in high-resolution or complex-texture areas, where flexible partitioning is essential (Fig. 1(c)).



(a) Type 1 prediction



(b) Type 2 prediction



(c) Type 3 prediction

Fig. 1: Prediction constraints and decision tasks for the three CU split types in LAI-CNN.

Fig. 2 shows the architecture of LAI-CNN. The network takes three feature sources: CU texture (luma and residual), motion (MV field), and side information. (1) Texture branch: Luma and residual, both of size $H \times W$, are concatenated into a $H \times W \times 2$ tensor F_c and fed to the left CNN branch. (2) Motion branch: MVs are defined on 4×4 blocks, giving an MV field of $\frac{H}{4} \times \frac{W}{4}$ with four channels (horizontal/vertical for forward/backward references). This F_{mv} is processed by two convolutional layers in the right branch. (3) Fusion: The outputs of both branches are concatenated and passed through an additional convolutional module for

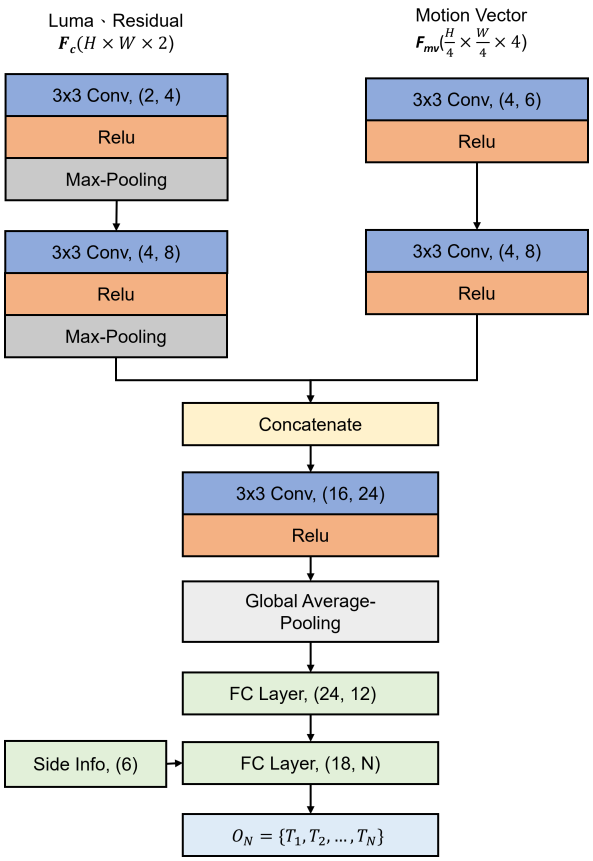


Fig. 2: Architecture of the proposed LAI-CNN.

high-level feature extraction. **Pooling and side parameters:** Global average pooling ensures invariance to CU size. Six side parameters—{QP, Height, Width, prediction mode, QT depth, MTT depth}—are concatenated with CNN features before the fully connected (FC) layer. The final FC layer outputs N classes depending on the prediction stage: $N = 2$ (Type 1: {NS, SP}), $N = 3$ (Type 2: {HS, VS, NS}), or $N = 4$ (Type 3: {NS, QT, VS, HS}). This design allows the network to handle variable CU sizes, exploit both spatial and motion cues, and make constraint-compliant split decisions.

D. LAI-CNN Training Strategy

The LAI-CNN training dataset comprises 15 UVG 4K videos: Beauty, CityAlley, FlowerKids, HoneyBee, Lips, ReadySetGo, ShakeNDry, YachtRide, Bosphorus, FlowerFocus, FlowerPan, Jockey, RaceNight, RiverBank, and Twilight.. We downsampled each video to resolutions corresponding to those in Classes B, C, D, and E, resulting in a total of $15 \times 4 = 60$ sequences for the training dataset. Since high-resolution frames provide more training samples than low-resolution ones, we adjusted the frame counts to ensure a balanced dataset across resolutions. Under the *Random Access* configuration, every 32 frames requires two intra-coding frames. To maximize the inter-coding training dataset, we set the number of encoding frames to $32 \times n + 1$, where

$n \in \{2, 3, 4, 5\}$, and the number of frames for Classes B, C, D, and E are 65, 97, 129, and 161 frames, respectively.

As our LAI-CNN has to manage different-sized CUs, we trained network models for the three types separately. Each network undergoes training for all CU sizes in that category, constituting one epoch, with the order of sizes shuffled randomly within each epoch. This effectively ensures the model's adaptable capability for different CU sizes. We used Adamoptimizer with a progressively decreasing learning rate per epoch. Since the training target is classification, we adopted cross-entropy as the loss function:

$$Loss = \frac{1}{S_B} \times \sum_{i=1}^{S_B} Y_i \times logP_i. \quad (1)$$

where S_B denotes batch size, Y_i is the ground truth label and P_i is the confidence of prediction. We split the dataset into non-overlapping training, validation, and test sets with a ratio of 8:1:1. After training, the three proposed models can yield 92%, 82% and 90% prediction accuracy, respectively.

E. CU Split Type Prediction

When the QTMT process selects NS, it indicates that the CU's texture, motion, or residuals are simple enough to be encoded as a single block. If LAI-CNN can correctly predict such CUs as NS, further split tests become redundant and can be skipped. Fig. 3 shows the proposed decision flow, where predictions at each stage guide selective pruning under VVC constraints. Each CU is first categorized into one of three types. For Type 1 and Type 2 CUs, an NS prediction skips all QT and MTT tests. If a Type 2 CU is predicted as VS, horizontal splits (BH, TH) are skipped while vertical ones (BV, TV) are retained; conversely, an HS prediction keeps BH/TH and skips BV/TV. For Type 3 CUs, all split modes are initially considered, but VS predictions disable BH/TH while HS predictions disable BV/TV. Thus, Type 3 always retains {NS, QT, VS} or {NS, QT, HS} as valid candidates.

Different applications may require different speed—quality trade-offs, which we regulate using confidence thresholds for NS predictions. The specific values of these thresholds, together with other experimental settings, are detailed in Section IV.

IV. Experimental Study

The experimental setup is summarized in Table II. We employed 24 video sequences from Classes A1 to E under the Common Test Conditions (CTC) [13] for encoding and performance comparison. For the programming environment, the models were trained using PyTorchand deployed via LibTorch (PyTorch C++ interface) for inference within a C++ framework. All algorithms and implementations were developed based on the VTM-22.0 reference software.

Test Classes in CTC: The 24 sequences used in our experiments follow the JVET Common Test Conditions (CTC) [13], which organize videos into five classes (A1, A2, B, C, D, and E). These classes differ in resolution, frame rate, and typical content, as summarized in Table III. Such differences are important for interpreting class-wise performance: higher-resolution classes (A1/A2) contain larger CUs, yielding more opportunities for pruning and greater acceleration, while low-resolution Class D offers fewer CU partitions and thus limited speedup. Class E, despite its higher resolution, often contains static content, so its acceleration trend differs from purely resolution-driven effects.

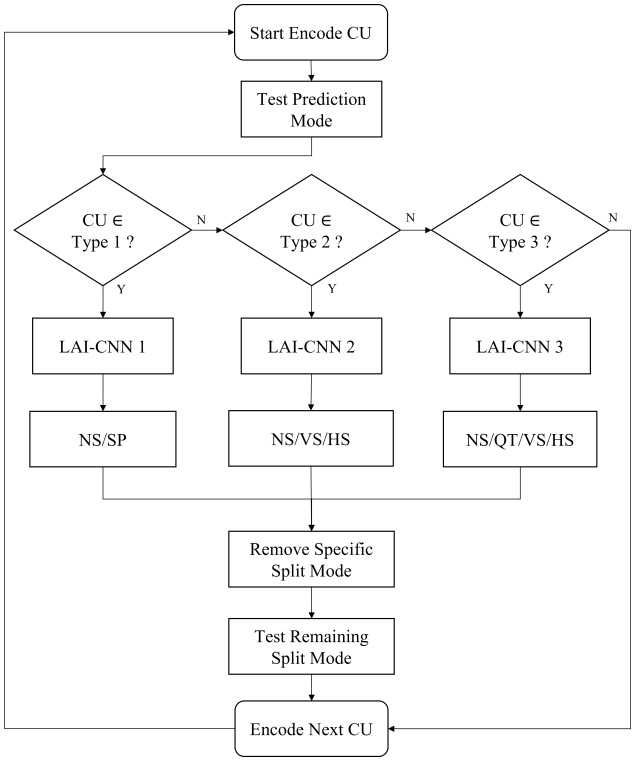


Fig. 3: Control flow of the proposed CU split type decision scheme.

TABLE II: Experimental Setup.

System Setting and Configuration	
CPU	AMD Ryzen × 16-cores × 32
RAM	64GB 3200MHz
GPU	NVIDIA GeForce RTX 3060
SSD	TEAMGROUP MP34 2TB
OS	Ubuntu 20.04.5 LTS
Reference Software	VTM Encoder Version 22.0
Configuration File	encoder_random_access.cfg
QP	22, 27, 32, 37
libtorch/pytorch	2.0.1/2.1.0
CUDA	12.2

We evaluate performance using BDBR, time-saving ratio ΔT, and normalized speedup $\Delta \tilde{T}$, which are calculated as:

$$\Delta T = \frac{1}{n} \sum_{i=1}^{n} \frac{T_o(q_i) - T_p(q_i)}{T_o(q_i)} \times 100\%, \quad (2)$$

$$\Delta \tilde{T} = \frac{\Delta T}{\text{BDBR}}, \quad (3)$$

TABLE III: Overview of JVET CTC test classes.

Class	Resolution	Frame Rate	Typical Content
A1	3840 × 2160 (UHD)	24/30 fps	High-motion, 4K
A2	3840 × 2160 (UHD)	60 fps	High-motion, 4K
B	1920 × 1080 (FHD)	24–60 fps	Mixed content
C	832 × 480 (WVGA)	30 fps	Medium motion
D	416 × 240 (WQVGA)	30 fps	Low resolution
E	1280 × 720 (HD)	30 fps	Mostly static scenes

where $T_o(q_i)$ and $T_p(q_i)$ represent the original and reduced VTM encoding times at quantization parameter q_i, and $q_i \in QP = \{q_1, q_2, \cdots, q_n\}$ is the set of QP values used in the experiments.

To flexibly balance speedup and coding efficiency, we define four configurations (C1–C4) with different confidence thresholds for Type 1 and Type 2 NS predictions (Table IV). Correct NS predictions greatly reduce encoding time, while false alarms increase BDBR.

TABLE IV: Threshold settings for Type 1 and Type 2 NS predictions.

Configuration	Threshold Setting			
	C1	C2	C3	C4
Type 1 NS	0	0.90	0.90	0.95
Type 2 NS	0	0	0.60	0.80

A. Encoding Speedup Evaluation on VTM-22.0

The coding acceleration performance under four configurations, C_1 through C_4, is summarized in this section. Under C_1, the proposed method achieves a 49.20% time-saving ratio (ΔT) with a 3.48% BDBR increase and a normalized gain ($\Delta \tilde{T}$) of 15.87, showing significant encoding time reduction with acceptable quality trade-offs. Configurations C_2 to C_4 show decreasing speed gains (44.93% to 36.35%) and BDBR (2.23% to 1.36%). Notably, C_4 delivers the highest $\Delta \tilde{T} = 28.67$, indicating the best trade-off between complexity and coding performance.

As summarized in Table III, higher-resolution classes (A1/A2, B) contain larger CUs, yielding more pruning opportunities and better acceleration. Class E, despite its static content, outperforms Class C due to its higher resolution and lower motion complexity... While VTM includes early termination for CUs $\leq 64\times64$ [14], limiting Type 2/3 effectiveness, our Type 1 scheme addresses this by targeting larger CUs ($\{128\times128, 128\times64, 64\times128\}$), thereby improving efficiency overall.

B. Comparisons with SOTA Methods

The acceleration performance of our method compared with state-of-the-art (SOTA) algorithms [1, 10, 12] is shown in Table V. We evaluated using C2 and C4 threshold settings. Boldface numbers highlight the best performance in each row (highest speedup, lowest BDBR, and highest $\Delta \tilde{T}$). Our C2 setting achieves the highest speedup for most classes (except Class E), with 2.23% BDBR. The C4 setting yields the highest average $\Delta \tilde{T}$, offering the best trade-off between speed and quality. Shang [1] proposed a fast inter-coding method that reduces complexity by 40.08% with 1.56% BDBR, using neighboring CU information to prune splits and refine modes. In contrast, our threshold-based approach enables flexible speed–quality trade-offs and outperforms these benchmarks on average.

C. Comparisons of Deep Learning Models

Model configuration is a crucial factor in this study. Table VI summarizes the sizes and parameter counts of the models used for comparison. Our three proposed models collectively occupy only 125.4 KB and contain merely 0.015 million trainable parameters—approximately 100 to 200 times fewer than the MobileNetV2 model employed by Tissier et al. [15]. This remarkable compactness makes our approach the smallest deep-learning-based solution for accelerating VVC coding in recent years. However, despite this slight increase in computation, higher resolutions yield higher speedup performances.

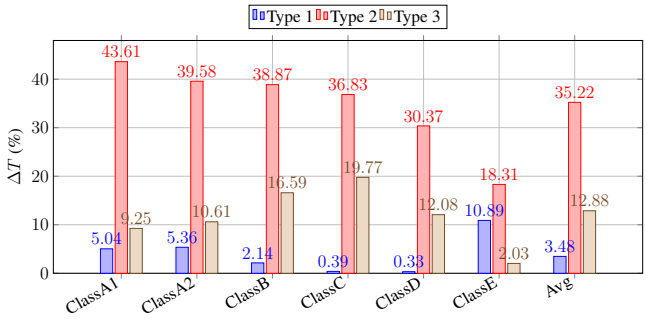


Fig. 4: Ablation study for encoding performance under configuration C2, comparing ΔT across Type 1, Type 2, and Type 3 predictors.

D. Ablation Study

Fig. 4 shows the ablation study of our three-type CU split prediction under C2. Type 1 yields only small overall gains but is more effective in static content (e.g., Class E). Type 2 is the main contributor, achieving 30–45% speedup across classes due to its focus on MTT-based partitions. Type 3 provides moderate benefits by pruning unlikely QT or directional splits, and for Class D it even improves ΔT by helping VTM escape local minima. Overall, Type 2 dominates the speedup, while Types 1 and 3 offer complementary advantages for specific content.

V. Conclusion

This paper presents a lightweight acceleration framework for VVC inter-frame coding, utilizing a novel three-stage CNN-based prediction model (LAI-CNN) specifically tailored to the QTMT partitioning process. By categorizing CU splitting into three types based on VVC constraints, the proposed method improves prediction accuracy while reducing model complexity. LAI-CNN achieves a 36.35% –49.20% encoding

TABLE V: Performance comparison of ΔT, BDBR and $\Delta\tilde{T}$ with SOTA methods.

Test Sequence	Proposed C2(VTM-22.0) Our			Proposed C4(VTM-22.0) Our			Tissier(VTM-10.2)[12] 2022			Shang(VTM-11.0) [1] 2023			Pan(VTM-6.0) [10] 2021		
	ΔT(%)	BDBR(%)	$\Delta\tilde{T}$	ΔT(%)	BDBR(%)	$\Delta\tilde{T}$	ΔT(%)	BDBR(%)	$\Delta\tilde{T}$	ΔT(%)	BDBR(%)	$\Delta\tilde{T}$	ΔT(%)	BDBR(%)	$\Delta\tilde{T}$
ClassA1	**52.14**	2.42	24.85	39.86	**1.51**	30.25	51.1	1.81	28.23	45.96	1.63	**30.86**	40.97	2.98	16.04
ClassA2	**50.66**	2.57	22.68	40.29	1.57	28.95	44.6	1.86	23.98	42.14	**1.40**	**32.00**	32.92	4.72	8.20
ClassB	**48.32**	2.40	20.19	39.03	**1.50**	26.37	46.5	2.21	21.04	41.69	1.68	**26.62**	30.93	3.76	8.89
ClassC	**45.36**	2.17	22.78	39.44	**1.31**	**33.49**	43.1	3.20	13.47	38.15	1.57	24.85	25.975	2.56	10.27
ClassD	**36.96**	1.75	21.49	31.17	**1.04**	**31.87**	36.8	3.02	12.19	36.25	1.41	26.18	22.05	2.35	9.85
ClassE	34.14	1.99	17.14	25.42	**1.19**	21.46	**38.7**	1.45	26.69	36.79	1.28	**29.02**	35.16	2.81	12.91
Average	**44.93**	2.23	21.35	36.35	**1.36**	**28.67**	43.4	2.33	18.63	40.03	1.51	27.86	30.63	3.18	10.75

TABLE VI: Comparisons of size and parameters with SOTA.

Model	Proposed	Tissier[12]	Shang[1]	Pan[10]
Size	125.4 KB	14 MB	-	1.05 MB
Parameter	0.015 M	3.4 M	-	0.245 M

speedup with only 1.36%–3.48% BDBR increase, demonstrating strong efficiency-quality trade-offs. The compact model (125.4 KB, 0.015M parameters) significantly outperforms larger deep models regarding computational overhead (only 2.6% of encoding time). Extensive experiments and ablation studies validate the effectiveness and stability of our method, highlighting its potential for practical deployment in real-time or resource-constrained encoding scenarios. Overall, our research provides an efficient and lightweight solution for QTMT splitting in VVC encoding, offering significant computational savings while maintaining competitive performance. ...While six side parameters (QP, Height, Width, prediction mode, QT depth, MTT depth) were included to enhance split-type discrimination, a detailed ablation of their individual impact is left as future work, since such experiments require extensive retraining beyond the scope of this paper. The proposed approach lays a foundation for further research on real-time video encoding and optimization.

REFERENCES

[1] Xiwu Shang et al. "Low complexity inter coding scheme for Versatile Video Coding (VVC)". In: *Journal of Visual Communication and Image Representation* 90 (2023), p. 103683.

[2] Benjamin Bross et al. "Overview of the Versatile Video Coding (VVC) Standard and its Applications". In: *IEEE Transactions on Circuits and Systems for Video Technology* 31.10 (2021), pp. 3736–3764. DOI: 10.1109/TCSVT.2021.3101953.

[3] Jonathan Pfaff et al. "Intra Prediction and Mode Coding in VVC". In: *IEEE Trans. CSVT.* 31.10 (2021), pp. 3834–3847.

[4] Yiqun Liu et al. "Statistical Analysis of Inter Coding in VVC Test Model (VTM)". In: *IEEE International Conference on Image Processing (ICIP)*. 2022, pp. 3456–3459.

[5] Jingyuan Tang and Songlin Sun. "Optimization of CU Partition Based on Texture Degree in H.266/VVC". In: *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. 2022, pp. 402–408.

[6] Guoqing Wu et al. "SVM Based Fast CU Partitioning Algorithm for VVC Intra Coding". In: *IEEE International Symposium on Circuits and Systems (ISCAS)*. 2021, pp. 1–5.

[7] Tianyi Li et al. "DeepQTMT: A Deep Learning Approach for Fast QTMT-Based CU Partition of Intra-Mode VVC". In: *IEEE Trans. Image Processing* 30 (2021), pp. 5377–5390.

[8] Genwei Tang et al. "Adaptive CU Split Decision with Pooling-variable CNN for VVC Intra Encoding". In: *IEEE Visual Communications and Image Processing (VCIP)*. 2019, pp. 1–4.

[9] Jiann-Jone Chen, Yeh-Guan Chou, and Chi-Shiun Jiang. "Speed Up VVC Intra-Coding by Learned Models and Feature Statistics". In: *IEEE Access* 11 (2023), pp. 124609–124623.

[10] Zhaoqing Pan et al. "A CNN-Based Fast Inter Coding Method for VVC". In: *IEEE Signal Processing Letters* 28 (2021), pp. 1260–1264.

[11] Yiqun Liu et al. "Light-Weight CNN-Based VVC Inter Partitioning Acceleration". In: *IEEE Image, Video, and Multidimensional Signal Processing Workshop (IVMSP)*. 2022, pp. 1–5.

[12] A. Tissier et al. "Machine Learning Based Efficient Qt-Mtt Partitioning for VVC Inter Coding". In: *IEEE International Conference on Image Processing (ICIP)*. 2022, pp. 1401–1405.

[13] Karsten Suehring Jill Boyce and Xiang Li. *VTM common test conditions and software reference configurations for SDR video*. 2020.

[14] K. Andersson W. Ahmad P. Wennersten. *AHG12: MTT split modes early termination*. 2023.

[15] Mark Sandler et al. "MobileNetV2: Inverted Residuals and Linear Bottlenecks". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4510–4520.