# CNN-based Prediction of Partition Path for VVC Fast Inter Partitioning Using Motion Fields

Yiqun Liu, Marc Riviere, Thomas Guionnet, Aline Roumy, Christine Guillemot

# CNN-based Prediction of Partition Path for VVC Fast Inter Partitioning Using Motion Fields

Yiqun Liu, *Student Member, IEEE,* Marc Riviere, *Fellow, IEEE,* Thomas Guionnet, *Fellow, IEEE,* Aline Roumy, *Fellow, IEEE,* and Christine Guillemot, *Fellow, IEEE*

***Abstract*—The Versatile Video Coding (VVC) standard has been recently finalized by the Joint Video Exploration Team (JVET). Compared to the High Efficiency Video Coding (HEVC) standard, VVC offers about 50% compression efficiency gain, in terms of Bjontegaard Delta-Rate (BD-rate), at the cost of a 10-fold increase in encoding complexity. In this paper, we propose a method based on Convolutional Neural Network (CNN) to speed up the inter partitioning process in VVC. Firstly, a novel representation for the quadtree with nested multi-type tree (QTMT) partition is introduced, derived from the partition path. Secondly, we develop a U-Net-based CNN taking a multi-scale motion vector field as input at the Coding Tree Unit (CTU) level. The purpose of CNN inference is to predict the optimal partition path during the Rate-Distortion Optimization (RDO) process. To achieve this, we divide CTU into grids and predict the Quaternary Tree (QT) depth and Multi-type Tree (MT) split decisions for each cell of the grid. Thirdly, an efficient partition pruning algorithm is introduced to employ the CNN predictions at each partitioning level to skip RDO evaluations of unnecessary partition paths. Finally, an adaptive threshold selection scheme is designed, making the trade-off between complexity and efficiency scalable. Experiments show that the proposed method can achieve acceleration ranging from 16.5% to 60.2% under the RandomAccess Group Of Picture 32 (RAGOP32) configuration with a reasonable efficiency drop ranging from 0.44% to 4.59% in terms of BD-rate, which surpasses other state-of-the-art solutions. Additionally, our method stands out as one of the lightest approaches in the field, which ensures its applicability to other encoders.**

***Index Terms*—VVC, multi-scale motion vector field, VTM, QTMT, inter partitioning acceleration, U-Net, multi-branch CNN, multi-class classification.**

## I. Introduction

ACCORDING to [1], global internet traffic has increased substantially, primarily due to the growing video usage, which now accounts for 65% of internet traffic. In addition, the rapid development of Ultra-High Definition (UHD) and Virtual Reality (VR) makes it critical to design more efficient video compression codecs. For this purpose, the latest video coding standard VVC has been finalized in 2020. In comparison to its predecessor, HEVC, its efficiency of inter coding is boosted by about 50% in terms of BD-rate at the cost of 10 times higher complexity [2]. The substantial complexity of VVC impedes its direct implementation in real-time applications such as TV broadcasting. Apart from multiple newly added inter coding tools [3]–[5], a novel partition structure introduced in VVC, called QTMT [6], is the main contributor to this complexity surge. In particular, it has been observed in [7] that the VVC Test Model (VTM) encoder, which is an implementation of the VVC codec, dedicates 97% of its encoding time to searching for the optimal partition. Consequently, fast partitioning methods emerge as the most promising approaches to speed up the whole VVC encoding process.

Manuscript received October 10, 2023

Yiqun Liu, Aline Roumy and Christine Guillemot are with the INRIA (Institut National de Recherche en Informatique et en Automatique) Rennes Bretagne Atlantique, Rennes, France (e-mail: yiqun.liu@irisa.fr; aline.roumy@inria.fr; christine.Guillemot@inria.fr).

Yiqun Liu, Thomas Guionnet and Marc Riviere are with the ATEME company Rennes Bretagne Atlantique, Rennes, France (e-mail: y.liu@ateme.com; m.riviere@ateme.com; t.guionnet@ateme.com).

### A. Partitioning Acceleration for VVC

#### 1) Fast Intra Partitioning Methods

Numerous works achieve an important acceleration of intra-frame partitioning in the All-Intra (AI) encoding configuration. These approaches fall primarily into two categories: heuristic-based methods and machine learning-based methods.

Some heuristic-based methods are built upon pixel-wise statistics, such as gradients [8]–[10] and variances [8, 10]. To simplify the partitioning process, other heuristic methods reuse some data generated during the encoding process, such as Rate-Distortion cost (RD-cost) of Coding Unit (CU) encoding [11], coding tool decisions [12], best split type, and the intra mode of sub-CUs [13].

Machine learning-based methods utilize CNN or Decision Tree (DT) models to expedite intra partitioning. In [14]–[16], a CNN model is trained to predict the split boundaries inside CTU partitions. In [17], Feng *et al.* propose a fast partitioning method by predicting a QT depth map, multiple MT depth maps, and multiple MT direction maps with CNN. Regarding the DT-based approach, various Light Gradient Boosting Machine (LGBM) classifiers are separately trained for different CU sizes to predict the possible splits, as demonstrated in [18].

#### 2) Fast Inter Partitioning Methods

Fewer contributions of fast inter partitioning methods have been proposed for VVC. Due to the fact that the inter coding consists of predicting pixels of current frame depending on previously encoded reference frames, encoding errors resulting from the use of fast coding methods are propagated and accumulated between frames. Therefore, the acceleration of inter partitioning is a more challenging task compared to that of intra partitioning. Nevertheless, the acceleration of inter-frame coding is key to speeding up the overall encoding process, especially in RandomAccess (RA) and Low-Delay

(LD) configurations. These configurations are employed more widely than the AI configuration in scenarios such as broadcasting and streaming.

Generally, fast partitioning approaches aim to reduce the search space of potential partitions. Therefore, accurately predicting the subset of partitions is of crucial importance. Heuristic methods proposed for fast intra partitioning of VVC [8, 13] heavily depend on handcrafted features to determine whether to check a partition. These methods are fast and simple to implement but lack accuracy for two reasons. Firstly, the features are computed locally on the CU and/or sub-CUs, which fails to provide a synthesized view of the entire CTU. Secondly, these features, including variances, gradients, and coding information, are low dimensional and do not adequately capture the complexity of CTU.

One approach to improve the accuracy of partition prediction involves increasing the dimension of the extracted features. This is the case with the methods based on Random Forest (RF) [19] or DT [20], which use over 20 features from a given CU and its sub-CUs. As a result, decisions made by these methods remain confined to the local context of CU, without considering the entirety of CTU. Rather than relying on local information, a more effective selection of subsets of partitions should be based on global features computed on the entire CTU. This can be accomplished through the utilization of CNN-based methods.

Several approaches [21]–[23] use CNN to partially accelerate the partition search process. For instance, in [21], Pan *et al.* propose a multi-branch CNN to perform a binary classification of the "Partition" or "Non-partition" at the CU level. In [22], the split type at the CTU level is predicted, whereas the partitions of its sub-CUs are not determined. Liu *et al.* in [23] employ a CNN to estimate an 8x8 grid map of QT depth, which is used to discard a portion of the MT splits. These methods cover only a part of the partition search space, while the partition search is conducted exhaustively on the remainder. These methods could be referred as partial partitioning acceleration methods by CNN.

A complete partitioning acceleration of inter coding by CNN is proposed in [15]. A vector containing probabilities of the existence of split boundaries in the partition is predicted similarly to [24]. This method is fast in the sense that a single vector is computed for each CTU. Nevertheless, it is observed in [24], that the predictions are more accurate at higher levels of the partitioning tree. Hence, they propose improving the decisions by adding 16 trained DTs to process the CNN output, introducing additional complexity to the method.

### *B. Proposed Method*

In the MT partitioning, both binary and ternary (with sub-CUs of two different sizes) splits are available. Consequently, CUs at a specific depth in the tree do not correspond to the same size and shape, introducing dependence between the MT splits along the partition path. This dependence partly explains the decrease in partition prediction accuracy as the depth of the partitioning tree increases, as observed in recent studies [15, 24] presented in the previous section.

More precisely, since the size and shape of a CU depend not only on its depth in the tree but also on consecutive MT splits, depth alone is insufficient for defining a partition. Therefore, we propose making decisions on MT partitioning in a hierarchical manner, considering their dependence on the partition path. We also introduce a one-shot approach for QT partitioning which precedes MT partitioning, since there is a one-to-one correspondence between the QT depth and the CU size at that particular depth.

Hence, our overall proposition involves predicting the partition path, which includes a one-shot prediction for the QT partitioning, followed by a hierarchical prediction for the MT partitioning. Additionally, to further improve the accuracy of partition prediction, we suggest basing the partition decision not only on pixel values and residual values but also on motion vector fields, as these fields exhibit a strong correlation with partitioning [19].

Our two main contributions are as follows:

- We propose a novel partition-path-based representation of the QTMT partition at the CTU level as a map of QT depth plus three maps of MT split well adapted to the sophisticated partitioning scheme in VVC.
- We design a U-Net-based CNN model taking multi-scale fields of motion vectors as input to effectively predict QT depth map as well as split decisions at different MT levels.

We also have other contributions such as:

- We build MVF-Inter[1], a large scale dataset for inter QTMT partition of VVC, which could facilitate the research in this field.
- We propose a fine tuned loss function for this complex multi-branch multi-class classification problem.
- We develop a fast partition scheme effectively exploiting the prediction of a CNN model in a way that the most possible splits are determined at each partition level.
- We design a specific threshold-based selection approach to match with the partition scheme, which allows us to realize a large range of trade-offs between complexity and compression efficiency.

The remainder of this paper is organized as follows. In Section II, we provide an overview of QTMT partitioning scheme in VVC, including the concept of the partition path. The motivation and detailed description for our proposed representation of the QTMT partition are presented in Section III. In Section IV, the structure of the proposed CNN model is illustrated. We give a detailed description of the partitioning acceleration algorithm in Section V. The loss function of CNN and the dataset generation process are described in Section VI. In section VII, the evaluation of the prediction accuracy of

[1]Our dataset MVF-Inter is available at https://1drv.ms/f/s!Aoi4nbmFu71Hgx9FJphdskXfgIVo?e=fXrs0o

our CNN model is carried out. Furthermore, we compare our result with the state-of-the-art of RF-based methods and CNN-based methods, respectively. The complexity analysis of our method is also included in this section. Finally Section VIII concludes the paper. Our source code is available at https://github.com/Simon123123/MSMVF_VVC_TIP2023.git.

## II. Overview of Partitioning Structure in VVC

### A. QTMT Partitioning

The partitioning of VVC is done in a top-to-bottom manner. Starting from CTU, the encoder applies possible split types recursively on CU at each level of the partitioning tree, in order to find the partition which best exploits spatial and temporal redundancy. The QTMT partitioning scheme in VVC consists in splitting the CU using either a QT split or a MT split. For the MT split, four split types are introduced: Horizontal Binary Tree (HBT) split, Vertical Binary Tree (VBT) split, Horizontal Ternary Tree (HTT) split and Vertical Ternary Tree (VTT) as illustrated in Figure 1.

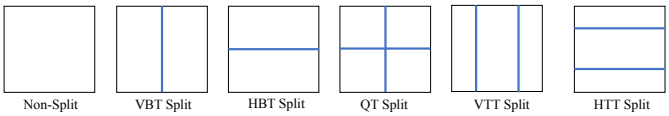



Fig. 1: Different split types in VVC.

Based on the QTMT partitioning scheme, dimensions of the final encoded CU could range from 128×128 to 4×4 [6] including squared and rectangular CUs of 32 different sizes in the RA configuration with the CTU size of 128x128. Figure 2, QTMT can achieve fine partitioning adapted to the local frame texture.

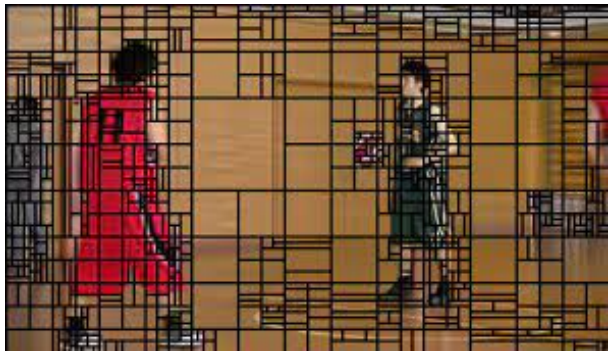


Fig. 2: Example of a partitioned frame in VVC. [8]

A worth-noting characteristic of QTMT partitioning is that the available split types depend on CU sizes. Figure 3 presents the number of available split types including No Split (NS) per CU size for luma samples. The number of split options varies from 1 to 6 for different sizes of CU, making the partition acceleration at the CU level more complicated. Except for the CU size restrictions on available split types, VVC implements various shortcuts [25], including speed-up rules based on content gradients and QT search restrictions estimated on neighboring CUs, etc.

The partitioning is executed at CTU level with dual tree allowing separated partitioning tree for luma and chroma. Our algorithm only accelerates the luma partitioning search using luma samples, and the resulting luma partitioning tree is then applied to the chroma components.

<table>
<tr><td></td><td colspan="7">Width</td></tr>
<tr><td rowspan="8">Height</td><td></td><td>128</td><td>64</td><td>32</td><td>16</td><td>8</td><td>4</td></tr>
<tr><td>128</td><td>4</td><td colspan="3">3</td><td colspan="2">-</td></tr>
<tr><td>64</td><td rowspan="3">3</td><td>6</td><td colspan="2">5</td><td rowspan="3">4</td><td rowspan="3">3</td></tr>
<tr><td>32</td><td rowspan="2">5</td><td>6</td><td>5</td></tr>
<tr><td>16</td><td>5</td><td>6</td></tr>
<tr><td>8</td><td rowspan="2">-</td><td colspan="3">4</td><td>3</td><td>2</td></tr>
<tr><td>4</td><td colspan="3">3</td><td>2</td><td>1</td></tr>
</table>

Fig. 3: Number of split types per CU in VVC for Luma.

### B. Partition Path

In this work, we introduce the concept of partition path to depict the partition of a CU. The partition path refers to the sequence of splits applied to obtain a CU during the partitioning. In the RDO process of partitioning, numerous partition paths included in the partition search space are checked and the optimal one leading to the final partition is selected. Figure 4 illustrates a simplified tree representation showing all possible partition paths checked for a CTU. The red arrows indicate the selected partition path with the lowest RD-cost [26], while the blue arrows represent other paths that have been tested by RDO, but were not selected.

Specifically, within the QTMT partition, it is important to note that QT splits are prohibited for the child nodes of a MT split. Consequently, the search for optimal partition path in VVC can be conceptualized as a sequential two-step decision-making process, comprising a sequence of QT splits followed by a series of MT splits.

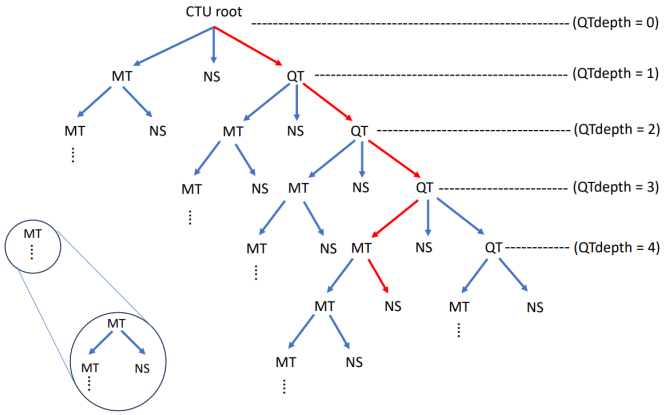



Fig. 4: Example of partition paths.

## III. Novel Representation of QTMT Partition

Based on the QTMT partitioning structure and the partition path of VVC presented in the previous section, we introduce

in this section a novel representation of QTMT partition by partition path. In III-A, we explain the motivation for this new representation. The partition path representation is illustrated in III-B.

### A. Motivation

Previous partition representations at CTU level have typically used binary vectors to depict split boundaries. In [14, 15, 24], the authors intend to predict the split boundary of each 4x4 sub-block in CTU. Lately, Wu *et al.* improve this representation in [16] by proposing hierarchical boundaries. This adaptation is designed to better align with the QTMT partition pattern. In this work, binary labels for split boundaries of varying lengths are predicted. Collectively, these methods provide a geometric representation of the partition.

The limitations of the geometric representation mainly lie in two aspects. Firstly, it is an implicit representation of the partitioning process, requiring conversions from boundary vectors to split decisions. In the case of [14], conversions are carried out by computing the average probability at the location of the specific split. [15] and [24] convert boundary vectors to split decisions by DT models separately trained for different CU sizes. Secondly, different partition paths could be deduced from a particular partition presented in a geometric way. For example, as demonstrated in Figure 5, the partition defined by the split boundaries can lead to three distinct partition paths. These partition paths correspond to different coding performances and are individually tested in the RDO process. This multiplicity of partition paths of the geometric representation limits the acceleration potential of the method.

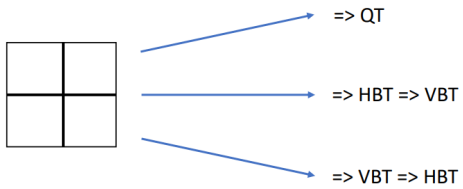



Fig. 5: Possible partition paths for a final partition given by split boundaries

To address the above limitations, we introduced a novel representation based on the partition path. Our representation comprises the QT depth map and the MT split maps. Firstly, the split decisions at each depth can be directly deduced from either the QT depth map or the split map. This eliminates the need for decision trees, reducing method overhead, and simplifying implementation. Secondly, it corresponds to a unique partition path, maximizing the potential for complexity reduction.

### B. QT Depth Map and MT Split Maps

Considering that the maximum number of QT splits and MT splits is typically set to 4 and 3 in VTM, any partition can be effectively described by a QT depth map (*i.e.* QTdepthMap) along with three MT split maps (*i.e.* MTsplitMap) in sequence. Each element within QTdepthMap and MTsplitMap corresponds to an 8x8 and 4x4 area, which aligns with the dimensions of the smallest sub-CUs for the QT split and the MT split in VTM.

A detailed example of our partition representation is shown in Figure 6. To keep it simple and without loss of generality, we represent this example for a CTU size of 64×64. In this figure, (a) shows an instance of QTMT partition with its corresponding tree representation shown in (b). (c)-(f) illustrate the QTdepthMap and MTsplitMaps generated from this partition. Given that the CTU size in this example is 64x64, the sizes of QTdepthMap and MTsplitMap are 8x8 and 16x16, respectively. The QTdepthMap in (c) consists of QT depth values ranging from 0 to 4, while each element in MTsplitMap in (d)-(f) represents the split decision among five options: NS, HBT, VBT, HTT and VTT. This representation depicts a distinct partition path for every CU within the partition. To provide an example, consider the CU highlighted in the green circle in Figure 6. Its partition path can be expressed as three QT splits (QT depth 3), followed by a HBT split and two NS decisions.

## IV. CNN-based Prediction of Partition Path

Predicting the optimal partition is equivalent to predicting the optimal partition path. In VTM, the size of CTU is set to 128x128 by default, consequently yielding QTdepthMap and MTsplitMap dimensions of 16x16 and 32x32, respectively. The representation of partition path can be predicted by a multi-branch CNN, where one branch infers the QTdepthMap of regression values with dimension 16x16x1, while the other three branches produce the MTsplitMap. Each element of MTsplitMap is classified into one of five classes, corresponding to five split types, resulting in three MT outputs with dimensions of 32x32x5. We have handled the classification of MT splits as an image segmentation problem based on 4x4 sub-blocks. Accordingly, we adopted the classical U-Net structure [27] to design our CNN model to address this segmentation-like task.

In this section, the U-Net structure is briefly introduced. Then we present the structure of the proposed CNN in Section IV-B. Afterwards, we list its input features and explain the reasons for choosing them in Section IV-C.

### A. U-Net

The U-Net structure is derived from Fully Convolutional Network (FCN) [28]. It consists of an encoder part which is composed of a sequence of convolutional layers plus maxpooling layers. Then this part is followed by a decoder part in which the maxpooling layers are replaced by upsampling layers. In addition, skip connections concatenate feature maps from the encoder and decoder with the same dimension. The U-Net and its variations have been widely applied to image segmentation tasks.

### B. MS-MVF-CNN Structure

The CNN structure proposed in this paper, named Multi-Scale Motion Vector Field CNN (MS-MVF-CNN), is depicted in Figure 7. The proposed CNN has 7 inputs and 4 outputs.

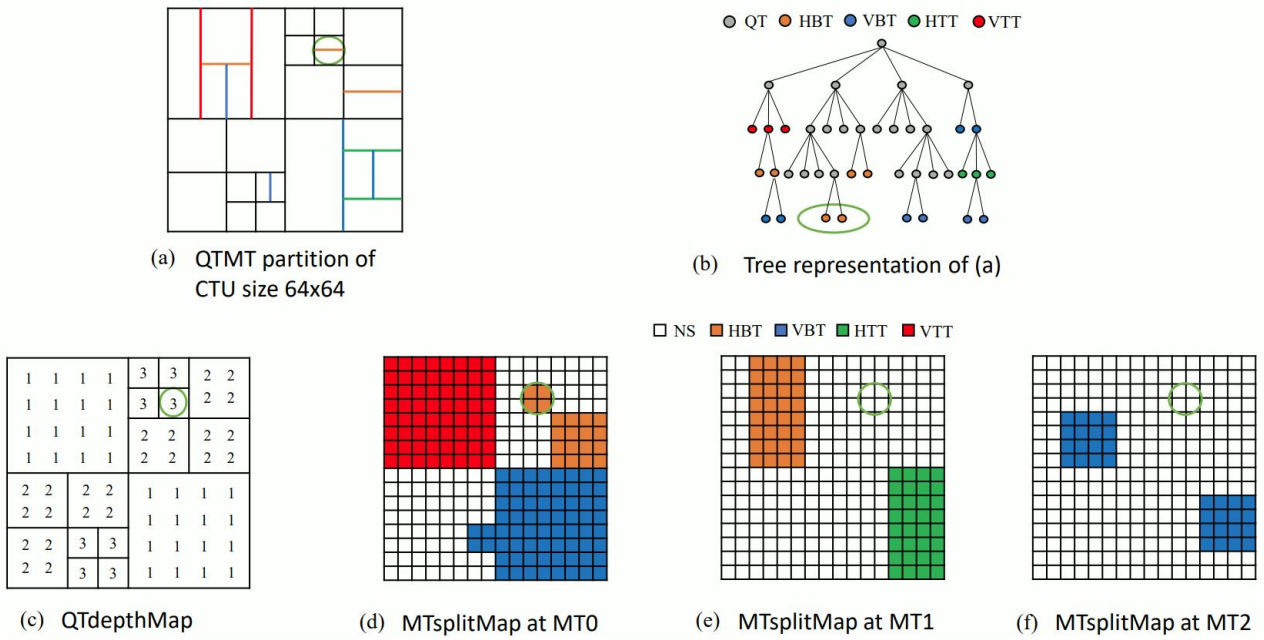



Fig. 6: Example of QTMT partition, tree representation, QTdepthMap and MTsplitMaps of CTU size of 64x64

After two convolutional layers with stride, the tensor of Input 1 is downsampled to dimension 32x32x8 and then concatenated with the Input 2. The merged input is then fed to the U-Net feature extractor demonstrated in Figure 8. Regarding the design of this module, we are referring to the classical structure of U-Net depicted in [27]. Specifically, we concatenate the upsampled feature map in the decoder part of U-Net, the feature map copied from the encoder part with the motion vector field of the same scale. At the decoding part, the feature map is gradually expanded and merged with normalized motion field of 2x2x6, 4x4x6, 8x8x6, 16x16x6 and 32x32x6. As a result, the U-Net feature extractor outputs a feature map of dimension 32x32x8, combining pixels features with motion estimation features.

Since the split at each level depends on previous splits, we employ a hierarchical multi-branch prediction mechanism. QTdepthMap is predicted after shrinking the features extracted from U-Net by four convolutional layers. For MT branches, we designed the MT branch module presented in Figure 8. Two inputs of this module are the extracted features of U-Net and outputs from previous partition levels. We utilize the asymmetric kernel structure to process the extracted features. This structure is originally proposed by [29] in HEVC to pay attention to near-horizontal and near-vertical textures for predicting split decision of intra coding by CNN. We adopt this structure to exploit the horizontal and vertical information contained in Multi-Scale Motion Vector Field (MS-MVF). The MT branch module contains branches of kernel size MxN, LxL, and NxM. The values of (M, N, L) are set as (5, 7, 9) for branch MT0, (3, 5, 7) for branch MT1, and (1, 3, 3) for branch MT2. On deeper MT levels, splits are made on smaller CUs. Thus, smaller kernel sizes are applied to extract finer features. After the asymmetric kernels, the feature map is then concatenated with outputs from previous levels. In the end, the merged feature maps are given to two residual blocks [30] before yielding classification results of MT branches. No activation is applied to the fully connected output layer of the QT depth branch. The output layer of the MT branch is with softmax activation.

### C. Input Features

This network structure takes three different types of input. The involved inputs are presented below:

#### 1) Original and Residual CTU

In Figure 7, Input 1, with dimensions of 128x128x2, is created by merging the original CTU with the residual CTU. The original luma pixels carry the texture details of the CTU, while the residual CTU is generated through motion compensation of the original CTU based on the nearest frame.

Several studies [21, 22, 31] have adopted a method in which both the original CTU values and the residual of CTU are fed to a CNN. Combining the original and residual values as input allows CNN to assess the similarity between current CTU and reference CTU. This combined input offers features that reflect the temporal correlation between frames which is a crucial factor in inter partition prediction.

#### 2) QP and Temporal ID

The Input 2, as illustrated in Figure 7, has dimensions of 32×32×2, consisting of two separate 32x32 matrices. These matrices are assigned specific values: one holds the Quantization Parameter (QP) value, while the other contains the temporal identifier. This temporal identifier in VVC, similar to its usage in HEVC, signifies a picture's position within a hierarchical temporal prediction structure, controlling temporal scalability [32].

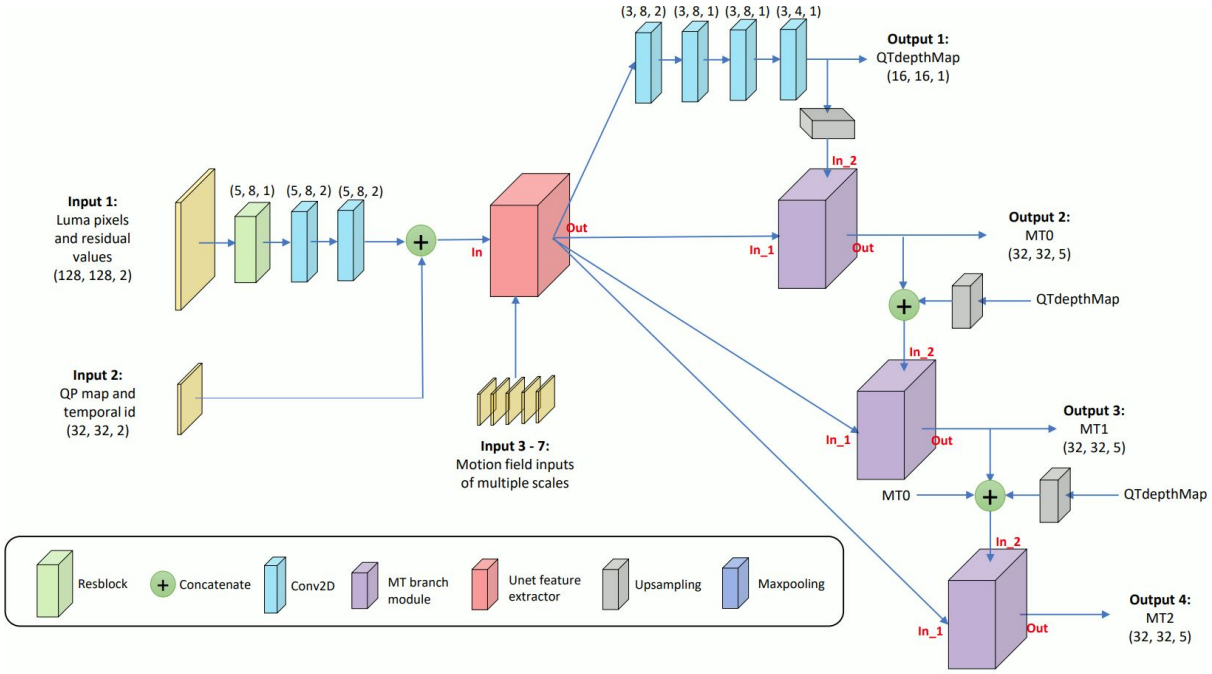



Fig. 7: Multi-Scale Motion Vector Field CNN. The vector above Resblock and Conv2D represents (kernel size, number of filters, stride).

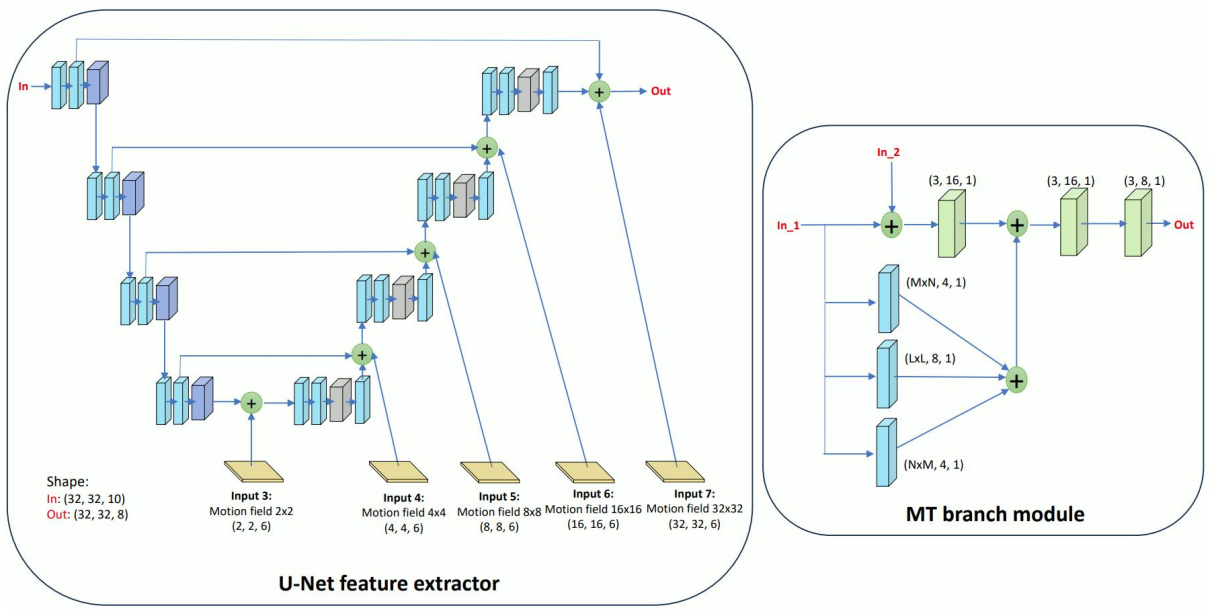



Fig. 8: U-Net feature extractor and MT branch module

We specifically utilize the QP value and temporal identifier as input features since inter partitioning depends on them. In essence, a higher temporal layer identifier or a lower QP value tends to result in finer partitions, as outlined in [22]. Instead of developing separate models for each parameter instance, our approach focuses on training a model with adaptability to varying values of QP and temporal identifier.

### *3) Multi-Scale Motion Vector Field*

In this paper, we have introduced a CNN model based on a novel input feature called MS-MVF. Our MS-MVF at five scales is presented as Input 3-7 in Figure 8. To compute MS-MVF, we divide the 128x128 CTU into multiple scale sub-blocks ranging from 4x4 pixels to 64x64 pixels, and perform motion estimation on these sub-blocks. Each motion vector of sub-block comprises a vertical and horizontal motion value, along with the associated Sum of Absolute Differences (SAD) cost value as the third element. By concatenating elements pointing to reference frame of L0 with those of L1, each sub-block corresponds to 6 elements in the motion vector field. For example, the motion vector field input for 8x8-pixel scale has dimensions of 16x16x6.

A significant challenge in inter partition prediction is the

large motion search space, which spans up to 6 regions of 384x384 pixels across different reference frames in the RAGOP32 configuration. State-of-the-art methods typically employ motion fields or pixels from reference frames as input features for machine learning models. Notably, in [19] and [21], a crucial feature used is the motion field, which comprises motion vectors calculated for each 4x4 sub-block referring to the nearest frame. As mentioned in [19], this motion field is strongly correlated with the optimal partition. In a different approach, Tissier *et al.* in [15] opt to utilize two reference CTUs in the nearest frames.

The choice of using MS-MVF as the CNN input, instead of motion fields and reference pixels, is based on the following reasons. First, the MS-MVF contains crucial motion information for the current CTU, which is essential for both inter prediction and inter partitioning. This information can be interpreted more effectively by the CNN model compared to using reference pixels as CNN input. Second, the multi-scale nature of MS-MVF aligns well with the multi-level structure of U-Net and can leverage this structure effectively. Essentially, MS-MVF represents motion features at various resolutions, allowing for the combination with features extracted from CTU pixels at the same resolution scale.

To demonstrate the effectiveness of our MS-MVF input, we conducted an experiment involving the training of two CNN models. The only distinction between these models is their input: the first model, PIX-CNN, takes the pixels of two reference CTUs as input, while the second model, MVF-CNN, utilizes our proposed MS-MVF as input. Both models share the same architecture as in Figure 7. The training dataset comprises 250k samples randomly selected from the RAGOP32 encoding of 200 sequences with a resolution of 540p from [33]. Performance evaluations in Figure 9 are based on Class C sequences of Common Test Condition (CTC). The results consistently show that MVF-CNN outperforms PIX-CNN at all four data points, which justifies the advantages of using the MS-MVF input over pixel input.

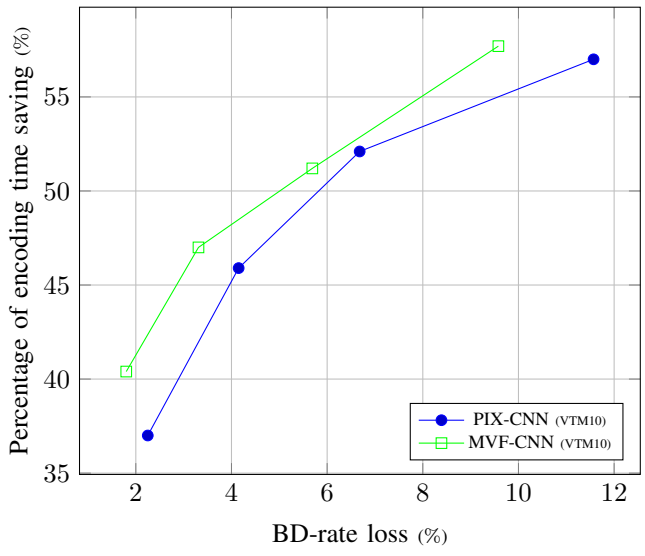



Fig. 9: Comparison of performances between PIX-CNN and MVF-CNN.

Based on our evaluation conducted on the first 64 frames of all CTC sequences using the RAGOP32 configuration, the computation of MS-MVF for each CTU consumes, on average, a mere 0.52% of the encoding time in VTM10. Importantly, the generation of MS-MVF introduces only minimal encoding overhead, making it a task that can be readily preprocessed or parallelized.

## V. Proposed CNN-based Acceleration Method

After the prediction by our trained CNN model, we obtain one QTdepthMap and three MTsplitMaps per CTU. The predicted QTdepthMap is composed of floating-point values. The predicted MTsplitMaps comprise probabilities of five split types for each 4x4 sub-block within the CTU. In this section, we elucidate the post-processing of the CNN prediction, with the aim of achieving a wide range of acceleration-loss trade-off.

**Algorithm 1** MT splits early skipping

**Input:**
QTdepthMap; MTsplitMap; Thm; $\text{QTdepth}_{\text{cur}}$, CU; $\text{Size}_{\text{CU}}$; $\text{Pos}_{\text{CU}}$
**Output:**
SkipMT: Boolean to decide whether to skip MT split types or not.
CandSplit: Candidate list of splits for RDO check

Compute the average $\text{QTdepth}_{\text{pred}}$ based on $\text{Size}_{\text{CU}}$, $\text{Pos}_{\text{CU}}$ and QTdepthMap
**if** round($\text{QTdepth}_{\text{pred}}$) > $\text{QTdepth}_{\text{cur}}$ **and** QT is possible for current CU **then**
    SkipMT = True
    CandSplit = {NS, QT}
**else**
    SkipMT = False
    CandSplit = {NS}
    **for** sp in {BTH, BTV, TTH, TTV} **do**
        Compute average $\text{Proba}_{\text{sp}}$ based on $\text{Size}_{\text{CU}}$, $\text{Pos}_{\text{CU}}$ and MTsplitMap
        **if** $\text{Proba}_{\text{sp}}$ > Thm **then**
            CandSplit append split sp
        **end if**
    **end for**
**end if**

Decision errors at low partitioning depth can result in large loss of BD-rate. Based on the predictions of our CNN, selecting the best single partition path, equivalent to choosing the best split at each MT level, will not be optimal or scalable. Our approach involves generating candidate lists at each level, which means that multiple partition paths are chosen for the RDO test. This approach of creating candidate lists at various levels is designed to achieve satisfying trade-off between acceleration and coding loss while assuring the scalabilty of method.

The acceleration algorithm is precisely described in Algorithm 1 and Figure 10. We introduce two parameters *Thm* and *QTskip* to regulate the acceleration-loss trade-off. Specifically,

*Thm* is the threshold for the split probability. *QTskip* represents whether we should accelerate RDO of QT splits or not. Increasing the *Thm* value and setting *QTskip* to true will lead to greater acceleration at the cost of increased coding loss.

Regarding the algorithm applied at the CU level, Algorithm. 1 is first executed. This algorithm produces two outputs: the *SkipMT* variable and the *CandSplit* list, both of which are subsequently utilized in the flowchart in Figure 10. To start with, the mean $QTdepth_{pred}$ of current CU is calculated based on the corresponding area in Quad Tree depth map (QTdepthMap). If the rounded $QTdepth_{pred}$ is larger than the QT depth of the CU and QT split is feasible, the current CU should be split by QT. Consequently, all MT splits are excluded from the *CandSplit* list and *SkipMT* is set to true. Otherwise, the mean probability of each available split is computed on the corresponding MTsplitMap in a similar way to that of the QTdepth. Then *CandSplit* is filled by splits with $Proba_{sp}$ larger than the threshold *Thm*. In this case, the value assigned to *SkipMT* is False.

In the flowchart of Figure 10, if the *SkipMT* is true after the execution of Algorithm 1, we directly check the *CandSplit*. In this scenario, the encoder conducts RDO of CU and splits CU with QT because *CandSplit* contains only NS and QT. If *SkipMT* is false, then we will verify if NS is the only choice in *CandList*. If this is the case, we will add the MT split with the highest probability to the list. Next, if QT split is not allowed for CU due to CU shape or shortcuts, we go directly to the check of *CandSplit*. If the QT split is feasible, we refer to *QTskip* to determine whether to add QT to the *CandList* or not. Setting the *QTskip* to true signifies that we will always check QT if possible. This is for rectifying the potential error of predicting a $QTdepth_{pred}$ value smaller than the actual ground truth value. However, it comes at the expense of sacrificing some acceleration. Finally, we execute RDO on CU and partition it by split types in the *CandSplit* list. The partition search then repeats for the next CU, and the algorithm described above is applied anew.

Our inter partitioning acceleration method is designed on top of the partitioning algorithm of VTM which performs a nearly exhaustive search on possible partition paths of a CTU, except that it incorporates a handful of handcrafted conditional shortcuts as mentioned in Section II. Therefore, this work can be considered as a CNN-based shortcut to reduce the search space of partition paths.

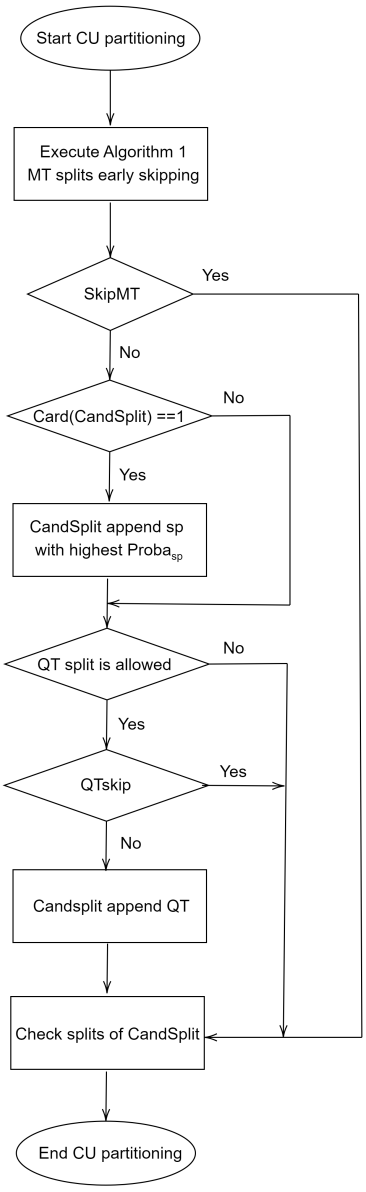



Fig. 10: Flowchart of acceleration algorithm

## VI. Training of MS-MVF-CNN

To effectively train our CNN model, we have designed a hybrid loss function and created a large-scale dataset named MVF-Inter[1]. First, we will explain how this loss function is determined in Section VI-A. Then Section VI-B describes training details and the generation of dataset.

### A. Loss Function

The outputs of MS-MVF-CNN contain one regression output as well as three classification outputs. Therefore, a hybrid loss function is developed in our case. We choose the category cross-entropy for classification loss and mean square error for regression loss as follows:

$$L = a\frac{1}{n_q}\sum_{i=1}^{n_q}(d_i - \hat{d}_i)^2 - (1-a)\left(\sum_{b=1}^{n_b}\sum_{i=1}^{n_m}\sum_{s=1}^{n_s} w_{b,s} y_{b,i,s}\log(\hat{y}_{b,i,s})\right) \quad (1)$$

Here, we have $n_q$ = 256, $n_b$ = 3, $n_m$ = 1024 and $n_s$ = 5, representing the number of elements in QTdepthMap, the number of MT branches, the number of elements in MTsplitMap, and the number of split types, respectively. In this equation, $d_i$ denotes the ground-truth QT depth value, while $\hat{d}_i$ represents the predicted QT depth value. Additionally, $\hat{y}_{b,i,s}$ is used to denote the predicted probability of split type $s$ for the $i$-th element of the MT decision map at the $b$-th MT branch. Similarly, $y_{b,i,s}$ signifies the ground-truth label for the same case. Notably, we introduce a parameter $a$, which falls

within the range [0, 1], in Equation 1 to fine-tune the relative weights of the regression loss and classification loss.

The split types are distributed unbalancedly at different MT depths as illustrated in Figure 11. To counteract this imbalance, we introduce class weights for split type $s$ on MT branch $b$, denoted as $w_{b,s}$. The definition of these weights is as follows:

$$w_{b,s} = \frac{\lambda_s p_{b,s=ns}}{p_{b,s}} \tag{2}$$

where $p_{b,s}$ represents the percentage of split type $s$ within MT branch $b$. For each branch $b$, $\frac{p_{\text{b,s=ns}}}{p_{\text{b,s}}}$ can be interpreted as the inverse percentage of the split type $s$ normalized by the inverse percentage of the NS split. In [6], a series of tests were performed to evaluate the coding gain and increase of complexity associated with the Binary Tree (BT) and Ternary-type Tree (TT) splits individually as demonstrated in Table I.

TABLE I: Settings of split type in VTM9 under RA [6]

| | BT split | TT split | BD-rate | Encoding Time |
|---|---|---|---|---|
| Anchor Setting | X | X | - | - |
| Setting 1 | ✓ | X | -8.26% | 337% |
| Setting 2 | X | ✓ | -10.22% | 732% |

When comparing Setting 1 and Setting 2 to the anchor configuration, it's observed that Setting 1 and Setting 2 exhibit similar BD-rate gains, but the encoding time in Setting 2 is twice that of Setting 1. These tests suggest that BT split performs better in terms of the trade-off between complexity and coding gain compared to TT split. Thus, placing greater importance on the prediction of the BT split can result in a better acceleration-loss trade-off. To achieve this, the ratio between the proportion of NS and proportion of split $s$ is computed for MT branch $b$. The class weight $w_{b,s}$ in Equation 2 is formulated as the product of this ratio and $\lambda_s$ which is another weight added to prioritize the split type $s$.

After fine-tuning the model, we find that the best performance is achieved with a value of 0.8 for $a$ and $\lambda_s$ set to 2 for BT splits and 1 for TT splits and NS.

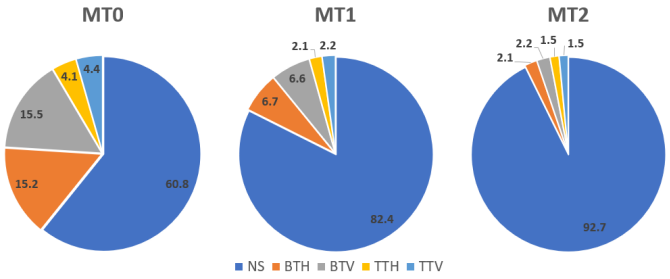



Fig. 11: Distribution of split types for MT0, MT1, MT2

### B. Dataset Generation and Training Details

Constructing a large scale inter partition dataset is more challenging than that of intra partition because the former needs to encode a substantial number of video sequences, while the latter could be done by encoding images. To the best of our knowledge, there exists no prior work focused on developing an inter partition dataset.

Our MVF-Inter[1] dataset involved the encoding of 800 sequences from [33] and an additional 28 sequences of 600 frames in 720p resolution extracted from [34]. Sequences of [33] cover resolutions of 240p, 540p, 1080p, and 4k, with 200 videos of 64 frames for each resolution. We have encoded all these videos with the VTM10 [35] encoder in the RAGOP32 configuration with QP 22, 27, 32, and 37. We randomly selected a total of 820k CTU partition samples, equally distributed per resolution and QP, with 120k samples reserved as a validation set.

Each sample of our dataset contains the following components of each CTU: pixel values, residual values, motion vector fields at five scales, QP value, temporal ID value, QTdepthMap with depths ranging from 0 to 4, and MTsplitMaps for MT0, MT1, and MT2. MTsplitMap labels are encoded as VTT (0), VBT (1), NS (2), HBT (3), and HTT (4).

In terms of training details, we employed the Adam optimizer [36] to train the model. The initial learning rate was set to $1e^{-3}$ and was exponentially decreased by 3% every 5 epochs. The batch size set for training is 400.

## VII. Experimental Results and Analyses

In this section, we present the results of our experiments and provide an in-depth analysis of the results. To begin, in Section VII-A, we assess the precision of the prediction of our CNN model. Subsequently, comparisons with the RF and CNN based approaches are made in Section VII-B. Finally, the complexity analysis of our framework is carried out in Section VII-C.

### A. Prediction Accuracy Evaluation

At the CU level, our algorithm can be broken down into two decisions: the decision of *SkipMT* and the decision of *CandSplit* list. To evaluate the precision of decisions based on our model's output, we have performed the encoding where both the ground truth partitioning and the CNN output were collected. The analysis is done on the first 64 frames of all CTC sequences excluding class D with QP 22, 27, 32, 37. The accuracy of these decisions presented in Table II and Figure 12 are calculated by averaging four QPs and various test sequences.

There is no need to make a *SkipMT* decision on QT depth 4 since the partitioning is forced to proceed to MT splits with the maximum of QT depth reached. The accuracy of *SkipMT* decision is independently measured on QT depth from 0 to 3. If the current CU requires further splitting of QT and *SkipMT* is equal to False, then this decision of *SkipMT* is classified as False Negative (FN). The proportion of True Positives (TP), FN, True Negatives (TN), False Positives (FP) and their corresponding Precision (Prec) and Recall (Rec) are shown in Table II. Precision, recall, and F1 score are calculated as follows:

$$Precision_{\text{QTdepth}} = \frac{TP_{\text{QTdepth}}}{TP_{\text{QTdepth}} + FP_{\text{QTdepth}}} \tag{3}$$

$$Recall_{\text{QTdepth}} = \frac{TP_{\text{QTdepth}}}{TP_{\text{QTdepth}} + FN_{\text{QTdepth}}} \quad (4)$$

$$F1score_{\text{QTdepth}} = 2\frac{Precision_{\text{QTdepth}} Recall_{\text{QTdepth}}}{Precision_{\text{QTdepth}} + Recall_{\text{QTdepth}}} \quad (5)$$

Generally, our model exhibits strong performance at QT depths ranging from 0 to 2, as depicted in Table II. Both precision and F1 score decrease as QT depth increases. At QT depth 3, the precision and F1 score drop to 25% and 40%, respectively, suggesting that the *SkipMT* decision at this level is less reliable. These observations could be explained by two reasons:

First of all, the scale of decision-making diminishes as the QT depth increases. More explicitly, the *SkipMT* decision at QT depth 0 is made at the CTU scale by computing the mean of 256 values from the QTdepthMap. Nevertheless, the decision at QT depth 3 relies only on 4 values from the QTdepthMap within the 16x16 CU. Consequently, decisions at smaller scales are less resilient to incorrectly predicted QTdepthMap values, resulting in lower overall accuracy at higher QT depths.

Secondly, decisions at higher QT depths are noticeably more imbalanced than those at lower QT depths. Positive cases of ground truth at QT depth 3 represent only 0.02% , while the proportion of positive cases is 49.65% at QT depth 0. In conclusion, the model is trained in such a way that it tends to make negative *SkipMT* decision at larger QT depths. This explains the decline in precision as the QT depth increases.

TABLE II: Table of confusion for *SkipMt* (Unit: %)

| | TP | FN | TN | FP | Prec | Rec | F1score |
|---|---|---|---|---|---|---|---|
| QT depth 0 | 41.84 | 7.81 | 45.83 | 4.52 | 90.3 | 84.3 | 87.2 |
| QT depth 1 | 19.53 | 0.58 | 72.57 | 7.32 | 72.7 | 97.1 | 83.1 |
| QT depth 2 | 2.69 | 0.08 | 94.67 | 2.57 | 51.1 | 97.1 | 67.0 |
| QT depth 3 | 0.02 | 0 | 99.92 | 0.06 | 25.0 | 100.0 | 40.0 |

In Figure 12, the accuracy of the *CandSplit* list decision is determined by whether the list contains the ground truth split at the MT level. We calculate and draw separate accuracy curves for MT0, MT1 and MT2 separately by varying the threshold *Thm*. As *Thm* increases, the size of the *CandSplit* list decreases, leading to decreasing precision. Once *Thm* reaches a certain value, the accuracy stabilizes because *CandList* is constant, containing only the MT split type with the highest probability and NS. It's worth noting that the minimum accuracy of the MT increases with the MT depth. This is mainly due to the fact that NS is more frequent at larger MT depths, as illustrated in the pie chart in Figure 11. Since our *CandSplit* list consistently includes NS, the accuracy tends to be relatively higher at larger MT depths.

In general, our model achieves a satisfactory F1 score for QT depths 0, 1 and 2 regarding the *SkipMT* decision. As for the *CandSplit* list decision, our algorithm maintains an accuracy exceeding 65% while adjusting the value of *Thm* at various MT levels. These performance evaluations justifies the high accuracy of the decisions made by our method during the partition search process in VVC.

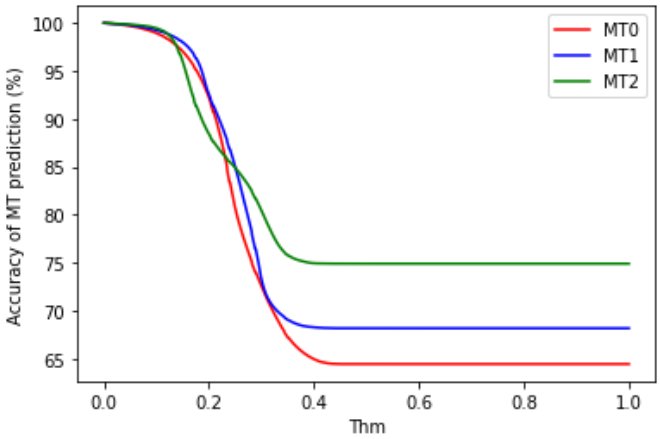



Fig. 12: Curves of accuracy and *Thm* for MT0, MT1, MT2

### B. Comparison with the State of the Art

The proposed method has been implemented in the VTM10.0 encoder using the Frugally deep library [37] for CPU-based inference in real time. To showcase the effectiveness of our method in the latest version of VTM, we conducted experiments using VTM21, as represented by the black curve in Figure 13. Encodings of CTC sequences are performed on a Linux machine with Intel Xeon E5-2697 v4 in a single-threaded manner. These experiments were conducted on the first 64 frames of CTC sequences with the RAGOP32 configuration on four QPs values of 22, 27, 32, 37.

Two metrics were used to assess the performance: BD-rate [38] and Time Saving (TS). The formula for computing TS is provided in Equation 6. Here, $T_{\text{Test}}$ denotes the encoding time of the proposed method, while $T_{\text{VTM}}$ represents the encoding time of the original VTM10 under the same conditions. The average BD-rate loss and Time Saving (TS) are computed as the arithmetic mean and geometric mean, respectively, on four QPs values over CTC sequences as defined in [39]. In addition, sequences of class D are excluded when computing the overall average performance.

$$TS = \frac{1}{4} \sum_{q \in \{22,27,32,37\}} \frac{T_{VTM}(q) - T_{\text{Test}}(q)}{T_{VTM}(q)} \quad (6)$$

The acceleration performances obtained from the state-of-the-art RF-based methods could not be directly compared with our performance. There are two main reasons for this. First of all, the results of [19] and [20] are based on VTM5.0 and VTM8.0, respectively. The differences of encoder complexity among various VTM versions are not negligible as highlighted in [40], which makes it less valid to directly compare our performances with theirs. Secondly, the training dataset was generated from a subset of CTC sequences, and the results were not obtained from the entire CTC. This approach results in possible overfitting and reduces the credibility of their results. As a result, comparing our results obtained on the entire CTC with their results is not fair.

[20] is an extended and specialized work for VVC based on [19]. We have reproduced the result of [20] in VTM10 to perform an unbiased comparison between our method

and RF-based method in [20]. First of all, we created a non-CTC dataset for training. Table III presents details on the composition of sequences for the dataset. For the 720p resolution, sequences are selected from [34] and sequences for other resolutions are from [33]. In the end, we generated a large dataset with $3.7e^7$ samples for the training of 17 Hor/Ver classifiers as well as $2.5e^6$ samples for the training of 4 QT/MTT classifiers. After generating the dataset, we trained, pruned and integrated the RF classifiers in VTM10.0. This was done in a manner consistent with the original article, including the implementation of the early termination rule for TT [2].

TABLE III: Breakdown of sequences used to train RFs of [20]

| | Resolution | | | | |
|---|---|---|---|---|---|
| Number of videos | 240p | 480p | 720p | 1080p | 4k |
| | 50 | 13 | 10 | 10 | 5 |

We reproduce the result of the medium and fast speed preset of [20] in VTM10. It should be noted that the maximum MT depth is limited to 2 for the fast preset. We plot the curve of BD-rate loss and TS of our method by gradually adjusting the threshold *Thm* and *QTskip* to build six settings. The curves obtained are shown in Figure 13. For example, the label (T, 0.125) signifies that in this particular setting, *QTskip* and *Thm* are assigned the values True and 0.125, respectively. Our method can achieve scalable acceleration varying from 16.5% to 60.2% with BD-rate loss ranging from 0.44% to 4.59%. Comparing with the fast preset, the setting (T, 0.175) produces the same acceleration with a 0.84% lower BD-rate loss. Similarly, the setting (T, 0) reaches the same BD-rate loss while providing a 17% higher speed-up compared to the medium preset. In summary, our method generally outperforms the state-of-the-art RF-based method. It is worth mentioning that the results in VTM21 are obtained by implementing our CNN model, which was originally trained on VTM10. Consequently, it is expected to exhibit reduced performance compared to the results in VTM10. Nonetheless, our method remains applicable and effective in the latest version of VTM.

Regarding CNN-based approaches, we compared our method with [21] and [15] in Table IV. The VTM version of [21] is VTM6. Thus we reimplement our method and integrate our model trained on VTM10 into VTM6 for a fair comparison within the same context. In Table IV, the reimplementation in VTM6 labeled as (T, 0, VTM6) reaches a slightly larger acceleration with only one-third of BD-rate loss compared to [21]. For [15], their VTM version is the same as ours, allowing for direct comparisons. Encoding with *Thm* = 0.125 yields a 40. 6% reduction in the encoding time, which is similar to the acceleration achieved by the C2 configuration in [15], but with only half of its BD-rate loss. Furthermore, our method with *Thm* set to 0.2 outperformed their C3 configuration, achieving a 0.52% lower BD-rate loss at the same level of acceleration. In conclusion, our method consistently outperforms all state-of-the-art methods.

[2]The code and dataset of reproduction is available at https://github.com/Simon123123/vtm10_fast_dt_inter_partition_pcs2021.git

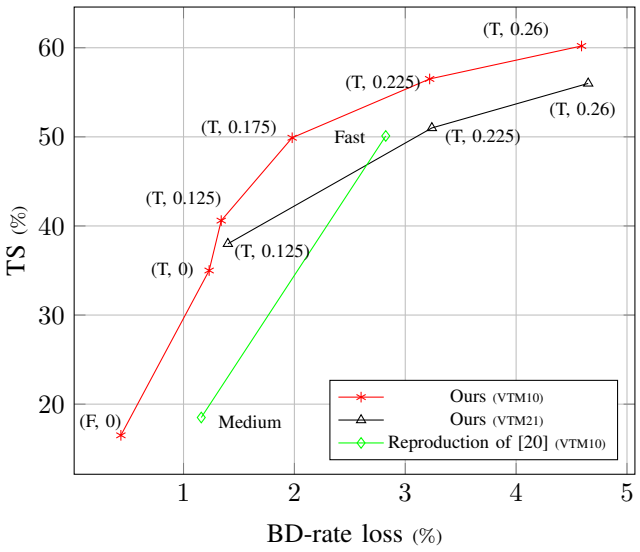



Fig. 13: Comparison of performances between proposed method and reproduction of [20].

It is important to note that the level of acceleration can vary depending on different sequence classes (e.g. resolution), which is consistent with other CNN-based methods. As discussed in [6], CTUs that exceed the picture boundary are called partial CTUs. These partial CTUs require a different partition search scheme compared to regular CTUs. Consequently, the encoding of partial CTUs are not accelerated since the CNN-based approaches are not applicable to them. Generally, the proportion of the frame region occupied by partial CTUs is larger for lower resolutions, resulting in less acceleration when fast partitioning approaches are used on smaller resolutions. This could partially explain the limited acceleration observed in class D which was excluded from the overall performance calculation. More specifically, our method tends to perform better on higher resolutions (e.g. class A and class B) while achieving less acceleration than state-of-the-art methods on lower resolutions (e.g. class C, class D and class E). Investigating and improving this aspect could be a focus of future work.

### C. Complexity Analysis

Machine learning-based fast partitioning methods may not be suitable for alternative implementations of the same codec. For example, VVenc [41] is a fast implementation of VVC. In the All Intra configuration, VTM10.0 is reported to be 27 times more complex compared to VVenc with fast preset, as mentioned in [42]. The overall complexity of the CNN-based method presented in [17] accounts for only 2.34% of the encoding time of the VTM10 encoder. However, when this method is implemented in VVenc without any adjustments, its overhead increases to about 67% of the encoding time with the fast preset, which means that this method is not directly applicable to VVenc. Consequently, it is crucial to develop a lightweight method to ensure its applicability across different implementations. Furthermore, lightweight methods do not

TABLE IV: Performance of the proposed method in comparison with reference CNN-based methods (Unit: %)

| Class | Sequence | Pan [21] (VTM6) | | Tissier [15] (C2) | | Tissier [15] (C3) | | Ours (T, 0, VTM6) | | Ours (T, 0.125, VTM10) | | Ours (T, 0.2, VTM10) | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | BD-rate | TS | BD-rate | TS | BD-rate | TS | BD-rate | TS | BD-rate | TS | BD-rate | TS |
| A (4k) | Tango2 | 4.03 | 38.56 | - | - | - | - | 1.35 | 32.3 | 1.84 | 43.7 | 3.08 | 57.5 |
| | FoodMarket4 | 1.74 | 46.12 | - | - | - | - | 0.75 | 29.4 | 0.85 | 55.1 | 1.13 | 53.6 |
| | Campfire | 3.17 | 38.23 | - | - | - | - | 1.49 | 40.7 | 1.83 | 48.5 | 3.22 | 63.2 |
| | CatRobot1 | 6.45 | 36.84 | - | - | - | - | 1.31 | 36.5 | 1.45 | 42.6 | 2.67 | 57.6 |
| | DaylightRoad2 | 5.63 | 35.47 | - | - | - | - | 1.57 | 39.4 | 2.00 | 45.6 | 3.94 | 57.9 |
| | ParkRunning3 | 2.10 | 26.45 | - | - | - | - | 0.99 | 42.6 | 0.98 | 45.9 | 1.93 | 59.8 |
| | **Average** | **3.85** | **36.46** | **1.84** | **47.7** | **3.06** | **59.7** | **1.25** | **37.0** | **1.49** | **45.3** | **2.66** | **58.4** |
| B (1080p) | MarketPlace | 4.33 | 33.64 | - | - | - | - | 0.99 | 37.6 | 1.48 | 46.3 | 2.78 | 57.7 |
| | RitualDance | 3.55 | 34.17 | - | - | - | - | 1.75 | 39.9 | 1.91 | 49.4 | 3.91 | 61.8 |
| | Cactus | 5.72 | 29.36 | - | - | - | - | 1.05 | 37.8 | 1.30 | 44.8 | 2.45 | 58.3 |
| | BasketballDrive | 3.30 | 37.28 | - | - | - | - | 1.34 | 39.6 | 1.95 | 49.7 | 3.65 | 63.4 |
| | BQTerrace | 1.90 | 20.21 | - | - | - | - | 0.99 | 32.6 | 1.18 | 39.8 | 2.23 | 52.2 |
| | **Average** | **3.76** | **30.27** | **2.21** | **46.5** | **3.09** | **58.2** | **1.22** | **37.5** | **1.56** | **46.1** | **3.00** | **58.9** |
| C (480p) | BasketballDrill | 2.29 | 29.23 | - | - | - | - | 1.04 | 26.9 | 1.08 | 30.3 | 2.60 | 39.7 |
| | BQMall | 2.69 | 27.48 | - | - | - | - | 1.20 | 29.1 | 1.18 | 32.2 | 2.71 | 39.7 |
| | PartyScene | 2.22 | 20.80 | - | - | - | - | 0.78 | 31.5 | 0.86 | 33.3 | 2.25 | 43.3 |
| | RaceHorses | 3.02 | 26.39 | - | - | - | - | 0.96 | 32.8 | 1.09 | 34.6 | 2.94 | 45.6 |
| | **Average** | **2.56** | **25.77** | **3.20** | **43.1** | **3.79** | **53.8** | **0.99** | **30.1** | **1.05** | **32.6** | **2.63** | **42.1** |
| D (240p) | BasketballPass | 1.85 | 26.97 | - | - | - | - | 0.76 | 19.0 | 0.85 | 22.2 | 1.72 | 25.1 |
| | BQSquare | 1.61 | 14.86 | - | - | - | - | 0.50 | 17.6 | 0.54 | 19.6 | 1.38 | 22.5 |
| | BlowingBubbles | 3.03 | 22.15 | - | - | - | - | 0.33 | 17.2 | 0.45 | 18.9 | 1.12 | 23.4 |
| | RaceHorses | 2.92 | 24.20 | - | - | - | - | 1.04 | 22.8 | 0.85 | 24.4 | 2.11 | 31.2 |
| | **Average** | **2.35** | **21.53** | **3.02** | **36.8** | **3.26** | **45.2** | **0.66** | **19.2** | **0.67** | **21.0** | **1.58** | **25.6** |
| E (720p) | FourPeople | 2.31 | 33.77 | - | - | - | - | 0.95 | 29.5 | 0.90 | 34.3 | 1.65 | 41.2 |
| | Johnny | 3.53 | 35.22 | - | - | - | - | 0.93 | 22.1 | 1.13 | 27.6 | 2.01 | 32.8 |
| | KristenAndSara | 2.58 | 36.50 | - | - | - | - | 1.00 | 24.3 | 1.11 | 30.3 | 1.73 | 36.4 |
| | **Average** | **2.81** | **35.15** | **1.45** | **38.7** | **2.2** | **49.6** | **0.96** | **25.4** | **1.04** | **30.8** | **1.79** | **36.9** |
| **Total average** | | **3.18** | **30.63** | **2.33** | **43.4** | **3.12** | **54.3** | **1.14** | **33.8** | **1.34** | **40.6** | **2.60** | **52.2** |

require parallel execution, enhancing the cost-effectiveness of such solutions.

TABLE V: Overhead of our method (Unit: %)

| | 240p | 480p | 720p | 1080p | 4k | Average |
|---|---|---|---|---|---|---|
| CNN | 0.23 | 0.37 | 0.99 | 0.90 | 0.84 | 0.60 |
| Preprocess | 0.24 | 0.41 | 1.15 | 0.81 | 0.86 | 0.62 |

As a result, we conducted a complexity analysis of our method to compare it with the state of the art. The overhead of a machine learning-based method typically consists of three components: preprocessing time, inference time, and postprocessing time. The post-processing of our method is integrated into the VVC partitioning process and introduces minimal overhead to the encoding process. However, preprocessing is necessary to compute the MS-MVF as model input. Table V provides the complexity of the preprocessing and the inference of CNN related to the encoding of the anchor VTM10. The last column corresponds to the geometric average of complexity for sequences from class A to E (including class D). Based on experimental results, the CNN inference time on a CPU accounts for only 0.60% of the total encoding time. Our approach consumes only 1.21% of the total encoding time, underscoring its lightweight nature.

Another important metric for evaluating the complexity of the model is its floating point operations (FLOPs). Our model has a FLOPs value of $1.12e^6$. In comparison, the FLOPs of the model in [43] is approximately $1.1e^9$ [16]. [16] employs a pruned ResNet-18 as the backbone with $9e^7$ FLOPs, and [15] utilizes the pretrained MobileNetV2 with $3.14e^8$ FLOPs. Our model is hundreds of times lighter than these methods. The lightweight nature of our proposed approach facilitates its adaptation to faster encoders.

## VIII. Conclusion

In this study, we propose a machine learning-based method to accelerate VVC inter partitioning. Our method leverages a novel representation of the QTMT partition structure based on partition path, consisting of QTdepthMap and MTsplitMaps. Our work is structured as follows. Firstly, we have built a large scale inter partition dataset. Secondly, a novel Unet-based model that takes MS-MVF as input is trained to predict the partition paths of CTU. Thirdly, we develop a scalable acceleration algorithm based on thresholds to utilize the output of the model. Finally, we speed up the VTM10 encoder under RAGOP32 configuration by 16.5%$\sim$60.2% with BD-rate loss of 0.44%$\sim$4.59%. This performance surpasses state-of-the-art methods in terms of coding efficiency and complexity trade-off. Notably, our method is among the most lightweight methods in the field, making it possible to adapt our approach to faster codecs.

For future work, we intend to investigate how video resolution influences partitioning acceleration, aiming to boost the speed-up of our method on lower resolutions. Furthermore, there is still acceleration potential lying in the selection of inter coding modes at the CU level, as discussed in [44]. An extension of our approach could be the incorporation of fast inter coding mode selection algorithm into our method to further accelerate the inter coding process.

## References

[1] Cisco. Cisco Annual Internet Report (2018–2023) White Paper, 2020.

[2] B. Bross, Y. Wang, Y. Ye, S. Liu, J. Chen, G.J. Sullivan, and J. Ohm. Overview of the versatile video coding (vvc) standard and its applications. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(10):3736–3764, 2021.

[3] Z. Wang, J. Zhang, N. Zhang, and S. Ma. Adaptive motion vector resolution scheme for enhanced video coding. In *2016 Data Compression Conference (DCC)*, pages 101–110, 2016.

[4] L. Li, H. Li, D. Liu, Z. Li, H. Yang, S. Lin, H. Chen, and F. Wu. An efficient four-parameter affine motion model for video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(8):1934–1948, 2018.

[5] A. Alshin, E. Alshina, and T. Lee. Bi-directional optical flow for improving motion compensation. In *28th Picture Coding Symposium*, pages 422–425, 2010.

[6] Y.W. Huang, J. An, H. Huang, X. Li, S.T. Hsiang, K. Zhang, H. Gao, J. Ma, and O. Chubach. Block partitioning structure in the vvc standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(10):3818–3833, 2021.

[7] A. Tissier, A. Mercat, T. Amestoy, W. Hamidouche, J. Vanne, and D. Menard. Complexity reduction opportunities in the future vvc intra encoder. In *2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6. IEEE, 2019.

[8] Y. Fan, J. Chen, H. Sun, J. Katto, and M'E. Jing. A fast qtmt partition decision strategy for vvc intra prediction. *IEEE Access*, 8:107900–107911, 2020.

[9] J. Cui, T. Zhang, C. Gu, X. Zhang, and S. Ma. Gradient-based early termination of cu partition in vvc intra coding. In *2020 Data Compression Conference (DCC)*, pages 103–112, 2020.

[10] J. Chen, H. Sun, J. Katto, X. Z., and Y. Fan. Fast qtmt partition decision algorithm in vvc intra coding based on variance and gradient. In *2019 IEEE Visual Communications and Image Processing (VCIP)*, pages 1–4, 2019.

[11] M. Lei, F. Luo, X. Zhang, S. Wang, and S. Ma. Look-ahead prediction based coding unit size pruning for vvc intra coding. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 4120–4124, 2019.

[12] M. Saldanha, G. Sanchez, C. Marcon, and L. Agostini. Fast partitioning decision scheme for versatile video coding intra-frame prediction. In *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, 2020.

[13] T. Fu, H. Zhang, F. Mu, and H. Chen. Fast cu partitioning algorithm for h.266/vvc intra-frame coding. In *2019 IEEE International Conference on Multimedia and Expo (ICME)*, pages 55–60, 2019.

[14] F. Galpin, F. Racapé, S. Jaiswal, P. Bordes, F. Le Léannec, and E. François. Cnn-based driving of block partitioning for intra slices encoding. In *2019 Data Compression Conference (DCC)*, pages 162–171. IEEE, 2019.

[15] A Tissier, W Hamidouche, J Vanne, and D Menard. Machine learning based efficient qt-mtt partitioning for vvc inter coding. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 1401–1405. IEEE, 2022.

[16] S. Wu, J. Shi, and Z. Chen. Hg-fcn: Hierarchical grid fully convolutional network for fast vvc intra coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(8):5638–5649, 2022.

[17] A. Feng, K. Liu, D. Liu, L. Li, and F. Wu. Partition map prediction for fast block partitioning in vvc intra-frame coding. *IEEE Transactions on Image Processing*, 32:2237–2251, 2023.

[18] M. Saldanha, G. Sanchez, C. Marcon, and L. Agostini. Configurable fast block partitioning for vvc intra coding using light gradient boosting machine. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(6):3947–3960, 2021.

[19] T. Amestoy, A. Mercat, W. Hamidouche, D. Menard, and C. Bergeron. Tunable vvc frame partitioning based on lightweight machine learning. *IEEE Transactions on Image Processing*, 29:1313–1328, 2020.

[20] G. Kulupana, V.P. Kumar M, and S. Blasi. Fast versatile video coding using specialised decision trees. In *2021 Picture Coding Symposium (PCS)*, pages 1–5, 2021.

[21] Z. Pan, P. Zhang, B. Peng, N. Ling, and J. Lei. A cnn-based fast inter coding method for vvc. *IEEE Signal Processing Letters*, 28:1260–1264, 2021.

[22] W. Yeo and B.G. Kim. CNN-based Fast Split Mode Decision Algorithm for Versatile Video Coding (VVC) Inter Prediction. *Journal of Multimedia Information System*, 8(3):147–158, 2021.

[23] Y. Liu, M. Abdoli, T. Guionnet, C. Guillemot, and A. Roumy. Lightweight cnn-based vvc inter partitioning acceleration. In *2022 IEEE 14th Image, Video, and Multidimensional Signal Processing Workshop (IVMSP)*, pages 1–5. IEEE, 2022.

[24] A. Tissier, W. Hamidouche, SBD. Mdalsi, J. Vanne, F. Galpin, and D. Menard. Machine learning based efficient qt-mtt partitioning scheme for vvc intra encoders. *IEEE Transactions on Circuits and Systems for Video Technology*, 2023.

[25] A Wieckowski, J Ma, H Schwarz, D Marpe, and T Wiegand. Fast partitioning decision strategies for the upcoming versatile video coding (vvc) standard. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 4130–4134. IEEE, 2019.

[26] G.J. Sullivan and T. Wiegand. Rate-distortion optimization for video compression. *IEEE Signal Processing Magazine*, 15(6):74–90, 1998.

[27] O Ronneberger, P Fischer, and T Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[28] J Long, E Shelhamer, and T Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[29] Z. Chen, J. Shi, and W. Li. Learned fast hevc intra coding. *IEEE Transactions on Image Processing*, 29:5431–5446, 2020.

[30] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[31] Z. Wang, S. Wang, X. Zhang, S. Wang, and S. Ma. Fast qtbt partitioning decision for interframe coding with convolution neural network. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 2550–2554, 2018.

[32] Y. Wang, R. Skupin, M. Hannuksela, S. Deshpande, V. Drugeon, R. Sjöberg, B. Choi, V. Seregin, Y. Sanchez, J. Boyce, et al. The high-level syntax of the versatile video coding (vvc) standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(10):3779–3800, 2021.

[33] D. Ma, F. Zhang, and D.R. Bull. Bvi-dvc: A training database for deep video compression. *IEEE Transactions on Multimedia*, 24:3847–3858, 2021.

[34] Y. Wang, S. Inguva, and B. Adsumilli. Youtube ugc dataset for video compression research. In *2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–5. IEEE, 2019.

[35] Y. Ye J. Chen and S. Kim. Algorithm description for versatile video coding and test model 10 (vtm 10). Technical Report document JVET-S2002, JVET, 2020.

[36] Adam: A method for stochastic optimization, author=Kingma, D.P. and Ba, J. *arXiv preprint arXiv:1412.6980*, 2014.

[37] Tobias Hermann. Frugally-Deep. https://github.com/Dobiasd/frugally-deep, 2018.

[38] G. Bjontegaard. Calculation of average PSNR differences between rd-curves. *VCEG-M33*, 2001.

[39] J Boyce, K Suehring, X Li, and V Seregin. Jvet common test conditions and software reference configurations. Technical Report document JVET-J1010, JVET, 07 2018.

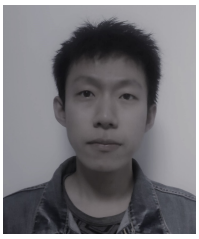
[40] Gary Sullivan. Versatile Video Coding (VVC) Delivers: Coding Efficiency and Beyond, DCC, 2021.

[41] A. Wieckowski, J. Brandenburg, T. Hinz, C. Bartnik, V. George, G. Hege, C. Helmrich, A. Henkel, C. Lehmann, C. Stoffers, I. Zupancic, B. Bross, and D. Marpe. Vvenc: An open and optimized vvc encoder implementation. In *2021 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, pages 1–2, 2021.

[42] I. Taabane, D. Menard, A. Mansouri, and A. Ahaitouf. Machine learning based fast qtmtt partitioning strategy for vvenc encoder in intra coding. *Electronics*, 12(6), 2023.

[43] T. Li, M. Xu, R. Tang, Y. Chen, and Q. Xing. Deepqtmt: A deep learning approach for fast qtmt-based cu partition of intra-mode vvc. *IEEE Transactions on Image Processing*, 30:5377–5390, 2021.

[44] Y Liu, M Abdoli, T Guionnet, C Guillemot, and A Roumy. Statistical analysis of inter coding in vvc test model (vtm). In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 3456–3459, 2022.



**Yiqun Liu** received the Engineering degree in electronic information engineering from the Institut national des sciences appliquées de Rennes (INSA Rennes), France, in 2019. He is currently pursuing the Ph.D. degree in partnership between the Institut National de Recherche en Informatique et en Automatique (INRIA) and the ATEME company, France. His research interests concern video compression and the real-time implementations of the new generation video coding standards.

**Thomas Guionnet** is a fellow research engineer at ATEME, where he currently leads the innovation team's research on artificial intelligence applied to video compression. Beyond his work for ATEME, he has also contributed to the ISO/MPEG - ITU-T/VCEG - VVC, HEVC, and HEVC-3D standardization process; he teaches video compression at the ESIR Engineering School, Rennes, France; and he has authored numerous publications including patents, international conference papers, and journal articles. Prior to joining ATEME, he spent 10 years at Envivio conducting research on real-time encoding, video-preprocessing, and video quality assessment. He holds a PhD from Rennes 1 University, Rennes.

**Marc Riviere** received the MSc degree from the École Nationale de la Statistique et de l'Analyse de l'Information (ENSAI) with highest honors and the MSc degree from the École Nationale Supérieure de Physique de Strasbourg (ENSPS). He joined the CTO Office of ATEME in 2022 as a Senior Data Scientist, applying Artificial Intelligence to various tasks including video analysis, video compression and video distribution. Prior to that, he was an Innovation Architect at the CTO Office of Technicolor Connected Home from 2019 to 2022, where he has notably designed and promoted a large scale data analysis platform. From 2004 to 2018 and within several companies, he has accumulated significant experience in very different topics such as robotics, embedded software, cybersecurity, video watermarking, trusted execution environments, software defined networks and network function virtualization. His current research interests include the areas of machine and deep learning, video compression, computer vision and automatic speech recognition.

**Aline Roumy** received the Engineering degree from École Nationale Supérieure de l'Éléctronique et de ses Applications (ENSEA), Cergy, France, in 1996, the Master's degree in June 1997, and the Ph.D. degree from the University of Cergy-Pontoise, France, in September 2000. From 2000 to 2001, she was a Research Associate at Princeton University, Princeton, NJ, USA. In November 2001, she joined INRIA Rennes, France. Her current research and study interests include areas of statistical signal processing, coding theory, and information theory. She was a recipient of a French Defense DGA/DRET Postdoctoral Fellowship from 2000 to 2001.

**Christine Guillemot, IEEE Fellow** received the Ph.D. degree from the École Nationale Supérieure des Télécommunications (ENST), Paris, and the Habilitation degree in research direction from the University of Rennes. From 1985 to 1997, she was with France Télécom, where she was involved in various projects in the area of image and video coding for TV, HDTV, and multimedia. From 1990 to 1991, she was a Visiting Scientist with Bellcore, NJ, USA. She is currently Director of Research with the Institut National de Recherche en Informatique et en Automatique (INRIA) and the Head of a research team dealing with image and video modeling, processing, coding, and communication. Her research interests are signal and image processing, and, in particular, 2D and 3D image and video processing for various problems, such as compression, and inverse pproblems such as restoration, super-resolution, inpainting. Dr. Guillemot served as a Senior Member of the Editorial Board of the IEEE JOURNAL OF SELECTED TOPICS IN SIGNAL PROCESSING from 2013 to 2015. She has served as an Associate Editor for IEEE TRANSACTIONS ON IMAGE PROCESSING from 2000 to 2003 and from 2014 to 2016, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY from 2004 to 2006, and IEEE TRANSACTIONS ON SIGNAL PROCESSING from 2007 to 2009. She has also been Senior Area Editor of the IEEE TRANSACTIONS ON IMAGE PROCESSING (2016–2020).