International Journal of Advances in Electronics and Computer Science, ISSN (p): 2394-2835 Volume-10, Issue-9, Sept.-2023
http://iraj.in

CLASSIFICATION ALGORITHM ANALYSIS FOR TEXTURE DETECTION IN BLOCK-BASED HYBRID VIDEO CODING

[1]MIGUEL ONOFRE MARTÍNEZ-RACH, [2]JAVIER RUIZ ATENCIA, [3]OTONIEL LÓPEZ-GRANADO, [4]MANUEL PÉREZ MALUMBRES

[1,2,3,4] Department of Computer Engineering, Miguel Hernández University, E-03202, Elche, Alicante, Spain
E-mail: [1]mmrach@umh.es,[2]javier.ruiza@umh.es, [3]otoniel@umh.es, [4]mels@umh.es

Abstract - This study presents a comprehensive comparison of various supervised classification algorithms for texture detection in the context of block-based hybrid video coding. To accomplish this, a dataset of images extracted directly from video encoder block partitions was created and manually classified according to their texture levels. The study utilizes the Mean Directional Variance (MDV) algorithm to extract orientation information from each block in the form of average variances for specific rational slopes. This vector of variances is then processed to obtain a set of descriptive statistics that serve as input elements for training and evaluating four popular supervised learning models: Decision Tree (DT), Random Forest (RF), Support Vector Machine (SVM), and Supervised Neural Networks (SNN). The objective is to identify the most effective algorithm for accurately classifying texture levels and utilizing this information in perceptual video coding.

Keywords - Classification Algorithms, Texture Detection, MDV, Perceptual Coding.

I. INTRODUCTION

In perceptual video coding, the information about the level of texture in the blocks into which an image is divided is essential for exploiting the texture masking effect provided by the human visual system (HVS). According to the HVS, quantization noise or errors are less perceptible in regions with a high level of texture, while they are much more noticeable in low-texture areas. Therefore, proper detection of texture information in perceptual video coding becomes particularly relevant to ensure the correct operation of perceptual coding techniques that leverage the characteristics of the HVS to compress in regions where the human eye is unable to detect these errors.

The detection of texture level in perceptual video coding requires robust and accurate classification algorithms. These algorithms should be capable of reliably identifying the type of texture present in each block and assigning the corresponding label. The choice of an efficient classification algorithm has a direct impact on the performance of perceptual coding algorithms and, therefore, on the visual quality of the video.The motivation behind this comparative analysis lies in the need to evaluate and compare different classification algorithms for texture detection in perceptual video coding. While there are various approaches and methods proposed in the literature, it is essential to understand their strengths and limitations to select the most suitable algorithm for a specific application. To conduct this study, a dataset of blocks of different sizes, ranging from 8×8 to 64×64 pixels, has been created, and these blocks have been manually classified according to their texture level.In this study, we present a comprehensive comparative analysis of different classification algorithms for texture detection in video coding blocks. We will evaluate and compare the performance of these algorithms in terms of accuracy and efficiency. The obtained results will provide a detailed insight into the strengths and limitations of each algorithm and will serve as a guide for future research and developments in the field of perceptual video coding.

II. METHODOLOGY

2.1. Dataset description

To prepare the training dataset, over 2000 blocks have been randomly extracted from ESPL Synthetic Image Database [1], USC-SIPI Image Database [2], TESTIMAGE [3], and Kodak image dataset [4]. These blocks have been partitioned into sizes of 8×8, 16×16, 32×32, and 64×64 pixels. Each block is manually labeled according to its texture level and can be classified as Plain (generally smooth with low spatial activity), Edge (with a clearly marked edge or direction), or Texture (with high spatial activity).These block sizes have been selected as they are the most representative in frame partitioning for current video codecs. The use of 4×4 blocks has been discarded as it is difficult to determine optimal classification with such few pixels.The choice of labeling with these three texture levels is not only based on its widespread use in the literature over the years, as shown in articles [5][6][7][8], but also because of its implications in the human visual system. In [5], the authors demonstrated that Texture blocks have more tolerance for quantization errors without being detected, while Plain blocks should not have any additional errors applied. Finally, Edge blocks can tolerate a certain level of quantization error, but to a lesser extent compared to Texture blocks.The block labeling process has been carried out manually using a graphical environment developed in MATLAB. For each classification, the tool displays the original block as well as its smoothed version using a median filter. The

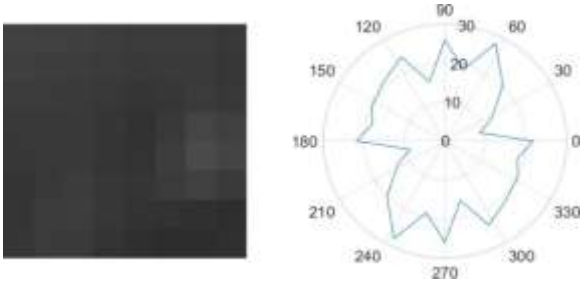
smoothed version has been added to assist in determining the texture level in cases where it is difficult to determine from the original block, such as in the boundary with Plain blocks. The final dataset contains a variety of blocks with different sizes and texture levels, providing a broad representation of the characteristics found in video coding. Table 1 details the distribution of labeled blocks according to their label and size. It can be observed that the dataset distribution is imbalanced; however, this will not pose any issues as the algorithms used and the evaluation metrics take this into account during the training process. This manually labeled dataset serves as the basis for training and evaluating the classification algorithms in this comparative study.

Size	Label			Total
	Plain	Edge	Texture	
8	128	129	103	360
16	227	165	135	527
32	182	145	157	484
64	180	168	283	631
Total	717	607	678	2002

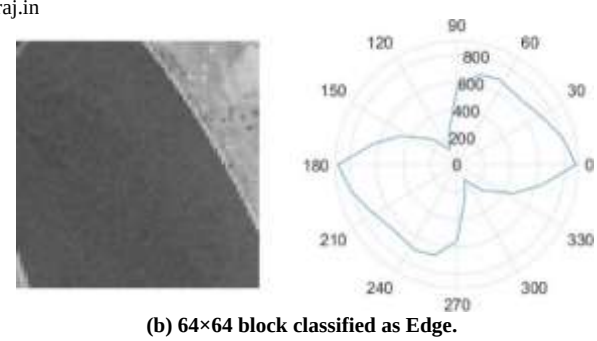
Table1: Distribution of the dataset according to labeling and block size

2.2. Texture Detection based on MDV

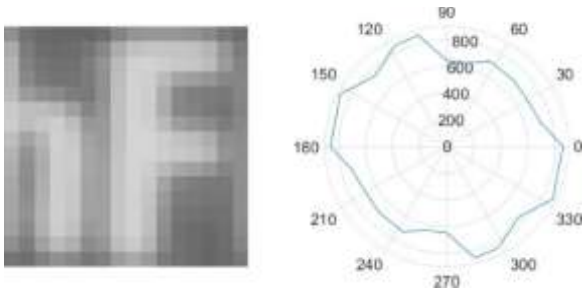
After constructing the dataset, it has been decided to discard the use of Convolutional Neural Networks (CNN) for the classification model. CNNs are widely used and effective for the classification of complex and high- resolution images, as they are designed to leverage the spatial structure of images by using convolutional and pooling layers to extract local features and build a hierarchical representation of the image.However, our dataset consists of low-resolution blocks, where the spatial information and pixel relationships are not rich enough to fully exploit the capabilities of these neural networks. Additionally, incorporating this architecture into the video codec would incur a significant computational cost. For example, in [9], decoding times increased by 11656%. Therefore, instead of using pixel intensities directly as input for the models, an alternative approach has been chosen. The Mean Directional Variance (MDV) algorithm [10][11] is utilized to obtain the gradient orientation of the blocks, as there is a strong relationship between the MDV values and the texture level of the blocks.



(a) 8×8 block classified as Plain.



(b) 64×64 block classified as Edge.



(c) 16×16 block classified as Texture.

Fig.1. Sample of manually labeled blocks (left) and their polar representation of the MDV metric (right).

The MDV algorithm calculates the mean directional variance along certain spatial directions with rational slopes. In [11], the authorsuse a total of 24 rational slopes to improve the estimation of the dominant directionality. However, for our purpose, such a level of precision is not necessary. Computing MDV using only 12 slopes is sufficient. One of the main advantages of working with this algorithm is that it yields a vector of 12 gradients for each block, regardless of its size. This simplifies the training process and allows for the use of simpler classification models.

Figure 1 illustrates three examples of manually classified blocks alongside their polar diagram representation of the MDV metric. It is evident that there is a strong relationship between the texture level of a block and its associated MDV metric. For Plain blocks, the shape of the polar diagram may vary, but the value of the direction with the highest variance is relatively low compared to the other two block types. For Edge blocks, there is always a minimum (dominant gradient) in the direction of the block's orientation. Finally, for Texture blocks, the data extracted from the metric is high, lacking a dominant minimum.

Based on this observation, some descriptive statistics of the gradient vector have been extracted for each manually labeled block, including minimum value, maximum value, average, range, and variance. By plotting these statistics on a scatter plot, as shown in Figure 2, it can be observed that the blocks form well- defined clusters according to their texture level. This demonstrates the strong relationship between the texture level and the previously mentioned MDV

metric observation, enabling the use of these statistics as input elements or features for the different training models used in this study.

2.3. Supervised Classification Model Selection

The selection of supervised learning models used in this comparative study is based on their relevance and proven effectiveness in detecting and classifying patterns from predictive variables. Each of these models has distinct characteristics that make them suitable for approaching this problem from different perspectives.

Decision Tree (DT): is an intuitive and easy-to-interpret classification model. It constructs a tree-shaped model where each internal node represents a feature or attribute, and each leaf represents a class label. The algorithm recursively splits the training data based on feature values, aiming to maximize class separation at each step. By following paths from the root to the leaves, the decision tree assigns class labels to new instances based on their feature values.

Random Forest (RF): This model combines multiple decision trees, known as a forest, to make predictions. Each decision tree is constructed using a random subset of features and training samples, promoting diversity, and reducing the risk of overfitting. During the prediction phase, the individual predictions of each tree are combined to achieve a final class label through majority voting or averaging.

Support Vector Machine (SVM): Its goal is to find an optimal hyperplane that separates different classes by maximizing the margin between them. This is achieved by transforming the input data into a higher-dimensional space and identifying the optimal decision boundary. It assigns data points to different classes based on their relative position to the hyperplane. SVM is particularly effective in handling complex datasets with nonlinear relationships using kernel functions.

Supervised Neural Networks (SNN): These models are inspired by the structure and functionality of biological neural networks. These networks can learn complex patterns and features from data through a training process with labeled data. SNNs consist of interconnected artificial neurons organized in layers. The algorithm learns by adjusting the weights and biases of neurons through a process called backpropagation, where errors between predicted and actual class labels are minimized.

All four algorithms are relevant in the context of data classification due to their characteristics and capabilities. DT and RF are especially suitable for capturing patterns and rules based on specific features of image blocks. SVMs are effective in classifying linearly inseparable data and can find complex decision boundaries. SNNs, on the other hand, can learn and capture complex features. Considering simplicity and robustness, these four algorithms provide a solid foundation for comparing and evaluating their performance in our study.

2.4. Feature selection and preprocessing

In Section 2.2, it was determined that the set of features or input elements for the evaluated learning models should not be the pixel intensities of the blocks, but rather their gradient vector obtained using the MDV algorithm. From this vector, numerous statistics have also been extracted, which can also be used as features since they are strongly related to the labeling or classification of the blocks. Lastly, an additional feature that has been considered is the indication of the block size.

Before using this set of input data for training and evaluating the classification algorithms, several data preprocessing steps have been performed, depending on the type of algorithm, to ensure the quality and consistency of the results. The main steps involved in this preprocessing are as follows:

Dataset Split: This involves separating the dataset into two groups: the training group and the testing group. This allows for evaluating the generalization ability of the classification models. The assigned proportion has been 80% for the training group and 20% for the testing group. The input dataset has been divided into two parts, the training part, and the testing part, ensuring that both contain a balanced representation of Texture, Edge, and Plain blocks. This division has been maintained for all four evaluated classification algorithms.

Cross-Validation: Due to the relatively small size of the dataset used, the cross-validation strategy has been adopted, specifically using the StratifiedShuffleSplit function from the Scikit-Learn (sklearn) library. This function stratifies and randomly divides the input data into training and testing sets. This allows for validating that the obtained results are independent of the training and testing sets, providing greater reliability to the results. In this study, a total of 5 splits have been selected for the four evaluated algorithms.

Data Standardization: This involves transforming the input data, subtracting the mean, and scaling each feature to achieve a unit standard deviation. This ensures that the features are in a consistent range, which helps to avoid biases in the training of the models. The StandardScaler function from sklearn library has been used for standardization. Unlike the previous steps, this step has only been applied to the SVM and SNN models and not to the DT and RF models. This is because decision tree-based algorithms rely on simple rules that directly compare the feature values at each node and are not affected by the scale of the features.

Data Augmentation: This step has been used only for the SNN model and involves artificially increasing the size of the dataset by applying various transformations to the manually labeled blocks. These

transformations include rotating each block by 90, 180, and 270 degrees, as well as applying a flip, resulting in a seven-fold increase in the size of the original dataset. However, for all these transformations, the descriptive statistics remain constant, so data augmentation has no impact when using these statistics as input elements.

2.5. Model training

The implementation of different machine learning algorithms has been carried out using the Scikit-Learn (sklearn) and Keras libraries for Python 3.10. These libraries offer a wide range of tools and functions for implementing and evaluating machine learning algorithms.

To systematically search for optimal hyperparameters, different methods have been used depending on the model. For the DT, RF, and SVM algorithms, the GridSearchCV class has been employed. This class allows for exploring different combinations of parameters and selecting the configuration that maximizes the model's performance. The parameters used for the hyperparameter search are listed in Table 2.

Algorithm	Hyperparameters
Decision Tree	(criterion, max_depth)
Random Forest	(criterion, max_depth, max_features, n_estimators)
SVM	(C, kernel='linear',decision_function_shape), (C, kernel='rbf', gamma,decision_function_shape)

Table2: List of classification algorithms using the GridSearchCV class and their associated hyperparameters.

For supervised neural networks, the KerasTuner library has been used for hyperparameter tuning. KerasTuner provides a simple and efficient interface for optimizing supervised neural networks. This tool allows for defining the hyperparameter search space and automates the process of tuning the models, making it easier to obtain optimal results.

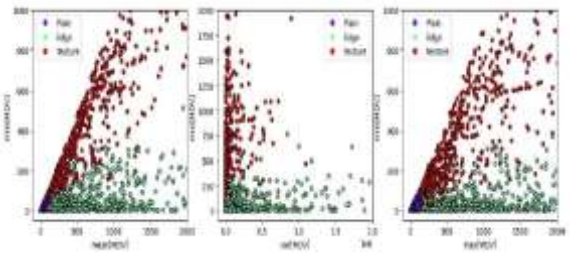


Fig.2. Scatter plot of manually labeled blocks according to some of their descriptive statistics. From left to right: minimum vs. average, minimum vs. variance, and minimum vs. maximum.

In the design of SNN models for this study, three different models have been developed, each with a specific configuration of layers and dropout. The main objective of this variation is to explore different architectures and evaluate their performance. All these models are compiled using the Adam optimizer, with a loss function called categorical crossentropy, commonly used in multi-class classification problems like ours. In addition to the loss function, two additional evaluation metrics, recall and precision, are defined to provide an additional measure of the model's performance during training and evaluation. These metrics will be detailed in Section 2.6. Regarding the architecture of the models, the first model consists of an input layer that receives the input data. Then, a regularization technique called dropout is applied, randomly deactivating a certain percentage of neurons from the previous layer during training. This helps prevent overfitting and improve the model's generalization. Finally, an output layer is included. The second SNN model has a slightly more complex architecture. In addition to the input layer and initial dropout, an intermediate layer and another dropout before the output layer are added. The third SNN model incorporates an additional intermediate layer and another dropout after it. Each of these layers, as well as the adjustments in terms of learning rate and dropout rate in the three models, has undergone an exhaustive hyperparameter search using KerasTuner. This allows for tuning the parameters of each layer, such as the number of neurons, the type of activation function, and the dropout rate, as well as the learning rate of the model, to find the optimal configuration that maximizes the model's performance. Regarding the input data or features, given the clear segmentation of the blocks based on the descriptive statistics of the gradient vectors seen in Figure 2, it has been found that using the statistics instead of the gradient vectors is more than sufficient for all the algorithms evaluated in this study. Specifically, the following statistics have been used: minimum, maximum, average, and variance. In addition to these statistics, one more feature has been considered, which is the size of the block, slightly improving the results obtained. Nevertheless, for the models based on neural networks, it has been decided to also test the option of directly using the gradient vector plus the block size. For this second option, identified as SNN*, the data augmentation preprocessing explained in Section 2.4 has been applied.

2.6. Model evaluation

To assess the performance of different machi- ne learning algorithms and maximize the hyperparameter search in SNN models, the following metrics have been used:

Precision: It measures the proportion of instances correctly classified as positive among all instances classified as positive by the model. In other words, it

calculates how many of the model's positive predictions are true positive. It is calculated by dividing the number of true positives (TP) by the sum of true positives and false positives (FP). It is calculated using the following formula: Precision=TP/(TP+FP).

Recall: It measures the proportion of positive instances that were correctly identified by the model compared to all actual positive instances. In other words, it calculates how many positive instances were correctly detected by the model. It is calculated by dividing the number of true positives (TP) by the sum of true positives and false negatives (FN). It is calculated using the following formula: Recall=TP/(TP + FN).

F1-score: It is a measure that combines precision and recall into a single value. It is the harmonic mean of precision and recall and provides a balanced measure of the model's performance. It is calculated using the following formula:

F1=2∗(Precision∗Recall)/(Precision+Recall).

These metrics are useful for verifying and comparing machine learning models on imbalanced datasets as they provide a comprehensive evaluation of the model's performance, considering both positive and negative cases.

III. RESULTS AND DISCUSSION

The results obtained for each optimized machine learning algorithm, aiming to achieve the highest performance, are presented in Table 3. These results reflect the models' performance in terms of their ability to correctly classify the blocks in the dataset. It can be observed that the SVM, SNN (Model 2), and RF algorithms show the best results for all evaluated metrics. These three models achieve a notable balance between precision and the ability to correctly identify positive cases.

Algorithm	Precision	Recall	F1-Score
Decision Tree	0.9350	0.9347	0.9347
Random Forest	0.9600	0.9596	0.9596
SVM	0.9624	0.9621	0.9621
SNN (Model 1)	0.9423	0.9423	0.9426
SNN (Model 2)	0.9579	0.9576	0.9575
SNN (Model 3)	0.9438	0.9402	0.9396
SNN* (Model 1)	0.9078	0.9077	0.9086
SNN* (Model 2)	0.9329	0.9326	0.9327
SNN* (Model 3)	0.9279	0.9280	0.9278

SNN* models correspond to those trained with the gradient vector as input

Table3: Average results of different machine learning models configured with the found optimal parameters.

Algorithm	Parameter	Value
Decision Tree	criterion	gini
	max_depth	8
Random Forest	criterion	log_loss
	max_depth	10
	max_features	sqrt
	n_estimators	9
SVM	C	1000
	kernel	linear
	decision_function_shape	ovr
SNN (Model 2)	units layer 1	12
	dense_activation 1	relu
	dropout 1	0
	units layer 2	12
	dense_activation 2	sigmoid
	dropout 2	0
	learning_rate	0.1
	epochs	14
SNN* (Model 2)	units layer 1	12
	dense_activation 1	relu
	dropout 1	0
	units layer 2	12
	dense_activation 2	tanh
	dropout 2	0.1
	learning_rate	0.01
	epochs	70

Table4: List of optimal parameters found for each evaluated machine learning algorithm

It is important to highlight the performance achieved by the Decision Tree (DT) algorithm, despite being lower compared to the other three models. Its simplicity allows for real-time applications, which is advantageous in certain scenarios.Regarding the supervised neural networks, it is observed that using the gradient vector directly (SNN* models) as input features results in lower performance compared to using their descriptive statistics (SNN models). However, it is worth mentioning that only architectures with up to 3 layers were evaluated. As shown in Table 2, the performance of the 3-layer SNN* model is slightly lower than the 2-layer model.It is possible that with four or more layers, the performance could be improved.

In particular, the SNN models show the best results with two hidden layers, although results with a single layer are also promising. This simpler architecture to implement may be preferable in certain cases.

Regarding the search for optimal hyperparameters, Table 4 shows the selected values for each of the algorithms presented in this study. For the SNN and SNN* models, only the optimal parameters for the architecture with the highest performance within their class are presented.

IV. CONCLUSION

In this study, a training dataset was developed for the classification of blocks based on their texture level in

the context of block-based hybrid video coding. An exhaustive analysis of various supervised learning algorithms was conducted to automate block classification and determine which algorithm offers the best performance for application in perceptual video coding.The results obtained revealed that the SVM, Supervised Neural Networks (SNN), and Random Forest (RF) algorithms showed the best results in terms of the main evaluated metrics. These models achieved a notable balance between precision and classification ability, positioning them as promising options for block classification in video coding.

An interesting finding of this study was the advantage of using descriptive statistics of the MDV metric for texture detection. The use of these statistics, such as minimum, maximum, average, and variance, along with block size, proved to be sufficient to achieve good results across all evaluated algorithms. This allows for promising results using simpler features with lower computational cost. Regarding future research, several avenues are suggested. First, it would be interesting to perform a comparison with other existing texture classification or detection methods in the literature, which would strengthen the results obtained in this study. Furthermore, the dataset size could be increased, as the 2000 samples distributed among different block sizes are all square blocks, and the new block-based hybrid video encoders include non-square blocks. Additionally, other descriptive metrics or additional features could be explored to improve the texture detection capability in coding blocks. For example, considering features related to the spatial distribution of pixels or using more advanced feature extraction techniques could provide additional information and improve classification accuracy.

ACKNOWLEDGMENTS

This research was supported by the Spanish Ministry of Science and Innovation and the Research State Agency under Grant PID2021-123627OB-C55 co-financed by FEDER funds (MCIU/AEI/FEDER, UE)and by the Valencian Ministry of Innovation, Universities, Science and Digital Society (Generalitat Valenciana) under Grants CIAICO/2021/278.

REFERENCE

[1] Debarati Kundu and Brian L. Evans,"Full-reference visual quality assessment for synthetic images:Asubjectivestudy,"in2015IEEEInternationalConferenceonImageProcessing(ICIP), 2015,pp.2374–2378.

[2] UniversityofSouthernCalifornia,Signal andImage Processing Institute,"The USC-SIPI Image Database,".

[3] NicolaAsuniandAndreaGiachetti, "TESTIMAGES:aLarge-scaleArchiveforTestingVisualDevicesandBasicImageProcessingAlgorithms," in Smart Tools and Apps for Graphics - Eurographics Italian Chapter Conference,AndreaGiachetti,Ed.2014,TheEurographicsAssociation.

[4] Kodak,"TheKodakColorImageDataset,".

[5] H.H.Y.TongandA.N.Venetsanopoulos,"Aperceptual model for JPEG applications based on blockclassification,texturemasking,andluminancemasking,"in Proceedings 1998 International Conference on Image Processing. ICIP98 (Cat.No.98CB36269),Oct1998,pp.428–432 vol.3.

[6] X.H. Zhang, W.S. Lin, and P. Xue, "Improved estimation for just-noticeable visual distortion," Signal Processing, vol. 85, no. 4, pp. 795 – 808, 2005

[7] TaoXiang,ShangweiGuo,andXiaoguoLi,"PerceptualVisualSecurityIndexBasedonEdgeand TextureSimilarities," IEEE Transactions on Information Forensics and Security, vol. 11, no. 5, pp.951–963,2016.

[8] DinaA.Abdulqader,MohammedSadoonHathal, BasheeraM.Mahmmod,SadiqH.Abdulhussain, andDhiyaAl-Jumeily,"Plain,Edge,andTexture DetectionBasedonOrthogonalMoment," IEEE Access,vol.10,pp.114455–114468,2022.

[9] Chuanmin Jia, Shiqi Wang, Xinfeng Zhang, Shanshe Wang, Jiaying Liu, Shiliang Pu, and SiweiMa,"Content-AwareConvolutionalNeural NetworkforIn-LoopFilteringinHighEfficiency Video Coding,"IEEE Transactions on Image Processing,vol.28,no.7,pp.3343–3356,2019.

[10] DamiánRuiz-Coll,GerardoFemández-Escribano, JoséLuisMartínez,andPedroCuenca,"Fastintra mode decision algorithm based on texture orientationdetectioninHEVC," Signal Processing: ImageCommunication,vol.44,pp.12–28,Mar2016.

[11] Elena Georgiana Paraschiv, Damián Ruiz-Coll Maria Pantoja, and Gerardo Fernández-Escribano,"Parallelization and improvement of the MDV-SW algorithm for HEVC intra-predictioncoding," TheJournalofSupercomputing,vol.75,pp.1150–1162,Mar2019.

★★★