

Fully Neural Network Mode Based Intra Prediction of Variable Block Size

Heming Sun^{*†}, Lu Yu[‡], Jiro Katto^{*§}

^{*}Waseda Research Institute for Science and Engineering, Waseda University, Tokyo, Japan

[†]JST, PRESTO, 4-1-8 Honcho, Kawaguchi, Saitama, Japan

[‡]Institute of Information and Communication Engineering, Zhejiang University, Hangzhou, China

[§]Department of Computer Science and Communication Engineering, Waseda University, Tokyo, Japan

Abstract—Intra prediction is an essential component in the image coding. This paper gives an intra prediction framework completely based on neural network modes (NM). Each NM can be regarded as a regression from the neighboring reference blocks to the current coding block. (1) For variable block size, we utilize different network structures. For small blocks 4×4 and 8×8 , fully connected networks are used, while for large blocks 16×16 and 32×32 , convolutional neural networks are exploited. (2) For each prediction mode, we develop a specific pre-trained network to boost the regression accuracy. When integrating into HEVC test model, we can save 3.55%, 3.03% and 3.27% BD-rate for Y, U, V components compared with the anchor. As far as we know, this is the first work to explore a fully NM based framework for intra prediction, and we reach a better coding gain with a lower complexity compared with the previous work.

Index Terms—Intra prediction, image compression, deep learning, fully connected layer, convolutional neural network

I. INTRODUCTION

Intra prediction is used to remove the spatial redundancy in the image/video coding. For each block, the predicted pixels are the linearly interpolated results based on the neighboring pixels. To enhance the prediction accuracy, more prediction modes and block sizes have been developed in the past standards. For the modes, the amount has been extended from 9 in H.264 [1] to 35 in HEVC [2]. For the block size, the largest block has been enlarged from 16×16 in H.264 to 64×64 in HEVC. Though providing more prediction modes and blocks can improve the coding gain, there is a limitation especially for the blocks without explicit directional texture.

So far, all the standards utilized a single reference line composed of upper and left pixels. To further extract the correlation between adjacent pixels, multiple reference lines are used. As reported in [3], using more reference pixels could reach up to 4.3% coding gain. In addition, to obtain the non-linearity and a high-level relationship between reference blocks and current block, deep learning has been used for intra prediction in several literature. Li *et al.* [4] exploited fully connected (FC) networks for various block sizes from 4×4

to 32×32 . Dumas *et al.* [5] stated that using convolutional neural network (CNN) performs better than FC for blocks larger than 8×8 . Hu *et al.* [6] presented a new structure based on recurrent neural network (RNN). Sun *et al.* [7] studied different combination schemes of traditional modes (TM) and neural network modes (NM) for the fixed block 8×8 . Zhu *et al.* [8] regarded intra prediction as an inpainting problem and provided 35 NMs for 64×64 .

Though the previous works have achieved significant coding gain, there are still some remaining problems. First is that TM still remains in the coding framework. In [4], [5] and [6], one or two NMs were provided. In [7], at most seven NMs were exploited. As a result, the hybrid mode handling of TM and NM might lead to biased selection to one of them. Second is that the complexity is extremely high for some networks such as [8] which is more than 5000x decoding complexity.

In this paper, different from the hybrid mode handling in previous works, we explore a fully NM based intra coding framework. The contributions are listed in the follows.

- For all the 35 modes of variable block sizes from 4×4 to 32×32 , we propose a corresponding network model.
- For the small blocks 4×4 and 8×8 , FC networks are presented. For the large blocks 16×16 and 32×32 , CNN are used.
- We select the optimized model by exploring the coding gain and complexity, and analyze the probability of the best NM for the mode signaling.

II. TRADITIONAL AND LEARNED INTRA PREDICTION

A. HEVC Intra Prediction

HEVC provides 35 TMs, which can be categorized to non-directional and directional. Non-directional TMs are composed of Planar and DC. For the 33 directional TMs, the prediction is performed according to the predictive angles. About the mode signaling of 35 TMs, first, one *bin* is consumed to represent whether the mode is among the most probable mode (MPM) set or not. If so, one or two *bins* are used to indicate the MPM index. Otherwise, five *bins* are cost to represent the mode among the remaining 32 TMs as shown in Table I.

This work was supported in part by JST, PRESTO Grant Number JP-MJPR19M5, Japan.

TABLE I: Mode signaling for 35 TMs

Mode	MPM ₀	MPM ₁	MPM ₂	Non-MPM
Codeword	10	110	111	0{5bins}

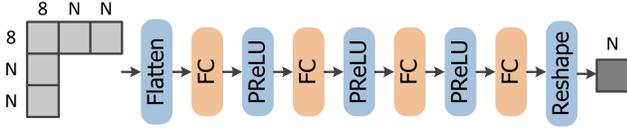


Fig. 1: Network for block 4×4 and 8×8. N is the block size.

B. Learned Intra Prediction with Appending and Substitution Scheme

When introducing NMs, there are two schemes to integrate NMs with TMs as described in [7]. One is appending scheme that is to append NMs to all the 35 TMs. In this case, additional flags are required to signal the new modes. The other is substituting TMs by NMs. In this case, since the overall number of modes is the same as origin, the signaling scheme follows Table I. As reported in [5] and [7], the coding gain by appending scheme is obviously larger than the substitution scheme. However, the substitution scheme is only limited to at most three modes in [7]. In this work, we extend the number from 3 to 35 to create a fully NM-based framework.

III. PROPOSED FULLY NEURAL NETWORK MODE BASED INTRA CODING FRAMEWORK

A. Network Structure Analysis

As described in [5], FC and CNN are suitable for smaller and larger blocks respectively. Motivated by [5], we also use FC network for 4×4 and 8×8 as shown in Fig. 1. First, the neighboring references blocks are flattened to one-dimension vector with $(4 \times N + 8) \times 8$ nodes. By passing through four FC layers, we reshape the one-dimension vector to two-dimension $N \times N$ block. The number of nodes for each layer is determined according to the analysis of the coding gain and complexity. The coding gain is evaluated by PSNR and the complexity is measured by FLOPs. The results are shown in Table II. We first train a baseline heavy model with 512 nodes, and then reduce the number of nodes by half. When reducing the number of nodes to 256 and 128, the coding loss is small. However, when further reducing the dimension to 64, there is an obvious coding loss that is 0.21dB (25.03dB-24.82dB) for 4×4 and 0.34dB (27.08dB-26.74dB) for 8×8. Therefore, we select the node as 128.

For larger blocks, we utilize a CNN network as shown in Fig. 2. To keep the spatial information, the above three blocks and the left two blocks are sent to two separate convolutional paths. The convolutional path is composed of several convolutional layers as shown in Table III. For each path, we conduct the down-sampling to obtain the latent information, and then flatten to one-dimensional vector. Two vectors are concatenated and then pass a FC layer. The number of outputs nodes of the FC layer is 1/5 of the input nodes. Finally, we reshape to two-dimension and use deconvolutional layers to up-sample to the original block size $N \times N$. The

TABLE II: Select the number of node/filter based on the trade-off between coding gain (PSNR) and complexity (FLOPs)

TU	Node/Filter	512	256	128	64
	PSNR (dB)	25.09	25.06	25.03	24.82
	FLOPs (K)	1270	373	121	44
TU 8×8	Node/Filter	512	256	128	64
	PSNR (dB)	27.26	27.20	27.08	26.74
	FLOPs (K)	1450	464	167	67
TU 16×16	Node/Filter	64	32	16	8
	PSNR (dB)	28.24	28.25	28.13	27.93
	FLOPs (M)	97.5	24.5	6.4	1.7
TU 32×32	Node/Filter	64	32	16	8
	PSNR (dB)	29.44	29.37	29.41	29.29
	FLOPs (M)	547.2	138.4	35.4	9.2

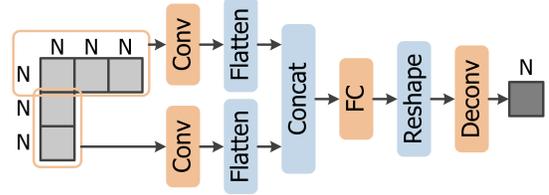


Fig. 2: Network for block 16×16 and 32×32. N is the block size.

selections of filters and strides are shown in Table III. We use four and five convolutional layers for 16×16 and 32×32, respectively. Different from [5], we utilize PReLU as the activation function. The number of filters F is selected as 16 for 16×16 and 32×32 as a trade-off between coding gain and complexity as shown in Table II.

Though [5] has concluded that using convolutional layers could help the prediction of larger block such as 16×16, the conclusion was drawn when utilizing one NM. To ensure that this conclusion also works for all the 35 NMs, we adopt a heavy FC network with 1024 nodes, and the PSNR comparison between FC and CNN network for 16×16 is shown in Fig. 3. We can see that even using a powerful FC model with large nodes, the average performance is still worse than CNN model in most scenarios. In addition, CNN model is better than the original TM in almost all the cases.

B. Coding Framework with Fully Neural Network Modes

There are overall 35 NMs, we select the best NM by the following steps. First, several candidate modes are selected by sum of absolute transformed differences (SATD) cost. Eight candidates are picked up for block 4×4 and 8×8, while three candidates are chosen for the other blocks. In addition to the candidate modes selected by SATD, MPMs are also appended in the candidate mode list. After creating the candidate mode list, the best NM is determined by comparing the R-D cost.

The mode signaling for the 35 TM in Table I was designed based on probability analysis. Modes with higher probability to be the best mode will be allocated fewer *bins*. Given that we have two following equations

$$P(M = BM | M \in MPM) > P(M = BM | M \in Non-MPM) \quad (1)$$

$$P(MPM_0 = BM) > P(M = BM | M \in \{MPM_1, MPM_2\}) \quad (2)$$

where BM is the best mode, thus MPMs are allocated fewer *bins* compared with Non-MPMs. Among MPMs, MPM₀ is allocated one fewer *bin* than the other two MPMs.

TABLE III: Convolutional layer structures for 16×16 and 32×32

Layer number	16×16			32×32		
	Kernel size	Filter number	Stride	Kernel size	Filter number	Stride
1	5×5	F	2	5×5	F	2
2	3×3	F	1	5×5	2F	2
3	5×5	2F	2	5×5	4F	2
4	3×3	2F	1	5×5	8F	2
5	-	-	-	3×3	8F	1

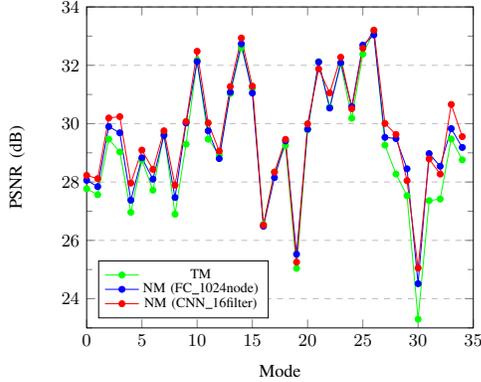


Fig. 3: PSNR comparison of FC and CNN network for 35 modes of 16×16 .

However, Eq. 1 and Eq. 2 are for TM. To ensure that this mode signaling is applicable to 35 NM, we analyze the probability for the sequence *RaceHorses* at QP32, and the results are shown in Table IV. We can see that MPM still owns larger probability than non-MPM in our proposal, which is 59.9% against 40.1%. Among MPMs, MPM_0 has 29.2% to be the best mode which is higher than MPM_1 and MPM_2 . We also evaluate the probability of TMs being the best mode in the original case. We can see that there is no obvious bias between the probability distribution of our proposal and origin.

About the composition of the MPM, when MPM_0 and MPM_1 are not same, we set MPM_2 as one of Planar (mode 0), DC (mode 1) and Vertical (mode 26) considering that these three modes have the highest probability to be the best mode. Therefore, we also evaluate the probability of all the 35 NMs to be the best mode in Fig. 4. We can see that mode 0, 1 and 26 still own the highest probability in our proposals.

IV. TRAINING PROCESS

The training is a regression problem from reference blocks R to original block Y with the network parameter θ . The loss function is composed of MSE and a regularization term as given in Eq. 3

$$J(\theta) = \frac{1}{M} \sum_{m=0}^{M-1} \|F(R^m, \theta) - Y^m\|_2 + \lambda \|\theta_w\|_2^2 \quad (3)$$

where λ is set as 0.0005 and the batch size M is 16. We used the New York city library [9] as the training set. We encode each image with four QPs (22, 27, 32, 37), and the best block is used for the training set $\Gamma = \bigcup_0^{34} \Gamma_i$ where i is the mode number and Γ_i is composed of the blocks which select i as the best mode. First, we train one baseline model based on all the training set Γ . After that, we fine tune the

TABLE IV: Probability to be the best mode for MPMs and Non-MPMs in the case of using TM and NM

	MPM_0	MPM_1	MPM_2	$\bigcup \{32 \text{ Non-MPMs}\}$
Origin (35 TMs)	29.5	18.7	14.9	36.9
Proposal (35 NMs)	29.2	16.7	14.0	40.1

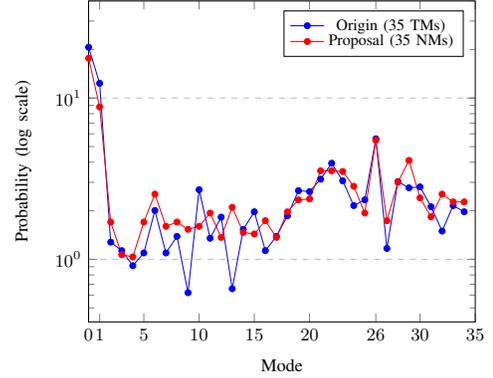


Fig. 4: Probability of 35 NMs and TMs to become the best mode.

baseline model for each specific mode i from 0 to 34 based on the corresponding training set Γ_i . The training procedures are shown in *Algorithm 1*. ADAM [10] is used as the optimizer and the learning rate is set as 0.0001 and 0.0004 for FC and CNN network respectively.

Algorithm 1 Proposed Training Method for 35 NMs

- 1: **for** training iterations $I_1 = \frac{\text{card}(\Gamma)}{M}$ **do**
- 2: Update θ with ADAM optimizer
- 3: **end for**
- 4: Obtain baseline model with θ
- 5: **for** each mode i **do**
- 6: **for** fine tuning iterations $I_2 = \frac{\text{card}(\Gamma_i)}{M}$ **do**
- 7: Update θ_i with ADAM optimizer
- 8: **end for**
- 9: **return** θ_i
- 10: **end for**

V. EXPERIMENTAL RESULTS

A. Coding Gain Analysis

We integrate the proposal into HM 16.9 [11], and test the first frame of the sequences in Table V. We strictly follow the coding configuration of "all-intra main" given by HEVC common test condition (CTC) [12]. Four QPs (22, 27, 32, 37) are tested, and the coding efficiency comparison with the anchor (original HM16.9) is measured by BD-rate [13]. It should be noted that there is no overlap between training set [9] and test set [12].

The results of BD-rate are shown in Table V. The results show that we can save BD-rates for all the test sequences. On average, 3.55%, 3.03% and 3.27% Y, U, V BD-rate can be saved compared with the anchor. Compared with [4], we can achieve large BD-rate reduction for all the three channels. We also compare the class-level results to the three previous works with the same-level complexity in Table VI. When using the proposed model, we can also achieve best coding gain at Class B and E among all the works.

TABLE V: Coding gain comparison with previous work

Class	Sequence	TIP [4]			Proposal		
		Y	U	V	Y	U	V
A	Tango	-7.4	-0.8	-4.3	-7.08	-7.27	-6.74
	Drums100	-3.8	-1.6	-1.6	-2.94	-3.25	-3.15
	CampfireParty	-3.0	-3.3	-3.0	-1.37	-1.99	-1.51
	ToddlerFountain	-3.4	2.8	-1.5	-2.75	0.14	-1.71
	CatRobot	-4.2	-2.4	-2.6	-4.13	-3.23	-3.58
	TrafficFlow	-4.2	-1.3	-1.3	-4.62	-2.89	-2.41
	DaylightRoad	-4.5	0.1	-1.8	-4.85	-3.49	-4.15
	Rollercoaster	-5.8	-3.7	-2.7	-5.11	-4.48	-4.45
Average of Class A		-4.5	-1.3	-2.4	-4.11	-3.31	-3.46
B	Kimono	-3.1	-2.1	-1.5	-2.58	-3.25	-2.91
	ParkScene	-3.6	-2.2	-2.4	-2.58	-2.33	-2.23
	Cactus	-3.2	-1.8	-1.5	-4.27	-1.82	-4.10
	BQTerrace	-2.1	-1.3	-0.5	-4.72	-2.41	-2.59
	BasketballDrive	-3.6	-2.9	-2.7	-2.48	-1.53	-2.77
Average of Class B		-3.1	-2.1	-1.7	-3.33	-2.27	-2.92
C	BasketballDrill	-1.5	-3.3	-2.2	-2.70	-4.75	-4.28
	BQMall	-2.2	-1.9	-1.0	-2.03	-2.53	-3.14
	PartyScene	-1.6	-1.2	-0.1	-1.81	-1.62	-1.20
	RaceHorsesC	-3.2	-1.9	-2.8	-3.01	-1.98	-2.80
Average of Class C		-2.1	-2.1	-1.5	-2.39	-2.72	-2.85
D	BasketballPass	-1.2	-0.3	1.1	-2.32	-2.01	-1.31
	BQSquare	-0.9	-0.1	-2.8	-1.77	0.24	-2.34
	BlowingBubbles	-1.9	-2.8	-3.5	-1.76	-4.23	-3.36
	RaceHorses	-3.2	-2.6	-2.8	-3.52	-4.08	-4.38
Average of Class D		-1.8	-1.5	-2.0	-2.34	-2.52	-2.85
E	FourPeople	-4.4	-4.5	-3.1	-5.81	-5.05	-4.81
	Johnny	-5.3	-3.1	-3.5	-5.33	-4.06	-3.75
	KristenAndSara	-3.9	-3.0	-3.1	-5.55	-4.91	-4.70
Average of Class E		-4.5	-3.5	-3.2	-5.56	-4.67	-4.42
Average of All Sequences		-3.4	-1.9	-2.1	-3.55	-3.03	-3.27

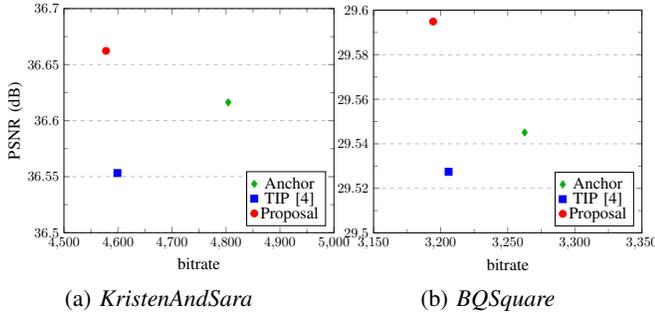


Fig. 5: R-D comparison at low bitrates (QP37). [4] achieved smaller bitrates than the anchor at the cost of worse PSNR, while we can achieve both smaller bitrates and better PSNR.

To clarify the relationship of bitrate and PSNR when using the proposal, we give the R-D results of QP37 of the sequence *KristenAndSara* and *BQSquare* in Fig. 5. We can see that we can save bitrates when achieving better PSNR compared with the anchor. The method [4] can also reduce the bitrate. However, there is some PSNR loss. It is because [4] appended NMs. In the case of NM being the best mode, the number of bit consumption for the mode signaling can be significantly reduced while the coding quality is also degraded.

B. Coding Complexity Analysis

In addition to the coding gain, we also evaluate the coding time in Table VII. The evaluation is executed on Intel Core i7-7820X CPU@3.60GHz with 32GB memory. Same as previous works, we measure the time under the CPU platform. When using the proposed model, 36x and 174x encoding and decoding complexity is cost. Compared with [4], 60% encoding and

TABLE VI: Class-level Y-BD-rate comparison with previous works

	TIP [4]	TIP [5]	TMM [6]	Proposed
Class A	-4.5	N/A	N/A	-4.11
Class B	-3.1	-3.24	-2.39	-3.33
Class C	-2.1	-3.09	-2.31	-2.39
Class D	-1.8	-2.81	-2.54	-2.34
Class E	-4.5	N/A	-3.68	-5.56

TABLE VII: Coding complexity comparison with previous works

	TIP [4]	TIP [5]	TMM [6]	Proposed
Enc.	91x (-60%)	51x (-29%)	N/A	36x
Dec.	230x (-24%)	191x (-9%)	207x (-16%)	174x

24% decoding complexity can be reduced. Compared with [5], 29% encoding and 9% decoding complexity can be decreased. Compared with [6], 16% decoding complexity can be reduced. The relatively low complexity mainly comes from our network with few nodes or filters.

VI. CONCLUSIONS

This paper proposes a fully NM based intra coding. First, we propose the network for all the 35 modes of variable block sizes from 4×4 to 32×32 . Second, we propose a coding framework with NM based on the best mode probability analysis. The experimental results show that we can outperform the previous works in terms of coding gain and complexity. For the future work, we will extend the method to the next-generation video coding standard VVC.

REFERENCES

- [1] Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra, "Overview of the h. 264/avc video coding standard," *IEEE Transactions on circuits and systems for video technology*, vol. 13, no. 7, pp. 560–576, 2003.
- [2] Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE Transactions on circuits and systems for video technology*, vol. 22, no. 12, pp. 1649–1668, 2012.
- [3] Jiahao Li, Bin Li, Jizheng Xu, and Ruiqin Xiong, "Efficient multiple-line-based intra prediction for hevc," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 4, pp. 947–957, 2016.
- [4] Jiahao Li, Bin Li, Jizheng Xu, Ruiqin Xiong, and Wen Gao, "Fully connected network-based intra prediction for image coding," *IEEE Transactions on Image Processing*, vol. 27, no. 7, pp. 3236–3247, 2018.
- [5] Thierry Dumas, Aline Roumy, and Christine Guillemot, "Context-adaptive neural network-based prediction for image compression," *IEEE Transactions on Image Processing*, vol. 29, pp. 679–693, 2019.
- [6] Yueyu Hu, Wenhan Yang, Mading Li, and Jiaying Liu, "Progressive spatial recurrent neural network for intra prediction," *IEEE Transactions on Multimedia*, vol. 21, no. 12, pp. 3024–3037, 2019.
- [7] Heming Sun, Zhengxue Cheng, Masaru Takeuchi, and Jiro Katto, "Enhanced intra prediction for video coding by using multiple neural networks," *IEEE Transactions on Multimedia*, 2020.
- [8] Linwei Zhu, Sam Kwong, Yun Zhang, Shiqi Wang, and Xu Wang, "Generative adversarial network-based intra prediction for video coding," *IEEE Transactions on Multimedia*, vol. 22, no. 1, pp. 45–58, 2019.
- [9] Kyle Wilson and Noah Snavely, "Robust global translations with 1dsfm," in *European Conference on Computer Vision*. Springer, 2014, pp. 61–75.
- [10] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
- [11] "HEVC Test Model (HM-16.9)," https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-16.9/, Accessed on June 30, 2020.
- [12] K. Suehring K. Sharman, *Common Test Conditions*, document JCTVC-Z1100, 26th Meeting, Geneva, Switzerland, Jan. 2017.
- [13] Gisle Bjontegaard, "Calculation of average psnr differences between rd-curves," *VCEG-M33*, 2001.