OPTIMIZED CONVOLUTIONAL NEURAL NETWORKS FOR VIDEO INTRA PREDICTION

Maria Meyer, Jonathan Wiesner, Christian Rohlfing

Institut für Nachrichtentechnik, RWTH Aachen University
Melatener Str. 23, 52074 Aachen, Germany
meyer@ient.rwth-aachen.de

ABSTRACT

Based on a previously published neural network-based video intra prediction approach, this paper proposes and evaluates several extensions of both the training process as well as the network architectures. In particular, the influence of coding artifacts in the training samples as well as the effect of using different approximations of the residual coding costs as loss functions are investigated. In addition, the architecture is optimized and extended by final deconvolutional layers. Combined with the use of network pruning, it was not only possible to increase the achieved compression gain in comparison to the previous work, but also to decrease the needed number of floating point operations per pixel by more than 72% at the same time.

Index Terms— video coding, intra prediction, neural networks, architecture optimization, pruning

1. INTRODUCTION

As traditional intra prediction methods can reduce redundancy in smooth areas or at straight edges but are usually unable to capture curves or complex textures, several neural network-based intra prediction approaches were presented within the last years.

The first approach for a fully-connected intra prediction network was published in [1] and [2]. It already proved that NNs can lead to notable bit-rate reductions. Since then, numerous authors tried more sophisticated but also computationally more costly architectures. While [3] and [4] presented CNN-based architectures, an RNN-based approach was published in [5] and further improved in [6] and [7]. In [8] the complete inpainting network from [9] was integrated into a video codec with very promising compression results, but also with large complexity problems. Recently an autoencoder-based approach, transmitting quantized feature space values, was described in [10], but not yet tested in a complete video codec.

A lower compression gain at very low complexity is achieved by the approach from [11]. Here, a network returning multiple predictions simultaneously was reduced to a single layer. Interestingly, not only the branches for these different results but also the networks for the different block sizes were trained jointly, as described in [12]. The result was of low enough complexity to be adopted into the VTM test model [13]. The problem remains however, that it is difficult to draw conclusions from the comparison of these papers as the used training set is a very significant factor for the achieved compression results and almost all of them use a different one.

A first comparison of a fully-connected and two convolutional architectures trained on the same data as well as the comparison of two different loss functions were reported in [14]. It is the goal of this paper to extend the analysis started there in four aspects: First, the loss function comparison is extended by two new candidate functions that approximate the resulting coding cost of a prediction more closely than the previously tested ones. Second, the effects of using training material with coding artifacts are investigated. Third, changes in the architecture's hyper parameters are evaluated in order to compare not just any instances of each architecture type, but either those with the best achievable results for each category or those of comparable computational complexity. Fourth, the investigated architecture types are extended to those with final deconvolutional layers and to pruned networks.

The results of the first two of these evaluations are described in section 3 followed by the results of the architecture related analysis in section 4. However, first it is necessary to briefly summarize the results from [14] in section 2.

2. PREVIOUS WORK

There are several aspects of using neural networks for intra prediction that were analyzed in [14]. Among the main novelties was the use of convolutional networks with final fully-connected layers as shown in figure 1, the training of separate networks for chroma prediction that integrates cross-component information and the integration of the network-based mode by extending the most probable mode list used for intra mode signaling in HEVC.

In detail, three architectures for the prediction of luma blocks were compared that had two, one or zero convolutional layers at their beginning followed by four fully-connected layers. As the number of used reference lines was restricted to four and no padding was used, the options for the used kernel sizes were limited. Every layer but the last used a leaky ReLU as its activation function and optionally available parts of the reference area were randomly masked during training, which was performed on patches from 104 raw sequences using an Adam optimizer. The comparison showed, that while on average the version with two convolutional layers gave the highest BD-rate gains, the purely fully-connected version outperformed the others on noisy high-resolution content.

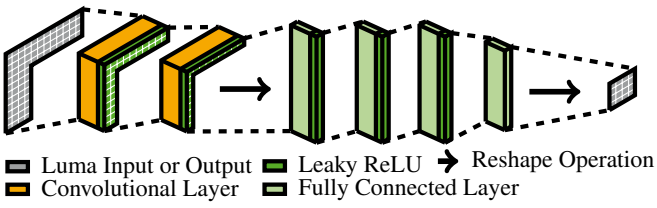


Fig. 1: Example architecture from [14] for a 4x4 luma block with two initial convolutional layers of kernel size $\mathbf{k}_c = (3, 2)$ as well as four fully-connected ones. Each layer except the final one is followed by a leaky ReLU.

Simulations were performed with computing resources granted by RWTH Aachen University under project nova0034.

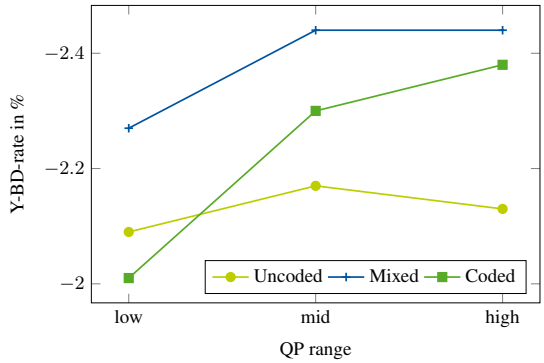


Fig. 2: Y-BD-rate improvement for prediction networks trained with uncoded, coded and mixed reference samples over QP-ranges low $= (17, 22, 27, 32)$, mid $= (22, 27, 32, 37)$ and high $= (27, 32, 37, 42)$.

For the joint prediction of the two chroma channels, an architecture that first separately extracted features from the available chroma and luma reference using two convolutional layers each before concatenating the results and feeding them into four fully-connected layers was compared with a version without the additional luma information as well as one without a network-based prediction of the chroma channels. Utilizing the additionally available information clearly outperformed the other versions.

Furthermore, it was compared if the sum of absolute transformed differences (SATD), that had already been shown to outperform the MSE as a loss function in [5], also outperformed the L1-norm, which was the case. Finally, a novel way to signal the network-based intra mode in HEVC was proposed that is based on extending the most probable mode list. Two versions of where to place the additional mode on the list were compared, showing that the best result is achieved by placing the network-based mode directly behind the modes from the adjacent blocks. If not explicitly stated otherwise, all training parameters, preprocessing options and signaling methods used in the following sections are the same as in [14].

3. TRAINING IMPROVEMENTS

Performance improvements gained by changes in the training process are always especially beneficial as they come without any additional complexity at inference time. Thus, both improvements in the used training data as well as in the loss function were investigated.

3.1. Training Material

As with most learning-based approaches, the availability and quality of the training data has a huge impact on the results of our prediction approach. In [14], all samples used for training the prediction models were generated from raw video sequences. As in the later application only a coded version of the reference area is available for prediction, this might affect the performance severely since coding artefacts might be propagated. Thus, in order to improve the prediction quality, the networks were retrained with previously coded reference samples. Taking a quarter of the training samples from videos coded with QP 22, 27, 32 and 37 respectively, already improved the average Y-BD-rate by 0.13% for these same QPs.

Interestingly, it was found that it works even better to mix both coded and uncoded samples in the training set. When mixing half of the uncoded and half of the coded training material, the total Y-DB-rate improvement for these same QPs reached 0.27%. In order to explain this behavior, the performance of the three sets of prediction networks was also tested for QPs 17 and 42. As can be seen in figure 2, while the network trained with coded features outperforms the version trained on uncoded samples for the middle and high QP ranges, the latter gives a better result for low QPs. Combining the training material outperforms both other tests for all three ranges. For the chroma channels the effect of using the coded training material is even larger. Here, the version with only coded training samples already outperforms the uncoded version for all QP ranges. The mixed version further improves this result to a total of 1.37% and 0.83% BD-rate gain for U and V respectively.

Loss Function	L_{SATD}	L_{DCT}	L_{LOG}
Class B	-2.62 %	-2.62 %	-2.54 %
Class C	-2.14 %	-2.14 %	-2.10 %
Class D	-2.09 %	-2.06 %	-2.01 %
All	-2.31 %	-2.30 %	-2.24 %

Table 1: BD-rate change for the Y channel when using network-based intra prediction with different loss functions compared to HEVC.

3.2. Loss Function

Another problem when training a prediction network is the choice of the loss function. The problem here is that the exact coding costs that a prediction residual $\boldsymbol{R}$ of a block of size $b \mathbf{x} b$ would cause in the later application cannot be calculated exactly during training due to the many dependencies on neighboring blocks and the exact encoder settings. Also, the quantization would have to be approximated as it otherwise cannot be derived. It was already shown previously in [5] and [14], that the SATD (1) outperforms simpler approximations such as MSE and L1-Norm as a loss function. Nonetheless, it is still a very coarse estimation of the actual costs. Thus, further refinements were tested. One of these, was to exchange the Hadamard transform $\boldsymbol{T}_{\mathrm{HAD}}$ used in the SATD-function by the DCT and DST transforms $\boldsymbol{T}_{\mathrm{DCT}}$ used in HEVC [15] for the respective block size as shown in (2).

$$L_{\mathrm{SATD}} = b^{-2} \sum |\boldsymbol{T}_{\mathrm{HAD}} \cdot \boldsymbol{R}| \tag{1}$$

$$L_{\mathrm{DCT}} = b^{-2} \sum |\boldsymbol{T}_{\mathrm{DCT}} \cdot \boldsymbol{R}| \tag{2}$$

$$L_{\mathrm{LOG}} = b^{-2} \sum f(\boldsymbol{T}_{\mathrm{DCT}} \cdot \boldsymbol{R}) \tag{3}$$

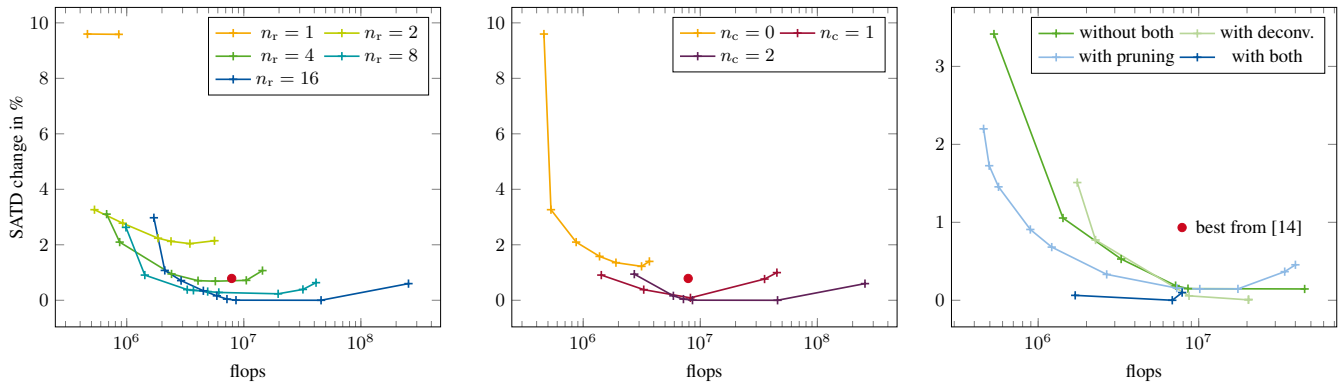
$$\text{with} \quad f(x) = |x| + \alpha(1 + \exp(-\beta|x| - \gamma))^{-1} \tag{4}$$

Another interesting approximation for the block wise loss (3) was suggested in [12]. Here, the DCT-coefficients are weighted with a logistic function $f(x)$ (4), that can estimate the needed number of bits for a residual before entropy coding for a certain QP with the right choice of α, β and γ.

It can be seen in table 1, that using the DCT-transform did not improve the prediction results in terms of Y-BD rate. Only for the chroma channels improvements of 0.15% and 0.20% were achieved compared to using the normal SATD. The coding cost estimation adapted from [12] actually decreased the coding gain for all channels. Note, that the same networks were used for all QPs with α, β and γ adapted to QP32, which might cause the decrease. However, training different networks for each QP would not only be very time consuming during training, but also lead to a high memory demand on every application device.

4. ARCHITECTURE OPTIMIZATION

Architecture optimization can both improve the prediction performance, but also reduce the computational complexity. Unfortunately, the number of options and parameters in this optimization



(a) Lower bounds on the SATD for different numbers of reference lines among all tests without deconvolution or pruning

(b) Lower bounds on the SATD for different numbers of convolutional layers among all tests without deconvolution or pruning

(c) Lower bounds on the achieved SATD for test with or without deconvolutional layers and / or pruning for $n_r = 16$

Fig. 3: Lower bounds on the relative change in the achieved SATD with regard to the best result within the respectively shown set of results.

is quite high. Thus, a pretest was performed, that was limited to optimizing the luma prediction network for a block size of $b = 16$ and evaluated by the normalized SATD achieved on the validation set. A table with the exact results and settings of all 234 tests as well as some further visualizations can be found on the website accompanying this paper[1]. The most interesting findings of this pretest are reported in sections 4.1–4.3, before the results the best architectures achieved in the HM are shown in section 4.4.

4.1. Reference Lines and Convolutional Layers

In a first step, different numbers of reference lines n_r were compared as shown in figure 3a. As expected, the achievable performance increases with the size of the used reference area. Thus, the overall best result was achieved with $n_r = b$. It can also be seen that the prediction network can benefit considerably from including more than the single reference line used by the HEVC intra modes. However, as the gain from adding additional lines decreases and the complexity needed to utilize these lines increases, using $n_r < b$ can be useful, when low complexity needs to be prioritized.

A similar observation can be made for the number of used convolutional layers n_c shown in figure 3b. While using two initial convolutional layers gives the overall best result, architectures with only one convolutional layer give better results at less than $5 \cdot 10^6$ flops. Also, while in [14] it was only possible to say that more complex convolutional architectures outperform purely fully-connected ones of less complexity, it can now be seen that even at the same complexity convolutional architectures outperform fully-connected ones over the complete considered range.

4.2. Kernels and Depth Choice

For both architectures, either with one or with two convolutional layers, different kernel sizes $\mathbf{k}_c$ were tried. It became obvious that those that do not satisfy $\sum_{i=1}^{n_c} \mathbf{k}_{c,i} = n_r + n_c - 1$ lead to significantly more complexity than other combinations with similar performance. Thus, for architectures with $n_c = 1$ the optimal kernel size is $\mathbf{k}_c = n_r$. For architectures with $n_c = 2$, this results in $k_{c,1} + k_{c,2} = n_r + 1$. In this case it additionally seems to be beneficial to choose the first kernel $k_{c,1}$ significantly larger than the second one $k_{c,2}$.

Another observation is, that the achieved loss is very sensitive to the minimum $\min(\mathbf{s}_f)$ of the chosen dense layer sizes $\mathbf{s}_f$, independently of which layer is the smallest. As the dense layer size also has a relatively small influence on the complexity, setting all elements of $\mathbf{s}_f$ to 512, which gave the optimal result, is the best choice unless operating at very low complexity.

When lowering the depth $\mathbf{s}_c$ of the convolutional layers, the measured performance drop is quite consistent across the different combinations. The lower the sum $\sum_i \mathbf{s}_{c,i}$ becomes, the more and the faster the performance decreases. Tests with different depth values $s_{c,i}$ for the different layers i did not show a clear preference for choosing one of the layers larger than the other.

4.3. Deconvolution and Pruning

In addition to optimizing the architecture types used in [14], two new architecture features were tried. The first one of these is using deconvolutions as final layers of networks for larger block sizes, as first proposed in [3]. As these deconvolutional layers can conceptually be seen as an upsampling operation, a fixed stride of two was used. An advantage of these layers is their low complexity, as there are many inherent zero multiplications. In the scope of this work, simulations with either one or two deconvolutional layers replacing the final fully-connected one, were performed. While only using one deconvolutional layer did not noticeably improve the performance at any complexity, using two deconvolutional layers leads to an SATD improvement of 0.14% over the best architecture without deconvolutional layers at 45% of the complexity. The chosen kernel sizes $\mathbf{k}_d$ for these layers do not seem to have much effect on the result. While the depth of the final deconvolutional layer needs to be fixed to the number of output channels, increasing the depth of the first deconvolutional layer slightly but steadily improves the performance.

Another approach to efficiently reduce the number of needed floating point operations is pruning the network. As stated in [19], pruning a larger architecture can even improve the results compared to smaller, unpruned architectures of same overall complexity. Thus, pruning at numerous target sparsities was applied to a set of the so far best parameter combinations. As can be seen in figure 3c, this significantly improves the lower bound on the results in the range below 7 million flops.

Pruning by a relatively small amount was indeed able to improve the SATD results further. The overall best result, combining both deconvolutional layers and pruning of 25%, needs 1 million less floating point operations per prediction block while increasing the performance by 0.92% compared to the best architecture used in [14]. When pruning with higher target sparsities, an even more significant complexity reduction can be achieved at a very low per-

[1] http://www.ient.rwth-aachen.de/cms/icip2020/

Version	**[14]**			**with coded training material**			**optimized CNN with $n_r = b$**			**with pruning of 87.5%**			**with pruning and deconvolution**		
Channel	**Y**	**U**	**V**	**Y**	**U**	**V**	**Y**	**U**	**V**	**Y**	**U**	**V**	**Y**	**U**	**V**
BQTerrace	-1.45	-0.16	0.52	-1.77	**-0.27**	**0.29**	**-1.79**	0.31	0.63	-1.51	0.21	0.79	-1.54	0.31	0.97
Bask.Drive	-2.35	-1.35	-1.62	**-2.79**	**-1.75**	**-2.09**	-2.69	-1.07	-1.45	-2.36	-0.87	-1.06	-2.39	-0.84	-1.18
Cactus	-2.51	-1.90	-2.00	**-2.98**	**-2.08**	**-2.20**	-2.85	-1.85	-2.00	-2.57	-1.57	-1.54	-2.56	-1.56	-1.58
Kimono	-2.50	-2.26	-1.78	**-2.82**	**-2.46**	**-2.78**	-2.75	-2.36	-2.62	-2.53	-1.99	-2.18	-2.57	-1.91	-2.26
ParkScene	-2.50	-1.76	-1.77	-2.71	**-1.95**	**-2.56**	**-2.74**	-1.75	-2.09	-2.56	-1.15	-1.58	-2.59	-1.24	-1.72
AVG Class B	-2.26	-1.49	-1.33	-2.61	-1.70	-1.87	-2.56	-1.34	-1.51	-2.31	-1.07	-1.11	-2.33	-1.05	-1.15
BQMall	-2.48	-1.99	-2.02	**-3.00**	**-2.60**	**-2.92**	-2.92	-2.18	-2.53	-2.71	-1.89	-1.92	-2.69	-1.77	-1.91
BasketballDrill	-2.14	-2.45	-2.32	**-2.62**	**-3.18**	**-3.21**	-2.36	-2.69	-2.29	-1.93	-1.55	-1.90	-1.88	-1.81	-1.85
PartyScene	-1.50	-1.05	-1.09	-1.75	**-1.07**	**-1.21**	**-1.79**	-0.94	-1.02	-1.60	-0.73	-0.77	-1.59	-0.67	-0.80
RaceHorses	-1.82	-1.45	-1.35	**-2.19**	**-1.70**	**-1.90**	-2.05	-1.51	-1.53	-1.79	-1.14	-0.98	-1.83	-1.27	-1.17
AVG Class C	-1.99	-1.74	-1.70	**-2.39**	**-2.14**	**-2.31**	-2.28	-1.83	-1.84	-2.01	-1.33	-1.39	-2.00	-1.38	-1.43
BQSquare	-0.95	**-0.38**	**-0.49**	**-1.21**	**-0.38**	-0.26	-1.20	-0.37	-0.27	-1.06	-0.23	-0.40	-1.05	-0.26	-0.19
BasketballPass	-2.29	-2.18	-2.06	**-2.69**	**-2.56**	**-2.58**	-2.64	-2.40	-2.10	-2.30	-2.00	-1.95	-2.33	-1.91	-2.00
BlowingBubbles	-1.53	-1.13	-1.27	**-1.81**	**-1.38**	**-1.34**	-1.79	-1.22	-0.93	-1.62	-0.75	-0.98	-1.61	-0.72	-0.65
RaceHorses	-2.21	-1.81	-1.92	**-2.70**	**-2.55**	**-2.32**	-2.54	-2.20	-1.87	-2.28	-1.60	-1.28	-2.20	-1.59	-1.33
AVG Class D	-1.75	-1.38	-1.44	**-2.10**	**-1.72**	**-1.63**	-2.04	-1.55	-1.29	-1.82	-1.15	-1.15	-1.80	-1.12	-1.04
AVG All Classes	-2.02	-1.53	-1.47	**-2.39**	**-1.84**	**-1.93**	-2.32	-1.56	-1.54	-2.06	-1.18	-1.21	-2.06	-1.17	-1.21
Flops per Pixel in 10^3	182.0			202.2			253.0			**49.8**			50.4		

Table 2: BD-rate [16] change compared to HEVC [17] in %. All tests were conducted according to the common testing conditions [18] with the AllIntra settings. Note that, as in [14], tests were only conducted on the first 100 frames, while here the full sequences were used, there are deviations in the reported gains even with the otherwise exactly same code.

formance reduction. In explicit, by using a pruning rate of 87.5% it was possible to lower the complexity below 22% of the old architecture while still maintaining a performance improvement of 0.86%.

4.4. HM Comparison

In a final step the architectures that performed best in the previous sections, were adapted to different block sizes and channels in order to test them in a full coding environment. The results are shown in table 2. The first column shows the results achieved with the networks from [14]. The second reports the results of a similar architecture but with the partially coded training material. Column three shows the results of the SATD-wise best architecture without deconvolutional layers or pruning, followed by the same architecture pruned at 87.5% in the fourth column. Finally, the last column shows results for an architecture that combines pruning and deconvolution. While there is a clear compression gain due to the coded training material as already discussed in section 3.1, the architecture optimization, pruning and deconvolution all decreased the performance despite their good pretest results, especially on the chroma channels.

Part of this result can be explained when considering the validation SATDs of all finally used networks[2]. While the results for luma blocks of size 32 show similar improvements to those achieved in the pretests on luma blocks of size 16, the deconvolutions seem to decrease the achieved gains for smaller blocks. Furthermore, the pretest findings seem not to be easily transferable to the chroma prediction networks. For these both the performed architecture optimisation as well as the pruning increase the average resulting SATD, which could explain the higher BD-rate loss on the chroma channels.

Another reason for the decreases could be found in the block size distribution. As the applied block size within HEVC is subject to the RD-decision, larger sizes are only chosen for smooth areas while smaller blocks are applied in highly structured regions. This property is so far neither reflected in the choice of the training nor of the validation data. It can thus happen, that the SATD improvement on the validation set is caused by improvements for cases for which the regarded block size is not chosen. In explicit, while there is a clear SATD improvement for all luma block sizes between the version of columns two and three, the usage of the NN-based mode decreased by more than 11% for blocks of sizes 16 and 32.

In contrast to the SATD improvements that could be seen in the pretest, the complexity reduction due to the network pruning is clearly notable in these results as well. Compared to the same architecture without pruning, a complexity decrease of more than 80% is achieved by the pruned version while only losing 0.26% of the Y-BD-rate gains. When comparing these results with those from [14], the average Y-BD-rate was even slightly improved while decreasing the number of needed floating point operations by more than 72%.

5. CONCLUSION

In summary, four improvements to the NN-based intra prediction approach from [14] are reported in this paper. First, it was shown that mixing training material with and without coding artifacts can improve the achieved BD-rate gain by 0.37%, 0.31% and 0.46% for the Y, U and V channel respectively. Second, it proved to be difficult to model the coding costs of a prediction residual more accurately than the SATD which could be explained by the QP-dependence of the quantization. Third, while an architecture optimization and the use of final deconvolutional layers improved the general prediction performance of the networks in terms of SATD, these improvements cannot easily be translated into compression gain due to the RD-optimization of the applied block sizes. Finally, using pruning methods reduces the computational complexity of the prediction networks by more than 72% while still achieving a better BD-rate than the previously reported results for the luma channel.

In future work, a feasible way to consider the RD-optimization of the block size during training, that does not consolidate the block size distribution too much, has to be found. Also, there might still be some additional gains in optimizing the chroma architecture separately, as adapting the findings from the $16\mathbf{x}16$ luma prediction, showed problems for the chroma cases in the validation set SATD.

[2]See www.ient.rwth-aachen.de/cms/icip2020/ for detailed results

6. REFERENCES

[1] J. Li, B. Li, J. Xu, and R. Xiong, "Intra prediction using fully connected network for video coding," in *2017 IEEE International Conference on Image Processing (ICIP)*, Sept. 2017.

[2] J. Li, B. Li, J. Xu, R. Xiong, and W. Gao, "Fully connected network-based intra prediction for image coding," *IEEE Transactions on Image Processing*, vol. 27, no. 7, pp. 3236–3247, July 2018.

[3] T. Dumas, A. Roumy, and C. Guillemot, "Context-adaptive neural network based prediction for image compression," *Computing Research Repository (CoRR)*, 2018.

[4] Y. Wang, X. Fan, S. Liu, D. Zhao, and W. Gao, "Multi-scale convolutional neural network based intra prediction for video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, 2019.

[5] Y. Hu, W. Yang, M. Li, and J. Liu, "Progressive spatial recurrent neural network for intra prediction," *Computing Research Repository (CoRR)*, 2018.

[6] Y. Hu, W. Yang, S. Xia, W. Cheng, and J. Liu, "Enhanced intra prediction with recurrent neural network in video coding," in *2018 Data Compression Conference*. Mar. 2018, IEEE.

[7] Y. Hu, W. Yang, S. Xia, and J. Liu, "Optimized spatial recurrent network for intra prediction in video coding," in *2018 IEEE Visual Communications and Image Processing (VCIP)*. Dec. 2018, IEEE.

[8] L. Zhu, S. Kwong, Y. Zhang, S. Wang, and X. Wang, "Generative adversarial network based intra prediction for video coding," *IEEE Transactions on Multimedia*, pp. 45–58, 2019.

[9] S. Iizuka, E. Simo-Serra, and H. Ishikawa, "Globally and locally consistent image completion," *ACM Transactions on Graphics*, vol. 36, no. 4, pp. 107:1–107:14, July 2017.

[10] F. Brand, J. Seiler, and A. Kaup, "Intra frame prediction for video coding using a conditional autoencoder approach," in *Picture Coding Symposium 2019 (PCS)*, Nov. 2019.

[11] J. Pfaff, P. Helle, D. Maniry, S. Kaltenstadler, W. Samek, H. Schwarz, D. Marpe, and T. Wiegand, "Neural network based intra prediction for video coding," in *Applications of Digital Image Processing XLI*. Sept. 2018, SPIE.

[12] P. Helle, J. Pfaff, M. Schäfer, R. Rischke, H. Schwarz, D. Marpe, and T. Wiegand, "Intra picture prediction for video coding with neural networks," in *2019 Data Compression Conference (DCC)*. Mar. 2019, IEEE.

[13] "VVC test model," https://vcgit.hhi.fraunhofer.de/, 2019.

[14] M. Meyer, J. Wiesner, J. Schneider, and C. Rohlfing, "Convolutional neural networks for video intra prediction using cross-component adaptation," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. May 2019, IEEE.

[15] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. 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, Dec. 2012.

[16] G. Bjontegaard, "Calculation of average PSNR differences between RD-Curves," ITU-T SG16/Q6 VCEG, Austin, USA, VCEG-M33, 2001.

[17] "HEVC test model, 16.9," https://hevc.hhi.fraunhofer.de/, 2016.

[18] K. Suehring and X. Li, "JVET common test conditions and software reference configurations," Doc. JVET-G1010, Joint Video Exploration Team of ITU-T VCEG and ISO/IEC MPEG, July 2017.

[19] M. Zhu and S. Gupta, "To prune, or not to prune: exploring the efficacy of pruning for model compression," *Computing Research Repository (CoRR)*, 2017.