Neural Data-Dependent Transform for Learned Image Compression –Supplementary Material–

Dezhao Wang Wenhan Yang Yueyu Hu Jiaying Liu*
Wangxuan Institute of Computer Technology, Peking University

This supplementary material presents (1) the detailed network structure and hyper-parameters, (2) results on our MS-SSIM-oritended models, (3) more ablation studies, (4) more subjective results and (5) limitation analyses of our proposed method.

Contents

1. Network Details

In this section, we provide the hyper-parameters of our network. The hyper-parameters of the transform model and entropy model are shown in Table. 1, which are almost the same as [9] except for the additional data-dependent convolutional layer.

The hyper-parameters of the proposed Neural-syntax Generator and Weights Generator are shown in Table. 2. In the Syntax Generator, we concatenate multi-scale global pooling results together to send them into the final fully-connected layer. In the Weights Generator, after the last fully-connected layer, we reshape the one-dimension feature map to fit the shape of a convolutional kernel.

N and M are set to 192 and 16 respectively for lower bit-rate models. For higher bit-rate models, we set N and M to 384 and 32 to preserve more information. k is set to 1.

The hyper-parameters of our neural-syntax based post-processing are shown in Table. 3. Our post-processing is based on HAN [12][1] but without upsampler and with dynamically generated weights to produce the final reconstruction. We use two configurations for low and high bit-rate models.

2. Results on MS-SSIM-Oriented Models

Besides Mean Square Error (MSE), we also train models with data-dependent transform using MS-SSIM as the distortion metric. The loss function is shown as follows,

$$L = \lambda(1 - D_S(x, \hat{x})) + R(\hat{z}_s) + R(\hat{z}_c) + R(\hat{z}_h), \tag{1}$$

where $D_S(\cdot, \cdot)$ calculates MS-SSIM over two images. $R(\hat{z}_s)$, $R(\hat{z}_c)$, $R(\hat{z}_h)$ in Eq. (1) represent the bit-rate of the neural-syntax, content stream and hyper-prior, respectively. Here, λ belongs to $\{20, 64, 120, 160\}$. Similar to MSE-oriented models, our high bit-rate models, whose λ ranges among $\{64, 120, 160\}$, double the number of their parameters for stronger model capacity.

*Corresponding author. Our project is available at: https://dezhao-wang.github.io/Neural-Syntax-Website/.

[1]Here we implement our post-processing with reference to https://github.com/wwlCape/HAN.

Table 1. Hyper-parameters of our encoder, decoder, hyper encoder, hyper decoder and prediction model. (De)Conv: $a\times a$ $m\rightarrow n$ sl denotes (De)Convolution (transposed convolution) layer with kernel size a and stride l. Its input channel is m and output channel is n. Conv* means the parameters are dynamically generated.

Encoder	**Decoder**	**Hyper Encoder**	**Hyper Decoder**	**Prediction Model**
Conv: 5×5 3→N s2	DeConv: 5×5 (N-M)→N s2	Conv: 3×3 N→N s1	DeConv: 5×5 N→N s2	Conv: 3×3 (2N-M)→N s1
GDN	IGDN	Leaky ReLU	Leaky ReLU	Leaky ReLU
Conv: 5×5 N→N s2	DeConv: 5×5 N→N s2	Conv: 5×5 N→N s2	DeConv: 5×5 N→N s2	Conv: 3×3 N→N s1
GDN	IGDN	Leaky ReLU	Leaky ReLU	Leaky ReLU
Conv: 5×5 N→N s2	DeConv: 5×5 N→N s2	Conv: 5×5 N→N s2	DeConv: 3×3 N→N s1	Conv: 3×3 N→N s1
GDN	IGDN			Leaky ReLU
Conv: 5×5 N→N s2	DeConv: 5×5 N→M s2			Flatten
	IGDN			FC: 4N→2(N-M)
	Conv*: $k\times k$ M→3 s1			

Table 2. Hyper-parameters of our syntax generator and weights generator.

Syntax Generator	**Weights Generator**
Conv: 3×3 M→2M s2	FC: M→8M
ReLU	Leaky ReLU
Conv: 3×3 2M→4M s2	FC: 8M→16M
ReLU	Leaky ReLU
GlobalAvgPooling	FC: 16M→k*k*M*N
Concat	
FC: 7M→M	

Table 3. Hyper parameters of our neural-syntax based post-processing.

	Low Bit-rate	**High Bit-rate**
Residue Group	4	6
Residue Block	8	12
Feature Map Width	64	
Reduction	32	
Activation	ReLU	

The training strategies and settings are the same as the ones adopted by MSE-driven models. We compare our method with other methods on Kodak [8] and CLIC Profession Validation Dataset [1]. We convert the MS-SSIM values to decibels, *i.e.* $-10\log_{10}(1-d)$ where d refers to the MS-SSIM value, for a clear illustration.

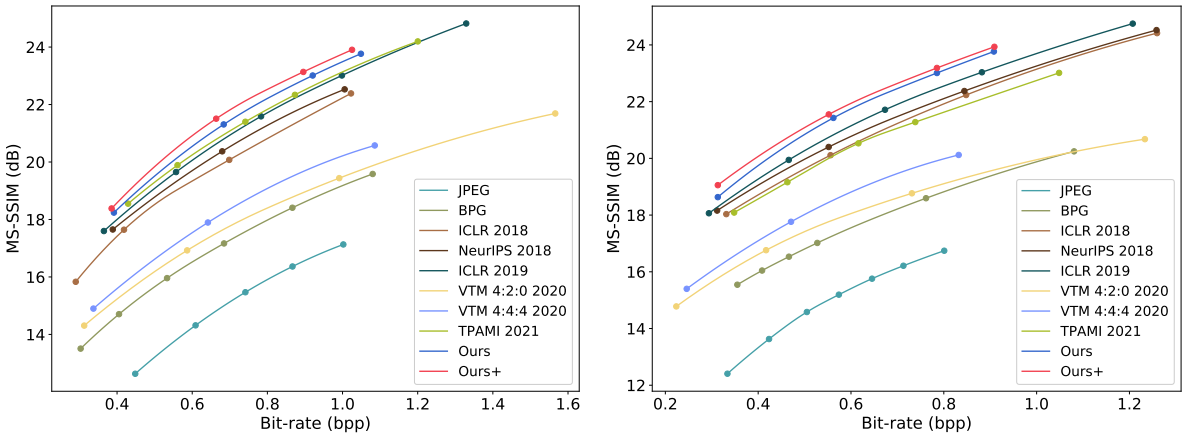
We compare our method to existing end-to-end learned image compression methods optimized for MS-SSIM [2,7,9,11][2]. We also compare our method with hybrid codecs, *i.e.* JPEG [14], BPG [6] and VVC [4]. We use VTM 8.0 [5] with chroma format 4:2:0 and 4:4:4 in the evaluation. We set BPG as the anchor and calculate the BD-rate [3] over it. The results are shown in Table 4. We can find that our method outperforms all compared methods on both Kodak and CLIC. Compared to VTM, we can even save almost 40% bit-rate at the same distortion level on CLIC.

We also show the R-D curves of all compared methods in Fig. 1. As illustrated, our method outperforms both end-to-end learned compression methods and hybrid codecs.

[2]For NeurIPS 2018, we evaluate the released models based on the mean and scale hyper-prior but without the auto-regressive context model.

Table 4. BD-rate results (↓) on Kodak [8] and CLIC [1]. We set BPG [6] as the anchor. The best results are shown in bold and the second best are underlined.

	Kodak	CLIC
NeurIPS 2018 [2]	-47.0%	-52.7%
ICLR 2019 [9]	-49.8%	-55.1%
VTM8 4:2:0 2020 [5]	-10.2%	-10.8%
VTM8 4:4:4 2020 [5]	-19.0%	-25.0%
TPAMI 2021 [7]	-51.4%	-46.8%
Ours	<u>-53.4%</u>	<u>-59.4%</u>
Ours+	**-55.5%**	**-62.2%**



(a) Kodak (b) CLIC

Figure 1. R-D curves on Kodak and CLIC.

3. Ablation Studies

In this section, we provide more ablation studies to verify the designs of our framework. It should be noted that, for all compared settings, online finetuning mechanism and post-processing is turned off.

1) Study on Kernel Size. The kernel size of the neural-syntax-controlled convolutional layer is an important hyper-parameter in the proposed method, as it affects both the bit-rate usage and the modeling capacity. To study the effects of different kernel sizes, we further compare the performance of alternative kernel sizes, *i.e.* 1×1 and 3×3 on Kodak. According to the experimental results in Fig. 2, *3×3 conv* and *Ours* (which uses 1×1 convolutional kernel) have a marginal difference in R-D performance. Since 1×1 kernels are more light-weight in computation, we adopt the 1×1 settings in the proposed framework.

2) Study on Channel Split. We further conduct an experiment on the 'SPLIT' operation in the network. One way is to directly map the whole latent feature map to the neural-syntax stream without an explicit split. We compare this non-split setting to the split one on Kodak. The results are shown in Fig. 2, corresponding to *w/o SPLIT* and *Ours*. As shown, the performance degrades a little without explicit partitioning despite the fact that more information and parameters are used to generate reconstruction images in *w/o SPLIT* setting.

3) Network Depth Where We Inject the Generated Weights. In our proposed framework, we generate weights of the last layer in the decoder with the aid of neural-syntax, which improves the coding efficiency. We also try generating weights of other layers. However, the computation burden is heavy for the middle layers. For a 5×5 transposed convolutional layer in a low bit-rate model with the input channel 192 and output channel 192, it has 5×5×192×192=921,600 parameters to be generated. Considering that we use sequential fully-connected layers to generate these parameters, it will utilize quite a lot of GPU memory and computation resources. For the penultimate convolutional layer, the width of the output channel is 16, which makes the weight injection more feasible. Therefore, we fix other four layers and only generate the weights of the

penultimate convolutional layer. The results are shown in Fig. 2. As illustrated, the generated penultimate layer degrades the overall performance by a large margin.

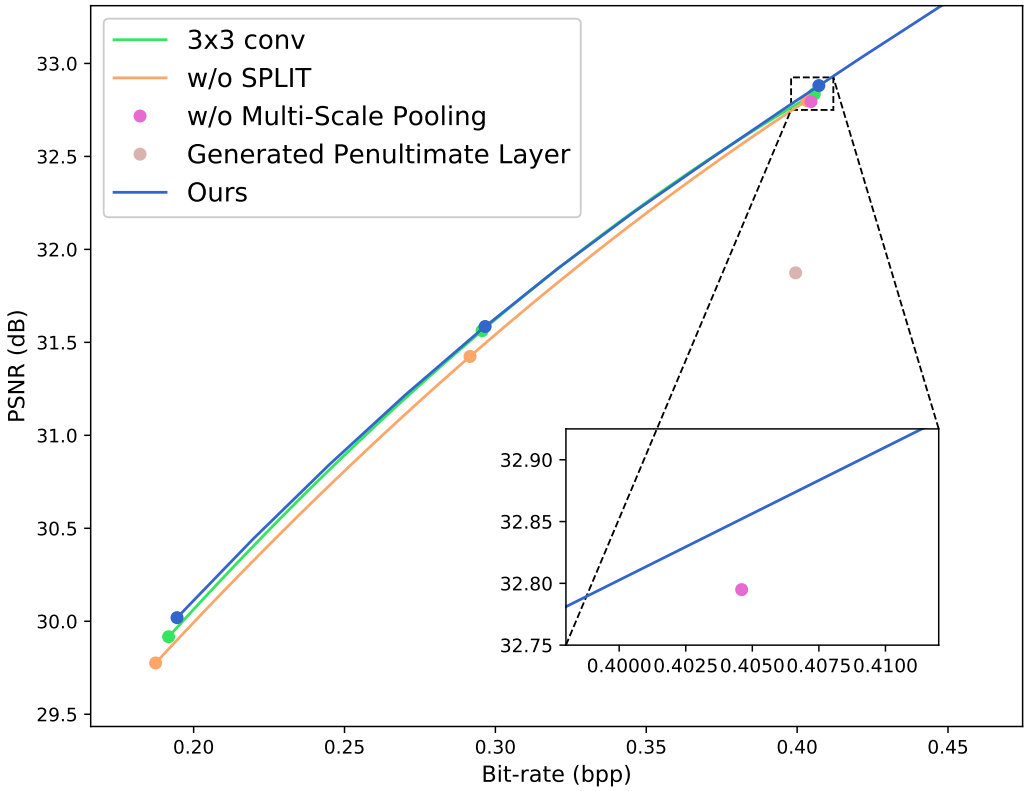


Figure 2. Illustration of ablation studies.

4) Effectiveness of Multi-Scale Pooling. In our Syntax Generator, we first extract multi-scale features then fuse them after the global pooling. The global pooling helps us handle the input with arbitrary resolution. However, the simple global pooling will inevitably lose too much spatial information. To handle this, we gradually down-scale the feature to remove spatial information progressively to obtain better representation. We compare the simple global pooling with our multi-scale pooling in Fig. 2. From the figure we can find that, with the aid of a multi-scale extraction mechanism, the performance can be improved by 0.06 dB in PSNR at the same bit-rate.

4. Subjective Results

In this section, we will provide more visual results in Fig. 3-6. We compare our method to classic image standard JPEG [14], advanced conventional codecs BPG [13] and VTM [5], and our baseline model [9].

In Fig. 3, we crop several patches from the reconstruction results to show that our method can preserve more details in a high-resolution image. In Fig. 3 (a) and (b), strings are fractured or vanished in other methods. In our method, though they are inevitably blurred due to compression, they still exist in most cases.

We also show reconstruction results in full resolution rather than cropped patches. In Fig. 4, compared to VTM, letters on the hat are clearer and there are no blocking artifacts in the sky. In Fig. 5, our method preserves the edges better than VTM.

Beyond common natural images, we also compare our method to the state-of-the-art codec VTM on human face images. In Fig. [10] (a), the face compressed by VTM is suffered from ringing artifacts, which doesn't exist in our result. In Fig. [10] (b), our result retains better subjective quality in both foreground and background. In background, our method keeps the edge of the text sharp without ringing artifacts. In foreground, the teeth are over-smoothed in VTM results while the teeth boundary in our reconstruction is clearer.

In all, beyond high objective results, our method can also provide better visual quality.

Figure 3. Subjective results compared to JPEG [14], BPG [6], ICLR 2019 [9] and VTM [5] on *vita-vilcina-3055.png* from CLIC.

5. Limitations

In this paper, we propose a data-dependent transform based learned image compression framework. Neural-syntax is introduced to generate the image-specific convolutional kernels at the decoder side. We further introduce a continuous mode decision mechanism at the testing stage to find a better latent representation as well as decoder parameters. Experimental results demonstrate the effectiveness of our method. However, there still exist some limitations.

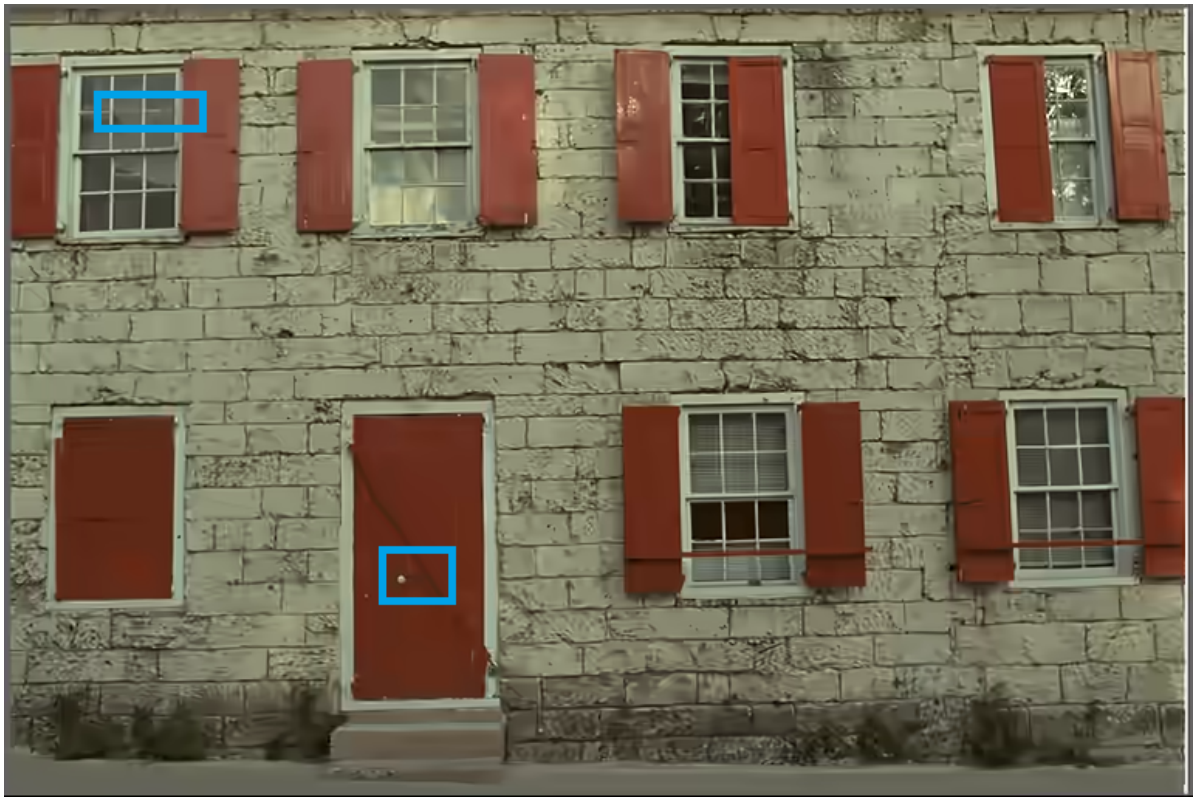
First, a gap exists between training and testing. When testing, we use the continuous mode decision mechanism to finetune the encoder. But in the training stage, we simply inference the encoder instead of overfitting it with the decoder fixed since it is extremely time-consuming to simply implement such a training strategy. A high-efficiency training algorithm which is able to narrow the gap between training and testing is still to be designed.

Second, continuous mode decision needs considerable GPU memory, especially for high-resolution images. It leaves room for a more delicate design to effectively implement the continuous mode decision without huge memory overhead.

(a) VTM 4:4:4, bpp=0.103

(b) Ours+, bpp=0.094

Figure 4. Subjective results on *kodim03* from Kodak [8].



(a) VTM 4:4:4, bpp=0.327

(b) Ours+, bpp=0.313

Figure 5. Subjective results on *kodim01* from Kodak [8]

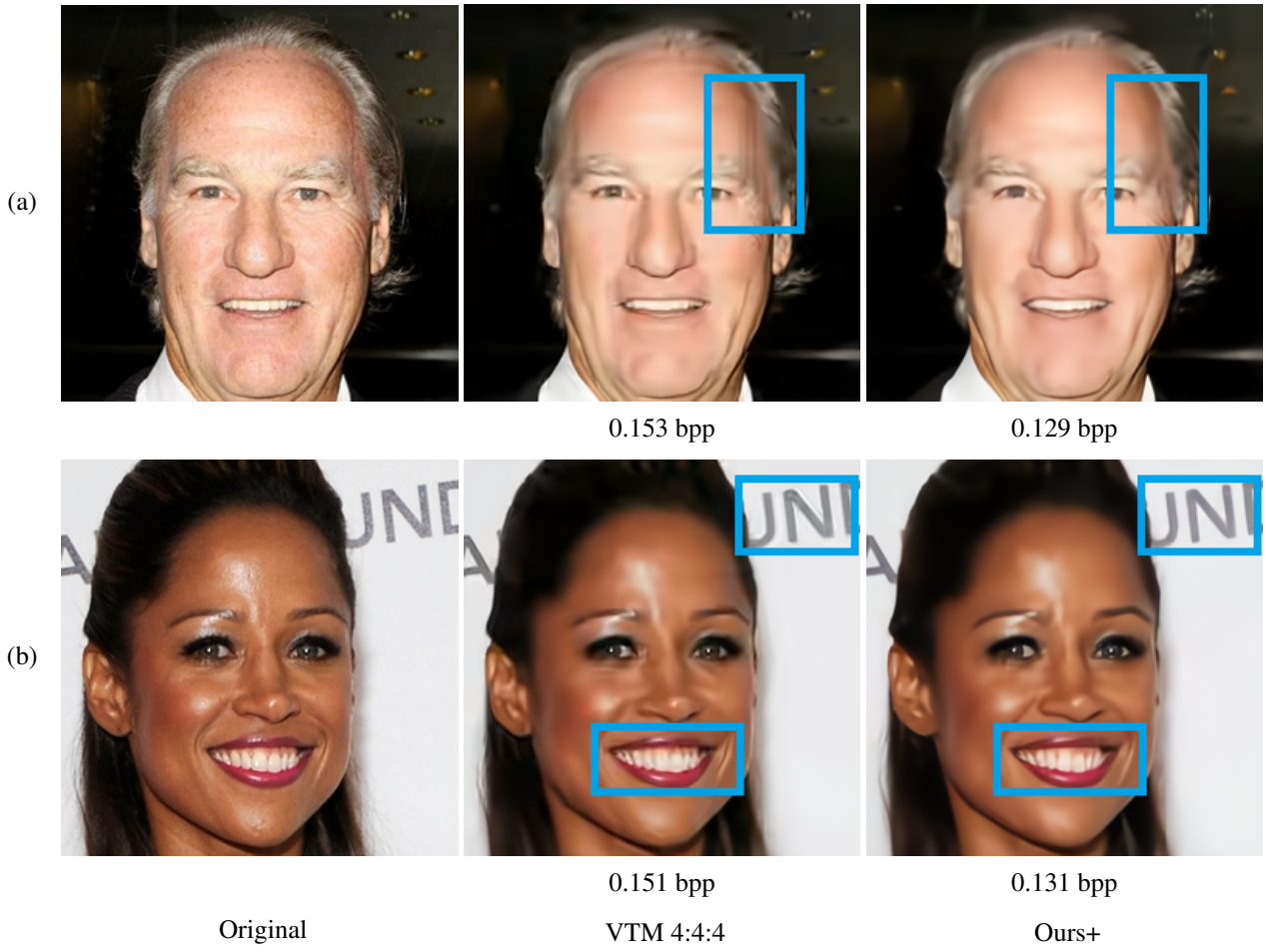


Figure 6. Subjective results on *#165111* and *#202591* from CelebA [10].

References

[1] Workshop and challenge on learned image compression, 2020. http://www.compression.cc. 2, 3

[2] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. In *Proc. of International Conference on Learning Representations*, 2018. 2, 3

[3] Gisle Bjontegarrd. Calculation of average PSNR differences between RD-curves. *VCEG-M33*, 2001. 2

[4] Benjamin Bross, Jianle Chen, and Shan Liu. Versatile video coding. *JVET-K1001*, 2018. 2

[5] Jianle Chen, Yan Ye, and Seung Hwan Kim. Versatile video coding (draft 8). *JVET-Q2002-v3*, 2020. 2, 3, 4, 5

[6] Bellard Fabrice. BPG image format (http://bellard.org/bpg/). accessed: 2021-09. 2018. 2, 3, 5

[7] Yueyu Hu, Wenhan Yang, Zhan Ma, and Jiaying Liu. Learning end-to-end lossy image compression: A benchmark. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2021. 2, 3

[8] Eastman Kodak. Kodak lossless true color image suite (photocd pcd0992). [online]. http://r0k.us/graphics/kodak/. 2013. 2, 3, 6, 7

[9] Jooyoung Lee, Seunghyun Cho, and Seung-Kwon Beack. Context adaptive entropy model for end-to-end optimized image compression. In *Proc. of International Conference on Learning Representations*, 2019. 1, 2, 3, 4, 5

[10] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proc. of IEEE International Conference on Computer Vision*, 2015. 4, 8

[11] David Minnen, Johannes Ballé, and George D Toderici. Joint autoregressive and hierarchical priors for learned image compression. In *Proc. of Advances in Neural Information Processing System*, 2018. 2

[12] Ben Niu, Weilei Wen, Wenqi Ren, Xiangde Zhang, Lianping Yang, Shuzhen Wang, Kaihao Zhang, Xiaochun Cao, and Haifeng Shen. Single image super-resolution via a holistic attention network. In *Proc. of European Conference on Computer Vision*, 2020. 1

[13] Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (HEVC) standard. *IEEE Trans. on Circuit System for Video Technology*, 22(12):1649–1668, 2012. 4

[14] Gregory K Wallace. The JPEG still picture compression standard. *IEEE Trans. on Consumer Electronics*, 38:18–34, 1992. 2, 4, 5