

END-TO-END LEARNED IMAGE COMPRESSION WITH FIXED POINT WEIGHT QUANTIZATION

Heming Sun^{*†}, Zhengxue Cheng[‡], Masaru Takeuchi^{*}, Jiro Katto^{*‡}

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

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

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

ABSTRACT

Learned image compression (LIC) has reached the traditional hand-crafted methods such as JPEG2000 and BPG in terms of the coding gain. However, the large model size of the network prohibits the usage of LIC on resource-limited embedded systems. This paper presents a LIC with 8-bit fixed-point weights. First, we quantize the weights in groups and propose a non-linear memory-free codebook. Second, we explore the optimal grouping and quantization scheme. Finally, we develop a novel weight clipping fine tuning scheme. Experimental results illustrate that the coding loss caused by the quantization is small, while around 75% model size can be reduced compared with the 32-bit floating-point anchor. As far as we know, this is the first work to explore and evaluate the LIC fully with fixed-point weights, and our proposed quantized LIC is able to outperform BPG in terms of MS-SSIM.

Index Terms— Image compression, neural networks, quantization, fixed-point, fine-tuning

1. INTRODUCTION

Image compression is important to relieve the burden of the image transmission and storage. In the past decades, several standards have been developed such as JPEG [1], JPEG2000 [2], WebP [3] and HEVC intra (BPG) [4]. Different from the hand-crafted ways, deep learning has shown a promising compression ability as reported in [5], [6], [7], [8], [9], [10]. By employing a proper neural network structure and enhanced probability models such as factorized and hyper prior, learned image compression (LIC) has outperformed the BPG in terms of MS-SSIM. Though LIC methods can achieve a good coding gain, utilizing many layers and channels will enlarge the network model, which prohibits the potential usage of LIC on resource-limited embedded devices.

Recently, weight quantization has shown a superior capability for the model compression in many networks such as AlexNet, ResNet and GoogleNet. A binary network with

1-bit weight and activation were proposed in [11]. Similarly, the number of bits can be reduced to two in a ternary weight network [12]. [13] pruned the network and quantized each network connection from 32-bit to 5-bit. An incremental quantization scheme with fine tuning was proposed in [14]. [15] kept the network accuracy while dividing the weights to the arbitrary bit-widths. [16] exploited the vector quantization to achieve 16-24 times compression ratio. An optimized quantization scale is learned in [17] to achieve a 4-bit precision at a comparable accuracy with full precision models.

Though quantization has achieved good performance for many popular models, there is almost no related work for LIC. [18] designed an integer network and provided a heuristic training scheme. However, using integer networks for main path will diminish the coding gain. In this paper, we quantize the weights to 8-bit fixed-point for both main path and hyper path by 1) formulating the quantization in grouping, 2) determining the optimal grouping and quantization scheme by coding gain, and 3) proposing a weight clipping fine tuning method. As a result, we can reduce about 75% model size compared with the 32-bit floating-point edition, and outperform BPG in terms of MS-SSIM.

2. QUANTIZATION METHOD FOR LIC

2.1. Formulation of Baseline Hyperprior Architecture

The baseline network we used is shown in Fig. 1, which is the *hyperprior-5* in [10]. The operation can be formulated as the following two equations

$$\begin{aligned} y &= g_a(x; \phi) \\ \hat{y} &= Q(y) \\ \hat{x} &= g_s(\hat{y}; \theta) \end{aligned} \quad (1)$$

$$\begin{aligned} z &= h_a(y; \phi_h) \\ \hat{z} &= Q(z) \\ \mu_y, \sigma_y &= h_s(\hat{z}; \theta_h) \end{aligned} \quad (2)$$

where Eq. 1 and Eq. 2 defines the main and hyper path operation respectively. x and \hat{x} are the raw and reconstructed image, y and z are two-layer latent nodes that will become \hat{y}

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

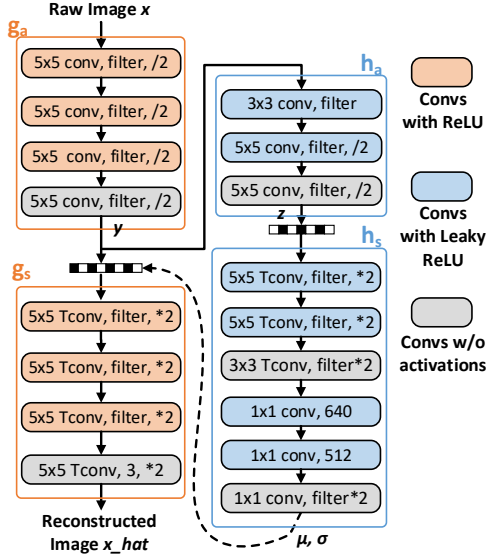


Fig. 1: Diagram of mean-scale hyperprior network.

and \hat{z} through a uniform quantization. ϕ and θ are the trained parameters. μ_y and σ_y are the estimated mean and variance for the usage of the probability model of y .

About the activation function, all the layers in g_a and g_s utilized ReLU, while all the layers in h_a and h_s used leaky-ReLU. Noted that there is no activations for the final layer in the analysis and synthesis transforms.

2.2. Grouping and Quantization Formulation

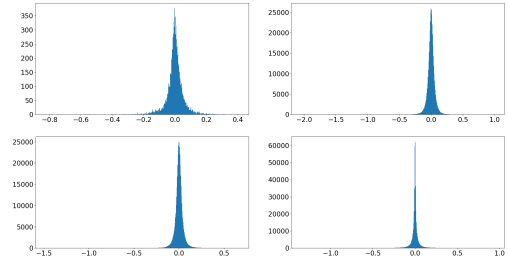
At first, we visualize the weight histograms in Fig. 2. We can see that both layer-wise and channel-wise weights follow the Gaussian distribution well with zero mean. According to [19], 8-bit integer can save around 19x multiply and 30x accumulation power reduction compared with the 32-bit floating point. Moreover, 8-bit can align with the bit-width of most on-chip memories. Therefore, the bit budget is set as 8-bit in this work, which can be represented as a fixed-point manner $\{1, IL, FL\}$ where 1 is the signed bit, IL and FL stands for the integer and fractional bits.

First, we scale the weights $\mathbf{w} \in R$ to $\mathbf{sw} \in (-2, 2)$, and then set IL and FL as 1-bit and 6-bit, respectively. By doing so, the most significant bit for IL and FL will not be wasted. The scaling is performed in groups. For the weights in the k -th group, \mathbf{w}_k are scaled with a scalar scaling factor (sf_k).

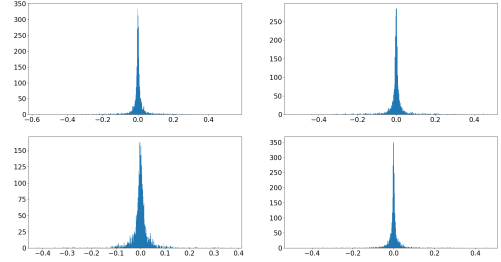
$$\mathbf{sw}_k = \mathbf{w}_k \times sf_k \quad (3)$$

where sf_k can be calculated by Eq. 4 which can be easily implemented as a shift operation in the hardware.

$$sf_k = 2^{-\lfloor \log_2 \max(|\mathbf{w}_k|) \rfloor} \quad (4)$$



(a) Four layers of analysis transform g_a .



(b) First four channels of the final layer of analysis transform g_a .

Fig. 2: Histograms of layer-wise and channel-wise weights.

After scaling, for each scaled weight element sw_i in the k -th group, the quantization is conducted as follows

$$Q(sw_i^k) = q_j^k \quad (5)$$

where $j \in 0, \dots, 2^{FL} - 1$. For the linear quantization (LQ), the operation of each group is performed as follows

$$q_j = \frac{\lfloor sw_i \times 2^{FL} \rfloor}{2^{FL}} + \frac{\xi_i}{2^{FL}} \quad (6)$$

where ξ_i is the rounding function in Eq. 7.

$$\xi_i = \begin{cases} 1, & \text{if } sw_i \times 2^{FL} - \lfloor sw_i \times 2^{FL} \rfloor > 0.5 \\ 0, & \text{otherwise} \end{cases} \quad (7)$$

For the non-linear quantization (NLQ), each q_j is determined according to the distributions of weights. One method is to use Lloyd's method [20], so that the codebook (i.e. q_j) can be optimized according to the following equation

$$\min \sum_{i=0}^{\text{len}(\mathbf{w}_k)-1} \sum_{j=0}^{2^{FL}-1} \|sw_i - q_j\|_2^2 \quad (8)$$

Despite Lloyd's algorithm is optimal, it requires hardware cost such as Look Up Table (LUT) to memorize the codebook for each group. To relieve the memory overhead, we developed an alternative memory-free codebook as shown in Eq. 9. By doing so, we only need to compare sw_i with power of two (i.e. 0.25 and 0.5) to obtain the quantized result

Table 1: Comparison of different quantization and grouping in terms of coding gain

	Origin	Layer-wise			Channel-wise	
		LQ	NLQ	Lloyd	LQ	NLQ
PSNR (dB)	32.32	28.81	32.08	32.18	31.60	32.25
bpp	0.529	0.770	0.555	0.537	0.569	0.539

Table 2: Comparison of different precision in terms of coding gain

Precision	$\frac{1}{2^{FL}}$	$\frac{1}{2^{FL+1}}$	$\frac{1}{2^{FL+2}}$	$\frac{1}{2^{FL+3}}$
PSNR (dB)	31.60	32.17	32.28	32.30
bpp	0.569	0.541	0.532	0.530

q_j at runtime so that there will be no memory consumption for the codebook.

$$q_j = \begin{cases} \left\lfloor \frac{sw_i \times 2^{FL-1}}{2^{FL-1}} \right\rfloor + \frac{\xi_i}{2^{FL-1}}, & \text{if } |sw_i| \in [0.5, 2) \\ \left\lfloor \frac{sw_i \times 2^{FL}}{2^{FL}} \right\rfloor + \frac{\xi_i}{2^{FL}}, & \text{if } |sw_i| \in [0.25, 0.5) \\ \left\lfloor \frac{sw_i \times 2^{FL+2}}{2^{FL+2}} \right\rfloor + \frac{\xi_i}{2^{FL+2}}, & \text{else} \end{cases} \quad (9)$$

2.3. Quantization and Grouping Scheme Determination

As described in the above, scaling is conducted in groups so that scaling factor (sf) for each group has to be stored. With more groups, there will be more non-zero weights after the quantization, while more consumption is required to store sf . In this paper, we explore two structured group scheme that is layer-wise (LW) and channel-wise (CW) grouping.

In the case of LW grouping, each group contains the weight $\mathbf{w} \in R^4$ where four dimension represents input channel, kernel width, kernel height and output channel. In the case of CW grouping, each group contains the weight $\mathbf{w} \in R^3$ where three dimension represents input channel, kernel width and height. For both grouping scheme, we attempt the method in Eq. 6 and Eq. 9. Besides, we also exploit Eq. 8 for LW rather than CW since it is not feasible to memorize LUTs for all the channels. For the model with a moderate rate ($\lambda=0.015$, MSE optimized, 4×10^5 iterations), the results are shown in Table 1. From the results, we can conclude that using CW-NLQ can reach the best coding gain.

For the NLQ in Eq. 9, the least magnitude that will not be quantized to zero is $\frac{1}{2^{FL+2}}$. To ensure this value is enough for the precision, we explore the coding gain of CW-LQ with different precisions as shown in Table 2. We can see that when increasing the precision from $\frac{1}{2^{FL}}$ to $\frac{1}{2^{FL+2}}$, there are obvious improvements for both PSNR and bpp. However, when further increasing the precision to $\frac{1}{2^{FL+3}}$, there is only 0.02dB and 0.002bpp difference that is quite trivial. Therefore, we decide to use $\frac{1}{2^{FL+2}}$ as the highest precision.

2.4. Weight Clipping Fine Tuning (WCFT)

By using proposed CW-NLQ, the coding loss has been trivial as shown in Table 1. However, according to the experimental

Algorithm 1 Proposed Training Method with Weight Clipping Fine Tuning

```

1: for number of training iterations  $I_1$  do
2:    $J(g_a, g_s, h_a, h_s) = \lambda D(x, \hat{x}) + R(\hat{y}) + R(\hat{z})$ 
3:   Update  $|\mathbf{w}_k|$  of  $g_a, g_s, h_a, h_s$  by descending its SGD
4: end for
5: Clip  $|\mathbf{w}_k|$  to  $2^{\lfloor \log_2 \max(|\mathbf{w}_k|) \rfloor} - \epsilon$ 
6: for number of fine tuning iterations  $I_2$  do
7:    $J(g_a, g_s, h_a, h_s) = \lambda D(x, \hat{x}) + R(\hat{y}) + R(\hat{z})$ 
8:   Update  $|\mathbf{w}_k|$  with straight-through estimator (STE)
9: end for

```

results, for the higher rate models, the loss of CW-NLQ is still not negligible.

As described in the above, the least magnitude of $|sw_i|$ that will not be quantized to zero is $\frac{1}{2^{FL+2}}$. Therefore, to generate more non-zero quantized results, larger $|sw_i|$ is desired. From Eq. 4, we can see that sf_k is fully dependent on $\max(|\mathbf{w}_k|)$, and sf_k will become larger with a smaller $\max(|\mathbf{w}_k|)$. Therefore, our target is to reduce $\max(|\mathbf{w}_k|)$.

First, we train the network as usual to obtain the optimized \mathbf{w} . After that, for each group, we clip the maximum magnitude to $2^{\lfloor \log_2 \max(|\mathbf{w}_k|) \rfloor} - \epsilon$ where ϵ is a very small value. After the clipping, we fine tune the network to compensate the loss caused by the clipping. During the fine tuning, we adopt the straight-through estimator [21] that is to preserve the gradient and cancels the gradient when w_i is larger than the clipping value as shown in Eq. 11. The pseudo code of overall training procedures with fine tuning is given in **Algorithm 1**.

$$w'_i = \begin{cases} w_i, & \text{if } w_i \leq 2^{\lfloor \log_2 \max(|\mathbf{w}_k|) \rfloor} - \epsilon \\ 2^{\lfloor \log_2 \max(|\mathbf{w}_k|) \rfloor} - \epsilon, & \text{otherwise} \end{cases} \quad (10)$$

$$g_{w_i} = g_{w'_i} 1_{|w_i| \leq 2^{\lfloor \log_2 \max(|\mathbf{w}_k|) \rfloor} - \epsilon} \quad (11)$$

3. EXPERIMENTAL RESULTS

3.1. Network and Training Details

For the training, we use 256×256 patches cropped from ImageNet [22], and set batch size as eight. I_1 and I_2 in **Algorithm 1** are set as 10^6 and 10^5 . The loss function is given in the following equation

$$J = \lambda D(x, \hat{x}) + R(\hat{y}) + R(\hat{z}) \quad (12)$$

where $D(x, \hat{x})$ is MSE and MS-SSIM to optimize PSNR and MS-SSIM, respectively, $R(\hat{y})$ and $R(\hat{z})$ are the consumed bits of y and z . λ is set as [0.001625, 0.00325, 0.0075, 0.015, 0.03, 0.05] for the MSE, and [3, 5, 10, 40, 80, 128] for the MS-SSIM to generate six models, respectively. For both MSE and MS-SSIM, we use 128 filters for four lower rate models, and 192 filters for two higher rate models. We adopt the WCFT for the two higher rate models.

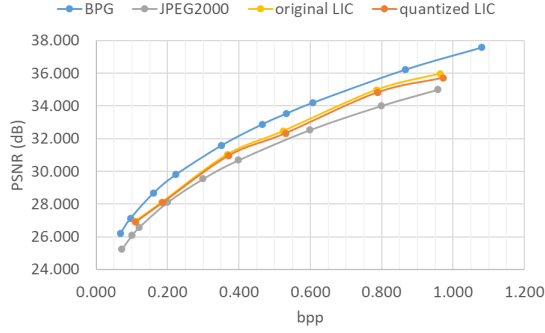


Fig. 3: PSNR comparison of 32-bit floating-point original LIC, proposed 8-bit fixed-point quantized LIC, BPG and JPEG2000.

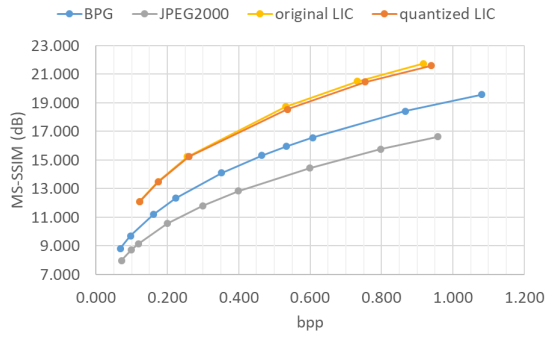


Fig. 4: MS-SSIM comparison of 32-bit floating-point original LIC, proposed 8-bit fixed-point quantized LIC, BPG and JPEG2000.

3.2. Coding Performance Evaluation

First, we evaluate the coding gain of our proposal by the Kodak dataset [23] with 24 distortion-free images, and the results are shown in Fig. 3 and Fig. 4. We can see that our proposed 8-bit fixed-point LIC is quite close to original 32-bit floating-point edition. For the four middle rate models, the BD-psnr [24] loss compared with the original anchor is only 0.1183dB and 0.1486dB for MSE and MS-SSIM, respectively. Besides, we can outperform JPEG2000 in terms of PSNR and perform better than BPG in terms of MS-SSIM. Noted that the PSNR of fixed-point LIC can be further improved by enhancing the floating-point anchor model.

We also evaluate the effect of the proposed WCFT in Table 3. Before using this scheme, for two high rate MSE models, the coding loss caused by the quantization is 0.382dB and 0.334dB, while it can be reduced to 0.148dB and 0.258dB. In addition, the bit increment caused by the quantization can also be reduced. Without using WCFT, the bpp is increased by 0.019 and 0.012, while the bpp is only increased by 0.004 and 0.009 after using WCFT. For two high rate MS-SSIM models, the coding loss can be decreased from 0.285dB and 0.509dB to 0.107dB and 0.264dB, respectively. The bpp increment can be reduced from 0.018 and 0.038 to 0.015 and

Table 3: Coding gain improvement by using proposed WCFT

	λ	MSE		MS-SSIM	
		0.03	0.05	80	128
w/o WCFT	PSNR (dB)	-0.382	-0.334	-0.285	-0.509
	bpp	+0.019	+0.012	+0.018	+0.038
with WCFT	PSNR (dB)	-0.148	-0.258	-0.107	-0.264
	bpp	+0.004	+0.009	+0.015	+0.013

Table 4: LIC model size comparison (MB)

Filter	128			192		
Component	weight	sf	total	weight	sf	total
Original	20.72	0	20.72	44.04	0	44.04
Proposed	5.18	0.0016	5.1816	11.01	0.0021	11.0121

0.013, respectively. Therefore, using WCFT is quite helpful for the coding gain improvement.

3.3. Memory Consumption Evaluation

We evaluate the model size comparison in Table 4. For the network structure in Fig. 1, the number of bytes for the weight can be calculated by Eq. 13 where I , O , H , W are input channel number, output channel number, kernel height and width, and i is the index of convolution layers. Overall, we have 17 layers. After quantizing each weight to 8-bit, the total weight storage can become one-fourth while there is memory overhead to store sf . According to our experiments, 4-bit is adequate to save one scalar sf . In the case of CW grouping, the number of sf is equal to the number of output channels. Overall, required bytes for the weights can be obtained by Eq. 14. From the results, we can see that about 75% memory consumption can be saved since the overhead of the additional scaling factor is negligible compared with the storage of weight itself.

$$\text{Original model size} = \sum_{i=0}^{layer-1} I_i \times H_i \times W_i \times O_i \times 4 \quad (13)$$

$$\begin{aligned} \text{Proposed model size} = & \sum_{i=0}^{layer-1} I_i \times H_i \times W_i \times O_i \times 1 \\ & + \sum_{i=0}^{layer-1} O_i \times 0.5 \end{aligned} \quad (14)$$

4. CONCLUSIONS

This paper proposes a fixed-point weight quantization method for LIC. First, we explore different kinds of grouping and quantization schemes, and then determine the optimal one based on the coding gain. In addition, to alleviate the coding performance loss caused by the quantization error, a fine tuning method is proposed. The results show that we can outperform the BPG in terms of MS-SSIM. For the future work, we will quantize the activations by fixed-point arithmetic and design the corresponding hardware architectures such as FPGA and ASIC.

5. REFERENCES

- [1] Gregory K Wallace, “The jpeg still picture compression standard,” *IEEE transactions on consumer electronics*, vol. 38, no. 1, pp. xviii–xxxiv, 1992.
- [2] Majid Rabbani and Rajan Joshi, “An overview of the jpeg 2000 still image compression standard,” *Signal processing: Image communication*, vol. 17, no. 1, pp. 3–48, 2002.
- [3] Li Lian and Wei Shilei, “Webp: A new image compression format based on vp8 encoding,” *Microcontrollers & Embedded Systems*, vol. 3, 2012.
- [4] 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.
- [5] Oren Rippel and Lubomir Bourdev, “Real-time adaptive image compression,” in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 2922–2930.
- [6] George Toderici, Damien Vincent, Nick Johnston, and et al., “Full resolution image compression with recurrent neural networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5306–5314.
- [7] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston, “Variational image compression with a scale hyperprior,” in *International Conference on Learning Representations*, 2018.
- [8] Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto, “Deep convolutional autoencoder-based lossy image compression,” in *2018 Picture Coding Symposium (PCS)*. IEEE, 2018, pp. 253–257.
- [9] Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto, “Learning image and video compression through spatial-temporal energy compaction,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10071–10080.
- [10] Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto, “Deep residual learning for image compression,” *IEEE Conference on Computer Vision and Pattern Recognition Workshop and Challenge on Learned Image Compression*, pp. 1–5, 2019.
- [11] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio, “Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1,” *arXiv preprint arXiv:1602.02830*, 2016.
- [12] Fengfu Li, Bo Zhang, and Bin Liu, “Ternary weight networks,” *arXiv preprint arXiv:1605.04711*, 2016.
- [13] Song Han, Huizi Mao, and William J Dally, “Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding,” *arXiv preprint arXiv:1510.00149*, 2015.
- [14] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen, “Incremental network quantization: Towards lossless cnns with low-precision weights,” *arXiv preprint arXiv:1702.03044*, 2017.
- [15] Hanmin Park and Kiyong Choi, “Cell division: weight bit-width reduction technique for convolutional neural network hardware accelerators,” in *Proceedings of the 24th Asia and South Pacific Design Automation Conference*, 2019, pp. 286–291.
- [16] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev, “Compressing deep convolutional networks using vector quantization,” *arXiv preprint arXiv:1412.6115*, 2014.
- [17] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan, “Pact: Parameterized clipping activation for quantized neural networks,” *arXiv preprint arXiv:1805.06085*, 2018.
- [18] Johannes Ballé, Nick Johnston, and David Minnen, “Integer networks for data compression with latent-variable models,” in *International Conference on Learning Representations*, 2019.
- [19] Mark Horowitz, “1.1 computing’s energy problem (and what we can do about it),” in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. IEEE, 2014, pp. 10–14.
- [20] Stuart Lloyd, “Least squares quantization in pcm,” *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.
- [21] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky, “Neural networks for machine learning,” *Coursera, video lectures*, vol. 264, pp. 1, 2012.
- [22] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.
- [23] Download from, “Kodak lossless true color image suite,” <http://r0k.us/graphics/kodak/>.
- [24] Gisle Bjontegaard, “Calculation of average psnr differences between rd-curves,” *VCEG-M33*, 2001.