Extreme Low Bitrate Image Compression System for Mobile Deployment

Junqi Wu[1], Wenhong Duan[2], Xianping Ma[3], Jianhui Chang[1], Shanshe Wang[1], Siwei Ma[1] and Chuanmin Jia[1]
[1]Peking University [2]Shanghai Jiao Tong University [3]Chinese University of Hong Kong, Shenzhen
junqiwu@alumni.pku.edu.cn

2024 IEEE 26th International Workshop on Multimedia Signal Processing (MMSP) | 979-8-3503-8725-4/24/$31.00 ©2024 IEEE | DOI: 10.1109/MMSP61759.2024.10743248

***Abstract*—End-to-end image compression has achieved satisfactory results in recent studies. However, existing methods suffer from high complexity of complicated neural network computation and cannot be directly deployed on mobile devices due to the limitations of computing ability and storage. Therefore, considering the resource and computing ability constrains of the mobile devices, we make a trade-off in this paper between rate-distortion (R-D) performance, inference time, and model complexity. Then we design a novel lightweight perceptual image compression framework to alleviate the storage and complexity burden of mobile devices. Moreover, we design a hardware-friendly deployment scheme to apply the proposed compression framework on high-end mobile devices, which can achieve efficient image compression. Based on the above structures, we propose the first mobile system that achieves image compression on mobile devices. The supplementary material of our system demo is on https://sigport.org/documents/extreme-low-bitrate-image-compression-system-mobile-deployment.**

***Index Terms*—Image compression, low bitrate, mobile devices, hardware-friendly**

I. Introduction

Image compression has played an important role in the field of signal processing for efficient image transmission and storage. Conventional compression standards JPEG [1], JPEG2000 [2], High Efficiency Video Coding (HEVC) [3] and Versatile Video Coding (VVC) [4] designed hand-crafted module, which uses transform matrix, intra prediction, quantization, loop filters technology to remove the redundancy for higher coding efficiency. With the development of deep learning technologies, end-to-end image compression frameworks have been widely investigated.

Learned image compression (LIC) [5]–[12] methods have outperformed the latest compression standard Versatile Video Coding (VVC). Most image compression frameworks can achieve higher image coding efficiency by designing more effective analysis and synthesis transform networks. Ballé *et al.* first proposed the learned image compression frameworks [6] transforming images to compact representations. Cheng *et al.* proposed attention modules and used discretized Gaussian Mixture Likelihoods to parameterize the distributions [9], which achieved a comparable performance with VVC on the PSNR metric. Besides, He *et al.* proposed the spatial-channel contextural adaptive model to improve performance without extra complexity [13]. Moreover, Liu *et al.* proposed a mixed Transformer-CNN [14] architecture, which took advantage of local and non-local information to improve the coding efficiency. The proposed architecture can achieve superior performance than CNN-based frameworks.

Though the above methods have obtained satisfactory performance, they commonly target at a general bitrate range (0.1–0.5 bpp) and exhibit severely degraded visual reconstruction quality on an extremely low bitrate range ($\leq$ 0.1 bpp or $\geq$ 1 Mbps for image) [15]. Therefore, the low bitrate compression framework is necessary for mobile devices which have limited hardware resources. Considering the real application, we design a lightweight low-bitrate image compression framework. In this framework, we first transfer the image to a semantic map. Then the semantic map is compressed and decompressed based on the variational auto-encoder (VAE) architectures. Finally, the reconstructed semantic map is used to reconstruct the image synthesis network. Besides, the above methods neglect the demand for real applications. Considering the characteristics of mobile devices, we design a hardware-friendly deployment scheme to achieve efficient image compression.

Specifically, we decouple the entire structure into six modules, including encoder, hyper encoder, arithmetic encoder, arithmetic decoder, hyper decoder, and decoder. Among these, arithmetic encoder and arithmetic decoder are achieved by transferring based on Java Native Interface (JNI) tools from C++ to Java class. The model requires retraining to adapt to the operational conditions of mobile devices. After retraining, we deploy it on mobile devices using Android development tools. Additionally, we divide the framework into six modules, which are then frozen to facilitate subsequent computation on mobile devices. Then the arithmetic encoder and decoder are achieved based on the Java Native Interface (JNI) tools. Then the index and scale of the Cumulative Distribution Function (CDF) are calculated by the Java class. CDF is obtained by the look-up table (LUT). Finally, we use the transfer from tensor to Java class for image reconstruction and visual performance based on the coordinate alignment.

To evaluate the effectiveness of the framework, we test the proposed framework on multiple datasets in terms of LIPIS metric, inference time, system overhead, and FLOPS. Lower LPIPS indicates better visual quality. The experimental results show that the proposed framework can achieve satisfactory visual quality on mobile devices. Overall, the main contributions can be summarized as follows:

This work was supported in part by the National Natural Science Foundation of China under grants 62025101, 62031013, U21B2012, 62371008, and in part by the Xplore Prize, which are gratefully acknowledged.

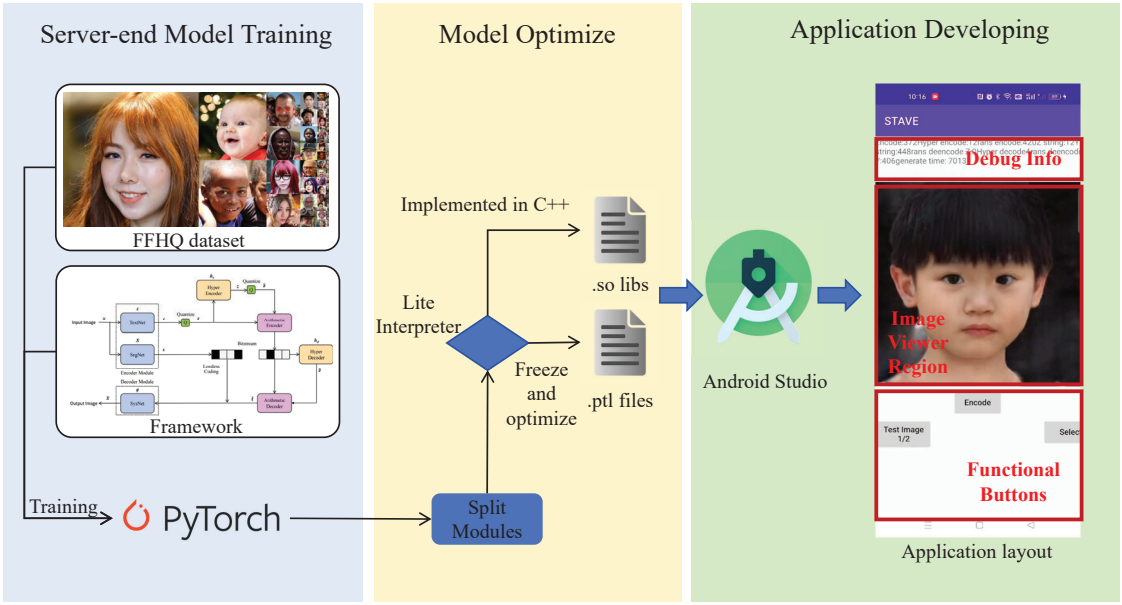


Fig. 1: Pipeline of deployment, the .ptl file format is specifically optimized for mobile devices and other lightweight environments, aiming to reduce model loading time and memory usage, while enhancing inference speed.

- Considering the computing ability and storage characteristics of mobile devices, we make a trade-off to balance the Rate-Distortion (R-D) performance and inference time.
- We first propose a system deployed on mobile devices, which achieves an end-to-end image compression framework. We design a hardware-friendly deployment scheme to achieve image compression on mobile devices.
- The experimental results show that the proposed mobile system can achieve high visual quality on multiple representative datasets.

II. Related Work

A. CNN-based Learned Codec

In 2018, Ballé first proposed a CNN-based architecture [6]. They further proposed a variational auto-encoder (VAE) architecture [8] and employed a hyper for higher coding efficiency. In works [16], the authors applied autoregressive modules to an image compression model. Moreover, attention modules are adopted in the VAE architectures. Chen *et al.* proposed a non-local attention mechanism [17] to obtain the global and local information for performance improvement. Cheng *et al.* further optimized the attention block and used discretized Gaussian Mixture likelihoods to parameterize the distributions of latent representation [9]. Furthermore, Chen *et al.* proposed an octave residual network [18], which can preserve more meaningful information in limited bitstream. Duan *et al.* proposed an information-guided compression framework using cross-component attention mechanism [11], which can achieve efficient image compression in YUV420 format. The proposed framework outperforms Versatile Video Coding (VVC) with 8.37% BD-rate reduction on average.

B. Transformer-based Learned Codec

With the development of Vision Transformer (ViT) and Swin-Transformer, transformer-based models are designed. Zhu *et al.* proposed the SwinT-ChARM model [19], which outperformed VTM-12.1 with comparable decoding speed. Koyuncu *et al.* proposed a transformer-based context model [20], which utilized an attention mechanism to spatio-channel attention. Kim *et al.* proposed the Information Transformer (Informer) [21] that exploits both global and local information in a content-dependent manner by an attention mechanism.

III. Proposed Method

A. Network Architecture

The architecture of the framework proposed for practical deployment, as shown in Figure 2, is founded on the structure-texture decomposition philosophy [22]. The framework is segmented into seven components: Semantic Encoder $\boldsymbol{S}$, Texture Encoder $\boldsymbol{\epsilon}$, hyper encoder $\boldsymbol{h_e}$, hyper decoder $\boldsymbol{h_d}$, arithmetic encoder (AE), arithmetic decoder (AD) and decoder $\boldsymbol{g}$. At the encoder side, Semantic Encoder $\boldsymbol{S}$ and Texture Encoder $\boldsymbol{\epsilon}$ amalgamate to extract both structural and textural information from the input image $\boldsymbol{x}$. The extraction process encompasses two primary steps: (1) the structure map, guided by the semantic map, is downsampled and encoded losslessly; (2) the textural information undergoes compression into texture representations $\boldsymbol{t}$ with the assistance of the semantic map and is subsequently quantized into bitstreams. The bitstreams are employed at the decoder side to synthesize the decoded image, effectively reconstructing the original image's content

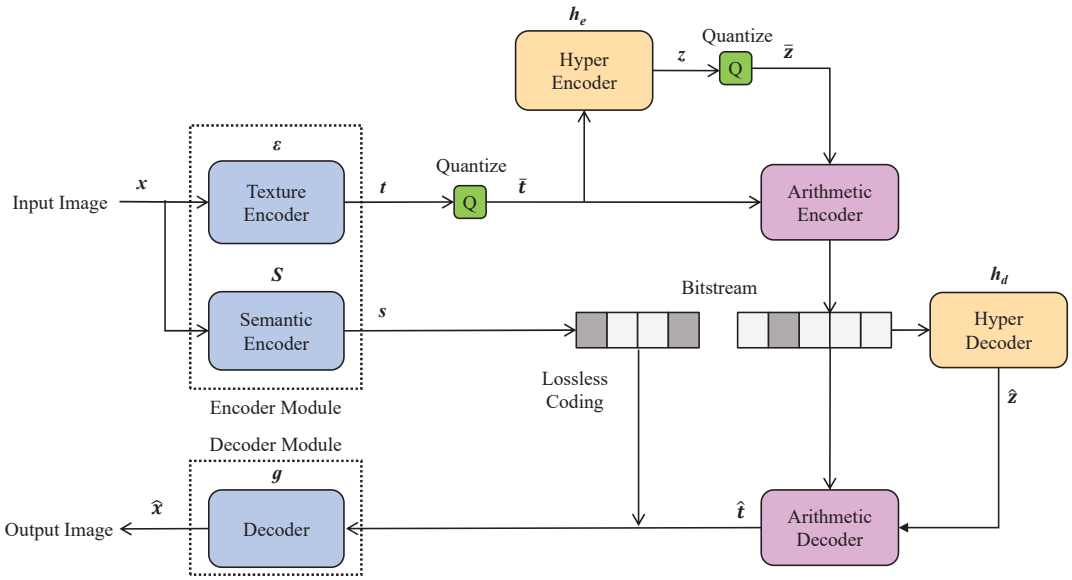


Fig. 2: Overview of the proposed mobile deployed perceptual image coding framework.

and intricate details by integrating semantic and textural information. Thus the proposed framework efficiently encodes semantic-texture representations into bitstreams, which can further optimizing these streams for enhanced storage and transmission efficiency through lossless coding techniques.

At encoder side, input image $\boldsymbol{x}$ is decomposed into structural representation $\boldsymbol{s}$ and semantic-wise textural representation $\boldsymbol{t}$. The structural representation encapsulates spatial information, semantic details, shape characteristics, and positional attributes from the original image and is extracted using Semantic Encoder $\boldsymbol{S}$, a pre-trained semantic segmentation network as proposed by Zhao *et al.* [23]. The semantic-wise texture representation, which conveys the texture information of each semantic region, is learned by Texture Encoder $\boldsymbol{\epsilon}$, built upon the attentive representation model proposed by Chang *et al.* [15].

To further compress the generated two types of representations, we employ the cross-channel entropy model proposed in [24]. The cross-channel entropy model is divided into four modules: hyper encoder, hyper decoder, arithmetic encoder, and arithmetic decoder. The hyper encoder and decoder, are executed through a lite interpreter on the Android platform. The hyper encoder estimates the distribution information of the semantic-wise textural representation and stores it in vector $\boldsymbol{z}$. The hyper decoder reconstructs the distribution information of semantic-wise textural representation from vector $\boldsymbol{z}$. Concurrently, the arithmetic encoder and decoder, functioning as encoding modules, are implemented based on the range Asymmetric Numeral Systems (rANS) as defined by Duda [25].

At the decoder side, the received bitstreams are inversely decoded to structural and texture representations. Subsequently, the generative network reconstructs the target image $\hat{\boldsymbol{x}}$ under the guidance of the decoded semantic segmentation map and semantic-wise texture representations.

B. Model Preparation

In Android platform, certain operations of the PyTorch neural network can be efficiently executed using the Torch Mobile Lite Interpreter. Due to the presence of operations in the original network not supported by the lite interpreter, we partition the framework into six distinct modules: encoder, hyper encoder, arithmetic encoder, arithmetic decoder, hyper decoder, and decoder. Encoder, hyper encoder, hyper decoder, and decoder are supported by the lite interpreter.

To enhance performance, we implement the following optimization steps: integrating Semantic Encoder and Texture Encoder to a single TorchScript model defined as encoder module in Fig. 1, and folding adjacent Conv2d layer and BatchNorm layer into a single Conv2d layer.

To further elucidate the principle of folding adjacent Conv2d and BatchNorm layers, we denote rearranged original input $\boldsymbol{X}$ by im2col algorithm [26] as $\boldsymbol{X'}$. Assuming that $\boldsymbol{W_c}$, $\boldsymbol{b_c}$ are the weights and bias from the original Conv2d layer and scalar γ and β are the scale parameter and shift parameter from original BatchNorm layer. Operations of the adjacent Conv2d and BatchNorm can be represented as :

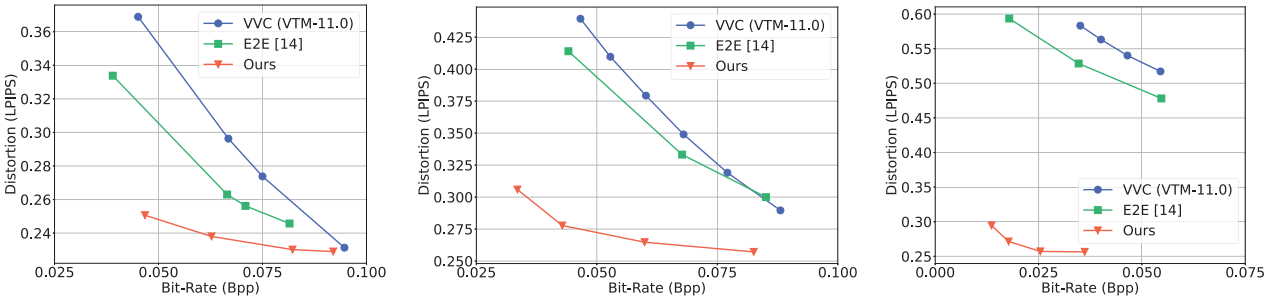
$$BN(Conv2d(\boldsymbol{X})) = \gamma(\boldsymbol{X'W_c} + \boldsymbol{b_c}) + \beta. \tag{1}$$

The equation above can be simplified as a new Conv2d layer with weights $\boldsymbol{W_{fc}}$ and bias $\boldsymbol{b_{fc}}$:

$$\boldsymbol{W_{fc}} = \gamma \boldsymbol{W_c}, \tag{2}$$

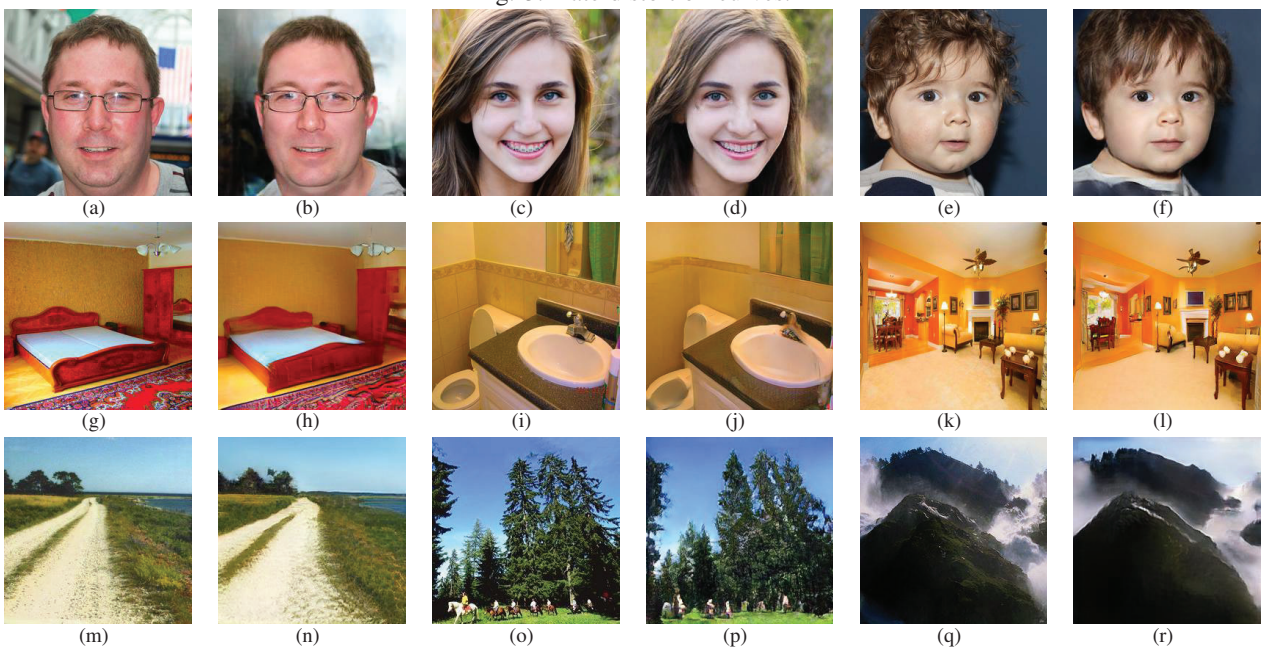
$$\boldsymbol{b_{fc}} = \gamma \boldsymbol{b_c} + \beta. \tag{3}$$

Then we deploy the modules to Android devices by involving the following steps: converting the trained model to TorchScript format, integrating PyTorch Mobile into the Android project by adding it as a Gradle dependency, loading the saved model using PyTorch Mobile for making predictions



(a) Rate-distortion curve (FFHQ) (b) Rate-distortion curve (Indoor) (c) Rate-distortion curve (Outdoor)

Fig. 3: Rate-distortion curves.



(a) (b) (c) (d) (e) (f)

(g) (h) (i) (j) (k) (l)

(m) (n) (o) (p) (q) (r)

Fig. 4: Comparison between original images and reconstructed images. In each pair, the left image is original and the right image is reconstructed by proposed framework on the server end.

TABLE I: System level feature illustration.

CPU Occupancy	Memory Consumption	Min Storage Required	Average Power	Energy Consumption Per Image
50.0%	1.8GB	1.0GB	9.9W	81.2J

TABLE II: Inference time and model complexity of our approach. The FLOPs of each module are estimated from torchScript and the process of rANS coding uses fixed-point operation rather than float-point operation.

Module	Inference Time(ms)	FLOPs
Encoder	277.57	21.98G
Decoder	6972.14	209.28G
hyper encoder	4.00	0.04M
hyper decoder	2.00	0.16M
rANS encoder	471.14	-
rANS decoder	429.43	-
Total	8170.14	231.26G

in the Android application. The IDE settings are Android Studio version 4.0.1, build tools version 34.0.0. And our target mobile device is OPPO PFFM10[1] with Android API level 31.

C. Entropy Model

Challenges in the deployment of neural networks on mobile devices encompass two pivotal aspects: firstly, the deployment and inference of network weights through quantization, and secondly, the efficient implementation of entropy coding. As stated in III-B, we can leverage the lite interpreter to facilitate the implementation of most neural network operations on mobile devices directly. However, the arithmetic coding operations, which are denoted as AE and AD in Fig. 2, are not supported by the lite interpreter, necessitating the implementation of a custom interface accessible to Java.

[1]https://www.oppo.com/en/smartphones/series-find-x/find-x5/specs

The arithmetic coding in our framework is based on range Asymmetric Numeral Systems (rANS) algorithm [25]. Given the computationally intensive nature of the rANS algorithm, we select to compile static library files based on an efficient rANS C library [27].

$$\begin{aligned} p =& (a \gg 32) * (b \gg 32) \\ &+ (a \gg 32) * (b \& (2^{32} - 1) \gg 32 \\ &+ (a \& (2^{32} - 1)) * (b \gg 32) \gg 32. \end{aligned} \tag{4}$$

We propose a mobile-friendly 128-bit fixed-point multiplication function, offering advantages in terms of broader compat ibility with various versions of the GNU compiler in Java Native Development Kit (NDK). During the rANS encoding process, mobile-friendly 128-bit fixed-point multiplication function can return a 64-bit integer p (Equation 4), the high 64-bit of the product of two 64-bit unsigned integers input a and b, to update encoding state value.

Moreover, we observe that the process of building indexes for $\boldsymbol{t}$ in the Gaussian Conditional model involves a sequential search to get index of pre-trained Gaussian cumulative distribution function (CDF) that best matches to $\boldsymbol{z}$. However, due to the finite range of quantized values in $\boldsymbol{z}$, we replace the sequential search process by directly recording each $\boldsymbol{z}$ with its corresponding index.

IV. Experiments

A. Experimental Settings

We train and implement the proposed architecture using PyTorch. To evaluate the effectiveness of the proposed framework, we perform experiments on multi-scenarios including human face [28], natural outdoor, and indoor scenes [29]. The implementation details follow the configurations of the framework [15].

The mobile device OPPO PFFM10 is chosen as the experimental platform. We use the rate-distortion curve as the metric for evaluation on the coding efficiency. We also adopt the CPU occupation, memory consumption, storage requirement, and energy consumption as the system overhead to evaluate the performance of system.

B. Performance Evaluation

In Fig. 3, we compare the rate-distortion performance of our system to the following representative baselines: (1) the latest video coding standard, namely Versatile Video Coding (VVC) [4], where VVC reference software with common test condition intra configuration is utilized; (2) the typical deep learning-based end-to-end codec [16], named as E2E. We also provide the comparison between images reconstructed from mobile devices with original images in Fig. 4, which indicates that our system on mobile devices can reconstruct images with favorable subjective quality at extremely low bitrates.

C. System-Level Features

We provided our system level feature in Table I. The CPU occupancy, memory consumption, and storage requirement data are monitored in real-time using the Android Studio profiler, connected to the mobile device via a data cable. The runtime power consumption of the system is calculated by subtracting the idle power from the run time power of the mobile device. The idle state power is measured using a power meter socket when the device has the screen turned on but without running any applications. Energy consumption is determined by subtracting the idle state energy consumption from the total energy consumed for processing a single image, which is measured directly by power meter socket.

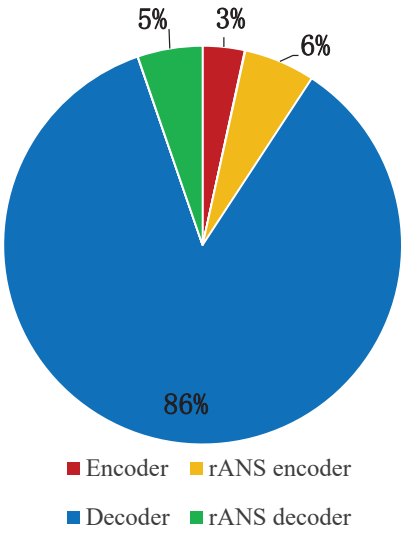


Fig. 5: Run time distribution of each module in our framework.

The system level feature reveals that the system experiences relatively low overhead in terms of memory and power consumption. However, the observed performance bottleneck is primarily constrained by the computational capabilities of the CPU. Our system's minimal hardware requirements allow it to be deployed across a wide range of smartphone models, demonstrating its compatibility and effectiveness for mobile deployment.

In our work, we provide the inference time and estimated FLOPs for each module of our system on the mobile device, as shown in Table II. The latency analysis for each module, as depicted in Fig. 5, reveals that the generator module is the main bottleneck within the system. The performance bottleneck primarily results from the substantially greater computational requirements of decoder compared to other system modules.

V. Conclusion

We proposed a lightweight low-bitrate image compression framework and designed a hardware-friendly deployment scheme, which applied the proposed framework to mobile devices. Furthermore, we proposed the first mobile system that achieved efficient image compression on mobile devices. Experimental results demonstrated the effectiveness of the proposed mobile system.

REFERENCES

[1] Gregory K Wallace, "The JPEG still picture compression standard," *IEEE Trans. Consum. Electron.*, vol. 38, no. 1, pp. xviii–xxxiv, Feb. 1992.

[2] Majid Rabbani, "JPEG2000: Image compression fundamentals, standards and practice," *J. Electron. Imag.*, vol. 11, no. 2, pp. 286, 2002.

[3] Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, 2012.

[4] Benjamin Bross, Ye-Kui Wang, Yan Ye, Shan Liu, Jianle Chen, Gary J Sullivan, and Jens-Rainer Ohm, "Overview of the versatile video coding (VVC) standard and its applications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 10, pp. 3736–3764, 2021.

[5] Siwei Ma, Xinfeng Zhang, Chuanmin Jia, Zhenghui Zhao, Shiqi Wang, and Shanshe Wang, "Image and video compression with neural networks: A review," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 6, pp. 1683–1698, 2019.

[6] Johannes Ballé, Valero Laparra, and Eero P Simoncelli, "End-to-end optimization of nonlinear transform codes for perceptual quality," in *Proc. Picture Coding Symp.* IEEE, 2016, pp. 1–5.

[7] Johannes Ballé, Valero Laparra, and Eero P Simoncelli, "End-to-end optimized image compression," in *Proc. Int. Conf. Learn. Represent.*, 2017.

[8] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston, "Variational image compression with a scale hyperprior," in *Proc. Int. Conf. Learn. Represent.*, 2018.

[9] Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto, "Learned image compression with discretized gaussian mixture likelihoods and attention modules," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 7939–7948.

[10] Wenhong Duan, Kai Lin, Chuanmin Jia, Xinfeng Zhang, Siwei Ma, and Wen Gao, "End-to-end image compression via attention-guided information-preserving module," in *Proc. Int. Conf. Multimedia Expo*. IEEE, 2022, pp. 1–6.

[11] Wenhong Duan, Zheng Chang, Chuanmin Jia, Shanshe Wang, Siwei Ma, Li Song, and Wen Gao, "Learned image compression using cross-component attention mechanism," *IEEE Trans. Image Process.*, 2023.

[12] Chuanmin Jia, Xinyu Hang, Shanshe Wang, Yaqiang Wu, Siwei Ma, and Wen Gao, "FPX-NIC: An FPGA-accelerated 4K ultra-high-definition neural video coding system," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 9, pp. 6385–6399, 2022.

[13] Dailan He, Ziming Yang, Weikun Peng, Rui Ma, Hongwei Qin, and Yan Wang, "ELIC: Efficient learned image compression with unevenly grouped space-channel contextual adaptive coding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 5718–5727.

[14] Jinming Liu, Heming Sun, and Jiro Katto, "Learned image compression with mixed transformer-cnn architectures," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2023.

[15] Jianhui Chang, Jian Zhang, Jiguo Li, Shiqi Wang, Qi Mao, Chuanmin Jia, Siwei Ma, and Wen Gao, "Semantic-aware visual decomposition for image coding," *Int. J. Comput. Vis.*, pp. 1–23, 2023.

[16] David Minnen, Johannes Ballé, and George D Toderici, "Joint autoregressive and hierarchical priors for learned image compression," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, vol. 31, pp. 10771–10780.

[17] T Chen, H Liu, Z Ma, Q Shen, X Cao, and Y Wang, "End-to-end learnt image compression via non-local attention optimization and improved context modeling," *IEEE Trans. Image Process.*, 2021.

[18] Fangdong Chen, Yumeng Xu, and Li Wang, "Two-stage octave residual network for end-to-end image compression," in *Proc. AAAI Conference on Artificial Intelligence*, 2022, vol. 36, pp. 3922–3929.

[19] Yinhao Zhu, Yang Yang, and Taco Cohen, "Transformer-based transform coding," in *Proc. Int. Conf. Learn. Represent.*, 2022.

[20] A Burakhan Koyuncu, Han Gao, Atanas Boev, Georgii Gaikov, Elena Alshina, and Eckehard Steinbach, "Contextformer: A transformer with spatio-channel attention for context modeling in learned image compression," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2022, pp. 447–463.

[21] Jun-Hyuk Kim, Byeongho Heo, and Jong-Seok Lee, "Joint global and local hierarchical priors for learned image compression," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 5992–6001.

[22] Jianhui Chang, Zhenghui Zhao, Chuanmin Jia, Shiqi Wang, Lingbo Yang, Qi Mao, Jian Zhang, and Siwei Ma, "Conceptual compression via deep structure and texture synthesis," *IEEE Trans. Image Process.*, vol. 31, pp. 2809–2823, 2022.

[23] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia, "Pyramid scene parsing network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2881–2890.

[24] Jianhui Chang, Zhenghui Zhao, Lingbo Yang, Chuanmin Jia, Jian Zhang, and Siwei Ma, "Thousand to one: Semantic prior modeling for conceptual coding," in *Proc. Int. Conf. Multimedia Expo*. IEEE, 2021, pp. 1–6.

[25] Jarek Duda, "Asymmetric numeral systems: entropy coding combining speed of huffman coding with compression rate of arithmetic coding," *arXiv preprint arXiv:1311.2540*, Dec. 2013.

[26] Kumar Chellapilla, Sidd Puri, and Patrice Simard, "High performance convolutional neural networks for document processing," in *Proc. 10th Int. Workshop Frontiers Handwriting Recognit.* Suvisoft, 2006.

[27] Fabian Giesen, "Interleaved entropy coders," *arXiv preprint arXiv:1402.3392*, 2014.

[28] Tero Karras, Samuli Laine, and Timo Aila, "A style-based generator architecture for generative adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4401–4410.

[29] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba, "Scene parsing through ADE20K dataset," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 633–641.