DCDiff: Enhancing JPEG Compression via Diffusion-based DC Coefficients Estimation

Ziyuan Zhang[†], Han Qiu[†]*, Tianwei Zhang[‡], Bin Chen[§], Chao Zhang[†]
[†]Tsinghua University, China, ziyuan-z23@mails.tsinghua.edu.cn, {qiuhan,chaoz}@tsinghua.edu.cn
[‡]Nanyang Technological University, Singapore, tianwei.zhang@ntu.edu.sg
[§]Harbin Institute of Technology, Shenzhen, China, chenbin2021@hit.edu.cn

2025 62nd ACM/IEEE Design Automation Conference (DAC) | 979-8-3315-0304-8/25/$31.00 ©2025 IEEE | DOI: 10.1109/DAC63849.2025.11132562

***Abstract*—JPEG is the most widely-used image compression method on low-cost cameras which cannot support learning-based compressors. One promising approach to enhance JPEG aims to drop DC coefficients at the cameras' ends (without extra computation) and reconstruct those DC coefficients after receiving them. They all face the challenge that their DC reconstruction relies on a statistical property, which will cause deviation-introduced errors and propagate. In this paper, we propose `DCDiff`, a novel end-to-end DC estimation method to tackle the above challenge. Instead of using statistical methods to recover DC coefficients and then fix errors, we directly leverage a generative model to estimate DC coefficients in an end-to-end manner. In the meantime, we generate masks to correct certain image locations that do not satisfy the statistical distribution to suppress error propagation. Extensive experiments show that `DCDiff` not only outperforms all baselines on compression performance but also introduces a tiny impact on downstream tasks and is fully compatible with 2 typical low-cost processors with JPEG support.**

***Index Terms*—JPEG, Image compression, Diffusion model**

I. Introduction

The deployment of numerous low-cost front-end cameras enables the development of smart Internet of Things (IoT) systems such as security surveillance [1], smart cities [2], and intelligent transportation [3]. Due to the limited computing and power capacity, front-end cameras capture large volumes of images but transmit them to cloud servers for further processing [4], which generates an increasing need for efficient image compression on front-end sensors. Many image compression approaches are proposed such as transformation-based ones (e.g. JPEG2000 [5], WebP [6], etc.) and learning-based ones (e.g. compressive sensing [7], neural image codec [8], etc.). However, many actual cameras tend to use JPEG [9] still due to JPEG's real-time capability, low computational requirements, and high compatibility. For example, ESP32-CAM, a widely-used low-power camera [10]–[12], has a max power of 1.55 Watt and can only support JPEG [13] instead of learning-based encoders.

JPEG is a tunable method to compress images block by block (e.g. 8×8 block) that allows users to choose a lower quality factor (i.e. Q table [14]) to enhance compression ratio on high-frequency (AC) coefficients [15]. However, using a lower-quality factor decreases image quality at decompression. As shown in Fig. 1, another idea [16] to enhance JPEG compression is to preserve AC coefficients but drop the direct current (DC) coefficients at the sender's end and recover them at the receiver's end. This is based on 3 reasons (details in Section II-B). First, DC coefficients are the average pixel values of each image block, which means after discarding DC, the difference between each pixel value within a block remains unchanged. Second, DC coefficients usually have large values, which need more bits to present when coding. Third, dropping DC coefficients does not require any modifications to the JPEG implementation, making it compatible with any low-cost cameras.

*Corresponding author.

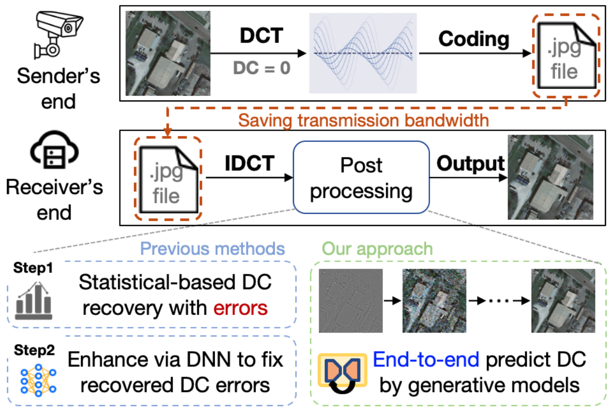

Sender's end
DCT
DC = 0
Coding
.jpg file
Saving transmission bandwidth
Receiver's end
.jpg file
IDCT
Post processing
Output
Previous methods
Step1
Statistical-based DC recovery with errors
Step2
Enhance via DNN to fix recovered DC errors
Our approach
End-to-end predict DC by generative models

Fig. 1: Pipeline for DC recovery enhanced JPEG compression: existing methods (statistical-based DC recovery with errors need fixing) and our novel approach (i.e. a learned end-to-end method).

Various works [16]–[21] propose different approaches to tackle the challenge: *how to recover the DC coefficients more accurately from rest AC coefficients*. The fundamental idea is based on a statistical property: the differences between neighboring pixel values in natural images generally follow a Laplacian distribution [22]. Thus, based on this property, they recover one unknown DC coefficient from the known neighbor block to iteratively recover all DC coefficients. However, certain locations in natural images (e.g. less than 1% as pointed in [18]) deviate from this distribution, which causes an error when recovering the DC coefficient and will further propagate this error to limit the reconstructed image quality. Even though [19], [20] use deep neural networks (DNNs) to revise the errors after this iterative-based DC recovery, these two-step methods are still suboptimal since deviation-introduced errors are hard to fix.

In this paper, to avoid deviation-introduced errors, we propose `DCDiff` *to enhance JPEG via a learned end-to-end DC estimation*. Our basic idea is to avoid statistical recovery but directly leverage a pre-trained generative model (i.e., Stable Diffusion [23]) to utilize its knowledgeable priors for images to generate DC coefficients based on remaining AC coefficients. This is an end-to-end prediction process that can be optimized using learning methods with more data. Different from existing methods, `DCDiff` is not an iterative computation method block by block, so a deviation of image pixel values does not affect the whole image. However, as the generative model is not specifically designed for DC estimation, it faces challenges in predicting the content consistent average pixel value for each image block. To fill this gap, we propose a novel loss function to constrain the predicted image and the four corner blocks where the DC coefficients are retained to satisfy the Laplacian distribution while generating a mask to filter the image regions that deviate from the statistical property. What's more, to guide the diffusion model in generating the DC coefficients, we propose a frequency modulation sampling strategy to enhance low-frequency generation.

We compare DCDiff with 3 state-of-the-art (SOTA) baselines across 6 datasets considering 4 metrics. Extensive experiments prove that DCDiff significantly outperforms baselines across different metrics on all datasets such as improving the reconstructed image quality with $3 \sim 6.7$ dB more PSNR. DCDiff also has minimal impact on typical downstream tasks such as image classification and can be fully compatible with 2 typical low-cost devices.

II. Preliminaries

A. JPEG compression in IoT system

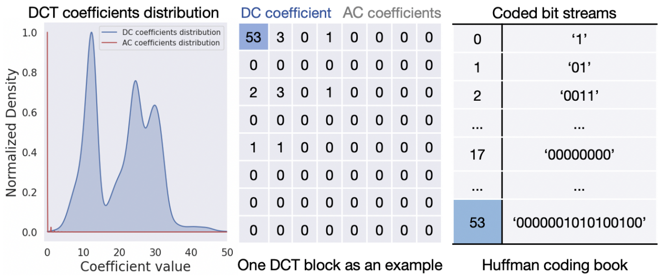
Due to the broad compatibility and low computational requirements, JPEG is irreplaceable in IoT scenarios, such as safety and security monitoring [24]–[26]. For example, [25] use JPEG for image compression in resilient smart cities, monitoring in real time, and transmitting images to the cloud for processing. Figure 1 shows the workflow of JPEG in IoT systems. The sender's end is often equipped with IoT devices with limited memory and computing capacity, such as surveillance, which can only perform simple mathematical transformations on captured images. The process of JPEG compression includes Discrete Cosine Transform (DCT), quantization according to a quantization table, which determines the quality of compressed image, and entropy coding, typically Huffman coding. The compressed JPEG files are transmitted to the receiver's end, which are often cloud servers with powerful computing resources, for different applications.

Enhanced JPEG compression methods are proposed to save transmission bandwidth. [27] build a JPEG encoder for high visual quality of compressed image with high compression ratio, but introduces significant computation and memory overhead, which is infeasible for resource-constrained devices. [16]–[21] follows the pipeline that drops the DC coefficients at the sender's end to save the transmission bandwidth (see Fig. 2, DC coefficients are generally larger) without any overhead and recover the image at the receiver's end. In this paper, we follow this setting of saving transmission bandwidth by dropping DC coefficients and focus on the post-processing stage at the receiver's end to recover higher-quality images.

B. DC coefficient estimation

DC estimation was first proposed in [22], which demonstrated that it is possible to recover DC coefficient from AC coefficient with reasonable image quality in JPEG encryption systems to provide higher protection for DC coefficient. This method is based on the Laplacian property that the difference between two neighboring pixels follows a Laplacian distribution with zero mean and small variance, as shown in Figure 4. The blue line "w/o mask" means the difference between adjacent pixels across the entire image. Denote $B(i,j)$ as the 8×8 blocks at the $i-th$ row and $j-th$ column of the image. DC value is the average of $B(i,j)$, which is the same for all pixels in $B(i,j)$. Assuming the AC value of $B(i,j)$ and the neighbor block $B(i,j-1)$ is known and DC value of $B(i,j-1)$, $dc_{i,j-1}$, is also known. Based on the Laplacian property, it is highly likely that the pixel values between two adjacent blocks are very similar. [22] minimizes the differences of pixels between two adjacent blocks from three directions to identified DC values.

Recently, some other works have applied DC estimation methods to enhance JPEG compression and further improve DC estimation accuracy. [18] improved the DC estimation from minimizing the value between adjacent pixels to minimizing the distribution trends of the last two columns and the first column of $B(i,j)$ and $B(i,j+1)$.



53	3	0	1	0	0	0	0
0	0	0	0	0	0	0	0
2	3	0	1	0	0	0	0
0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Coded bit streams	
0	'1'
1	'01'
2	'0011'
...	...
17	'00000000'
...	...
53	'0000001010100100'

Fig. 2: An example of AC and DC coefficients distribution and corresponding Huffman coding.

With the development of the neural network, [19] proposed a residual network to improve the image quality after calculating the DC coefficient using the methods proposed in [18]. Observing that the optimal prediction direction of each boundary pixel pair is inconsistent, [20] proposed a novel prediction pattern, which chooses pixel pairs from different directions, to improve the DC estimation.

Although the above methods use different pixel pair patterns to suppress the error propagation caused by the region deviated from Laplacian distribution, there are still some pixels that exhibit a significant difference compared to adjacent pixels in any three directions. Furthermore, the iterative process between blocks will amplify this error. To solve the above problems, we first analyze the characteristics of the region deviated from the Laplacian distribution. In addition, instead of using a block-by-block iterative approach, we directly predict the value of each pixel using a neural network, which can avoid the error propagation between adjacent pixels.

III. Method

A. Overview

In this section, we describe in detail our end-to-end post-processing method based on generative models. Figure 3 shows our major component. Compared with previous methods [19], DCDiff no longer relies on mathematical methods to iterate between DCT blocks to calculate the DC coefficients, but directly predicts the entire image using the diffusion model (section III-B). To constrain the reconstructed image to have content consistency with the origin image, DCDiff is trained using a masked Laplacian distribution loss to ensure the reconstructed images satisfy the Laplacian property while filtering the regions that may cause error (section III-C). Finally, to enhance the DC coefficients generated by the diffusion model, DCDiff uses a frequency modulation sampling strategy during the inference process, and a frequency modulation parameter predictor is applied to predict different scale factors adapting to different input images (section III-D).

B. DC coefficient feature extraction and generation

We denote the origin image as x_0, the reconstructed image as $\hat{x}$, and $\tilde{x}$ presents the image without DC coefficients after the receiver performs IDCT.

DC coefficients feature extraction. As illustrated in section I, DC coefficients represent the average of each image block, which means the DC coefficients have a much smaller feature space than the origin image. First, we introduce an image encoder (DC encoder) and an image decoder, denoted as $\mathcal{E}^{DC}$ and $\mathcal{D}$, to compress the image into a smaller latent space, which can present the DC component feature space, and the extracted feature denoted as z_0. To ensure that $\mathcal{E}^{DC}$ pays more attention to extracting the DC component of the image, we introduce another image encoder (AC encoder, $\mathcal{E}^{AC}$), taking $\tilde{x}$

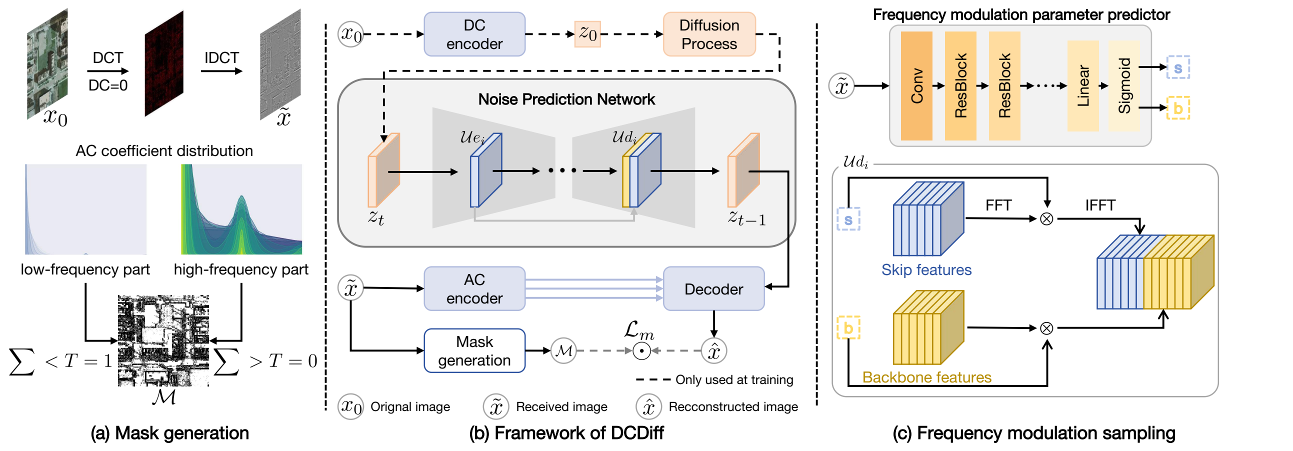


Fig. 3: Main components of `DCDiff`. (a) The insight to generate a spatial mask to apply Laplacian property (section III-C). (b) The pipeline of `DCDiff` to recover the JPEG images at the receiver's end (section III-B). (c) During inference, frequency modulation sampling is used to enhance the low-frequency generation (section III-D).

as input. $\mathcal{E}^{AC}$ can only extract the feature of AC coefficients, as the DC coefficients in $\tilde{x}$ have been set to zero at the sender's end. The image decoder $\mathcal{D}$ receives two parts of information, one of which is the AC features from the AC encoder, and the other is z_0 from the DC encoder. To better reconstruct the image, $\mathcal{D}$ needs both AC features and DC features, which forces the DC encoder to extract the features of DC coefficients from the origin image.

Diffusion generation. The diffusion model [28] is a generative model that can generate images from noise through multi-step denoising. It includes two processes of diffusion and denoising. During the diffusion process, the diffusion model gradually adds Gaussian noise to the image through a T-step Markov process until the image becomes complete Gaussian noise, following a variance schedule denoted by hyperparameter β_t (Eq. 1).

$$q\left(z_t|z_{t-1}\right) = \mathcal{N}\left(z_t; \sqrt{1-\beta_t}z_{t-1}, \beta_t\mathcal{I}\right) \tag{1}$$

The denoising process is the reverse of the diffusion process. Given z_t, a noise prediction network ϵ_θ is used to predict noise added to the image at t-th step to recover z_{t-1} (Eq. 2).

$$z_t = \sqrt{1-\beta_t}z_{t-1} + \beta_t\epsilon_t, \tag{2}$$

where ϵ_t is the noise predicted by the noise prediction network.

In this article, we aim to leverage the powerful generation ability and knowledgeable priors for images of the diffusion model to generate the DC coefficients of the images. Figure 3 (b) shows the framework of `DCDiff`. We use a noise prediction network to predict the DC features z_0 from Gaussian noise. Although the diffusion model demonstrates superior performance in generation tasks, it lacks the ability to preserve the structure information of the input images. Inspired by [29], we use a control module to extract the structure information from $\tilde{x}$ and inject it into the convolution block of diffusion model to improve structure consistency.

C. Masked Laplacian distribution loss

While the diffusion model with the control module demonstrates superior performance in image generation with given structure information, it lacks the ability to preserve the content consistency. For example, with no constraints, the generated image $\hat{x}$ may show a car that is yellow or blue—both appearing harmonious to the human but differing from x_0, mainly due to no explicit color information in $\tilde{x}$. To solve this problem, we propose the Laplacian distribution loss to constrain the reconstructed image to have content consistency with four corner blocks, whose DC coefficients are retained.

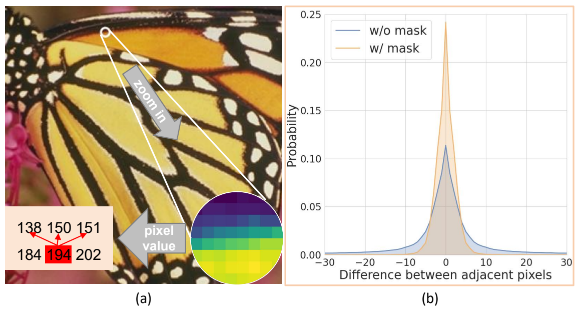


Fig. 4: (a) An example of pixel pairs have an abrupt change and significant differences across all three prediction directions. (b) comparison of the distribution of the difference between adjacent pixels with the high-frequency mask.

We first analyze the characteristics of pixel pairs that deviate from the Laplacian distribution. For example, as shown in Figure 4 (a), there are significant differences in pixel pairs across all three prediction directions. As a result, selecting any direction as the prediction direction will lead to substantial errors. We find that these pixel pairs are usually located in image regions with complex textures and sharp edges, where the pixel values exhibit significant fluctuations. The abrupt changes in pixel values in the spatial domain correspond to the increase in high-frequency components in the frequency domain. As illustrated in Figure 4 (b), we generate a mask to set all high-frequency components in the image to zero and recalculate the pixel value distribution of the low-frequency components in the image. Without the high-frequency components, the difference between two neighboring pixels follows the distribution with a much smaller variance. And the probability of pixel pairs with similar values is much higher than without the mask. Therefore, it is more appropriate to apply the Laplacian property to the low-frequency part of the image to calculate accurate pixel values.

Directly applying a low-pass filter in the frequency domain to filter out high frequencies may lead to changes in pixel values in the spatial domain, thereby affecting the accuracy of the prediction results. So we try to generate a spatial mask. When DC coefficients are zero, after IDCT, the pixel values of $\tilde{x}$ are the weighted sum of AC coefficients. As illustrated in Figure 3 (a), we show the AC coefficients of different parts of the images, the low-frequency part corresponding to the small pixel values part and the high-frequency part corresponding to the large pixel values part. Thus, the larger the value of $\tilde{x}$, the more high-frequency components correspond at this position. So we follow Equation 3 to generate the mask.

$$\mathcal{M}(i,j) = \begin{cases} 0, \tilde{x}(i,j) > \mathcal{T} \\ 1, \tilde{x}(i,j) \leqslant \mathcal{T} \end{cases} \tag{3}$$

where $\mathcal{T}$ is the threshold used to control the number of high-frequency components to be masked, and we discuss in detail the effect of different $\mathcal{T}$ on the results in section IV-C.

After obtaining the mask, we calculate the Laplacian constraint between pixels in the low-frequency region and define the masked Laplacian distribution (MLD) loss as Equation 4.

$$\mathcal{L}_m = \sum_{i,j} \mathcal{M}_{i,j} \odot ((\varDelta_h \hat{x}_{i,j} - \varDelta_h \hat{x}_{i,j-1})^2 + (\varDelta_w \hat{x}_{i,j} - \varDelta_w \hat{x}_{i-1,j})^2) \tag{4}$$

where $\varDelta_h \hat{x}_{i,j}$ denotes $\hat{x}_{i,j-1} - \hat{x}_{i,j}$.

D. Frequency modulation sampling strategy

As a generative model, the diffusion model tends to produce high-frequency detail information in images. However, in this paper, we aim to use the diffusion model to generate the DC coefficients, which represent the low-frequency features of the images. Inspired by [30], which investigates that the noise prediction network's skip connections mainly introduce high-frequency features, we try to modulate the frequency to enhance the low-frequency generation. [30] proposes to re-weight the backbone features and skip features during the denoising process using two manual scale factors. However, for image restoration, since the frequency information varies for each image, manually adjusting these two scale factors as hyperparameters is not suitable for every image.

To accommodate the frequency variations of different images, as shown in Figure 3 (c), we use a frequency modulation parameter predictor (FMPP), which relies on the ResNet architecture, to predict the scale parameters s and b. The predictor taking $\tilde{x}$ as input, estimates the scale factors to control the intensity of the enhancement of low-frequency information based on the high-frequency information contained in $\tilde{x}$. To control the magnitude of the enhancement, a sigmoid function is used in the final layer to regulate the range of the scale factor. Based on the experimental results in [30], we constrain the scale factor between 0 and 2. During the denoising process, FMPP initially takes $\tilde{x}$ as input to predict the scale factors and then DDIM is applied to denoise from noise, generating the DC coefficients of the images.

E. Training details

The training is divided into two stages. In the first stage, we train $\mathcal{E}^{DC}$, $\mathcal{E}^{AC}$ and $\mathcal{D}$ together using the loss function as Equation 5.

$$\mathcal{L}_{fir} = \mathcal{L}_{rec} + \mathcal{L}_{per} + \mathcal{L}_{dis}, \tag{5}$$

where $\mathcal{L}_{rec}$ is the $\mathcal{L}_1$ loss between the reconstructed image $\hat{x}$ and the origin image x_0, $\mathcal{L}_{per}$ is the perceptual loss and $\mathcal{L}_{dis}$ is the discriminator loss. In the second stage, we freeze the parameters of $\mathcal{E}^{DC}$, $\mathcal{E}^{AC}$ and $\mathcal{D}$, and train the noise prediction network with the goal of making the DC features $z_0\prime$ generated by the diffusion model as close as possible as z_0. Further, to ensure the generated image satisfies the Laplacian constraint, during each training step, we use the predicted noise ϵ_t to project from z_t to $z_0\prime$ following the Markov process: $z_0\prime = \frac{z_t - \sqrt{1-\bar{\alpha}_t}\epsilon_t}{\sqrt{\bar{\alpha}_t}}$, where $\bar{\alpha}_t = \prod_{i=1}^{t} \sqrt{1-\beta_i}$. Then use the $\mathcal{D}$ to project $z_0\prime$ to the pixel space and calculate the masked Laplacian distribution loss. The loss function used for training the noise prediction network is as Equation 6.

$$\mathcal{L}_{sec} = \mathcal{L}_{ldm} + \sigma \mathcal{L}_m, \tag{6}$$

where $\mathcal{L}_{ldm}$ is the MSE loss to minimize the noise distance between the target and learning noise in latent space, and σ is the weighting parameter. For FMPP, we freeze all other parameters and optimize FMPP using MSE loss, minimizing the distance between x_0 and $\hat{x}$ sampled under the scale factors predicted by FMPP.

IV. Experiment

A. Experiment setup

Dataset. For training data, we randomly select 300K images from OpenImages [31] and randomly crop the image to a resolution of 256 × 256 pixels. For testing data, we use 5 general CV datasets and one dedicated remote sensing dataset [32].

Metrics. We follow [33] to use 4 metrics for quantitative measures. We use the Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) to validate the pixel fidelity and the structure similarity. What's more, we use another two perceptual metrics, Multi-Scale Structural Similarity Index Measure (MS-SSIM) and Learned Perceptual Image Patch Similarity (LPIPS) to evaluate the quality of reconstructed images for human perception.

Implementation details. For the first training stage, we train the DC encoder $\mathcal{E}^{DC}$, the AC encoder $\mathcal{E}^{AC}$ and the decoder $\mathcal{D}$ together using the loss function in Equation 5 for 5 epochs with a batchsize of 16 and a learning rate of $1e-4$. For the second training stage, we use the pre-trained stable diffusion [23] model as noise prediction network and finetune it using $\mathcal{L}_{ldm}$ for 5 epochs with a learning rate of $1e-4$ to learn how to retain structural information in $\tilde{x}$ first. And then we add $\mathcal{L}_m$ to loss function, and finetune the model for 10 epochs with a learning rate of $1e-5$ and $\sigma = 2e-4$ to ensure generated images satisfy Laplacian constraint. All training experiments are performed on a server with 8× Nvidia H800 GPUs using the Adam optimizer. During inference, we utilize the DDIM sampling [34] with 50 steps. For baseline comparison, we consider three baselines for DC estimation, including methods based on mathematical calculations [18], [20] and based on neural networks [19].

B. Evaluation results

Qualitative comparisons. Table I shows the comparison results with baselines on six datasets. We compress the RGB images in each dataset with the standard Q_{50} quantization table following the previous works. Compared to compressing grayscale images, during JPEG compression, the discarded DC coefficients of the U and V channels retrain the chrominance information of the image. The error propagation caused by incorrect DC coefficients for U and V channels may greatly affect the color of the final reconstructed image, resulting in lower PSNR and SSIM for SmartCom 2019 [18] and ICIP 2022 [20]. Note that due to AC coefficients retaining high-frequency details from the origin image, the detailed information of the image is not lost during the discarding of DC coefficients, resulting in a relatively low LPIPS. Therefore, compared to directly using a low-quality Q quantization table to reduce the amount of transmission data, discarding the DC coefficients can better preserve image detail without affecting human perception. IEEE TII 2021 [19] tries to use a residual neural network to correct error propagation. However, optimizing solely for mean square error (MSE) can lead to excessive image smoothing, corrupting the detail of high-frequency information in the image, resulting in visual artifacts. As shown in Table I, the LPIPS of IEEE TII 2021 is much higher than others. In contrast,

TABLE I: Quantitative results of comparison DCDiff with 3 baselines on 6 datasets. Throughout this paper, ↑ (↓) means higher (lower) is better. Best performances are highlighted in **bold**.

Methods	Set5				Set14				Kodak			
	PSNR↑	SSIM↑	MS-SSIM↑	LPIPS↓	PSNR↑	SSIM↑	MS-SSIM↑	LPIPS↓	PSNR↑	SSIM↑	MS-SSIM↑	LPIPS↓
SmartCom 2019 [18]	21.70	0.8557	0.9501	0.0396	21.34	0.8804	0.9224	0.0508	22.09	0.9132	0.8861	0.0533
IEEE TII 2021 [19]	22.99	0.8819	0.9511	0.0408	21.40	0.8967	0.9203	0.0582	21.87	0.9101	0.8809	0.0624
ICIP 2022 [20]	23.36	0.8764	0.9585	0.0354	22.04	0.8954	0.939	0.0436	23.47	0.9296	0.9134	0.0437
DCDiff	**28.46**	**0.9494**	**0.9652**	**0.0254**	**25.84**	**0.9498**	**0.9490**	**0.0280**	**26.96**	**0.9544**	**0.9358**	**0.0309**

Method	BSDS200				Urban100				Inria			
	PSNR↑	SSIM↑	MS-SSIM↑	LPIPS↓	PSNR↑	SSIM↑	MS-SSIM↑	LPIPS↓	PSNR↑	SSIM↑	MS-SSIM↑	LPIPS↓
SmartCom 2019 [18]	21.68	0.8981	0.9049	0.0502	19.34	0.8681	0.8759	0.0517	21.15	0.9207	0.9066	0.0497
IEEE TII 2021 [19]	21.85	0.9030	0.9019	0.0579	19.62	0.8782	0.8737	0.0585	20.96	0.9116	0.9001	0.0623
ICIP 2022 [20]	22.76	0.9128	0.9264	0.0429	20.67	0.8902	0.9050	0.0416	22.53	0.9381	0.9290	0.0421
DCDiff	**25.76**	**0.9481**	**0.9436**	**0.0314**	**26.01**	**0.9553**	**0.9532**	**0.0187**	**27.94**	**0.9556**	**0.9457**	**0.0308**

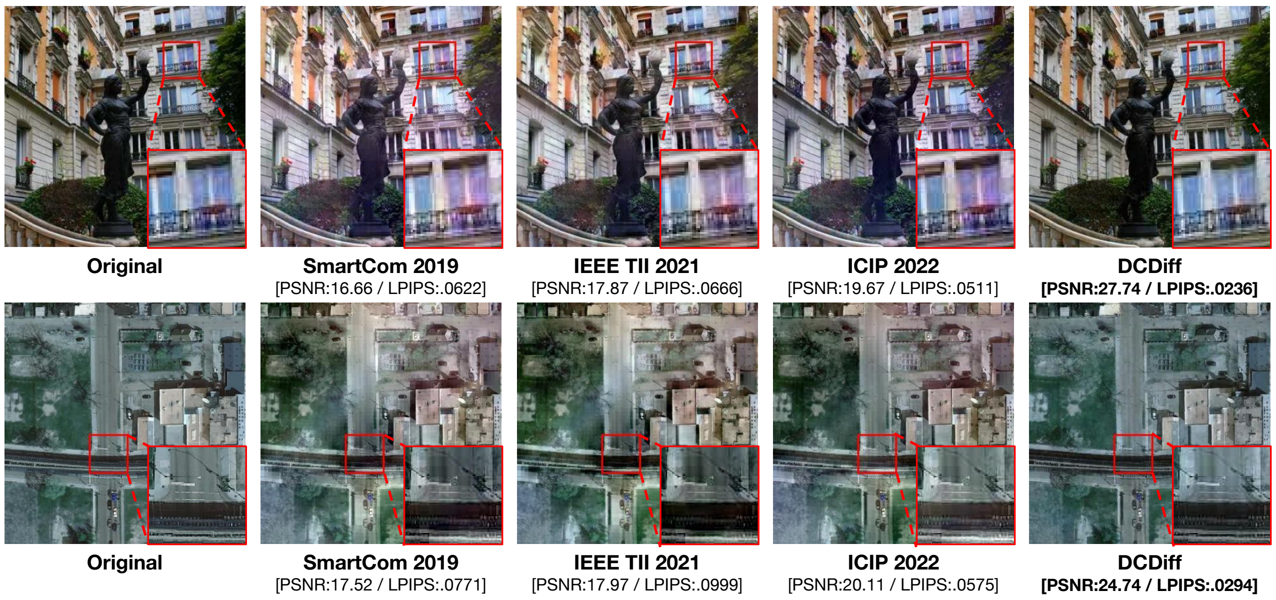

Original
SmartCom 2019
[PSNR:16.66 / LPIPS:.0622]
IEEE TII 2021
[PSNR:17.87 / LPIPS:.0666]
ICIP 2022
[PSNR:19.67 / LPIPS:.0511]
DCDiff
[PSNR:27.74 / LPIPS:.0236]
Original
SmartCom 2019
[PSNR:17.52 / LPIPS:.0771]
IEEE TII 2021
[PSNR:17.97 / LPIPS:.0999]
ICIP 2022
[PSNR:20.11 / LPIPS:.0575]
DCDiff
[PSNR:24.74 / LPIPS:.0294]

Fig. 5: The reconstructed image after post-processing.

DCDiff is not a block-based iterative method, which avoids error propagation. Additionally, it constrains the generated images using a mask Laplacian distribution loss, preventing excessive smoothing. The results demonstrate that DCDiff can achieve SOTA performance across all metrics on all datasets, increasing PSNR by $3 \sim 6.7dB$.

Visual results. Figure 5 shows the reconstructed images after post processing of DCDiff. We selected a street-view image and an aerial image for visualization comparison. As analyzed above, error propagation in U and V channels may lead to unnatural colors, such as the purple windows in the first row of Figure 5. Furthermore, the baseline's optimization objective is to enforce Laplacian constraints as closely as possible between any two adjacent blocks, which leads to color bleeding in areas with abrupt pixel value changes. For example, at the intersection in the image of the second row, the vertical road is dyed black. Since we use a mask to filter out high-frequency parts of the image and do not rely on block-based iteration, there is no color bleeding or unnatural colors due to error propagation, achieving SOTA reconstruction results.

Compression Ratio. At the sender's end, we follow the setting in [19] to set all DC coefficients to zero except 4 corner blocks, and calculate the compression ratio compared with standard JPEG using standard JPEG coding. As shown in Table II, we first use the same Q-table (i.e. Q_{50} quantization table) for standard JPEG and DCDiff. On all datasets, just by dropping DC coefficients, DCDiff can compress 25% more than JPEG on average. Besides, we also compare DCDiff with more compression of standard JPEG when

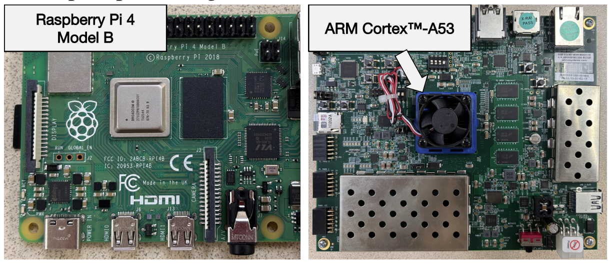


Fig. 6: Two low-cost testbeds for evaluating DCDiff.

using a more aggressive Q-table. We tune the Q-table of JPEG until the reconstructed images achieve a similar image quality (i.e. LPIPS) after DCDiff post-processing. While maintaining the same image quality reconstructed at the receiver's end, DCDiff can still compress 10-14% more than tuning Q-table in default JPEG.

C. Ablation study

W/o MLD loss. The masked Laplacian distribution loss can better guide the image reconstruction based on DC coefficients of four corner blocks and AC coefficients. We remove the MLD loss and train the noise prediction network until converges. As shown in Table III, the results show that without MLD loss, the model cannot capture the relation between four corner blocks and the other part of the images, leading to noticeable color distortion in reconstructed images.

W/o FMPP. To show that the frequency modulation can enhance the low-frequency information, we remove the frequency modulation

TABLE II: Compression ratios of JPEG with zero DC coefficients except for 4 corner blocks.

Dataset	Set5	Set14	Kodak	BSDS200	Urban100	Inria
	Compression ratio of JPEG (the same Q-table, Q_{50})					
min	68.66%	68.37%	63.53%	59.96%	63.89%	43.71%
max	81.11%	90.19%	83.39%	86.26%	93.96%	86.34%
avg	74.73%	78.64%	73.41%	75.99%	81.38%	78.89%
	Compression ratio of JPEG under similar LPIPS					
min	77.84%	77.65%	75.16%	66.38%	68.62%	46.62%
max	96.00%	97.70%	98.76%	98.30%	98.14%	95.24%
avg	86.98%	89.16%	89.10%	86.45%	88.50%	86.46%

TABLE III: Ablation study of DCDiff's variants. "w/o MLD" indicates removing masked Laplacian distribution loss at the 2nd stage training. "w/o FMPP" denotes that during sampling the predictor is removed. Different thresholds are evaluated when generating masks.

Dataset	Methods	PSNR↑	SSIM↑	MS-SSIM↑	LPIPS↓
Kodak	w/o MLD	24.80	0.9425	0.9238	0.0380
	w/o FMPP	26.57	0.9544	0.9341	0.0321
	$\mathcal{T} = 0$	26.04	0.9530	0.9325	0.0324
	$\mathcal{T} = 5$	26.11	0.9539	0.9341	0.0322
	$\mathcal{T} = 10$	26.96	0.9544	0.9358	0.0309
	$\mathcal{T} = 15$	26.17	0.9487	0.9307	0.0347
Inria	w/o MLD	26.35	0.9511	0.9409	0.0356
	w/o FMPP	27.46	0.9553	0.9445	0.0319
	$\mathcal{T} = 0$	27.29	0.9553	0.9449	0.0322
	$\mathcal{T} = 5$	27.59	0.9555	0.9456	0.0316
	$\mathcal{T} = 10$	27.94	0.9556	0.9457	0.0308
	$\mathcal{T} = 15$	27.17	0.9497	0.9412	0.0349

parameter predictor during sampling. The scale factors s and b are equal to 1, which is the same as regular DDIM sampling. The results in Table III show that frequency modulation can help to enhance the backbone features and improve the quality of reconstructed images.

Influence of mask threshold. We further investigate the impact of the threshold used to generate the mask $\mathcal{M}$ on the reconstructed image quality as shown in Table III. A smaller threshold results in most of the image content being masked, leaving only a small portion of the image associated with the information from the four corner blocks, which causes the model to rely on diffusion to generate most of the unconstrained content. A larger threshold causes some high-frequency information in the image, such as object edges, to be included to calculate the loss, which results in blurred edges around objects in the image, creating visual artifacts. Based on the results in Table III, we select $\mathcal{T} = 10$ as the threshold to generate the mask, which can capture most of the image information while filtering out high-frequency content such as object edges.

D. Use case analysis in IoT systems

We deploy DCDiff on 2 widely-used low-power processors: ARM Cortex™-A53[1] and Raspberry Pi 4 Model B[2] (as shown in Fig. 6) for evaluation. On these 2 processors, we compress and transmit captured images using standard JPEG and DCDiff and calculate the throughput. The results in Table IV show that compared to standard JPEG, DCDiff does not add extra computational overhead, which proves that DCDiff can be easily adapted to low-cost front-end cameras with JPEG support.

E. Influence on downstream tasks

Images reconstructed at the receivers' ends are usually used for further processing. To confirm that DCDiff has a tiny influence on the downstream tasks, we choose a remote-sensing image classification task [35] as an example. For all baselines, we use the same setting to drop DC coefficients at the sender's end and input the reconstructed images after post-processing to the image classification model at the receiver's end. As shown in Table V, DCDiff outperforms all baselines with a tiny impact on model accuracy (drop 0.49%).

[1]https://www.arm.com/products/silicon-ip-cpu/cortex-a/cortex-a53

[2]https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

TABLE IV: Deploying DCDiff on 2 low-power front devices.

Methods	Compression throughput (Gbps)	
	Raspberry Pi 4	ARM Cortex™-A53
JPEG Encoder	1.85	0.92
DCDiff Encoder	1.95	0.92

TABLE V: Post-processing influence on a remote-sensing image classification tasks. ↓ denotes the accuracy reduction.

Methods	Original	SmartCom 2019 [18]	IEEE TII 2021 [19]	ICIP 2022 [20]	DCDiff
ACC (%)	97.54	↓ 1.96%	↓ 4.42	↓ 1.47	↓ **0.49**

V. Discussion and Future Work

JPEG v.s. Learned image compression. Transformation-based image compression method mainly applies a spatial-to-frequency transformation (e.g. DCT, DWT) to get the frequency coefficients and then uses a tunable quantization step to compress them. Today, they are outperformed by novel learned compression methods [8], [36]–[38] in various scenarios. Those learned compression methods generally replace the transformation-based compressor with a DNN-based encoder to extract image features and use a decoder to reconstruct images. Since DNNs can be trained to fit data in a similar distribution, learned image compression methods can outperform static ones (e.g. JPEG/JPEG2000) with sufficient data to train powerful encoders [39], [40]. However, there are still a large number of cameras that do not have enough resources to support a DNN-based encoder. On the contrary, JPEG, as the most widely-used image compression method, is well implemented on all existing cameras such that enhanced JPEG methods like DCDiff can also be easily adopted in real-world scenarios to improve image compression.

More improvements in future. First, in this paper, we use the default coding technique of JPEG for evaluating DCDiff. We are aware that, as a step after transformation and quantization, better coding techniques [41], [42] may get better results. Thus, it is important to emphasize that DCDiff studies how to reduce the number of DCT coefficients which is orthogonal to coding techniques. Second, DCDiff uses the diffusion model to reconstruct the images but it can be replaced as any other generative models as long as they can be trained to get rid of deviation-induced errors.

VI. Conclusion

We present DCDiff, an end-to-end DC estimation method for enhanced JPEG compression based on generative models. We propose the MLD loss to suppress error propagation while improving the content consistency of reconstructed images and a frequency modulation sampling strategy to enhance DC coefficient generation. Extensive results indicate that DCDiff can achieve SOTA performance to recover high-quality images and is compatible with various IoT devices to save transmission bandwidth while maintaining high throughput.

Acknowledgment

This work is supported by NSFC under grant No. U24A20337 and 62301189, and also the Center of High performance computing, Tsinghua University, China.

References

[1] M. Imani, A. Zakeri, H. Chen, T. Kim, P. Poduval, H. Lee, Y. Kim, E. Sadredini, and F. Imani, "Neural computation for robust and holographic face detection," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, 2022, pp. 31–36.

[2] S. Xia, T. Xing, C. Q. Wu, G. Liu, J. Yang, and K. Li, "AQMon: A Fine-grained Air Quality Monitoring System Based on UAV Images for Smart Cities," *ACM Transactions on Sensor Networks*, vol. 20, no. 2, pp. 1–20, 2024.

[3] S. Liu, B. Yu, J. Tang, and Q. Zhu, "Towards fully intelligent transportation through infrastructure-vehicle cooperative autonomous driving: Challenges and opportunities," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2021, pp. 1323–1326.

[4] C. Jia, M. Hu, Z. Chen, Y. Yang, X. Xie, Y. Liu, and M. Chen, "AdaptiveFL: Adaptive heterogeneous federated learning for resource-constrained AIoT systems," in *Proceedings of the 61st ACM/IEEE Design Automation Conference*, 2024, pp. 1–6.

[5] R. M. Asiyabi, A. Anghel, A. Focsa, M. Datcu, M. Martone, P. Rizzoli, and E. Imbembo, "On the use of JPEG2000 for SAR raw data compression," in *EUSAR 2024; 15th European Conference on Synthetic Aperture Radar*. VDE, 2024, pp. 249–253.

[6] J. Bankoski, P. Wilkins, and Y. Xu, "Technical overview of vp8, an open source video codec for the web," in *2011 IEEE International Conference on Multimedia and Expo*. IEEE, 2011, pp. 1–6.

[7] Y. Huang, B. Chen, J. Zhang, Q. Han, and S.-T. Xia, "Compressive sensing based asymmetric semantic image compression for resource-constrained iot system," in *Proceedings of the 59th ACM/IEEE Design Automation Conference (DAC)*, 2022, pp. 877–882.

[8] R. Yang and S. Mandt, "Lossy image compression with conditional diffusion models," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[9] G. K. Wallace, "The jpeg still picture compression standard," *Communications of the ACM*, vol. 34, no. 4, pp. 30–44, 1991.

[10] R. Salikhov, V. K. Abdrakhmanov, and I. Safargalin, "Internet of things (iot) security alarms on esp32-cam," in *Journal of Physics: Conference Series*, vol. 2096, no. 1. IOP Publishing, 2021, p. 012109.

[11] K. Elhattab, K. Abouelmehdi, and S. Elatar, "New model to monitor plant growth remotely using esp32-cam and mobile application," in *2023 10th International Conference on Wireless Networks and Mobile Communications (WINCOM)*. IEEE, 2023, pp. 1–6.

[12] T. Jaya, S. Nath *et al.*, "Surveillance monitoring using esp32-cam module for fog detection in agriculture." *Grenze International Journal of Engineering & Technology (GIJET)*, vol. 10, no. 1, 2024.

[13] https://github.com/espressif/esp32-camera.

[14] X. Luo, H. Talebi, F. Yang, M. Elad, and P. Milanfar, "The rate-distortion-accuracy tradeoff: JPEG case study," *arXiv preprint arXiv:2008.00605*, 2020.

[15] Y. Peng, C. Fu, G. Cao, W. Song, J. Chen, and C.-W. Sham, "JPEG-compatible joint image compression and encryption algorithm with file size preservation," *ACM Transactions on Multimedia Computing, Communications and Applications*, vol. 20, no. 4, pp. 1–20, 2024.

[16] S. Li, J. J. Ahmad, D. Saupe, and C.-C. J. Kuo, "An improved DC recovery method from AC coefficients of DCT-transformed images," in *2010 IEEE International Conference on Image Processing*. IEEE, 2010, pp. 2085–2088.

[17] S. Ong, S. Li, K. Wong, and K. Tan, "Fast recovery of unknown coefficients in DCT-transformed images," *Signal Processing: Image Communication*, vol. 58, pp. 1–13, 2017.

[18] H. Qiu, Q. Zheng, M. Qiu, and G. Memmi, "Dc coefficients recovery from ac coefficients in the jpeg compression scenario," in *Smart Computing and Communication: 4th International Conference, SmartCom 2019, Birmingham, UK, October 11–13, 2019, Proceedings 4*. Springer, 2019, pp. 266–276.

[19] H. Qiu, Q. Zheng, G. Memmi, J. Lu, M. Qiu, and B. Thuraisingham, "Deep residual learning-based enhanced jpeg compression in the internet of things," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 3, pp. 2124–2133, 2020.

[20] J. Zhang, B. Chen, Y. Huang, H. Qiu, Z. Wang, and S. Xia, "Improved dc estimation for jpeg compression via convex relaxation," in *2022 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2022, pp. 2461–2465.

[21] M. Ouyang and Z. Chen, "JPEG Quantized Coefficient Recovery via DCT Domain Spatial-Frequential Transformer," *IEEE Transactions on Image Processing*, 2024.

[22] T. Uehara, R. Safavi-Naini, and P. Ogunbona, "Recovering DC coefficients in block-based DCT," *IEEE Transactions on Image Processing*, vol. 15, no. 11, pp. 3592–3596, 2006.

[23] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 684–10 695.

[24] A. Sharma, P. K. Singh, and Y. Kumar, "An integrated fire detection system using iot and image processing technique for smart cities," *Sustainable Cities and Society*, vol. 61, p. 102332, 2020.

[25] S. A. Shah, D. Z. Seker, M. M. Rathore, S. Hameed, S. B. Yahia, and D. Draheim, "Towards disaster resilient smart cities: Can internet of things and big data analytics be the game changers?" *IEEE Access*, vol. 7, pp. 91 885–91 903, 2019.

[26] F. Saeed, A. Paul, A. Rehman, W. H. Hong, and H. Seo, "Iot-based intelligent modeling of smart home environment for fire prevention and safety," *Journal of Sensor and Actuator Networks*, vol. 7, no. 1, p. 11, 2018.

[27] https://github.com/google/guetzli.

[28] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[29] L. Zhang, A. Rao, and M. Agrawala, "Adding conditional control to text-to-image diffusion models," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 3836–3847.

[30] C. Si, Z. Huang, Y. Jiang, and Z. Liu, "Freeu: Free lunch in diffusion u-net," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 4733–4743.

[31] I. Krasin, T. Duerig, N. Alldrin, V. Ferrari, S. Abu-El-Haija, A. Kuznetsova, H. Rom, J. Uijlings, S. Popov, A. Veit *et al.*, "Openimages: A public dataset for large-scale multi-label and multi-class image classification," *Dataset available from https://github. com/openimages*, vol. 2, no. 3, p. 18, 2017.

[32] E. Maggiori, Y. Tarabalka, G. Charpiat, and P. Alliez, "Can semantic labeling methods generalize to any city? the inria aerial image labeling benchmark," in *2017 IEEE International geoscience and remote sensing symposium (IGARSS)*. IEEE, 2017, pp. 3226–3229.

[33] Z. Wang, D. Li, G. Li, Z. Zhang, and R. Jiang, "Multimodal low-light image enhancement with depth information," in *Proceedings of the 32nd ACM International Conference on Multimedia*, 2024, pp. 4976–4985.

[34] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," *arXiv preprint arXiv:2010.02502*, 2020.

[35] A. Bahri, S. G. Majelan, S. Mohammadi, M. Noori, and K. Mohammadi, "Remote sensing image classification via improved cross-entropy loss and transfer learning strategy based on deep convolutional neural networks," *IEEE Geoscience and Remote Sensing Letters*, vol. 17, no. 6, pp. 1087–1091, 2019.

[36] Z. Liu, T. Liu, W. Wen, L. Jiang, J. Xu, Y. Wang, and G. Quan, "DeepN-JPEG: A deep neural network favorable JPEG-based image compression framework," in *Proceedings of the 55th annual design automation conference*, 2018, pp. 1–6.

[37] F. Mentzer, G. D. Toderici, M. Tschannen, and E. Agustsson, "High-fidelity generative image compression," *Advances in Neural Information Processing Systems*, vol. 33, pp. 11 913–11 924, 2020.

[38] M. Li, S. Gao, Y. Feng, Y. Shi, and J. Wang, "Content-oriented learned image compression," in *European Conference on Computer Vision*. Springer, 2022, pp. 632–647.

[39] Y. Yang and S. Mandt, "Computationally-efficient neural image compression with shallow decoders," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 530–540.

[40] J. Liu, H. Sun, and J. Katto, "Learned image compression with mixed transformer-cnn architectures," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 14 388–14 397.

[41] Z. Hu, Z. Chen, D. Xu, G. Lu, W. Ouyang, and S. Gu, "Improving deep video compression by resolution-adaptive flow coding," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*. Springer, 2020, pp. 193–209.

[42] J. Ballé, P. A. Chou, D. Minnen, S. Singh, N. Johnston, E. Agustsson, S. J. Hwang, and G. Toderici, "Nonlinear transform coding," *IEEE Journal of Selected Topics in Signal Processing*, vol. 15, no. 2, pp. 339–353, 2020.