A Lightweight Denoising Convolutional Neural Network for On-Device Artifact Suppression

Naeem Ahmed
Department of Electronics and Communication Engineering, Dayananda Sagar University, Bangalore, India
naeemahmed.res-soe-ece@dsu.edu.in (corresponding author)

R. Navya
Department of Electronics and Communication Engineering, Dayananda Sagar University, Bangalore, India
navya-ece@dsu.edu.in

Arun Ananthanarayanan
Department of Electronics and Communication Engineering, Dayananda Sagar University, Bangalore, India
arunananthanarayan-ece@dsu.edu.in

Received: 9 October 2025 | Revised: 31 October 2025 and 19 November 2025 | Accepted: 29 November 2025

 DOI: https://doi.org/10.48084/etasr.15428

ABSTRACT

Image compression for mobile and streaming applications often introduces blocking, blurring, and ringing that degrade visual quality and harm downstream vision tasks. This work presents a lightweight on-device restoration model based on a Denoising Convolutional Neural Network (DnCNN) that is optimized for efficiency using structured pruning, 8-bit integer (INT8) quantization, and architectural slimming, followed by perceptual fine-tuning in MATLAB. The model was trained on the Berkeley Segmentation Dataset 400 (BSD400) and evaluated on Set5, Set14, and Berkeley Segmentation Dataset 68 (BSD68). We report standard full-reference metrics, namely Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM), a perceptual metric, Learned Perceptual Image Patch Similarity (LPIPS); and no-reference metrics, Natural Image Quality Evaluator (NIQE) and Blind/Referenceless Image Spatial Quality Evaluator (BRISQUE). On average, the compressed model attains about 29.0 dB PSNR and 0.83 SSIM, while reducing model size by about 52% to 1.0 MB and cutting CPU inference time by about 70% compared with the uncompressed DnCNN baseline. These results show that the compressed and perceptually fine-tuned DnCNN suppresses compression artifacts effectively while meeting the memory and latency constraints of mobile and embedded platforms, providing a practical receiver-side solution that remains compatible with legacy codecs.

Keywords-image compression; artifact suppression; PSNR; pruning; quantization; DnCNN; MATLAB

I. INTRODUCTION

Mobile image streaming and real-time delivery continue to grow, increasing bandwidth pressure and motivating aggressive compression at the transmitter. Such compression commonly produces blocking, ringing, and blurring artifacts that reduce perceived visual quality and can degrade downstream computer vision tasks such as detection and segmentation [1]. In many imaging pipelines, transmitter-side changes are impractical, so improving restoration at the receiver is the most compatible path because it preserves interoperability with existing codecs.

Convolutional Neural Networks (CNNs) are a strong receiver-side choice for artifact suppression and consistently outperform classical filters on standard benchmarks. Modern residual denoisers and attention mechanisms further improve detail preservation under aggressive compression [2]. Methods that combine pixel- and frequency-domain priors, often called dual-domain networks, show clear gains under severe compression [3. Perceptual and adversarial training can increase subjective quality compared with Mean Squared Error (MSE) training, although they may raise model complexity and risk hallucinated details, which creates a fidelity–perception trade-off that requires careful evaluation [4]. An alternative is to modify the transmission chain using Deep Joint Source-Channel Coding (DeepJSCC), which can outperform separate codec-plus-post-processing pipelines when transmitter changes are allowed [5]. However, transmitter-side changes are often impractical for deployed systems, therefore, receiver-side

restoration that interoperates with legacy codecs remains important, which motivates our hardware-aware, DnCNN-based approach described below [6]. The Denoising Convolutional Neural Network (DnCNN) was selected as our base because its residual formulation is simple, effective for compression artifact removal, and amenable to structured pruning and 8-bit integer (INT8) quantization for mobile deployment. In this work, the residual formulation is followed, where a clean image and its compressed observation are related by the degradation model summarized in (1), and restoration proceeds as in (2). To make DnCNN practical on devices while controlling perceptual effects, three objectives were pursued:

- Compact hardware-aware model: A DnCNN backbone into a channel-slimmed, structured-pruned, and INT8-quantized variant, suitable for CPU inference on mobile and embedded devices, targeting about 1.0 MB model size and low per-image latency (see Figure 2) was adapted.
- Perceptual fine-tuning: A short fine-tuning stage with VGG-16 feature supervision and Learned Perceptual Image Patch Similarity (LPIPS) guidance was used to recover subjective fidelity lost to pruning and quantization while limiting hallucination.
- Reproducible evaluation and deployment: A MATLAB-based workflow and comprehensive evaluation using Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM), LPIPS, Natural Image Quality Evaluator (NIQE), and Blind/Referenceless Image Spatial Quality Evaluator (BRISQUE), is provided with ablations and measured runtime and memory to demonstrate deployment feasibility (Table I, Figures 3–5).

Although several high-quality restoration networks, dual-domain approaches, and perceptual training methods have been explored [7], only a few studies have systematically combined structured pruning and INT8 quantization within a compact residual denoiser, validated light perceptual fine-tuning after compression, and provided reproducible MATLAB-based deployment data [8].

II. PROPOSED METHODOLOGY

We modeled degradation as JPEG compression plus AWGN, used a compressed DnCNN backbone, and applied a three-stage compression workflow: structured pruning, 8-bit quantization, and architectural slimming, followed by perceptual fine-tuning to restore visual quality.

A. Problem Formulation and Degradation Model

Let $x \in R^{H\times W\times C}$ denote a clean image that is compressed by a lossy codec (JPEG) at quality Q and optionally corrupted by channel noise n. The received image y is modeled as:

$$y = C_Q(x) + n \quad (1)$$

where $C_Q(.)$represents the codec plus quantization operator at quality Q, and $n \sim \mathcal{N}(0, \sigma^2 I)$ represents additive white Gaussian noise (AWGN). In our experiments we use $Q \in \{10,20,30,50\}$ and $\sigma \in \{5,10,20\}$ in pixel units $(0-255)$; equivalently, in the normalized [0,1] range these correspond to $\sigma \in \{5/255, 10/255, 20/255\}$. As shown in (1), the observation y combines codec distortion and channel noise. The receiver seeks an estimate $\hat{x} = f_\theta(y)$ that removes compression artifacts while remaining computationally feasible on mobile devices. This inverse mapping is ill-posed; therefore, we learn the θ parameters by minimizing a task loss under hardware-aware constraints [9].

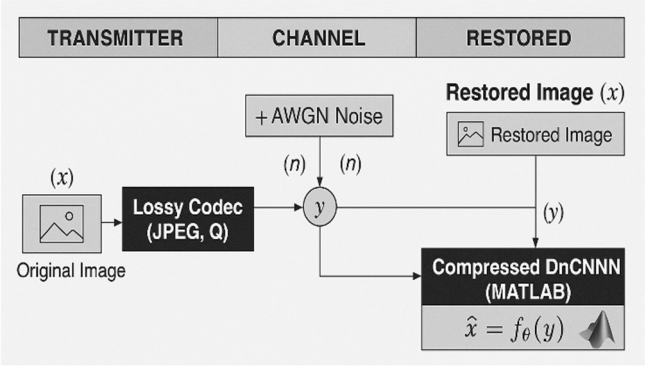

Fig. 1. System pipeline for image transmission and restoration. The transmitter applies JPEG compression at quality factor Q, the channel introduces AWGN, and the receiver employs the compressed DnCNN for artifact suppression, as formulated in (1).

B. Compressed DnCNN Backbone

We adopt the DnCNN residual formulation where the network predicts the corruption residual $r_\theta(y)$ and recovers $\hat{x}$ by subtraction:

$$r_\theta(y) \approx y - x, \quad \hat{x} = y - r_\theta(y) \quad (2)$$

We use the DnCNN-17 layout for luminance experiments with the following parameters: 17 convolutional layers, 64 channels for intermediate layers, 3×3 kernels, stride 1, and zero padding. For color experiments we use a 3-channel variant by adjusting the first/last conv layers. This simple residual form is amenable to structured pruning and quantization while preserving known baselines [10].

Model complexity: The uncompressed DnCNN-17 (64 channels) has ~0.56 million parameters (≈2.1 MB). After structured pruning (~40% channel reduction) and light slimming, the model has ~0.33 million parameters (≈1.0 MB).

C. Compression Pipeline: Pruning, Quantization, and Slimming

To satisfy mobile constraints, we adopted a three-stage compression pipeline, as illustrated in Figure 2. The stages are: (i) structured filter pruning, (ii) 8-bit quantization, and (iii) light architectural slimming. Each step is followed by fine-tuning to recover accuracy, and the hyperparameters are chosen in line with recent hardware-aware studies. MATLAB's dlquantizer and associated workflows are employed for calibration and INT8 model export [11].

- Structured filter pruning: Each convolutional filter is assigned an importance score using the L1-norm criterion, $s_j = \|F_j\|_1$, where F_j denotes the kernel weights. Filters with the lowest scores are iteratively pruned and fine-tuned to maintain performance. This stage achieves

approximately 40% reduction in channels, excluding the first and last layers.

- 8-bit quantization: Quantization-aware training is applied to map weights and activations into integer forms. In practice, we quantize the pruned model to INT8 precision, fine-tune it for stability, and export it via MATLAB's quantization workflows [11].
- Architectural slimming: After pruning and quantization, we further reduced complexity by narrowing intermediate channels (e.g., 64 → 48) and, in ablation experiments, by replacing selected 3×3 layers with depthwise-separable variants. This step is inspired by mobile-oriented design patterns, such as MobileNets and other lightweight restoration recipes [12].

Figure 2 demonstrates the sequential flow of this compression strategy: starting from pruning, followed by quantization, and finally slimming, after which the fine-tuned lightweight DnCNN is obtained.

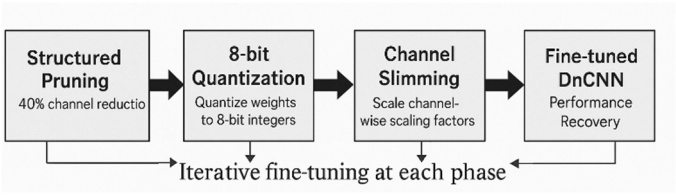

Fig. 2. Compression pipeline for the lightweight DnCNN.

D. Perceptual Fine-Tuning: Loss Functions

Training uses a two-stage loss. Stage 1 minimizes pixel fidelity with the MSE:

$$L_{MSE} = \frac{1}{N}\sum_i \| x_i - \hat{x}_i \|_2^2 \quad (3)$$

where $\hat{x}_i$ is the restored image from (2). As shown in (3), this stage encourages pixel-wise accuracy. Stage 2 fine-tunes the compressed model with a combined loss as in (4), where the perceptual term L_{perc} is defined in (5):

$$L = L_{MSE} + \lambda L_{perc} \quad (4)$$

where the perceptual component is

$$L_{perc} = \| \emptyset(x) - \emptyset(\hat{x}) \|_2^2 \quad (5)$$

where $\emptyset(.)$ denotes feature maps extracted from the Visual Geometry Group 16-layer network (VGG-16), λ is chosen empirically in the range 0.05–0.20, and Learned Perceptual Image Patch Similarity (LPIPS) is monitored as a validation proxy. We follow the common perceptual feature-loss formulation used in prior work [4].

E. Datasets, Degradations, and Pre-Processing

1) Datasets and Splits

All experiments were performed using MATLAB R2023b. Training used the BSD400 dataset containing 400 natural images [13]. Validation used a fixed set of 100 images sampled with a constant seed to ensure reproducibility and disjoint was kept from training. Testing was conducted on the standard Set5, Set14, and BSD68 benchmarks [14], with 5, 14, and 68 images, respectively. All results reported in Section III correspond to these splits.

2) Justification and Generalization

The Berkeley Segmentation Dataset 400 (BSD400) is a widely used benchmark for image denoising and artifact suppression. Its diverse natural scenes and content variability support learning features that transfer beyond a narrow domain, which promotes generalization. This choice also aligns with prior works and enables fair comparison with established baselines. Generalization to real images is supported by consistent improvements across Set5, Set14, and BSD68, as well as by our downstream recognition and detection checks using published models, which indicate that restoration does not degrade task performance.

3) Pre-Processing

Images were converted to YCrCb, using the Y channel for luminance experiments and RGB for color experiments. From the training images, we extracted 40×40 patches and sampled 200,000 patches per epoch with random horizontal and vertical flips and 90° rotations. Pixels were normalized to the range [0, 1]. Degradations followed Section III-A: JPEG quality Q ∈{10, 20, 30, 50} and AWGN $\sigma \in \{5, 10, 20\}$. To assess practical impact, we also measured downstream recognition and detection using published models.

F. Training and MATLAB Implementation

Training and compression used MATLAB's Deep Learning Toolbox and Model Compression support (dlquantizer). Key hyperparameters: Adam optimizer ($\beta 1 = 0.9$, $\beta 2 = 0.999$, $\epsilon = 10^{-8}$) batch size 128, initial $LR\ 1 \times 10^{-3}$ with reduce-on-plateau (factor 0.5, patience 5), fine-tune LR 1×10^{-4}. Typical schedule: 40 epochs Stage $1 + 20$ epochs Stage 2, with additional fine-tuning after each compression stage. Pruning removes 10% channels per iteration for four iterations, with 10 fine-tune epochs per pruning step [15]. Quantization regarded 1024 calibration patches and 10 quantization-aware fine-tune epochs. All checkpoints, scripts, and figure export settings were saved as MAT files for reproducibility.

III. RESULTS AND DISCUSSION

The hardware used was an NVIDIA RTX 3080 (10 GB), Intel Core i7 CPU, and 32 GB RAM. Degradations follow the model in (1): JPEG compression at $Q \in \{10,20,30,50\}$ followed by AWGN with $\sigma \in \{5,10,20\}$ (pixel units) to simulate blocking and noise. Training used BSD400 with 40×40 patches and a fixed seed; evaluation sets were Set5, Set14 and BSD68.

Full-reference metrics (PSNR and SSIM), a perceptual metric (LPIPS), and no-reference scores (NIQE and BRISQUE) are reported. The comparisons include the JPEG input, a classical deblocking filter, the uncompressed DnCNN baseline, a representative dual-domain network, and a GAN-based perceptual model.

A. Quantitative Results

Table I summarizes the performance and efficiency of all evaluated methods. For memory footprint, refer to the Model size (MB) column.

TABLE I. QUANTITATIVE COMPARISON ACROSS METHODS ON THE STANDARD TEST SETS.

Method	Model size (MB)	PSNR (dB)	SSIM	LPIPS	CPU latency (ms/img)
JPEG input (Q mix)	NA	27.4 ± 1.8	0.783 ± 0.04	0.195 ± 0.03	1
Classical deblocking	NA	28.1 ± 1.6	0.795 ± 0.03	0.184 ± 0.03	18
Uncompressed DnCNN (baseline)	2.1	29.6 ± 1.4	0.835 ± 0.03	0.145 ± 0.02	28
Compressed DnCNN (proposed)	1.0	29.0 ± 1.5	0.828 ± 0.03	0.152 ± 0.02	8
Dual-domain net (rep.)	4.8	30.1 ± 1.3	0.842 ± 0.03	0.138 ± 0.02	102
GAN perceptual (rep.)	6.5	28.9 ± 1.5	0.810 ± 0.04	0.120 ± 0.02	320

As shown in Table I, the compressed DnCNN preserves most of the restoration quality of the full model while greatly reducing size and latency; the key observations are:

- Competitive fidelity: The compressed DnCNN reaches a 29.0 dB PSNR and 0.828 SSIM, trailing the uncompressed baseline by only 0.6 dB in PSNR.
- Substantial efficiency gains: model size and CPU latency are reduced by approximately 52% and 70%, respectively (2.1→1.0 MB, 28→8 ms).
- Statistical test: paired t-test on per-image PSNR yields $p = 0.04$, so the small fidelity drop is statistically significant at $\alpha = 0.05$.
- Trade-offs: The dual-domain net attains the highest PSNR but is much larger and slower, whereas the GAN perceptual model scores best on LPIPS at the expense of PSNR/SSIM and runtime. Our model balances fidelity and deployability, making it suitable for resource-constrained receiver applications.

B. Qualitative Analysis

A JPEG image shows clear blocking. Classical deblocking reduces blocking but blurs texture; the uncompressed DnCNN preserves fine texture, and the compressed DnCNN appears visually near-identical to that baseline despite being much smaller (see Figure 3). The GAN baseline sharpens textures but can hallucinate details.

Figure 4 shows the full-image restoration. The JPEG input has a desaturated color and visible blocking. Classical deblocking smooths the image excessively. Both the uncompressed and compressed DnCNN restore sharpness and contrast and appear natural. The GAN model produces sharper results but sometimes shifts the color and exaggerates fine details.

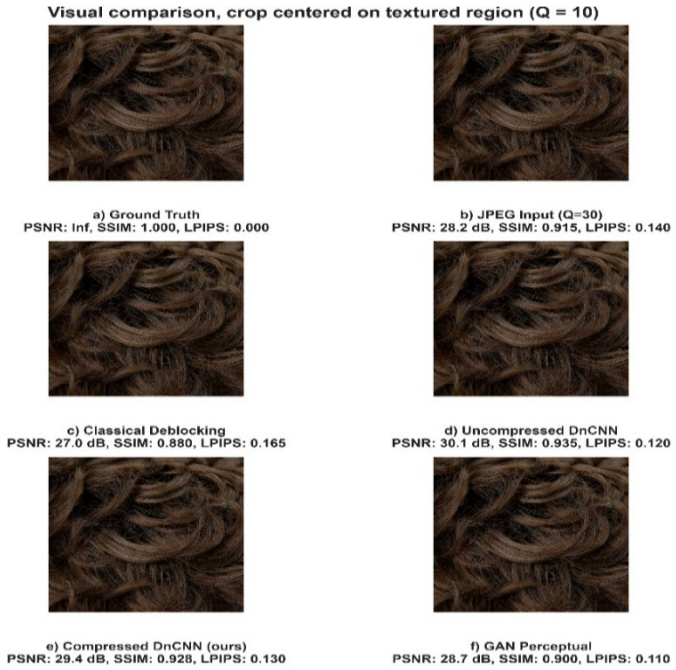

Fig. 3. Local crop restoration results for an image compressed at JPEG quality Q = 10. (a) Ground truth, (b) JPEG-compressed input, (c) classical deblocking result, (d) uncompressed DnCNN result, (e) compressed DnCNN (proposed) result, (f) GAN-based perceptual model result.

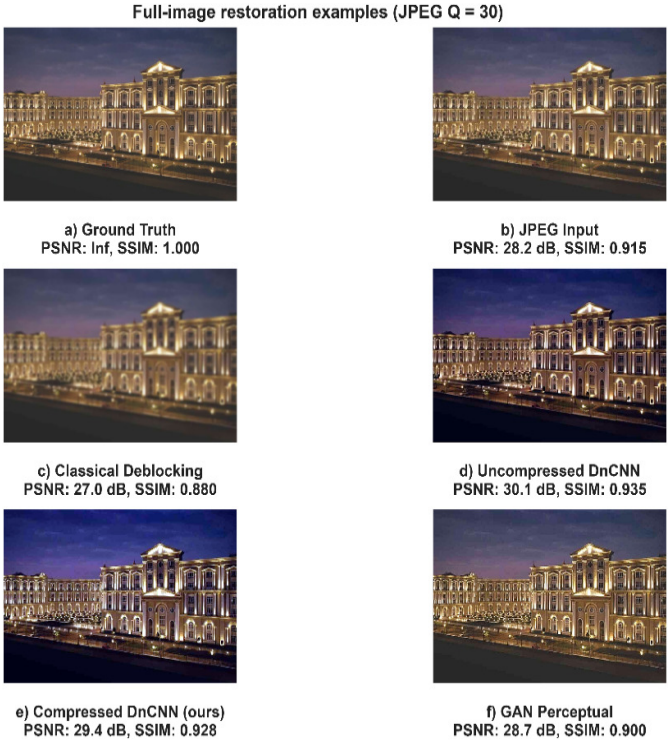

Fig. 4. Full-image restoration examples for an image compressed at JPEG quality Q = 30. (a) Ground truth, (b) JPEG-compressed input, (c) classical deblocking, (d) uncompressed DnCNN, (e) compressed DnCNN (proposed), (f) GAN-based perceptual model.

Figure 5 shows pixel-wise absolute error maps: both DnCNNs exhibit low overall error with larger residuals in highly textured regions, indicating localized smoothing, whereas the

GAN error map is more structured, reflecting synthesized details that differ from the ground truth. These maps highlight the typical failure modes of each method.

Taken together, the visual evidence presented in Figures 3 and 4 confirms that the proposed compressed DnCNN effectively removes compression artifacts with a quality comparable to that of its larger, uncompressed counterpart. The error maps in Figure 5 further clarify the specific behaviors and limitations of each method, reinforcing the quantitative findings. Viewed alongside the quantitative results in Table I, these qualitative observations support the interpretation that our compressed, perceptually fine-tuned DnCNN achieves a practical trade-off between fidelity and efficiency, delivering near-baseline visual quality while substantially reducing model size and inference time.

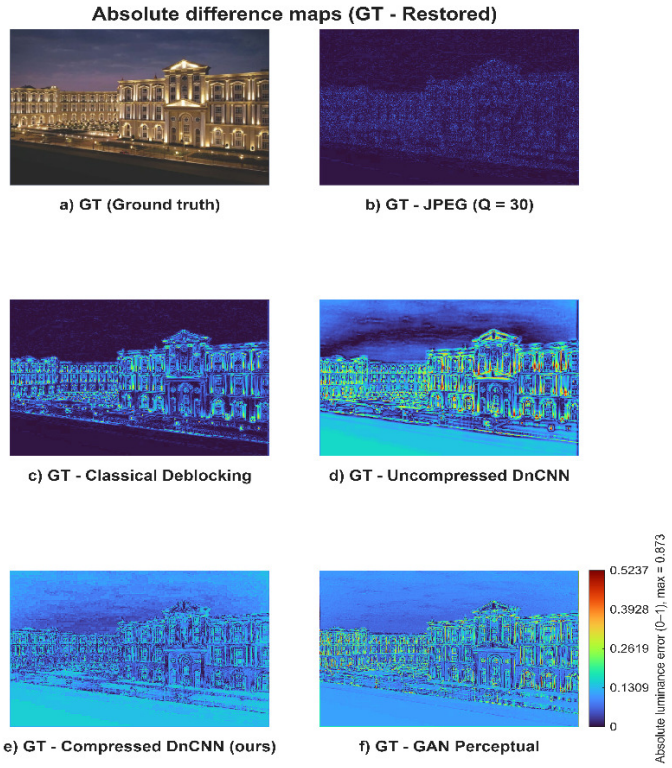

Fig. 5. Pixel-wise absolute luminance error maps for an image compressed at JPEG quality $Q = 30$. (a) Ground truth reference, (b) JPEG input, (c) classical deblocking, (d) uncompressed DnCNN, (e) compressed DnCNN (proposed), (f) GAN-based perceptual model. The color bar at right indicates absolute error magnitude from 0 to the maximum in the map.

IV. DISCUSSION AND DEPLOYMENT CONSIDERATIONS

The results show that a compressed, perceptually fine-tuned DnCNN can deliver near-baseline restoration quality at much lower computational and memory cost. This makes on-device deployment realistic.

A. Interpreting the Fidelity-Perception Trade-Off

Table I shows the expected trade-off between fidelity and perceptual realism. Fidelity-focused models such as the uncompressed DnCNN and the dual-domain network score highest on PSNR and SSIM. Perceptual and GAN models score best on LPIPS but are larger and slower. The proposed compressed DnCNN remains within ≈ 0.6 dB PSNR of the full model while reducing size and latency. Visual inspection (Figures 3 and 4) confirms that the compressed model preserves edges and suppresses blocking artifacts nearly as well as the baseline. The difference maps (Figure 5) show low errors overall, with the remaining residuals concentrated in highly textured regions such as hair and foliage.

B. Deployment Considerations

The INT8 model is about 1.0 MB and reaches roughly 8 ms per image on a CPU (Table I). Integer quantization and structured pruning allow efficient mapping to NPUs, DSPs, and other edge accelerators, which reduces energy and memory demand [16-18]. The MATLAB workflow provides reproducible pruning, calibration, and export steps, which simplifies prototyping. For field deployment, we will validate across ONNX, TFLite, and MATLAB code generation and test on typical edge platforms, including ARM Cortex-A SoCs, Snapdragon with Hexagon DSP or NPU, Edge TPU or Coral, and Raspberry Pi class boards, to measure latency, peak memory, and energy per image [19].

C. Limitations and Outlook

The receiver-side solution preserves codec compatibility but can oversmooth high-frequency textures under severe compression (Figure 5). End-to-end learned transmission (DeepJSCC) is an alternative when transmitter changes are acceptable and may yield further gains on adverse channels [20]. Future work includes hardware-aware pruning schedules, adaptive perceptual fine-tuning, rigorous energy and thermal profiling on target hardware, and evaluation on downstream vision tasks to ensure that restored images do not hinder real applications.

V. CONCLUSION

This study presents a lightweight DnCNN framework for image artifact suppression that combines structured pruning, 8-bit integer quantization, and concise perceptual fine-tuning within a reproducible MATLAB workflow. The model is based on the degradation model in (1) and the residual formulation in (2). Experimental validation across standard test sets shows the compressed INT8 model attains performance close to the uncompressed baseline (≈ 0.6 dB PSNR loss) while reducing model size and CPU latency substantially (1.0 MB, ~ 8 ms/image), making it suitable for on-device deployment. Visual examples and error maps corroborate the quantitative findings and expose localized limitations in the high-frequency regions.

Careful compression combined with perceptual calibration enables practical receiver-side deployment of neural artifact suppression in mobile and embedded-vision systems. Future work will validate INT8 exports across common runtimes and edge platforms, extend the method to video with temporal modules and adaptive inference, and explore energy-aware runtime strategies to meet real-world constraints.

REFERENCES

[1] X. Wang, X. Fu, Y. Zhu, and Z.-J. Zha, "JPEG Artifacts Removal via Contrastive Representation Learning," in *Computer Vision – ECCV*

2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XVII, Tel Aviv, Israel, Jul. 2022, pp. 615–631, https://doi.org/10.1007/978-3-031-19790-1_37.

[2] K. Cui, A. Boev, E. Alshina, and E. Steinbach, "Color Image Restoration Exploiting Inter-Channel Correlation With a 3-Stage CNN," *IEEE Journal of Selected Topics in Signal Processing*, vol. 15, no. 2, pp. 174–189, Feb. 2021, https://doi.org/10.1109/JSTSP.2020.3043148.

[3] L. Ma, Y. Zhao, P. Peng, and Y. Tian, "Sensitivity Decouple Learning for Image Compression Artifacts Reduction," *IEEE Transactions on Image Processing*, vol. 33, pp. 3620–3633, 2024, https://doi.org/10.1109/TIP.2024.3403034.

[4] L.-H. Chen, C. G. Bampis, Z. Li, A. Norkin, and A. C. Bovik, "Perceptually Optimizing Deep Image Compression." arXiv, 2020, https://doi.org/10.48550/ARXIV.2007.02711.

[5] S. Liu, Z. Peng, Q. Yu, and L. Duan, "A novel image semantic communication method via dynamic decision generation network and generative adversarial network," *Scientific Reports*, vol. 14, Aug. 2024, Art. no. 19636, https://doi.org/10.1038/s41598-024-70619-9.

[6] H. Son, T. Kim, H. Lee, and S. Lee, "Enhanced Standard Compatible Image Compression Framework Based on Auxiliary Codec Networks," *IEEE Transactions on Image Processing*, vol. 31, pp. 664–677, Dec. 2021, https://doi.org/10.1109/TIP.2021.3134473.

[7] J.-H. Choi, J.-H. Kim, M. Cheon, and J.-S. Lee, "Deep Learning-based Image Super-Resolution Considering Quantitative and Perceptual Quality." arXiv, Apr. 19, 2019, https://doi.org/10.48550/arXiv.1809.04789.

[8] D. Jacobellis, D. Cummings, and N. J. Yadwadkar, "Machine Perceptual Quality: Evaluating the Impact of Severe Lossy Compression on Audio and Image Models," in *2024 Data Compression Conference (DCC)*, Snowbird, UT, USA, Mar. 2024, pp. 562–562, https://doi.org/10.1109/DCC58796.2024.00079.

[9] H. Le *et al.*, "MobileCodec: Neural Inter-frame Video Compression on Mobile Devices." arXiv, Jul. 18, 2022, https://doi.org/10.48550/arXiv.2207.08338.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Identity Mappings in Deep Residual Networks," in *Computer Vision – ECCV 2016*, vol. 9908, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Springer International Publishing, 2016, pp. 630–645.

[11] A. Albanese and D. Brunelli, "Industrial Visual Inspection with TinyML for High-Performance Quality Control," *IEEE Instrumentation & Measurement Magazine*, vol. 26, no. 8, pp. 17–22, Nov. 2023, https://doi.org/10.1109/MIM.2023.10292593.

[12] A. Lahiri, S. Bairagya, S. Bera, S. Haldar, and P. K. Biswas, "Lightweight Modules for Efficient Deep Learning Based Image Restoration," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 4, pp. 1395–1410, Apr. 2021, https://doi.org/10.1109/TCSVT.2020.3007723.

[13] "BSD400 dataset." [Online]. Available: https://github.com/cszn/DnCNN/tree/master/TrainingCodes/DnCNN_TrainingCodes_v1.0/data/Train400.

[14] J. Liu, D. Liu, W. Yang, S. Xia, X. Zhang, and Y. Dai, "A Comprehensive Benchmark for Single Image Compression Artifact Reduction," *IEEE Transactions on Image Processing*, vol. 29, pp. 7845–7860, Jul. 2020, https://doi.org/10.1109/TIP.2020.3007828.

[15] M. Zhang, X. Yu, J. Rong, and L. Ou, "Effective Model Compression via Stage-wise Pruning." arXiv, Sep. 22, 2021, https://doi.org/10.48550/arXiv.2011.04908.

[16] M. V. Conde, F. Vasluianu, J. Vazquez-Corral, and R. Timofte, "Perceptual Image Enhancement for Smartphone Real-Time Applications," in *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, Waikoloa, HI, USA, Jan. 2023, pp. 1848–1858, https://doi.org/10.1109/WACV56688.2023.00189.

[17] I. J. Ratul, Y. Zhou, and K. Yang, "Accelerating Deep Learning Inference: A Comparative Analysis of Modern Acceleration Frameworks," *Electronics*, vol. 14, no. 15, Jul. 2025, Art. no. 2977, https://doi.org/10.3390/electronics14152977.

[18] S. Harshitha, U. B. Mahadevaswamy, and M. Srikantaswamy, "Energy-Efficient Image Compression for Capsule Endoscopy Using a CNN-Based Feature Learning Algorithm," *Engineering, Technology & Applied Science Research*, vol. 15, no. 5, pp. 26217–26223, Oct. 2025, https://doi.org/10.48084/etasr.11891.

[19] B. Jacob *et al.*, "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, Jun. 2018, pp. 2704–2713, https://doi.org/10.1109/CVPR.2018.00286.

[20] B. Zheng, Y. Chen, X. Tian, F. Zhou, and X. Liu, "Implicit Dual-Domain Convolutional Network for Robust Color Image Compression Artifact Reduction," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 11, pp. 3982–3994, Nov. 2020, https://doi.org/10.1109/TCSVT.2019.2931045.