ECCV'20

ONLINE

23-28 AUGUST 2020

16TH EUROPEAN CONFERENCE ON

COMPUTER VISION

WWW.ECCV2020.EU

Quantization Guided JPEG Artifact Correction

Max Ehrlich
maxehr@umiacs.umd.edu
University of Maryland

Larry Davis
lsd@umiacs.umd.edu
University of Maryland

Ser-Nam Lim
sernamlim@fb.com
Facebook AI

Abhinav Shrivastava
abhinav@cs.umd.edu
University of Maryland

JPEG Overview

Compression

1. Convert to YCbCr, subsample color channels
2. Center the pixels around zero
3. Compute the DCT of non-overlapping 8 x 8 blocks
4. Divide the DCT coefficients by a quantization matrix and round them to integers
5. Vectorize the quantized coefficients putting high frequencies at the end
6. Run-length code and entropy code

Decompression

1. Undo entropy coding, run-length coding
2. Rearrange vectors into 8 x 8 blocks
3. Multiply coefficients by the quantization matrix
4. Inverse DCT
5. Uncenter pixels
6. Upsample color channels, convert to RGB

JPEG Artifact Correction

- JPEG images are ubiquitous in modern computing
  - Despite some serious drawbacks and advances in compression since the 90s, JPEG has stuck around
  - Good compression ratios, not so good quality
- Quality reduction introduces artifacts, correcting these artifacts in post-processing can make a low quality image look acceptable
- Most modern JPEG software (e.g. libjpeg) includes basic deblocking filters
- More complex classical techniques exist based on analysis of the DCT coefficients

JPEG Artifact Correction - Deep Learning

- We can treat this problem as image-to-image regression and do correction with a deep network
- Long history of work in this space, some highlights:
  - ARCNN [1] - first to apply deep learning to this problem
  - MWCNN [2] - apply a wavelet based network, significant improvement
  - Galetiri et. al [3] - use a GAN to produce nicer looking mages
  - DMCNN [4] - use pixels and DCT coefficients

Major Problems in Prior Works

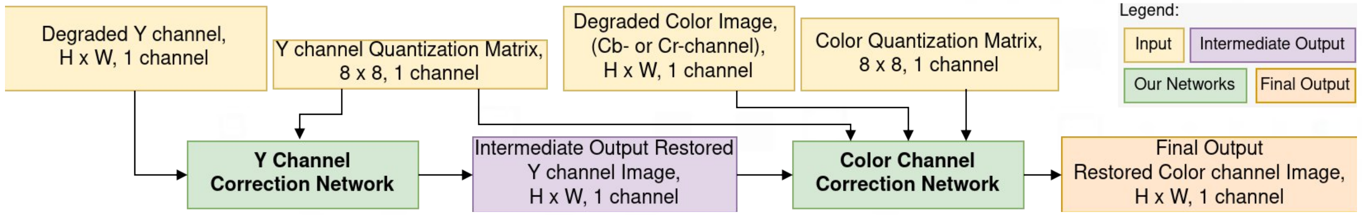
Three major issues

- They only correct the Y channel (grayscale).
- They train a separate model for each quality level (potentially 101 different models).
- They use regression (or classical GANs) which give a blurry or unrealistic result.

- Importantly: the JPEG quality level is not stored with the image, so a real system has no way to know which model to use for correction
- Only recently have people started considering the "blind" or "pseudo-blind" scenario where the network has no or limited knowledge of the JPEG quality, results have so far been mixed

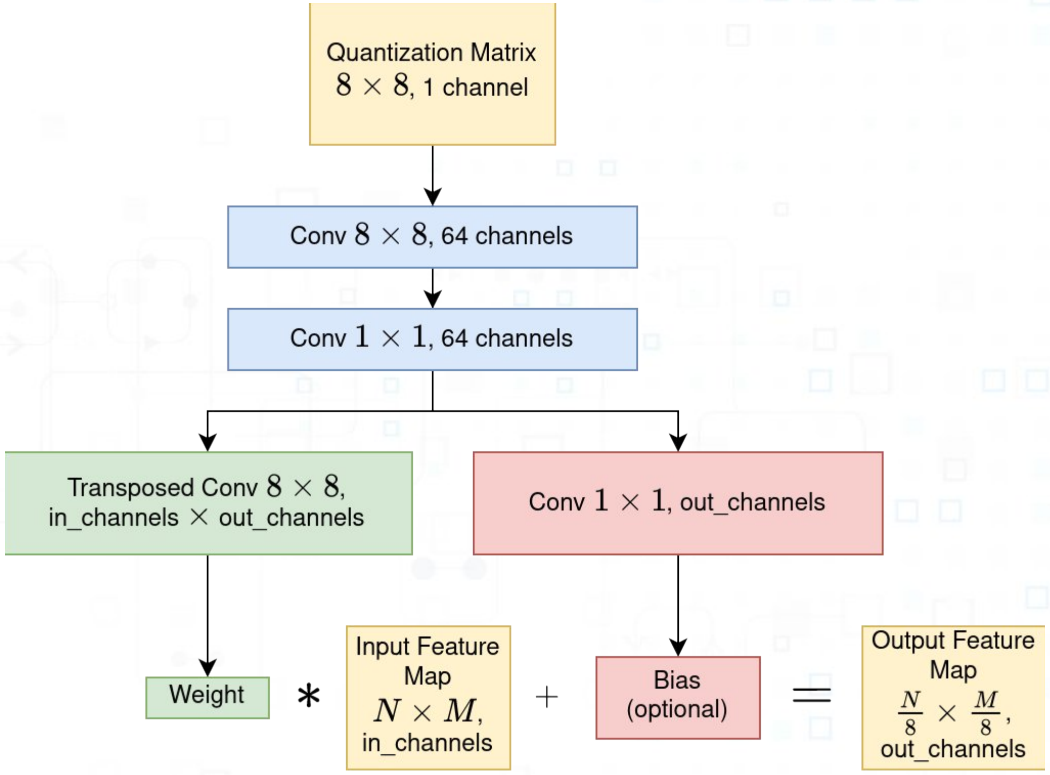
Our Method Solves These Problems with State of the Art Results

- We use a single network which takes the quantization matrix as a parameter
    - Allows us to adapt weights to different quantization levels, so the single network achives state-of-the-art performance on a range of qualities
    - Since the quantization matrix works on DCT coefficients, our network takes and produces only DCT coefficients, no pixels are used.
- We correct full color images
    - The Y channel has less compression applied to it, so we correct it first and use it to correct the color channels
- We add a GAN loss with an explicit texture term to restore texture to blurry regions



Parameterized Weights

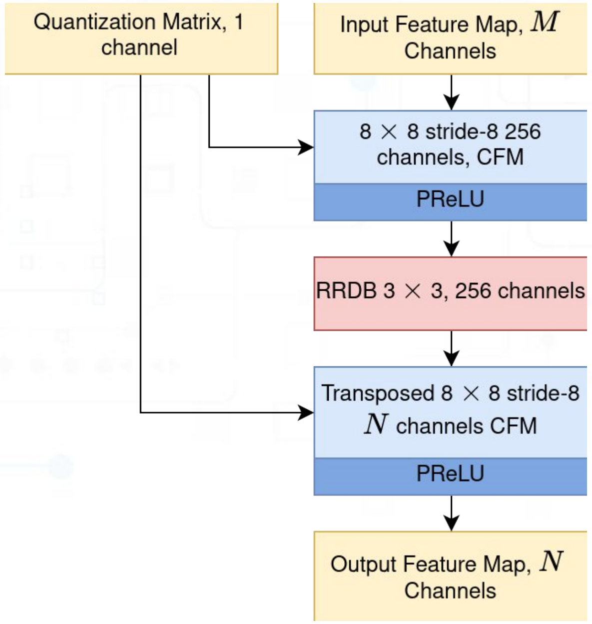
- Convolutional Filter Manifolds
  - Learn a manifold of weights that is paramterized by the quantization matrix
  - Lightweight, three layer CNN per CFM layer
  - Input is the quanitzation matrix, output is a conv weight and bias that is applied to the input feature map
- Simple extension of Filter Manifolds which take a scalar as input [5]



DCT Coefficient Regression

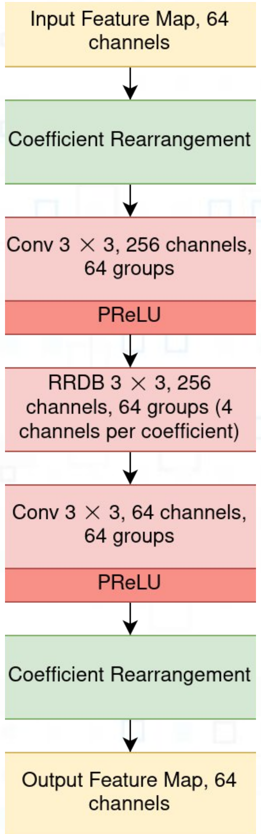
BlockNet [6]

- Processes single blocks of DCT coefficients using an 8x8 stride-8 CFM layer
- For RRDB, see ESRGAN [8]



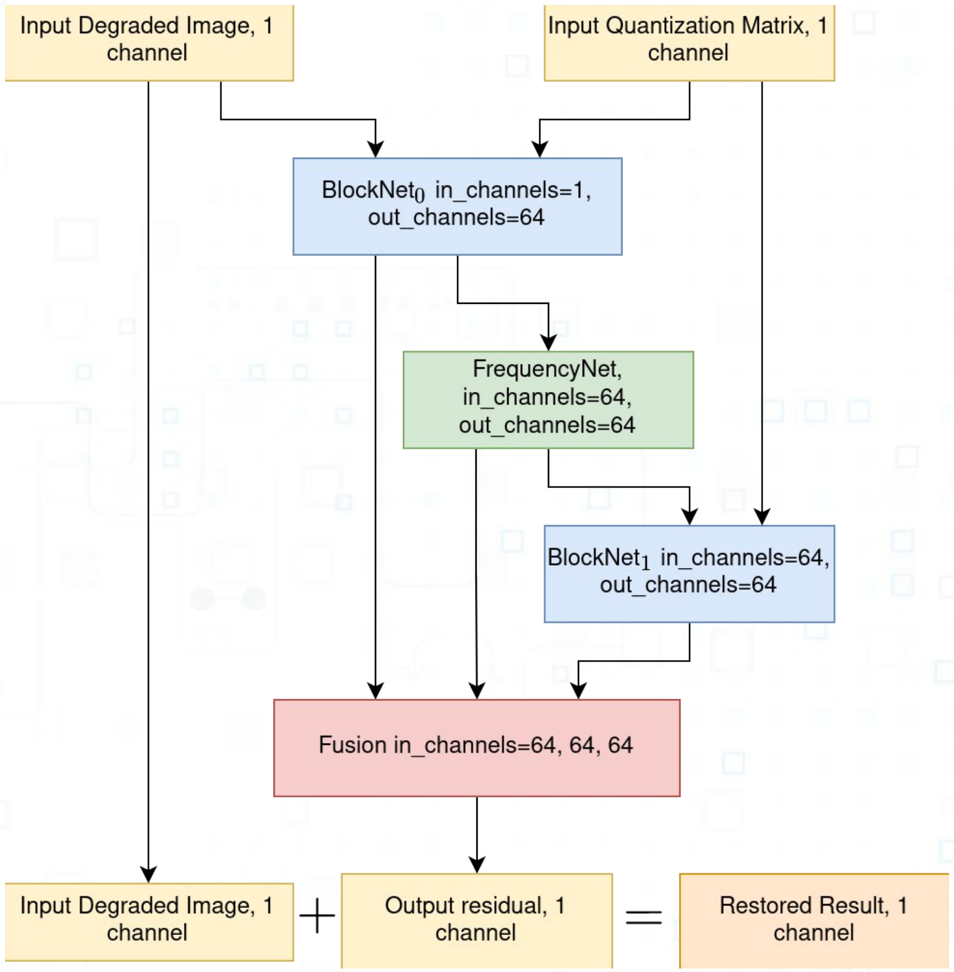
FrequencyNet [7]

- Processes each frequency in isolation
- Rearrange the frequencies channel-wise (e.g. W x H -> W/8 x H/8 x 64)
- Process in 64 groups to isolate the frequencies



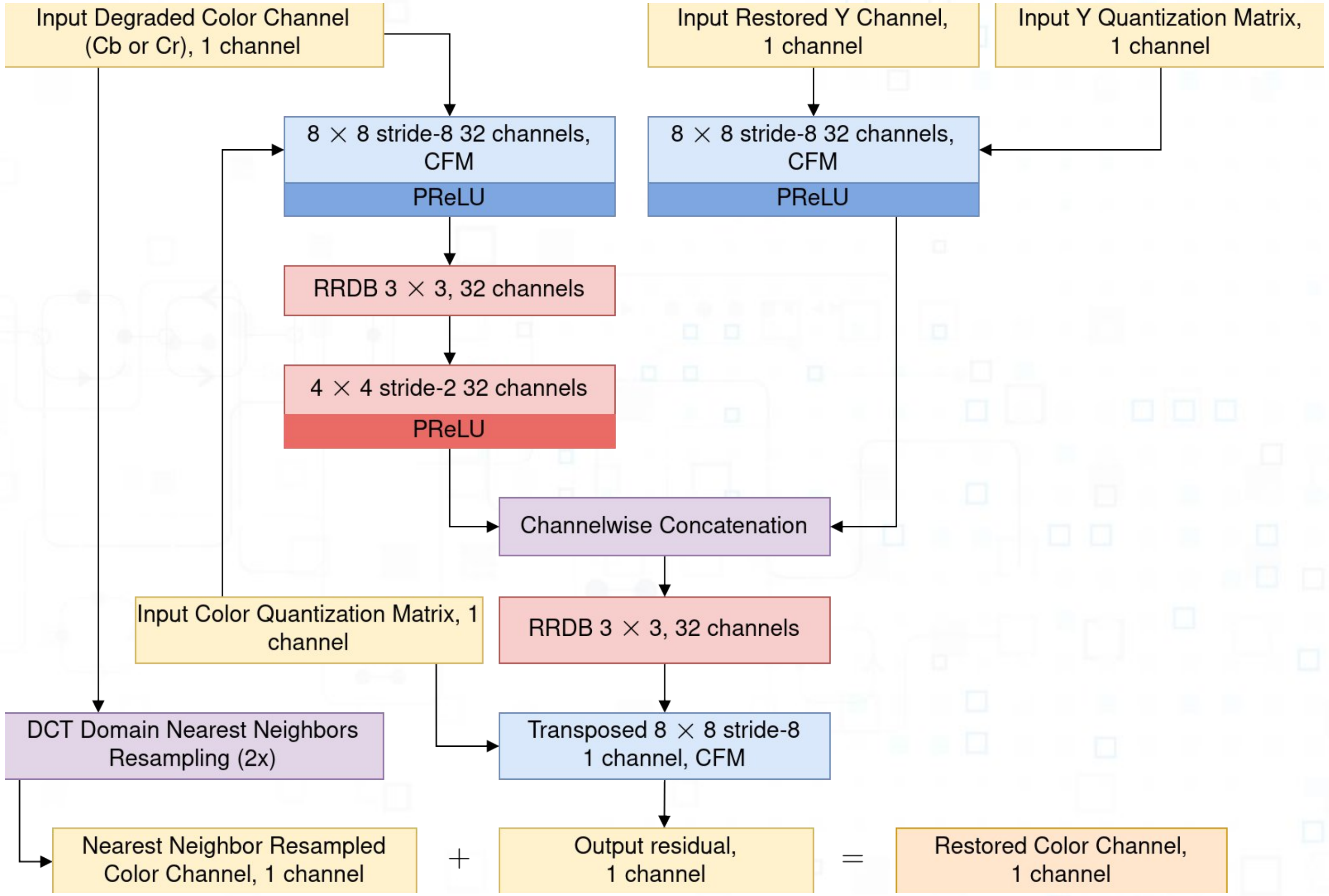
Y-Channel Correction

- Three components
  - First restore information to blocks
  - Then restore frequencies
  - Then using the additional context from previous operations, restore blocks again
- All three results are fused to promote gradient flow
- Result is treated as a residual and added to the degraded input



Color Correction

- Chroma subsampling is assumed, so the input must be doubled in size
- Use the restored Y channel to guide the color channel restoration, the color channel loses all structure during the subsampling process
- Shared weights for Cb and Cr channel restoration



Training and Loss

- First train Y channel network
- Then freeze Y channel network weights and train color network.
- Training minimizes MAE and maximizes structural similarity:

$$\mathcal{L}_{\text{JPEG}}(x, y) = \|y - x\|_1 - \lambda \text{SSIM}(x, y)$$

- Fine-tune this result with our GAN loss:

$$\mathcal{L}_{\text{GAN}}(x, y) = \mathcal{L}_{\text{texture}}(x, y) + \gamma \mathcal{L}_G^{Ra}(x, y) + \nu \|x - y\|_1$$

- Using reletivistic average GAN, MAE, and texture loss:

$$\mathcal{L}_{\text{texture}}(x, y) = \|\text{MINC}_{5,3}(y) - \text{MINC}_{5,3}(x)\|_1$$

- Where MINC is a VGG network trained for material classification (this replaces the perceptual loss term)

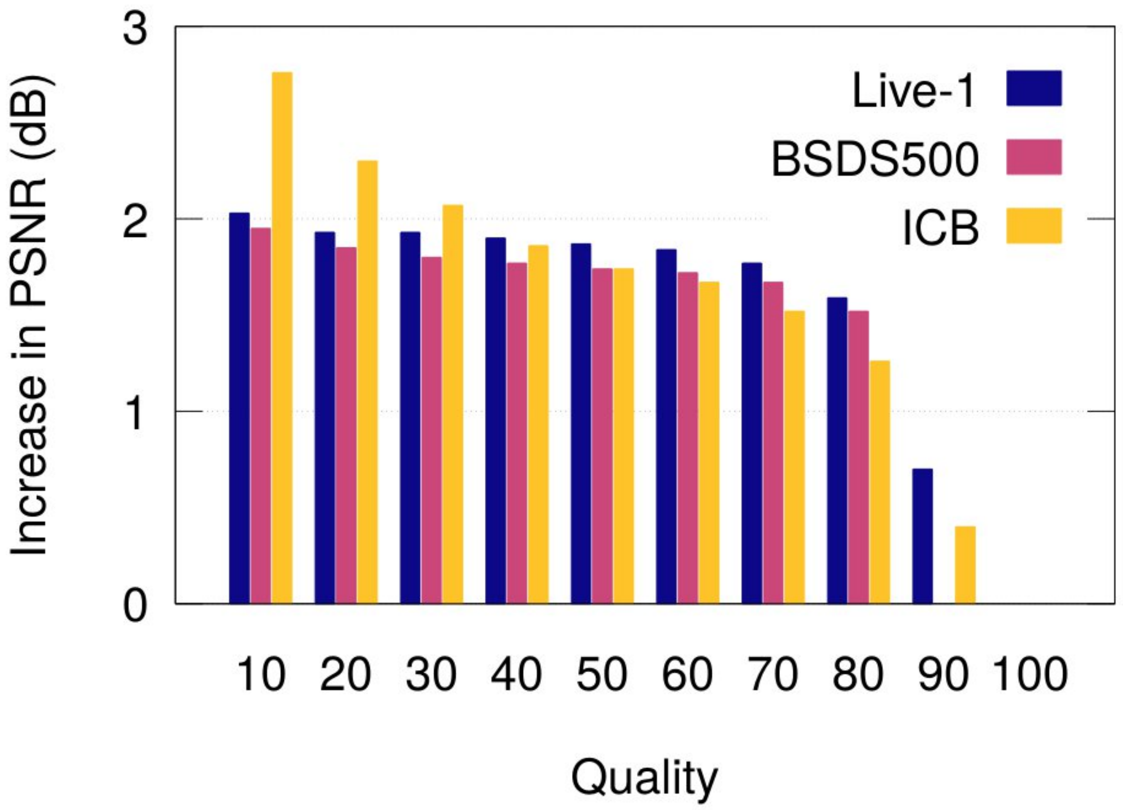
Numerical Results

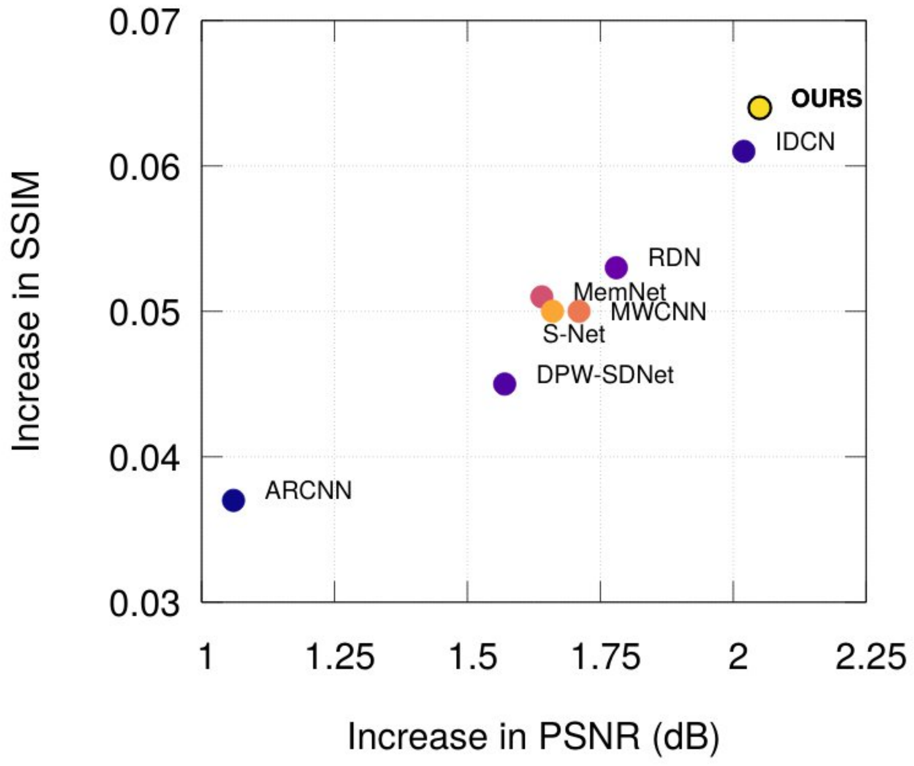
Dataset	Quality	JPEG	ARCNN[8]	MWCNN [27]	IDCN [51]	DMCNN [49]	Ours
Live-1	10	25.60 / 23.53 / 0.755	26.66 / 26.54 / 0.792	27.21 / 27.02 / 0.805	<u>27.62</u> / <u>27.32</u> / <u>0.816</u>	27.18 / 27.03 / 0.810	**27.65 / 27.40 / 0.819**
	20	27.96 / 25.77 / 0.837	28.97 / 28.65 / 0.860	29.54 / 29.23 / 0.873	**30.01** / <u>29.49</u> / <u>0.881</u>	29.45 / 29.08 / 0.874	<u>29.92</u> / **29.51 / 0.882**
	30	29.25 / 27.10 / 0.872	30.29 / 29.97 / 0.891	<u>30.82</u> / <u>30.45</u> / <u>0.901</u>	-	-	**31.21 / 30.71 / 0.908**
BSDS500	10	25.72 / 23.44 / 0.748	26.83 / 26.65 / 0.783	27.18 / 26.93 / 0.794	<u>27.61</u> / <u>27.22</u> / <u>0.805</u>	27.16 / 26.95 / 0.799	**27.69 / 27.36 / 0.810**
	20	28.01 / 25.57 / 0.833	29.00 / 28.53 / 0.853	29.45 / 28.96 / 0.866	**29.90** / <u>29.20</u> / <u>0.873</u>	29.35 / 28.84 / 0.866	<u>29.89</u> / **29.29 / 0.876**
	30	29.31 / 26.85 / 0.869	30.31 / 29.85 / 0.887	<u>30.71</u> / <u>30.09</u> / <u>0.895</u>	-	-	**31.15 / 30.37 / 0.903**
ICB	10	29.31 / 28.07 / 0.749	30.06 / 30.38 / 0.744	30.76 / 31.21 / 0.779	<u>31.71</u> / <u>32.02</u> / <u>0.809</u>	30.85 / 31.31 / 0.796	**32.11 / 32.47 / 0.815**
	20	31.84 / 30.63 / 0.804	32.24 / 32.53 / 0.778	32.79 / 33.32 / 0.812	<u>33.99</u> / <u>34.37</u> / <u>0.838</u>	32.77 / 33.26 / 0.830	**34.23 / 34.67 / 0.845**
	30	33.02 / 31.87 / 0.830	33.31 / 33.72 / 0.807	<u>34.11</u> / <u>34.69</u> / <u>0.845</u>	-	-	**35.20 / 35.67 / 0.860**

Format: PSNR/PSNR-B/SSIM
Bold: Best
<u>Underline:</u> Second Best

All compared works use a separet model per quality, ours uses only one model.

Numerical Results



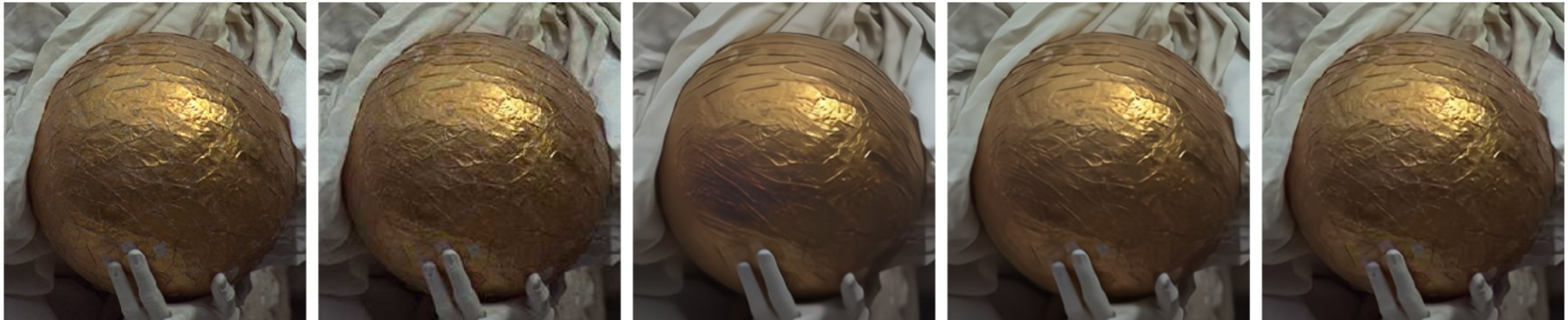


Generalization: Do Prior Works Need Separate Models?

Table 3: **Generalization Capabilities**. Live-1 dataset (PSNR / PSNR-B / SSIM).

Model Quality	Image Quality	JPEG	IDCN [51]	Ours
10	50	30.91 / 28.94 / 0.905	30.19 / 30.14 / 0.889	**32.78 / 32.19 / 0.932**
20	50	30.91 / 28.94 / 0.905	31.91 / 31.65 / 0.916	**32.78 / 32.19 / 0.932**
10	20	27.96 / 25.77 / 0.837	29.25 / 29.08 / 0.863	**29.92 / 29.51 / 0.882**
20	10	25.60 / 23.53 / 0.755	26.95 / 26.24 / 0.804	**27.65 / 27.40 / 0.819**

Original JPEG IDCN Q=10 IDCN Q=20 Ours



References

1. Chao Dong et al. “Compression artifacts reduction by a deep convolutionalnetwork”. In:Proceedings of the IEEE International Conference on ComputerVision. 2015, pp. 576–584.
2. Pengju Liu et al. “Multi-level wavelet-CNN for image restoration”. In:Pro-ceedings of the IEEE Conference on Computer Vision and Pattern RecognitionWorkshops. 2018, pp. 773–782.
3. Leonardo Galteri et al. “Deep generative adversarial compression artifact re-moval”. In:Proceedings of the IEEE International Conference on ComputerVision. 2017, pp. 4826–4835.
4. Xiaoshuai Zhang et al. “DMCNN: Dual-domain multi-scale convolutional neuralnetwork for compression artifacts removal”. In:2018 25th IEEE InternationalConference on Image Processing (ICIP). IEEE. 2018, pp. 390–394.
5. Di Kang, Debarun Dhar, and Antoni B Chan. “Crowd counting by adapting convo-lutional neural networks with side information”. In:arXiv preprint arXiv:1611.06748(2016).
6. Benjamin Deguerre, Cl´ement Chatelain, and Gilles Gasso. “Fast object detectionin compressed JPEG Images”. In:arXiv preprint arXiv:1904.08408(2019).
7. Shao-Yuan Lo and Hsueh-Ming Hang. “Exploring Semantic Segmentation onthe DCT Representation”. In:Proceedings of the ACM Multimedia Asia on ZZZ.2019, pp. 1–6
8. Xintao Wang et al. “Esrgan: Enhanced super-resolution generative adversarialnetworks”. In:Proceedings of the European Conference on Computer Vision(ECCV). 2018, pp. 0–0

https://gitlab.com/Queuecumber/quantization-guided-ac

Please Come To Our Zoom Session

Max Ehrlich
maxehr@umiacs.umd.edu
University of Maryland

Larry Davis
lsd@umiacs.umd.edu
University of Maryland

Ser-Nam Lim
sernamlim@fb.com
Facebook AI

Abhinav Shrivastava
abhinav@cs.umd.edu
University of Maryland