Quantization Guided JPEG Artifact Correction

Max Ehrlich[1,2], Larry Davis[1], Ser-Nam Lim[2], and Abhinav Shrivastava[1]

[1] University of Maryland, College Park, MD 20742, USA
{maxehr,lsd}@umiacs.umd.edu, abhinav@cs.umd.edu
[2] Facebook AI, New York, NY 10003, USA
sernamlim@fb.com

Abstract. The JPEG image compression algorithm is the most popular method of image compression because of it's ability for large compression ratios. However, to achieve such high compression, information is lost. For aggressive quantization settings, this leads to a noticeable reduction in image quality. Artifact correction has been studied in the context of deep neural networks for some time, but the current methods delivering state-of-the-art results require a different model to be trained for each quality setting, greatly limiting their practical application. We solve this problem by creating a novel architecture which is parameterized by the JPEG file's quantization matrix. This allows our single model to achieve state-of-the-art performance over models trained for specific quality settings. ...

Keywords: JPEG, Discrete Cosine Transform, Artifact Correction, Quantization

1 Introduction

The JPEG image compression algorithm [43] is ubiquitous in modern computing. Thanks to its high compression ratios, it is extremely popular in bandwidth constrained applications. The JPEG algorithm is a lossy compression algorithm, so by using it, some information is lost for a corresponding gain in saved space. This is most noticable for low quality settings

For highly space-constrained scenarios, it may be desirable to use aggressive compression. Therefore, algorithmic restoration of the lost information, referred to as artifact correction, has been well studied both in classical literature and in the context of deep neural networks.

While these methods have enjoyed academic success, their practical application is limited by a single architectural defect: they train a single model per JPEG quality level. The JPEG quality level is an integer between 0 and 100, where 100 indicates very little loss of information and 0 indicates the maximum loss of information. Not only is this expensive to train and deploy, but the quality setting is not known at inference time (it is not stored with the JPEG image [43]) making it impossible to use these models in practical applications. Only recently have methods begun considering the "blind" restoration scenario [24, 23] with a single network, with mixed results compared to non-blind methods.

Fig. 1: **Correction process.** Excerpt from ICB RGB 8bit dataset "hdr.ppm". Input was compressed at quality 10.

We solve this problem by creating a single model that uses quantization data, which is stored in the JPEG file. Our CNN model processes the image entirely in the DCT [2] domain. While previous works have recognized that the DCT domain is less likely to spread quantization errors [45, 49], DCT domain-based models alone have historically not been successful unless combined with pixel domain models (so-called "dual domain" models). Inspired by recent methods [9, 7, 8, 16], we formulate fully DCT domain regression. This allows our model to be parameterized by the quantization matrix, an 8×8 matrix that directly determines the quantization applied to each DCT coefficient. We develop a novel method for parameterizing our network called Convolution Filter Manifolds, an extension of the Filter Manifold technique [22]. By adapting our network weights to the input quantization matrix, our single network is able to handle a wide range of quality settings. Finally, since JPEG images are stored in the YCbCr color space, with the Y channel containing more information than the subsampled color channels, we use the reconstructed Y channel to guide the color channel reconstructions. As in [53], we observe that using the Y channel in this way achieves good color correction results. Finally, since regression results for artifact correction are often blurry, as a result of lost texture information, we fine-tune our model using a GAN loss specifically designed to restore texture. This allows us to generate highly realistic reconstructions. See Figure 1 for an overview of the correction flow.

To summarize, our contributions are:

1. A single model for artifact correction of JPEG images at any quality, parameterized by the quantization matrix, which is state-of-the-art in color JPEG restoration.
2. A formulation for fully DCT domain image-to-image regression.
3. Convolutional Filter Manifolds for parameterizing CNNs with spatial side-channel information.

2 Prior Work

Pointwise Shape-Adaptive DCT [10] is a standard classical technique which uses thresholded DCT coefficients reconstruct local estimates of the input signal.

Yang *et al.* [47] use a lapped transform to approximate the inverse DCT on the quantized coefficients.

More recent techniques use convolutional neural networks [26, 39]. ARCNN [8] is a regression model inspired by superresolution techniques; L4/L8 [40] continues this work. CAS-CNN [5] adds hierarchical skip connections and a multi-scale loss function. Liu *et al.* [27] use a wavelet-based network for general denoising and artifact correction, which is extended by Chen *et al.* [6]. Galteri *et al.* [12] use a GAN formulation to achieve more visually appealing results. S-Net [52] introduces a scalable architecture that can produce different quality outputs based on the desired computation complexity. Zhang *et al.* [50] use a dense residual formulation for image enhancement. Tai *et al.* [42] use persistent memory in their restoration network.

Liu *et al.* [28] introduce the dual domain idea in the sparse coding setting. Guo and Chao [17] use convolutional autoencoders for both domains. DMCNN [49] extends this with DCT rectifier to constrain errors. Zheng *et al.* [51] target color images and use an implicit DCT layer to compute DCT domain loss using pixel information. D3 [45] extends Liu *et al.* [28] by using a feed-forward formulation for parameters which were assumed in [28]. Jin *et al.* [20] extend the dual domain concept to separate streams processing low and high frequencies, allowing them to achieve competitive results with a fraction of the parameters.

The latest works examine the "blind" scenario that we consider here. Zhang *et al.* [48] formulate general image denoising and apply it to JPEG artifact correction with a single network. DCSC uses convolution features in their sparse coding scheme [11] with a single network. Galteri *et al.* [13] extend their GAN work with an ensemble of GANs where each GAN in the ensemble is trained to correct artifacts of a specific quality level. They train an auxiliary network to classify the image into the quality level that it was compressed with. The resulting quality level is used to pick a GAN from the ensemble to use for the final artifact correction. Kim *et al.* [24] also use an ensemble method based on quality factor estimation. AGARNET [23] uses a single network by learning a per-pixel quality factor extending the concept [13] from a single quality factor to a per-pixl map. This allows them to avoid the ensemble method and using a single network with two inputs.

3 Our Approach

Our goal is to design a single model capable of JPEG artifact correction at any quality. Towards this, we formulate an architecture that is parameterized by the quantization matrix.

Recall that a JPEG quantization matrix captures the amount of rounding applied to DCT coefficients and is indicative of information lost during compression. A key contribution of our approach is utilizing this quantization matrix directly to guide the restoration process using a fully DCT domain image-to-image regression network. JPEG stores color data in the YCbCr colorspace. The compressed Y channel is much higher quality compared to CbCr channels since

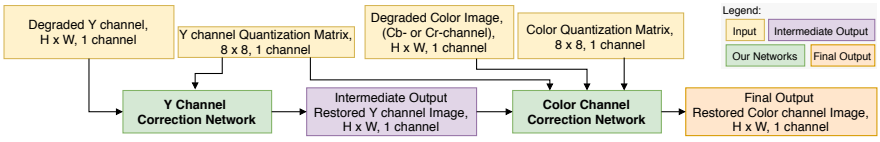

Fig. 2: **Overview.** We first restore the Y channel of the input image, then use the restored Y channel to correct the color channels which have much worse input quality.

human perception is less sensitive to fine color details than to brightness details. Therefore, we follow a staged approach: first restoring artifacts in the Y channel and then using the restored Y channel as guidance to restore the CbCr channels.

An illustrative overview of our approach is presented in Figure 2. Next, we present building blocks utilized in our architecture in §3.1, that allow us to parameterize our model using the quantization matrix and operate entirely in the DCT domain. Our Y channel and color artifact correction networks are described in §3.2 and §3.3 respectively, and finally the training details in §3.4.

3.1 Building Blocks

By creating a single model capable of JPEG artifact correction at any quality, our model solves a significantly harder problem than previous works. To solve it, we parameterize our network using the 8×8 quantization matrix available with every JPEG file. We first describe Convolutional Filter Manifolds (CFM), our solution for adaptable convolutional kernels parameterized by the quantization matrix. Since the quantization matrix encodes the amount of rounding per each DCT coefficient, this parameterization is most effective in the DCT domain, a domain where CNNs have previously struggled. Therefore, we also formulate artifact correction as fully DCT domain image-to-image regression and describe critical frequency-relationships-preserving operations.

Convolutional Filter Manifold (CFM). Filter Manifolds [22] were introduced as a way to parameterize a deep CNN using side-channel scalar data. The method learns a manifold of convolutional kernels, which is a function of a scalar input. The manifold is modeled as a three-layer multilayer perceptron. The input to this network is the scalar side-channel data, and the output vector is reshaped to the shape of the desired convolutional kernel and then convolved with the input feature map for that layer.

Recall that in the JPEG compression algorithm, a quantization matrix is derived from a scalar quality setting to determine the amount of rounding to apply, and therefore the amount of information removed from the original image. This quantization matrix is then stored in the JPEG file to allow for correct scaling of the DCT coefficients at decompression time. This quantization matrix is then a strong signal for the amount of information lost. However, the quantization matrix is an 8×8 matrix with spatial structure, applying the Filter Manifold technique to it has the same drawbacks as processing images with multilayer perceptrons, *e.g.*, a large number of parameters and a lack of spatial relationships.

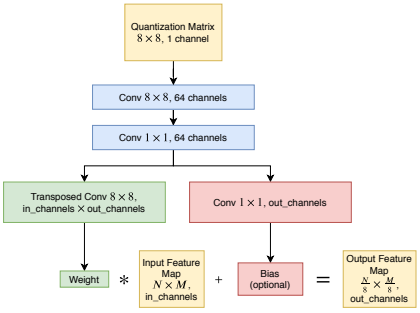

Fig. 3: **Convolutional Filter Manifold**, as used in our network. Note that the convolution with the input feature map is done with stride-8.

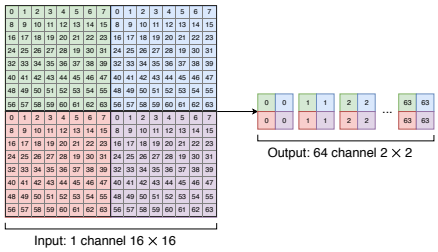

Fig. 4: **Coefficient Rearrangement**. Frequencies are arranged channelwise giving an image with 64 times the number of channels at $\frac{1}{8}$th the size. This can then be convolved with 64 groups per convolution to learn per-frequency filters.

To solve this, we propose an extension to create Convolutional Filter Manifolds (CFM), replacing the multilayer perceptron by a lightweight three-layer CNN. The input to the CNN is our quantization matrix, and the output is reshaped to the desired convolutional kernel shape and convolved with the input feature map as in the Filter Manifold method. For our problem, we follow the network structure in Figure 3 for each CFM layer. However, this is a general technique and can be used with a different architecture when spatially arranged side-channel data is available.

Coherent Frequency Operations. In prior works, DCT information has been used in dual-domain models [45, 49]. These models used standard 3×3 convolutional kernels with U-Net [35] structures to process the coefficients. Although the DCT is a linear map on image pixels [38, 9], ablation studies in prior work show that the DCT network alone is not able to surpass even classical artifact correction techniques.

Although the DCT coefficients are arranged in a grid structure of the same shape as the input image, that spatial structure does not have the same meaning as pixels. Image pixels are samples of a continuous signal in two dimensions. DCT coefficients, however, are samples from different, orthogonal functions and the two-dimensional arrangement indexes them. This means that a 3×3 convolutional kernel is trying to learn a relationship not between spatially related samples of the same function as it was designed to do, but rather between samples from completely unrelated functions. Moreover, it must maintain this structure throughout the network to produce a valid DCT as output. This is the root cause of CNN's poor performance on DCT coefficients for image-to-image regression, semantic segmentation, and object detection (Note that this should not affect whole image classification performance as in [16, 14]).

A class of recent techniques [7, 29], which we call Coherent Frequency Operations for their preservation of frequency relationships, are used as the building block for our regression network. The first layer is an 8×8 stride-8 layer [7],

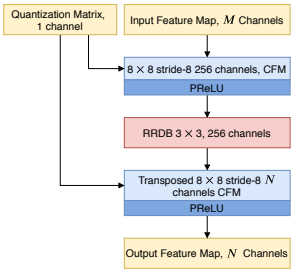

Fig. 5: **BlockNet.** Both the block generator and decoder are parameterized by the quanitzation matrix.

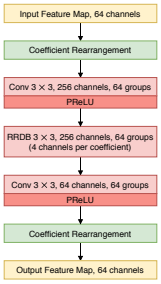

Fig. 6: **FrequencyNet.** Note that the 256 channels in the RRDB layer actually compute 4 channels per frequency.

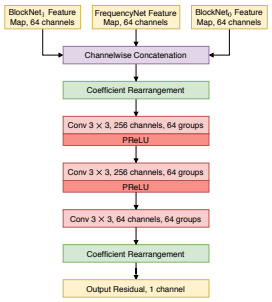

Fig. 7: **Fusion subnetwork.** Outputs from all three subnetworks are fused to produce the final residual.

which computes a representation for each block (recall that JPEG blocks are non-overlapping 8×8 DCT coefficients). This block representation, which is one eighth the size of the input, can then be processed with a standard CNN.

The next layer is designed to process each frequency in isolation. Since each of the 64 coefficients in an 8×8 JPEG block corresponds to a different frequency, the input DCT coefficients are first rearranged so that the coefficients corresponding to different frequencies are stored channelwise (see Figure 4). This gives an input, which is again one eighth the size of the original image, but this time with 64 channels (one for each frequency). This was referred to as Frequency Component Rearrangement in [29]. We then use convolutions with 64 groups to learn per-frequency convolutional weights.

Combining these two operations (block representation using 8×8 8-stride and frequency component rearrangement) allows us to match state-of-the-art pixel and dual-domain results using only DCT coefficients as input and output.

3.2 Y Channel Correction Network

Our primary goal is artifact correction of full color images, and we again leverage the JPEG algorithm to do this. JPEG stores color data in the YCbCr colorspace. The color channels, which contribute less to the human visual response, are both subsampled and more heavily quantized. Therefore, we employ a larger network to correct only the Y channel, and a smaller network which uses the restored Y channel to more effectively correct the Cb and Cr color channels.

Subnetworks. Utilizing the building blocks developed earlier, our network design proceeds in two phases: block enhancement, which learns a quantization invariant representations for each JPEG block, and frequency enhancement, which tries to match each frequency reconstruction to the regression target. These phases are fused to produce the final residual for restoring the Y channel. We employ two purpose-built subnetworks: the block network (BlockNet) and

the frequency network (FrequencyNet). Both of these networks can be thought of as separate image-to-image regression models with a structure inspired by ESRGAN [44], which allows sufficient low-level information to be preserved as well as allowing sufficient gradient flow to train these very deep networks. Following recent techniques [44], we remove batch normalization layers. While recent works have largely replaced PReLU [19] with LeakyReLU [31, 44, 12, 13], we find that PReLU activations give much higher accuracy.

BlockNet. This network processes JPEG blocks to restore the Y channel (refer to Figure 5). We use the 8×8 stride-8 coherent frequency operations to create a block representation. Since this layer is computing a block representation from all the input DCT coefficients, we use a Convolutional Filter Manifold (CFM) for this layer so that it has access to quantization information. This allows the layer to learn the quantization table entry to DCT coefficient correspondence with the goal to output a quantization-invariant block representation. Since there is a one to one correspondence between the quantization table entry and rounding applied to a DCT coefficient, this motivates our choice to operate entirely in the DCT domain. We then process these quantization-invariant block representations with Residual-in-Residual Dense Blocks (RRDB) from [44]. RRDB layers are an extension of the commonly used residual block [18] and define several recursive and highly residual layers. Each RRDB has 15 convolution layers, and we use a single RRDB for the block network with 256 channels. The network terminates with another 8×8 stride-8 CFM, this time transposed, to reverse the block representation back to its original form so that it can be used for later tasks.

FrequencyNet. This network, shown in Figure 6, processes the individual frequency coefficients using the Frequency Component Rearrangement technique (Figure 4). The architecture of this network is similar to BlockNet. We use a single 3×3 convolution to change the number of channels from the 64 input channels to the 256 channels used by the RRDB layer. The single RRDB layers processes feature maps with 256 channels and 64 groups yielding 4 channels per frequency. An output 3×3 convolution transforms the 4 channel output to the 64 output channels, and the coefficients are rearranged back into blocks for later tasks.

Final Network. The final Y channel artifact correction network is shown in Figure 8. We observe that since the FrequencyNet processes frequency coefficients in isolation, if those coefficients were zeroed out by the compression process, then it can make no attempt at restoring them (since they are zero valued they would be set to the layer bias). This is common with high frequencies by design, since they have larger quanitzation table entries and they contribute less to the human visual response. We, therefore, lead with the BlockNet to restore high frequencies. We then pass the result to the FrequencyNet, and its result is then processed by a second block network to restore more information. Finally, a three-layer fusion network (see Figure 7 and 8) fuses the output of all three subnetworks into a final result. Having all three subnetworks contribute to the final result in this way allows for better gradient flow. The effect of fusion, as well as the three subnetworks, is tested in our ablation study. The fusion output is treated as a

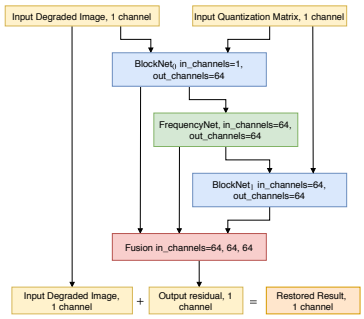

Fig. 8: **Y Channel Network.** We include two copies of the BlockNet, one to perform early restoration of high frequency coefficients, and one to work on the restored frequencies. All three subnetworks contribute to the final result using the fusion subnetwork.

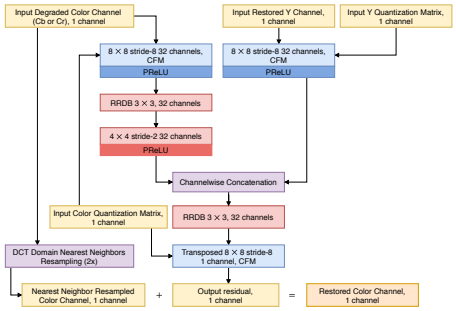

Fig. 9: **Color Channel Network.** Color channels are downsampled, so the block representation is upsampled using a learned upsampling. The Y and color channel block representations are then concatenated to guide the color channel restoration. Cb and Cr channels are processed independently with the same network.

residual and added to the input to produce the final corrected coefficients for the Y channel.

3.3 Color Correction Network

The color channel network (Figure 9) processes the Cb and Cr DCT coefficients. Since the color channels are subsampled with respect to the Y channel by half, they incur a much higher loss of information and lose the structural information which is preserved in the Y channel. We first compute the block representation of the downsampled color channel coefficients using a CFM layer, then process them with a single RRDB layer. The block representation is then upsampled using a 4×4 stride-2 convolutional layer. We compute the block representation of the restored Y channel, again using a CFM layer. The block representations are concatenated channel-wise and processed using a single RRDB layer before being transformed back into coefficient space using a transposed 8×8 stride-8 CFM. By concatenating the Y channel restoration, we give the network structural information that may be completely missing in the color channels. The result of this network is the color channel residual. This process is repeated individually for each color channel with a single network learned on Cb and Cr. The output residual is added to nearest-neighbor upsampled input coefficients to give the final restoration.

3.4 Training

Objective. We use two separate objective functions to train, an error loss and a GAN loss. Our error loss is based on prior works which minimize the l_1 error of the result and the target image. We additionally maximize the Structural

Similarity (SSIM) [46] of the result since SSIM is generally regarded as a closer metric to human perception than PSNR. This gives our final objective function as

$$\mathcal{L}_{\text{JPEG}}(x, y) = \|y - x\|_1 - \lambda \text{SSIM}(x, y) \tag{1}$$

where x is the network output, y is the target image, and λ is a balancing hyperparameter.

A common phenomenon in JPEG artifact correction and superresolution is the production of a blurry or textureless result. To correct for this, we fine tune our fully trained regression network with a GAN loss. For this objective, we use the relativistic average GAN loss $\mathcal{L}_G^{Ra}$ [21], we use l_1 error to prevent the image from moving too far away from the regression result, and we use preactivation network-based loss [44]. Instead of a perceptual loss that tries to keep the outputs close in ImageNet-trained VGG feature space used in prior works, we use a network trained on the MINC dataset [4], for material classification. This texture loss provided only marginal benefit in ESRGAN [44] for super-resolution. We find it to be critical in our task for restoring texture to blurred regions, since JPEG compression destroys these fine details. The texture loss is defined as

$$\mathcal{L}_{\text{texture}}(x, y) = \|\text{MINC}_{5,3}(y) - \text{MINC}_{5,3}(x)\|_1 \tag{2}$$

where $\text{MINC}_{5,3}$ indicates that the output is from layer 5 convolution 3. The final GAN loss is

$$\mathcal{L}_{\text{GAN}}(x, y) = \mathcal{L}_{\text{texture}}(x, y) + \gamma \mathcal{L}_G^{Ra}(x, y) + \nu \|x - y\|_1 \tag{3}$$

with γ and ν balancing hyperparameters. We note that the texture restored using the GAN model is, in general, not reflective of the regression target at inference time and actually produces worse numerical results than the regression model despite the images looking more realistic.

Staged Training. Analogous to our staged restoration, Y channel followed by color channels, we follow a staged training approach. We first train the Y channel correction network using $\mathcal{L}_{\text{JPEG}}$. We then train the color correction network using $\mathcal{L}_{\text{JPEG}}$ keeping the Y channel network weights frozen. Finally, we train the entire network (Y and color correction) with $\mathcal{L}_{\text{GAN}}$.

4 Experiments

We validate the theoretical discussion in the previous sections with experimental results. We first describe the datasets we used along with the training procedure we followed. We then show artifact correction results and compare them with previous state-of-the-art methods. Finally, we perform an ablation study. Please see our supplementary material for further results and details.

4.1 Experimental Setup

Datasets and Metrics. For training, we use the DIV2k and Flickr2k [1] datasets. DIV2k consists of 900 images, and the Flickr2k dataset contains 2650 images.

Table 1: **Color Artifact Correction Results.** PSNR / PSNR-B / SSIM format. Best result in bold, second best underlined. JPEG column gives input error. For ICB, we used the RGB 8bit dataset.

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	27.62 / 27.32 / 0.816	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** / 29.49 / 0.881	29.45 / 29.08 / 0.874	29.92 / **29.51 / 0.882**
	30	29.25 / 27.10 / 0.872	30.29 / 29.97 / 0.891	30.82 / 30.45 / 0.901	-	-	**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	27.61 / 27.22 / 0.805	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** / 29.20 / 0.873	29.35 / 28.84 / 0.866	29.89 / **29.29 / 0.876**
	30	29.31 / 26.85 / 0.869	30.31 / 29.85 / 0.887	30.71 / 30.09 / 0.895	-	-	**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	31.71 / 32.02 / 0.809	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	33.99 / 34.37 / 0.838	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	34.11 / 34.69 / 0.845	-	-	**35.20 / 35.67 / 0.860**

We preextract 256×256 patches from these images taking 30 random patches from each image and compress them using quality in $[10, 100]$ in steps of 10. This gives a total training set of 1,065,000 patches. For evaluation, we use the Live1 [36, 37], Classic-5 [10], BSDS500 [3], and ICB datasets [34]. ICB is a new dataset which provides 15 high-quality lossless images designed specifically to measure compression quality. It is our hope that the community will gradually begin including ICB dataset results. Where previous works have provided code and models, we reevaluate their methods and provide results here for comparison. As with all prior works, we report PSNR, PSNR-B [41], and SSIM [46].

Implementation Details. All training uses the Adam [25] optimizer with a batch size of 32 patches. Our network is implemented using the PyTorch [32] library. We normalize the DCT coefficients using per-frequency and per-channel mean and standard deviations. Since the DCT coefficients are measurements of different signals, by computing the statistics per-frequency we normalize the distributions so that they are all roughly the same magnitude. We find that this greatly speeds up the convergence of the network. Quantization table entries are normalized to [0, 1], with 1 being the most quantization and 0 the least. We use libjpeg [15] for compression with the baseline quantization setting.

Training Procedure. As described in Section 3.4, we follow a staged training approach by first training the Y channel or grayscale artifact correction network, then training the color (CbCr) channel network, and finally training both networks using the GAN loss.

For the first stage, the Y channel artifact correction network, the learning rate starts at 1×10^{-3} and decays by a factor of 2 every 100,000 batches. We stop training after 400,000 batches. We set λ in Equation 1 to 0.05.

For the next stage, all color channels are restored. The weights for the Y channel network are initialized from the previous stage and frozen during training. The color channel network weights are trained using a cosine annealing learning rate schedule [30] decaying from 1×10^{-3} to 1×10^{-6} over 100,000 batches.

Finally, we train both Y and color channel artifact correction networks (jointly referred to as the generator model) using a GAN loss to improve qualitative textures. The generator model weights are initialized to the pre-trained models from the previous stages. We use the DCGAN [33] discriminator. The model is

Table 2: **Y Channel Correction Results.** PSNR / PSNR-B / SSIM format, the best result is highlighted in bold, second best is underlined. The JPEG column gives with input error of the images. For ICB, we used the Grayscale 8bit dataset. We add Classic-5, a grayscale only dataset.

Dataset	Quality	JPEG	ARCNN[8]	MWCNN [27]	IDCN [51]	DMCNN [49]	Ours
Live-1	10	27.76 / 25.32 / 0.790	28.96 / 28.68 / 0.821	29.68 / 29.30 / 0.839	29.68 / 29.32 / 0.838	**29.73** / **29.43** / 0.839	29.53 / 29.15 / **0.840**
	20	30.05 / 27.55 / 0.868	31.26 / 30.73 / 0.887	32.00 / 31.47 / **0.901**	32.05 / 31.46 / 0.900	**32.07** / **31.49** / **0.901**	31.86 / 31.27 / **0.901**
	30	31.37 / 28.90 / 0.900	32.64 / 32.11 / 0.916	**33.40** / **32.76** / **0.926**	-	-	33.23 / 32.50 / 0.925
Classic-5	10	27.82 / 25.21 / 0.780	29.03 / 28.76 / 0.811	**30.01** / 29.59 / **0.837**	29.83 / 29.48 / 0.833	29.98 / **29.65** / 0.836	29.84 / 29.43 / **0.837**
	20	30.12 / 27.50 / 0.854	31.15 / 30.59 / 0.869	**32.16** / **31.52** / **0.886**	31.99 / 31.46 / 0.884	32.11 / 31.48 / 0.885	31.98 / 31.37 / 0.885
	30	31.48 / 28.94 / 0.884	32.51 / 31.98 / 0.896	**33.43** / **32.62** / **0.907**	-	-	33.22 / 32.42 / **0.907**
BSDS500	10	27.86 / 25.18 / 0.785	29.14 / 28.76 / 0.816	29.63 / 29.16 / 0.831	29.60 / 29.13 / 0.829	**29.66** / **29.27** / 0.831	29.54 / 29.04 / **0.833**
	20	30.08 / 27.28 / 0.864	31.27 / 30.52 / 0.881	31.88 / 31.12 / **0.894**	31.88 / 31.05 / 0.893	**31.91** / **31.13** / **0.894**	31.79 / 30.96 / **0.894**
	30	31.37 / 28.56 / 0.896	32.64 / 31.90 / 0.912	**33.23** / **32.29** / **0.920**	-	-	33.12 / 32.07 / **0.920**
ICB	10	32.08 / 29.92 / 0.856	31.13 / 30.97 / 0.794	34.12 / 34.06 / 0.884	32.50 / 32.42 / 0.826	34.18 / 34.15 / 0.874	**34.73** / **34.58** / **0.896**
	20	35.04 / 32.72 / 0.905	32.62 / 32.31 / 0.821	36.56 / 36.44 / 0.902	34.30 / 34.18 / 0.851	35.93 / 35.79 / 0.918	**37.12** / **36.88** / **0.924**
	30	36.66 / 34.22 / 0.927	33.79 / 33.52 / 0.841	38.20 / 37.96 / 0.927	-	-	**38.43** / **38.05** / **0.938**

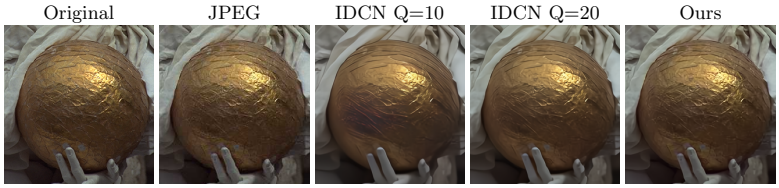

Fig. 10: **Generalization Example**. Input was compressed at quality 50. Please zoom in to view details.

trained for 100,000 iterations using cosine annealing [30] with the learning rate starting from 1×10^{-4} and ending at 1×10^{-6}. We set γ and ν in Equation 3 to 5×10^{-3} and 1×10^{-2} respectively.

4.2 Results: Artifact Correction

Color Artifact Correction. We report the main results of our approach, color artifact correction, on Live1, BSDS500, and ICB in Table 1. Our model consistently outperforms recent baselines on all datasets. Note that of all the approaches, only ours and IDCN [51] include native processing of color channels. For the other models, we convert input images to YCbCr and process the channels independently.

For quantitative comparisons to more methods on Live-1 dataset, at compression quality 10, refer to Figure 12. We present qualitative results from a mix of all three datasets in Figure 13 ("Ours"). Since our model is not restricted by which quality settings it can be run on, we also show the increase in PSNR for qualities 10-100 in Figure 11.

Intermediate Results on Y Channel Artifact Correction. Since the first stage of our approach trains for grayscale or Y channel artifact correction, we can also compare the intermediate results from this stage with other approaches. We report results in Table 2 for Live1, Classic-5, BSDS500, and ICB. As the table

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		30.91 / 28.94 / 0.905	31.91 / 31.65 / 0.916	
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**

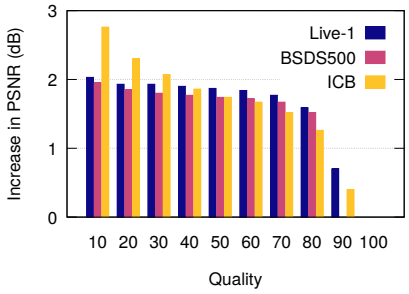

Fig. 11: **Increase in PSNR on color datasets.** For all three datasets we show the average improvement in PSNR values on qualities 10-100. Improvement drops off steeply at quality 90.

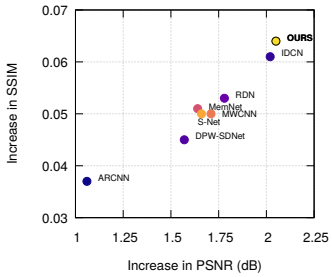

Fig. 12: **Comparison for Live-1 quality 10.** Where code was available we reevaluated, otherwise we used published numbers.

shows, intermediate results from our model can match or outperform previous state-of-the-art models in many cases, consistently providing high SSIM results using a single model for all quality factors.

GAN Correction Finally, we show results from our model trained using GAN correction. We use model interpolation [44] and show qualitative results for the interpolation parameter (α) set to 0.7 in Figure 13. ("Ours-GAN") Notice that the GAN loss is able to restore texture to blurred, flat regions and sharpen edges, yielding a more visually pleasing result. We provide additional qualitative results in the supplementary material. Note that we do not show error metrics using the GAN model as it produces higher quality images, at the expense of quantitative metrics, by adding texture details that are not present in the original images. We instead show FID scores for the GAN model compared to our regression model in Table 4, indicating that the GAN model generates significantly more realistic images.

4.3 Results: Generalization Capabilities

The major advantage of our method is that it uses a single model to correct JPEG images at any quality, while prior works train a model for each quality factor. Therefore, we explore if other methods are capable of generalizing or if they really require this ensemble of quality-specific models. To evaluate this, we

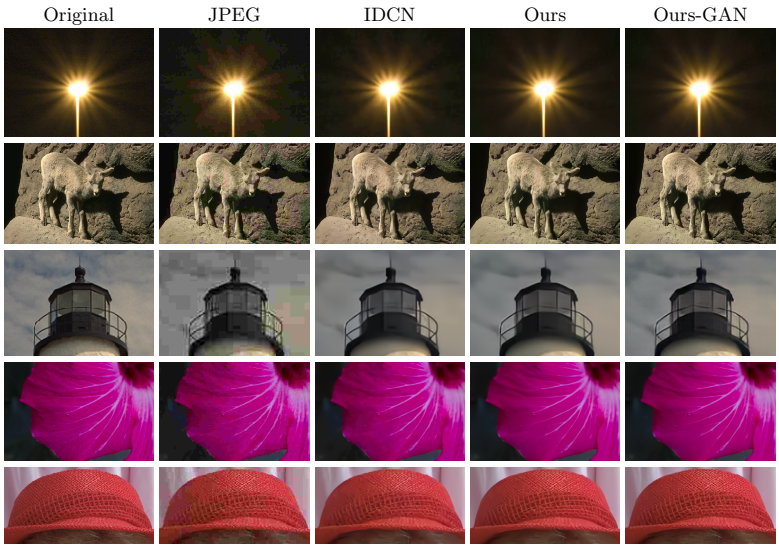

Fig. 13: **Qualitative Results.** All images were compressed at Quality 10. Please zoom in to view details.

Table 4: **GAN FID Scores.**

Dataset	Quality	Ours	Ours-GAN
Live-1	10	69.57	**35.86**
	20	36.32	**16.99**
	30	24.72	**12.20**
BSDS500	10	75.15	**34.80**
	20	42.46	**18.74**
	30	29.04	**13.03**
ICB	10	33.37	**26.08**
	20	17.23	**13.53**
	30	11.66	**10.13**

Table 5: **Ablation Results.** (refer to Section 4.4 for details).

Experiment	Model	PSNR	PSNR-B	SSIM
CFM	None	29.38	28.9	0.825
	Concat	29.32	28.94	0.823
	CFM	**29.46**	**29.05**	**0.827**
Subnetworks	FrequencyNet	28.03	25.58	0.787
	BlockNet	**29.45**	**29.04**	**0.827**
Fusion	No Fusion	27.82	25.21	0.78
	Fusion	**29.22**	**28.76**	**0.822**

use our closest competitor and prior state-of-the-art, IDCN [51]. IDCN does not provide a model for quality higher than 20, we explore if their model generalizes by using their quality 10 and quality 20 models to correct quality 50 Live-1 images. We also use the quality 20 model to correct quality 10 images and use the quality 10 model to correct quality 20 images. These results are shown in Table 3 along with our result.

As the table shows, the choice of model is critical for IDCN, and there is a significant quality drop when choosing the wrong model. Neither their quality 10 nor their quality 20 model is able to effectively correct images that it was not trained on, scoring significantly lower than if the correct model were used. At quality 50, the quality 10 model produces a result worse than the input JPEG, and the quality 20 model makes only a slight improvement. In comparison, our single model provides consistently better results across image quality factors. We stress that the quality setting is not stored in the JPEG file, so a deployed

system has no way to pick the correct model. We show an example of a quality 50 image and artifact correction results in Figure 10.

4.4 Design and Ablation Analysis

Here we ablate many of our design decisions and observe their effect on network accuracy. The results are reported in Table 5, we report metrics on quality 10 classic-5.

Implementation details: For all ablation experiments, we keep the number of parameters approximately the same between tested models to alleviate the concern that a network performs better simply because it has a higher capacity. All models are trained for 100,000 batches on the grayscale training patch set using cosine annealing [30] from a learning rate of 1×10^{-3} to 1×10^{-6}.

Importance of CFM layers. We emphasized the importance of adaptable weights in the CFM layers, which can be adapted using the quantization matrix. However, there are other simpler methods of using side-channel information. We could simply concatenate the quantization matrix channelwise with the input, or we could ignore the quantization matrix altogether. As shown in the "CFM" experiment in Table 5, the CFM unit performs better than both of these alternatives by a considerable margin. We further visualize the filters learned by the CFM layers and the underlying embeddings in the supplementary material which validate that the learned filters follow a manifold structure.

BlockNet *vs.* FrequencyNet. We noted that the FrequencyNet should not be able to perform without a preceding BlockNet because high-frequency information will be zeroed out from the compression process. To test this claim, we train individual BlockNet and FrequencyNet in isolation and report the results in Table 5 ("Subnetworks"). We can see that BlockNet alone attains significantly higher performance than FrequencyNet alone.

Importance of the fusion layer. Finally, we study the necessity of the fusion layer presented. We posited that the fusion layer was necessary for gradient flow to the early layers of our network. As demonstrated in Table 5 ("Fusion"), the network without fusion fails to learn, matching the input PSNR of classic-5 after full training, whereas the network with fusion makes considerable progress.

5 Conclusion

We showed a design for a quantization guided JPEG artifact correction network. Our single network is able to achieve state-of-the-art results, beating methods which train a different network for each quality level. Our network relies only on information that is available at inference time, and solves a major practical problem for the deployment of such methods in real-world scenarios.

Acknowledgement This project was partially supported by Facebook AI and Defense Advanced Research Projects Agency (DARPA) MediFor program (FA87501620191). There is no collaboration between Facebook and DARPA.

References

[1] Eirikur Agustsson and Radu Timofte. "Ntire 2017 challenge on single image super-resolution: Dataset and study". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2017, pp. 126–135.

[2] Nasir Ahmed, T_ Natarajan, and Kamisetty R Rao. "Discrete cosine transform". In: *IEEE transactions on Computers* 100.1 (1974), pp. 90–93.

[3] Pablo Arbelaez et al. "Contour detection and hierarchical image segmentation". In: *IEEE transactions on pattern analysis and machine intelligence* 33.5 (2010), pp. 898–916.

[4] Sean Bell et al. "Material recognition in the wild with the materials in context database". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3479–3487.

[5] Lukas Cavigelli, Pascal Hager, and Luca Benini. "CAS-CNN: A deep convolutional neural network for image compression artifact suppression". In: *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2017, pp. 752–759.

[6] Honggang Chen et al. "DPW-SDNet: Dual pixel-wavelet domain deep CNNs for soft decoding of JPEG-compressed images". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2018, pp. 711–720.

[7] Benjamin Deguerre, Clément Chatelain, and Gilles Gasso. "Fast object detection in compressed JPEG Images". In: *arXiv preprint arXiv:1904.08408* (2019).

[8] Chao Dong et al. "Compression artifacts reduction by a deep convolutional network". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 576–584.

[9] Max Ehrlich and Larry S Davis. "Deep residual learning in the jpeg transform domain". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 3484–3493.

[10] Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. "Pointwise shape-adaptive DCT for high-quality deblocking of compressed color images". In: *2006 14th European Signal Processing Conference*. IEEE. 2006, pp. 1–5.

[11] Xueyang Fu et al. "JPEG Artifacts Reduction via Deep Convolutional Sparse Coding". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 2501–2510.

[12] Leonardo Galteri et al. "Deep generative adversarial compression artifact removal". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 4826–4835.

[13] Leonardo Galteri et al. "Deep universal generative adversarial compression artifact removal". In: *IEEE Transactions on Multimedia* (2019).

[14] Arthita Ghosh and Rama Chellappa. "Deep feature extraction in the DCT domain". In: *Pattern Recognition (ICPR), 2016 23rd International Conference on*. IEEE. 2016, pp. 3536–3541.

[15] Independant JPEG Group. *libjpeg*. URL: `http://libjpeg.sourceforge.net`.
[16] Lionel Gueguen et al. "Faster neural networks straight from JPEG". In: *Advances in Neural Information Processing Systems*. 2018, pp. 3933–3944.
[17] Jun Guo and Hongyang Chao. "Building dual-domain representations for compression artifacts reduction". In: *European Conference on Computer Vision*. Springer. 2016, pp. 628–644.
[18] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.
[19] Kaiming He et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034.
[20] Zhi Jin et al. "Dual-stream Multi-path Recursive Residual Network for JPEG Image Compression Artifacts Reduction". In: *IEEE Transactions on Circuits and Systems for Video Technology* (2020).
[21] Alexia Jolicoeur-Martineau. "The relativistic discriminator: a key element missing from standard GAN". In: *arXiv preprint arXiv:1807.00734* (2018).
[22] Di Kang, Debarun Dhar, and Antoni B Chan. "Crowd counting by adapting convolutional neural networks with side information". In: *arXiv preprint arXiv:1611.06748* (2016).
[23] Yoonsik Kim, Jae Woong Soh, and Nam Ik Cho. "AGARNet: Adaptively Gated JPEG Compression Artifacts Removal Network for a Wide Range Quality Factor". In: *IEEE Access* 8 (2020), pp. 20160–20170.
[24] Yoonsik Kim et al. "A pseudo-blind convolutional neural network for the reduction of compression artifacts". In: *IEEE Transactions on Circuits and Systems for Video Technology* 30.4 (2019), pp. 1121–1135.
[25] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).
[26] Yann LeCun et al. "Handwritten digit recognition with a back-propagation network". In: *Advances in neural information processing systems*. 1990, pp. 396–404.
[27] Pengju Liu et al. "Multi-level wavelet-CNN for image restoration". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2018, pp. 773–782.
[28] Xianming Liu et al. "Data-driven sparsity-based restoration of JPEG-compressed images in dual transform-pixel domain". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 5171–5178.
[29] Shao-Yuan Lo and Hsueh-Ming Hang. "Exploring Semantic Segmentation on the DCT Representation". In: *Proceedings of the ACM Multimedia Asia on ZZZ*. 2019, pp. 1–6.
[30] Ilya Loshchilov and Frank Hutter. "Sgdr: Stochastic gradient descent with warm restarts". In: *arXiv preprint arXiv:1608.03983* (2016).

[31] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. "Rectifier nonlinearities improve neural network acoustic models". In: *Proc. icml*. Vol. 30. 1. 2013, p. 3.

[32] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[33] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks". In: *arXiv preprint arXiv:1511.06434* (2015).

[34] Rawzor. *Image Compression Benchmark*. URL: http://imagecompression.info/.

[35] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[36] Hamid R Sheikh, Muhammad F Sabir, and Alan C Bovik. "A statistical evaluation of recent full reference image quality assessment algorithms". In: *IEEE Transactions on image processing* 15.11 (2006), pp. 3440–3451.

[37] HR Sheikh et al. *LIVE image quality assessment database*. URL: http://live.ece.utexas.edu/research/quality.

[38] B Smith. "Fast software processing of motion JPEG video". In: *Proceedings of the second ACM international conference on Multimedia*. ACM. 1994, pp. 77–88.

[39] Ilya Sutskever, Geoffrey E Hinton, and A Krizhevsky. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* (2012), pp. 1097–1105.

[40] Pavel Svoboda et al. "Compression artifacts removal using convolutional neural networks". In: *arXiv preprint arXiv:1605.00366* (2016).

[41] Trinadh Tadala and Sri E Venkata Narayana. "A Novel PSNR-B Approach for Evaluating the Quality of De-blocked Images". In: (2012).

[42] Ying Tai et al. "Memnet: A persistent memory network for image restoration". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 4539–4547.

[43] Gregory K Wallace. "The JPEG still picture compression standard". In: *IEEE transactions on consumer electronics* 38.1 (1992), pp. xviii–xxxiv.

[44] Xintao Wang et al. "Esrgan: Enhanced super-resolution generative adversarial networks". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 0–0.

[45] Zhangyang Wang et al. "D3: Deep dual-domain based fast restoration of jpeg-compressed images". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2764–2772.

[46] Zhou Wang et al. “Image quality assessment: from error visibility to structural similarity”. In: *IEEE transactions on image processing* 13.4 (2004), pp. 600–612.

[47] Seungjoon Yang et al. “Blocking artifact free inverse discrete cosine transform”. In: *Proceedings 2000 International Conference on Image Processing (Cat. No. 00CH37101)*. Vol. 3. IEEE. 2000, pp. 869–872.

[48] Kai Zhang et al. “Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising”. In: *IEEE Transactions on Image Processing* 26.7 (2017), pp. 3142–3155.

[49] Xiaoshuai Zhang et al. “DMCNN: Dual-domain multi-scale convolutional neural network for compression artifacts removal”. In: *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE. 2018, pp. 390–394.

[50] Yulun Zhang et al. “Residual dense network for image restoration”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).

[51] Bolun Zheng et al. “Implicit dual-domain convolutional network for robust color image compression artifact reduction”. In: *IEEE Transactions on Circuits and Systems for Video Technology* (2019).

[52] Bolun Zheng et al. “S-Net: a scalable convolutional neural network for JPEG compression artifact reduction”. In: *Journal of Electronic Imaging* 27.4 (2018), p. 043037.

[53] Simone Zini, Simone Bianco, and Raimondo Schettini. “Deep Residual Autoencoder for quality independent JPEG restoration”. In: *arXiv preprint arXiv:1903.06117* (2019).