A Deep Learning Approach to JPEG Colour Restoration Without Ground Truth

Michael Barrack

Abstract—Deep learning methods have seen growing use in the field of image denoising. While traditionally used alongside large datasets of paired clean and noisy images, novel unsupervised techniques have been introduced that work without clean ground truth data. These approaches successfully leverage the structure of these networks to act as a prior for a noisy image. This paper focuses on the effectiveness of these existing methods at recovering colour content lost during JPEG compression with low quality factors, and how they can be augmented with this goal in mind. Specifically, a focus on smoothing the distribution of values in each colour band of the reconstruction is implemented using Kullback-Liebler Divergence as a component of model loss.

Index Terms—Computational Imaging, JPEG, Chroma Subsampling,

1 Introduction

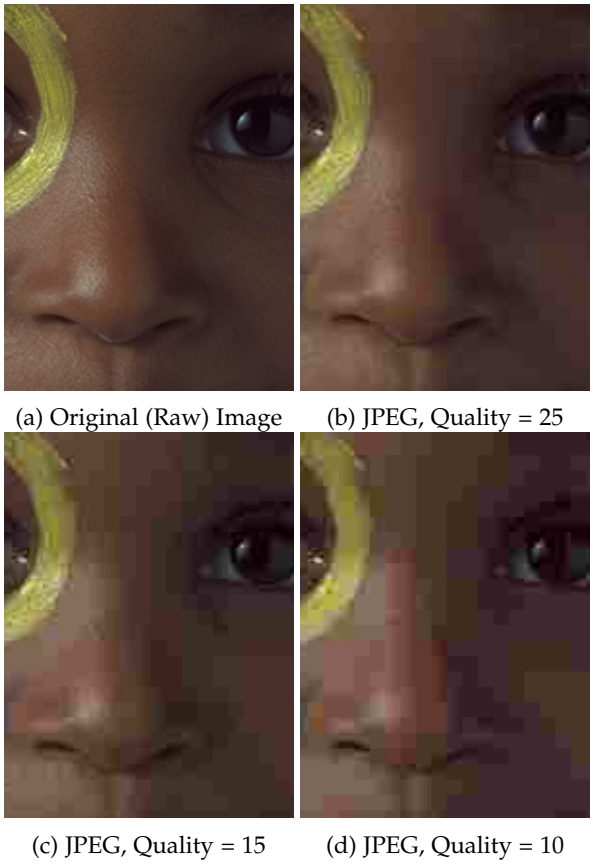
Advancements in deep learning have enabled innovative approaches to the classic problem of image denoising. With the growing accessibility of computational resources, experimentation with convolutional neural networks (CNNs) for denoising continues to expand. Traditional supervised learning approaches to this problem rely on large datasets of paired noisy and ground truth images—typically generated by corrupting clean images with a known noise distribution. Recently, there has been growing interest in unsupervised and self-supervised methods, where models learn to denoise images without relying on clean ground-truth data. [1] [2]

These novel techniques leverage the inherent structure of CNNs, allowing them to achieve strong results without the need for large datasets. As these methods continue to advance, there is growing potential to explore their effectiveness on various types of noise and image artifacts. In particular, through applying these approaches to heavily compressed JPEG images, they can be evaluated on their ability to to mitigate loss of colour information caused by chroma subsampling.

Chroma subsampling is a process used in JPEG compression that reduces the resolution of colour content in an image. It does this by averaging the colour values over multiple pixels, and is designed to exploit the human eye's weaker ability to notice differences in colour than in luminance. [3]

JPEG compression allows for the selection of a quality factor in the range 0-100, with lower quality factors resulting in smaller files but more distortion in the reconstruction. At these lower qualities, chroma subsampling becomes more pronounced, and the image may show more noticeable colour degradation, especially in areas with subtle colour gradients or high detail. [3]

This colour degradation is illustrated in Figure 1, where a small section of an image from the Kodak Dataset is compared against JPEGs with low quality factors. Note that in the lowest quality images, blocking artifacts are more prominent and the colour gradients are much harsher.



(a) Original (Raw) Image (b) JPEG, Quality = 25

(c) JPEG, Quality = 15 (d) JPEG, Quality = 10

Fig. 1: Comparison of colour loss due to chroma subsampling at various low quality factors.

2 Related Work

The foundation of this investigation comes from results presented in two papers. The first, "*Deep Image Prior*" works to show that a randomly-initialized neural network can serve as a prior for denoising with strong results. [2] The second, "*Unsupervised Learning with Stein's Unbiased Risk Estimator*" builds off this work by replacing the loss function.

2.1 Deep Image Prior

This paper introduces a framework in which a randomly initialized neural network is optimized to denoise a single noisy image. In their experiments on JPEG restoration they use an UNet-like CNN equipped with skip connections. They explain that the benefit of such hourglass architectures with skip connections is that they "impose self-similarity at multiple scales, making the corresponding priors suitable for the restoration of natural images" [2] [4].

Their approach is implemented as follows: For a single, noisy RGB image $x_0 \in \mathbb{R}^{3\times H\times W}$, a CNN is initialized with parameters θ. This network defines a mapping $f_\theta : \mathbb{R}^{3\times H\times W} \to \mathbb{R}^{3\times H\times W}$. An array of noise $z \in \mathbb{R}^{3\times H\times W}$ is independently sampled from $U(0, 0.1)$ to serve as the constant input to the network.

At every iteration t the noise z is passed through the network resulting in output $f_{\theta_t}(z) = x_t \in \mathbb{R}^{3\times H\times W}$. This output x_t and the noisy image x_0 are used to compute mean squared error loss $\text{MSE}(x_t, x_0)$. Finally, the gradient of the loss with respect to the parameters θ is calculated and used to update the model. [2] This process is depicted below in Figure 2 from [2, Fig. (2)].

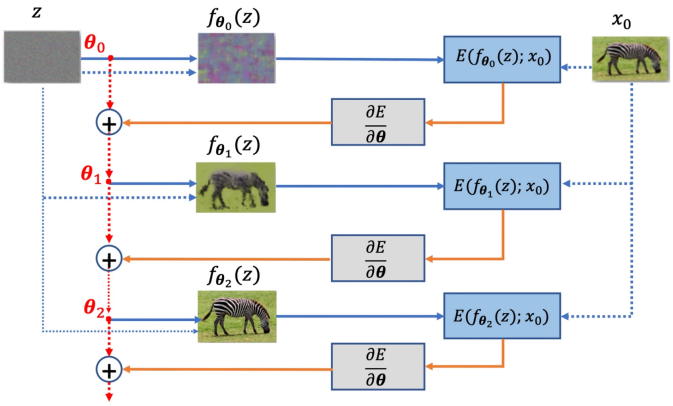


Fig. 2: Depiction of optimization process for single noisy image. [2, Fig. (2)]

The primary limitation of this approach is that the optimal number of iterations is unknown,—if allowed to run indefinitely—the network eventually overfits to the noisy image. The solution proposed is to average recent network outputs using an exponential sliding window. [2] This allows for easier selection of a stopping time. However, it can still only be determined through experimentation.

2.2 Unsupervised Learning with Stein's Unbiased Risk Estimator

This paper aims to solve the stopping-time problem from *Deep Image Prior* by replacing MSE loss with Stein's Unbiased Risk Estimator (SURE). SURE is designed to provide an estimate of MSE with respect to ground truth without access to the ground truth itself. By assuming properties of the noise in the available noisy image, it estimates the MSE from just the noisy image and the model's reconstruction. [5]

The SURE loss function is presented as follows: Let x^* denote the ground truth image. Assume that the noisy image x_0 can be modelled by $x_0 = x^* + w$, where $w \sim \mathcal{N}(0, \sigma^2 I)$. Assume also that instead of inputting noise to the network f_θ as is done in the previous approach, the noisy image x_0 is input. From [5, eq. (1)], the expectation of MSE with respect to w can be expressed as

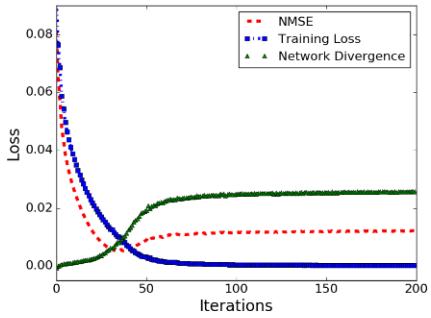
$$\mathbb{E}\left[\frac{1}{n}\|x^* - f_\theta(x_0)\|^2\right] = \mathbb{E}\left[\frac{1}{n}\|x_0 - f_\theta(x_0)\|^2\right] - \sigma_w^2 + \frac{2\sigma_w^2}{n}\text{div}_{x_0}(f_\theta(x_0)). \tag{1}$$

where $\text{div}(\cdot)$ is the divergence (from [5, eq. (2)]) defined as

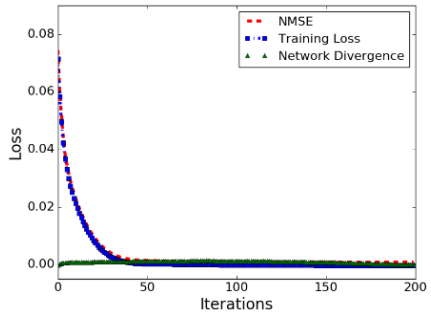
$$\text{div}_{x_0}(f_\theta(x_0)) = \sum_{n=1}^{N} \frac{\partial f_{\theta_n}(x_0)}{\partial x_{0_n}} \tag{2}$$

The idea here is that while the first term minimizes the difference between the noisy image and its reconstruction, the second term "penalizes the denoiser for varying as the input is changed" [5]. Note that the divergence term cannot be easily expressed analytically, so a Monte Carlo method is used to give an estimate. [5]. Further details about how the for this is derived and implemented can be found in [6]

To evaluate the effectiveness of using SURE in place of MSE, two networks were optimized for the same noisy image: One model was optimized using MSE and the other using SURE. The normalized MSE (NMSE), training loss, and network divergence were recorded for each model after each iteration. The results are shown in Figure 3 as presented in the [5, Fig. (1)]:



(a) MSE Training Loss



(b) SURE Training Loss

Fig. 3: Comparison of training metrics between MSE and SURE training loss [5, Fig. (1)]

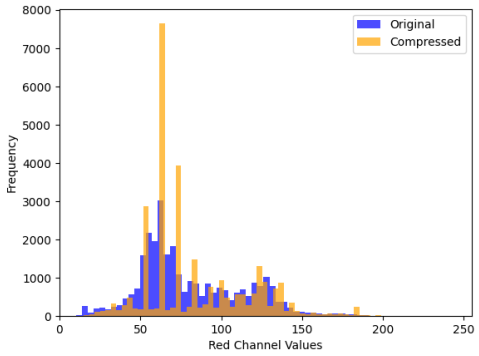
The results show that when using MSE loss, there is a point after which NMSE and divergence increase with each iteration, while training loss continues to decrease. However, when using SURE loss, NMSE and training loss decrease in tandem and quickly approach an asymptote. [5]

3 Proposed Method

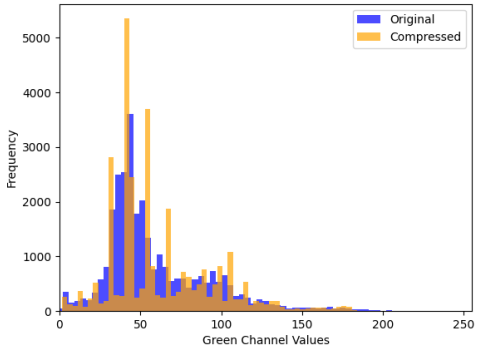
Equipped with the approaches detailed above, we can turn our attention to the effectiveness of these methods at restoring colour content, and how they might be improved upon with this goal in mind.

3.1 Effects of Chroma Subsampling

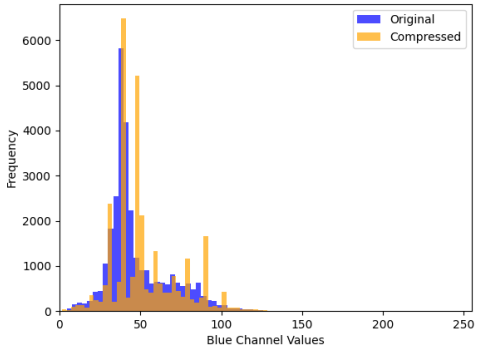
This process begins by further considering the impact of chroma subsampling on the distribution of colour values. Figure 4 below uses histograms to compare the distribution of values in each colour channel of the images shown in Figure 5a (the original image) and Figure 1d (compressed with Quality = 10).



(a) Comparison of Red channel distribution



(b) Comparison of Green channel distribution



(c) Comparison of Blue channel distribution

Fig. 4: Comparison of colour value distributions in uncompressed and JPEG (Quality = 10) images.

Representing the data this way makes clear why chroma subsampling has such a significant effect on areas with subtle colour gradients. It also provides motivation to experiment with techniques that can smooth the colour distributions in the reconstructed images. We propose that this can be accomplished via the inclusion of an additional term in the loss function.

3.2 KL-Divergence

Kullback-Liebler divergence, often denoted as $D_{KL}(P\|Q)$ measures how different a modelled distribution Q is from a true distribution P. [7] For distributions P and Q with the same support $\mathcal{X}$, the formula for KL-divergence is given by

$$D_{KL}(P\|Q) = \sum_{x\in\mathcal{X}} P(x)\log\left(\frac{P(x)}{Q(x)}\right) \tag{3}$$

The value it returns can be loosely thought of as the distance between two distributions. However, it does not satisfy the symmetry property and thus it is not a metric. Despite this, it can still be used as a loss function since it is non-negative and $D_{KL}(P\|Q) = 0 \iff Q = P$. [7]

To implement a loss term that uses KL-divergence, we must first identify the distributions that can be compared. Since the focus of this investigation is on methods that do not require ground truth data, we must use distributions approximated from a compressed image. We do this by assuming that the distribution of colour values in each channel can be sufficiently modelled by a Gaussian distribution with the same mean and variance as the JPEG. That is, for a given colour $C \in \{R, G, B\}$ and compressed image x_0, let x_0^C denote the set of values in the C colour channel. From this set we calculate the variance $\sigma_C^2 = \mathrm{Var}(x_0^C)$ and the mean $\mu_C = \mathbb{E}\left[x_0^C\right]$. We then model a distribution for this colour channel as $Q_C \sim \mathcal{N}(\mu_C, \sigma_C)$. To actually perform divergence calculations with this distribution, the density function of Q_C is converted to a mass function by sampling at each of the 256 possible pixel values and normalizing.

Letting P_C similarly denote the mass function for values in the C colour channel of the model's output x_t, we can compute divergence between these distributions as either $D_C = D_{KL}(P_C\|Q_C)$ or $D_C = D_{KL}(Q_C\|P_C)$. Note that to ensure that the divergence formula converges, we add a value of $\varepsilon = 1 \times 10^{-10}$ during calculation. Finally, choosing to weight each colour channel equally, we arrive at two possible KL-divegence loss terms given by:

$$L_{KL}^P = \frac{1}{3}\sum_{C\in\{R,G,B\}}\sum_{i=0}^{255}(P_C(i)+\varepsilon)\log\left(\frac{P_C(i)+\varepsilon}{Q_C(i)+\varepsilon}\right) \tag{4}$$

and,

$$L_{KL}^Q = \frac{1}{3}\sum_{C\in\{R,G,B\}}\sum_{i=0}^{255}(Q_C(i)+\varepsilon)\log\left(\frac{Q_C(i)+\varepsilon}{P_C(i)+\varepsilon}\right) \tag{5}$$

3.3 Loss Functions

Equipped with a method for determining KL-divergence loss, we can define a set of loss functions of the form,

$$L(x_0, x_t) = L_{KL} + \lambda d(x_0, x_t) \tag{6}$$

where $L_{KL} \in \{L_{KL}^P, L_{KL}^Q\}$, and $d \in \{\text{MSE}, \text{SURE}\}$ are weighted by a non-negative scalar $\lambda \in \mathbb{R}^+$. We can evaluate this class of functions by experimenting with different combinations of KL-divergence and distortion loss terms, and different relative weightings.

3.4 Testing Methodology

We seek to identify if these loss functions can outperform the existing approaches identified in Section 2. To do this, a total of ten loss functions must be compared: MSE as implemented in [2], SURE as implemented in [5], and eight implementations of our novel loss functions. These eight consist of each possible pair of loss terms, with $\lambda = 10$ and $\lambda = 100$.

Since the implementation of SURE in [5] was done by modifying the source code provided in [2], we choose to implement our loss functions in the same way to allow for direct comparison. We test each loss function across three model architectures with different levels of complexity. Each of these models has a symmetric "hourglass" architecture: This means corresponding pairs of downsampling and upsampling layers have the same kernel size, and number of filters. [2] Each of these models is also equipped with skip connections between some or all of these pairs of layers. In the model descriptions below $n_u[i]$, $n_d[i]$ and $n_s[i]$ correspond to the number of filters at depth i in the downsampling layers, upsampling layers, and skip-connections respectively. [2]

Low Complexity Model: This model is identical to the example used in [2] for JPEG artifact removal.

- $n_u = n_d = [8, 16, 32, 64, 128]$
- $n_s = [0, 0, 0, 4, 4]$
- *Upsampling Method:* Bilinear at each layer
- *Activation Function:* Leaky ReLU

Medium Complexity Model: This model is designed to fill the gap in complexities between the other two models.

- $n_u = n_d = [8, 16, 32, 64, 128, 256]$
- $n_s = [6, 6, 4, 4, 6, 6]$
- *Upsampling Method:* Bilinear at each layer
- *Activation Function:* Leaky ReLU

High Complexity Model: This model is identical to the implementation in the source code from [5].

- $n_u = n_d = [128, 128, 128, 128, 128, 128, 128, 128]$
- $n_s = [15, 13, 11, 9, 7, 5, 3, 1]$
- *Upsampling Method:* $i \leq 4$: Nearest, $i > 4$: Bilinear
- *Activation Function:* Leaky ReLU

To enable broad comparison of performance, we test each loss function and model complexity with and without skip connections enabled, and record their performance across a set of images.

4 Experimental Results

The Kodak dataset is a collection of 24 images commonly used for evaluation in the field of image processing. [8] For testing using the models outlined above, each image in the dataset was compressed to quality factors of 10, 15, and 25 using the Python Imaging Library (PIL). The models for each image were trained using a variety of hardware including NVIDIA's T4, L4, and 4080 GPUs.

4.1 Quantitative Results

It became immediately clear during testing that models without skip connections performed worse than those with skip connections in all cases. As a result the PSNR and MS-SSIM values for these models are not included in the tables below. However, we can report that including skip connections increased PSNR by 0.996 dB on average.

Similarly, averaging network outputs via an exponential sliding window as proposed in [2] improves both PSNR and MS-SSIM in models using MSE loss. While the benefit of this choice is less significant than the use of skip connections, PSNR increases by an average 0.176 dB, and thus only the smoothed outputs are included for these models.

Tables 1 and 2 below show the PSNR and MS-SSIM results for each loss function, JPEG quality, and model complexity. For each pair of loss terms, the λ value that yielded the best results was selected. This was not constant across each quality and model complexity, and is left arbitrary in the table headings.

TABLE 1: Restored JPEG PSNR with respect to ground truth.

Complexity	Quality	MSE	SURE	$L_{KL}^P + \lambda$MSE	$L_{KL}^Q + \lambda$MSE	$L_{KL}^P + \lambda$SURE	$L_{KL}^Q + \lambda$SURE
High	10	27.20	26.84	27.20	27.26	26.82	26.82
	15	28.01	27.85	28.34	28.32	27.84	27.88
	25	28.75	28.71	29.31	29.37	28.76	28.66
Medium	10	26.51	25.55	26.53	26.54	25.47	25.52
	15	26.76	26.19	27.12	27.13	26.26	26.35
	25	27.03	26.62	27.54	27.47	26.87	26.88
Low	10	26.83	23.66	26.85	26.83	23.57	23.64
	15	27.15	23.59	27.53	27.52	23.77	23.87
	25	27.49	23.77	28.00	27.98	23.85	24.01

TABLE 2: Restored JPEG MS-SSIM with respect to ground truth.

Complexity	Quality	MSE	SURE	$L_{KL}^P + \lambda$MSE	$L_{KL}^Q + \lambda$MSE	$L_{KL}^P + \lambda$SURE	$L_{KL}^Q + \lambda$SURE
High	10	0.911	0.899	0.914	0.914	0.900	0.900
	15	0.926	0.922	0.934	0.933	0.922	0.922
	25	0.937	0.938	0.947	0.947	0.938	0.938
Medium	10	0.892	0.887	0.892	0.893	0.887	0.888
	15	0.897	0.905	0.905	0.905	0.908	0.908
	25	0.901	0.918	0.913	0.912	0.919	0.920
Low	10	0.901	0.826	0.901	0.901	0.824	0.827
	15	0.908	0.828	0.916	0.915	0.829	0.832
	25	0.912	0.834	0.925	0.924	0.833	0.836

To put these values in context, the average PSNR and MS-SSIM values of the JPEGs used to create each model are given below in Table 3.

TABLE 3: PSNR and MS-SSIM values for JPEGs created from Kodak dataset.

Quality	PSNR	MS-SSIM
10	26.67	0.894
15	28.14	0.927
25	29.89	0.956

4.2 Quantitative Analysis

4.2.1 SURE Loss Functions

From the PSNR results, we can see that loss functions using MSE for distortion outperformed those using SURE across the board. The difference in PSNR between the two distortion measures appears to be inversely proportional to model complexity, indicating that methods using SURE loss may benefit from larger models with more skip connections.

It should be noted that the high complexity model architecture was copied directly from the source code provided in [5], and yet MSE loss performed better with this architecture across all JPEG qualities. The only instances where SURE loss functions improved PSNR or MS-SSIM above the baselines given in Table 3, are with the high complexity model and JPEGs with a quality factor of 10. Even in these exceptions the PSNR changes were small, improving by only 0.17 dB and 0.15 dB.

The impact of including a KL-divergence term was inconsistent across quality factors and complexities. For images with a quality factor of 10, neither the PSNR or MS-SSIM results improved. This is also the case for all quality factors using the high complexity model. Across all other quality-complexity combinations, the inclusion of a Kl-divergence term increased PSNR by an average of 0.13 dB. However, none of these methods resulted in PSNR or MS-SSIM above those of the source JPEGs.

These observations suggest that using SURE is not optimal for this specific testing approach. Likely, this is in large part due to the assumption that the image being restored has been corrupted by noise with a known distribution.

4.2.2 MSE Loss Functions

For JPEGs with a quality of 25, none of the models was able to improve the PSNR or MS-SSIM. Using the low and medium complexity models this was also the case for a quality of 15. However, the high complexity model using KL-divergence terms L_{KL}^{P} and L_{KL}^{Q} improved PSNR by 0.20 dB and 0.18 dB, and MS-SSIM by 0.76% and 0.65%. It should be noted the model that used MSE loss without a KL-divergence term did not improve either metric.

Models using MSE loss were most effective on JPEGs with a quality of 10, seeing improvements over the baseline with both low and high complexities. The high complexity results were the strongest: The model using no divergence loss, and the one using L_{KL}^{Q} improved PSNR by 0.53 dB, and 0.59 dB and MS-SSIM by 1.9% and 2.3%.

4.2.3 Comparison of L_{KL}^{P} and L_{KL}^{Q}

Ultimately the data suggests that the two KL-divergence loss terms perform very similarly. When used alongside both MSE and SURE distortion terms, the average absolute difference in PSNR is 0.05 dB, and in MS-SSIM is 0.0009.

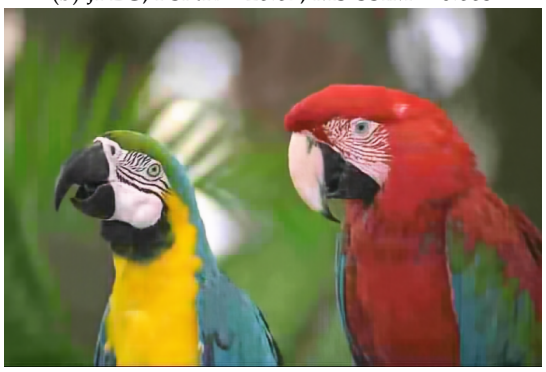
Including one of these terms in a loss function alongside MSE improved PSNR by 0.29 dB and MS-SSIM by 0.73% on average. This is a positive result that suggests use of Kl-divergence terms can improve performance in JPEG restoration under the right circumstances.

4.3 Qualitiative Results

Shown in Figure 5 are two example reconstructions of a JPEG with quality 10, along with the JPEG itself and the original image. The loss functions used to restore these images are the best performing for each distortion loss type as identified in the Section 4.2; both use the high complexity architecture.

(a) Original (Uncompressed) Image

(b) JPEG, PSNR = 28.87, MS-SSIM = 0.883



(c) $L = L_{KL}^{Q} + \lambda$MSE, PSNR=29.97, MS-SSIM=0.925

Fig. 5: JPEG Reconstruction Comparison

(d) L = SURE, PSNR=28.83, MS-SSIM=0.899

Fig. 5: JPEG Reconstruction Comparison (cont.)

We can see in both reconstructions that the blocking artifacts are less noticeable than in the JPEG. Focusing on the upper middle third of the image we can also see that the colour gradients appear smoother. However, the colour in these regions is still clearly different than in the original image.

The SURE reconstruction in Figure 5d appears visibly noisy, similar to what would be expected when corrupting an image with Gaussian noise. This is likely a consequence of the assumption noted in Section 4.2.1, and could be improved by experimenting with different noise distributions, or incorporating traditional denoising techniques.

Ultimately, the reconstruction in Figure 5c shows significant improvement from the JPEG used to create it, effectively improving image quality in regions where colour distortion is most apparent.

5 Conclusion

The results presented above reveal several limitations in our experimental approach. To properly evaluate the effectiveness of SURE as a loss function, further experimentation is essential. Without additional investigation, the impact of modifying the assumed noise distribution or increasing model complexity can only be theorized.

While positive results were achieved with MSE distortion in a handful of cases, we did not meaningfully address the overfitting limitations that SURE was proposed to prevent. The stopping time for models using MSE was determined by optimizing for more iterations than was necessary, and identifying the average best epoch to stop at after the fact, using ground truth for this evaluation.

It should also be noted that the models were designed without any consideration of image size, and were only evaluated on one size of image. These, among other limitations, prevent strong conclusions from being drawn using the results we've presented.

However, the reported instances where improvement in PSNR and MS-SSIM values can only be attributed to the inclusion of the a KL-divergence loss term justify further investigation. In this report we only evaluated one method of modelling the source distribution. It is reasonable to believe that improving this model could have a significant impact on results. As could novel approaches to computing the divergence term, such as averaging divergence from multiple smaller regions within an image, or weighing the divergences differently for each colour band.

This report highlights the potential benefit of considering the specific effects of chroma subsampling when designing models to restore JPEG images without access to ground truth data. In conclusion, we have laid a foundation from which better approaches can be designed to solve this classic problem in image processing.

References

[1] J. Lehtinen, J. Munkberg, J. Hasselgren, S. Laine, T. Karras, M. Aittala, and T. Aila, "Noise2noise: Learning image restoration without clean data," *CoRR*, vol. abs/1803.04189, 2018. [Online]. Available: http://arxiv.org/abs/1803.04189

[2] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Deep image prior," *arXiv:1711.10925*, 2017.

[3] D. Issue, "Chrominance subsampling in digital images," *The Pumpkin*, vol. 1, 01 2009.

[4] X. Mao, C. Shen, and Y. Yang, "Image restoration using convolutional auto-encoders with symmetric skip connections," *CoRR*, vol. abs/1606.08921, 2016. [Online]. Available: http://arxiv.org/abs/1606.08921

[5] C. A. Metzler, A. Mousavi, R. Heckel, and R. G. Baraniuk, "Unsupervised learning with stein's unbiased risk estimator," *CoRR*, 2020. [Online]. Available: https://arxiv.org/abs/1805.10531

[6] S. Ramani, T. Blu, and M. Unser, "Monte-carlo sure: A black-box optimization of regularization parameters for general denoising algorithms," *Trans. Img. Proc.*, vol. 17, no. 9, p. 1540–1554, Sep. 2008. [Online]. Available: https://doi.org/10.1109/TIP.2008.2001404

[7] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed. Burlington, MA, USA: Morgan Kaufmann, 2011.

[8] Eastman Kodak Company, "Kodak lossless true color image suite," 2010, available at: http://r0k.us/graphics/kodak/ [Accessed Dec. 6, 2024].