Restoring Classics to 4K Glory

Gnana Prakash
CSD
IIIT Hyderabad

Swayam Agrawal
CSE
IIIT Hyderabad

Abstract

Colorizing grayscale images is a pivotal process in image processing and computer vision, aiming to restore or assign colors to monochromatic images. This technique enhances the visual appeal and interpretability of images across various domains. The primary objective of this project is to develop an automated method for colorizing grayscale images, leveraging image-processing algorithms to produce realistic and contextually appropriate colors. This approach not only enriches the visual experience but also enhances the informational value of images, making them more accessible and engaging to a broader audience.

1 Introduction

This project focuses on grayscale image colorization leveraging a multi-step methodology to achieve high-quality, visually appealing results. The dataset employed for training is the widely-used ImageNet-1k (Russakovsky et al., 2015), comprising over a million annotated images across 1,000 object classes. ImageNet-1k serves as a benchmark for various computer vision tasks and provides diverse real-world images, ensuring robust model training.

To incorporate noise removal into the pipeline, **speckle noise**, a type of multiplicative noise commonly observed in imaging systems, is added to the grayscale images. This noise mimics realistic distortions, necessitating effective denoising techniques. Three such methods are explored:

- **Non-Local Means (NLM):** An algorithm that reduces noise by averaging pixel values with similar intensity profiles in a non-local neighborhood.
- **Total Variation (TV Chambolle):** A method that minimizes total variation in an image, preserving edges while smoothing noise.
- **Wavelet Denoising:** A multiresolution approach that decomposes the image into wavelet coefficients, allowing targeted noise suppression.

The optimal parameters for these methods are determined using the **J-invariance technique**, which involves creating a denoising function that ensures each pixel's prediction is independent of its original value, enabling robust parameter optimization using noisy data alone by leveraging statistical independence between noise components.

Following denoising, feature extraction is performed using two architectures:

- **SIFT (Scale-Invariant Feature Transform):** A classical approach that extracts descriptors robust to scaling and rotation.
- **ResNet:** A deep convolutional neural network leveraging residual connections to learn rich features effectively.

These features inform the prediction of the chrominance (U and V) channels of the image in the YUV color space, where the luminance (Y) is preserved from the original grayscale image. A **Joint**

Bilateral Filter (JBF) is then applied to refine the predicted chrominance values using the original grayscale image as guidance, enhancing spatial consistency and visual quality.

Finally, the colorized images are upscaled to 4K resolution using **bilinear interpolation**. This architecture ensures high-resolution outputs suitable for modern display requirements.

2 Noise Addition

In this work, speckle noise (George et al., 1976) is added to grayscale images to simulate the noise commonly observed in old movies (Verschaffelt et al., 2016; Liu et al., 2018). Speckle noise is a type of multiplicative noise that arises due to interference in coherent imaging systems and is characterized by intensity-dependent variations. It is mathematically modeled as:

$$N_{\text{speckle}}(.) = \mathcal{N}(\mu, \sigma^2) \cdot \mathbb{1}_{\text{shape}}$$

where:

- $N_{\text{speckle}}(.)$: The speckle noise to be added to the image.
- $\mathcal{N}(\mu, \sigma^2)$): Gaussian noise with mean μ and variance σ^2.
- $\mathbb{1}_{\text{shape}}$: A matrix of ones with the same shape as the image.

The noise is added to the image as:

$$I_{\text{noisy}}(i, j) = I(i, j) + I(i, j) \cdot N_{\text{speckle}}(i, j)$$

where $I(i, j)$ represents the original image intensity at pixel (i, j), and $N_{\text{speckle}}(i, j)$ is the speckle noise at the corresponding point.

Adding this type of noise replicates the granular and uneven textures observed in old movie frames, making it an essential step in simulating realistic degradation effects for our image colorization study.

Figure 1: Image with Speckle Noise

3 Denoising Methods

3.1 Non Local Means

The non-local means algorithm (Darbon et al., 2008) replaces the value of a pixel by an average of a selection of other pixels values: small patches centered on the other pixels are compared to the patch centered on the pixel of interest, and the average is performed only for pixels that have patches close to the current patch. As a result, this algorithm can restore well textures, that would be blurred by other denoising algorithms (Brox and Cremers, 2007).

Formulation

Given a noisy image v, the denoised value $u(p)$ at pixel p is computed as a weighted average of all pixel intensities in the image:

$$u(p) = \frac{1}{C(p)} \int_{\Omega} v(q) f(p, q) \, \mathrm{d}q$$

where:

- Ω denotes the image domain.
- $f(p, q)$ is the weighting function that quantifies the similarity between pixels p and q.
- $C(p)$ is a normalization factor ensuring that the weights sum to one:

$$C(p) = \int_{\Omega} f(p, q) \, \mathrm{d}q$$

In practice, the weighting function $f(p, q)$ is often defined using a Gaussian function of the squared Euclidean distance between patches centered at p and q:

$$f(p, q) = \exp\left(-\frac{\|v(N_p) - v(N_q)\|_2^2}{h^2}\right)$$

where:

- N_p and N_q are square neighborhoods (patches) centered at pixels p and q, respectively.
- $\|v(N_p) - v(N_q)\|_2$ denotes the Euclidean distance between the intensity vectors of patches N_p and N_q.
- h is a filtering parameter that controls the decay of the exponential function.

Figure 2: Non-Local Means Denoising

3.2 TV Chambolle

Total Variation (TV) denoising (Rudin et al., 1992) is a technique aimed at reducing noise in images while preserving edges. Chambolle's algorithm (Chambolle, 2004) is an efficient method to minimize the TV norm, which will be balancing noise suppression and edge retention.

Formulation

Given a noisy image f, the goal is to find a denoised image u that minimizes the following functional:

$$\min_u \left\{ \|u - f\|_2^2 + \lambda \, \mathrm{TV}(u) \right\}$$

where:

- $\|u - f\|_2^2$ is the data fidelity term, ensuring u remains close to f.
- λ is a regularization parameter controlling the trade-off between fidelity and smoothness.
- $\mathrm{TV}(u)$ represents the total variation of u, defined as:

$$\mathrm{TV}(u) = \sum_{i,j} \sqrt{(D_x u_{i,j})^2 + (D_y u_{i,j})^2}$$

  with D_x and D_y denoting discrete gradient operators in the horizontal and vertical directions, respectively.

Figure 3: TV-Chambolle Denoising

3.3 Wavelet Denoising

Wavelet denoising (Chang et al., 2000) is a technique to reduce noise in signals or images by leveraging the wavelet transform, which represents data in both time and frequency domains.

Formulation

1. **Wavelet Transform**:

- Decompose the noisy signal $y = v + e$, where v is the true signal and e is additive noise, using a discrete wavelet transform (DWT). This produces approximation coefficients $a_{j,k}$ and detail coefficients $b_{j,k}$ at each decomposition level j.
- Mathematically, the coefficients are derived as:

$$a_{j,k} = \int y(t)\phi_{j,k}(t)\,dt, \quad b_{j,k} = \int y(t)\psi_{j,k}(t)\,dt,$$

  where $\phi_{j,k}$ and $\psi_{j,k}$ are the scaling and wavelet functions.

2. **Thresholding**:

- Apply a shrinkage function (soft or hard thresholding) to the detail coefficients $b_{j,k}$, assuming noise mainly affects high-frequency components.
- Soft thresholding:

$$s_\lambda(x) = \begin{cases} x - \lambda & \text{if } x > \lambda, \\ 0 & \text{if } |x| \leq \lambda, \\ x + \lambda & \text{if } x < -\lambda. \end{cases}$$

- Universal threshold (λ_{univ}) is commonly chosen as:

$$\lambda_{\text{univ}} = \hat{\sigma}\sqrt{2 \ln M},$$

where M is the number of coefficients, and $\hat{\sigma}$ is the noise level estimated via the median absolute deviation (MAD):

$$\hat{\sigma} = \frac{\text{median}(|b_{j,k} - \text{median}(b_{j,k})|)}{0.6745}.$$

3. **Inverse Wavelet Transform**: Reconstruct the denoised signal by applying the inverse wavelet transform to the modified coefficients. The reconstruction ensures the true signal is preserved while minimizing noise.

4. **Result**: The process retains sharp features like edges in images or spikes in signals, unlike Fourier-based filtering which may blur such details.

Figure 4: Wavelet Denoising

4 Optimality of denoising methodologies

Batson and Royer (2019) proposed optimal parameterization for various denoising methods given a noisy image. The algorithm operates as follows:

4.1 Concept of J-Invariance

A denoising function f is J-invariant if the prediction it makes for each pixel does not depend on the value of that pixel in the original image. Mathematically, for a subset $J \subset \{1, \ldots, m\}$ of the dimensions of the measurement x, a function $f : \mathbb{R}^m \to \mathbb{R}^m$ is J-invariant if $f(x)_J$ does not depend on x_J.

4.1.1 Self-Supervised Loss

The self-supervised loss is defined as:

$$\mathcal{L}(f) = \mathbb{E}\|f(x) - x\|^2$$

For J-invariant functions, this loss can be used to find the optimal denoiser without access to clean data. This is because the self-supervised loss is equal to the ground-truth loss plus the variance of the noise:

$$\mathbb{E}\|f(x) - x\|^2 = \mathbb{E}\|f(x) - y\|^2 + \mathbb{E}\|x - y\|^2$$

4.1.2 Conversion to J-Invariance

Any denoising function g_θ can be converted into a J-invariant function f_θ using a masking procedure. For a partition $\mathcal{J}$ of the pixels such that neighboring pixels are in different subsets, the J-invariant version of g_θ is given by:

$$f_\theta(x)_J = g_\theta\left(\mathbf{1}_J \cdot s(x) + \mathbf{1}_{J^c} \cdot x\right)_J$$

where $s(x)$ is an interpolation function that replaces each pixel with the average of its neighbors.

4.2 Calibrating Denoising Functions

4.2.1 NL-Means Denoising

The NL-means algorithm replaces the center pixel of each patch with a weighted average of central pixels from similar patches. The hyperparameter to calibrate is the cut-off distance h.

- **Convert to J-Invariant Function:**

  $$f_\theta(x)_J = g_\theta\left(\mathbf{1}_J \cdot s(x) + \mathbf{1}_{J^c} \cdot x\right)_J$$

  where g_θ is the NL-means function with parameter h.

- **Compute Self-Supervised Loss:**

  $$\mathcal{L}(f_\theta) = \mathbb{E}\|f_\theta(x) - x\|^2$$

- **Optimize Hyperparameter:**

  $$h^* = \arg\min_h \mathcal{L}(f_\theta)$$

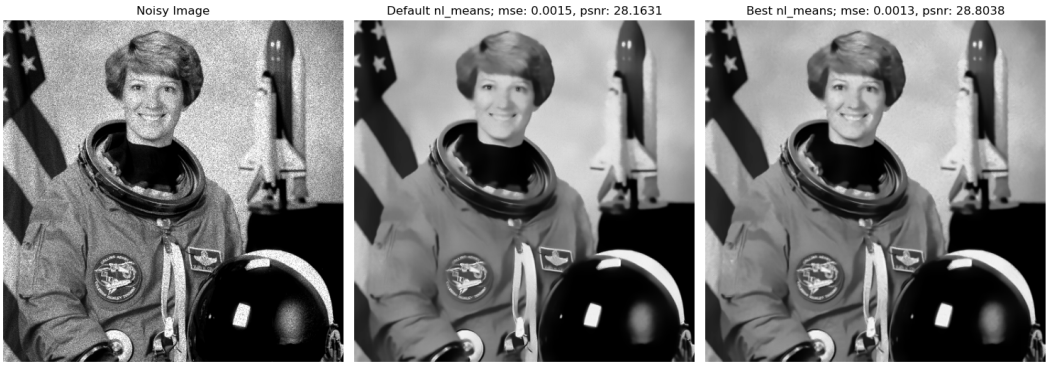

Figure 5: Comparison between default and calibrated Non-Local Means denoising

4.2.2 TV Chambolle Denoising

The TV Chambolle algorithm minimizes the total variation of the image. The hyperparameter to calibrate is the regularization parameter λ.

- **Convert to J-Invariant Function:**

$$f_\theta(x)_J = g_\theta\left(\mathbf{1}_J \cdot s(x) + \mathbf{1}_{J^c} \cdot x\right)_J$$

  where g_θ is the TV Chambolle function with parameter λ.
- **Compute Self-Supervised Loss:**

$$\mathcal{L}(f_\theta) = \mathbb{E}\|f_\theta(x) - x\|^2$$

- **Optimize Hyperparameter:**

$$\lambda^* = \arg\min_\lambda \mathcal{L}(f_\theta)$$

Figure 6: Comparison between default and calibrated TV Chambolle denoising

4.2.3 Wavelet Denoising

The wavelet denoising algorithm thresholds the wavelet coefficients of the image. The hyperparameter to calibrate is the threshold σ.

- **Convert to J-Invariant Function:**

$$f_\theta(x)_J = g_\theta\left(\mathbf{1}_J \cdot s(x) + \mathbf{1}_{J^c} \cdot x\right)_J$$

  where g_θ is the wavelet denoising function with parameter σ.
- **Compute Self-Supervised Loss:**

$$\mathcal{L}(f_\theta) = \mathbb{E}\|f_\theta(x) - x\|^2$$

- **Optimize Hyperparameter:**

$$\sigma^* = \arg\min_\sigma \mathcal{L}(f_\theta)$$

Figure 7: Comparison between default and calibrated Wavelet denoising

4.3 Comparing Different Classes of Denoisers

The self-supervised loss can also be used to compare different classes of denoisers. By computing the self-supervised loss for each class of denoiser and selecting the one with the minimum loss, we can choose the best denoiser for a given image.

5 Colorizer

5.1 Overview

This pipeline implements a deep learning framework for grayscale image colorization inspired from Cheng et al. (2016). The process begins with a grayscale input image I_Y, and the workflow is structured as follows:

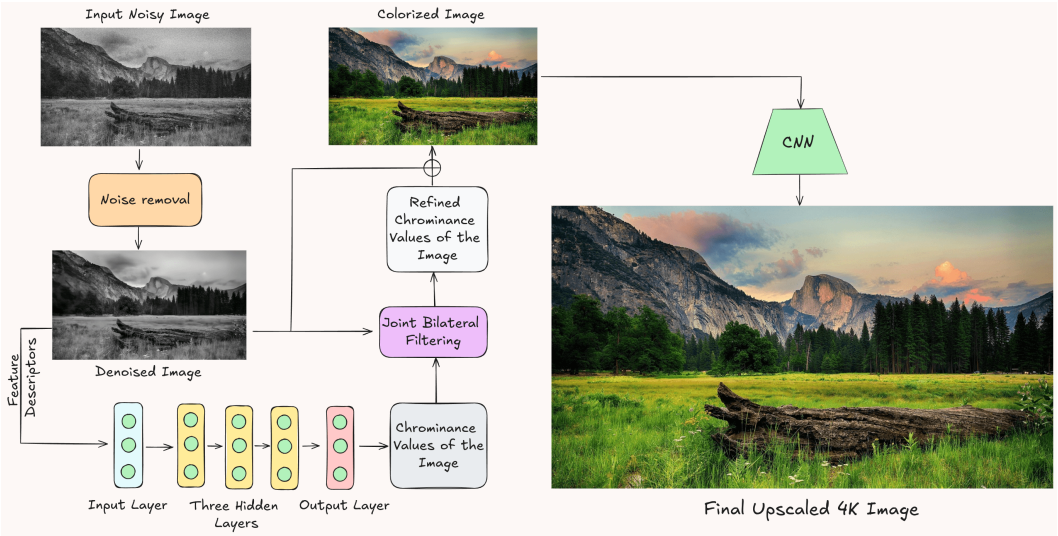

Figure 8: Overall Pipeline

1. **Colorization Model:** The input grayscale image I_Y is fed into a colorization model $\mathcal{C}(\cdot)$, which predicts the chrominance channels (U, V). These predictions are represented as $(\hat{U}, \hat{V})$:
$$(\hat{U}, \hat{V}) = \mathcal{C}(I_Y)$$
2. **Joint Bilateral Filtering:** The predicted chrominance channels $(\hat{U}, \hat{V})$ are refined using joint bilateral filtering, guided by the input grayscale image I_Y. The filtered chrominance channels $(\tilde{U}, \tilde{V})$ are computed as:
$$(\tilde{U}, \tilde{V}) = \mathcal{J}(I_Y, \hat{U}, \hat{V})$$
where $\mathcal{J}(\cdot)$ denotes the joint bilateral filter operation.
3. **Ground Truth and Loss Computation:** A ground truth RGB image I_{RGB} is used for comparison. This image is converted to obtain its grayscale representation and its chrominance channels $(U, V)_{gt}$. The loss is calculated as the Mean Squared Error (MSE) between the refined chrominance channels $(\tilde{U}, \tilde{V})$ and the ground truth $(U, V)_{gt}$:
$$\mathcal{L}_{MSE} = \frac{1}{N} \sum_{i=1}^{N} \left[(\tilde{U}_i - U_{gt,i})^2 + (\tilde{V}_i - V_{gt,i})^2 \right]$$
where N represents the total number of pixels.

5.2 Choice of YUV Color Space for Image Colorization

In the implemented colorization model, the YUV color space is used due to its effective separation of luminance and chrominance components, which decomposes an image into one luminance channel (Y) and two chrominance channels (U and V). This separation aligns with the human visual system's higher sensitivity to luminance variations than to chrominance changes (Podpora et al., 2014), allowing the model to focus on adding color information without altering the perceived brightness of the original grayscale image.

5.3 Colorization Model with SIFT Backbone

Our proposed colorization model leverages the Scale-Invariant Feature Transform (SIFT) (Lowe, 2004) to extract robust local features from grayscale images, facilitating the prediction of chrominance channels. This approach ensures that the colorization process is resilient to variations in scale, rotation, and illumination.

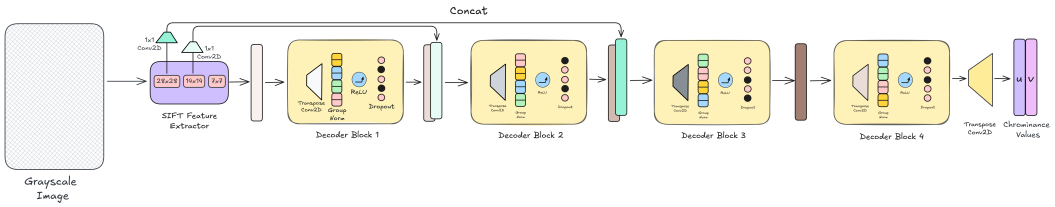

Figure 9: SIFT Colorization

5.3.1 SIFT Descriptor Extraction

SIFT descriptors are computed over image patches at multiple scales using the functions provided as part of the Kornia package (Riba et al., 2019).

- **7x7 Grid**: The grayscale image is divided into 7×7 patches, each of size 32×32 pixels. The SIFT descriptor for each patch is a 128-dimensional vector, reshaped to form a tensor of dimensions $128 \times 7 \times 7$.
- **14x14 Grid**: The image is partitioned into 14×14 patches, each 16×16 pixels, producing SIFT descriptors reshaped to $128 \times 14 \times 14$.
- **28x28 Grid**: The image is segmented into 28×28 patches, each 8×8 pixels, with descriptors reshaped to $128 \times 28 \times 28$.

These multi-scale SIFT features capture local gradients and are invariant to common image transformations, providing a robust representation for colorization tasks.

5.3.2 Decoder Architecture with Skip Connections

The decoder reconstructs the chrominance channels from the extracted SIFT features through a series of transposed convolutional layers (Shelhamer et al., 2016), incorporating skip connections to integrate multi-scale information. Skip connections are implemented by projecting SIFT features from intermediate scales using 1×1 convolutions and concatenating them with the corresponding decoder outputs, similar to the U-Net architecture (Ronneberger et al., 2015).

5.3.3 Chrominance Prediction and Post-Processing

The final output of the decoder provides the U and V chrominance channels, which are scaled to their respective ranges:

$$\mathbf{C}_{\text{pred}} = \tanh(\mathbf{F}) \odot \mathbf{S}$$

where $\mathbf{S} = [0.436, 0.615]^T$ represents the maximum absolute values for the U and V channels, respectively.

5.3.4 Conversion to RGB

The final colorized image is obtained by combining the original luminance channel Y with the refined chrominance channels U and V, and converting from the YUV color space to RGB:

$$\mathbf{I}_{\text{RGB}} = \text{YUVtoRGB}(\text{Concat}(\mathbf{Y}, \mathbf{C}_{\text{pred}}))$$

5.4 Colorization with ResNet features

The colorization model here combines a pre-trained ResNet-18 encoder (He et al., 2015) with a custom decoder incorporating skip connections in the way done in the U-Net architecture. This design helps in capturing both low-level and high-level features, facilitating the generation of realistic color images from grayscale inputs.

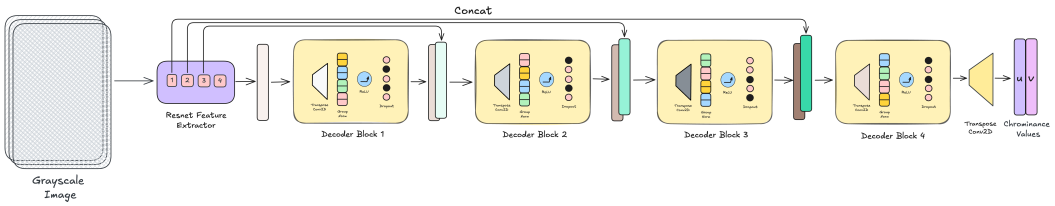

Figure 10: ResNet Colorization

5.4.1 Encoder

The encoder is based on a pre-trained ResNet-18 model, segmented to extract features at multiple depths:

- **Initial Layer**: Comprises the initial convolution, batch normalization (Ioffe and Szegedy, 2015), ReLU activation (Agarap, 2018), and max pooling (Scherer et al., 2010).
- **Intermediate Layers**: Includes four sequential layers, each producing feature maps of increasing depth.

These layers extract hierarchical features, with deeper layers capturing more abstract representations.

5.4.2 Decoder

The decoder reconstructs the color information through a series of transposed convolutional layers, each followed by group normalization (Wu and He, 2018), ReLU activation, and dropout (Le Cun et al., 1989). Skip connections from corresponding encoder layers are concatenated to preserve spatial details:

- **Decoder Layers**: Each layer upsamples the feature maps and integrates features from the encoder via skip connections.
- **Final Layer**: Outputs the chrominance channels (U and V) of the YUV color space.

5.4.3 Color Space Transformation

The model operates in the YUV color space, where the input grayscale image serves as the Y (luminance) channel. The network predicts the U and V (chrominance) channels, which are combined with Y to reconstruct the RGB image:

$$\mathbf{I}_{\text{RGB}} = \text{YUV\_to\_RGB}(\mathbf{Y}, \mathbf{U}, \mathbf{V})$$

5.5 Joint Bilateral Filtering for Chrominance Refinement

In the colorization process, the chrominance channels (U and V) predicted by the neural network may exhibit noise and artifacts, particularly in low-texture regions. To mitigate these imperfections, we employ joint bilateral filtering (Petschnigg et al., 2004), utilizing the original grayscale image as a guidance map to refine the chrominance values.

5.5.1 Bilateral Filter Overview

The bilateral filter (Tomasi and Manduchi, 1998) is an edge-preserving smoothing technique that computes the output intensity at a pixel p as a weighted average of neighboring pixel intensities:

$$I_{\text{filtered}}(p) = \frac{1}{W_p} \sum_{q \in \Omega} I(q)\, f_r\left(\|I(q) - I(p)\|\right)\, g_s\left(\|q - p\|\right)$$

where:

- I is the input image.
- Ω denotes the neighborhood of pixel p.
- f_r is the range kernel, typically a Gaussian function, measuring intensity similarity.
- g_s is the spatial kernel, also a Gaussian function, measuring spatial proximity.
- W_p is a normalization factor ensuring that the weights sum to one:

$$W_p = \sum_{q \in \Omega} f_r\left(\|I(q) - I(p)\|\right)\, g_s\left(\|q - p\|\right)$$

5.5.2 Joint Bilateral Filter Application

The joint bilateral filter (JBF) extends this concept by incorporating a guidance image G to influence the range kernel, allowing the filter to respect edges present in G. The filtered output for the chrominance channels C is computed as:

$$C_{\text{filtered}}(p) = \frac{1}{W_p} \sum_{q \in \Omega} C(q)\, f_r\left(\|G(q) - G(p)\|\right)\, g_s\left(\|q - p\|\right)$$

In our application:

- C represents the chrominance channels (U and V) predicted by the neural network.
- G is the original grayscale image, serving as the guidance image.

By using the grayscale image as guidance, the joint bilateral filter effectively smooths the chrominance channels while preserving edges and fine details, thereby reducing artifacts and enhancing the visual quality of the colorized image. It diminishes noise and artifacts in the chrominance channels, leading to cleaner colorization results (Petschnigg et al., 2004). By respecting the edges in the guidance image, it maintains sharp transitions and details.

6 Upscaler

6.1 Architecture

The upscaling model is designed to enhance colorized images to 4K resolution, using transposed convolutions and residual connections to effectively reconstruct high-resolution outputs.

6.1.1 Network Architecture

The architecture comprises a sequence of transposed convolutional layers interspersed with non-linear activation functions, normalization layers, dropout for regularization, and residual connections to facilitate gradient flow and learning efficiency.

- **Transposed Convolutions:** These layers perform upsampling by learning how to map low-resolution inputs to higher-resolution outputs, effectively increasing the spatial dimensions of the feature maps.
- **Group Normalization:** Applied after transposed convolutions to stabilize and accelerate training by normalizing feature maps across groups of channels.
- **ReLU Activation:** Introduces non-linearity into the model, enabling it to learn complex mappings.
- **Dropout:** Employed to prevent overfitting by randomly setting a fraction of input units to zero during training.
- **Residual Connections:** Skip connections add the input of a layer to its output, aiding in the preservation of spatial information and facilitating gradient propagation.

6.1.2 Forward Pass

Given an input tensor $\mathbf{x}$ of shape (N, C, H, W), where N is the batch size, C the number of channels, and $H \times W$ the spatial dimensions, the forward pass proceeds as follows:

1. **First Upsampling:** Apply a transposed convolution to double the spatial dimensions:

$$\mathbf{y}_1 = \text{ReLU}(\text{GroupNorm}(\text{ConvTranspose2d}(\mathbf{x})))$$

   Add a residual connection from the input $\mathbf{x}$, adjusted to match $\mathbf{y}_1$'s dimensions using a 1×1 convolution and bilinear interpolation:

$$\mathbf{y}_1 = \mathbf{y}_1 + \text{Interp}(\text{Conv2d}_{1\times1}(\mathbf{x}))$$

2. **Intermediate Layers:** Repeat the process of transposed convolution, normalization, activation, dropout, and residual addition for subsequent layers, progressively increasing the spatial resolution.
3. **Final Upsampling:** The last transposed convolution layer outputs the upscaled image:

$$\mathbf{y}_{\text{final}} = \text{ConvTranspose2d}(\mathbf{y}_{\text{prev}})$$

   Clamp the output to ensure pixel values remain within the valid range:

$$\mathbf{y}_{\text{final}} = \text{Clamp}(\mathbf{y}_{\text{final}}, 0, 1)$$

6.2 Computational Constraints and Alternative Approaches

Despite designing an upscaler model leveraging transposed convolutions and residual connections, training this architecture was infeasible due to high computational requirements. Consequently, we opted for bilinear interpolation to upscale images to 4K resolution (2160×4096). Bilinear interpolation estimates the pixel value $f(x, y)$ at a point (x, y) using the weighted average of its four nearest neighbors $Q_{11}, Q_{12}, Q_{21}, Q_{22}$:

$$f(x, y) \approx (1 - x)(1 - y)f(Q_{11}) + (1 - x)yf(Q_{12}) + x(1 - y)f(Q_{21}) + xyf(Q_{22})$$

This approach balances simplicity and computational efficiency, producing visually acceptable results for our application.

7 Results and Observations

7.1 Experimental Details

All model variants were trained using the complete training split of the ImageNet-1k dataset with shuffling. Training was conducted for two epochs with a batch size of 128. Validation was performed on the full validation split after every 1,000 batches. The AdamW (Loshchilov and Hutter, 2017) optimizer was employed, configured with a learning rate of 1×10^{-4} and a weight decay of 1×10^{-2}. A learning rate scheduler with fractional scaling (decay factor: 0.8, patience: 2) was utilized to improve training stability (You et al., 2019).

7.2 Colorizer with SIFT Backbone

In this section, we present the outcomes of our colorization model employing a Scale-Invariant Feature Transform (SIFT) backbone, with a focus on the impact of applying Joint Bilateral Filtering (JBF) as a post-processing step.

7.2.1 Qualitative Analysis

Figure 11: Colorized image with Joint Bilateral Filtering.

Figure 12: Colorized image without Joint Bilateral Filtering.

Figures 11 and 12 showcase the colorized images produced by the model with and without the application of JBF, respectively. The image processed with JBF exhibits attempts to atleast colorize the grayscale input image. In contrast, the image without JBF displays minimal colorization.

7.2.2 Quantitative Analysis

The training and validation loss curves for both models are depicted in Figure 13. It is evident that the model incorporating JBF consistently achieves lower training and validation losses compared to the model without JBF. This indicates that JBF effectively reduces errors during both training and validation phases, leading to relatively better model performance.

7.3 Colorizer with ResNet Backbone

In this section, we present the outcomes of our colorization model employing a ResNet backbone, emphasizing the enhancement in results due to the improved feature extraction. Additionally, we analyze the impact of applying Joint Bilateral Filtering (JBF) as a post-processing step.

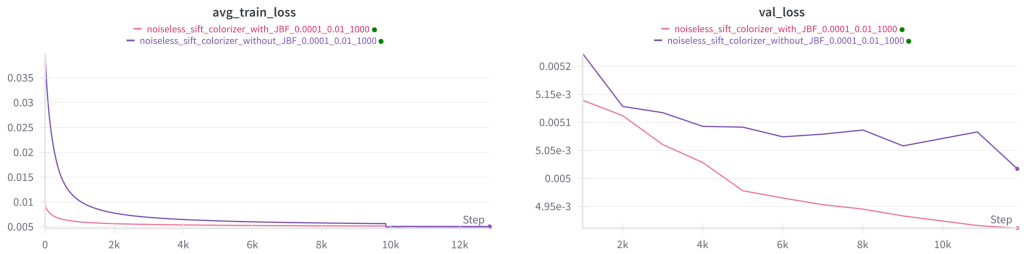

Figure 13: Train and Validation loss curves with SIFT backbone

7.3.1 Qualitative Analysis

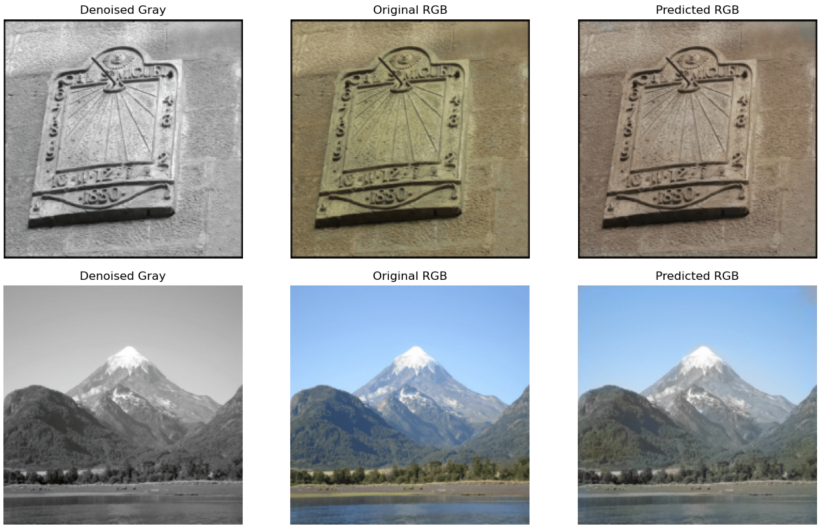

Figure 14: Colorized images with Joint Bilateral Filtering.

Figures 14 and 15 display the colorized images produced by the model with and without the application of JBF, respectively. The image processed with JBF demonstrates improved colorization quality with reduced artifacts and smoother transitions, while the image without JBF shows noticeable artifacts and less refined results.

7.3.2 Quantitative Analysis

The training and validation loss curves for both models are illustrated in Figure 16. The model utilizing the ResNet backbone consistently achieves better performance metrics, demonstrating its superiority in feature extraction compared to the SIFT backbone. Furthermore, the inclusion of JBF lowers training and validation losses, highlighting its effectiveness in reducing artifacts and enhancing the overall smoothness of the colorized images.

8 Conclusion

This project successfully demonstrates the application of a multi-step methodology to enhance grayscale image colorization, achieving visually compelling results with practical applicability. The incorporation of speckle noise during preprocessing and the exploration of various denoising methods, including Non-Local Means, Total Variation, and Wavelet Denoising, highlights the robustness of the pipeline in handling realistic distortions. The use of J-invariance for parameter optimization ensures unbiased and effective calibration of these denoisers.

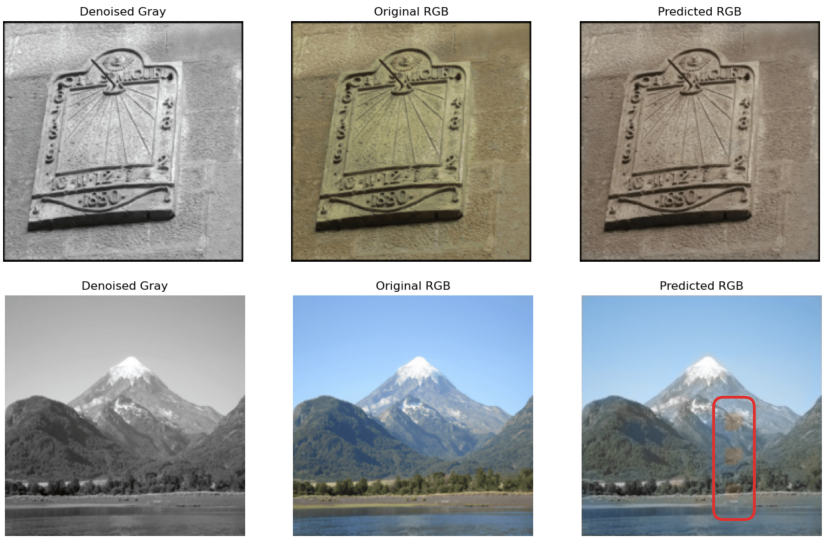

Figure 15: Colorized images without Joint Bilateral Filtering.

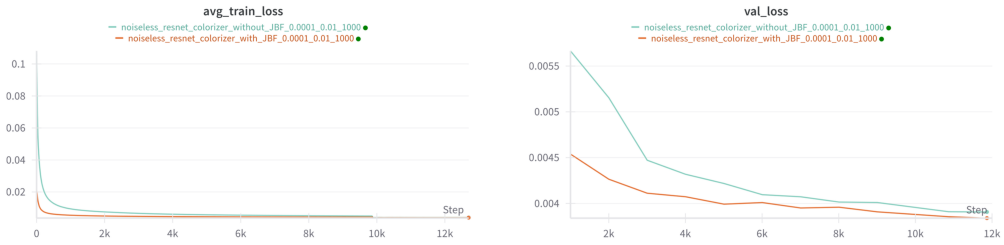

Figure 16: Train and Validation loss curves with ResNet backbone

Feature extraction through both classical (SIFT) and deep learning-based (ResNet) approaches enables a comprehensive understanding of image structures, leading to accurate chrominance prediction in the YUV color space. The Joint Bilateral Filter refines the chrominance outputs, enhancing spatial coherence and visual fidelity. Finally, the upscaling to 4K resolution using bilinear interpolation underscores the adaptability of the model to modern high-resolution display standards.

Future work could explore extending the methodology to video sequences, integrating temporal coherence, and leveraging generative models for enhanced realism. This project lays a solid foundation for further advancements in image colorization and related computer vision tasks.

References

Abien Fred Agarap. 2018. Deep learning using rectified linear units (relu). *CoRR*, abs/1803.08375.

Joshua Batson and Loic Royer. 2019. Noise2Self: Blind denoising by self-supervision. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 524–533. PMLR.

Thomas Brox and Daniel Cremers. 2007. Iterated nonlocal means for texture restoration. In *Scale Space and Variational Methods in Computer Vision*, pages 13–24, Berlin, Heidelberg. Springer Berlin Heidelberg.

A. Chambolle. 2004. An algorithm for total variation minimization and applications. *Journal of Mathematical Imaging and Vision*, 20:89–97.

S. Grace Chang, Bin Yu, and Martin Vetterli. 2000. Adaptive wavelet thresholding for image denoising and compression. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, 9 9:1532–46.

Zezhou Cheng, Qingxiong Yang, and Bin Sheng. 2016. Deep colorization. *CoRR*, abs/1605.00075.

Jerome Darbon, Alexandre Cunha, Tony F. Chan, Stanley Osher, and Grant J. Jensen. 2008. Fast nonlocal filtering applied to electron cryomicroscopy. In *2008 5th IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pages 1331–1334.

Nicholas George, C. R. Christensen, J. S. Bennett, and B. D. Guenther. 1976. Speckle noise in displays. *J. Opt. Soc. Am.*, 66(11):1282–1290.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167.

Yann Le Cun, John S. Denker, and Sara A. Solla. 1989. Optimal brain damage. In *Proceedings of the 2nd International Conference on Neural Information Processing Systems*, NIPS'89, page 598–605, Cambridge, MA, USA. MIT Press.

Chen Liu, Xiaoguang Li, Li Zhuo, Jiafeng Li, and Qingfeng Zhou. 2018. A novel speckle noise reduction algorithm for old movies recovery. In *2018 11th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pages 1–6.

Ilya Loshchilov and Frank Hutter. 2017. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101.

David G. Lowe. 2004. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110.

Georg Petschnigg, Richard Szeliski, Maneesh Agrawala, Michael Cohen, Hugues Hoppe, and Kentaro Toyama. 2004. Digital photography with flash and no-flash image pairs. *ACM Trans. Graph.*, 23(3):664–672.

Michal Podpora, Grzegorz Pawel Korbas, and Aleksandra Kawala-Janik. 2014. Yuv vs rgb-choosing a color space for human-machine interaction. In *Conference on Computer Science and Information Systems*.

Edgar Riba, Dmytro Mishkin, Daniel Ponsa, Ethan Rublee, and Gary R. Bradski. 2019. Kornia: an open source differentiable computer vision library for pytorch. *CoRR*, abs/1910.02190.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597.

Leonid I. Rudin, Stanley Osher, and Emad Fatemi. 1992. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259–268.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

Dominik Scherer, Andreas Müller, and Sven Behnke. 2010. Evaluation of pooling operations in convolutional architectures for object recognition. In *Artificial Neural Networks – ICANN 2010*, pages 92–101, Berlin, Heidelberg. Springer Berlin Heidelberg.

Evan Shelhamer, Jonathan Long, and Trevor Darrell. 2016. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1605.06211.

Carlo Tomasi and Roberto Manduchi. 1998. Bilateral filtering for gray and color images. *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, pages 839–846.

Guy Verschaffelt, Stijn Roelandt, Youri Meuret, Wendy Van den Broeck, Katriina Kilpi, Bram Lievens, An Jacobs, Peter Janssens, and Hugo Thienpont. 2016. Speckle perception and disturbance limit in laser based projectors. In *Optics, Photonics and Digital Technologies for Imaging Applications IV*, volume 9896, page 98960R. International Society for Optics and Photonics, SPIE.

Yuxin Wu and Kaiming He. 2018. Group normalization. *CoRR*, abs/1803.08494.

Kaichao You, Mingsheng Long, Jianmin Wang, and Michael I. Jordan. 2019. How does learning rate decay help modern neural networks?