Deep-learning: Colorization of Images

Elora Drouilhet, Camille L'Herminé

November 2025

1 Introduction

1.1 Presentation of the project

The primary objective is to develop a robust, LAB-space colorization model based on the following pipeline:

1. **Generator:** A deep FCN (U-Net) predicts the **AB channels** (color) given the **L channel** (lightness).
2. **Loss Function:** Implements **AB class rebalancing** via a per-pixel weight map to penalize common, desaturated colors and favor diverse colorization.
3. **Training Strategy:** GAN training combining A generator with an architecture similar to U-Net and a PatchGan Discriminator.

   The generator is trained using a combination of a **reconstruction loss** and an **adversarial loss**. These two components capture complementary aspects of the colorization task (detailed later).

1.2 Main challenges

The core challenge in image colorization is the **ambiguity** (a single grayscale value can correspond to multiple colors) and the **class imbalance** in the AB color space. Most natural images have common, desaturated colors (grays, browns, blues) but fewer vibrant ones. We used AB rebalancing technique to directly address this by applying a higher loss weight to rare colors.

1.3 Prior experiments (before choosing a final GAN model)

1.3.1 Adjusting learning rates

Discriminator overpowering the generator:

During early experiments, we observed that the discriminator consistently achieved very low loss, while the generator's adversarial was high and even increased slowly. This indicated that the discriminator was too strong and "overpowering" the generator.

To mitigate this, we applied the following strategies:

1. **Reduced discriminator learning rate from 1e-4 to 5e-5**

   This slowed down discriminator updates, giving the generator a better chance to improve.
2. **Alternate training frequency:** We experimented with updating the discriminator only once every other epoch. However, this led to worse results in terms of visual colors. The training was stable, but the discriminator had become too weak to provide meaningful gradients for the Generator. Its loss was stable but ranging from 1.3 to 2.4 which is way higher than its range in our final model.

Outcome: Slowing the discriminator's learning rate provided a stable balance, allowing the generator to produce realistic colorizations while maintaining adversarial training.

1.3.2 Soft labels helped stabilize training:

Instead of using hard labels (0 for fake, 1 for real) in the discriminator, we used *soft labels*: 0.2 for fake and 0.8 for real. This technique stabilizes training by preventing the discriminator from becoming too confident:

$$\text{Label}_{\text{real}} = 0.8, \quad \text{Label}_{\text{fake}} = 0.2$$

Effect: By tempering the discriminator's confidence, soft labels help maintain balance between the generator and discriminator, leading to more stable and effective GAN training.

1.3.3 Weighted L1 reconstruction Loss with AB Rebalancing

For a long time, our results had very greyish colors, and were mostly desaturated, which showed that the generator was making easy predictions (mean values).

The standard L1 loss predicts the per-pixel absolute difference between the true and predicted AB channels:

$$\mathcal{L}_{L1} = \frac{1}{N}\sum_{i=1}^{N} \big|A_{\text{pred}}(i) - A_{\text{true}}(i)\big| + \big|B_{\text{pred}}(i) - B_{\text{true}}(i)\big|$$

- **Problem:** Most natural images contain desaturated colors (grays, browns, soft blues). Rare, vibrant colors appear less frequently.
- **Effect:** L1 loss encourages the network to predict the *average* color at each pixel. Since desaturated colors dominate, this leads to **desaturated predictions**.
- **Solution:** Introducing per-pixel AB rebalancing weights:

$$\mathcal{L}_{L1}^{\text{weighted}} = \frac{1}{N}\sum_{i=1}^{N} W_{\text{map}}(i)(\big|A_{\text{pred}}(i) - A_{\text{true}}(i)\big| + \big|B_{\text{pred}}(i) - B_{\text{true}}(i)\big|)$$

  increases the penalty for mispredicting rare colors, encouraging **more vibrant and diverse color outputs**.

To mitigate the imbalance of AB values in natural images, we made some research and found a technique used by Zhang, Isola Efros (2016) in "Colorful Image Colorization". We applied their idea of a per-pixel reweighting in the reconstruction loss. Let p_{ab} be the empirical probability of an AB bin and ε a small constant to avoid division by zero. The raw inverse-frequency weight is

$$w_{ab}^{\text{raw}} = \frac{1}{p_{ab} + \varepsilon}.$$

We then normalize and clamp the weights to a controlled range:

$$w_{ab} = \frac{w_{ab}^{\text{raw}}}{\max(w_{ab}^{\text{raw}})} \cdot w_{\max}, \qquad w_{\max} = 5.0, \quad w_{\min} = 0.5.$$

Earlier experiments using the range $[0, 1]$ caused the **weighted L1 loss to collapse toward zero**, which made the reconstruction signal too weak for training. Expanding the weights to $[0.5, 5.0]$ ensures that rare AB colors receive sufficiently strong gradients, while common colors remain proportionally less emphasized, preserving numerical stability.

1.3.4 Effect of the reconstruction weight λ_{recon}

To find lambda, we processed by trial and error. We tried lambda equal to 200, 100, 70, 10 and 1.

Since, the Generator is trained with a combination of reconstruction and adversarial terms:

$$\mathcal{L}_G = \lambda_{\text{recon}}\,\mathcal{L}_{\text{L1}} + \mathcal{L}_{\text{adv}}.$$

Choosing the value of λ_{recon} determines the balance between *accuracy* (L1) and *realism* (GAN).

We saw that the two extreme settings behave poorly:

Instability of the training for low values ($\lambda_{\text{recon}} = 10$). When λ_{recon} is too small, the L1 loss contributes very little to the Generator objective. The optimization is therefore dominated by the adversarial term:

$$\mathcal{L}_G \approx \mathcal{L}_{\text{adv}}.$$

The adversarial loss alone provides a *weak and highly non-convex* gradient signal. Without the L1 term anchoring the predictions to the ground truth AB values, the Generator becomes free to make large jumps in color space to attempt to fool the Discriminator. This leads to:

- **Unstable training dynamics**: the Generator oscillates, overshoots, or collapses to extreme color modes.
- **Inaccurate or unrealistic colors**: since pixel-wise accuracy is barely penalized, the colors drift away from the true distribution.
- **Higher risk of mode collapse**: the model may reproduce only a limited subset of colors that temporarily satisfy the Discriminator.

Thus, $\lambda_{\text{recon}} = 10$ or 1 overly amplifies the adversarial component, resulting in instability of the training, and it led to overpowering discriminator.

Desaturation for large values ($\lambda_{\text{recon}} = 100$). When λ_{recon} is too large, the Generator is driven almost entirely by the L1 term:

$$\mathcal{L}_G \approx \lambda_{\text{recon}} \mathcal{L}_{\text{L1}}.$$

The L1 loss has a regression-to-the-mean objective. For pixels where multiple colors are plausible, the L1 minimizer is the median or mean of all options, which lies near the neutral chrominance point $(A, B) = (0, 0)$ corresponding to gray. A high reconstruction weight therefore encourages

$$(A_{\text{pred}}, B_{\text{pred}}) \approx (0, 0),$$

resulting in overly smooth, desaturated, or muted outputs. The adversarial term cannot compensate because its influence is suppressed by the large λ_{recon}.

Balanced setting ($\lambda_{\text{recon}} = 70$). Empirically, $\lambda_{\text{recon}} = 70$ provided a stable compromise: the L1 term keeps colors aligned with the ground truth, while the adversarial term still injects texture realism and preserves saturation. This setting avoids both the instability of low λ and the desaturation of high λ, resulting in the highest perceptual quality.

2 Data preparation

We initially experimented with the low-resolution **CIFAR-10** dataset. For better performance and to handle varied resolutions, we migrated to the **COCO** dataset for training and testing the U-Net-GAN architecture. We leveraged the FiftyOne library for convenient data download and management.

2.1 AB Class rebalancing weights

To overcome the imbalance, as mentioned before, we compute rebalancing weights using COCO training samples:

$$w(a, b) = \frac{1}{\tilde{p}(a, b)}$$

where $\tilde{p}(a, b)$ is the empirical probability of the AB pair being sampled, calculated from a 2D histogram of the AB channels binned into $N \times N$ discrete classes. These weights are incorporated into the loss function as a per-pixel weight map, W_{map}.

3 Architecture implemented

3.1 Deep FCN U-Net style and PatchGAN

The final model is a GAN defined by the components shown in Figure **??**:

Generator (G)

G is a deep encoder-decoder structure (U-Net style) with 3×3 convolutions and skip connections.

- **Input:** Normalized L channel ($1 \times 256 \times 256$).
- **Encoder:** Sequential ConvBlocks (Conv $\rightarrow$ BatchNorm $\rightarrow$ ReLU) with strided convolutions for downsampling (s2). Channel progression: $1 \rightarrow 64 \rightarrow 128 \rightarrow 256 \rightarrow 512$.
- **Bottleneck:** Deepest layer with 1024 channels, increasing representational capacity.
- **Decoder:** Uses **Nearest-Neighbor Upsampling** (which differs from classical U-Net that uses learnable transposed convolution instead) followed by a ConvBlock to reduce checkerboard distortions on the output. Skip connections from the Encoder layers are concatenated.
- **Output:** Final Conv $\rightarrow$ Tanh activation, yielding the predicted AB channels ($2 \times 256 \times 256$) normalized to $[-1, 1]$.

Discriminator (D)

D is a **PatchGAN** that assesses the local realism of the generated image.

- **Input:** Concatenated true or fake LAB image ($3 \times H \times W$).
- **Architecture:** Series of 4×4 strided convolutional layers (s2) with LeakyReLU.
- **Output:** A $1 \times N \times M$ feature map (PatchGAN map) where each element represents the realness of a corresponding image patch.

Input Channels: The discriminator takes 3 channels: $\mathbf{L}_{\text{input}}$, $\mathbf{A}_{\text{pred/true}}$, and $\mathbf{B}_{\text{pred/true}}$.
This forces the discriminator to check the consistency of the predicted colors with the grayscale input, rather than just checking if the colors look real in isolation.

PatchGAN output: the final layer outputs a 16×16 grid (or "patch") of scores instead of a single scalar. Each score represents the probability that the corresponding 16×16 region of the input image is "real." This forces the Generator to produce realistic details locally across the entire image. Without this, a simple scalar output would only check global realism, allowing the Generator to create blurry, locally inconsistent images.

We tried different depths for the Discriminator: 3, 4, and 5 layers. 5 layers made it too strong and overpowering the generator which led to unstable training. 3 layers led to a weak discriminator which couldn't provide a strong enough signal to the Generator, and led to sepia colored images.

3.2 Optimization strategy

The loss functions guide the training process:

- **Reconstruction Loss ($\mathcal{L}_{L1}$):** Weighted L1 Loss (Mean Absolute Error, MAE):

$$\mathcal{L}_{L1}(\mathbf{A}_{pred}, \mathbf{A}_{true}, W_{map}) = \frac{1}{N} \sum_{i=1}^{N} W_{map}(i) \cdot ||\mathbf{A}_{pred}(i) - \mathbf{A}_{true}(i)||_1$$

- **Adversarial Loss ($\mathcal{L}_{GAN}$):** Standard binary cross-entropy with logits (BCEWithLogitsLoss) combined with **Label Smoothing** (**0.8**/**0.2**) for stability.

- **Generator Total Loss ($\mathcal{L}_G$):**

$$\mathcal{L}_G = \lambda_{recon} \cdot \mathcal{L}_{L1} + \mathcal{L}_{GAN}$$

  We set $\lambda_{\mathbf{recon}} = \mathbf{70}$ to prioritize accurate color prediction while $\mathcal{L}_{GAN}$ enhances perceptual realism.

Generator loss: $loss_G = loss_Gadv + lambda_recon * loss_Grecon$.
This is the classic loss of a conditional GAN for image translation (like Pix2Pix).
Without the reconstruction loss ($\mathbf{L_1}$), the Generator would produce plausible but likely inaccurate colors. Without the adversarial loss ($\mathbf{L_{adv}}$), the Generator would minimize L1 loss, producing desaturated, blurry results.
$\lambda_{\text{recon}} = 70$: The high weight on the reconstruction loss ensures the predicted colors are factually correct, while the adversarial loss ($\mathbf{L}_{\text{adv}}$) ensures the predicted colors look perceptually realistic.Soft/Noisy Labels (0.8, 0.2): Instead of strict 1.0 (real) and 0.0 (fake), soft labels are used to prevent the Discriminator from becoming too confident too early, which can cause the Generator's gradients to vanish. Without soft labels, the GAN training can become unstable and fail to converge.

Reconstruction loss

To ensure that the generated color channels match the ground-truth target, we use an L_1 reconstruction loss between the predicted AB channels $\hat{y}$ and the ground truth y:

$$\mathcal{L}_{\text{rec}} = \|y - \hat{y}\|_1.$$

The L_1 metric encourages pixel-wise accuracy and usually produces sharper reconstructions than L_2, which tends to overly penalize larger errors and results in blurrier outputs. This loss ensures that the generator preserves the structure of the image and produces realistic, spatially consistent colors.

Adversarial loss

The adversarial component follows the PatchGAN framework, where the discriminator D attempts to distinguish real LAB images from generated ones while the generator G attempts to fool it. The adversarial loss for the generator is given by:

$$\mathcal{L}_{\text{adv}} = -\mathbb{E}_x \left[\log D(x, G(x))\right],$$

where x is the L-channel input and $G(x)$ is the generated AB output. This objective pushes the generator to produce outputs that lie on the manifold of real color images, encouraging realism and natural color distribution.

The reconstruction loss alone forces the generator to approximate the ground truth as closely as possible, but it often leads to **desaturated or greyish** colorizations due to the pixel-wise averaging effect. Conversely, the adversarial loss encourages more **vivid and realistic** colors but does not provide strong spatial supervision, making the generator susceptible to create neon spots.

By combining the two:

$$\mathcal{L}_G = \lambda_{\text{rec}}\,\mathcal{L}_{\text{rec}} + \lambda_{\text{adv}}\,\mathcal{L}_{\text{adv}},$$

the model benefits from the strengths of both components:

- **Reconstruction loss**: enforces accuracy and structure.
- **Adversarial loss**: enforces realism and vibrant colors.

This hybrid objective leads to outputs that are both faithful to the original image and visually believable.

Validation: Uses Peak Signal-to-Noise Ratio (PSNR) for objective evaluation and visual inspection for perceptual quality.

4 Training results and analysis

We trained the GAN for 27 epochs on the COCO dataset. Figure 1 shows the evolution of the main training metrics.

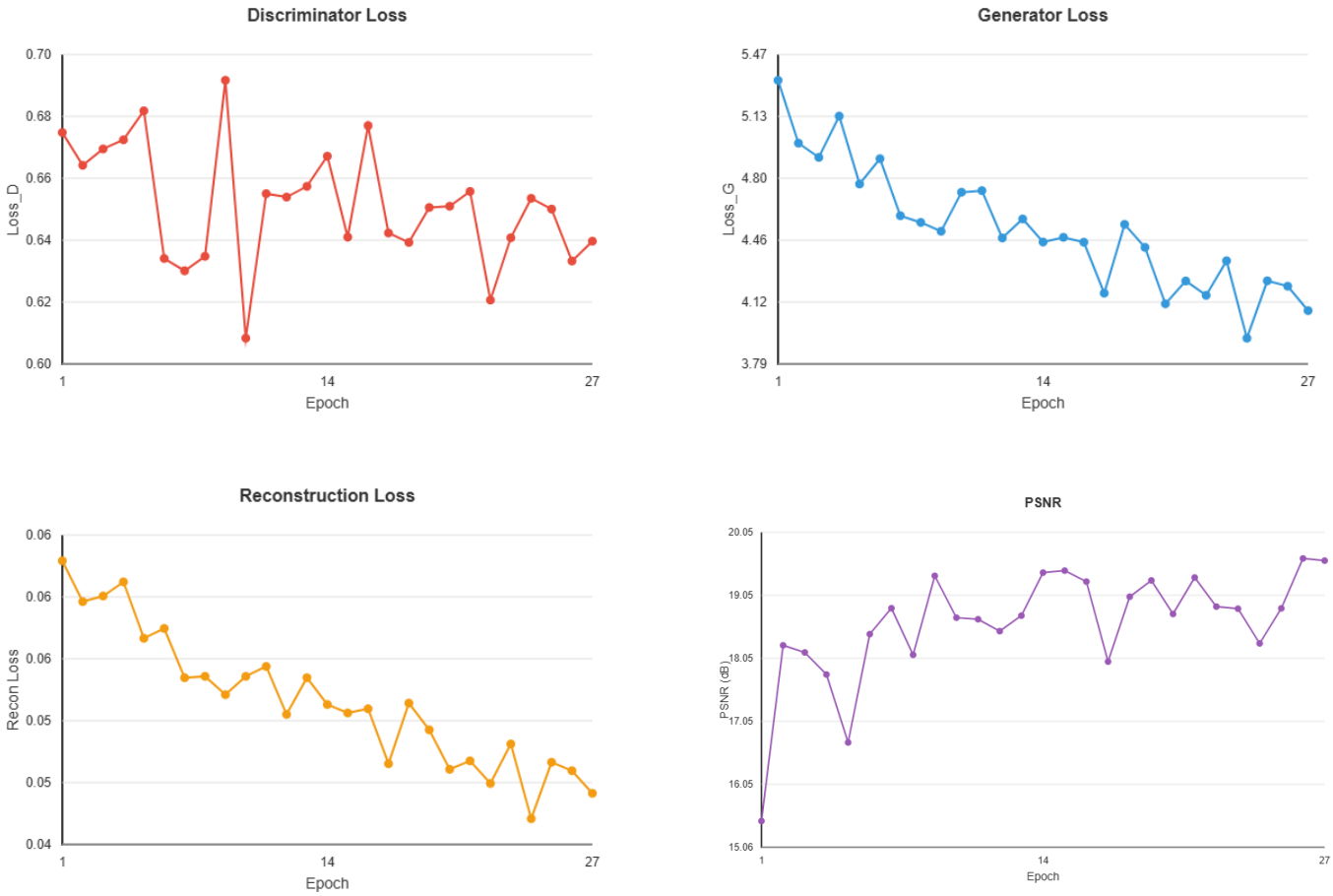


Figure 1: Training metrics evolution over 27 epochs. (a) Discriminator loss remains stable around 0.6–0.7. (b) Generator loss decreases from 5.27 to 3.93. (c) Reconstruction loss improves by 69% from 0.144 to 0.044. (d) PSNR increases from 15.47 to 19.63 dB with oscillations.

4.1 Discriminator stability

The discriminator loss remained remarkably stable around 0.6–0.7 throughout the entire training process, indicating a well-balanced adversarial setup. This stability was not achieved by chance but resulted from careful architectural and hyperparameter choices. First, we employed asymmetric learning rates with $lr_D = 5 \times 10^{-5}$ for the discriminator and $lr_G = 4 \times 10^{-4}$ for the generator. This eight-fold difference in learning rates ensures that the discriminator learns more slowly, preventing it from becoming too strong and overpowering the generator—a common failure mode in GAN training. Second, we applied label smoothing by using soft labels (0.8 for real, 0.2 for fake) instead of hard binary labels (1.0/0.0). This technique prevents the discriminator from making overconfident predictions, which would produce vanishing gradients for the generator and halt learning. Finally, our PatchGAN architecture outputs a 16×16 map of local realness scores rather than a single global scalar, providing spatially localized feedback that encourages the generator to produce realistic details across the entire image rather than just achieving global coherence.

The consistent discriminator loss around 0.6–0.7 indicates that D maintains sufficient difficulty to provide meaningful adversarial gradients to G without becoming so strong that it destabilizes training. This sweet spot is critical for successful GAN convergence.

4.2 Generator convergence and loss decomposition

The total generator loss decreased from 5.27 at epoch 1 to 3.93 at epoch 27, representing an approximately 25% reduction. However, examining the two components of this loss reveals distinct learning dynamics that illuminate how the model improves over time.

The weighted L1 reconstruction loss exhibited the most dramatic improvement, decreasing from 0.144 at epoch 1 to 0.044 at epoch 27—a striking 69% reduction. The learning curve reveals two distinct phases. During the first five epochs, the reconstruction loss dropped rapidly from 0.144 to approximately 0.057, indicating that the generator quickly learned coarse color distributions and global image structure. This initial rapid learning phase is characteristic of well-initialized deep networks with effective gradient propagation. Following this, epochs 6 through 27 showed a more gradual convergence as the model refined fine-grained color details and improved accuracy on edge cases. The smooth, monotonic decrease without significant oscillations demonstrates that the high reconstruction weight $\lambda_{\text{recon}} = 70$ successfully anchored the generator's predictions to ground truth colors throughout training.

The AB class rebalancing weights played a crucial role in this convergence pattern. Without rebalancing, the L1 loss would encourage the generator to predict average or median colors, leading to desaturated outputs dominated by common grays and browns. The rebalancing scheme assigns higher loss weights to rare, vibrant AB color combinations, explicitly penalizing the model for neglecting these underrepresented colors. The steep and consistent decrease in reconstruction loss indicates that the generator successfully learned to predict diverse, saturated colors rather than collapsing to safe, desaturated predictions.

In contrast to the reconstruction loss, the adversarial component remained relatively stable around 0.7–0.9 throughout training, exhibiting minor fluctuations rather than a monotonic decrease. This behavior is expected and healthy in GAN training, as the adversarial loss represents an ongoing competitive game between generator and discriminator rather than a fixed regression target. The generator does not simply minimize this loss to zero; instead, it seeks to maintain a balance where generated images are plausible enough to partially fool the discriminator. The stable range around 0.7–0.9 indicates that the generator consistently produces outputs that challenge the discriminator without completely defeating it, maintaining productive adversarial tension throughout training.

The combination of dramatically improving reconstruction loss with stable adversarial feedback validates our choice of $\lambda_{\text{recon}} = 70$. This high weight ensures that color accuracy dominates the optimization landscape while still allowing the adversarial term to inject perceptual realism. The U-Net skip connections proved essential for propagating gradients effectively through the deep encoder-decoder architecture, enabling simultaneous optimization of both loss components even with this aggressive weighting scheme.

4.3 PSNR evolution and interpretation

The PSNR metric improved from 15.47 dB at epoch 1 to a peak of 19.63 dB at epoch 26, representing a gain of approximately 4.2 dB. However, the PSNR trajectory was not monotonic, exhibiting notable oscillations throughout training—for instance, dropping to 16.72 dB at epoch 5 before recovering, and fluctuating between 18 and 19.5 dB during the middle epochs.

These oscillations are expected and represent healthy GAN training dynamics rather than instability. Unlike pure regression models that monotonically minimize pixel-wise error, GANs balance two competing objectives: reconstruction accuracy (pixel-perfect color matching) and adversarial realism (perceptually convincing appearance). During training, the generator occasionally explores color choices that are more perceptually realistic but less pixel-perfect, causing temporary PSNR drops. For example, the generator might produce a vibrant, saturated blue sky that looks natural to human observers but differs slightly in hue from the ground truth, resulting in lower PSNR despite improved visual quality. This exploratory behavior is driven by the adversarial loss component, which encourages the generator to sample from the manifold of realistic images rather than simply averaging over possible colors.

Interestingly, the PSNR oscillations are not mirrored in the reconstruction loss curve, which decreases smoothly and monotonically. This discrepancy arises from several factors. First, PSNR is computed in RGB space after LAB-to-RGB conversion, while the reconstruction loss operates directly on AB channels in LAB space. The nonlinear color space transformation can amplify or attenuate certain prediction errors. Second, PSNR uses a logarithmic scale based on mean squared error and is particularly sensitive to peak errors, whereas L1 loss uniformly penalizes all deviations. Third, PSNR measures global image quality, while the reconstruction loss is computed on a per-pixel basis with rebalancing weights. A single vivid but slightly inaccurate color choice can disproportionately affect PSNR while contributing modestly to the weighted L1 loss. The smooth reconstruction loss indicates consistent improvement in AB prediction accuracy, while PSNR fluctuations reflect the generator's exploration of perceptually realistic color distributions during adversarial training.

It is crucial to recognize the limitations of PSNR as an evaluation metric for GANs. PSNR is a pixel-level metric that correlates poorly with human perceptual quality for generative models. A colorization with higher PSNR may appear desaturated or blurry to human observers, while a slightly lower PSNR may indicate sharper textures and more vibrant

colors. For this reason, we prioritize qualitative visual inspection alongside PSNR for model evaluation, recognizing that human judgment of color naturalness and image quality is the ultimate arbiter of success in colorization tasks.

4.4 Summary

The training curves collectively demonstrate successful GAN convergence and validate our architectural and hyperparameter choices. The discriminator maintained stable loss around 0.6–0.7 throughout training, providing consistent adversarial feedback without overpowering the generator. The generator exhibited clear two-phase learning: rapid coarse learning during epochs 1–5 (reconstruction loss dropping from 0.144 to 0.057) followed by gradual refinement through epoch 27 (reaching 0.044), representing a total improvement of 69%. The total generator loss decreased by 25%, reflecting the combined benefits of improved color accuracy and maintained adversarial realism.

The PSNR improved from 15.47 dB to a peak of 19.63 dB, with expected oscillations that reflect the generator's exploration of perceptually realistic solutions. The divergence between smooth reconstruction loss convergence and oscillating PSNR highlights the fundamental difference between optimizing for pixel-wise accuracy versus perceptual realism—the central tension that our hybrid loss function $\mathcal{L}_G = \lambda_{\text{recon}}\mathcal{L}_{\text{L1}} + \mathcal{L}_{\text{adv}}$ is designed to balance.

These results validate our key design decisions: the U-Net architecture with skip connections for effective gradient propagation, the PatchGAN discriminator for localized realism feedback, AB class rebalancing for diverse color learning, asymmetric learning rates for stable adversarial dynamics, and $\lambda_{\text{recon}} = 70$ for the optimal balance between accuracy and realism.

5 Evaluation

- **PSNR calculation:** The PSNR (Peak Signal-to-Noise Ratio) is calculated on the reconstructed RGB image, not the LAB values. This is essential because PSNR is an image quality metric and should operate in the standard perceptual space (RGB, normalized to [0, 1]).
- **Visualizations:** Single-sample comparisons (`visualize_gan_output`) show the input grayscale L, the ground truth color, and the GAN-generated color, important for identifying failure modes like color bleeding or desaturation. We can observe that most of the colorized pictures have realistic colors, mostly respecting the shapes. We have therefore visually satisfying results. We can however notice some slight color bleeding (below the sky or around faces) , some over saturation of skin color sometimes and that for some pictures the background are sepia or greyish.

  But we observe limitations of the models on very few images which would require a more complex semantic understanding that the model doesn't have and gives weird outputs.

  - **Lack of semantic colorization:** In the tennis court sample, the model fails to identify the court surface as green. Similarly, the mountains in the first examples lacks the distinct color found in the ground truth, and shouldn't be blue like the sky.
  - **neon spots** the WGAN-GP exhibits neon spots, particularly in ambiguous regions: In the tennis court sample, the generator produces red dots and green patches.

The failure of the standard GAN in this context highlights the vanishing gradient problem inherent to the sigmoid-based discriminator. As the discriminator improves, the gradients passed to the generator vanish, preventing the model from learning the subtle textural differences required for vibrant colorization. Consequently, the L_1 loss component dominates the optimization, forcing the model into a conservative local minimum where it predicts the statistical average of all possible colors (brown/gray) rather than a specific, vibrant mode.

6 Wasserstein GAN

6.1 Methodological Transition: From GAN to WGAN-GP

For this project, we decided to compare the results of the standard Generative Adversarial Network (GAN) architecture with those of a Wasserstein GAN with Gradient Penalty (WGAN-GP).

6.2 The Wasserstein Distance and Kantorovich-Rubinstein Duality

The fundamental objective of the WGAN is to minimize the Earth Mover's (Wasserstein-1) distance between the real data distribution $\mathbb{P}_r$ and the generated model distribution $\mathbb{P}_g$. The primal form of the Wasserstein distance represents the minimum cost to transport mass from one distribution to another, which is computationally intractable for high-dimensional image data.

To overcome this, we utilize the **Kantorovich-Rubinstein Duality**, which allows us to express the Wasserstein distance as a maximization problem:

$$W(\mathbb{P}_r, \mathbb{P}_g) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{\tilde{x} \sim \mathbb{P}_g}[f(\tilde{x})] \tag{1}$$

where the supremum is taken over the set of all 1-Lipschitz continuous functions f. In our architecture, the Critic network C (formerly the Discriminator) acts as this function f.

This duality formulation justifies our loss function design: rather than outputting a probability of authenticity (as in standard GANs), the Critic outputs a scalar score. We train the Critic to maximize the difference between the mean scores of real images and generated images:

$$L_{Critic} = \underbrace{\mathbb{E}[C(\tilde{x})]}_{\text{Fake Score}} - \underbrace{\mathbb{E}[C(x)]}_{\text{Real Score}} \tag{2}$$

By maximizing this difference, the Critic approximates the Wasserstein distance, providing a meaningful gradient to the Generator even when the generated samples remain far from the data manifold.

6.3 The Lipschitz Constraint and Gradient penalty

The validity of Equation (1) relies strictly on the condition that the Critic C is a **1-Lipschitz function**. This implies that the norm of the gradient with respect to the input must be bounded:

$$\|\nabla_x C(x)\|_2 \leq 1 \tag{3}$$

Without this constraint, the Critic could arbitrarily scale its weights to make the score difference infinite, rendering the loss metric meaningless and destabilizing training.

We thus implemented the **Gradient Penalty (GP)** method. We apply a soft penalty to the gradients of the Critic's output with respect to its input. We sample interpolated points $\hat{x}$ along straight lines between real and generated data points. The Gradient Penalty term is added to the loss function:

$$L_{GP} = \lambda \cdot \mathbb{E}_{\hat{x} \sim \mathbb{P}_{\hat{x}}} \left[(\|\nabla_{\hat{x}} C(\hat{x})\|_2 - 1)^2 \right] \tag{4}$$

This term penalizes the model if the gradient norm moves away from 1, thereby enforcing the Lipschitz condition smoothly without restricting the capacity of the neural network. This results in faster convergence and higher stability compared to weight clipping.

7 Overview of Implementation Changes

Transitioning from a standard Deep Convolutional GAN (DCGAN) to a Wasserstein GAN with Gradient Penalty (WGAN-GP) required fundamental changes to the network architecture, loss functions, and the training loop. This section details exactly how the code was modified to satisfy the theoretical constraints of the Wasserstein metric.

7.1 Architectural Modifications

7.1.1 The Discriminator (Renamed to Critic)

The most significant architectural changes occurred in the Discriminator, which is renamed to "Critic" in WGAN literature to reflect its new role: it no longer classifies (0 vs 1) but scores the "realness" of an image.

1. **Normalization Layer :** `nn.BatchNorm2d`. replaced with `nn.InstanceNorm2d`. (for the Gradient penalty to be valid)
2. **Removal of Final Activation:** The Wasserstein distance requires the output to be an unbounded scalar score $D(x) \in (-\infty, \infty)$, whereas the standard GAN required a probability $D(x) \in [0, 1]$.

7.1.2 The Generator

Unchanged.

7.2 Loss Function Transformations

We completely removed the Binary Cross Entropy (BCE) loss and introduced the Wasserstein loss with a regularization term.

New Loss (Wasserstein):

$$\mathcal{L} = \underbrace{\mathbb{E}[D(\tilde{x})]}_{\text{Fake Mean}} - \underbrace{\mathbb{E}[D(x)]}_{\text{Real Mean}} + \lambda(\text{GP})$$

7.3 The Gradient Penalty (GP)

We introduced a new function, `compute_gradient_penalty`, which enforces the 1-Lipschitz constraint.

1. **Interpolation:** A random point $\hat{x}$ is sampled along the line between a real image pair and a fake image pair.
2. **Gradient Calculation:** We compute gradients of the Critic's output with respect to $\hat{x}$.
3. **Penalty:** We penalize the model if the L2 norm of this gradient moves away from 1.

7.4 Training Loop Adjustments

The training loop required structural changes to accommodate the different learning dynamics of the WGAN.

7.4.1 Critic-Generator Ratio (n_{critic})

- **Former Code:** 1 update of D for every 1 update of G.
- **New Code: 5 updates of C** for every 1 update of G.
- **Reasoning:** In a WGAN, the Critic must differ significantly from the Generator to provide a reliable approximation of the Wasserstein distance. If the Critic is not optimal, the gradient it provides to the Generator is inaccurate. We train the Critic more frequently to ensure it stays ahead of the Generator.

7.5 Training Dynamics and Hyperparameter Scheduling

The training of the WGAN-GP was conducted using a dynamic hyperparameter schedule to address the specific challenges of the colorization task: specifically, the trade-off between accuracy (governed by the L_1 loss) and realism (governed by the adversarial loss). The training process was divided into three distinct phases.

7.5.1 Phase 1: Structural Initialization (Epochs 0–18)

Configuration: $\lambda_{L1} = 70$ (we started with the same value as for the normal GAN)

The initial phase focused on stabilizing the U-Net generator and learning semantic segmentation (distinguishing sky, grass, and objects). To enforce strict spatial consistency, a high reconstruction weight of $\lambda_{L1} = 70$ was utilized.

By Epoch 18, the model achieved a high Peak Signal-to-Noise Ratio (PSNR ≈ 21.4 dB). However, qualitative inspection revealed regression to the mean. The Generator minimized the L_1 penalty by predicting safe, desaturated brownish-gray tones. The output images lacked the chromatic vibrancy required for perceptual realism.

7.5.2 Phase 2 (Epochs 19–25)

Configuration: $\lambda_{L1} = 10$

To force the Generator out of the conservative local minimum established in Phase 1, we implemented a drastic reduction of the reconstruction weight to $\lambda_{L1} = 10$. This effectively freed somehow the Generator, prioritizing the colors over pixel-perfect accuracy.

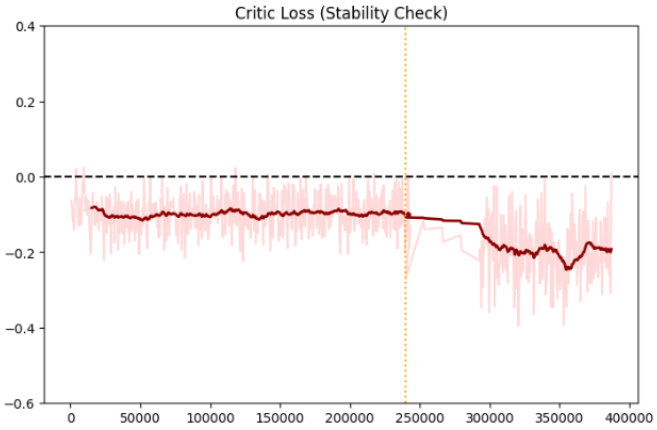
The model learned that the Critic penalizes desaturated colors since it output vibrant color like green.

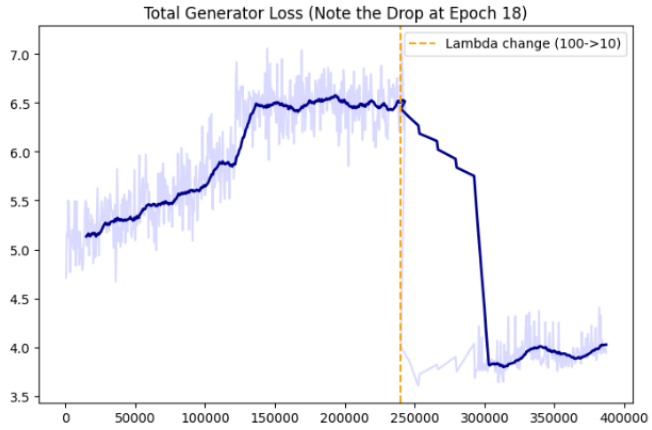
7.5.3 Phase 3: Balanced Refinement (Epochs 26-30)

Configuration: $\lambda_{L1} = 30$

While Phase 2 successfully introduced chromatic variance, the low λ_{L1} led to occasional neon spots (e.g., red stripes on a green court, red dots on zebras faces) and excessive noise and bleeding: a skater jumping in the sky became completely blue. There was no respect of pixel structure by the colors. To stabilize the training for the final convergence, we implemented a re-augmentation of the hyperparameter to $\lambda_{L1} = 30$ starting at Epoch 26.

This setting serves as a middle ground: it provides sufficient constraints to correct semantic errors (preventing the Generator from coloring grass red) while remaining loose enough to preserve the vibrancy and texture gained during the previous phase. The goal of this final phase was to allow the Generator to increase the magnitude of its predictions (saturation) without losing structural integrity. It did since the grass of the tennis court became green, but a dark green instead of the ground-truth brighter green. This shows a cautious strategy from the Generator as a brighter one could end up with a bigger loss. We'd need more computational power to train more epochs and see if with $\lambda = 30$ the model can reach better magnitude for the colors (this model trained already over 10 hours on colab).





7.6 Analysis of training losses

The evolution of the Generator's loss functions over the complete training period (Epochs 1–27) reveals two distinct operational regimes, demarcated by the hyperparameter adjustment from Epoch 18 on.

During the initialization phase, the total Generator loss ($\mathcal{L}_G$) showed a continuous, monotonic increase, rising from approximately 5.0 to 11.5. This trend was driven almost entirely by the Adversarial component ($\mathcal{L}_{adv}$), which drifted from ≈ 0.1 to ≈ 5.6. As the Critic learned to identify the "sepia" artifacts of the Generator, it assigned increasingly negative scalar scores to the generated samples. Simultaneously, the Reconstruction Loss ($\mathcal{L}_{recon}$) remained tightly constrained (≈ 0.04), confirming that the model prioritized structural safety over textural risk-taking.

At Epoch 18, the reconstruction weight was reduced by an order of magnitude.

1. The total Generator loss dropped sharply to the $2.5 - 4.5$ range. This step-change is not a result of sudden learning, but a direct mathematical consequence of reducing the weighted penalty term ($\lambda_{L1} \cdot \mathcal{L}_{recon}$).
2. **Adversarial Stabilization:** Crucially, the Adversarial Loss ($\mathcal{L}_{adv}$) ceased its rapid upward drift and stabilized in the range of $1.0 - 3.0$. This plateau indicates that the Generator has successfully adapted to the change. By injecting chromatic variance (vibrant colors) into the output, the Generator is now competitively matching the Critic's expectations, preventing the Wasserstein distance estimate from diverging further.
3. Despite the relaxed constraints, $\mathcal{L}_{recon}$ did not explode, remaining stable at $\approx 0.035 - 0.050$. This confirms that the Generator has retained the structural segmentation learned in Phase I while utilizing the freedom of Phase II to refine texture.

7.7 Evaluation results

The evolution of the Peak Signal-to-Noise Ratio (PSNR) throughout the training process shows a trade-off between statistical fidelity and perceptual realism.

The model achieved a rapid convergence to a high baseline PSNR of **21.35 dB** by Epoch 5. This early saturation correlates with the dominance of the L_1 reconstruction loss during the initial phase. It indicates that the Generator effectively utilized the U-Net's skip connections to resolve semantic segmentation tasks (e.g., assigning blue to sky regions) and minimize pixel error.

Following the hyperparameter adjustment at Epoch 18 ($\lambda_{L1} 70 \rightarrow 10$), the PSNR metric was stable, fluctuating narrowly between 21.2 dB and 21.5 dB, despite the increase in adversarial Loss.

8 Comparative Analysis: WGAN-GP vs. Standard GAN

Following the hyperparameter adjustment and extended training, a comparative analysis reveals distinct behavioral differences between the WGAN-GP model and the baseline Standard GAN. The WGAN-GP demonstrates superior spatial coherence and semantic precision, albeit with some bias towards sepia tones. Standard GAN has more vibrant colors but is less precise pixel-wise, despite a high lambda. It is overall good but sometimes creates neon spots when complex semantic understanding (tennis field).

8.1 Semantic color separation and spatial coherence

The most significant qualitative improvement observed in the WGAN-GP is its ability to disentangle foreground and background elements with high pixel-wise precision.

- WGAN-GP better adheres to semantic boundaries. For example, in samples depicting people, the faces are rendered in a distinct, skin color, while their clothes retain their own tones. There is minimal "color bleeding" across edges contrary to the normal GAN.
- We attribute this precision to the extended initialization phase (Epochs 1–18) characterized by a high reconstruction weight ($\lambda_{L1} = 70$). During this phase, the pixel-wise L_1 loss dominated the optimization landscape. Since L_1 heavily penalizes spatial misalignment, it acted as a strong regularizer, effectively treating the early training as a supervised segmentation task. This forced the Generator to leverage the U-Net's skip connections to preserve high-frequency structural details before the adversarial loss began driving the model towards vibrant colorization.

8.2 Saturation Dynamics

A counter-intuitive observation is the initial conservatism of the WGAN-GP compared to the vibrancy of the Standard GAN.

8.2.1 The Standard GAN: optimization via deception

The Standard GAN, utilizing a sigmoid activation and Binary cross entropy (BCE), operates on a probability manifold ($D(x) \in [0, 1]$). Its failure mode is best understood through the lens of binary classification mechanics.

- Exploiting Adversarial Vulnerabilities: The Generator's objective is solely to "trick" the Discriminator into outputting a 1 (real). It does not need to produce a semantically valid image to achieve this; it only needs to find a pattern that activates the Discriminator's features. Consequently, the Generator learns to produce high-frequency artifacts (such as checkerboard patterns or neon noise) that act as trigger for the Discriminator. If neon red dots successfully flips the Discriminator's decision boundary, the Generator outputs them, disregarding the semantic absurdity of the result.

  Because the Standard Discriminator only outputs a probability (Real vs. Fake) rather than a structural score, it can be easily "tricked" by these adversarial patterns. The Generator discovers that splashing a neon checkerboard pattern or specific chromatic artifacts (e.g., bright red dots) is a computationally cheaper way to force the Discriminator to output "1" (Real) than actually learning to render the complex texture of a tennis field.
- **Gradient Instability and Artifacts:** When the Standard Discriminator is confident (outputting values near 0 or 1), the gradient of the Sigmoid function approaches zero (vanishing gradients). To maintain a non-zero gradient flow, the Generator is incentivized to produce extreme pixel values (saturation spikes). Unlike the WGAN, which provides smooth gradients even for poor samples, the Standard GAN often forces the Generator to over-saturate specific features to escape the zone of zero gradients, resulting in the characteristic "red dot" artifacts observed in a few output.

8.2.2 WGAN-GP: Transition through the "Dark/Saturated" Bias

In contrast, the WGAN-GP currently exhibits a "dark/saturated" bias. This is linked to the transition phase where the Generator learns to push values away from zero (gray) in the ab color channels to satisfy the Critic, but has not yet perfectly calibrated the colors vs luminance.

- While visually imperfect, this dark bias is a positive indicator of stable adversarial learning. It proves that the Gradient Penalty is successfully preventing the vanishing gradient problem. The Generator is minimizing the Earth Mover's distance through gradual, structurally sound improvements rather than learning "cheat codes" (noise patterns).

8.3 Theoretical difference

The divergent behaviors of the two models can be explained by the fundamental differences in their loss landscapes.

1. The WGAN Critic outputs an unbounded scalar score. Crucially, the gradient of the Wasserstein loss is linear and constant. Even when the generated image is visually poor (e.g., desaturated or dark), the Critic provides a strong, meaningful gradient direction towards the real data manifold.
2. The Gradient Penalty enforces a 1-Lipschitz constraint on the Critic. This smoothness prevents the Critic from having infinite slopes around real data points, which in turn prevents the Generator from overfitting to specific pixel noises. This manifests visually as a slow, controlled transition from safe mean colors (sepia) to realistic textures, preserving the semantic boundaries learned during the L_1 phase.

Table 1: Comparative Dynamics of Standard GAN vs. WGAN-GP

Feature	Standard GAN (Baseline)	WGAN-GP (Ours)
Loss Landscape	Sigmoid / BCE (Saturating)	Wasserstein (Linear)
Gradient Signal	Vanishes when D is strong	Constant/Linear
Generator Strategy	Exploits artifacts to "trick" D	Minimizes transport cost
Visual Result	High saturation, Semantic bleeding	Precise semantics, Gradual saturation
Primary Failure	Neon spots	Regression to mean (Sepia)

9 ProGAN-stabilized U-Net with perceptual loss

After Wasserstein GAN, we tried a third approach, the Progressive GAN. It also retains the U-Net backbone but uses stabilization techniques introduced by Karras et al. in *Progressive Growing of GANs* (ProGAN). Furthermore, we added to the pixel-wise error minimization a measure of feature consistency using a VGG loss.

9.1 Generator stabilization: Pixel normalization

Standard Deep Convolutional GANs (DCGANs) typically rely on Batch Normalization (BatchNorm) to prevent covariate shift. However, BatchNorm creates dependencies between samples in a batch, which can lead to signal magnitude escalation and "bubbling" artifacts in generated images.

To address this, we replaced all BatchNorm layers in the Generator with **Pixel Normalization (PixelNorm)**. Unlike BatchNorm, PixelNorm has no learnable parameters and normalizes the feature vector in each pixel location independently to unit length. Formally, for a feature vector x at pixel indices (h, w), the normalized feature vector $b_{h,w}$ is given by:

$$b_{h,w} = \frac{x_{h,w}}{\sqrt{\frac{1}{N}\sum_{j=0}^{N-1}(x_{h,w}^{j})^2 + \epsilon}} \tag{5}$$

where N is the number of feature maps (channels), x^j denotes the j-th feature map, and $\epsilon = 10^{-8}$ is a constant for numerical stability. This constraint prevents the escalation of signal magnitudes during the forward pass, allowing for higher learning rates and more stable convergence without the "coloring outside the lines" artifacts often seen with BatchNorm.

9.2 Discriminator enhancement: minibatch standard deviation

A major failure mode in GAN-based colorization is mode collapse, where the generator learns to output a single, safe "sepia" or brown tone for every input to minimize risk.

To counter this, we introduced a **Minibatch Standard Deviation** layer at the end of the Discriminator. This mechanism explicitly measures the variation within a batch of generated images. The process is defined as follows:

1. Compute the standard deviation of features across the batch dimension.
2. Average these standard deviations over all spatial locations and feature maps to obtain a single scalar value.
3. Replicate this scalar to form a constant feature map matching the spatial resolution of the input.
4. Concatenate this new map to the input features.

If the generator produces a batch of identical or low-variance images (e.g., all brownish), the standard deviation feature map approaches zero. The Discriminator can easily detect this anomaly and penalize the Generator, forcing it to produce diverse color palettes.

9.3 Objective function: VGG perceptual loss

The standard L_1 loss used in our baseline model assumes that colorization is a deterministic process, penalizing any deviation from the exact pixel value of the ground truth. This results in the "averaging effect," producing desaturated, grayish colors.

To resolve this, we implemented a **Perceptual Loss** using a pre-trained VGG-16 network. Instead of comparing pixels, we compare the activation maps (features) extracted from intermediate layers of VGG-16. The Perceptual Loss $\mathcal{L}_{VGG}$ is defined as the Euclidean distance between the feature representations of the generated image $\hat{y}$ and the ground truth y:

$$\mathcal{L}_{VGG}(\hat{y}, y) = \sum_i \frac{1}{C_i H_i W_i} ||\phi_i(\hat{y}) - \phi_i(y)||_2^2 \tag{6}$$

where ϕ_i represents the feature map output of the i-th layer of the VGG network (specifically the ReLU layers of the first few blocks).

The final combined loss function for the Generator is:

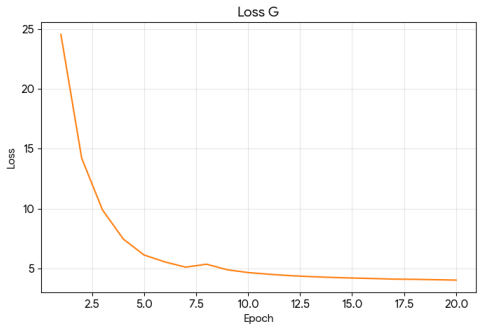
$$\mathcal{L}_G = \mathcal{L}_{BCE}(D(L, \hat{AB}), 1) + \lambda \cdot \mathcal{L}_{VGG}(\hat{RGB}, RGB_{true}) \tag{7}$$

We set the weighting hyperparameter $\lambda = 100$ to prioritize structural and semantic consistency (e.g., "grass should look like grass") over pixel-perfect accuracy.

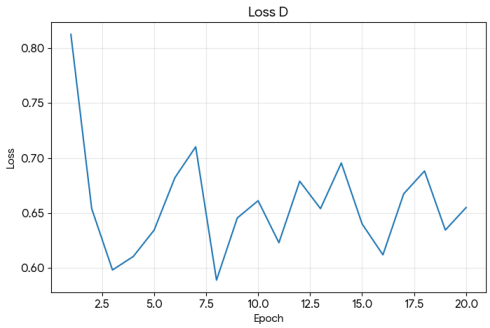
10 Implementation Changes: Transitioning from DCGAN to ProGAN-Hybrid

11 Experimental results and analysis

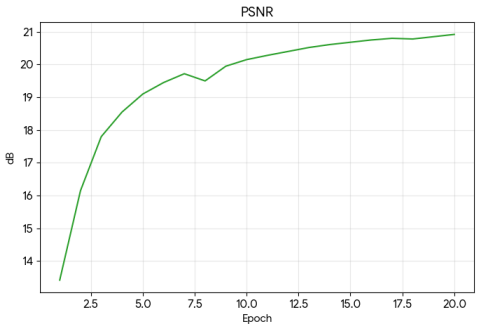
The training dynamics of the ProGAN-stabilized model are illustrated in Figure 11. We observe a sharp and consistent decrease in Generator loss, stabilizing around 10.085 after 20 epochs. Simultaneously, the Discriminator loss maintains a healthy equilibrium range (decreasing to 0.542), indicating that the adversary remains challenging without overpowering the generator. The PSNR metric shows a steady improvement, peaking at 21.12 dB, suggesting that structural fidelity is preserved even while optimizing for perceptual features.



Generator loss



Discriminator loss



PSNR evolution

Unlike baseline L1-based models which often revert to desaturated "sepia" averages, the ProGAN-Hybrid architecture successfully renders vibrant, distinct colors. As seen in the tennis player samples, the model accurately distinguishes between the skin tones, the white clothing, and the court surface.

12 Evaluation framework

Image colorization is a problem where a single grayscale input may correspond to multiple valid colorizations. Consequently, no single metric can capture the full performance of the model. Standard metrics like MSE or PSNR favor determinism and smoothness, while perceptual metrics favor texture and variance.

12.1 Justification for multi-metric evaluation

1. **The PSNR Paradox:** A model can achieve a high PSNR by predicting the statistical average of all possible colors (desaturated brown/gray). While mathematically "accurate" (minimizing Euclidean distance), these images are perceptually failures. Therefore, PSNR is necessary to ensure structural correctness but insufficient to measure realism.

2. **The Texture Gap:** VGG (Perceptual) Loss measures deep feature activation. It penalizes the "blur" that PSNR encourages. A lower VGG score indicates that the model has successfully hallucinated realistic high-frequency textures that match the statistical properties of natural images, even if they do not perfectly align pixel-for-pixel with the ground truth.

3. **The Human Element:** Finally, adversarial artifacts such as "neon blobs" or checkerboard patterns may not drastically impact global average metrics but render an image useless to a human observer. Visual inspection remains the ultimate fail-safe metric to detect mode collapse or architectural failures.

12.2 Models comparison

Table 3 presents the final evaluation metrics comparing the baseline Standard GAN with the proposed WGAN-GP and the optimized ProGAN-Hybrid.

The WGAN-GP outperformed the baseline in both pixel accuracy (PSNR +1.78 dB) and perceptual quality (VGG Loss -2.91%). The lower VGG score quantitatively confirms that the WGAN generates textures that are feature-wise closer to natural images, validating the visual observation of reduced artifacts. With even longer training with lambda=30, we could probably reach even better perceptual results.

Table 2: Comparison of architectural choices between the baseline DCGAN and the ProGAN-stabilized U-Net.

Component	Initial Baseline (DCGAN)	Optimized Approach (ProGAN-Hybrid)
Normalization	**Batch Normalization:** Normalizes using batch statistics. Led to signal spikes and training divergence in early epochs.	**Pixel Normalization:** Normalizes locally per pixel. Preventing signal explosion and stabilizing adversarial training.
Upsampling	**ConvTranspose2d:** Resulted in visible "checkerboard artifacts" in high-frequency areas (checker-like grid patterns).	**Upsample + Conv2d:** Nearest-neighbor interpolation followed by convolution eliminates checkerboard patterns for smoother textures.
Diversity Constraint	**None:** The model often converged to a single mean color (brown) for diverse inputs (Mode Collapse).	**Minibatch StdDev Layer:** Forces the generator to maintain statistical variance in the output batch, encouraging vibrant and distinct colors.
Loss Function	**L1 Distance:** Minimizes pixel-wise error. Mathematically favors the mean of possible colors, resulting in desaturated, gray outputs.	**VGG Perceptual Loss:** Minimizes feature discrepancy. Forces the model to generate colors that are perceptually plausible for the object type, even if they differ from the ground truth.

Table 3: Quantitative and Qualitative Comparison of Colorization Models

Model	Epochs†	PSNR (↑)	VGG (↓)	Visual Characteristics
Standard GAN	10	19.63	4.2968	Visually satisfying albeit sometimes high-frequency artifacts (neon colors). Some color bleeding across semantic boundaries.
WGAN-GP	27	21.41	4.1719	Superior semantic separation (e.g., faces vs. background). Cautious choice of colors, often darker or less vibrant than ground-truth. Realistic texture rendering (lower VGG loss) with no artifacts.
ProGAN-Hybrid	20	20.92	3.3715	Balanced saturation and high structural fidelity. `PixelNorm` eliminates checkerboard artifacts, while `MinibatchStdDev` prevents mode collapse, resulting in diverse textures and sharp object boundaries.

† *Note: The WGAN-GP requires a longer training schedule due to the $n_{critic} = 5$ update rule and the gradient penalty constraint, which slows convergence compared to the unconstrained Standard GAN.*

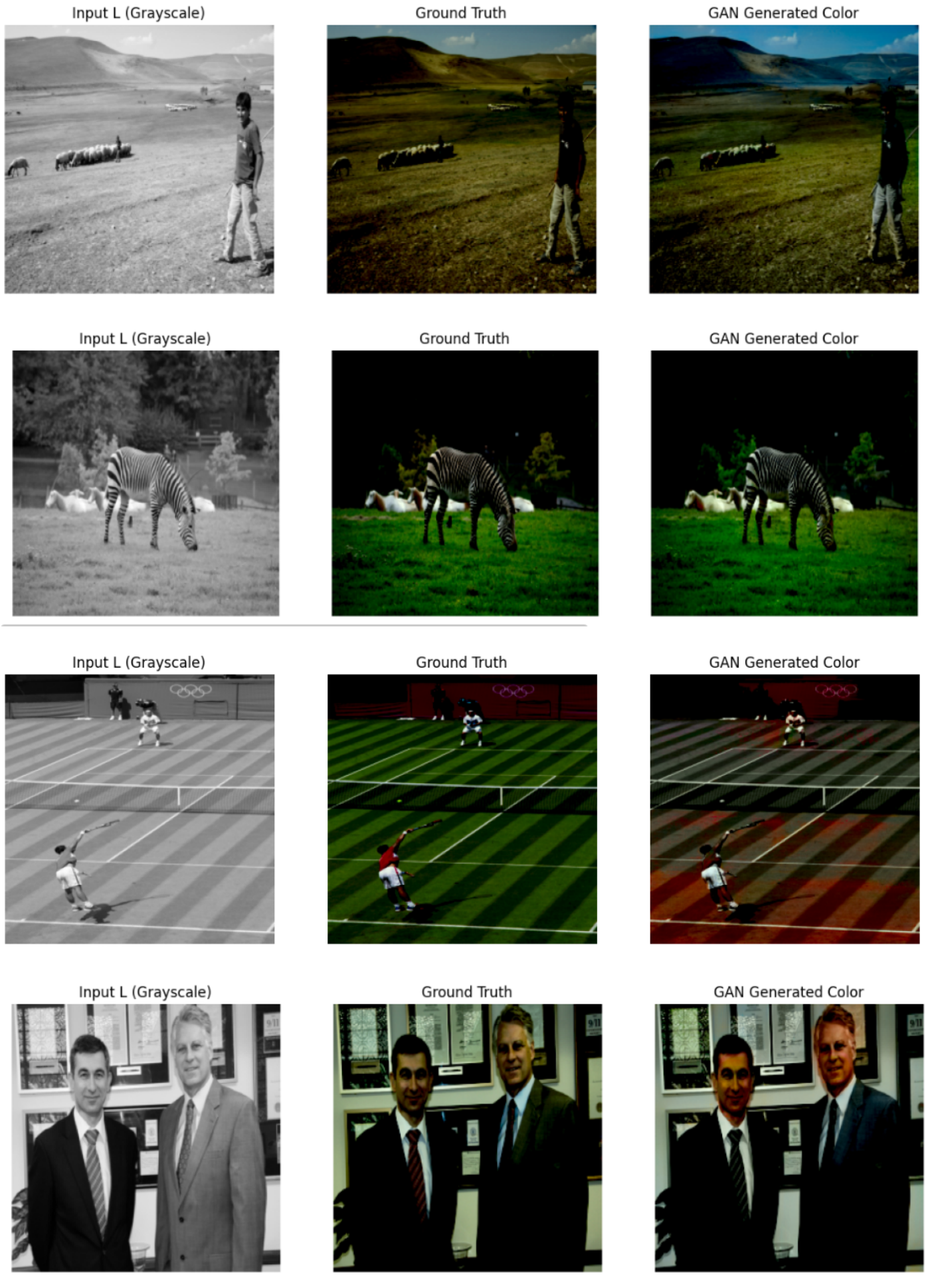

Input L (Grayscale)
Ground Truth
GAN Generated Color
Input L (Grayscale)
Ground Truth
GAN Generated Color
Input L (Grayscale)
Ground Truth
GAN Generated Color
Input L (Grayscale)
Ground Truth
GAN Generated Color

Figure 2: GAN

Figure 3: WGAN

Figure 4: ProGAN

12.3 Architecture of GAN employed

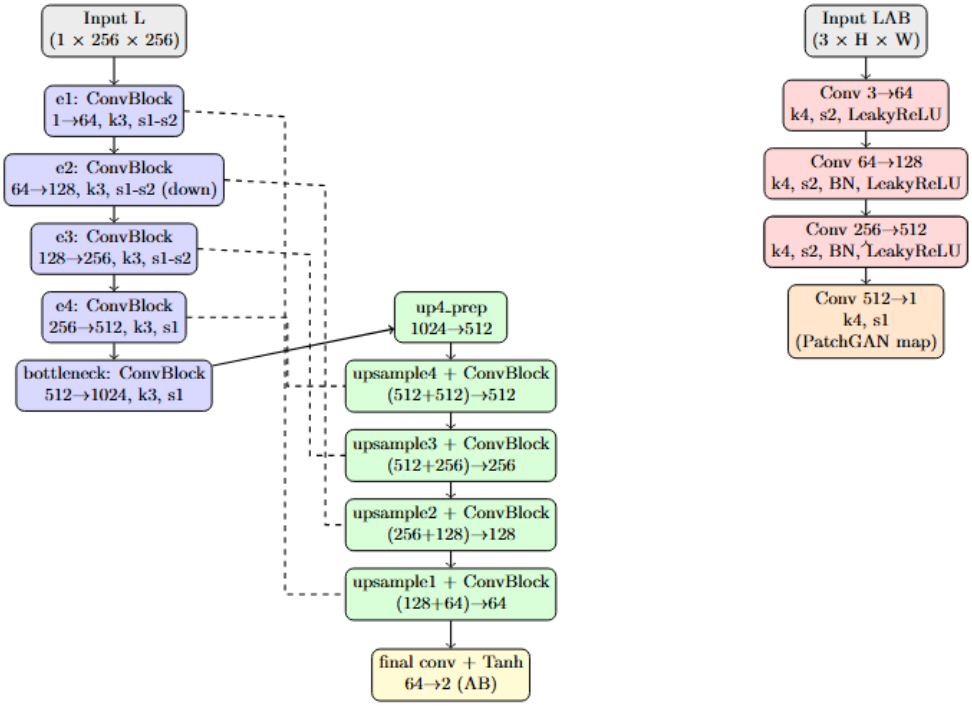


Figure 6: Architecture of Deep FCN U-Net generator and PatchGAN discriminator