

Automatic Grayscale Image Colorization Using a ResNet-18 U-Net in the CIELAB Color Space

Silas Shelly

Student (ID: 202301051)
Department of Computer Engineering
Xavier Institute of Engineering
Mumbai, India

Shannel Dcosta

Student (ID: 202301015)
Department of Computer Engineering
Xavier Institute of Engineering
Mumbai, India

Aryan Gupta

Student (ID: 202301029)
Department of Computer Engineering
Xavier Institute of Engineering
Mumbai, India

Prof. Teena

Guide
Department of Computer Engineering
Xavier Institute of Engineering
Mumbai, India

Abstract—Colorizing grayscale photographs is an ill-posed problem because a single lightness value is consistent with many plausible colors. This paper presents a trainable deep-learning colorizer that predicts the chrominance (a , b) channels of the CIELAB color space from the lightness (L) channel of a grayscale image. The network couples an ImageNet-pretrained ResNet-18 encoder with a U-Net-style decoder connected through skip connections, and is optimized with a Smooth-L1 loss using the Adam optimizer. At inference, the predicted chrominance is upsampled to the original resolution and recombined with the untouched full-resolution L channel, so fine detail is preserved. The system is delivered through a web interface with an interactive divider for comparing the grayscale input against the colorized output. Qualitative results on macro photographs of passion flowers show that the model assigns semantically appropriate colors to foliage and petals, while also revealing typical limitations of regression-based colorization such as local color bleeding and saturation shifts. We discuss these behaviors and outline improvements, including class-rebalanced classification, adversarial training, and quantitative evaluation with PSNR, SSIM, and LPIPS.

Index Terms—image colorization, deep learning, convolutional neural network, CIELAB, ResNet-18, U-Net, transfer learning, computer vision.

I. INTRODUCTION

Grayscale photographs remain abundant in family albums, newspaper archives, historical collections, and early film footage. Adding plausible color to such material improves its visual appeal and makes it more approachable for modern audiences. Automatic colorization is nevertheless difficult: each pixel of a grayscale image carries a single intensity value, whereas a color pixel has three degrees of freedom, so a vast number of color assignments are consistent with the same input. The problem is therefore ill-posed, yet not hopeless, because natural images obey strong semantic regularities: skies tend to be blue, foliage green, and skin within a narrow range of warm tones.

Early approaches resolved the ambiguity with human help. Levin *et al.* [1] propagate color scribbles drawn by a user across regions of similar intensity, while Welsh *et al.* [2] transfer color

statistics from a reference image. Both demand manual effort or a well-matched reference. With the arrival of deep convolutional networks, fully automatic methods became feasible: networks learn the mapping from lightness to color directly from large photo collections [3]–[6], and adversarial formulations further improve vividness [7].

In this paper we present a compact and reproducible colorization system built on transfer learning. A residual encoder pretrained for image classification supplies semantic features, and a U-Net-style decoder with skip connections converts them to per-pixel chrominance in the CIELAB color space. The system is trained end to end with a Smooth-L1 loss, and its output is composed with the original full-resolution lightness channel so that no spatial detail is lost. A web interface lets a user inspect the result with an interactive divider. The main contributions are:

- (i) an end-to-end colorization pipeline that predicts (a , b) chrominance from the L channel using a pretrained ResNet-18 encoder and a U-Net decoder of about 14.3 million parameters;
- (ii) a full-resolution inference procedure that upsamples the predicted chrominance and recombines it with the original lightness channel;
- (iii) an interactive web front end for side-by-side comparison of grayscale input and model prediction; and
- (iv) a qualitative evaluation on real photographs, with an analysis of the strengths and failure modes of regression-based colorization.

The rest of the paper is organized as follows. Section II reviews related work, Section III describes the proposed method, Section IV details the implementation, Section V presents and discusses the results, and Section VI concludes. Section VII outlines future work.

II. RELATED WORK

A. Classical and Interactive Methods

Scribble-based colorization [1] formulates color propagation as a quadratic optimization in which neighboring pixels with

similar intensity are encouraged to share color. It produces high-quality output but needs user input for every image. Example-based methods [2] match luminance and texture statistics between the target and a colored reference, which works well only when a similar reference is available.

B. Deep Learning Methods

Cheng *et al.* [3] were among the first to apply deep networks to automatic colorization, regressing chrominance from hand-designed patch features. Zhang *et al.* [4] treat colorization as classification over quantized (a, b) bins and reweight rare colors during training to avoid the desaturated outputs typical of plain regression. Iizuka *et al.* [5] fuse global image-level features with local features so that scene context guides local color, and Larsson *et al.* [6] predict per-pixel color distributions from hypercolumn features. Isola *et al.* [7] cast colorization as one instance of image-to-image translation using a conditional adversarial network. Table I summarizes these methods and contrasts them with ours.

TABLE I
COMPARISON OF COLORIZATION APPROACHES

Method	Formulation	Key idea
Levin <i>et al.</i> [1]	Optimization	User scribbles propagated by intensity affinity
Welsh <i>et al.</i> [2]	Example-based	Color transferred from a reference image
Cheng <i>et al.</i> [3]	Regression	DNN on patch features, bilateral refinement
Zhang <i>et al.</i> [4]	Classification	Quantized ab bins, class rebalancing
Iizuka <i>et al.</i> [5]	Regression	Fused global and local features
Larsson <i>et al.</i> [6]	Classification	Hypercolumns, per-pixel color histograms
Isola <i>et al.</i> [7]	Conditional GAN	Adversarial plus L1 loss (pix2pix)
Ours	Regression	ResNet-18 U-Net, Smooth-L1, Lab space

C. Architectural Building Blocks

Our design draws on two widely used components. Residual networks [8] ease the training of deep models through identity shortcuts. The U-Net [9] introduced encoder-decoder skip connections that carry high-resolution information to the decoder, a property valuable for dense per-pixel prediction. Initializing the encoder with weights pretrained on large-scale classification data such as ImageNet [10] provides strong semantic features that transfer well to colorization, where knowing *what* an object is largely determines *which* color it should have.

III. PROPOSED METHOD

A. Overview

Fig. 1 shows the overall pipeline. Given an RGB image, the system converts it to CIELAB, keeps only the lightness channel L as input, and lets the network predict the two chrominance channels a and b . The predicted chrominance is then combined with L and converted back to RGB. Working in Lab is convenient because lightness is separated from color, so the network never has to reproduce the intensity information that is already given.

B. Color Space and Normalization

In CIELAB, L ranges over $[0, 100]$ and a, b roughly over $[-128, 127]$. Each training image is resized, converted from RGB to Lab, and normalized as

$$L' = L/50 - 1, \quad (a', b') = (a, b)/128 \quad (1)$$

so that inputs and targets lie approximately in $[-1, 1]$. Training pairs are generated on the fly from ordinary color photographs: L' is the network input and (a', b') is the target, so no manual annotation is required.

C. Network Architecture

The network f_θ (Fig. 1, Table II) maps the lightness image to the two chrominance channels:

$$\hat{Y} = f_\theta(L'), \quad L' \in \mathbb{R}^{1 \times H \times W}, \quad \hat{Y} \in [-1, 1]^{2 \times H \times W} \quad (2)$$

The encoder is a ResNet-18 [8]. Because the pretrained stem expects three channels, the single-channel input is replicated three times, a standard way of reusing ImageNet weights on grayscale data. The stem (7×7 convolution, batch normalization, ReLU) halves the spatial resolution; max-pooling and four residual stages then produce feature maps at $1/4, 1/8, 1/16$, and $1/32$ of the input resolution with 64, 128, 256, and 512 channels, respectively.

The decoder mirrors the encoder with four up-sampling blocks. Each block applies a 2×2 transposed convolution that doubles the spatial size and halves the channel count, concatenates the result with the encoder feature map of matching resolution (the skip connection), and refines the concatenation with two 3×3 convolutions, each followed by batch normalization [11] and ReLU. A final transposed convolution restores full resolution, and a small head (3×3 convolution, ReLU, 1×1 convolution, tanh) outputs the two chrominance channels in $[-1, 1]$. When odd input dimensions cause a size mismatch with a skip connection, bilinear interpolation aligns the maps. The complete model has roughly 14.3 million parameters, about 11.2 million of which belong to the encoder.

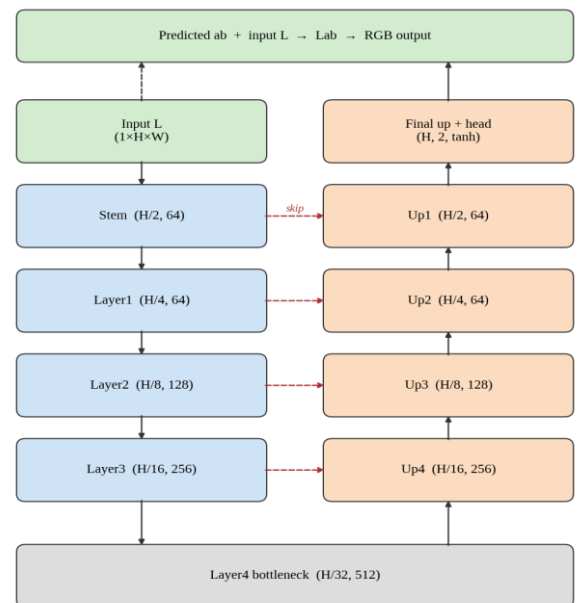


Fig. 1. Architecture of the proposed colorization network. Solid arrows show the data path; dashed arrows are skip connections. Labels give the spatial size (relative to input height H and width W) and channel count.

TABLE II
LAYER CONFIGURATION

Stage	Resolution	Channels
Input (L channel)	$H \times W$	1
Stem (ResNet-18)	$H/2$	64
Layer1 (after max-pool)	$H/4$	64
Layer2	$H/8$	128
Layer3	$H/16$	256
Layer4 (bottleneck)	$H/32$	512
Up4 (+ Layer3 skip)	$H/16$	256
Up3 (+ Layer2 skip)	$H/8$	128
Up2 (+ Layer1 skip)	$H/4$	64
Up1 (+ Stem skip)	$H/2$	64
Final up + head	H	2 (a, b)

D. Loss Function

Plain L2 regression is sensitive to outliers and tends to average over color modes, producing dull, brownish results. We instead use the Smooth-L1 loss [12],

$$\ell(d) = 0.5d^2 \text{ if } |d| < 1; |d| - 0.5 \text{ otherwise} \quad (3)$$

averaged over all pixels and both channels, where Y is the target chrominance:

$$J(\theta) = (1/2HW) \sum_{c,i,j} \ell(\hat{Y}_{c,i,j} - Y_{c,i,j}) \quad (4)$$

The loss is quadratic for small errors, which gives stable gradients near convergence, and linear for large errors, which limits the influence of hard, ambiguous regions.

E. Training Procedure

Parameters are optimized with Adam [13] at an initial learning rate of 10^{-4} . A plateau scheduler halves the learning rate when the validation loss has not improved for three epochs. A random 2% of the images is held out for validation with a fixed seed for reproducibility. After each epoch the model is evaluated on the validation split, the latest state is saved as a checkpoint, and the state with the lowest validation loss is stored separately for inference. A fixed validation batch is also colorized after every epoch and saved as a preview grid of grayscale input, prediction, and ground truth, so convergence can be monitored visually.

F. Inference

At test time the input photograph is converted to Lab. A copy resized to the training resolution (224×224 by default) supplies the L channel for the network, which predicts (a', b') at that size. The prediction is bilinearly upsampled to the original width and height and combined with the L channel of the original full-resolution image, then converted from Lab to RGB with values clipped to the valid gamut. Because L is never modified, the luminance structure of the source photograph is preserved exactly and only chrominance is estimated.

IV. IMPLEMENTATION

A. Software Stack

The system is implemented in Python using PyTorch [15] for model definition and training, torchvision for the pretrained ResNet-18 weights and image transforms, and scikit-image [16] for RGB-Lab conversion. The code is organized into modules for the network, data loading, color-space utilities, training, and

single-image inference, plus a helper that fetches a training corpus. The training script automatically selects a CUDA GPU, an Apple-silicon (MPS) device, or the CPU, in that order of preference.

B. Dataset and Preprocessing

The default training corpus is the unlabeled split of STL-10 [14], which contains 100,000 color images of 96×96 pixels; a subset is exported to disk as image files. Any folder of photographs can be used instead, and no labels are needed. During training each image is resized so that its shorter side is 1.12 times the network input size, a random 224×224 crop is taken, and a random horizontal flip is applied. Corrupt files are skipped by substituting a neighboring image.

C. Training Configuration

Table III lists the hyperparameters used by the training script.

TABLE III
TRAINING HYPERPARAMETERS

Parameter	Value
Optimizer	Adam
Initial learning rate	1×10^{-4}
LR scheduler	ReduceLROnPlateau (factor 0.5, patience 3)
Loss	Smooth-L1
Batch size	32
Epochs	30
Input resolution	224×224
Augmentation	Random crop, horizontal flip
Validation split	2% (seed 42)
Encoder initialization	ImageNet-pretrained ResNet-18
Trainable parameters	≈ 14.3 M

D. Web Interface

To make the model easy to use, the trained network is exposed through a web interface (Figs. 2 and 3). The user supplies a grayscale photograph, and the interface shows the grayscale input on the left and the model prediction on the right, separated by a draggable vertical divider so that the two can be compared region by region. A download button saves the colorized result.

V. RESULTS AND DISCUSSION

A. Experimental Setup

The trained model was applied to two grayscale macro photographs of passion flowers (*Passiflora*). This subject is demanding: it combines a fine radial corona of filaments, strongly contrasting petal and foliage colors, and a shallow depth of field with a blurred background.

B. Qualitative Results

Fig. 2 shows the first flower. The model paints the petals and corona filaments in violet and pink-lavender tones and the surrounding foliage in green, a plausible rendering for a flower of this kind. The dark central structures remain dark with a faint purple cast, and the out-of-focus background is rendered in yellow-green. Fig. 3 shows a second flower with pale petals. Here the model keeps the petals nearly white, adds a yellow-green tone to the central column of stamens, places a purple tint

at the base of the corona filaments, and renders leaves and blurred background in saturated shades of green.

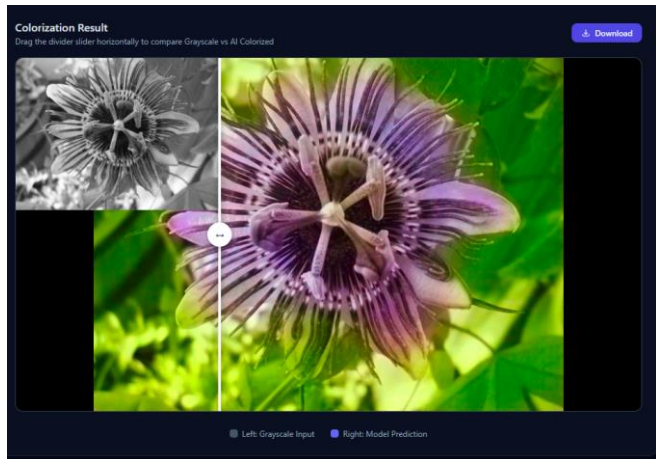


Fig. 2. Colorization result for the first passion flower. Left: grayscale input; right: model prediction, compared with the interactive divider.

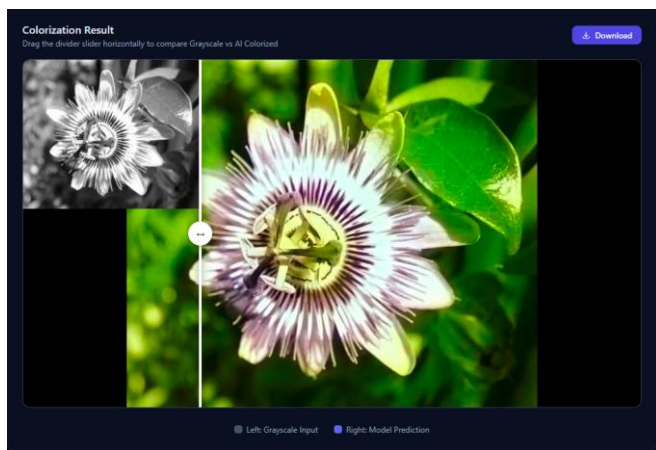


Fig. 3. Colorization result for the second passion flower, which has pale petals and a green-and-yellow central column.

C. Discussion

Three observations stand out. First, the model shows clear semantic awareness: vegetation is consistently green, and the two flowers receive different, plausible petal colors, which indicates that features from the pretrained encoder carry over to colorization. Second, colors are spatially coherent across large regions such as leaves and background. Third, because the original L channel is retained, edges and fine filaments remain crisp in the output even though the network operates at a lower internal resolution.

The results also expose limitations. (i) *Color bleeding*: in Fig. 2 a soft violet halo extends beyond the petal boundaries. This is expected because chrominance is predicted at a fixed low resolution and bilinearly upsampled, so color edges are smoother than luminance edges. (ii) *Saturation and hue shift*: greens, particularly in the background and leaf highlights, appear strongly saturated with a yellowish cast in both figures. Pixel-wise regression can develop such biases when the training distribution favors particular scenes, and STL-10 images are small (96×96) and were upsampled, which may limit exposure to fine texture. (iii) *Ambiguity*: the petal color of a flower cannot be determined from luminance alone; a regression model returns a single conditional estimate rather than several

alternatives, so the hue in Fig. 2 reflects a learned prior rather than certainty. (iv) *Aspect ratio*: the fixed square input resolution distorts non-square photographs before inference. Finally, color ground truth was not available for these test images, so the assessment is qualitative.

D. Toward Quantitative Evaluation

For a rigorous comparison, future experiments should colorize held-out color photographs and compare the outputs with the originals using PSNR and SSIM [17] for pixel and structural fidelity, the learned perceptual metric LPIPS [18] for perceptual similarity, and a user study of perceived realism in the style of the colorization Turing test [4]. Such an evaluation would also enable ablations of the pretrained encoder, the skip connections, and the loss function.

VI. CONCLUSION

This paper presented an automatic image colorizer that predicts CIELAB chrominance from lightness using a ResNet-18 encoder and a U-Net decoder, trained with a Smooth-L1 loss and delivered through an interactive web interface. Qualitative results on passion-flower photographs demonstrate semantically sensible coloring of petals and foliage while preserving the original luminance detail. The observed color bleeding, saturation shifts, and single-hypothesis prediction are consistent with known limitations of regression-based methods and motivate the improvements below.

VII. FUTURE SCOPE

Several extensions are planned. Replacing regression with classification over quantized (a , b) bins and a class-rebalanced loss [4] should produce more vivid and diverse colors. Adding an adversarial loss [7] can further sharpen and enrich results. Training on larger, higher-resolution, and more varied datasets such as ImageNet [10] should improve generalization, while predicting chrominance at higher resolution or guiding the upsampling with the luminance channel should reduce color bleeding. Aspect-preserving or tiled inference, optional user color hints for interactive control, and optimized deployment on cloud and mobile platforms would make the system more practical.

ACKNOWLEDGMENT

The authors thank Prof. Teena, Xavier Institute of Engineering, for their guidance and continued support.

REFERENCES

- [1] A. Levin, D. Lischinski, and Y. Weiss, "Colorization using optimization," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 689–694, 2004.
- [2] T. Welsh, M. Ashikhmin, and K. Mueller, "Transferring color to greyscale images," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 277–280, 2002.
- [3] Z. Cheng, Q. Yang, and B. Sheng, "Deep colorization," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2015, pp. 415–423.
- [4] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2016, pp. 649–666.
- [5] S. Iizuka, E. Simo-Serra, and H. Ishikawa, "Let there be color!: Joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification," *ACM Trans. Graph.*, vol. 35, no. 4, Art. 110, 2016.
- [6] G. Larsson, M. Maire, and G. Shakhnarovich, "Learning representations for automatic colorization," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2016, pp. 577–593.

- [7] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017, pp. 1125–1134.
- [8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 770–778.
- [9] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Med. Image Comput. Comput.-Assist. Interv. (MICCAI)*, 2015, pp. 234–241.
- [10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2009, pp. 248–255.
- [11] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2015, pp. 448–456.
- [12] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2015, pp. 1440–1448.
- [13] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2015.
- [14] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proc. Int. Conf. Artif. Intell. Statist. (AISTATS)*, 2011, pp. 215–223.
- [15] A. Paszke *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 32, 2019.
- [16] S. van der Walt *et al.*, "scikit-image: Image processing in Python," *PeerJ*, vol. 2, p. e453, 2014.
- [17] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.
- [18] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 586–595.