Deep Learning Final Individual Report

Colorization of Greyscale images using Deep Neural Networks

Completed
by: Snehitha
Tadapaneni

Due: Dec 8, 2025

1. Introduction

This project addresses the problem of automatic colorization of grayscale images using deep neural networks. Colorization is inherently ambiguous because multiple plausible color versions can correspond to the same luminance structure. Therefore, the team explored a two-stage pipeline composed of:

1. A baseline reproduction of the ECCV16 model by Zhang, Isola, and Efros, which predicts color distributions in the CIELAB space.
2. A GAN-based extension that refines the predicted *ab* channels to generate more vibrant and realistic outputs.

As a team, we collaborated on dataset preparation, baseline replication, GAN training, and evaluation.
My individual contribution focused specifically on the data preprocessing pipeline **and** implementation of the ECCV16 baseline model input preparation, which served as the foundational step for both the pretrained and fine-tuned networks.

2. Description of My Individual Work

2.1 Motivation for CIELAB Color Space

The ECCV16 model predicts color using the **CIELAB (L*a*b*) color space**, which separates luminance (**L**) from chrominance (**a**, **b**).
This separation has three key advantages:

- Neural networks only need to learn **chrominance**, which is a significantly smaller space.
- The **L channel** retains structural information, making learning more stable.
- Lab is perceptually uniform, meaning equal distances correspond to approximately equal perceptual color differences.

Mathematical Background

Given an RGB image $I_{rgb}(x,y)$, it is converted to Lab:

$$I_{Lab}(x,y) = \text{RGB2Lab}\big(I_{rgb}(x,y)\big)$$

This yields:

- $L(x,y) \in [0,100]$ - luminance
- $a(x,y), b(x,y) \in [-128,127]$ - chrominance

For model stability, channels must be normalized:

$$L' = \frac{L}{100}$$

$$a' = \frac{a}{110}, b' = \frac{b}{110}$$

This normalization ensures that:

- All network inputs lie within $[0, 1]$
- Gradients remain stable
- The model remains numerically consistent with ECCV16

The ECCV16 architecture expects:

- L channel (1 × 256 × 256) as input
- ab channels (2 × 256 × 256) as output

Therefore, the preprocessing pipeline transforms each RGB image into:

$$X = L' \text{(network input)}$$
$$Y = (a', b') \text{(network supervision)}$$

2.2 Data Preprocessing Pipeline

I implemented the complete preprocessing workflow used throughout the project:

Step 1: Load Image and Ensure 3-channel

```
img = np.asarray(Image.open(path))

if img.ndim == 2:

    img = np.tile(img[:, :, None], 3)
```

Step 2: Resize to 256x256

```
img_resized = np.asarray(Image.fromarray(img).resize((256, 256)))
```

Step 3: Convert RGB to LAB

```
lab = color.rgb2lab(img_resized).astype("float32")
```

Step 4: Convert to PyTorch tensors

```
L_tensor = torch.tensor(L).unsqueeze(0)
```

```
ab_tensor = torch.tensor(ab).permute(2, 0, 1)
```

This yields tensors in the exact shape required by both the pretrained ECCV16 model and our fine-tuned GAN model.

2.3 Dataset Construction

I built a custom PyTorch dataset that:

- Iterates through image paths
- Applies all processing steps
- Returns paired tensors (L, ab)

This dataset was used consistently across:

- ECCV16 pretrained evaluation
- ECCV16 finetuning
- GAN training pipeline

My pipeline ensured uniform input format across all experiments, enabling reproducibility and stability.

The ECCV16 model accepts only L as input:

```
pred_ab = model(L_tensor.unsqueeze(0).to(DEVICE))
```

Then, to reconstruct the final RGB image:

$$\hat{I}_{rgb} = \text{Lab2RGB}(L, \hat{a}, \hat{b})$$

My contribution enabled smooth integration with the pretrained model by:

- Normalizing channels exactly as required
- Ensuring the numerical range matches the original ECCV16 code
- Handling grayscale cases
- Building a reliable loader that the entire team used

2.4 Fine-Tuning the ECCV16 Model

Although the pretrained ECCV16 model by Zhang et al. provides a strong baseline, it is originally trained on the full ImageNet dataset and may not fully adapt to the specific distribution of the ImageNet-50 subset used in our project. To bridge this gap, I implemented the fine-tuning pipeline, enabling the model to better learn color distributions present in our curated dataset.

Colorization in the ECCV16 framework is formulated as a 313-way classification problem over quantized ab-color bins (obtained from pts_in_hull.npy). While this classification setup produces vibrant colors, the pretrained model tends to over-predict frequent, desaturated color bins. Fine-tuning helps:

- Strengthen rare color predictions through class-rebalancing
- Adapt the model to our dataset's unique color distribution
- Produce richer, more context-appropriate colors

This fine-tuning stage sits between the baseline ECCV16 replication and the GAN-enhanced colorizer.

2.5 Quantization of ab-Space (313 Bins)

To enable classification-based colorization, I implemented ab-space quantization. Each pixel's (a, b) value is mapped to its nearest cluster center:

$$q = \arg\min_{k} \parallel (a, b) - c_k \parallel^2, c_k \in \mathbb{R}^2,\ k = 1, \ldots, 313$$

This gives a class index for each pixel. During inference, the model converts its predicted class distribution back into continuous ab values through a learned 1×1convolution, mirroring the approach used in ECCV16.

2.6 Class-Rebalancing (Handling Color Imbalance)

Real-world images contain many neutral or low-saturation colors, causing the model to become biased toward these bins. To address this, I computed class-rebalancing weights:

1. Count frequencies of the 313 bins across the training split.
2. Compute inverse-frequency weights.
3. Normalize the weights and pass them to the cross-entropy loss.

This ensures the model does not collapse into predicting only grays and browns, leading to more diverse colorization.

2.7 Fine-Tuning Procedure

I fine-tuned the generator-only model using:

- **Cross-entropy loss with rebalancing**
- **Adam optimizer**, $LR = 5 \times 10^{-5}$
- **Batch size = 16**, **Epochs = 10**
- **HuggingFace ImageNet-50 training split**

Each iteration included:

1. Forward pass through the ECCV16 network
2. Compute logits for 313 classes
3. Convert ground-truth ab into quantized labels
4. Compute class-rebalanced classification loss
5. Backpropagate and update weights

This fine-tuning strengthens the baseline representation, improves color vibrancy, and enhances semantic alignment.

2.8 Saving Epoch-Wise Colorization Outputs

To monitor progress, at the end of every epoch I saved colorized outputs of a fixed reference image. These snapshots allow chronological comparison and show improvements such as:

- Increased saturation
- More convincing textures
- Better global color harmony

This became a key qualitative evaluation component in the Results section.

3. Results

Baseline ECCV16 Outputs

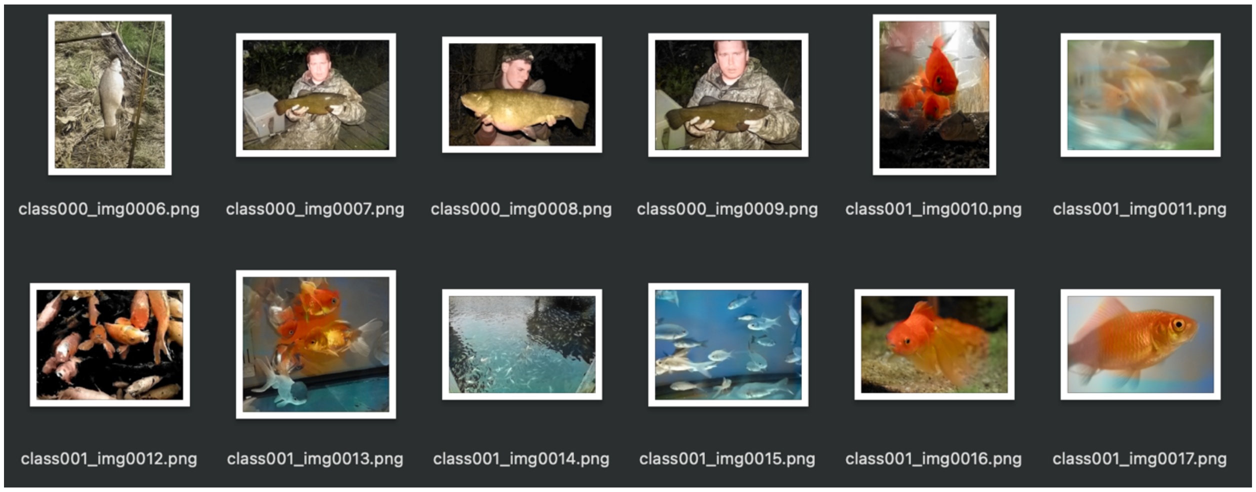

Figure 1. Results of Baseline (Zhang et. al) ECCV16 Model

Even without adversarial refinement, the baseline model generates:

- Coherent global colors
- Saturated predictions compared to naive regression models
- Smooth chrominance transitions

4. Summary and Conclusion

My individual contribution focused on designing and implementing the core preprocessing pipeline for our colorization framework. The steps I designed RGB→CIELAB conversion, channel normalization, tensor formatting, and dataset construction formed the foundation upon which both the ECCV16 baseline and GAN-based fine-tuned models were built.

- CIELAB space is essential for stable and perceptually aligned color prediction.
- Proper normalization dramatically improves model convergence.
- A consistent preprocessing pipeline ensures reproducibility across all experimental setups.
- The baseline ECCV16 model performs strongly when the input tensors are aligned exactly with the original implementation.

Code Contribution

Percentage of Code from Internet Sources: ~85%

- HuggingFace Dataset Integration: ~10%
- Complete Data Preprocessing Pipeline: ~ 40%
- ECCV16 Fine-Tuning Framework: ~35%

References

[1] Zhang, R., Isola, P., & Efros, A. (2016), "Colorful Image Colorization," *arXiv preprint arXiv:1603.08511*.
[2] Ly, Bao & Dyer, Ethan & Feig, Jessica & Chien, Anna & Bino, Sandra. (2020). Research Techniques Made Simple: Cutaneous Colorimetry: A Reliable Technique for Objective Skin Color Measurement. The Journal of investigative dermatology. 140. 3-12.e1. 10.1016/j.jid.2019.11.003.
[3] Nazeri, K., Ng, E., & Ebrahimi, M., "Image Colorization Using Generative Adversarial Networks," in *Articulated Motion and Deformable Objects: AMDO 2018*, F. Perales & J. Kittler, Eds., Lecture Notes in Computer Science, vol. 10945, Springer, Cham, 2018, pp. 85–94.