Polygon Colorization Using Conditional UNet – Final Training Report

---

1. Hyperparameters – Tuning and Final Settings

Objective:

To train a deep learning model (UNet) that can color polygon-based shapes based on a provided color condition. Each sample in the dataset includes:

- A **polygon input** (grayscale or RGB image)
- A **target color** (RGB triplet)
- A **ground truth image** showing the polygon filled with the target color

Final Hyperparameter Configuration:

Hyperparameter	**Values Tried**	**Final Choice**	**Reasoning**
Learning Rate	1e-3, **1e-4**, 5e-5	**1e-4**	1e-4 provided a stable and smooth loss curve without vanishing gradients. 1e-3 was unstable.
Batch Size	4, **8**, 16	**8**	Provided a good balance between memory usage and gradient diversity.
Epochs	50, **100**, 150	**100**	50 wasn't enough to fully learn, and 100 gave near-optimal convergence.
Optimizer	SGD, **Adam**	**Adam**	Adaptive learning rate allowed faster and more stable convergence.
Loss Function	MSELoss, **L1Loss**	**L1Loss**	L1Loss preserved edges better and minimized blurring compared to MSELoss.
Input Resolution	128×128, 256×256	**128×128**	128×128 used for faster training. Larger resolution planned for future enhancement.
Color Conditioning	Embedding, **Concatenation**	**Concatenation**	Conditioning was done by duplicating the RGB color over the image and concatenating it with the input polygon.

---

2. Architecture – UNet with Color Conditioning

Architecture Overview:

We implemented a **UNet-based** encoder-decoder architecture with skip connections. The unique design challenge was to condition the output image generation based on a given **target color**.

Design Details:

- **Input:** Concatenation of a polygon image (3 channels) and the target color (3 channels), forming a 6-channel input tensor.
- **Encoder:** 4 blocks of 2×Conv + ReLU + BatchNorm + MaxPool.
- **Bottleneck:** 2 Conv layers, with BatchNorm and ReLU.
- **Decoder:** 4 blocks of Upsample + Conv layers, using skip connections from encoder.
- **Output Layer:** 1×1 convolution to reduce channels to 3, followed by a sigmoid activation to clamp output between [0, 1].

Color Conditioning:

Instead of using an embedding, we passed the RGB color vector as a solid-color image (same height/width as the polygon), concatenated it with the polygon input to form a 6-channel input.

Additional Features:

- **Batch Normalization:** Used after each conv block for faster training.
- **Skip Connections:** Preserve spatial detail between encoder and decoder.
- **Output Resolution:** Same as input (128×128), enabling pixel-wise loss calculation.

---

3. Training Dynamics – Behavior, Metrics & Examples

Loss Curve:

- **Training Loss:** Started around 0.3 (L1) and gradually decreased to below 0.09 by epoch 100.
- **Validation Loss:** Closely followed training loss without diverging, indicating low overfitting.

We tracked loss and model predictions using **Weights & Biases (wandb)** for easy monitoring and debugging.

Qualitative Analysis:

Epoch	Visual Trend
0–10	Outputs were blank or noisy. Model hadn't learned correlation between color and shape.
10–40	Learned correct color application, but blurry edges.
40–70	Sharper results, color aligned with polygon boundaries.
70–100	Good quality, few artifacts, stable convergence.

Failure Modes & Fixes:

Issue	Cause	Fix Attempted	Result
Output blurry	MSE Loss smooths pixel-wise errors	Switched to L1Loss	Sharper outputs
Color mismatch	Improper conditioning	Used RGB solid-color channel conditioning	More accurate colors
Loss not decreasing	Wrong input scale	Normalized inputs and outputs to [0, 1]	Model started converging
Slight boundary artifacts	Limited resolution and early downsampling	Planned higher resolution (256×256)	Future enhancement

4. Key Learnings & Future Directions

Learnings:

- **Effective Conditioning:** Concatenating RGB color channels as input is a simple and powerful technique for conditioning a generative model.
- **L1 Loss Outperforms MSE** in structured tasks like image synthesis, due to better edge preservation.
- **UNet Is Strong for Dense Predictions**: Skip connections helped retain boundary information and spatial resolution.
- **Visualization Tools (wandb)** were extremely helpful for quickly identifying training issues and understanding model behavior.
- **Clean Data is Essential**: Even small misalignments in input-output pairs can cause blurry outputs or color spills.

Future Improvements:

- Increase resolution to **256×256** to improve visual sharpness and detail.
- Use **perceptual loss** (VGG-based) or **adversarial training** (e.g., with a PatchGAN discriminator) for more realistic outputs.
- Add **attention modules** (e.g., SE blocks or self-attention in bottleneck) to focus on important regions.
- Train longer and on more diverse datasets to improve generalization.