Insights Report:
UNet-based Conditional Polygon Coloring

Aayush Singh

August 5, 2025

Introduction

This report presents the insights gleaned from training a UNet model from scratch to generate colored polygon images, conditioning on both the polygon shape and a specified color. The problem tackled involves translating an input polygon image plus a color name into an output polygon filled with that target color. The dataset consists of polygon shapes and their colored counterparts, with conditioning implemented via a one-hot encoded color vector.

The implementation is in PyTorch, training leveraged wandb for experiment tracking, and inference is demonstrated through visualization of predicted vs. ground truth colored polygons. The following sections discuss the rationale behind the hyperparameter choices, the architectural design and conditioning mechanism, the training and validation dynamics, failure patterns noticed, and key learnings.

Hyperparameters and Training Configuration

For optimal model performance, several hyperparameters and design choices were experimentally explored, as summarized in Table 1. These choices were informed by established practices in image-to-image architectures and empirical trial, balancing convergence, stability, and model capacity.

Parameter	**Tried Values**	**Chosen Value**	**Rationale**
Learning Rate	1×10^{-2}, 1×10^{-3}, 1×10^{-4}	1×10^{-3}	Stable, non-divergent convergence with Adam.
Batch Size	16, 32, 64	32	Memory fits T4 GPU and yields stable gradients.
Optimizer	SGD, Adam	Adam	Adaptive for computer vision; handles variance well.
Epochs	100, 200, 300	200	Loss plateaus, overfitting avoided.
Loss Function	MSELoss, MAE	MSE	Pixel-wise color prediction matches image targets.
Image Size	64^2, 128^2	128^2	Captures edge details crisply but is still efficient.
UNet Depth	1–4 blocks	2 down/2 up	Prevents overfitting, sufficient for this task.
Color Encoding	One-hot, Embedding, Integer	One-hot	Unambiguous for classifying 8 colors.

Table 1: Summary of hyperparameters, settings, and rationale.

The final model used a constant learning rate of 1×10^{-3} with Adam, batch size of 32, 200 epochs, 128×128 pixel inputs, and MSE loss.

Model Architecture and Conditioning

Color conditioning is done by concatenating the polygon image (3 channels) with the broadcast one-hot color tensor along the channel axis, creating a $(3 + n_{colors}) \times H \times W$ input. Example code:

```
color_map = color_onehot.view(b, n_colors, 1, 1).expand(-1, -1, h, w)
x = torch.cat([img, color_map], dim=1)
```

The UNet is structured as follows:

- **Encoder blocks:** Two UNet blocks each with two (Conv2d $\rightarrow$ BatchNorm2d $\rightarrow$ ReLU); downsampling via MaxPool2d.
- **Middle block:** One dense block with increased channels.
- **Decoder blocks:** Two upsample blocks (ConvTranspose2d), skip-connections, and UNet blocks.
- **Output:** Final 1×1 convolution to 3 RGB channels, with sigmoid activation.

Batch normalization and ReLU activation ensure stable, non-saturating training.

Training Dynamics and Performance

Loss curves (tracked via wandb) show that both training and validation loss decrease steadily over 200 epochs, with no overfitting—the loss curves largely overlap as training progresses.

Qualitatively, the model learned to fill the polygons with the target color, matching the ground truth with high fidelity by epoch 200. Early outputs were blurry or faint, but improved rapidly after epoch 50.

Challenges and Failure Modes

Key observed limitations:

- **Blurry/bleeding edges:** Infrequent, but occurs on very sharp polygons or unseen shapes.
- **Color faintness/artefacts:** Model sometimes produces less saturated fills.
- **Overfitting risks:** Not observed, but deeper models or less augmentation could overfit.

Recommendations for Improvement

- **Further Data Augmentation:** Random rotation, scaling, noise to inputs/labels.
- **Regularization:** Dropout, weight decay, or early stopping for larger/deeper UNets.
- **Loss Functions:** Perceptual loss, dice/boundary loss to target sharp edges.
- **Color Conditioning:** Try learned embeddings or FiLM-modulation for more colors.

Key Learnings

- Early fusion one-hot color conditioning is simple and effective for discrete color targets.
- Modest-depth UNet suffices for this dataset; further depth only needed for more complex images.
- Visual validation and experiment tracking (e.g., with wandb) is essential for these tasks.
- Most errors can be traced to input ambiguity or minor mismatches in color interpretation.

Conclusion

This UNet-based system solves the conditional colored polygon generation task robustly with principled hyperparameter search, careful channel-wise color conditioning, and good monitoring. There is ample room to explore advanced conditioning and augmentation in future work.

Project page for reference and tracking:
https://wandb.ai/aayushsingh8217-indian-institute-of-information-technology/polygon-unet