Project Report: Polygon Colorization

Saketh Nellutla

August 4, 2025

1 Model Architecture

1.1 Full FiLM-UNet Overview

- **Input:** RGB polygon image $3 \times 128 \times 128$ and an 8-dimensional one-hot color vector
- **Color Embedding:** One-hot $\rightarrow$ Linear(8,256) $\rightarrow$ ReLU $\rightarrow$ Linear(256,64) $\rightarrow$ ReLU
- **Encoder (4 blocks):**
  1. $3 \rightarrow 64$ feature maps, FiLM-modulated
  2. $64 \rightarrow 128$, FiLM-modulated
  3. $128 \rightarrow 256$, FiLM-modulated
  4. $256 \rightarrow 512$, FiLM-modulated

  Each block: DoubleConv (Conv–BN–FiLM–ReLU)$\times 2$ + MaxPool
- **Bottleneck:** $512 \rightarrow 1024$ DoubleConv + FiLM
- **Decoder (4 blocks):**
  1. TransposeConv $1024 \rightarrow 512$; concat with Encoder-4; DoubleConv + FiLM $1024 \rightarrow 512$
  2. TransposeConv $512 \rightarrow 256$; concat with Encoder-3; DoubleConv + FiLM $512 \rightarrow 256$
  3. TransposeConv $256 \rightarrow 128$; concat with Encoder-2; DoubleConv + FiLM $256 \rightarrow 128$
  4. TransposeConv $128 \rightarrow 64$; concat with Encoder-1; DoubleConv + FiLM $128 \rightarrow 64$
- **Output Layer:** 1×1 Conv $64 \rightarrow 3$ (RGB)
- **Skip Connections:** Standard UNet links from each encoder block to corresponding decoder block
- **FiLM Applications:** 9 total (4 encoder + 1 bottleneck + 4 decoder)

1.2 FiLM Mechanism

Each FiLM layer generates scale (γ) and shift (β) parameters from the 64-dimensional color embedding:

$$\gamma, \beta = \text{MLP}(\text{color\_embedding}) \tag{1}$$

$$\text{output} = \gamma \times \text{features} + \beta \tag{2}$$

2 Hyperparameters

Parameter	**Value**	**Rationale**
Epochs	75-300 (early stop)	Model converges earlier; training halts on no validation loss improvement
Batch Size	16 (train), 8 (val)	Fits GPU memory while providing stable gradients
Optimizer	Adam	Well-established performance for vision tasks
Learning Rate	1×10^{-4}	Optimal after testing range 10^{-3} to 10^{-4}
Scheduler	ReduceLROnPlateau	Reduces LR by 0.5 after 10 epochs without improvement
Weight Decay	Light ($\approx 10^{-5}$)	Prevents overfitting on small dataset
Image Size	128×128	Balances detail retention with computational efficiency

Table 1: Final training hyperparameters

3 Training Configuration and Dynamics

3.1 Training Setup

- **Dataset:** PolygonColorDataset with paired RGB images and 8-class color labels
- **Augmentation:** Synchronized resize ($144 \rightarrow 128$), rotation ($\pm 30°$), horizontal/vertical flips
- **Loss Function:** $\mathcal{L} = \alpha \cdot \text{MSE} + \beta \cdot \text{Consistency}$ with $\alpha = 1, \beta = 2$
- **Metrics:** Pixel color accuracy, region color accuracy, validation loss (logged via Weights & Biases)
- **Checkpointing:** Best model saved based on lowest validation loss
- **Gradient Clipping:** Max norm 1.0 for training stability

3.2 Observed Learning Progression

- Rapid boundary learning in first 10 epochs
- Color fill improvement after consistency loss takes effect (around epoch 25)
- increasing learning rate decreased the accuracy
- tried to make the model learn using SSIM scores
- Early stopping typically occurs around epoch 40–50

4 Key Insights and Learnings

1. **FiLM Comparison:** Both full FiLM and decoder-only FiLM variants work effectively, but full FiLM provides marginally better consistency on irregular polygon shapes.
2. **Efficient Loss Function:** Adding a color-consistency term directly addressed the primary failure mode of correct boundaries but poor color fill. This domain-specific loss component was essential for quality results.
3. **Augmentation Alignment:** Synchronized transformations are critical–any misalignment between input and target images destroys the supervised learning signal.

4. **Learning Rate Scheduling:** Fixed learning rates tend to overshoot optimal solutions. Adaptive decay using ReduceLROnPlateau stabilized convergence significantly.

5. **Early Stopping Efficiency:** The model consistently peaks 30–40% before maximum epochs, demonstrating the value of early stopping for computational efficiency.

6. **Dataset Size Impact:** Small dataset size necessitated extensive data augmentation and careful regularization. This highlighted the importance of data quality and quantity in deep learning projects.

7. **One-hot vs Numerical Encoding:** One-hot color encoding provided much clearer conditional signals compared to numerical color representations, leading to better color consistency.

8. **Dataset Importanece:** I've learned again how dataset is the most important part here, and how augmentations can be helpful in real case applications where there is data scarcity.