Polygon Colorization using Conditional UNet: Report and Insights

Ayna Internship Assignment

1 Hyperparameters

What was tried

- Learning rates: `1e-3`, `5e-4`, `1e-4` (best).
- Optimizers: Adam, AdamW (Adam yielded smoother convergence).
- Loss weights: L1-only, MSE-only, combo (`L1: 1.0, MSE: 0.5` worked best).
- Dropout values: `0.0`, `0.1`, `0.25`.
- Schedulers: ReduceLROnPlateau (chosen), CosineAnnealing.

Final Settings

```
Image Size:       256x256
Embedding Dim:    64
Batch Size:       16
Learning Rate:    1e-4
Loss:             L1 + 0.5 * MSE
Dropout:          0.1
Optimizer:        Adam
Scheduler:        ReduceLROnPlateau
Early Stopping:   15 epochs
```

Rationale: Final settings were chosen based on validation loss trends and visual outputs, considering sharpness, saturation, and boundary quality.

2 Architecture and Conditioning

Design Overview

- **Backbone:** A standard 4-level UNet with symmetric encoder-decoder pathways and skip connections. Each block is composed of two convolutional layers with batch normalization and ReLU activations (via a `DoubleConv` module).
- **Condition Injection:** The color label is passed through an `nn.Embedding` layer to obtain a 64-dimensional vector. This vector is broadcast spatially (to 64xHxW) and concatenated with the 1-channel grayscale input along the channel dimension, forming a 65-channel tensor.
- **Encoder Path:** The concatenated tensor is passed through:
  - Initial DoubleConv block ($1 + 64 \rightarrow 64$ channels).
  - Downsampling layers with max pooling followed by DoubleConv blocks, with increasing channels: 64→128→256→512→1024.
- **Decoder Path:** Uses bilinear upsampling and skip connections from encoder. Each Up block merges feature maps from encoder and decoder before applying DoubleConv.
- **Output Layer:** A 1x1 convolution projects the final feature map (64 channels) to 3 output channels (RGB), followed by a `sigmoid` activation to constrain output values to $[0, 1]$.
- **Dropout:** Applied in DoubleConv blocks to prevent overfitting, set at 0.1.

Ablations Explored

- **No conditioning:** Model always predicted the same color regardless of input, confirming the importance of label injection.
- **FiLM vs. Concatenation:** FiLM-based conditioning (Feature-wise Linear Modulation) was tested but underperformed compared to simple channel-wise concatenation of the embedding.
- **Upsampling methods:** Transposed convolution vs. bilinear interpolation showed no significant difference; bilinear was chosen for simplicity and parameter efficiency.

Ablations Explored

- **No conditioning**: Output lacked diversity; same input always produced one color.
- **Concat vs. FiLM**: Concatenation of embedding as spatial channel gave better results.
- **Bilinear vs. TransposeConv**: Both were similar; bilinear was used for simplicity.

3 Training Dynamics

Loss Curves

Both training and validation losses steadily decreased over epochs. Mild overfitting appeared after epoch 60 and was addressed using early stopping.

Qualitative Trends

- Early epochs: Output images were desaturated or patchy.
- Mid training: Colors became more vivid and edges more defined.
- Final model: Accurately filled polygons with correct colors.

Failure Modes and Fixes

- **Blurry edges**: Resolved using L1 loss.
- **Wrong colors**: Occurred when conditioning was not injected early enough in the model.
- **Output artifacts**: Reduced using dropout and data augmentation.

4 Key Learnings

- Conditional embeddings are effective; the model learned distinct outputs for the same shape with different input colors.
- UNet architecture is suitable for structured image generation and preserved polygon boundaries well.
- Embedding injection should occur early (in the first convolutional layer) to provide strong conditioning.
- Combining L1 and MSE loss yielded better visual fidelity than using either one alone.
- Data augmentation was essential for generalization, especially on rotated or flipped polygon shapes.

Conclusion

This project demonstrates that a Conditional UNet, when combined with simple color conditioning and a stable training setup, can produce high-quality polygon colorizations that generalize well across shapes and classes.