# Project Report: Polygon Colorization (Saketh Nellutla)

This report provides a comprehensive summary of a machine learning project focused on conditional polygon colorization. The objective was to develop a UNet-based model capable of generating a colorized polygon from a grayscale input image and a specified color condition. This document details the selected hyperparameters, architectural design, training dynamics, and key conclusions drawn from the project.

## 1. Hyperparameters

The model's training parameters were established to balance computational efficiency with model performance. An important finding was that the model achieved convergence significantly earlier than the full training schedule, suggesting an opportunity for optimization.

### 1.1 Training Parameters

| Parameter | Value | |
|---|---|---|
| **Epochs** | 75-300 | A number of epochs were initially selected to ensure full convergence, particularly given the relatively small size of the dataset. |
| **Batch Size** | 8 | This batch size was selected to facilitate stable gradient updates while remaining within the available GPU memory constraints. |
| **Optimizer** | Adam | The Adam optimizer was chosen for its robust performance and efficient convergence on a wide range of deep learning tasks. |
| **Learning Rate** | 1e-4 | This value represents a standard and effective starting point for the Adam optimizer, providing a good balance between the speed of learning and the prevention of instability. |

### 1.2 Loss Function Parameters

- **Loss Function:** Custom ColorSegmentationLoss
    - alpha = 1.0(pixel wise accuracy)
    - beta = 2.0(colour consistency loss)

# 2. Model

The model's core is a UNet architecture, which is particularly well-suited for image-to-image translation tasks. The key architectural considerations focused on the most effective method for integrating the conditional color information.

## 2.1 Conditioning Mechanism

- **Method: Full Feature-wise Linear Modulation (FiLM)**
  FiLM layers were integrated into both the encoder and decoder blocks of the UNet. This approach was found to be highly effective, as it allows the color condition to influence feature learning at multiple levels of the network hierarchy, leading to a more robust and reliable output compared to more limited conditioning methods.

## 2.2 Input and Output Structure

- **Input:** The model receives a grayscale polygon image and a colour vector(one-hot) that specifies the target color.
- **Output:** The model generates an image containing the colorized polygon.

## 2.3 Custom Loss Function

The ColorSegmentationLoss was used as a composite function combining a standard pixel-wise reconstruction loss with a consistency loss. This was a crucial design choice for mitigating the initial failure mode of inconsistent color filling and ensuring the uniformity of the final output.

# 3. Training Dynamics and Analysis

Training was monitored via Weights & Biases, which provided valuable insights into the model's performance and learning progression.

## 3.1 Training Progression

| Stage | Qualitative Output |
| --- | --- |
| **Initial Stages** | Outputs were characterized by noise and indistinct boundaries. |

| | |
|---|---|
| **Mid-Training** | Polygon boundaries became clearer, but the interior color was often inconsistent. |
| **Final Stages** | The model consistently produced clean boundaries and uniformly filled polygons. |

### 3.2 Observed Failure Modes and Corrective Measures

| Challenge | Corrective Actions |
|---|---|
| **Primary Challenge:** The most significant challenge was achieving consistent and accurate color prediction using the basic UNet model. Early attempts using numerical encoding for colors were ineffective. | 1. The approach was revised to use one-hot encoding for color conditioning, which provided a more distinct signal.<br><br>2. Full FiLM was implemented to ensure the color signal's influence throughout the network.<br><br>3. Later the FiLM layer was only applied to the decoder layer as I thought that is where the coloring part takes place, but there was not much difference in both runs. Full FiLM results seemed satisfactory.<br><br>3. A weighted consistency term **(**beta**)** was integrated into the custom loss function to directly penalize non-uniform colorization within the predicted regions. |

# 4. Key Learnings and Conclusions

1. The Efficacy of Custom Loss Functions**:** For specialized generation tasks, a standard loss function may be inadequate. Developing a custom loss that directly addresses and penalizes specific failure modes, such as the color inconsistency addressed by the beta term in ColorSegmentationLoss, is essential for achieving high-quality results.
2. Importance of Early Stopping**:** The training data revealed that the model converged well before the completion of the 75-epoch training schedule. This highlights the importance of implementing early stopping to optimize computational resource usage and reduce training time without compromising final model performance.
3. Comprehensive Conditioning is Critical**:** The success of the Full FiLM architecture demonstrates that for conditional tasks, embedding the conditional signal across the entire network hierarchy is

more effective than localizing it to a single point. This approach enables the model to leverage conditional information more effectively at all stages of feature extraction and generation.

4. Dataset Size and Quality is Important**:** My small dataset was the reason I had to run so many epochs and was a big challenge at the start. It was hard for the model to learn to generalize well from limited examples. Data Augmentation has helped here and got to know how important it is.