Mid Evaluation Report for BYOP 2026

Hephaestus

Vedansh Sethi
Department of Computer Science and Engineering
IIT Roorkee

December 22, 2025

1 Task Description

1.1 Task

The project aims to -

- Use Deep Learning to find outliers in a batch of input images, using the input images as context
- Use interpretation methods to understand the model's behavior

1.2 Detailed Description

The primary aims of mine are as described in **1.1**
The interest of mine is to explore disentanglement in latent space and how the different parts of the model used to generate the latent space, i.e. encoder and decoder, work

1.3 Stages of the project

There are primarily 4 stages of the project's development -

- **Experimentation -**
  In this stage, I play with different hyperparameters on various datasets in order to find the best fit for our need
- **Training and Proving Model Competency -**
  Once I train our model using inferences from the experiments, I then prove the model competence using various metrics
- **Catching Outliers with the Model -**
  This is the stage in which I explore different metrics to catch outliers semantically, and set up a testing pipeline for the outlier detection
- **Understanding The Model -**
  This is the stage in which I implement different methods to interpret the models's working

2 Methodology and Details

2.1 Stage 1 : Experimentation

This part, as discussed in **1.3** is experimenting with different hyperparameters
The conducted experiments were as follows -

- Training a VAE on MNIST dataset
- Training a β-VAE on dSprites dataset using multiple hyperparameter values (β)
- Training a γ-VAE on dSprites dataset using multiple architectures, hyperparameters (γ and C_{max}) and loss structures (constant and decaying γ)
- Training a β-TCVAE on dSprites dataset

Details of the experiments -

A specific result which astonished me during the experimenting phase was the graph of losses while training γ-VAE on Dsprties dataset

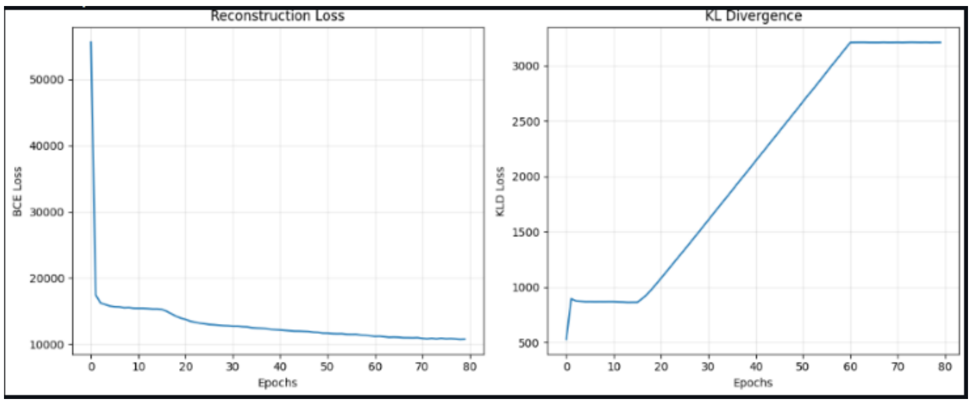


Figure 1: Partwise Loss while training γ-VAE after introducing Convolutional layers

We can see in Fig. 1, from epoch 5 to 15, neither of the two losses are decreasing and are effectively at a plateau, this is strange behavior, and it emerged only after the introduction of Convolutional layers in the architecture of the model.
The results of other Experiments are on the Github repository of the project

2.2 Stage 2 : Training and Proving Model Competence

Dataset

The final Dataset used to train the model is Shapes3D, it is a standard dataset used in disentanglement related tasks, it is synthetically generated dataset formed on basis of 6 ground truth factors -

1. Floor Hue (10 Values)
2. Wall Hue (10 values)
3. Object Hue (10 values)
4. Object Shape (4 values)
5. Scale (8 values)
6. Orientation (15 values)

Training

I have used a β-TCVAE in order to maximize disentanglement of latent space while not compromising the reconstruction quality
Hyperparameters of the model are -

- Learning Rate = 5e-4
- Epochs = 50
- Gamma = 1
- Beta = 6
- Anneal Steps = 5000

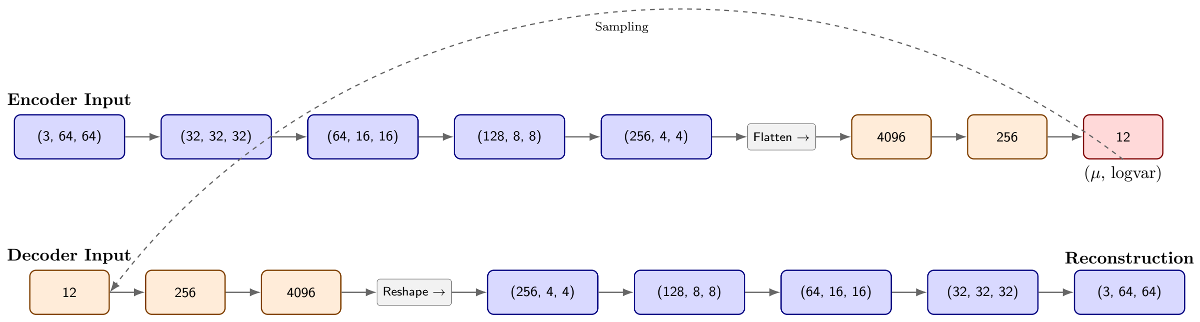
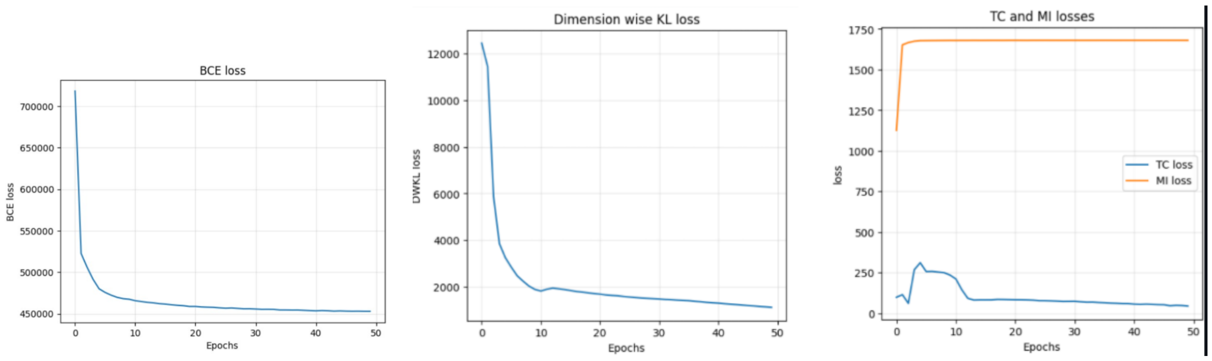


Figure 2: Block diagram illustrating the β-TCVAE architecture. Blue blocks represent 3D convolutional feature maps, while orange/red blocks represent 1D dense vectors. The transition points involve flattening or reshaping operations.



(a) Reconstruction Loss (b) DWKL loss (c) TC and MI loss

Figure 3: Loss profiles of the training of β-TCVAE used in the project

Proving Model Competence

I use MIG and SSIM to prove the model is competent enough to catch outliers

MIG or Mutual Information Gap, it is used to analyze the disentanglement of the model, it correlates latent code features to actual labels of the data and then uses the first and second most correlated latent code for each data label and calculates the gap between their correlation,

the larger the gap for a label, the more disentangled it is
The MIG score of the model was 0.6235

SSIM, or Structure Similarity Index Measure is used to analyse the reconstruction quality of the image, it uses 11 X 11 kernel made using 2D gaussian distribution centered at the central cell of the kernel it uses the kernel to extract certain features, namely luminance, contrast and structure for both original and reconstructed images, compares them, the takes a weighted product of these factors
The SSIM score of the model is 0.9195

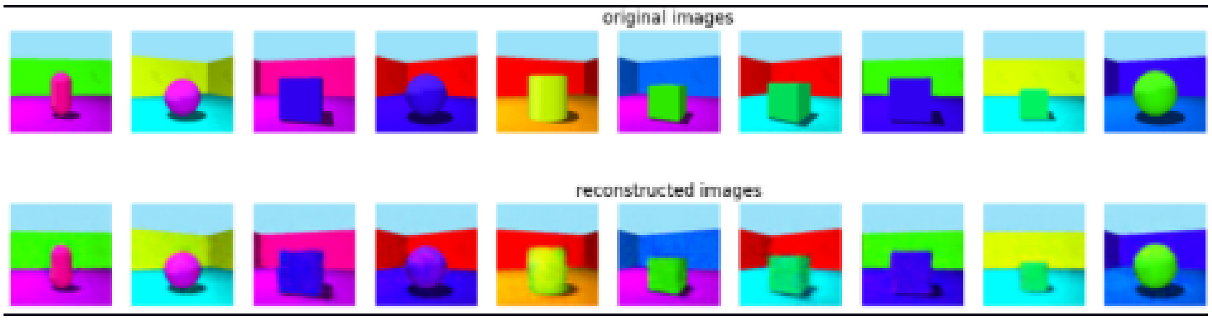


Figure 4: Reconstruction of images using the β-TCVAEs decoder

2.3 Stage 3 : Catching Outliers with the Model

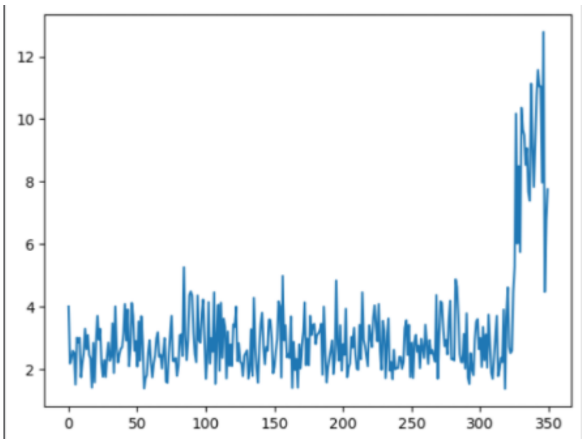
This stage included the exploration of various metrics to detect semantic outliers, the two candidates were Euclidean Distance from the mean of the batch, and Mahalanobis Distance from the mean of the batch. Finally I chose Mahalanobis Distance (MD) because -

- It is scale invariant, it measures distance in terms of units of standard deviation rather than plain axes values
- It takes into account the trends of the batch because of the covariance matrix term, this makes it an excellent choice for semantic outlier detection

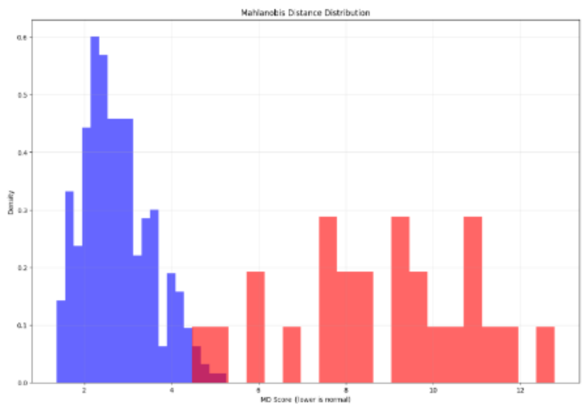
Testing Pipeline

I use a CuratedSampler class which is used to make a batch of images with certain characteristic, and some randomly that act as the semantic outliers I then calculate the MD for each of the samples, then plot MD density distribution, decide a threshold and all samples with scores greater than the threshold are flagged as outliers

Results



(a) MD variation with image index for 350 samples with 25 outliers, the outliers were put at the end of the batch and we can see clear peaks towards the end of the graph



(b) Density Distribution of MD for the whole batch where the red ones are the scores of outliers, we can see a clear separation between outlier scores (red) and normal scores (blue) with very less overlap

AUROC is the Area Under FPR (False Positive Rate) vs. TPR (True Positive Rate) curve, it should be high in order to indicate good detection
AUROC for this model is 0.9990
The precision for this model calculated across multiple statistically significant batches of images is 85 percent

2.4 Stage 4 : Understanding the Model

The project is currently at this stage, I have used Mutual Information Matrix, Latent Traversals, and Counterfactual Generation as methods of verifying inferences and get new details about the model

Mutual Information Matrix

The Mutual Information we calculated for MIG, before the sorting was in the form of matrix of shape (Latent Dimensions X Dataset Labels), this matrix whe plotted is the Mutual Information Matrix
In Figure 6, we can clearly see z_0 is wall hue factor, z_4 is orientation factor, z_7 is floor Hue, and z_8 is object hue factor, whereas z_5 is loosely related to scale, while shape factor is still entangled

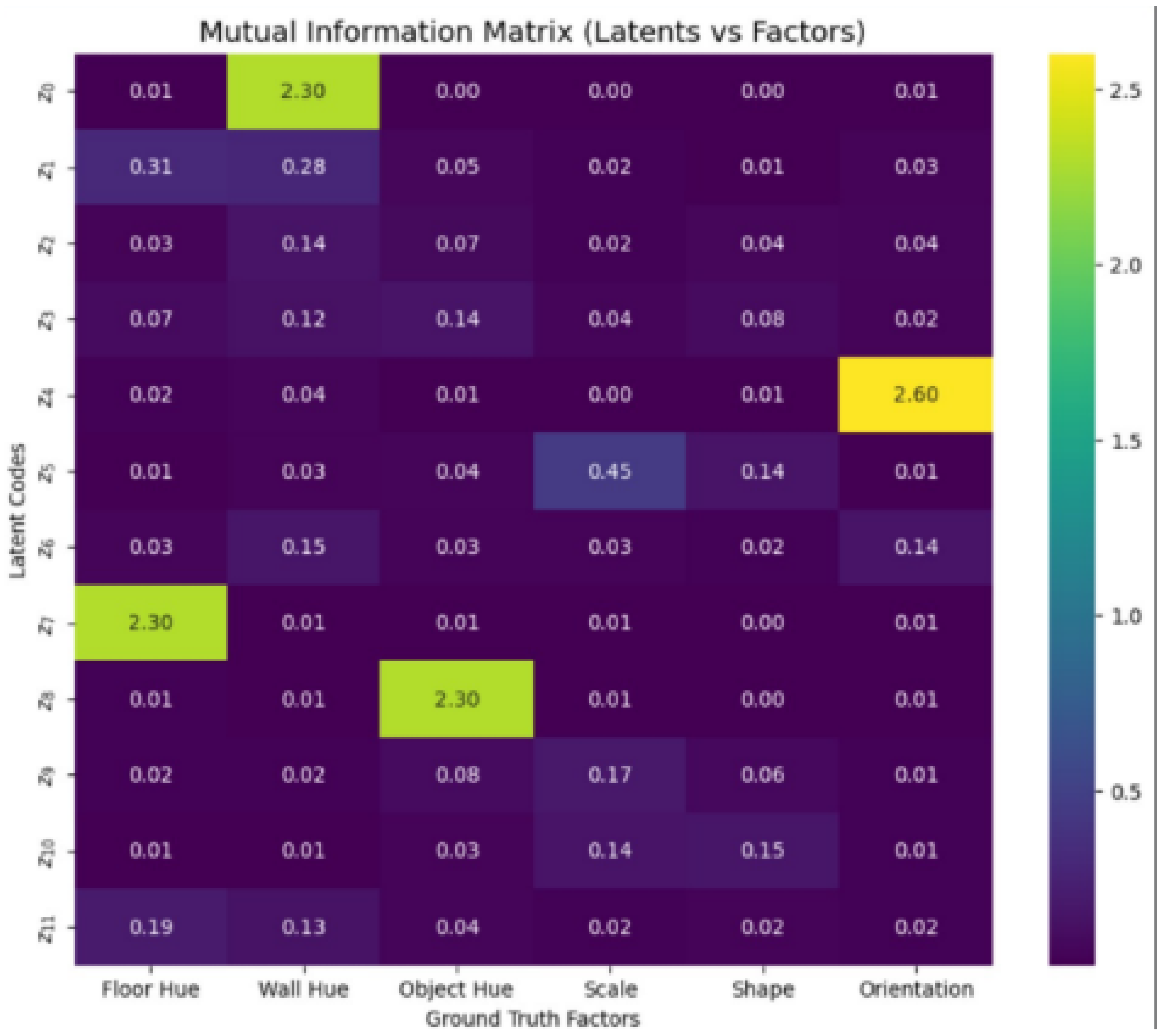


Figure 6: Heatmap of Mutual Information Matrix to show correlated factors

Latent Traversals

Latent Traversal is observing the variation of properties of an image of the dataset while we change only one component of the latent code
It can be used to see the verify claims inferred from the above discussed MI matrix
If we compare the MI matrix (Figure 6) with Latent Travesal from Figure 7, we see a clear verification of the following -

- z_4 (row 5) controls solely the orientation
- z_7 (row 8) controls solely the floor hue
- z_1, z_2, z_3, z_6, z_9 and z_{11} are dead dimensions, they have no effect on the properties of the image
- Shape seems to be jointly controlled by 2 components, that is z_5 (cube to capsule) and z_{10} (cube to sphere), but that too is not very well controlled

There are few unverified inferences that can be drawn from both figures

- While z_8 is highly correlated with object hue and shows no correlation with shape factor, the latent traversal shows a change in shape, it can be due to either range overshoot, that is, it condensed the object hue factor in less space than (-3, 3) or it is yet again a proof of decoder's bias

- z_0, while perfectly controls wall hue, also projects it's change on shape of object, while there is no correlation in MI matrix, hence this can be due to decoder's bias, that is a flaw of the dataset it was trained on
- The dead dimensions in MI matrix have mild correlation with multiple factors, but they do not show effect due to monopoly of one component over that factor

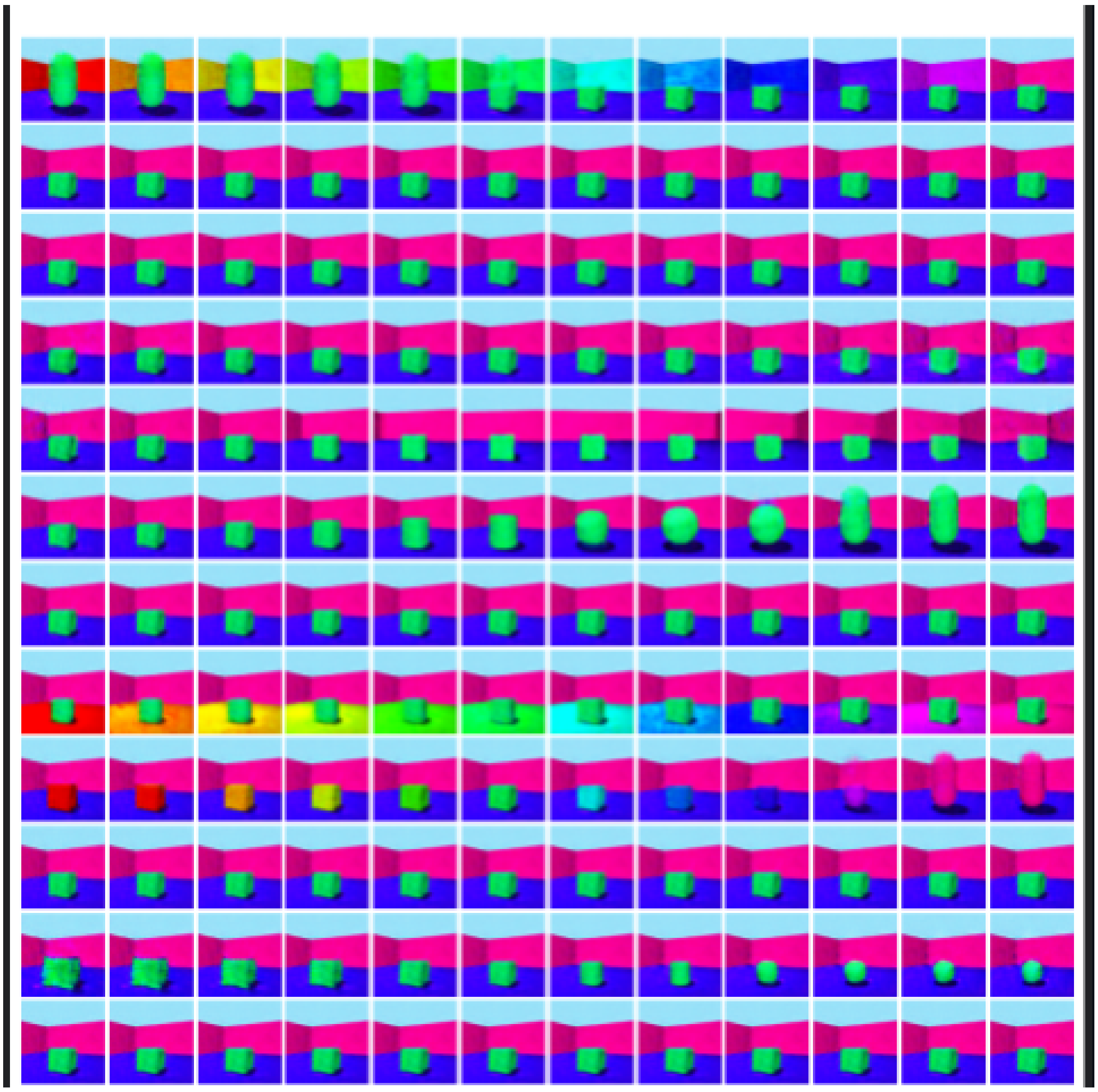


Figure 7: The Latent Traversal from (-3, 3) for each component of latent code

Counterfactual Generation

The technique is described as following -

1. I take a batch of images, calculate the MD for each of the images
2. Find an appropriate threshold to avoid false positives

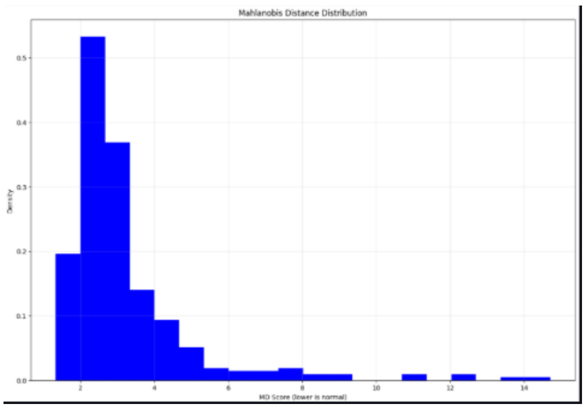
3. Calculate Z-scores (dimension wise distance from the mean in terms of standard deviation of dimension across the normal images) for each outlier classified image
4. I analyse the Z-scores by seeing which dimension was the farthest from the mean of the normally classified images
5. Then I replace that dimension's value with the value of mean of the dimension across the normally classified images
6. The latent code then generated is called the counterfactual and corresponding decoded image is called the counterfactual

Two experiments were conducted using this technique, which are described below

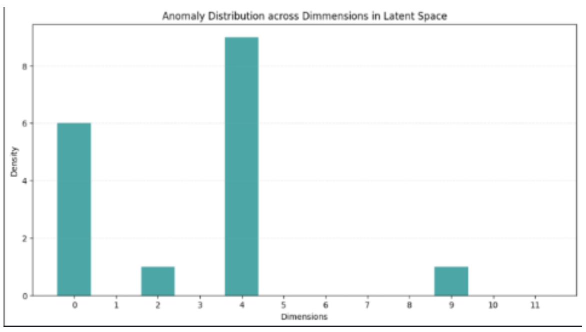
Counterfactual Generation : Experiment 1

This experiment was conducted on a set of 320 images with 20 outleirs and we do not know which are the outliers
The criteria for normal is as follows -

- Orientation - Forward facing
- Shape - Sphere
- Wall Hue - Green



(a) MD distribution of the test batch, it is evident that the outliers are less and away from the majority



(b) Dimension-wise Outlier Distribution identified using Z-scores, we can see that outliers majorly are in z_0 and z_4, which control the factors we fixed in the normal pictures

Figure 8

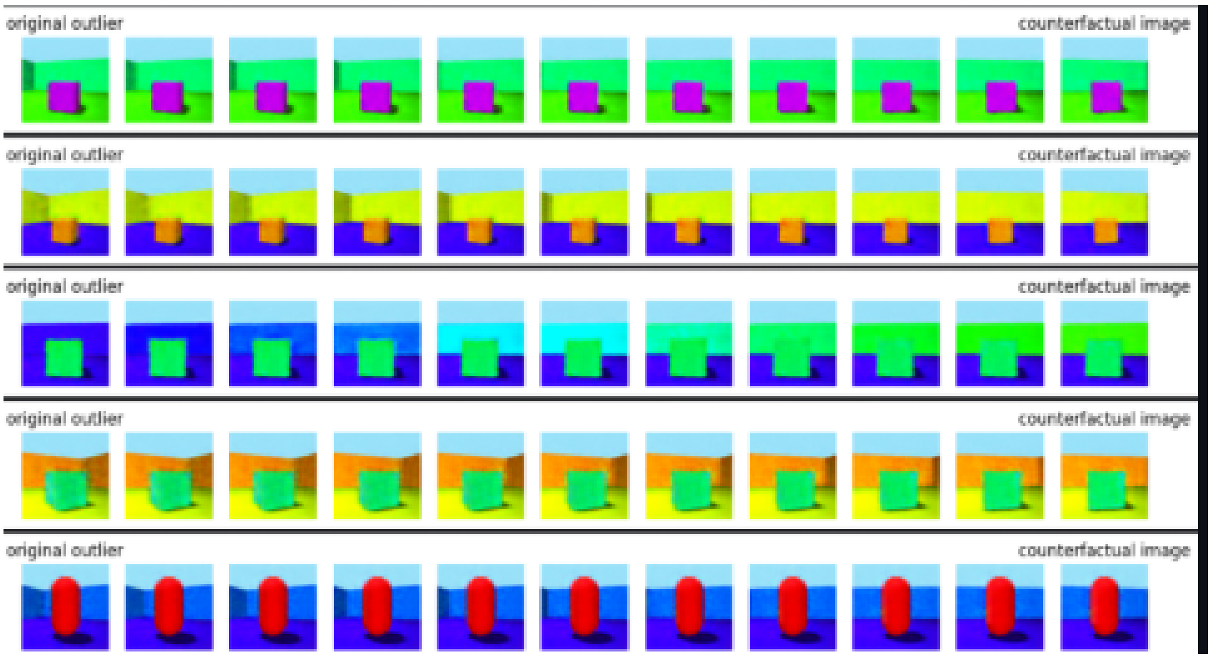


Figure 9: A few samples morphing from outlier image to counterfactual image

Inferences that can be drawn from the above experiment while keeping in mind the above derived results of MI matrix and Latent Traversal are -

1. Row 1, 2, 4, and 5 (Figure 9) are orientational outliers and their orientation changes pretty neatly to the intended one, this states the model is not only entangled, but the metrics applied are competent enough to fix outlier images based on context
2. Same goes with row 3 (Figure 9), where the outlier is a Wall Hue one, which too changes neatly from purple to green

Counterfactual Generation : Experiment 2

This experiment was conducted on a batch 320 samples, 20 outliers, with similar condition that we do not know the outliers
The criteria for normal is such -

- Wall Hue - Green
- Floor Hue - Yellow
- Orientation - Forward Facing

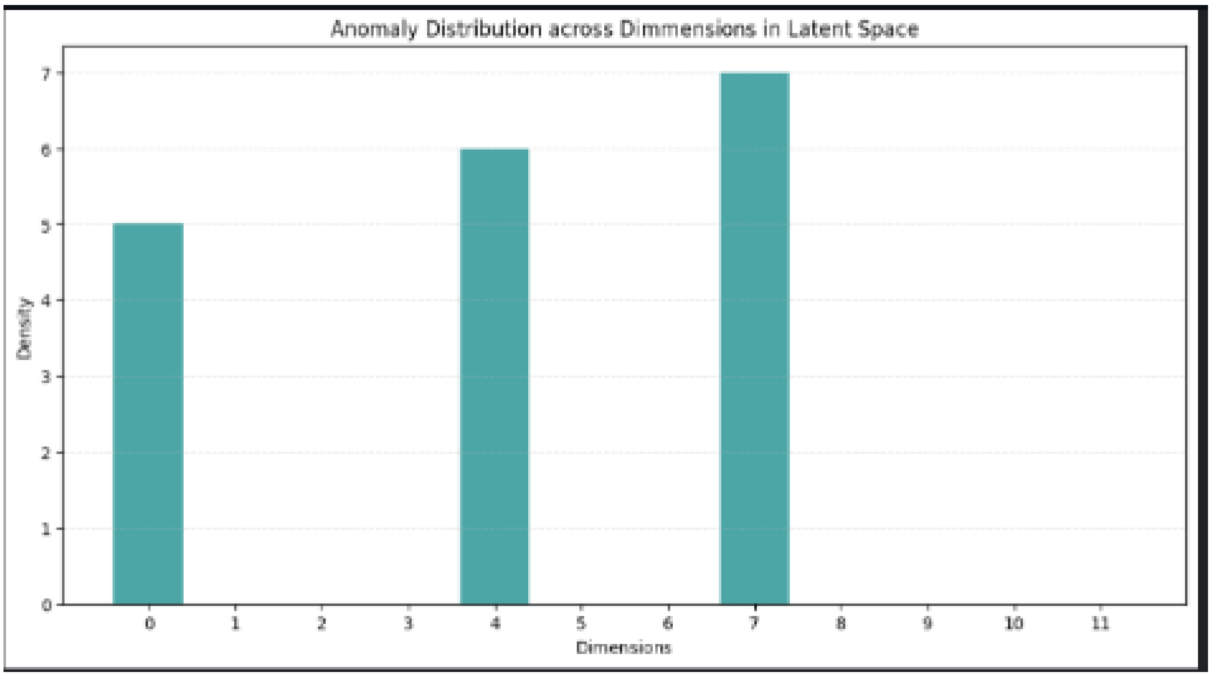


Figure 10: Dimension-wise outlier distribution for the current batch, similar trends can be observed as Figure 8(b) in corresponding dimensions

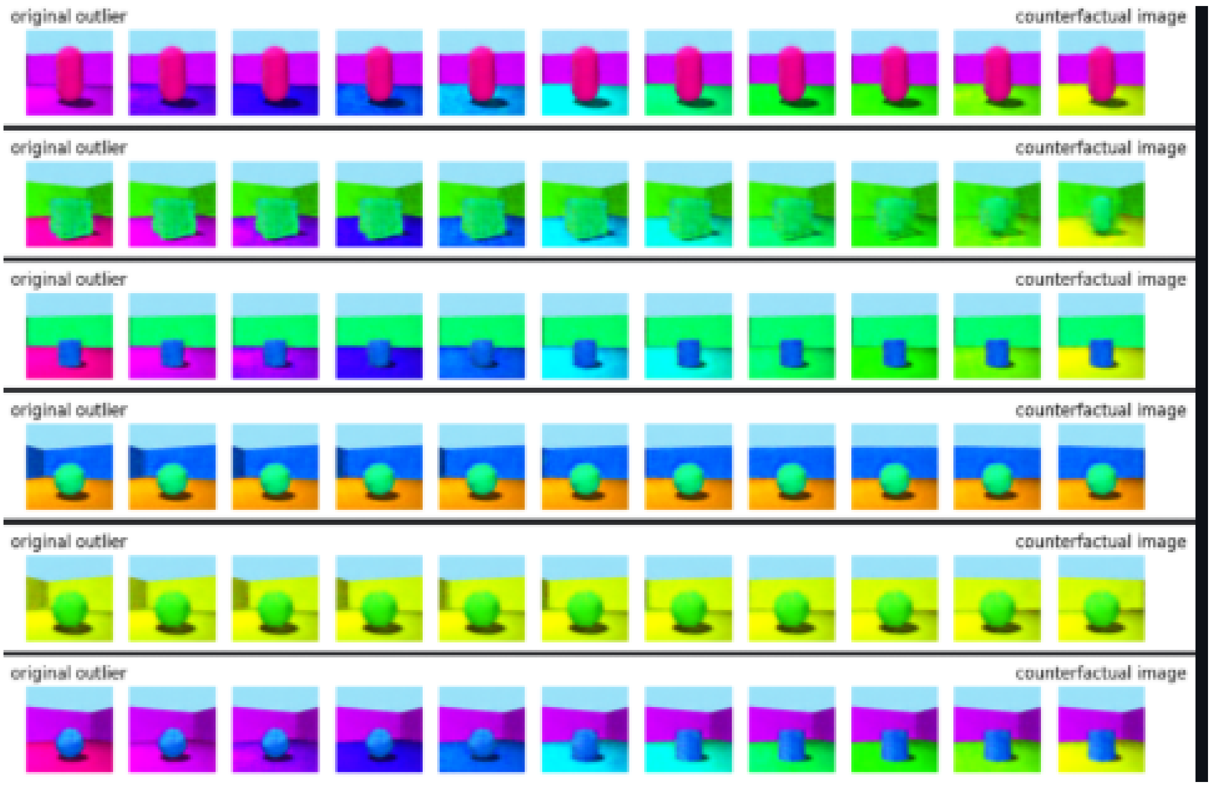


Figure 11: A few samples morphing form outlier image to counterfactual image

Inferences that can be drawn from the above experiment keeping in mind the above derived results of MI matrix and Latent Traversals are -

1. The outliers are strictly in z_0, z_4 and z_7, which were the intended dimensions

2. Floor Hue changes smoothly in Row 1 and Row 3, but the same cannot be seen in Row 2 and 6, and the change is in shape, as we saw in the latent traversal, this gives evidence again about the decoder's bias thoery

3 End Term Goals

There are majorly two avenues where the project will go from here on, first is to push harder on the Counterfactual Generation Method and explore resutls it can yield. The other tangent is Mechanistic Interpretation, it can be tough to look into both so the priority will be to explore Mech. Interp. first and then if time premits, go more into Counterfact. Gen.

3.1 Mechanistic Interpretation

I read one or two articles in the circuits thread by Anthropic recently, and I am interested to know which part of encoder uses which circuit element to identify certain features and then generate the latent code

The same goes with the decoder, I plan to look into how the latent code is converted from numbers to the image, and which layer of decoder plays prominent part in generation of the image

I have already started some work on it by implementing Activation Monitoring mechanisms and producing some activation graphs for the encoder's layers

3.2 Counterfactual Generation

I plan to look further into this section by experimenting with criteria for normal images

This can be helpful in confirming the extent to which shapes are entangled with other factors and a make the decoder biases we discussed earlier more visible

—

Each of the above goals are pretty time consuming, and they might not be completed to their fullest, that is why I have made clear a priority list, any recommendations are welcome

4 References and Additional Information

The timeline of the project is available at Google Doc

The Code for the project is available on the Github Repository associated with this project

References

1. Deep Learning for Computer Vision course from University of Michigan
2. Stanford CS229 By Andrew Ng
3. Understanding VAEs
4. Understanding the math of VAEs
5. Understanding the math of β-VAEs, metrics of disentanglement and other variations of VAEs

6. Learning Basic Visual Concepts through Variational Framework
7. Understanding Disentanglement in β-VAEs
8. Isolating Sources of disentanglement in VAEs
9. AntixK Pytorch VAEs
10. Circuits Thread by Anthropic