End Term Report for BYOP 2026

Hephaestus

Vedansh Sethi
Department of Computer Science and Engineering
IIT Roorkee

January 5, 2026

1 Abstract

This project aims to make a model which aims to identify outliers in a given input of images using the context of the input. This project also aims to understand the working of the model, esentially trying to dissect it
The methodology used focuses on disentanglement as it is the corner stone of this project, this project relies completely on good disentanglement of the latent space
The final outcome of the project is an outlier detection pipeline using a model trained on Shapes3D dataset, the outcomes also include a view at how the model processes the input, in the form of a comprehensive study

2 Preliminary Details of the Project

2.1 Data and Pre-processing

The dataset is Shapes3D dataset which was designed by Deepmind synthetically to study disentanglement specifically
The parameters of the dataset are as follows -

- Floor Hue (10 Values)
- Object Hue (10 Values)
- Wall Hue (10 Values)
- Object Shape (4 Values)
- Object Scale (8 Values)
- Scene Orientation (15 Values)

2.2 The Flow of Project

The project was carried out in four stages -

1. Experimentation
2. Training the Model and Proving it's Competence
3. Setting up the Outlier Detection Pipeline
4. Understanding the Working of the Model

3 Details of the Execution

3.1 Stage 1 : Experimentation

This stage focused on trying out different model architectures in order to understand what works best for the project, and to get a hang of the libraries in python. There were multiple models trained using different architectures and hyperparameters
The experiments are listed as follows -

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

As mentioned in the Mid Term report, there was one interesting result from the experiments conducted. Other results were not as interesting and are uploaded on Github repository of the project

3.2 Stage 2 : Training the Model and Proving its Competence

Training the Model

The model finally used was the architecture of β-TCVAE which had the following hyperparameters -

- Learning Rate = 5e-4
- Epochs = 50
- $\beta = 6$
- $\gamma_{max} = 1$
- Anneal Steps (N) = 5000 (10 epochs)

The loss structure of the β-TCVAE is given by:

$$\mathcal{L} = \mathbb{E}_q[\log p(x|z)] - \alpha I(z;x) - \beta D_{KL}\left(q(z)||\prod_j q(z_j)\right) - \gamma \sum_j D_{KL}(q(z_j)||p(z_j))$$

Where

$$\gamma = \min\left(\frac{n}{N}, 1\right)\gamma_{max}$$

n = Current Step

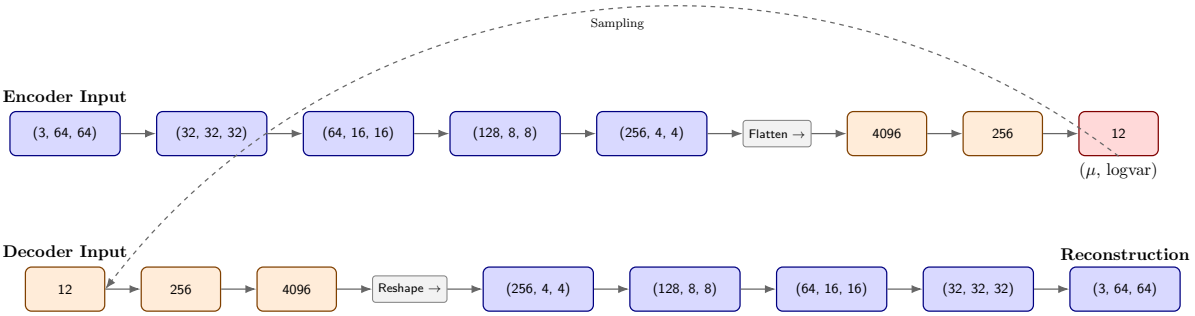
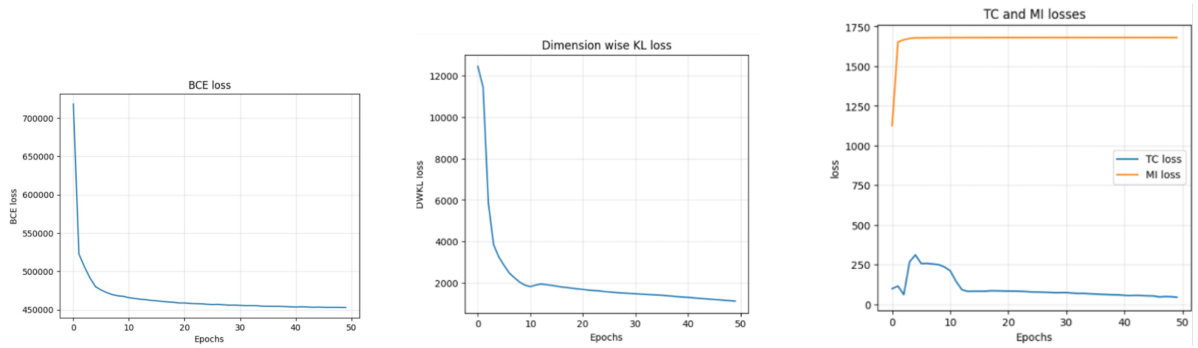


Figure 1: Block diagram illustrating the β-TCVAE architecture. Blue blocks represent 3D convolutional feature maps, while orange/red blocks represent 1D dense vectors. The transition points involve flattening or reshaping operations.



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

Figure 2: Loss profiles of training of Final Model used in the project

Proving Model Competence

I use MIG and SSIM to test the model

Mutual Information Gap (MIG) -

It is used to analyze the extent of disentanglement of the model, we see how that how likely is it that a certain value of a latent code and a certain value of ground truth factor occur together we do so for all the latent codes and all the ground truth factors, then we use this data to see how much mutual information does one latent code share with a ground truth factor.
For each ground truth factor v_k we find out $z_{(1)}$ and $z_{(2)}$, the latent codes with maximum and second maximum mutual information with v_k, then MIG for v_k is

$$\text{MIG}_{v_k} = \frac{\text{I}(v_k; z_{(1)}) - \text{I}(v_k; z_{(2)})}{H_{v_k}}$$

where I is mutual information function and H is the total entropy of the ground truth factor
The final MIG of the model, i.e. the mean of MIGs for all the v_k was 0.6235.

Structure Similarity Index Measure (SSIM) -

It is used to analyze the reconstruction quality of the model
For that it uses an 11×11 kernel made with weights in the form of a 2D Gaussian distribution.

It analyzes different qualities like luminance, contrast and structure for a pair of images, it finally calculates

$$\text{SSIM}(I_1, I_2) = \left(\frac{2\mu_1\mu_2}{{\mu_1}^2 + {\mu_2}^2}\right)\left(\frac{2\sigma_{12}}{\sigma_1^2 + \sigma_2^2}\right)$$

The SSIM score for the model averaged across a batch of images was 0.9195

An MIG of 0.6235 while maintaining SSIM of 0.9195 is a proof that the model is well disentangled while maintaining reconstruction error low

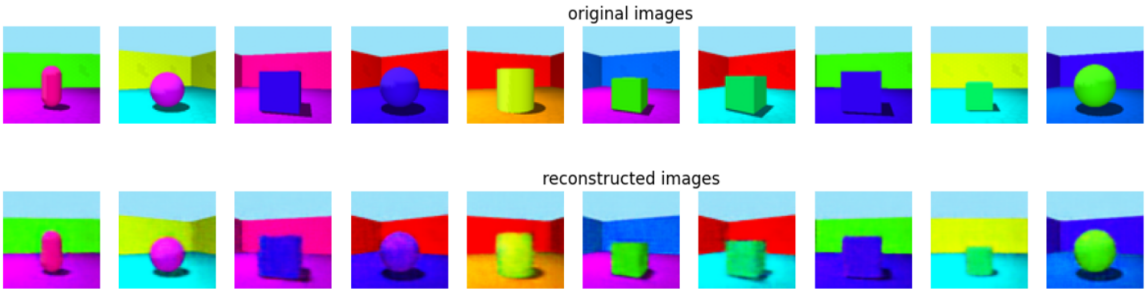


Figure 3: Reconstruction of images using the β-TCVAE decoder

3.3 Stage 3 : Outlier Detection Pipeline

In this stage I set up a pipeline to test the outlier detection capabilities of the model. The pipeline is -

1. Curated Sampling from the dataset for normal images and outliers
2. Calculation of the decided metric for the whole input batch
3. Final method of classifying a sample as outlier

Curated Sampler

It is a custom class made to sample specifically according to the needs of the task, it includes **Normalcy Criteria** as an input, i.e. it selects pictures with certain features as normal samples, and selects outliers randomly from the leftover dataset

Metric Used

Instead of the normally used **Euclidean distance** from the mean, I use the **Mahalanobis Distance** as the metric for detcting outliers as MD takes into account the shape of the input batch rather than assuming it to be spherical
Mahalanobis Distance is calculated as follows -

$$\text{MD}(\mathbf{x}, \boldsymbol{\mu}) = \sqrt{(\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu})}$$

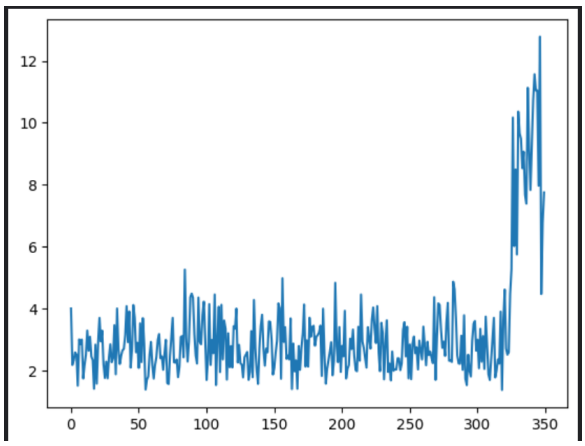
where $\mathbf{x}$ is input features, $\boldsymbol{\mu}$ is the mean of the input batch features, and Σ is the covariance of the input batch features

Results

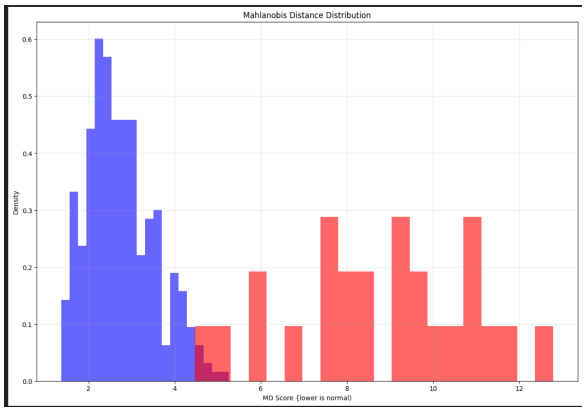
The detailed outlier detection method is as follows -

- Pass the input images through encoder to map them to latent space
- Calculate the mean vector ($\boldsymbol{\mu}$) of the batch and MD of each image from $\boldsymbol{\mu}$
- Plot the histogram of density of MD values for images, the ones which are visibly away from the peak, i.e. with abnormally high values are the outliers
- Decide an appropriate threshold, samples with MD greater than this will be flagged as outliers
- Keep threshold such as to avoid Formulationalse Positives

The test batch has 350 samples in total, with 25 outliers



(a) graph of MD distance for test batch, the outliers were kept at the end of the batch, and we can see the spike in MD for the samples towards the end



(b) The density plot of MD scores for test batch, the outliers have their scores in red color, we can see that the red scores are mostly distinct from the normal scores by a clear margin

Figure 4: Visual results for the outlier detection pipeline

Area Under Receiver Operating Characteristic curve (AUROC)

To evaluate the performance of the outlier detector, we utilize the **Area Under the Receiver Operating Characteristic (AUROC)**. AUROC is a threshold-independent metric that quantifies the model's ability to distinguish between normal samples and outliers.
The AUROC is calculated by integrating the area under the ROC curve, which plots the **True Positive Rate (TPR)** against the **False Positive Rate (FPR)** at various threshold settings:

- **True Positive Rate (Recall):** The proportion of actual outliers correctly identified as outliers.
- **False Positive Rate:** The proportion of normal samples incorrectly classified as outliers.

An AUROC of 1.0 indicates a perfect classifier, while an AUROC of 0.5 suggests the model performs no better than random guessing. The model had an AUROC of 0.999

3.4 Stage 4 : Understanding the Model

This is the final stage of the project, in this stage, I implemented multiple techniques to understand how the model processes the data, in essence trying to understand how the model "thinks"

Mutual Information Matrix

This is a matrix of $I(z_i; v_k)$ for all pairs of (i, k) in the form of V×M matrix where V is the number of latent factors and K is the number of ground truth factors. It helps us analyze which z_i and v_k share the most information

Table 1: Observations made using the MI matrix in Figure 5

Latent Code (z_i)	**Observed Ground Truth Factor (v_k)**
z_0	Wall Hue
z_4	Orientation
z_7	Floor Hue
z_8	Object Hue
z_5	roughly relates to scale

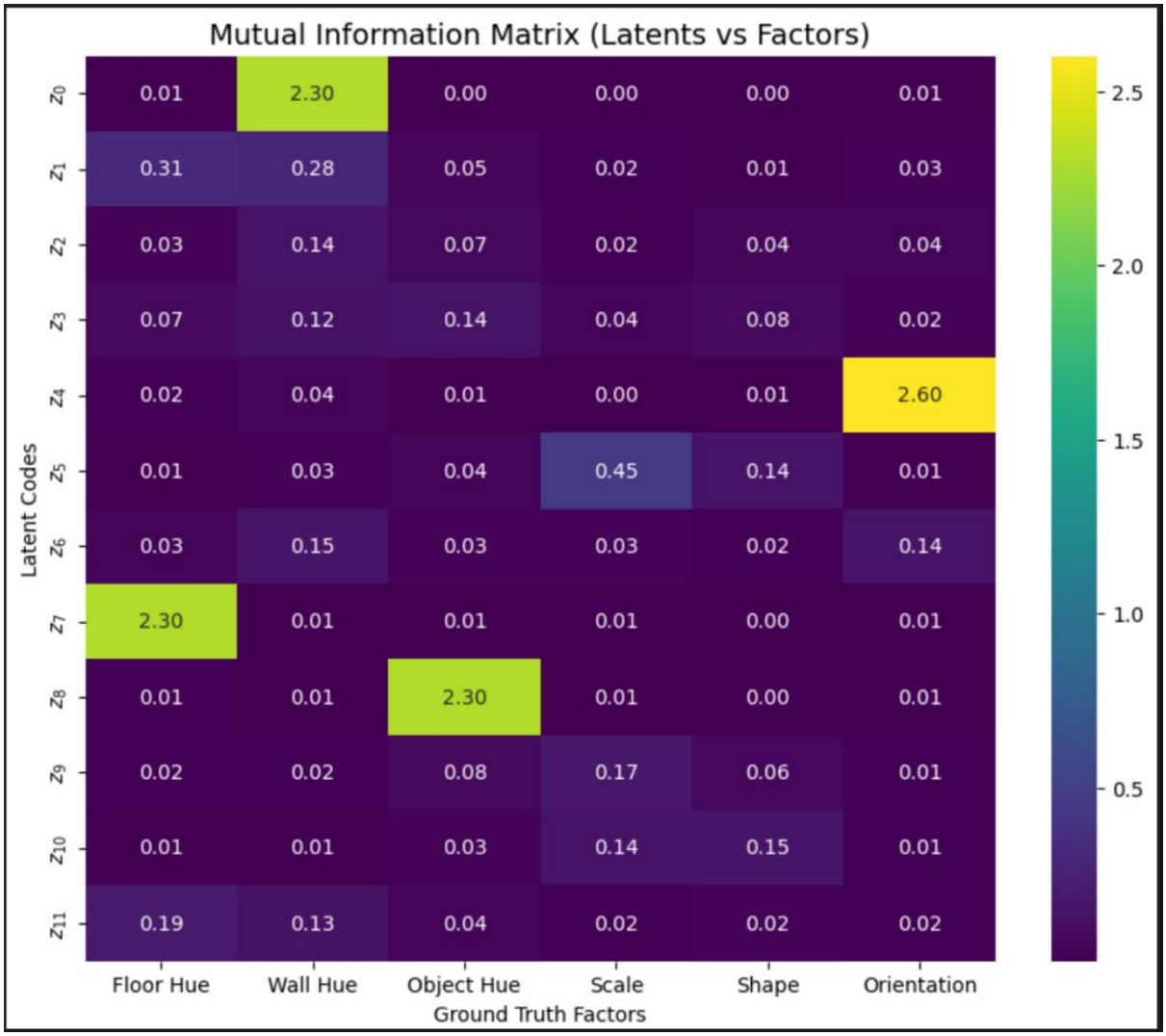


Figure 5: Mutual Information Matrix

Latent Traversals

Technique Description -

Latent Traversal is a series of images generated when one vector, say **z** is varied only across one axis at a time for all the axes in the latent space (here, 12)
In my case, I varied each component from -3 to 3, while keeping other components fixed

Observations that can be drawn from Figure 6 -

- Dimensions z_1, z_2, z_3, z_6, z_9 and z_{11} are effectively dead
- z_0, z_3, z_7 and z_8 follow their expected behavior of changing the factor suggested for them by the MI matrix
- z_0 in the MI matrix shows no mutual information with Object Shape, but the change in Wall Hue also changes the Shape
- z_5 and z_{10} show less mutual information with Shape factor, but we can see that they change the shape of the object Cube to Capsule via Sphere and Cube to Sphere via Cylinder respectively

Unverified claims that can be drawn -

- z_0 shows no mutual information with Object Shape, but still puts effect on it, this can be due to **Decoder Bias** rather than the encoder's mistake, i.e. the training data was biased towards showing a light green cube with pink background, rather than a light green capsule with pink background, so the decoder learnt such
- z_8 changes the Object Color but along with it also changes the shape towards the end, this is again looks like a case of Decoder Bias as the component has no correlationw with the shape factor
- The dead dimensions show correlation with multiple factors at the same time, but show no effect on the final picture, there can be two reasons for it, first reason can be that the domination of one component supresses the effect of other components on a particular factor due to the structure of the Decoder, second reason can be that as the component shares information with multiple factors, their effect in essence cancels out
- I(z_5;v_3) that is mutual information of z_5 and v_3 (Scale) is high because it shows the transition from cube to capsule, where capsule is bigger in size than cube, and the latent traversal of z_5 shows that it transition shape from cube to capsule
- z_{10} and z_5 seem to jointly control the shape factor (v_4), i.e. the model mapped v_4 to a plane made by axes z_5 and z_{10} rather than a single axis

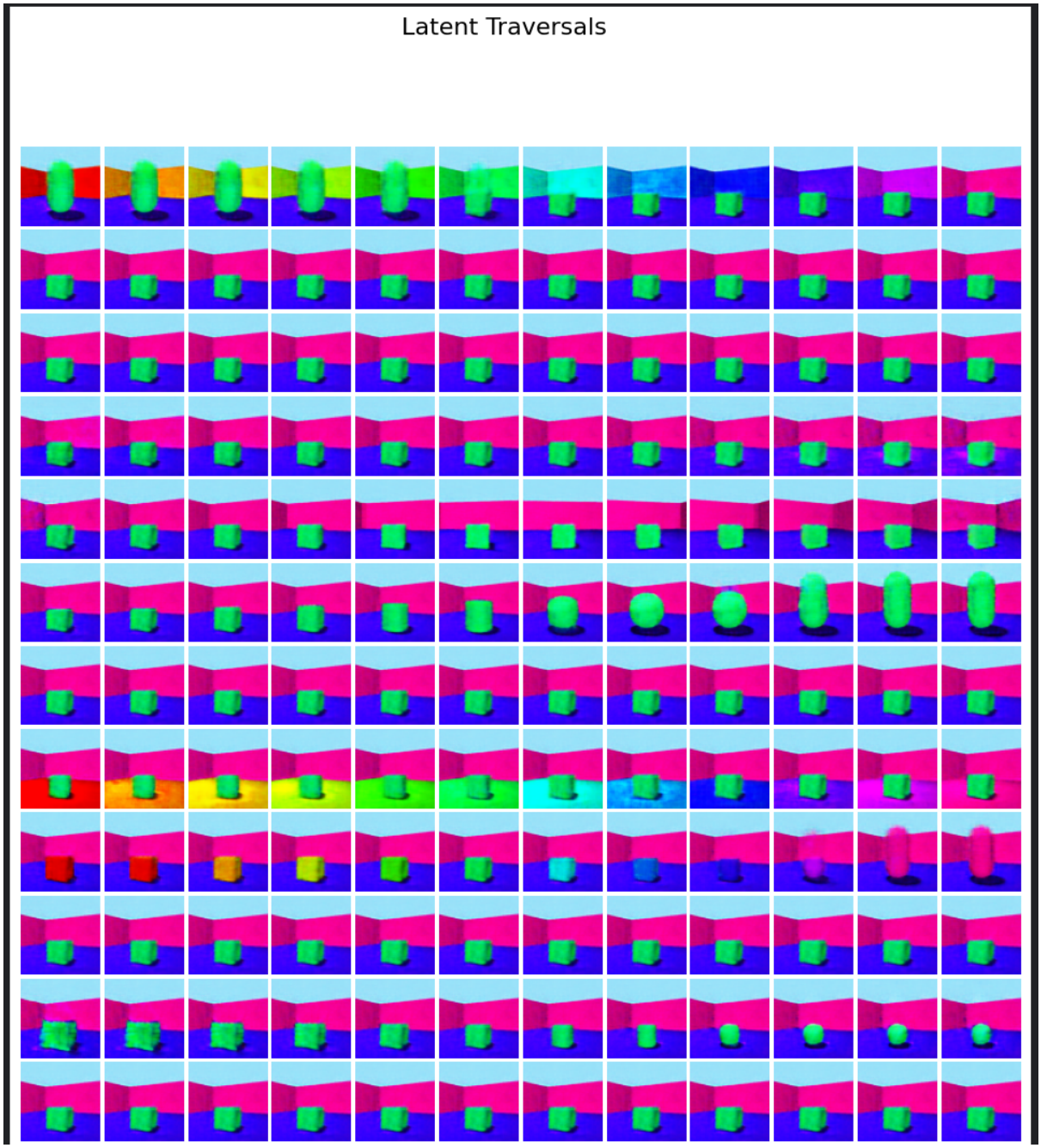


Figure 6: Latent Traversal for each of the 12 dimensions in the latent space

Counterfactual Generation and Dimensional Distribution of Outliers

Technique Description -

1. I take a batch of images, then I run the outlier detection pipeline and isolate the outliers
2. Then I use Z-scores to analyze which component of the sample lead to it being classified as an outlier, let us say for $\mathbf{x}$, the factor is z_4 (Z-score is a vector of sample $\mathbf{x}$ from $\boldsymbol{\mu}$ in terms of standard deviation of that dimension (σ), for each dimension)
3. I then replace $\mathbf{x}_{z_4}$ with the mean value of z_4 across the normal samples

I conducted 2 Experiments with the above technique

Experiment 1

The first experiment batch has 320 samples, 20 of which are outliers
The criertia of a normal sample in the given set is -

- Wall Hue = Green
- Shape = Sphere
- Scene Orientation = Fixed

Observations from the results in Figure 7 and 8 -

- Most outleirs are in dimensions z_4 and z_0, which was intended because we chose Wall Hue and Orientation as the fixed factors, this means that the model and metric work together and well understand the statistical meaning of "normal"

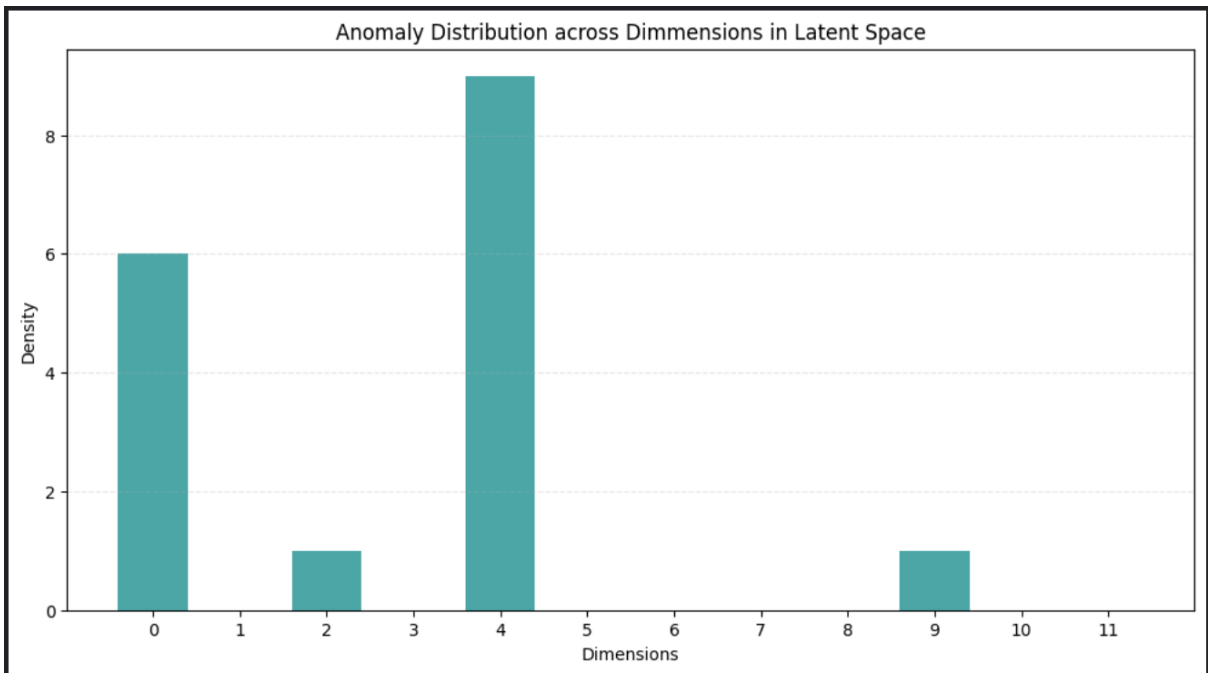


Figure 7: Outlier distribution across dimensions for Experiment 1

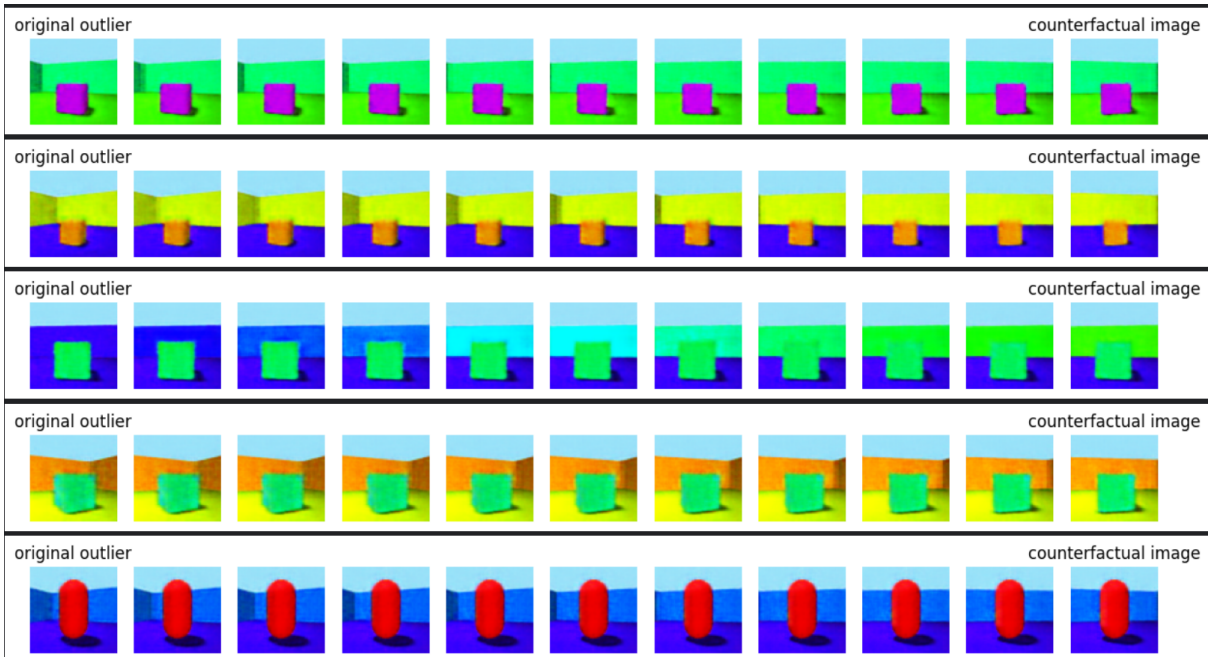


Figure 8: Morphing from outlier sample to Counterfactual for Experiment 1

Experiment 2

The Second experiment batch has 320 samples and 20 outliers
The criteria for normal in the given set is -

- Wall Hue = Green
- Floor Hue = Yellow
- Orientation = Fixed

Observations from the results in Figure 9 and 10 -

- The Outliers are strictly in z_0, z_4 and z_7 as intended
- Row 2 and 5 of Figure 10 also changes the shape along with the floor color

Unverified Claims that can be drawn and reinforced -

- The Decoder Bias claim gets stronger because of the fact that changing the floor hue also changes the object shape despite negligible value of $\mathrm{I}(z_0; v_4)$

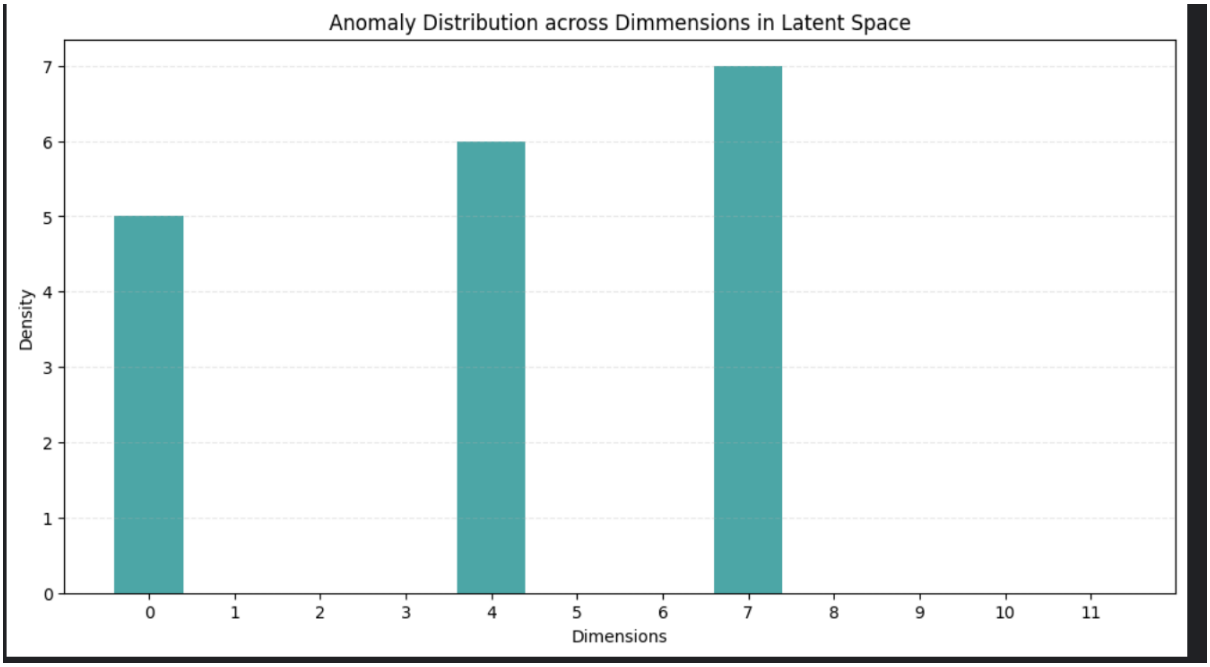


Figure 9: Outlier distribution across dimensions for Experiment 2

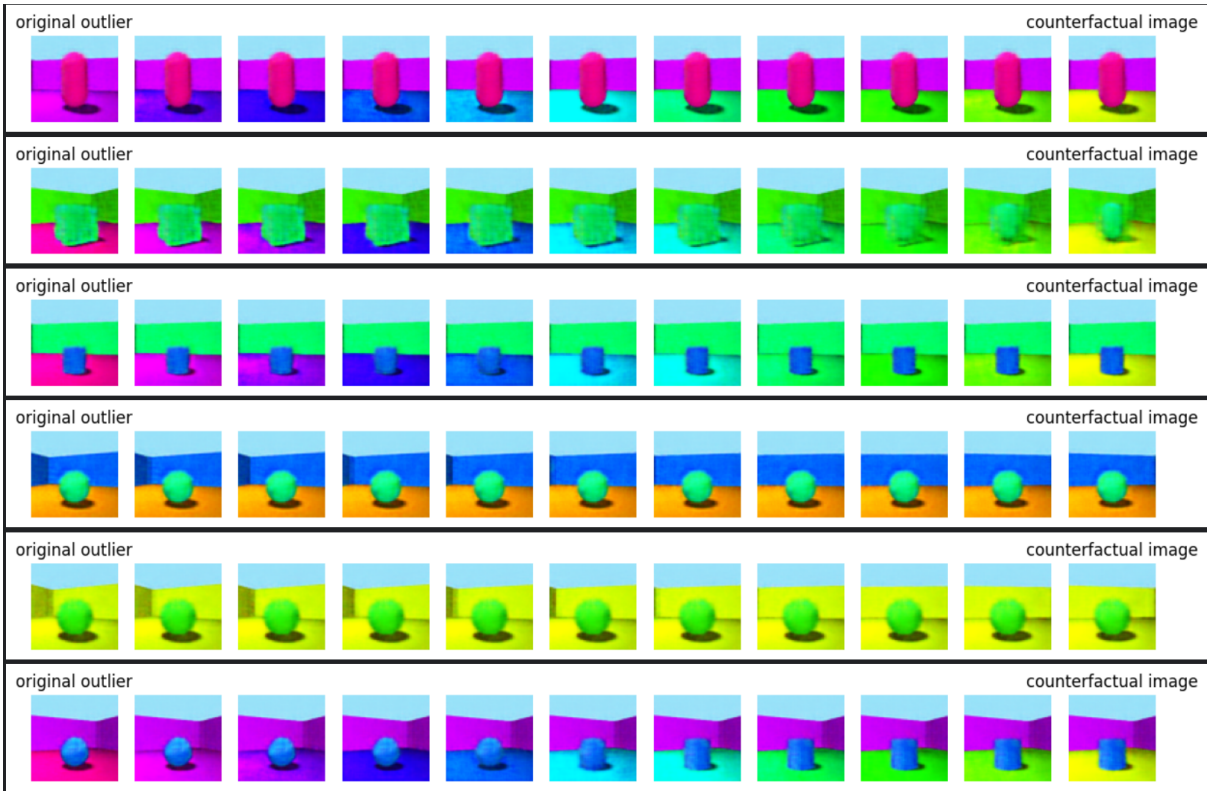


Figure 10: Morphing from outlier sample to Counterfactual for Experiment 2

Latent Plane Traversal (z_5 and z_{10})

Technique Description -

This technique builds upon the unverified claim drawn in the **Latent Traversals** section, which was that z_5 and z_{10} control the shape factor in a joint manner, that is, if I simulataneously vary

the value of z_5 and z_{10} across the plane made by their base vectors, I will be able to create different shapes smoothly

Observations that can be drawn from Figure 11 -

- The anti-diagonal shows a smooth scale transition
- The diagonal shows rather abrupt but intuitive shape transition
- In general the diagonals that go from bottom left to upper right deal with scale
- The diagonals that go from upper left to bottom right focus on shape transition

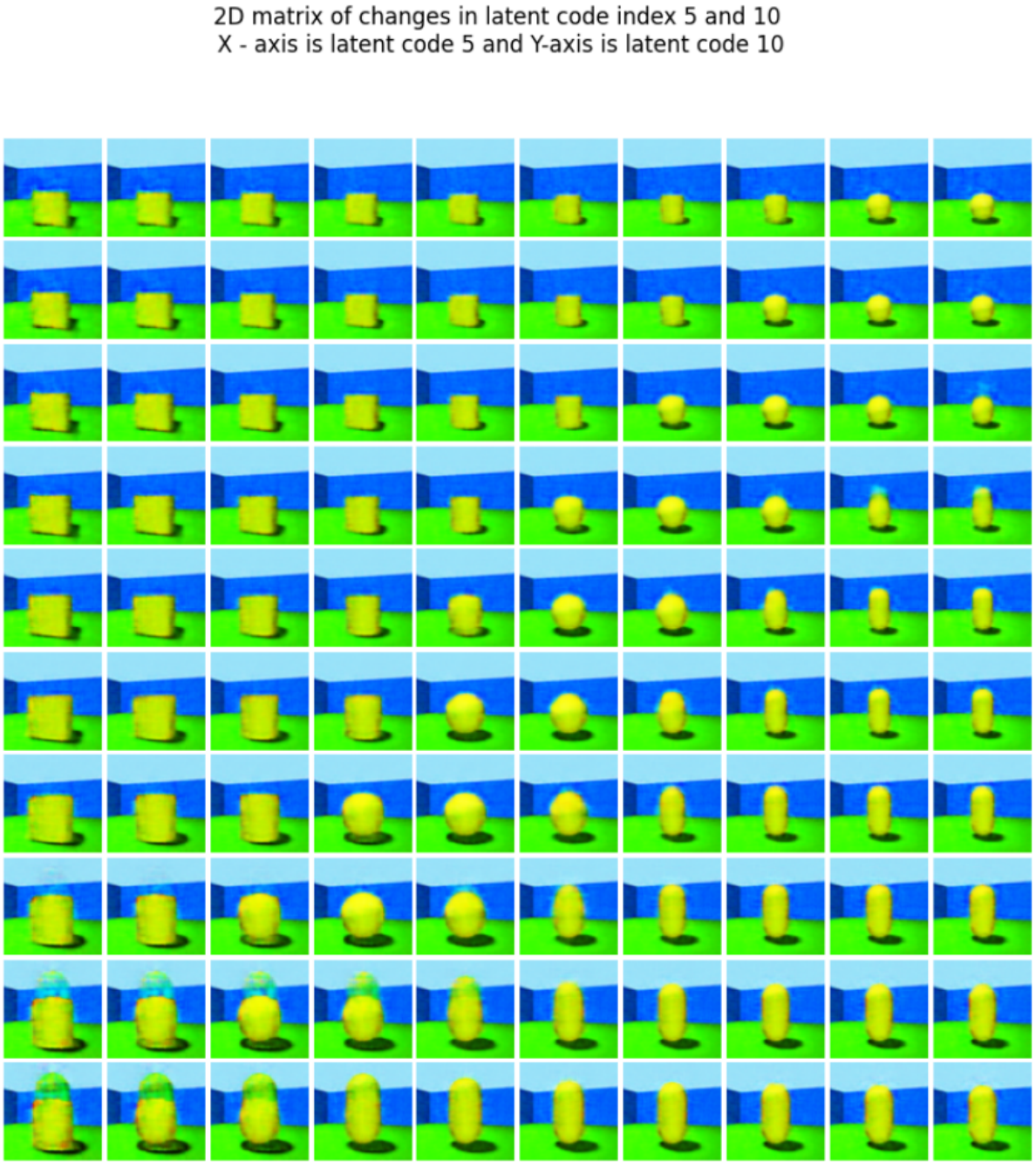


Figure 11: Variation of Shape across plane z_5-z_{10}

The above observations clearly tell that infact z_5 and z_{10} jointly control not only Shape, but also Scale

Gradient Class Activation Maps (GradCAM)

Technique Description -

GradCAM is a way of visualizing the focus of a particular convolutional layer in the model architecture with respect to a certain z_i, using it's activations and gradients We calculate a Class Activation Map for a layer using the following formulation -

$$\text{CAM} = \text{ReLU}\left(\sum_{k=1}^{K} \alpha_k A^k\right)$$

Where

$$\alpha_k = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} \frac{\partial z_i}{\partial A_{ij}^k} \text{ and } A^k \text{ is the activation of } k^{\text{th}} \text{ layer}$$

Target 1 : z_0 (Wall Hue)

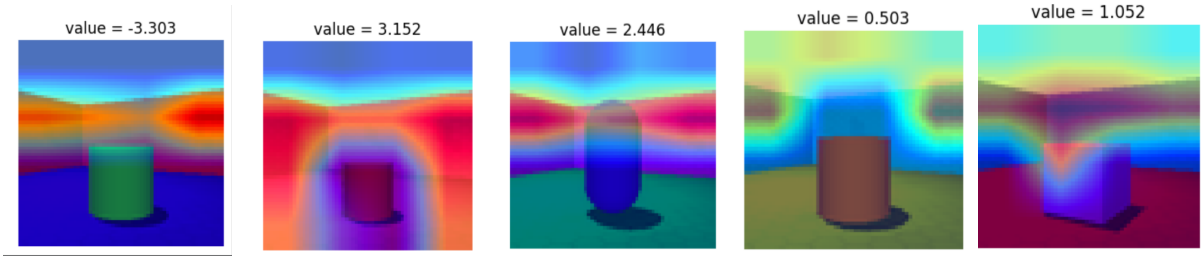


Figure 12: GradCAM visualizations for z_0

Target 2 : z_7 (Floor Hue)

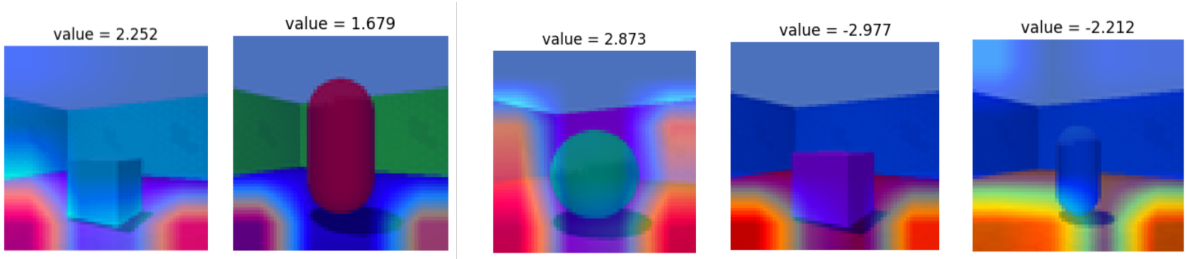


Figure 13: GradCAM visualizations for z_7

Target 3 : z_4 (Scene Orientation)
Observations -

- If value of z_4 is positive, the focus of positive activations remain on a constant region of the image for a certain orientation
- This behavior is observed till z_4 is greater than 0.25, after which the model starts getting confused about where to look for as z_4 approaches 0
- As z_4 becomes less than -0.3, the focus of negative activations remain on a constant region of the image for a certain orientation

- The negative in the first case, and the positive activations in the third case, seem confused to understand where to look at
- If we observe the focus of activations carefully, we can see that the way it decides the value of z_4 is by looking at part of sky visible, specifically the extremities of the sky

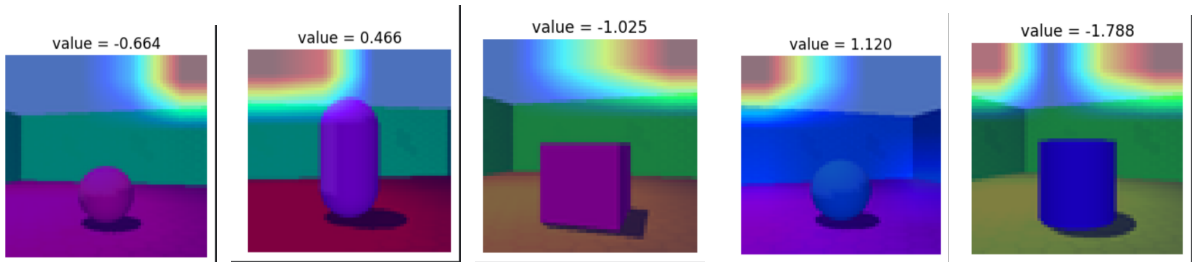


Figure 14: GradCAM visualizations for z_4 when the model is confident where to look

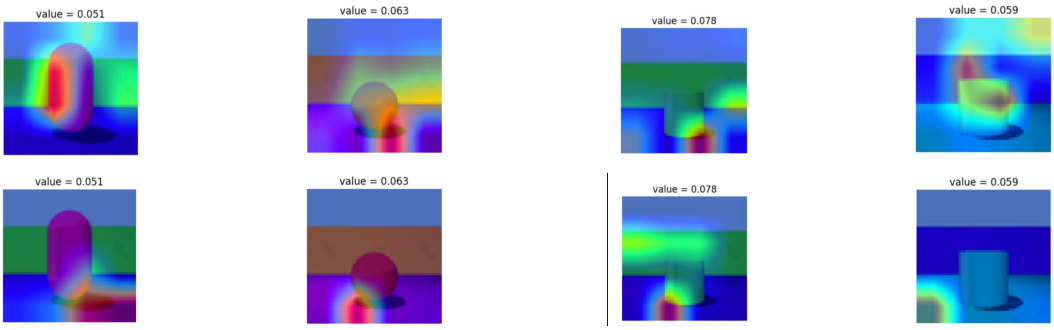


Figure 15: GradCAM visualizations for z_4 when the model is confused where to look (The above and below heatmaps show focus by different mathematical signs of activations)

Target 4 : z_8 (Object Hue)
Observations -

- This works by observing the Floor Hue and Wall Hue rather than Object directly
- Floor Hue and Wall Hue are focused on by different mathematical signs of activations
- This behavior is most prominent for when z_8 lies between -2 and 2
- This pattern fails for certain values of z_0, z_7 and z_8

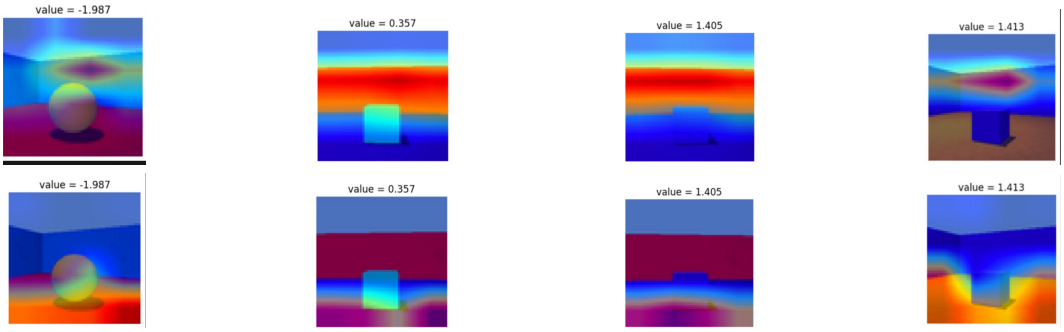


Figure 16: GradCAM visualizations for z_8 (The above and below heatmaps show focus by different mathematical signs of activations)

Target 5 : Any dead Latent Code
Observations -

- For any dead dimension, the positive and negative activations focus on different parts of the image, essentially, cancelling out the effect
- This confirms the claim in **Latent Traversals** section, that the mutual information of one component with multiple ground truths effectively cancels out

4 Final Note

This project was a deep dive into how Mathematical structure can shape a world inside itself in mysterious ways, and how we can uncover them using, well, Mathematics. This was my first all out excursion in the world of Disentanglement, Interpretation, and AI in general, I am glad that I chose the topic of something as exciting and rewarding as this.

I would like to thank both of my mentors Srijan and Manjot who helped me whenever I felt stuck and trusted me whenever I was feeling I was not doing enough

Further details of the project can be found in my Github Repository, and the project timeline is provided in the Google Doc

5 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
11. Explainable AI Methods
12. Understanding GradCAM