Diffusion Models: 'Unconditional and Conditional Approaches to 2D Image Synthesis'

MSc Artificial Intelligence & Machine Learning

Oluwatofunmi Isaac Akintaro

School of Computer Science

College of Engineering and Physical Sciences

University of Birmingham

2023-24

Abstract

This project addresses the challenge of generating high-quality and realistic 2D RGB images using the YCB dataset. The YCB dataset is a standardised collection of everyday objects with diverse shapes, sizes, and textures commonly used in robotic manipulation and 3D pose estimation. The project explores diffusion models in depth by training three different models from scratch. The three models are: an unconditioned model, a model conditioned on depth images and finally a model conditioned on class label images. These three models are trained and evaluated.

The models were trained using mean squared error (MSE) as a loss function due to its computational efficiency. The performance of the models was rigorously evaluated using both MSE and global feature-based metrics (PSNR, SSIM, LPIPS), as well as a quantitative human survey involving 20 participants. The results showed that the depth-conditioned model consistently generated the most realistic images. This model outperformed others in both quantitative metrics and the human study. The global feature-based metrics placed class-conditioned second overall, despite the MSE metric ranking it below the unconditioned model, which was rated the least effective by global metrics. The results from the human survey agree with the global feature-based metrics.

The survey also revealed gender-based differences in the image quality perception. Female participants perceived smaller differences in image quality between the models, whereas male participants seemed to sense more distinct variations.

Overall this project contributes to the field of image synthesis by providing a deep understanding of diffusion models and their applications in image generation. To the best of my knowledge, it is the first to explore 2D image diffusion on the YCB dataset. By investigating the effects of conditioning, this project provides insights into how both unconditional and conditional diffusion models influence image quality and diversity. Additionally, it introduces improved normalization techniques to enhance color accuracy for skewed distributions. By combining quantitative metrics with human survey results, the project is able to provide a well-rounded evaluation of the three models' performances in generating realistic images.

Contents

List of Figures

CHAPTER 1

Introduction

1.1 Introduction

The ability to generate visually realistic and physically plausible scenes has become increasingly important across various fields. For example in the development of autonomous vehicles, where simulations of scenes that are realistic and can incorporate real-world physics are key to providing safe autonomous vehicles. This reduces the need for extensive road testing. An example of this is Dosovitskiy et al. (2017), which introduces CARLA, an open urban driving simulator that focuses on the importance of realistic environments and physics. Similarly, medical professionals could be trained in virtual environments that closely mimic real-life operations. Seymour et al. (2002) demonstrated that virtual reality training improves operating room performance. Having the capacity to generate these scenes, will help improve the training of medical professionals and help them make less mistakes in real life settings.

Despite the progress made in image and video generation, current generative models still struggle with creating scenes that are both visually and physically accurate. This is particularly evident in complex scenarios, such as generating images of hands or creating physically plausible gymnastics routines. An example of when a video generation model fails to generate plausible physics is shown in Figure 1.1.

Figure 1.1: Video generation struggles to make realistic scenes that involve physical motion. Source: Luma AI Dream Machine (Luma AI 2024).

These limitations show the need for improved generative models that can better understand and replicate the physical properties of objects and scenes.

The main aim of this project is to improve upon existing methods of image synthesis, by using diffusion models. Diffusion models are able to sample from a Gaussian space and map to an image space. It currently offers a better alternative to traditional methods such as Autoencoders (AEs), Variational Autoencoders (VAEs), and Generative Adversarial Networks (GANs). While GANs have been the state of the art in generating realistic images, they often suffer from issues like mode collapse and instability during training (Xiao et al. 2022). Meanwhile, Diffusion models have shown the ability to overcome these limitations by generating more diverse and stable outputs.

A key differentiator of this project is the data set being used. The YCB Object dataset (Calli et al. 2015) allows the creation of scenes where multiple objects interact. You have situations where objects are stacked or obscured by others. This is challenging as little attention has been focused on generating hidden supporting objects. It will test the diffusion model's capacity to reason about the physical structure of scenes, particularly in terms of stability.

In summary, this project will focus on training and evaluating diffusion models, with an emphasis on generating visually realistic images. The project will involve comparing unconditional and conditional diffusion models. Specifically, the models are conditioned on depth information and semantic maps (class labels). By using both quantitative metrics and human surveys, this research aims to provide a full understanding of how different conditioning methods affect the quality and diversity of generated images. This will ultimately contribute valuable insights into the development of diffusion models for real-world applications.

1.2 Aims

Here are the main aims of this project:

1. **Understanding Diffusion Models**: The first goal is to develop a deep understanding of diffusion models, particularly their application in image synthesis. This project will explore the theoretical foundations and practical implementation of diffusion models.

2. **Investigating the Effect of Conditioning on Generated Outputs**: A key aspect of this project is to evaluate how conditioning affects the

quality and diversity of images generated by diffusion models. Specifically, this involves comparing unconditional diffusion models with those conditioned on depth information and semantic maps (i.e., class labels). By analysing these different conditioning methods, the project aims to determine their impact on the visual realism of the generated scenes.

3. **Quantitative Evaluation of Generated Outputs**: Finally, the project aims to rigorously evaluate the outputs generated by the diffusion models. This is done through metrics that will be used to assess the quality of the generated images. In addition, a quantitative survey will be used to gather feedback on the perceptual quality of the images. This will develop a better understanding of the strengths and limitations of diffusion models.

CHAPTER 2

Literature Review

2.1 Inspiration

The work by Basevi & Leonardis (2022) presented a novel approach to generating hidden supporting objects in complex scenes. These complex scenes were of household objects from the YCB dataset in a box. They used a volumetric conditional generative adversarial network (GAN) model and differentiable stability scores to create 3D physically stable scenes conditioned on 2D RGB, Depth and Class Label images. This GAN model was informed by a pre-trained stability scoring network. This research serves as an inspiration for this project by both providing the 2D dataset and the idea of moving from GANs to diffusion models. Diffusion models provide the opportunity to generate more diverse objects, as one of the issues GANs suffer from is the limited variety of generated output. This problem is known as mode collapse. The use of the YCB dataset in their work further underscores the importance of realistic object modelling, which this project aims to explore through a diffusion model-based approach. This exploration of 2D image synthesis provides the groundwork for 3D modeling and image synthesis in future PhD research.

2.2 Diffusion

Ho et al. (2020) introduce diffusion probabilistic models as an alternative method to generate high-quality, diverse samples by learning a reverse diffusion process. The neural network architecture which it utilises for learning is called the U-Net, which was originally introduced in biomedical image segmentation by Ronneberger et al. (2015). Since their introduction, diffusion models have emerged as a powerful alternative to GANs, particularly in the realm of image synthesis. Dhariwal & Nichol (2021) demonstrate that diffusion models can surpass GANs in generating high-quality and diverse images, both for unconditional and conditional image synthesis. As such, these qualities make diffusion models

a strong candidate for tasks that require both visual and physical plausibility, such as augmented reality and robotics. However, one of the drawbacks of diffusion models is the time needed to generate samples. This issue is exemplified in the generative learning trilemma illustrated in Figure 2.1. The idea of the trilemma is that there are trade-offs between generative models in terms of balancing sample quality, diversity, and computational efficiency (Xiao et al. 2022).

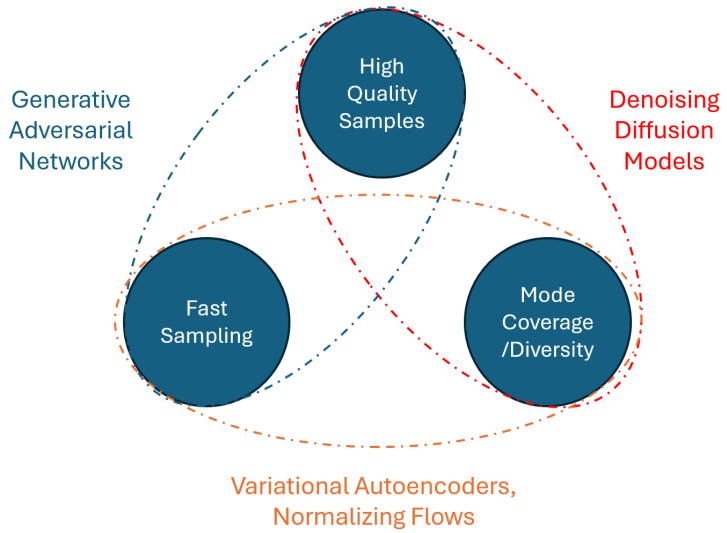


Figure 2.1: Trilemma of Generative models

To solve the issue of slow sampling Rombach et al. (2022) developed Latent Diffusion Models (LDMs), which shift the diffusion process to a compressed latent space, reducing computational costs while maintaining high-quality image synthesis. The Vector Quantized-Variational AutoEncoder (VQ-VAE) by van den Oord et al. (2018) can be used to encode the data into the latent space.

2.3 Data

The Yale-CMU-Berkeley (YCB) Object and Model Set, introduced by (Calli et al. 2015), provides a standardised collection of objects that are essential for benchmarking in robotic manipulation and related fields. This dataset's diversity—ranging from object shapes, sizes and textures—makes it particularly useful for image synthesis tasks that require models to generate physically and visually plausible scenes. Despite its primary use in 3D manipulation and pose estimation, the YCB dataset's potential application in 2D diffusion models remains largely unexplored. By incorporating the YCB dataset into diffusion model research, new benchmarks can be established. It can allow for the evaluation of the physical and visual plausibility of generated images.

2.4 Uses of YCB Data

The YCB dataset has been used as a benchmark in various applications. For example in the CosyPose framework, developed by Labbé et al. (2020), the 6D pose of multiple objects is estimated using RGB images. It had achieved state-of-the-art performance on the YCB-Video and T-LESS datasets.

Another example, Hönig et al. (2024) mentions the YCB-V dataset as part of a benchmark for 6D pose estimation. The dataset is used to evaluate dense correspondence maps, for estimating 6D object poses from RGB images using diffusion models. Finally Wang et al. (2024) explores dexterous manipulation using the YCB dataset, evaluating grasping success rates across various objects and poses, such as the Cheez-It box and a drill.

However, it is important to point out that while the YCB dataset is frequently used in 3D pose estimation tasks, it remains underutilised in 2D diffusion models and image synthesis. This under-utilisation highlights a gap in current research, as diffusion models could greatly benefit from the diversity and complex object scenes provided by the YCB dataset.

2.5 Conditioning Examples

Conditioning mechanisms in diffusion models are crucial for enhancing control over the generated outputs, as demonstrated by various recent works. Szarvas & Pogány (2024) introduce a 2D latent diffusion model for molecule generation, where conditioning is done through a Transformer-based Variational Autoencoder (TVAE) that creates spatially coherent latent representations. The use of conditioning mechanisms here allows for the fine-tuning of generated outputs.

Another example is the work of Lisanti & Giambi (2024), where diffusion models are used to overcome the weaknesses of GANs in producing diverse outputs when given the same conditioning. This paper's main contribution is the usage of multi-conditioning i.e. conditioning on multiple information, using cross-attention mechanisms. This is applied to the domain of face generation. The outcome of this paper was the generation of realistic and diverse faces, whilst having precise control over what is being generated by using multi-conditioning.

A different approach to conditioning is in the paper by Lei et al. (2023). They developed the RBD2 model in response to the problem of trying to recreate 3D scenes but with only a few images taken from different angles. Here they guide the diffusion process using those small set of RGBD images (color + depth). The model gradually generates new RGBD views as if a camera is moving through the scene. To keep the 3D structure accurate and consistent, they use a temporary 3D model (surface mesh), which is refined step by step using techniques that fill in missing parts (inpainting) and project the generated images back into 3D space (back-projection).

In the domain of developing 3D depth maps for the human body Kirch et al. (2023) developed the RGB-D-Fusion model. First, a conditional diffusion model takes a low-resolution RGB image as input and generates a low-quality depth map. In the second stage, this low-quality depth map is concatenated with the original RGB image. This creates a combined RGB-D image. This combined RGB-D image is passed through a super-resolution diffusion model that helps

enhance the quality of the depth map, producing a high-resolution depth map. By using a conditional approach their model is able to overcome issues of low-resolution data and model accuracy for human poses.

Through these examples, we are able to gain an understanding of the importance of conditioning in diffusion models which impacts image quality and the control of our generated output.

2.6 Evaluation Metrics

The choice of evaluation metrics is crucial in assessing the performance of image synthesis models. Among the most commonly used metrics is the **Mean Squared Error (MSE)**, which focuses on pixel-wise differences between the generated and target images. MSE is defined as:

$$\text{MSE} = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} \left(I(i,j) - K(i,j) \right)^2 \tag{2.1}$$

where $I(i,j)$ and $K(i,j)$ are the pixel values at position (i,j) in the generated and target images. The dimensions of the image are m and n. While MSE is computationally efficient and commonly used for training diffusion models, it captures only local pixel differences. As a result, it doesn't do a good job of capturing how well the overall image looks.

Another traditional metric, the **Peak Signal-to-Noise Ratio (PSNR)**, is derived from MSE. Conceptually PSNR uses the MSE to measure how strong the original image (signal) is compared to the errors or distortions (noise). Mathematically PSNR is essentially a logarithmic transformation of MSE. It takes into account the global image's scale by utilising the maximum pixel value of the image. PSNR is calculated as:

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{L^2}{\text{MSE}} \right) \tag{2.2}$$

where L represents the maximum pixel value of the image. For an 8-bit image, $L = 255$, while for a normalised image, $L = 1$. Although PSNR is widely used, its limitations in capturing perceptual quality are well documented. As highlighted by (Fardo et al. 2016) and (Horé & Ziou 2010), PSNR despite taking into account the global maximum pixel value, it still fails to account for structural differences and can lead to misleading evaluations of image quality.

In contrast, the **Structural Similarity Index (SSIM)**, introduced by (Wang et al. 2004), models image distortion based on three components: luminance, contrast, and structure. SSIM is defined as:

$$\text{SSIM}(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{2.3}$$

where μ_x and μ_y are the means of images x and y, σ_x^2 and σ_y^2 are the variances, and σ_{xy} is the covariance of the images. The constants C_1 and C_2 are used to stabilise the division. SSIM provides a more perceptually meaningful evaluation by considering structural information. SSIM has proven to be a reliable metric in tasks that require visual and structural consistency. Additionally, (Rouse & Hemami 2008) suggest that simplified versions of SSIM, can reduce

computational complexity while keeping the evaluation of image quality similar to human perception.

However, recent advancements in deep feature-based perceptual metrics, as discussed by (Zhang et al. 2018), expose the shortcomings of metrics like PSNR and SSIM. The paper shows that these traditional metrics struggle with geometric distortions such as image blurring and image warping. Deep feature-based metrics, such as the **Learned Perceptual Image Patch Similarity (LPIPS)**, better match how humans see images by focusing on high-level features extracted from neural networks. These metrics have been shown to outperform traditional ones in tasks like super-resolution, deblurring, and colourisation. As seen in Figure 2.2, perceptual metrics like LPIPS give a more accurate evaluation of image quality.

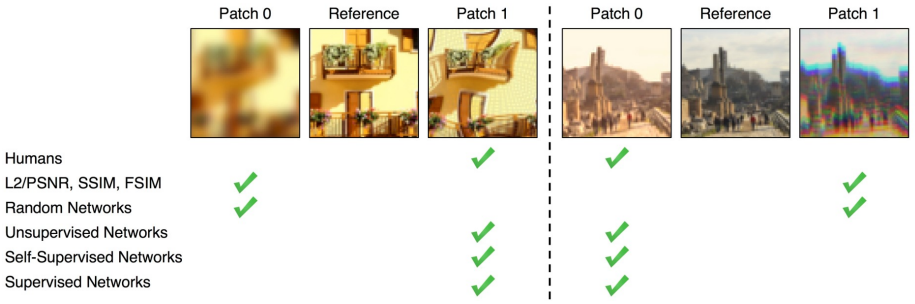


Figure 2.2: Perceptual Loss aligns more closely with human perception. Source: (Zhang et al. 2018).

LPIPS uses a perceptual loss function, which compares the feature maps of two images (clean and generated) at various layers of a pre-trained neural network. For this project, AlexNet was used as the pre-trained network, as it is known to provide a more accurate representation of human visual perception. This comparison is made using the following equation from Zhang et al. (2018):

$$d(x, x') = \sum_{l} \frac{1}{H_l W_l} \sum_{h,w} \|w_l \cdot (\hat{y}_{l,hw} - \hat{y}'_{l,hw})\|_2^2 \tag{2.4}$$

In this equation:

- $\hat{y}_{l,hw}$ and $\hat{y}'_{l,hw}$ are the feature maps at layer l for the original clean image x and the generated image x', respectively.
- H_l and W_l are the height and width of the feature map at layer l, which normalize the difference.
- w_l are learned weights for each layer, which adjust the importance of each layer's contribution to the perceptual loss.

The formula computes the squared difference (similar to Mean Squared Error) between the feature maps at each layer, then sums these differences across all layers.

2.7 Research Gap

While much research has compared the performance of unconditional and conditional diffusion models, especially for tasks involving attributes or class labels (Ho et al. 2020), there is still a lack of detailed studies on conditioning techniques in more complex situations. Specifically, there is a gap in exploring datasets where hidden objects support one another, such as in the YCB dataset (Calli et al. 2015).

The YCB dataset, which is commonly utilised for 3D pose estimation tasks, has yet to be fully leveraged for use in 2D image synthesis models. Its complex object scenes, where multiple objects interact with each other, provide a unique opportunity to test the capabilities of diffusion models. This project aims to address this gap by systematically training and evaluating three diffusion models from scratch: an unconditional model, a model conditioned on depth images (representing the distance from the camera to the object), and a model conditioned on class-label semantic maps.

Additionally, this project will evaluate the differences between local metrics, such as MSE, and more global metrics, including PSNR, SSIM, and LPIPS. A human study involving participants from diverse genders and ethnicities ranking generated images from the three models will provide further insights into performance and offer a meaningful contribution to the field of image synthesis in complex object scenes.

CHAPTER 3

Background

3.1 Introduction

In chemistry, when a liquid is introduced into water, it gradually disperses until the entire container is uniformly mixed as shown in Figure 3.1. Similarly, in diffusion models, an image is continuously corrupted by adding Gaussian noise at each timestep until it becomes indistinguishable from pure noise. The goal of the diffusion model or neural network is to learn how to reverse this process. The diffusion model is capable of transforming the noisy image back into the original clean image.

Figure 3.1: Process of diffusion in chemistry. Image source: Science Photo Library Ltd / Getty Images, retrieved from ThoughtCo.

3.2 Gaussian Noise in Diffusion Models

Gaussian noise is a fundamental component of the diffusion process. It is an additive type of noise, meaning that it is added directly to the pixel values of an image. Each pixel in the image is treated independently, with the noise to be added being drawn from a Gaussian distribution. The decision to use Gaussian

noise is deliberate. Gaussian noise is convenient because of its mathematical properties. It uses a normal distribution which has a smooth curve which makes it easy to add or remove noise at each step when training. In addition, the normal distribution is symmetric around its mean, which simplifies calculations, making it easier to understand how noise affects data.

The Gaussian normal distribution is defined by the equation:

$$N(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

where μ is the mean and σ^2 is the variance. In diffusion models, at each time step, a small amount of noise is added to every pixel in the image. This noise is drawn from a Gaussian distribution with a mean of zero, meaning the noise is centred around zero and doesn't push the pixel values in any particular direction. The variance of the noise can change depending on the timestep, which controls how much randomness is introduced at each step. Essentially, in the early timesteps, the noise has low variance, as only gentle changes are made to the image. Whilst in the later timesteps, the noise has a higher variance, adding more randomness and making the image much noisier.

3.3 The Diffusion Process

The full diffusion process is described below in Figure 3.2.

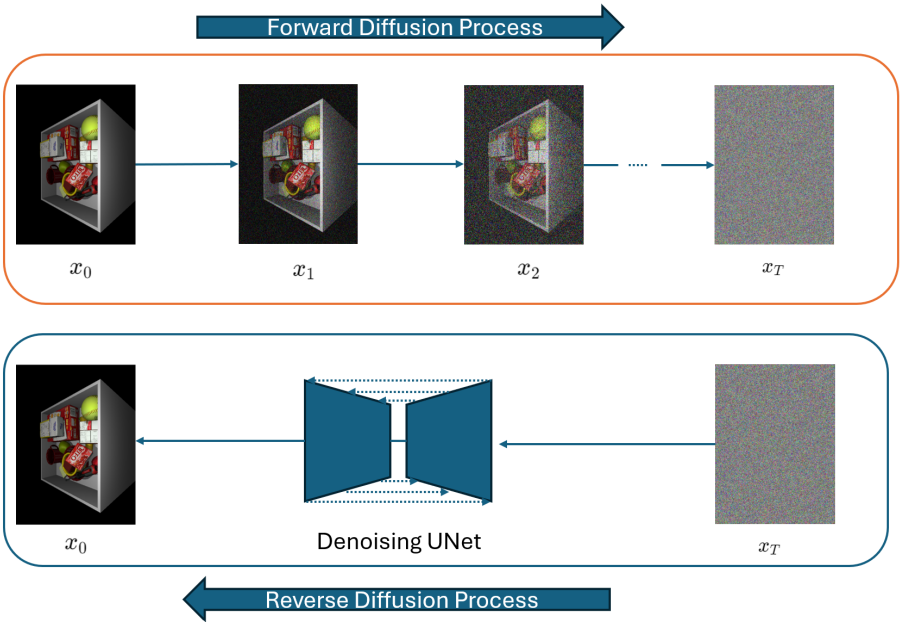


Figure 3.2: Visual diagram outlining the forward and reverse diffusion process.

The forward diffusion process involves gradually adding Gaussian noise to an image over a series of timesteps until it becomes purely noise. This process serves as a way to turn the complex image distribution into a Gaussian distribution, which is easier to handle computationally. The reverse process, or denoising

process, involves learning how to go from this noisy Gaussian distribution back to the clean image distribution.

The timesteps play a crucial role in this process. The model reverses the noise-adding process by carefully removing noise in small steps, bringing the noisy image back to its original form. This incremental approach breaks down the complex mapping into smaller, more manageable steps. This helps the model to better capture the finer details of the image distribution.

In the diffusion process, the forward process follows a Markov chain, where each noisy image x_t depends on the previous image x_{t-1}. Mathematically, this is expressed as:

$$q(x_t|x_{t-1}) = N(x_t; \sqrt{\alpha_t}x_{t-1}, \beta_t I) \tag{3.1}$$

where N denotes a normal (Gaussian) distribution, $\alpha_t = 1 - \beta_t$, and β_t controls the amount of noise added at each timestep. Larger values of β_t introduce more noise, while smaller values add less noise.

Getting the noisy image can be expressed in the equation below:

$$x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1 - \bar{\alpha}_t}\, \epsilon \tag{3.2}$$

Where:

- x_t: Represents the state of the variable x at time t. This represents the noisy version of the original data.
- x_0: Represents the initial state of the variable x, which is the original data at time $t = 0$.
- α_t represents the fraction of the original data that remains after the noise process. This is $1 - \beta_t$.
- $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$ represents the product of α_s from $s = 1$ to t, the timesteps.
- $\sqrt{\bar{\alpha}_t}$ and $\sqrt{1 - \bar{\alpha}_t}$: These square root terms scale the contributions of the original data and the noise term, respectively. The balance between these terms determines how much of the original data remains in the noisy version at time t.
- ϵ: This is the actual underlying noise added to x_0. This noise is drawn from a standard Normal distribution.

In the reverse process, the goal is to learn the posterior distribution $p_\theta(x_{t-1}|x_t)$ as show in Figure 3.3.

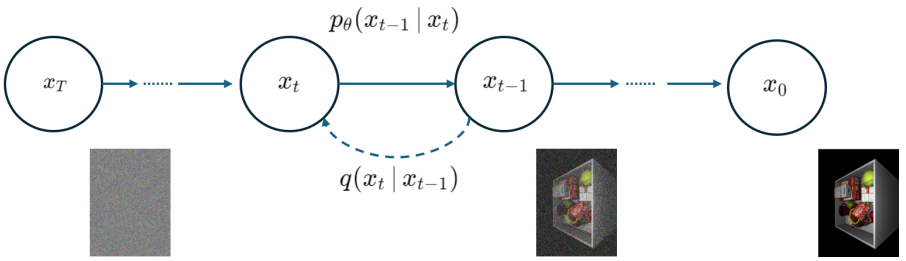


Figure 3.3: The reverse process where our Unet model parameters are trained to predict the noise added, so we can denoise our image.

Learning the posterior distribution allows the model to predict the noise added. This allows the denoised image at the previous timestep to be calculated given the noisy image at the current timestep. This is shown in the equation below:

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \hat{\epsilon}_\theta(x_t, t) \right) + \sigma_t z \tag{3.3}$$

- $\hat{\epsilon}_\theta(x_t, t)$ is the noise predicted by the U-Net at timestep t.
- $\sigma_t z$ is an optional additional noise term that is useful to add in the generation process to ensure diverse generated samples.

3.4 Learning an Image Distribution

Direct Learning: One approach is to learn the distribution of images directly. This involves constructing a model that understands the complex patterns and features in the data to generate realistic images. However, this can be computationally expensive because image data typically exists in a high-dimensional space, and capturing the full complexity of this distribution requires significant computational resources.

3.5 Simplifying the Problem

Mapping to a Gaussian Normal Distribution: Instead of directly learning the image distribution, a more efficient approach involves mapping the image distribution to a Gaussian normal distribution as shown in Figure 3.4. Gaussian distributions are easier to work with due to their well-understood properties and simplicity. By learning these series of mappings, the complex image distribution is effectively transformed into a simpler and more manageable form.

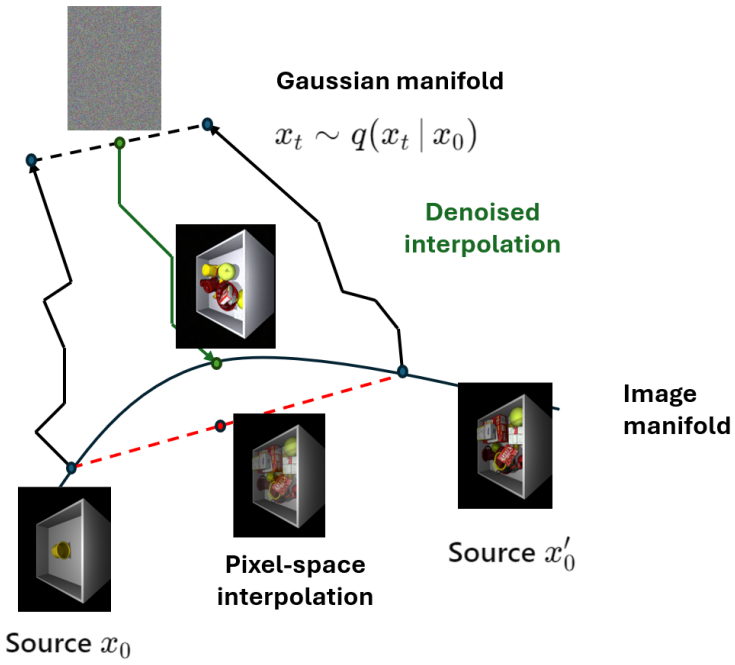


Figure 3.4: Pixel Space to Gaussian Space Representation.

3.6 Generating Images

Reverse Mapping: Once the mapping from the image distribution to the Gaussian distribution is learned, generating new images involves sampling from the Gaussian distribution and then using the reverse mapping to convert these samples back into the image space. This approach reduces computational complexity because working with Gaussian distributions is more straightforward than directly manipulating the high-dimensional image distribution.

3.7 U-Net Architecture in Diffusion Models

Figure 3.5 shows the U-Net architecture, which is key to the operation of diffusion models. It was originally developed for biomedical image segmentation. The U-Net is a neural network designed to take an input tensor of a particular size (the noisy image) and output a tensor of the same size (prediction of the noise that was added). The U-Net consists of two main parts: an encoder and a decoder.

- **Encoder:** The encoder reduces the size of the image while increasing the number of channels, which represent the features. This allows the model to capture more high-level conceptual information from the noisy image.
- **Decoder:** The decoder takes this high-level understanding and attempts to reconstruct the spatial dimensions of the noise added to the original

image. **Skip connections** shown as grey horizontal arrows between corresponding layers in the encoder and decoder help preserve spatial features that are key for reconstructing the image.

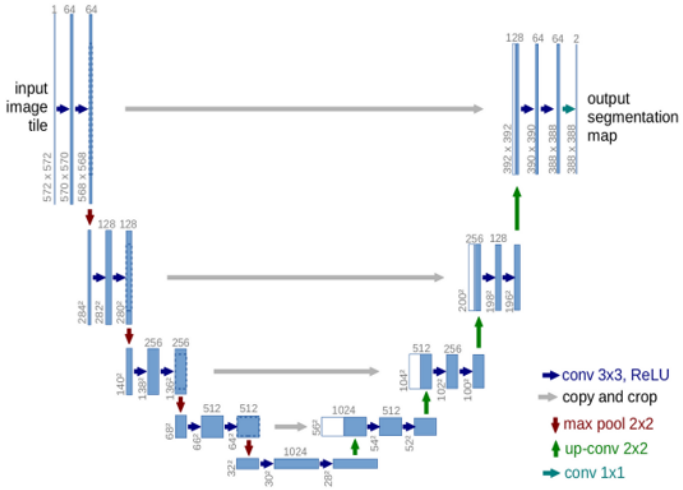


Figure 3.5: Original UNet Architecture. Source: Ronneberger et al. (2015)

In the context of diffusion models, the U-Net is also provided with timestep information, which it embeds using sinusoidal positional embeddings. This information can be introduced at the start of the model or at each stage of the U-Net, allowing the model to condition its predictions on the specific timestep. This is illustrated in Figure 3.6.

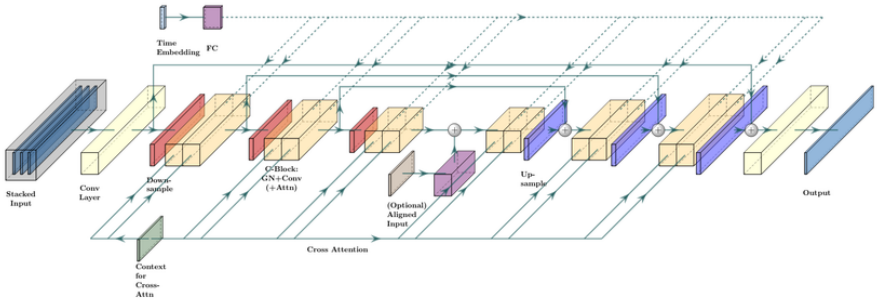


Figure 3.6: UNet Adapted for Diffusion. Source: Huang et al. (2023).

3.8 Loss Function

The loss (error) function is critical in the training of diffusion models, as it guides the model to accurately predict the noise added at each timestep. Specifically, the loss is calculated by comparing the predicted noise with the actual noise introduced during the diffusion process. A commonly used approach is MSE,

which measures the difference between these two noise values and serves as a computationally efficient metric for training.

3.9 Training Process

The training process begins by taking a random sample image x_0 which exists in the distribution of possible images $q(x_0)$. A noise level t is also selected uniformly between 1 and T.

The amount of Gaussian noise added to the image corresponds to the chosen noise level t. The higher the time steps, the more noise will be introduced. While lower timesteps add less noise. At each training step, the neural network is trained to predict the noise that was added at a randomly selected timestep t. It essentially learns how to reverse the diffusion process for any given timestep. The model is trained using mini-batch stochastic gradient descent. In this method, instead of using the entire dataset at once to learn, the model is updated using small groups of data called mini-batches. After processing each mini-batch, the model adjusts its parameters slightly to reduce its loss. This process is repeated over many batches, allowing the model to gradually improve its accuracy without overwhelming the system with too much data at once.

3.10 Summary

In summary, diffusion models rely on a process of gradually corrupting an image with Gaussian noise and then learning how to reverse this process to generate realistic images. The U-Net architecture plays a crucial role in predicting the noise added at each timestep, allowing the model to reconstruct the original image. The use of Gaussian noise, the Markov chain framework, and the training process all contribute to the ability of diffusion models to generate diverse and high-quality images.

CHAPTER 4

Methodology

4.1 Data

This project utilises three key types of 2D data: RGB images, Depth images, and Class label images as shown in Figure 4.1. RGB images show colour information and Depth images show how far objects are relative to a camera. Whilst Class label images show what kind of object category each pixel in the image belongs to. The dataset consists of 14,000 images in total, representing various complex scenes with multiple objects interacting inside a box. Inside the box, we can have up to 14 unique objects. Each object configuration, ranging from one object in the box to 14 objects, is represented by 1,000 scenes.

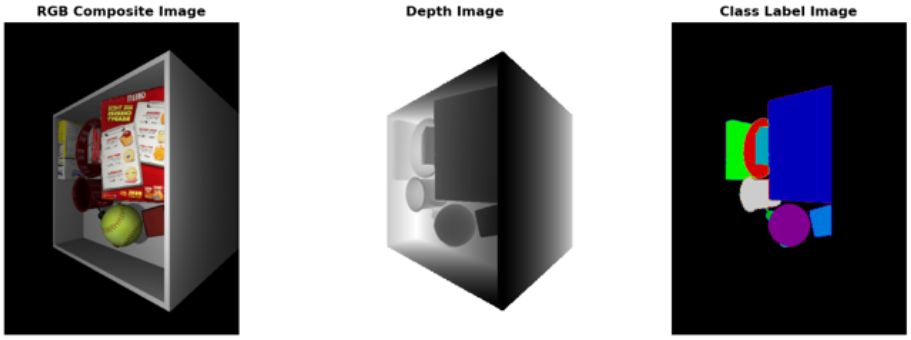


Figure 4.1: 2D Image Data including RGB, Depth and Semantic Class Information.

The RGB images have dimensions of (C, H, W), where $C = 3$ channels (representing RGB), $H = 640$ pixels in height, and $W = 480$ pixels in width. The pixel values range between 0 (black) and 1 (white). The Depth images also have dimensions of 640×480 pixels, with pixel values ranging from 0.28 to infinity. The pixel values represent the distance from the camera to objects in

the scene. Infinity represents air. Lastly, the Class label images, with dimensions 640×480, contain pixel values ranging from 0 to 14, where 0 corresponds to the table and 1-14 represents any of the possible objects inside the box.

The combination of these three types of data provides an understanding of each 2D scene. The RGB image provides the visual information, the Depth image offers a 3D understanding by showing the distance from the camera, and the Class label image shows the object category that each pixel in the image belongs to.

4.2 Data Preparation

To prepare the data for model training, the following steps are followed:

- Extract the data from the .mat file as a NumPy array using the h5py library.
- Convert the data from NumPy arrays to PyTorch tensors.
- Resize the data from 640×480 to 256×192, scaling down by a factor of 2.5 to reduce computational requirements and fit the model on the GPU.
- Normalize the data due to the sensitivity of the model to skewed distributions caused by predominantly black images.
- Split the data into 80% for training and 20% for validation.
  - During training, the model's performance is monitored using the validation set to ensure generalisation and prevent overfitting.
  - Overfitting is the process where the diffusion model learns the training data too well and performs poorly in predicting the noise on new unseen data.
- Organise the data into batches to ensure stable model updates during training.

4.3 Training Process

An overview of the training process is given in Figure 4.2:

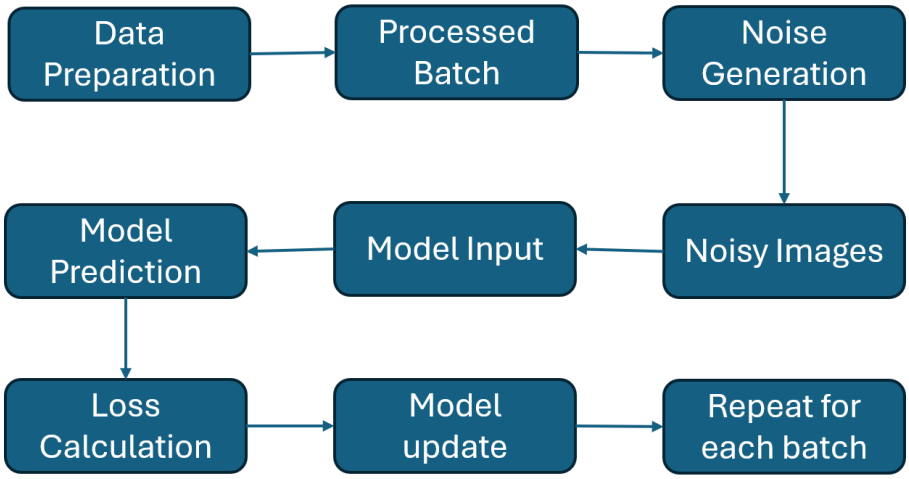


Figure 4.2: Training Process Illustration

The training process involves several key steps:

- **Batch Processing:** The data is fed into the model in batches for stable updates and to reduce computational costs.
- **Forward Diffusion Process:** For each image in a batch, a random timestep between 1 and 2000 is selected as shown in Figure 4.3. The corresponding noise is added to the image, mapping the clean image to a noisy version based on the timestep.

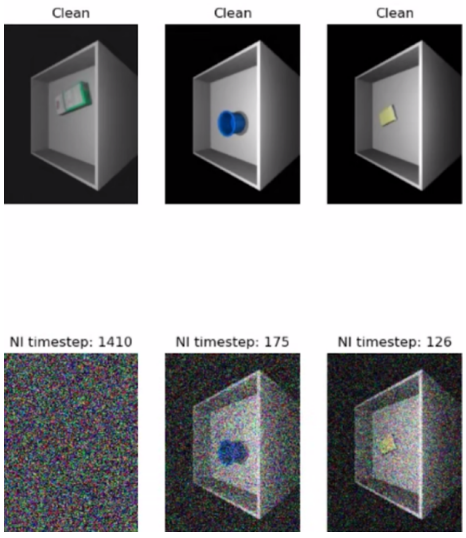


Figure 4.3: Random Timestep Analysis

- **Denoising Objective:** The model is trained to predict the noise added at a particular timestep and then subtract it from the noisy image to recover

the original image. The MSE loss is computed between the predicted noise and the actual noise.

- **Backpropagation:** The computed loss is used to update the model's weights through backpropagation. This process is repeated for each batch in training.

4.4 Conditioning in the Training Process

When conditioning the model (e.g., conditioning on depth data), the input image now has four channels instead of three. The first three channels (RGB) have noise added, while the fourth channel remains unchanged. This fourth channel is concatenated with the noisy RGB image, resulting in a four-channel input to the model. The model's output still remains as three channels, which predicts the added noise for the RGB.

4.5 Generation Process

After the UNet in the diffusion model is trained to learn the mappings between the image distribution and the Gaussian distribution. We can now generate images starting by sampling from the Gaussian distribution and mapping back through the reverse diffusion process to a clean image. An overview of the process of generating is given in Figure 4.4.

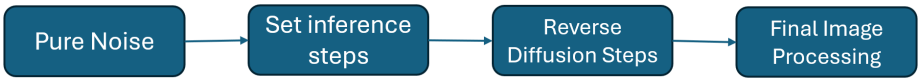


Figure 4.4: An overview of the process of generating images from Gaussian noise.

The generation process involves creating images from pure noise as shown in Figure 4.5. The process begins with a random tensor sampled from a Gaussian distribution. The majority of the values are between -4 and 4.

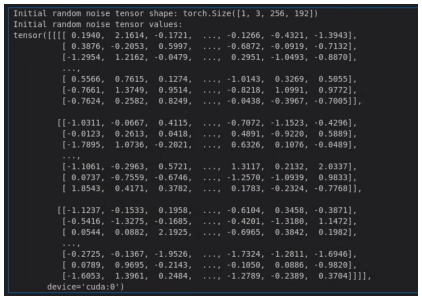
Through a series of denoising steps, the model gradually refines the noisy tensor into a meaningful image. The number of inference steps is set to 1000, balancing accuracy and speed. Each instantiation of the starting Gaussian noise results in a unique clean output image, showing that the process is random.

The generation process is like sculpting, where each step of removing noise shapes the random starting point into a final image.

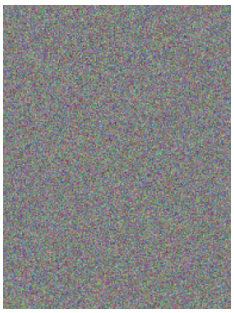
4.6 Model Architecture

The model architecture is based on the U-Net, specifically utilising the HuggingFace `UNet2DModel` architecture. The key components of the model include:

- **2D Down Blocks:** This is a sequence of convolutional layers followed by downsampling operations to capture key features at different resolutions.



(a) Initiated random Gaussian noise values.



(b) Visualisation of initiated random Gaussian noise values.

Figure 4.5: Intialisation of pure Gaussian noise

- **2D Attention Down Blocks:** Similar to the 2D Down Blocks, but with added attention mechanisms to focus on important regions in the image.
- **2D Up Blocks:** These blocks perform upsampling to refine and reconstruct the noise tensor, adding details to make the image clearer and more precise.
- **2D Attention Up Blocks:** These blocks combine upsampling with attention mechanisms to improve the reconstruction of important image regions.

4.6.1 UNet2D Model Architecture Configurations

The `UNet2DModel` architecture is configured with the following specifications:

- **Sample Size:** The size of a typical input sample is 256x192 after transforming the raw data.
- **Input Channels:** The number of channels in the input sample. In the unconditioned case this is just three channels for RGB. Whilst in the conditioned case this is four channels. This is because the extra conditioning information is concatenated at the end of the three channels.
- **Output Channels:** The number of channels in the output sample. These are three channels in both the unconditioned and conditioned cases as we are trying to predict the noise added to the RGB channels.
- **Layers per Block:** Each down block (encoder) and up block (decoder) has two ResNet layers, as this offers enough learning capacity whilst reducing GPU usage.
- **Encoder and Decoder:** The encoder is composed of seven down blocks, while the decoder contains seven up blocks. Each block type has specific configurations based on the model architecture. For example, some of the blocks include attention mechanisms to help the model focus on important features during encoding and decoding.

4.6.2 ResNet Blocks

The convolutional blocks follow a standard architecture:

- **Typical Structure:** Convolutional layer (spatial processing) → normalization layer (stability) → activation function (non-linearity).
- **Group Normalization:** Used for normalizing features within groups of channels. More information on this is in Section 4.7.
- **Downsampling:** Achieved through convolutions with a stride of 2. This reduces the spatial dimensions of the image by half at each downsampling operation. This may lead to some information loss, but it is more efficient than pooling operations.
- **Upsampling:** Achieved using nearest neighbour interpolation to enlarge the size of the image data, followed by convolution layers to smooth out and improve the details of the enlarged images. The nearest neighbour interpolation is used in this project instead of the convolutional transpose to avoid checkerboard artefacts in the generated images.
- **Activation Function:** The SiLU (Swish) function combines properties of ReLU and Sigmoid for smoother gradients to improve the model's learning. More information is given in Section 4.8.
- **Skip/Residual Connections:** As the model gets deeper with more layers, it can start to forget important information. These connections help the model retain information that might otherwise get lost. They also help prevent another issue called the vanishing gradient problem. In deep networks, the gradient (used to update the model's weights) becomes smaller as it passes through many layers, which can cause the model to stop learning effectively. Residual connections allow gradients to flow more easily through the network, ensuring that the weights get updated and learning continues. Refer to the U-Net architecture diagram in Figure 3.5 for a visual representation of skip connections.

4.6.3 Attention Blocks

Attention blocks in the model are structured as follows:

- **Structure:** Each attention block has an attention layer followed by a ResNet block.
- **Functionality:** The attention layer lets each pixel look at the entire image (self-attention), which helps the model understand the image as a whole and reduces mistakes in the output.
- **Mechanism:** The image is flattened, and calculations are done to create query, key, and value vectors. These vectors help the model decide which parts of the image should pay attention to other parts. This allows the model to better capture important details and improve overall understanding of the image.

4.7 Normalization Techniques

This section discusses normalization applied to values within our model's neural network, while Section 5.6.10 refers to normalization applied to the input data. Normalization is crucial for stabilising training and preventing issues like vanishing gradients. Group Normalization is used as it works with small batch sizes. Nonetheless here is an overview of some of the normalization techniques:

- **Batch Normalization:** Normalizes across batches, suitable for larger batch sizes.
- **Layer Normalization:** Normalizes across features (or channels) within each individual sample.
- **Group Normalization:** Normalizes groups of channels, balancing between batch and layer normalization.

The goal of normalization is to ensure that the output (activations) at each stage in the UNet has a mean of 0 and a standard deviation of 1. This helps prevent the vanishing gradient problem and leads to more stable and efficient training.

4.8 SiLU/Swish Activation Function

An activation function transforms the output of a neuron, helping the model learn complex patterns by introducing non-linearity. The SiLU (Swish) activation function is utilised in the model due to its advantageous properties:

- **Equation:** $x \times \text{sigmoid}(x)$, where the sigmoid function is defined as $\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$, and it squashes the input between 0 and 1.
- **Properties:** It allows for negative values, preventing vanishing gradients. It also suppresses large negative values as the sigmoid function outputs a value close to 0.
- **Performance:** SiLU combines the benefits of both ReLU (efficient training) and Sigmoid (smooth activations).

4.9 Positional Embedding

Positional embedding is used to provide the model with timestep information during the diffusion process:

- **Purpose:** The embedding encodes the current timestep.
- **Implementation:** Positional embeddings can be added as a separate channel or combined with existing channels in the input data.

4.10 Implementation

Software and Tools

The project is implemented in Python 3.8, using various libraries and frameworks, including:

- **Hugging Face Libraries:** Such as diffusers, transformers, and huggingface-hub for model implementation and training.
- **Data Science Libraries:** Such as numpy, pandas, torch, torchvision and h5py for data processing and manipulation.
- **Visualisation:** matplotlib and seaborn for visualising results.

Testing

- **Hardware Environment:** Smaller experiments were initially conducted on a laptop with an 8GB GPU. Intermediate experiments were conducted on a Linux local workstation running Ubuntu 24.04 LTS. It had an NVIDIA GeForce RTX 4090 GPU with 24GB of memory, which helped speed up the process and gave capabilities to train a bigger model. Finally, the largest experiments were done on BlueBear the University's supercomputer with an NVIDIA A100 40GB GPU.
- **Software Environment:** Includes the operating system, programming language, and libraries referenced earlier. The project environment was managed using conda, with dependencies specified in a requirements.txt file for reproducibility.
- **Evaluation Metrics:** The generated outputs are evaluated both through metrics and perceptual feedback from human surveys. The survey administered is available in Appendix 8.3.

CHAPTER 5

Results and Discussion

5.1 Results

5.1.1 Overview

Figure 5.1 shows the generated image results comparing the three diffusion models: unconditioned, conditioned on depth and conditioned on class semantic map.

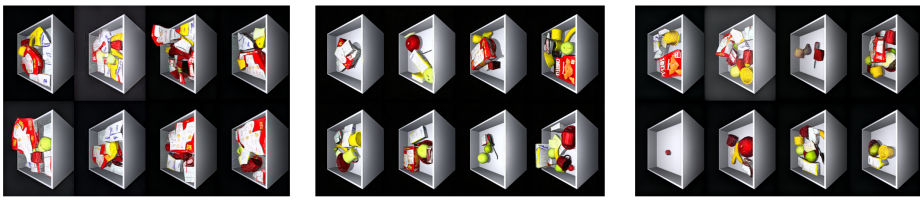


Figure 5.1: Unconditioned, conditioned on depth and conditioned on class semantic map images.

The performance of the models is measured using several metrics. During training, the validation loss (calculated using Mean Squared Error, or MSE) is a key indicator that helps monitor whether the models are overfitting or underfitting. Essentially, this loss measures how closely the model's predicted noise matches the actual noise added during the diffusion process.

While MSE is computationally efficient and useful for tracking training progress, it provides a limited, pixel-level view of model performance. To evaluate the models more holistically, additional global metrics such as Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM), and Learned Perceptual Image Patch Similarity (LPIPS) are tracked on the validation data throughout training. The results for these metrics are discussed in Section 5.2.

Figure 5.2 presents a comparison of the validation loss versus epochs for all three models. The range of validation loss/error values is between 0 and 1, with lower values indicating better performance. Figure 5.2 shows that conditioning on depth has the smallest validation MSE loss, whilst unconditioned and conditioned on class labels are similar with the unconditioned case having a lower loss for most of the epochs.

As depicted in Figure 5.2, during the initial epochs (0-10), all three models exhibit a steep decline in validation loss, reflecting rapid learning. This trend suggests that the models are effectively learning during the early stages of training and continue to improve, albeit at a slower rate, as training progresses.

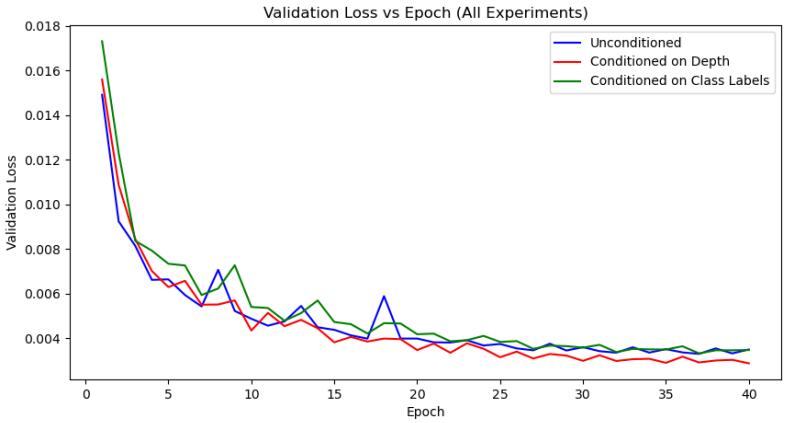


Figure 5.2: Comparison of the performance of the three models across validation sets. The lower the loss/error the better the performance.

The model conditioned on depth (red line) consistently achieves the lowest validation loss across all epochs. It has the least fluctuations compared to the two other models especially in the middle epochs (10-25). Its performance suggests that depth information plays a crucial role in enhancing the model's ability to generalize. By incorporating depth as a conditioning factor, the model can better understand and represent the data, leading to improved performance.

Compared to the class label-conditioned model, the validation loss of the unconditioned model (the blue line) is somewhat lower in general throughout most of the training, with similar trends for both models. That could indicate that, without any extra conditioning, the unconditioned model captures the underlying structure of the data quite well.

The green line is the class label-conditioned model, which behaves quite similarly to the unconditioned model but with a slightly higher validation loss during most of the epochs. This indicates that while class labels provide useful categorical information, they do not improve pixel-wise accuracy as much as depth information does. The marginal difference between the class-conditioned and unconditioned models suggests that class labels alone are not as effective for this task as depth conditioning, which provides more spatially informative cues for the model.

After approximately 30 epochs, the validation losses for all models stabilise, with minor fluctuations. This suggests that the models have reached a point of diminishing returns in their training, where further training yields little to no improvement.

In conclusion, the results of MSE validation loss indicate that conditioning on depth leads to the best pixel-wise accuracy. This is evidenced by the lowest validation loss throughout training. The unconditioned and class label-conditioned models exhibit similar MSE values, with the unconditioned model generally performing slightly better. This indicates that while class labels provide some useful guidance, they may not be as effective as depth information in reducing pixel-wise error.

5.1.2 Unconditional Generation

The images generated from pure Gaussian noise in the unconditional case show promising results as seen in Figure 5.3. Visually, we observe that the model can recreate the box structure quite well. Smaller objects like apples and balls are well-represented. However, larger items, such as the Cheez-It box or cereal boxes, appear distorted. The idea of physical stability or supporting objects is difficult to understand here because it's hard to distinguish the unique objects that provide support.

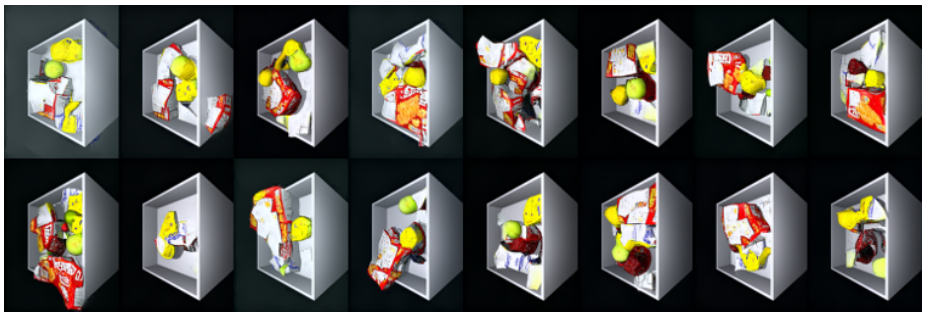


Figure 5.3: Unconditional Generated Images.

The training and validation loss curves follow a cosine annealing pattern described in Figure 5.4 and shown in Figure 5.5. This pattern rapidly increases the learning rate at specified intervals to help the model escape local minima on the loss surface and aim for the global minimum.

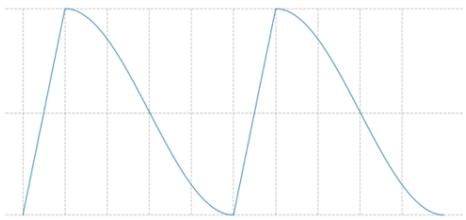


Figure 5.4: Cosine Annealing Pattern (Katsura & Baldassarre 2023).

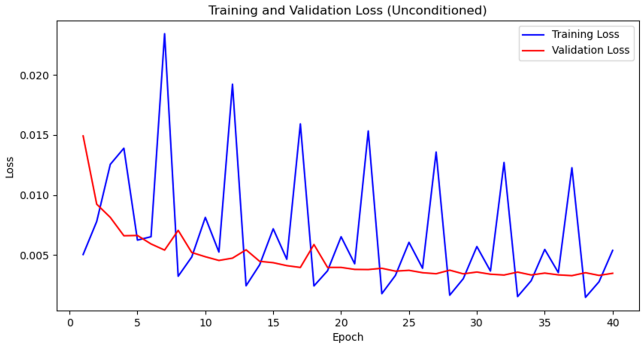


Figure 5.5: Unconditional Training and Validation Losses During Training.

The loss curves indicate that:

- The model's performance starts to plateau between 30 to 40 epochs.
- There is no sign of overfitting, as both training and validation loss curves decrease overall.
- The model does not underfit, as indicated by the continuous decrease in loss.

The loss diagram is derived from the Mean Squared Error (MSE), described using Equation 2.1.

5.1.3 Conditioned on Labels

In the class label-conditioned case:

- The model handles the shapes of larger items relatively well as shown in Figure 5.6 even better compared to the unconditioned model. The physical stability of the objects here looks feasible.
- The individual objects are also clearer than the unconditioned model, which raises some questions about the reliability of MSE validation as an evaluation metric for comparing models.
- However, some generated images still exhibit color issues similar to the unconditional case.

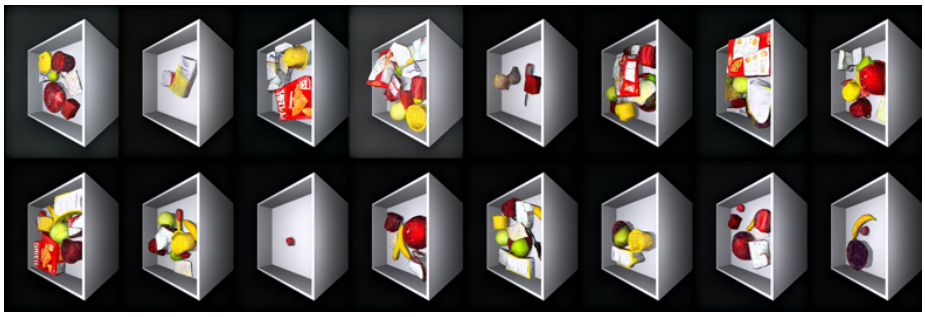


Figure 5.6: Generated images our model is able to create when aided by class label images as conditioning in the training process.

The training and validation loss is described in Figure 5.7.

Figure 5.7: Training and Validation Loss for the model that used class label images as conditioning.

5.1.4 Conditioned on Depth

Figure 5.8 shows that when conditioned on depth data, the model exhibits better performance:

- The box structure is formed accurately, and there is no noticeable background color issue.
- Larger items like the Cheez-It box and cereal boxes are generated with the correct shapes.
- The images incorporate physical stability here very well; for example, the first image and second image in the first row show a slanted Cheez-It box balanced by hidden supporting objects.
- The model can create niche representations, such as the tip of a screwdriver, with precision.

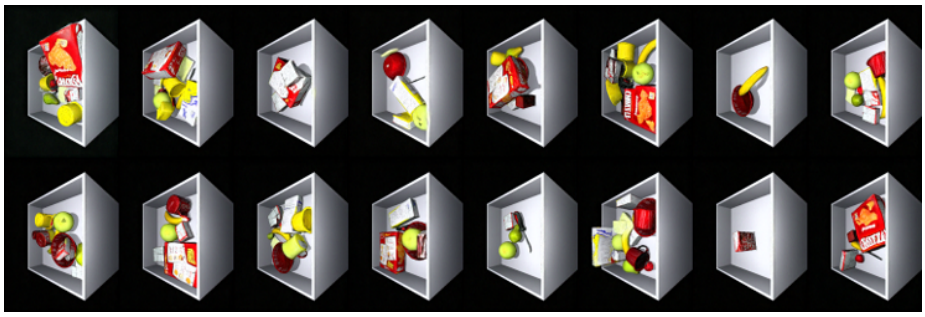


Figure 5.8: Generated images our model is able to create when aided by depth conditioning in the training process.

The depth loss is described in Figure 5.9.

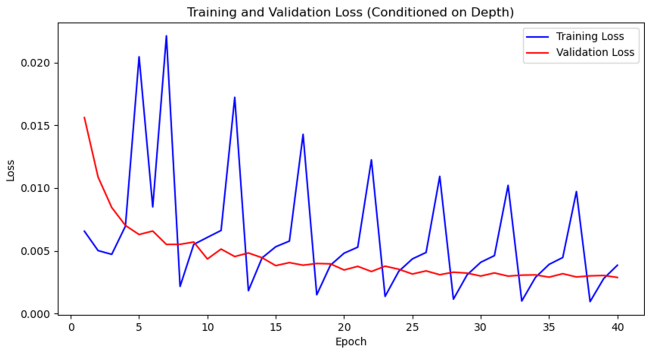


Figure 5.9: Training and Validation Loss for the model that used Depth as conditioning.

5.2 Evaluating with Global Metrics

SSIM, PSNR and LPIPS Metrics: Since MSE focuses on local pixel-to-pixel comparisons, better metrics were sought that are closer to human perception, essentially having the ability to compare global features. The Structural Similarity Index Measure (SSIM) and Peak Signal-to-Noise Ratio (PSNR) are two such metrics. Additionally, the Learned Perceptual Image Patch Similarity (LPIPS) metric is employed for estimating perceptual similarity. This metric maps the difference between deep feature representations of images, thus providing a more human-like quality assessment for images. All of these metrics were tracked for the same validation dataset used to compute the MSE loss.

5.2.1 Why do PSNR, SSIM, and LPIPS lines rapidly fluctuate before converging?

The variations seen in the PSNR, SSIM, and LPIPS graphs below can be attributed to the cosine annealing learning rate schedule used during training. As described in the validation loss section, the learning rate follows a cosine annealing pattern (Figure 5.4), which periodically increases the learning rate

at specified intervals. This helps the model escape local minima on the loss surface, allowing it to aim for the global minimum.

The rapid fluctuations in the learning rate impact the model's performance on global metrics like PSNR, SSIM, and LPIPS. During periods when the learning rate is high, the model may overcorrect, leading to more noticeable variations in these metrics. As the learning rate stabilises, so do the metrics. This pattern results in the "up and down" behaviour observed in the graphs to come.

5.2.2 PSNR Performance

- **PSNR:** Quantifies the difference between the original and distorted images based on signal strength. It is described in Equation 2.2. The higher the score the better the performance. Theoretically, if the predicted and actual images were the same, it would have a score of infinity. The depth-conditioned model scored the highest in this metric, as shown in Figure 5.10. The results stabilise after 35 epochs. PSNR ranks the model conditioned on class labels above the unconditioned in contrast to the validation loss based on MSE in Figure 5.2. However, the margin of difference between the three models is smaller for PSNR compared to the other global metrics. This suggests that the outputs from the models are less distinguishable based on this metric.

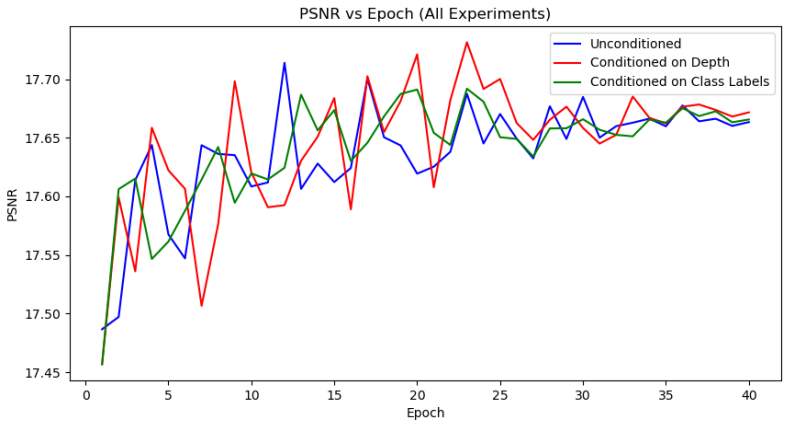


Figure 5.10: PSNR Metric Across All Experiments

5.2.3 SSIM Performance

- **SSIM:** Measures global features and patterns. Scores range from 0 to 1, where 1 is considered a perfect structural match. The results, shown in Figure 5.11, stabilise after 30 epochs. The depth-conditioned model achieved the best SSIM score, followed by class-label conditioning. This also contrasts with the validation loss based on MSE in Figure 5.2, where the unconditioned model performed similarly or better than the class label-conditioned model. The SSIM results further support the conclusion that depth conditioning leads to better performance.

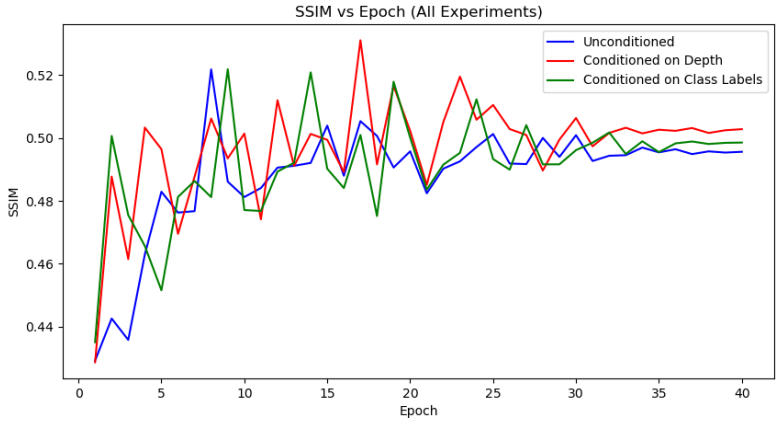


Figure 5.11: SSIM Metric Across All Experiments

5.2.4 LPIPS Performance

LPIPS Metric The Learned Perceptual Image Patch Similarity (LPIPS) metric, a neural network-based measure that detects features similar to human perception, further supports our findings. A score of 0 indicates a perfect match, so we plot $1 - \text{LPIPS}$ for consistency with the other metrics as shown in Figure 5.12. Therefore, we want a value as close to 1 as possible. Here, depth-conditioned generation wins, followed by class-label conditioning, and then unconditioned generation.

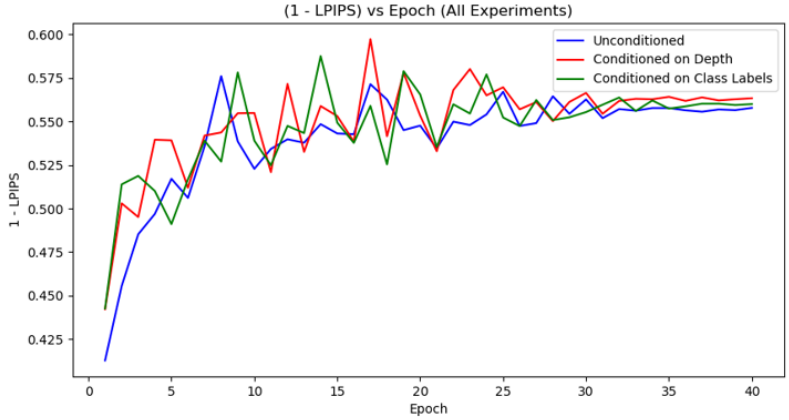


Figure 5.12: LPIPS Metric Across All Experiments

In conclusion, we can see the global metrics come to a different conclusion compared to the MSE validation loss about the 2nd and 3rd placed model, despite both agreeing that the depth-conditioned model performed best.

5.3 Visual Quality Survey

A visual quality survey as shown in Figure 5.13 was conducted where different generated images were shown to an audience. The survey administered is shown in Appendix 8.3. The survey size was 20 individuals, 10 males and 10 females. Participants rated the images on a scale from 1 to 5, with 5 different examples for each type of generation (unconditional, conditioned on depth, and conditioned on class labels). Users did not know which type of example was which, and the order in which the examples were shown was also randomised for fairer scoring. The scores from the 5 different examples were averaged for each type. Each example shown would contain three images as shown in Figure 5.13, to help aid user scoring.

The Rating Criteria were as follows:

1 - Very Poor Clarity Blurry and indistinct, Major artifacts (e.g. pixelations), Low contrast, Unrecognisable.

2 - Poor Clarity Somewhat blurry, Minor artifacts, Low detail, Recognisable with effort.

3 - Average Clarity Decent but not sharp, Minimal artifacts, Moderate detail, Easily recognisable.

4 - Good Clarity Sharp and clear, No noticeable artifacts, High detail, Very recognisable.

5 - Excellent Clarity Extremely sharp, No artifacts, Perfect detail, Stunning and recognisable.

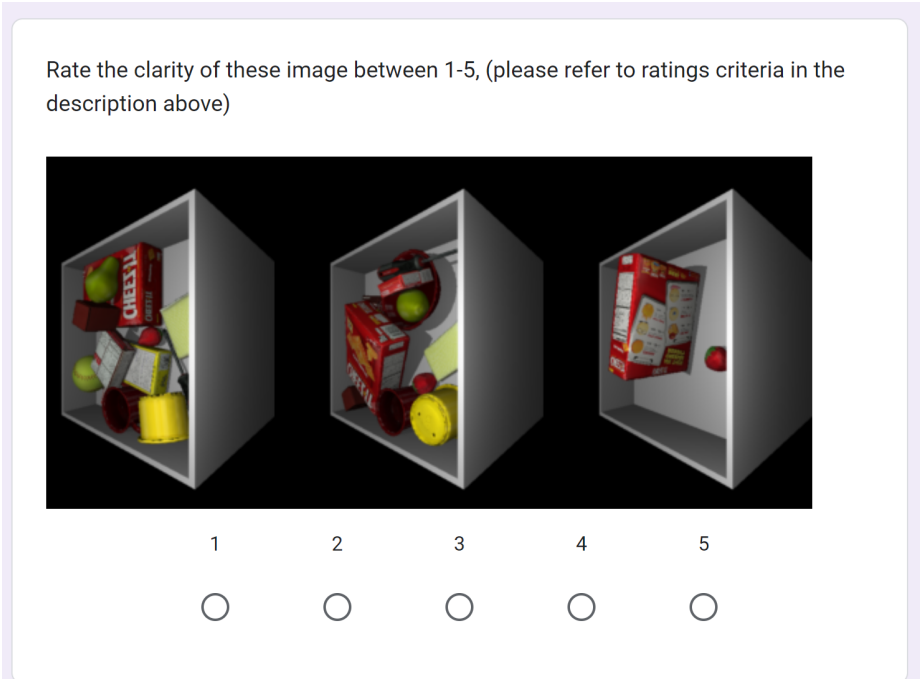


Figure 5.13: Example of the survey with ground truth images.

Figure 5.14 shows the results of the mean ratings received. Where the Original had a score of 3.9, Conditioned on Depth had a score of 3.1, Conditioned on Class had a score of 2.5 and Unconditional a score of 1.9. These results certainly agree with the global metrics evaluations where the PSNR, SSIM and LPIPS metrics ranked the models in the same order. This highlights how global metrics offer a more perceptually aligned evaluation of model performance compared to MSE. While MSE provided a lower error for the unconditioned model than for the class label-conditioned model (as seen in Figure 5.2), global metrics better capture the overall quality and structure of the generated images.

In conclusion, based on the scores in Figure 5.14, we can conclude that the model with conditioning on depth is the most likely to generate images that would pass for the original.

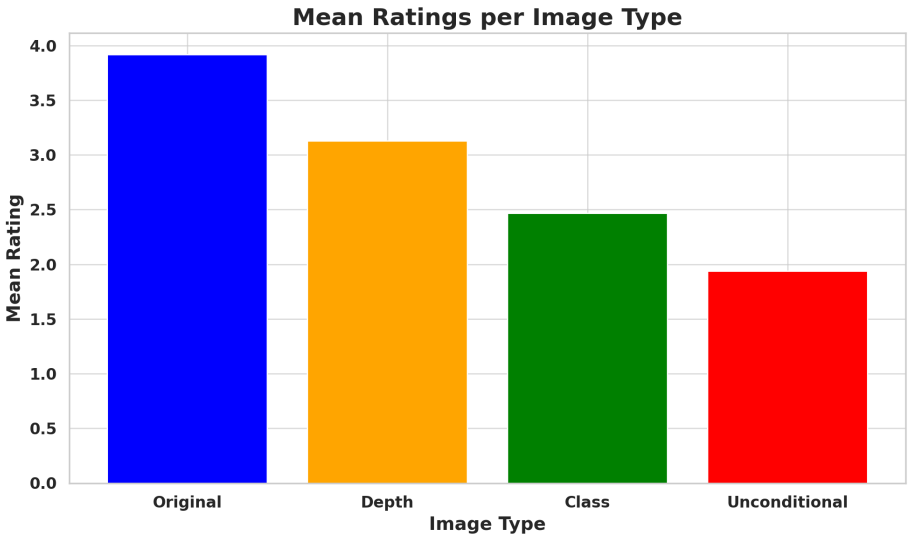


Figure 5.14: Mean Ratings from 1-5 of our varying image types

Figure 5.15 shows the rating across genders, with males giving higher scores for the original, depth and class image types. The results suggest that males rate the generated images of a higher quality compared to female participants, with the exception of the unconditional case. In addition the results show a smaller gap in scores between the different types of images for females versus males which shows a larger difference in ratings as we move between the different types. This suggests that the differences in the quality of the images seem to be more pronounced for males compared to females. Overall, both genders have the same perception in terms of the ranking of the different types of images with the original being the best quality, followed by conditioning on depth, then conditioning on class and finally the unconditional case.

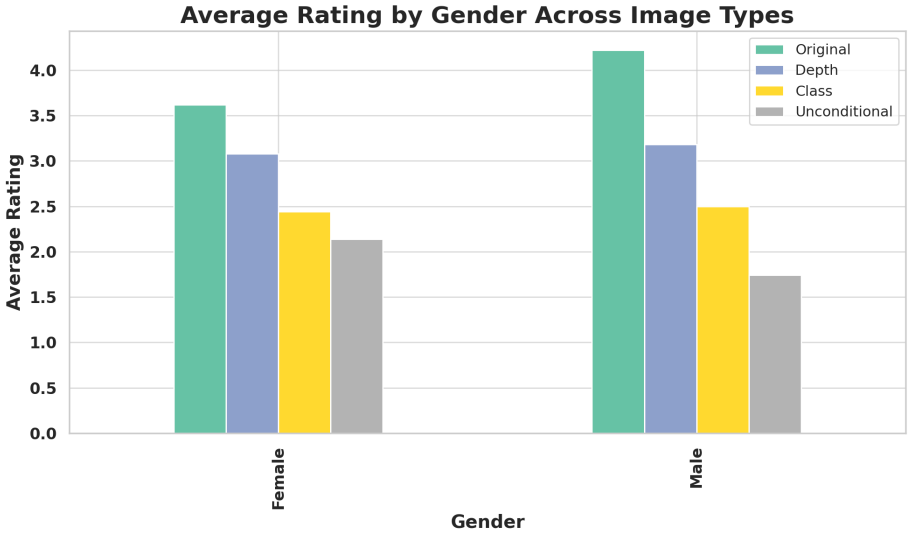
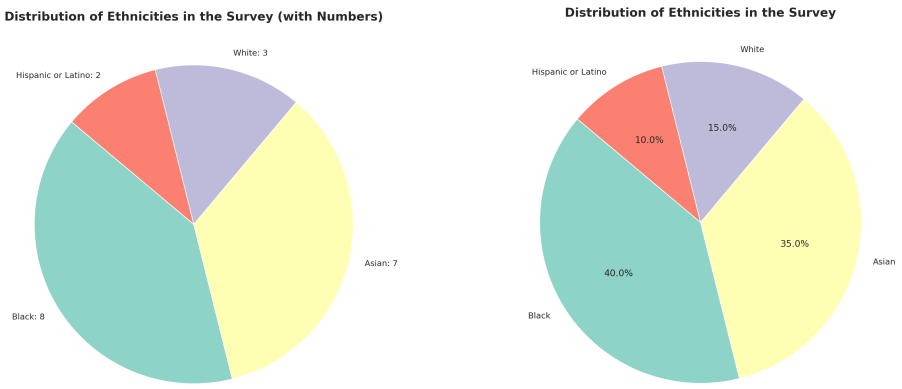


Figure 5.15: Mean Ratings of the Image Types based on Gender (10 females, 10 males).

The 20 participants also recorded their ethnicity shown below in Figure 5.16.



(a) Numbers under each ethnicity.

(b) Percentages under each ethnicity

Figure 5.16: Ethnicities Surveyed.

Figure 5.17 illustrates that, across all groups, the models performed in the following order: original, depth-conditioned, class-conditioned, and unconditioned. Hispanic/Latino participants ranked the original image and depth image of high quality. Surprisingly, they were the only population group to give the original image a score of 5, which means that the image is 'Extremely sharp, No artifacts, Perfect detail, Stunning and recognisable'. This may suggest our training data which consists of original images may need to be of higher quality, which would enable our generative models to output better quality images. Hispanic/Latino participants also had the largest differences in scores between

image types, suggesting the differences in quality were more pronounced for this population group. Another observation is that interestingly, the images conditioned on class scored similarly across most ethnic groups.

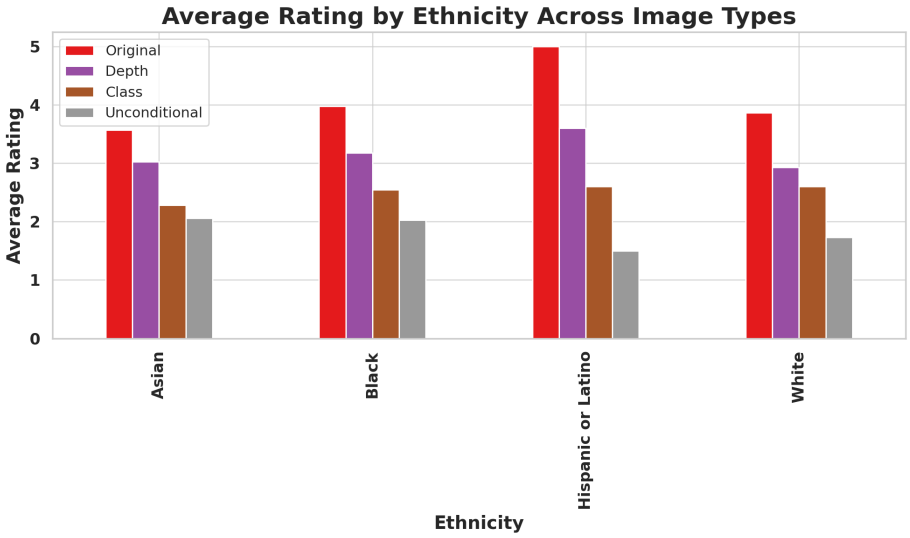


Figure 5.17: Mean Ratings of the Image Types based on Ethnicity (Average amongst participants of each background).

5.4 Discussion

This section discusses the reasons behind the performance of each model. Depth-conditioned generation outperforms the others because:

Depth-conditioned generation outperforms the others because depth conditioning helps form the dominant box shape early in training. This box shape occupies most of the image, allowing the model to focus on learning the objects more accurately, as seen in Figure 5.18. This allows the depth-conditioned model to produce **physically stable** scenes with plausible hidden supporting objects.

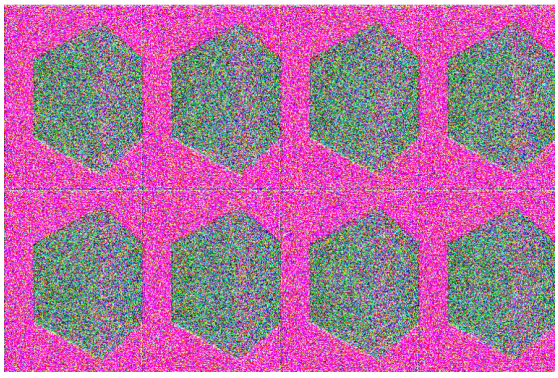


Figure 5.18: Shapes that form in the first epoch for depth conditioning.

The model conditioned on class labels ranks second according to the global metrics and human study. This is because it allows the model to map quickly from Gaussian space to image space by forming initial outlines of objects, as seen in Figure 5.19. However, these objects don't occupy as much space as the box when in the depth case. As a result, the model has more work to do to predict the finer details accurately.

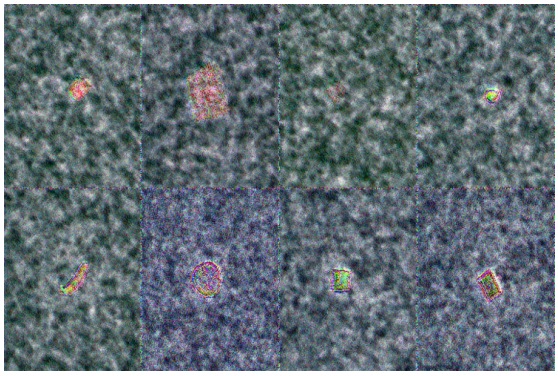


Figure 5.19: Shapes that form in the first epoch for class label conditioning.

The first epoch of training of the unconditional model is seen in Figure 5.20. It doesn't have a prior concept of shapes or objects. Without conditioning information, it faces a much harder task in generating accurate images, which explains its lower performance compared to the other models.

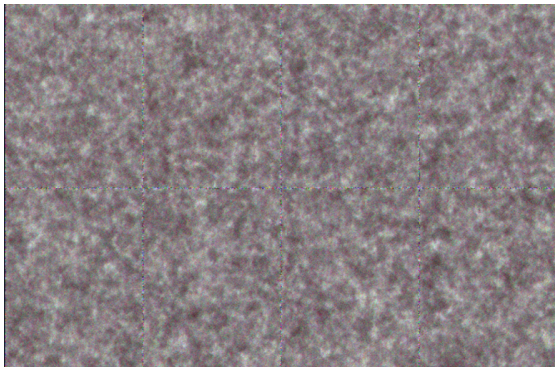


Figure 5.20: Unconditioned generated image visualisation at the first epoch of training

5.5 Technical Challenges

Recognisable Output: The model required additional layers and more parameters to increase its capacity to produce recognisable output, as shown in Figure 5.21.

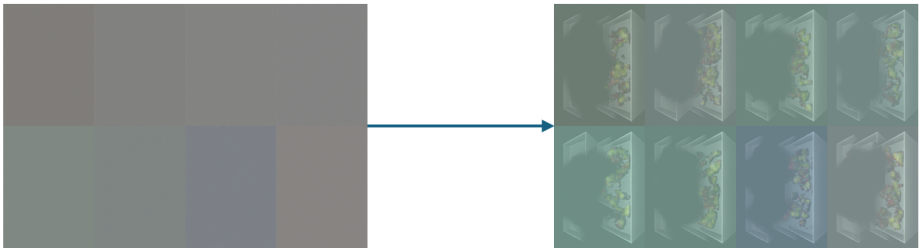


Figure 5.21: Transitioning from blank generations to jagged outline of boxes by increasing model capacity.

Coloration Issues: Images generated exhibited color overlay issues, as shown in Figure 5.22. This was solved ultimately by experimenting with different normalizations as seen in Section 5.6.

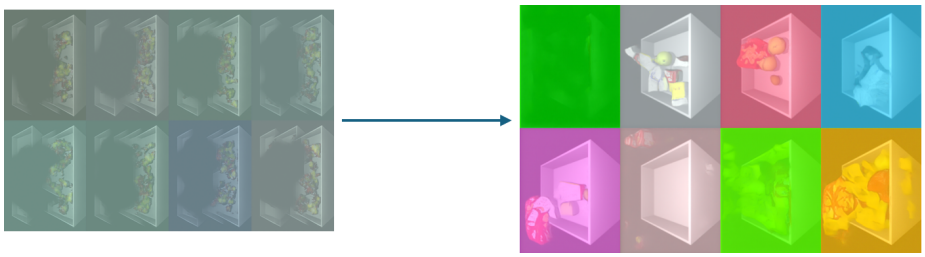


Figure 5.22: Example of the coloration issue.

Artefacts: After resolving coloration, artefacts remained the key challenge as shown in Figure 5.23. Artefacts were mostly resolved through conditioning, specifically depth conditioning. The conditioning process helped eliminate most artefacts observed during image generation.

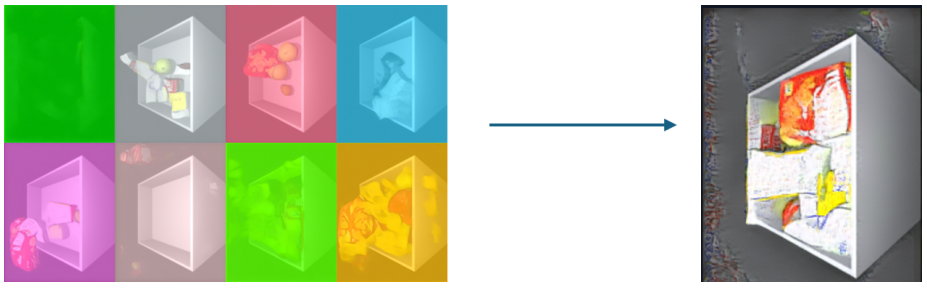


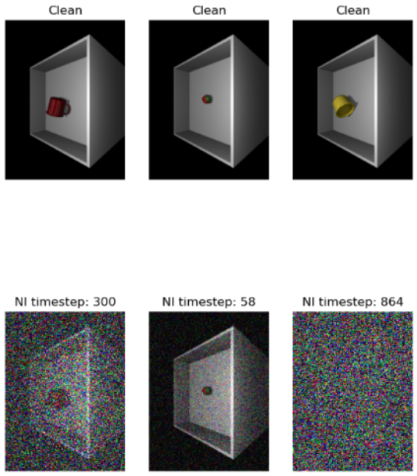
Figure 5.23: Example of Artefacts.

5.6 Normalization Is All You Need

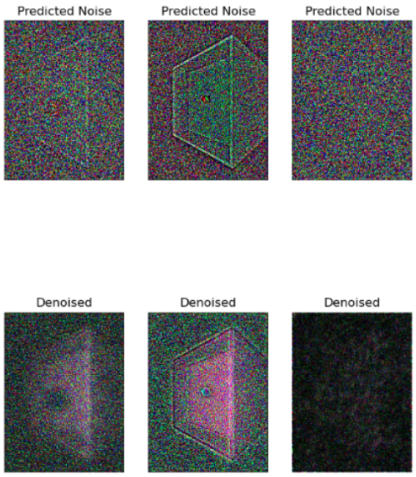
5.6.1 Introduction

One of the key challenges faced was the coloration issue with diffusion models. This section is dedicated to outlining the journey to solving this issue.

The first hypothesis of the reason behind the coloration issue was that this was occurring due to the contrast between the colour of the box and the background. The box takes up a large amount of space and in the denoising step the model struggles to give the appropriate color to the box as shown in Figure 5.24b.



(a) Clean Image and Noisey Image (NI) of a particular timestep of noise.



(b) Predicted Noise and subtracting the noise to get the denoised image.

Figure 5.24: The denoising process, showing coloration issues.

Based on this hypothesis various approaches were tried to vary the color in the training images such as color jitter described in sub-section 5.6.4.

Before implementing color variation techniques, two important steps were taken: hyperparameter tuning and establishing a baseline. Hyperparameter tuning focused on adjusting model parameters, such as increasing the number of layers and altering the learning rate, to see if these changes could mitigate the coloration issue when working with multiple objects in a box.

In addition, a baseline was established using a simpler dataset consisting of 1,000 images, each containing a single object in a box. This smaller dataset enabled quicker experimentation and provided clearer insights into model behaviour. Establishing this baseline allowed for comparison when introducing color variation techniques, such as color jitter, into the training process.

5.6.2 Hyperparameter Tuning

Introducing a deeper UNet by going from 5 to 7 layers in the encoder and decoder of the UNet, using a smaller learning rate of 1e-4 and training for 45 epochs reduced the color significantly for the unconditional case as seen in Figure 5.25 below.

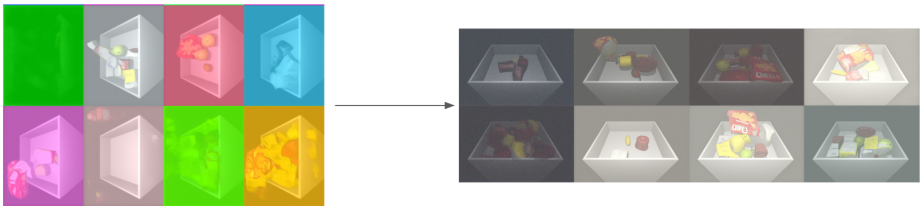


Figure 5.25: Effect of hyperparameter tuning on coloration.

5.6.3 Baseline

A baseline was established for the unconditional model as shown in Figure 5.26 below at 10 epochs. A subset of 1,000 data points where there was only one object in the box, this was done for quick iteration. The color based variations in the upcoming sections were compared to this baseline.

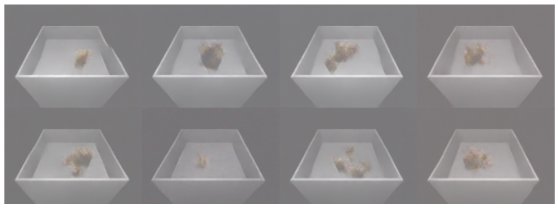


Figure 5.26: Comparison baseline of coloration effect.

For data with a single object in the box the coloration was more subdued compared to training with data of multiple objects inside the box.

5.6.4 Color Jitter

The idea behind color jitter is to introduce small random variations in the colors of an image as a form of data augmentation. Brightness, contrast, saturation, and hue are randomly adjusted. The goal of this experiment was to observe how these small color variations in the objects would affect the coloration issue.

Implementing this in PyTorch only seemed to change the objects inside the box rather than the box itself.

Example of color jitter and its results which were limited is shown in Figure 5.27 and Figure 5.28.

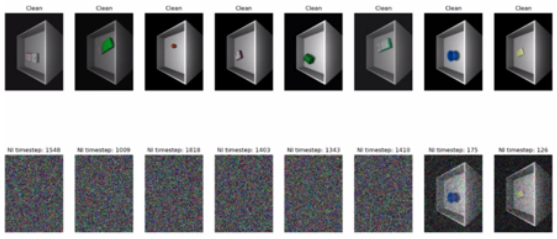


Figure 5.27: Color Jitter Examples.

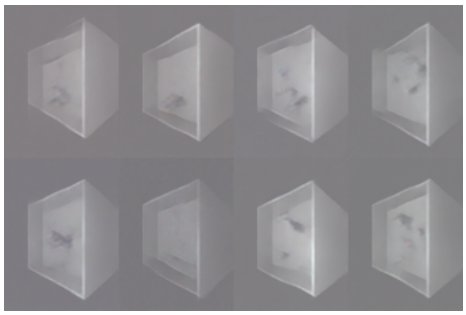


Figure 5.28: Color Jitter Results.

5.6.5 Inversion

Another approach explored was color inversion, where the colors of an RGB image are inverted by subtracting each pixel value from the maximum possible value, as seen in Equation 5.1.

$$p_{\text{inverted}} = M - p \tag{5.1}$$

- p_{inverted} is the inverted pixel value.
- M is the maximum possible pixel value (e.g., 255 for an 8-bit image or 1 for a typical normalized image).
- p is the original pixel value.

Example of inversion and its results are shown in Figure 5.29 and Figure 5.30.

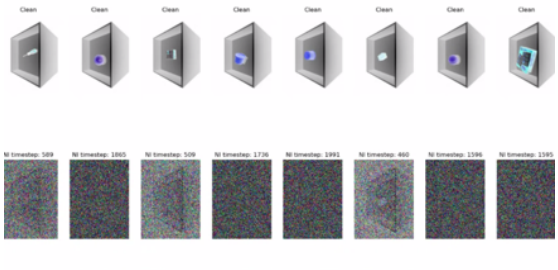


Figure 5.29: Inversion Examples.

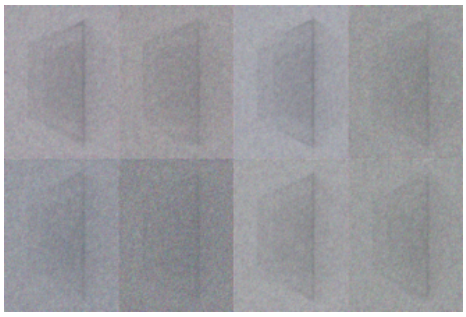


Figure 5.30: Inversion Results.

5.6.6 YCBCr

Another idea was to change the color space from RGB to YCbCR. The Y stands for Luma, which represents the brightness information in an image. Cb stands for Chroma Blue which is the difference between the blue component and the luma (Y). Cr stands for Chroma Red and contains the difference between the red component and the luma. The green component can be derived indirectly from the Y, Cb and Cr values. It is more efficient for certain applications such as video compression as the Cb and Cr can be reduced without impacting image quality as human vision is more sensitive to Y.

The results are seen below in Figure 5.31 and Figure 5.32.

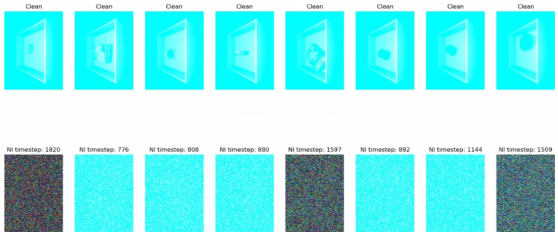


Figure 5.31: YCBCr Examples.

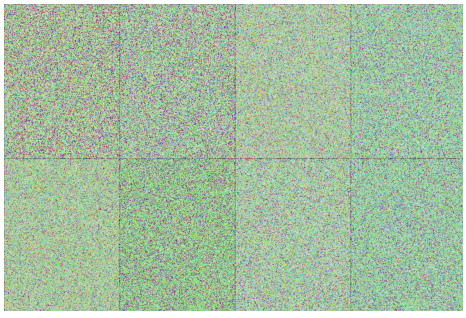


Figure 5.32: YCBCr Results.

5.6.7 White Background

Another idea was to change from a black background to a white background to see if that made it easier for the model to determine the color. The results are shown below in Figure 5.33 and Figure 5.34.

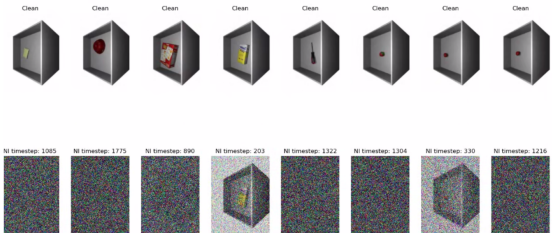


Figure 5.33: White Background Examples.

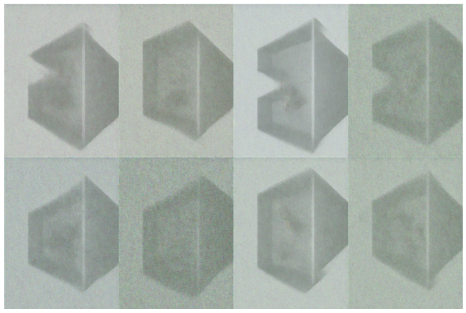


Figure 5.34: White Background Results.

5.6.8 Objects Only Black Background

To test the assumption of the influence of the box, experiments were done without the box. The first of which were the objects only with a black background which still produced coloration as seen in Figure 5.35 and Figure 5.36.

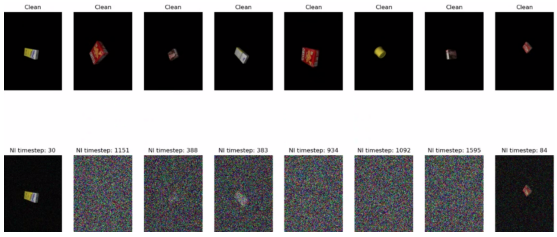


Figure 5.35: Objects Only Black Background Examples.

Figure 5.36: Objects Only Black Background Results after 10 epochs.

5.6.9 Objects Only White Background

Afterwards the objects were tested in a white background setting. This still produced coloration as seen in Figure 5.37 and Figure 5.38. In addition, the objects were harder to distinguish.

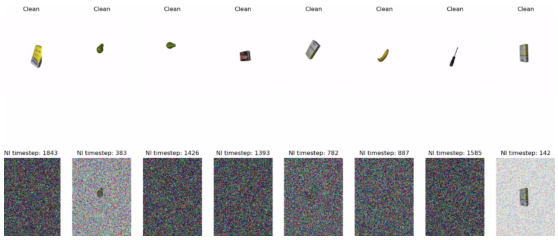


Figure 5.37: Objects Only White Background Examples.

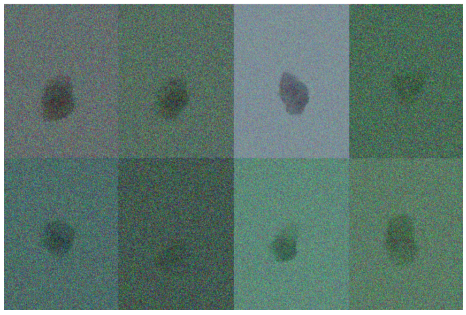


Figure 5.38: Objects Only White Background Results after 10 epochs.

5.6.10 Normalization Experimentation

At this point with a number of options exhausted, a deeper diagnostic was required. As such, the count of the pixel values was taken to understand our color distribution.

5.6.11 0 to 1 Normalization

The pixel values existed between 0 to 1. The data distribution is shown in Figure 5.39 for a single image. As you can see, it is heavily right-skewed with the majority of the pixel values on zero, which represents black.

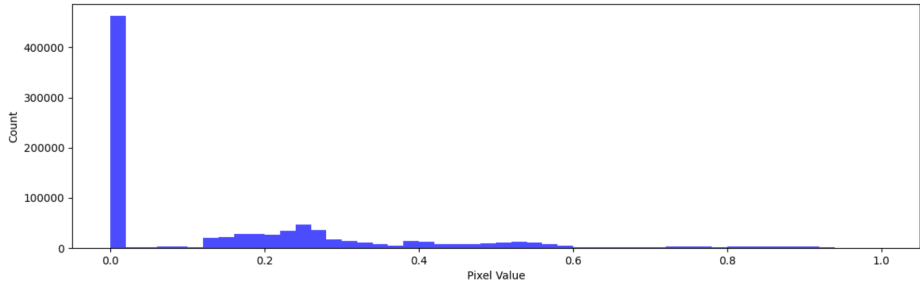


Figure 5.39: Pixels had values between 0 to 1.

5.6.12 -1 to 1 Normalization

A common method for normalization of diffusion models is -1 to 1 normalisation, where all the pixel values fall between -1 and 1. The results of this is shown in Figure 5.40 and Figure 5.41.

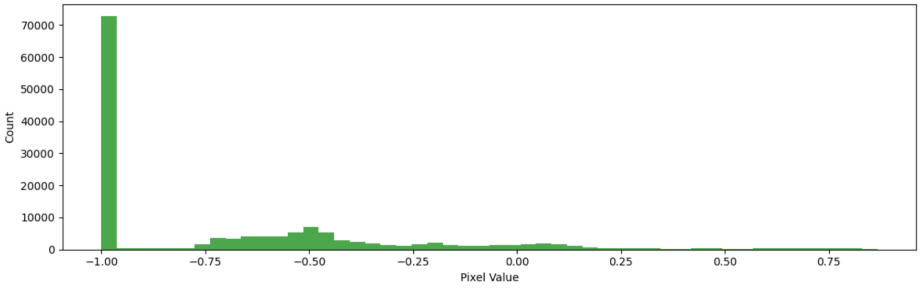


Figure 5.40: Pixels were normalized to have values between -1 to 1.

Figure 5.41: Image results when pixels were normalized to have values between -1 to 1.

5.6.13 Mean and Standard Deviation Normalization

Another method attempted was the calculation of the mean and standard deviation across the images and using that to normalize the data. The results of this is shown in Figure 5.42 and Figure 5.43.

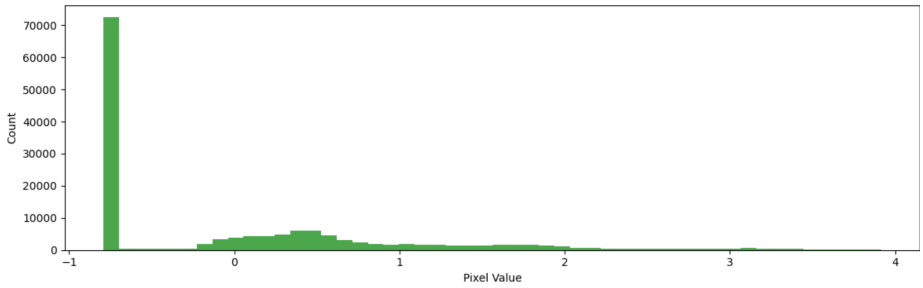


Figure 5.42: Pixels were normalized using the mean and standard deviation across all images. Pixel values were between -1 and 4

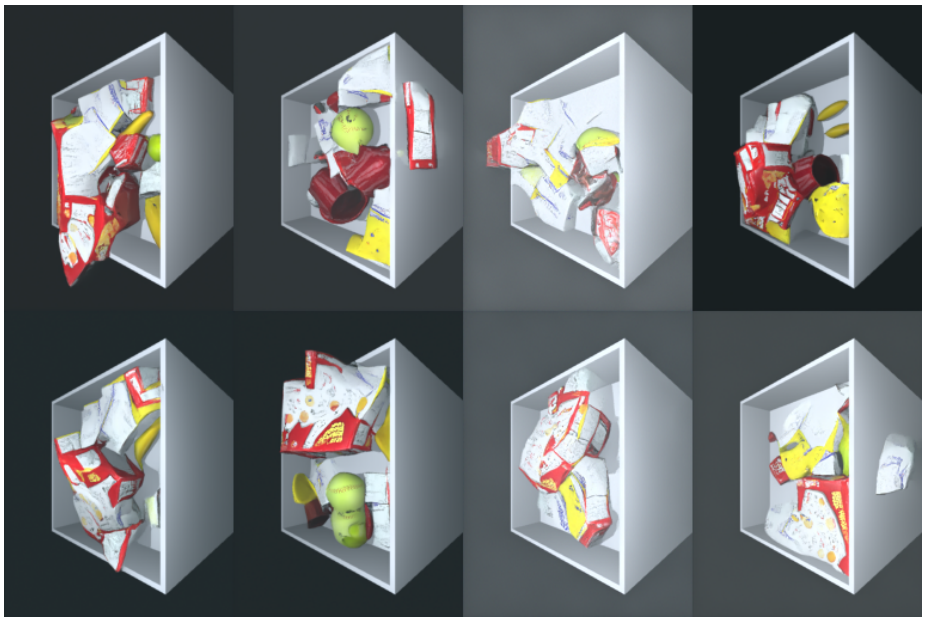


Figure 5.43: Image results of normalizing using the mean and standard deviation across all images.

5.6.14 Exclusive Mean and Standard Deviation Normalization

Following the success of the mean and standard deviation normalization, the final method was to perform the same calculation but only for values greater than 0, effectively excluding the black pixels. The results of this are shown in Figure 5.44 and Figure 5.45. It gives a closer coloration to black but performs worse in the generation of objects as it produces some artefacts.

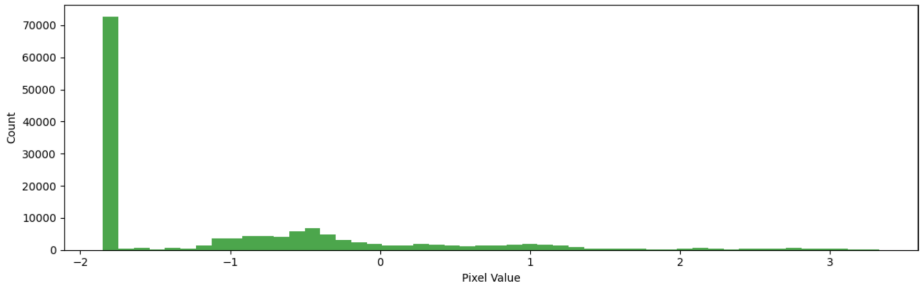


Figure 5.44: Pixels were normalized using the mean and standard deviation across all images (excluding pixels of value zero from the calculation). Pixel values were between -2 and 4.

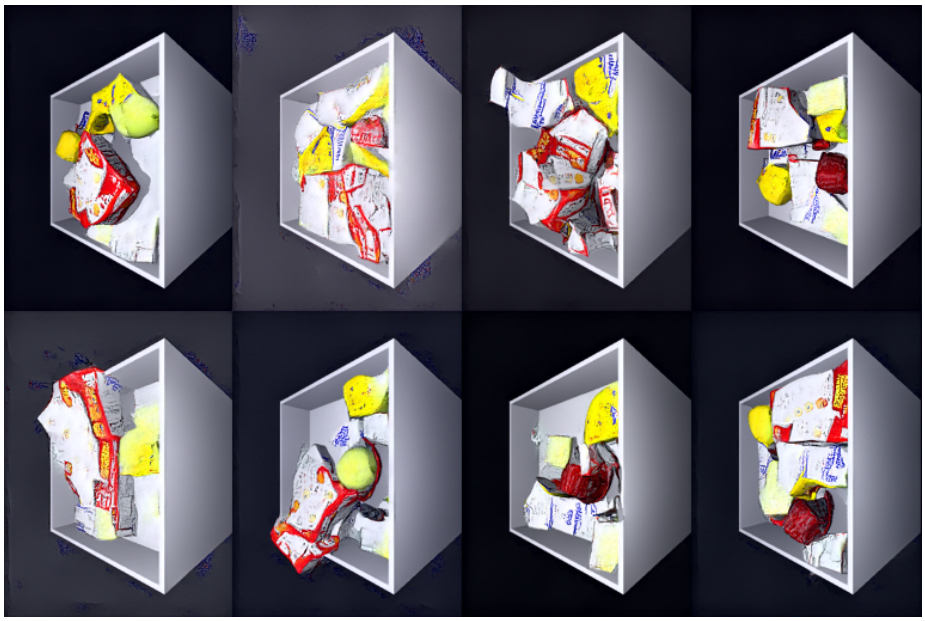


Figure 5.45: Image results of normalizing using the mean and standard deviation across all images(excluding pixels of value zero from the calculation).

5.6.15 Summary

In summary, coloration was a significant issue to overcome for a data set that was heavily right-skewed. A number of techniques were attempted to vary the color, ultimately they failed to produce a more balanced distribution. Taking the time to plot distributions led to exploring different normalization techniques, resulting in a method that significantly reduced the coloration for the unconditioned case and hence for any other conditioning. This method known as exclusive mean and standard deviation normalization, were only pixels with a greater value than 0 were considered.

5.7 Key Contributions

In summary the key contributions made in this project have been the following:

- To the best of my knowledge this is the first study on 2D image diffusion on the YCB-data set.
- A study that considers the effects of unconditional and conditional diffusion on image generation.
- A study on different normalization techniques to improve the color performance of diffusion models.
- Quantitative analysis through metrics like perceptual loss and conducting participant surveys to determine model performance.

- Analysis and discussion of the differences between the groups in our user survey for image generation evaluation.

CHAPTER 6

Project Management

6.1 Time Management and Planning

6.1.1 Initial Gantt Chart

The project was structured to have a series of weekly objectives. These objectives were represented by an initial project Gantt chart as seen in Figure 6.1. This plan was updated on a weekly basis.:

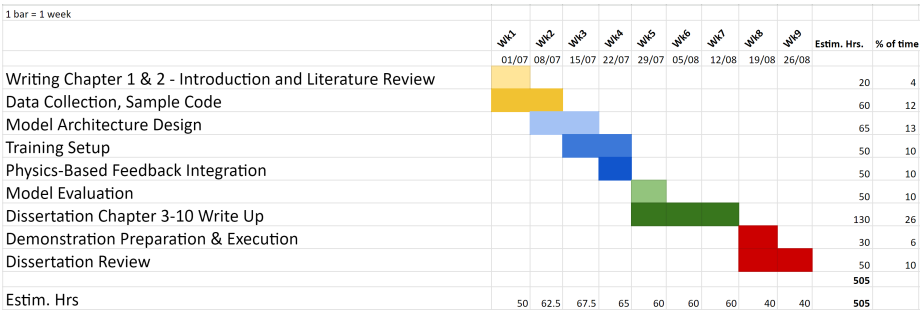


Figure 6.1: Intial project Gantt chart proposed at the start of project.

6.1.2 Weekly Progress

Each week, progress was summarised through presentation slides. These slides documented the work completed, challenges encountered, and questions that arose, and were presented during weekly meetings with the supervisor. This routine allowed for regular feedback and adjustments to the plan to be made as needed.

6.1.3 Time Allocation

Due to the complexity of the project, particularly with the implementation and debugging of diffusion models, longer working hours were required. Additional time was spent on upskilling to use HPC systems and gaining a deep understanding of diffusion models. This phase was crucial, as it laid the foundation for the project's successful implementation.

6.1.4 Challenges Encountered

- **Understanding Previous Work:** Much time was spent understanding the previous work on GANs by Basevi & Leonardis (2022) as it contained almost 5,000 lines of code. In addition, many of the packages had changed. Hence, it was difficult to run on the BEAR HPC system. For diffusion models, the choice was made to start from scratch, given how integrated the previous codebase was.
- **BEAR HPC System:** It was quite a learning curve being able to access BEAR and being able to run successfully on BEAR. Courses such as Introduction to Linux (non-traditional computer science background), Introduction to BlueBEAR and conversations with Advanced Computing Research staff were able to help in successfully navigating the system. This allowed the project to make use of the 40GB GPUs.
- **Understanding Diffusion Models:** Significant time was spent understanding the underlying principles behind diffusion models and working through tutorials.
- **Technical Skills:** Developing proficiency with specific software tools and libraries (e.g., PyTorch, Hugging Face) also took longer than anticipated.

6.1.5 Actual Timeline Gantt Chart

Figure 6.2 shows in reality how the project developed and the time was spent:

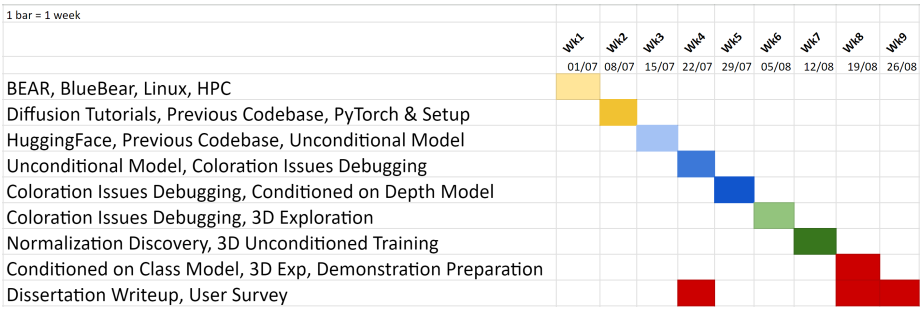


Figure 6.2: Actual week-by-week progress Gantt chart.

6.2 Ethical Considerations (BCS Code of Conduct)

A key consideration for this project was ethical issues when it comes to generative models, which have been used to create deep fakes or to spread disinformation. The BCS Code of Conduct emphasises the importance of producing work that is in the public interest with respect to privacy and well-being. These factors were considered with respect to this project as shown below:

6.2.1 Privacy and Security (Public Interest)

The data is of rendered household objects and doesn't contain any confidential data or privacy concerns. There is a very low risk of harm for the models developed to be adapted to deepfakes or to spread disinformation due to the data it has been trained on.

Next an LSEP Analysis was performed on the project.

6.3 LSEP Analysis

6.3.1 Legal Issues

The YCB Objects are a standard benchmark for use in computer vision. It is open source and freely available for use by the research community.

6.3.2 Economic Factors

This project leveraged BlueBEAR, the University of Birmingham's High Performance Computing cluster for researchers. BEAR was used for data storage and GPU resource allocation to reduce the training time.

6.3.3 Social and Ethical Impact

The social implications of generating synthetic data and its potential misuse were considered. The data used did not have any misleading or harmful content, therefore in its generation it is extremely unlikely to do so. In addition, users are not able to inject prompts to guide the model via text towards a particular desired output. The system is closed to users on what to generate. Users can only influence how many images are generated of household items.

6.3.4 Technological Factors

Diffusion models are at the cutting edge of technology in generative computer vision, this was employed and evaluated during this project.

6.3.5 Environmental Factors

The environmental impact of computational experiments was considered. Efforts were made to optimise code and reduce unnecessary computations, minimising the project's carbon footprint.

6.4 Summary

The project was managed with a focus on ethical responsibility, and the plans were well thought out. There were some challenges to achieving our aims and objectives, but the project was successfully completed.

CHAPTER 7

Conclusion and Future Work

7.1 Conclusion

In this project, three diffusion models have successfully been trained, and their performance has been evaluated quantitatively and qualitatively. A deep understanding of diffusion models was developed by exploring their theoretical foundations and practical implementation in image synthesis, fulfilling the first aim of this project set out in Section 1.2. The three models were both trained with the mean squared error as a loss function that is computationally simple to calculate and has a differentiable form.

In the evaluation of the validation data, we saw that the best model was the diffusion model conditioned on depth, followed by the unconditioned and then the model conditioned with class labels when using the MSE validation loss. However, the results differed when evaluating using global feature-based metrics (PSNR, SSIM, and LPIPS). These global metrics are able to capture more perceptually meaningful characteristics beyond local pixel values. These metrics highlighted that the class-conditioned model outperformed the unconditioned model in terms of global image quality. While MSE was effective for training, the global metrics provided a more comprehensive understanding of the visual realism of the generated outputs, reflecting the second aim of evaluating how conditioning affects the quality and diversity of generated images.

Following this, an image quality study was conducted with 20 participants, 10 males and 10 females from diverse ethnic backgrounds. Participants, with no prior exposure to the images, scored examples from the original training data and the generated images from the three models. The study provided a crucial real-world perspective, confirming that the depth-conditioned model produced the most realistic images, followed by the class-conditioned and unconditioned models, in agreement with the global feature-based metrics. This demonstrated the effectiveness of combining both objective metrics and human evaluations to

rigorously evaluate the models, addressing the third aim of quantitative evaluation.

A key insight from the survey was the gender-based and ethnicity-based differences in the perception of image quality. Female participants perceived fewer differences between the images, while male participants sensed more distinct variations. Additionally, the Hispanic/Latino participants were the only group to give the original image an average score of '5', suggesting they perceived the original image quality to be higher than other groups. In contrast, participants from other ethnic backgrounds rated the original image lower, indicating that the training data could be improved, which would ultimately improve the output of our generative models. These findings provide valuable feedback for future research into the impact of demographic factors on image quality perception.

Another significant lesson learned was the impact of normalization techniques on reducing coloration issues. The data was heavily right-skewed, where the majority of the pixels had a value of zero. By excluding zero-valued pixels from mean and standard deviation calculations, the project effectively reduced the color overlay effect, improving the overall visual quality of the images. The approach taken to solve the coloration issue was methodical and evidence-based and serve to show the success of the project in being able to overcome significant technical challenges.

In conclusion, the project met its objectives by developing a thorough understanding of diffusion models and successfully training three types of models: an unconditioned model, a depth-conditioned model, and a class-conditioned model. The depth-conditioned model consistently generated the most realistic images, demonstrating the importance of conditioning for improving visual quality. Quantitative evaluations confirmed that conditioning on depth significantly improved the model's performance. The project successfully combined quantitative metrics with human surveys to provide a comprehensive evaluation of the models' performances, thereby contributing valuable insights to the field of image synthesis and diffusion models.

7.2 Future Work

7.2.1 3D Generation using Voxels

The next phase of this work involves extending diffusion models to 3D generation using voxels from the dataset by Basevi & Leonardis (2022). Initial experiments included training on four pieces of data over 50 epochs in an unconditioned manner. This resulted in generating pure noise as shown in Figure 7.1 with the loss curve in Figure 7.2 showing it is underfitting (poor performance on training data). This is due to the model capacity needed for the 3D case. The final loss was 0.5, where 1 is the maximum. In the 2D case we would get a final loss of around 0.004. In the 2D case a loss of around 0.018 was when we had forms that were recognisable. Larger datasets are expected to take significantly longer to process with diffusion models compared to their GAN equivalents. For example, while the GAN equivalent took about a week, diffusion models are expected to

take even longer but will promise more diverse outputs.

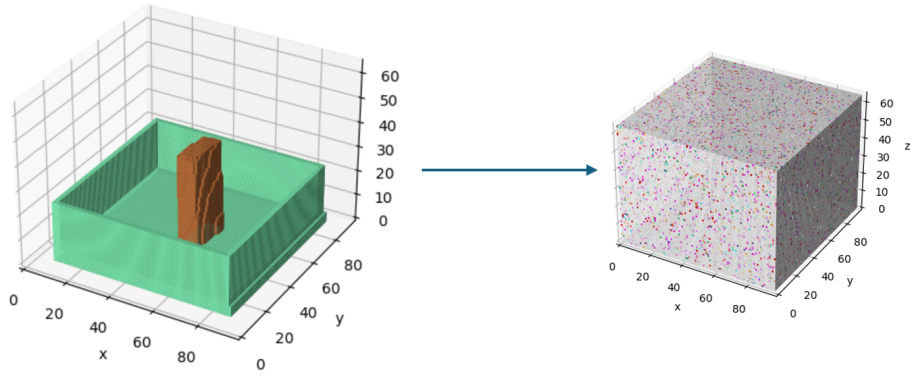


Figure 7.1: Voxel Training Data vs Generated Data

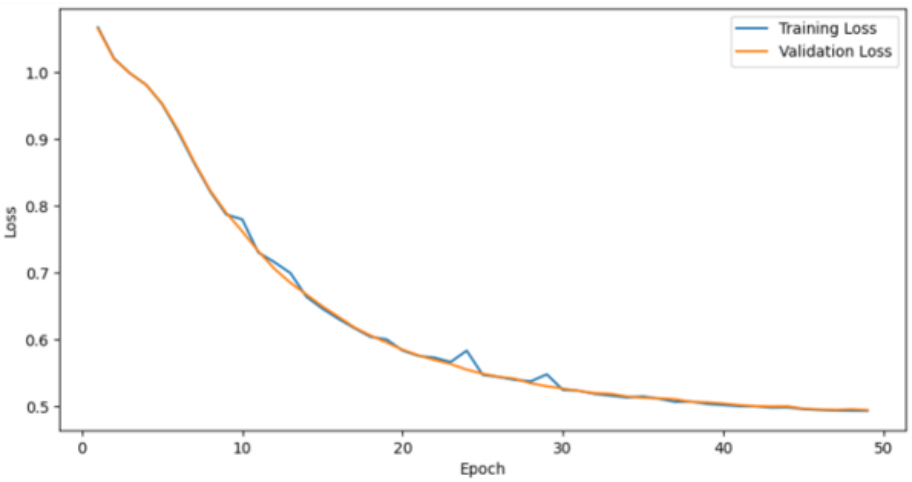


Figure 7.2: Voxel Training Process

To reduce training time, we can use latent diffusion models. This is where we introduce a latent space that compresses the image data in a smaller form through an encoder. The diffusion process is done in this latent space. Afterwards the predicted noise is then expanded to the full image dimensions using a decoder.

7.2.2 2D to 3D meshes

Additionally, exploring the transition from 2D RGB images to 3D meshes is another promising direction. The One-2-3-45 system (Liu & Xu 2023), which involves generating multi-view images with Zero123 (Liu & Wu 2023), presents a relevant approach. However, this method hasn't worked well with our data, often generating distorted shapes instead of accurate representations as shown in Figure 7.3. Therefore, future work will involve generating multi-view images

ourselves using the Renderer from PyTorch3D and adopting a similar system to One-2-3-45 to convert these multi-view images into a 3D mesh.

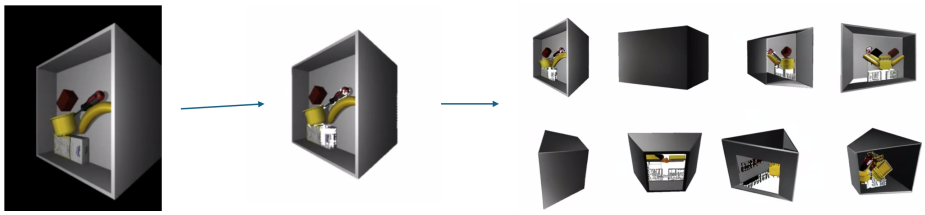


Figure 7.3: Generating Multi-view images with Zero123.

7.2.3 Evaluation of Image Quality Through Downstream Tasks

Another area of future research involves using downstream tasks to evaluate the quality of our objects in the images generated. One example of these downstream tasks is object detection. Although initial experiments showed promise, current models have been trained on real-world data rather than our specific data as shown in Figure 7.4, which is contained within boxes.

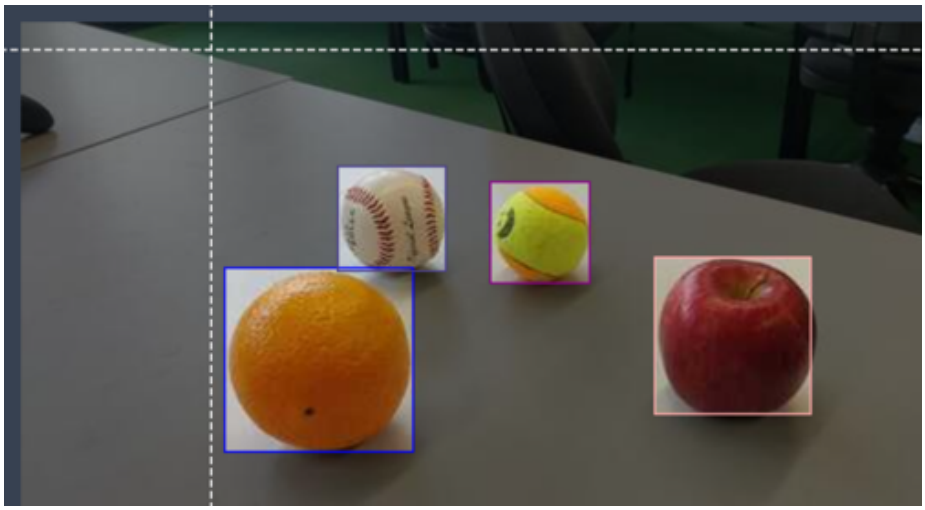


Figure 7.4: Training Data from Object Detection Model found on Roboflow (YCBtrain Dataset 2023).

As a result of this, the system performs better on round or circular objects like balls or cups as seen in Figure 7.5. Future efforts will focus on building a custom object detection system tailored to our data. This will help evaluate the ability of future models to generate realistic objects.

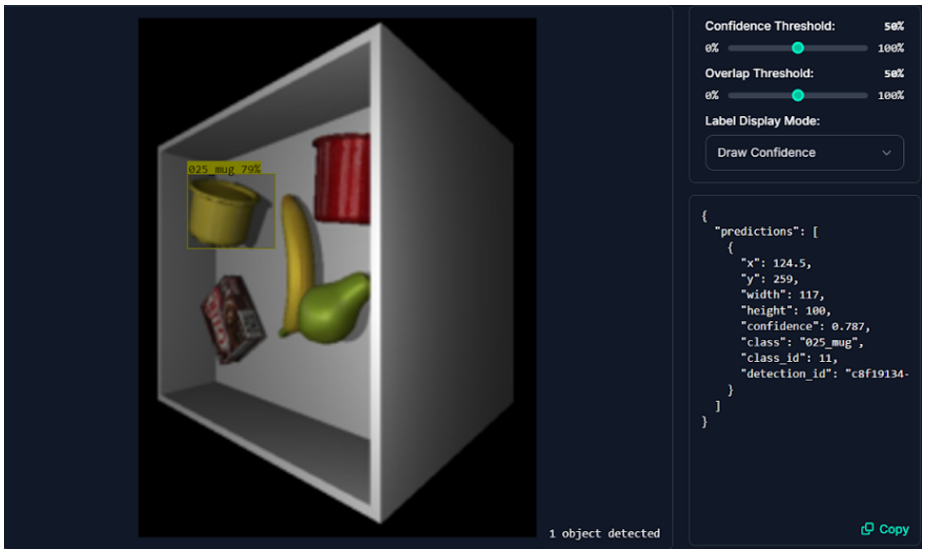


Figure 7.5: Roboflow Object Detection identifies a single object in this scene which is the mug (YCBtrain Model 2023).

7.2.4 Leveraging One-Hot Encoding for Category-Based Image Generation

In the future, instead of training the model on a semantic class label map we can change by using 1D multi-hot vector encoding. Multi-hot vector encoding is explained as follows:

- Imagine you have four items an apple, a banana, an orange, and grapes.
- Using multi-hot vector encoding an image with only apples would be represented as [1,0,0,0], bananas only as [0,1,0,0], oranges only as [0,0,1,0] and grapes only as [0,0,0,1]. A '1' in a position represents the presence of that item and a '0' means it's absent.
- If we had apples and grapes in the image our one-hot vector would look like [1,0,0,1].

In this context, it means we are moving from the semantic class label map to a one-hot vector described in Figure 7.6. Where each of the 14 objects have their one-hot vector representation.

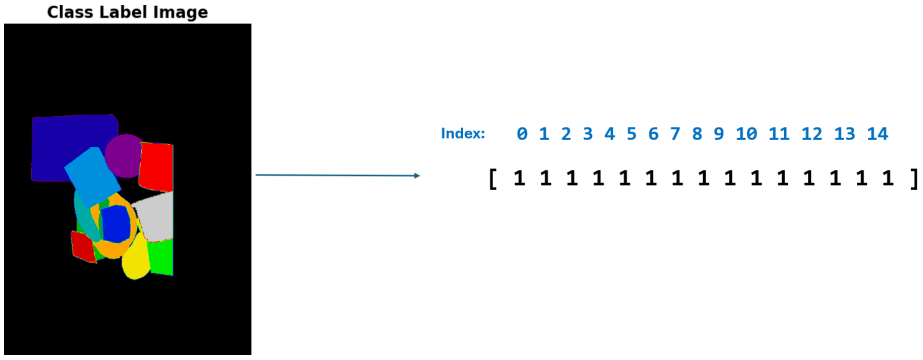


Figure 7.6: Here is a class label image which has all 14 objects, this goes from being represented as a semantic class map to a multi-hot vector encoding where the '1's show the activation of all 14 objects. This is a representation format that trades structural information for conceptual understanding.

This means we lose the structural information from the semantic map, but the model has a more conceptual knowledge of the objects that are in the training image, rather than knowing the positions of the objects explicitly on the map. One of the advantages of this is that it leads to a more general model where users are able to specify the objects that they want in the image. However, there would be a reduction in the quality of generated images. In future, this control can also be expanded to include the number of objects required in each image.

7.2.5 Incorporating Physical Understanding

Another aspect of future work is to incorporate a physical understanding of scenes. This could be achieved by introducing physical stability considerations into the generation process. For example, we could use a pre-trained stability scoring network. The pipeline would convert voxels into actual objects and then use a physics simulator to test the stability of the generated scene. This would build off the work of (Basevi & Leonardis 2022) as shown in Figure 7.7 and Figure 7.8.

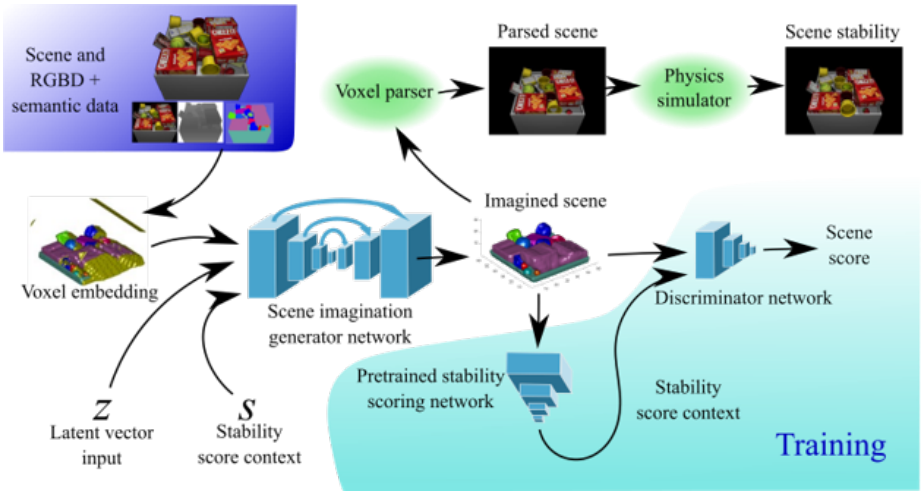


Figure 7.7: Scene Generation System proposed by Basevi & Leonardis (2022)

Figure 7.8: Comparison of 3D Scenes considering physical stability simulations by Basevi & Leonardis (2022) for non-generative regression and generative GAN based approaches.

Figure 7.9 below shows the proposed system for the diffusion model. Where we transform the voxel embeddings into latent space representations due to the data size. An encoder transforms the data into a latent space. The diffusion is done in the latent space before a decoder transforms things back into voxel space.

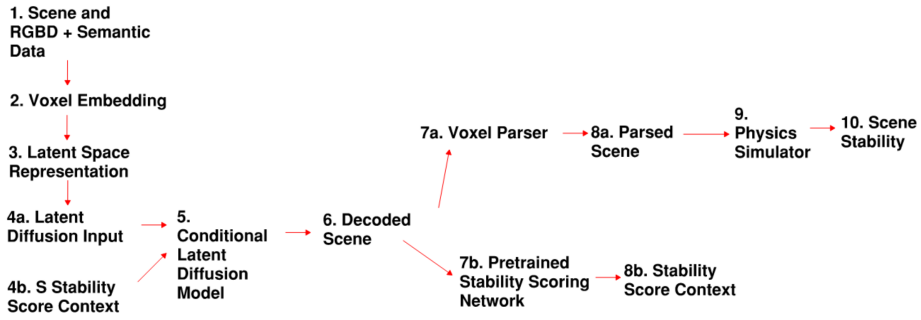


Figure 7.9: Proposed Scene Generation System for Diffusion Models.

7.2.6 Visualising Feature Maps

A final area of exploration involves visualising the feature maps of the U-Net to better understand the concepts the model is learning. An example of feature map visualisation is shown in Figure 7.10. Visualising the feature maps could provide insights into how the model processes images and help refine future models.

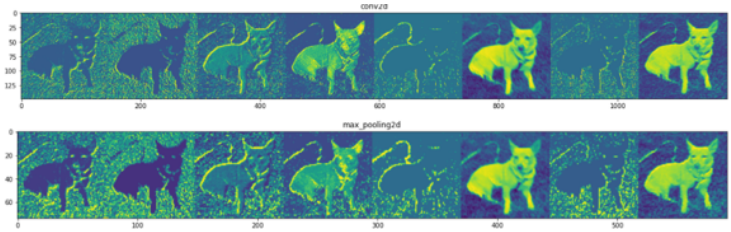


Figure 7.10: Feature maps visualisation of an image of a dog. Source: Shrivastav (2020)

Bibliography

Basevi, H. & Leonardis, A. (2022), 'Imagining hidden supporting objects using volumetric conditional gans and differentiable stability scores'.

Calli, B., Walsman, A., Singh, A., Srinivasa, S., Abbeel, P. & Dollar, A. M. (2015), 'Benchmarking in manipulation research: The ycb object and model set', *IEEE Robotics and Automation Magazine* **22**(3), 36–52.

Dhariwal, P. & Nichol, A. (2021), 'Diffusion models beat gans on image synthesis', *arXiv preprint arXiv:2105.05233* .

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A. & Koltun, V. (2017), Carla: An open urban driving simulator, *in* 'Proceedings of the 1st Annual Conference on Robot Learning (CoRL)', pp. 1–16.

Fardo, F. A., Conforto, V. H., de Oliveira, F. C. & Rodrigues, P. S. (2016), 'A formal evaluation of psnr as quality measurement parameter for image segmentation algorithms', *arXiv preprint arXiv:1605.07116* .

Ho, J., Jain, A. & Abbeel, P. (2020), 'Denoising diffusion probabilistic models', *arXiv preprint arXiv:2006.11239* .

Horé, A. & Ziou, D. (2010), Image quality metrics: Psnr vs. ssim, *in* 'Proceedings of the 2010 International Conference on Pattern Recognition', pp. 2366–2369.

Huang, Q., Park, D. S., Wang, T. & Han, W. (2023), 'Noise2music: Text-conditioned music generation with diffusion models', *arXiv preprint arXiv:2302.03041* . Licensed under Creative Commons Attribution-ShareAlike 4.0 International.
URL: *https://doi.org/10.48550/arXiv.2302.03917*

Hönig, P., Thalhammer, S. & Vincze, M. (2024), 'Improving 2d-3d dense correspondences with diffusion models for 6d object pose estimation', *arXiv preprint arXiv:2402.06436* . Submitted to the First Austrian Symposium on AI, Robotics, and Vision 2024.
URL: *https://doi.org/10.48550/arXiv.2402.06436*

Katsura, N. & Baldassarre, F. (2023), 'pytorch-cosine-annealing-with-warmup', `https://github.com/katsura-jp/pytorch-cosine-annealing-with-warmup`. Accessed: 13/09/2024.

Kirch, S., Olyunina, V., Ondřej, J., Pagés, R., Martín, S. & Pérez-Molina, C. (2023), 'Rgb-d-fusion: Image conditioned depth diffusion of humanoid subjects', *IEEE Access* **11**, 99111–99127.

Labbé, Y., Carpentier, J., Aubry, M. & Sivic, J. (2020), 'Cosypose: Consistent multi-view multi-object 6d pose estimation', *arXiv preprint arXiv:2008.08465* .

Lei, J., Tang, J. & Jia, K. (2023), 'Rgbd2: Generative scene synthesis via incremental view inpainting using rgbd diffusion models', pp. 1–12.
URL: *CVPR 2023 Paper*

Lisanti, G. & Giambi, N. (2024), 'Conditioning diffusion models via attributes and semantic masks for face generation', *Computer Vision and Image Understanding* **244**, 104026.

Liu, M. & Xu, C. (2023), 'One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization', *arXiv preprint arXiv:2306.16928* .

Liu, R. & Wu, R. (2023), 'Zero-1-to-3: Zero-shot one image to 3d object', *arXiv preprint arXiv:2303.11328* .

Luma AI (2024), 'Biblically accurate gymnastics, dream machine', `https://www.youtube.com/watch?v=gJvuwLybUjs`. Generated using Luma Dream Machine, `https://lumalabs.ai/dream-machine`.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P. & Ommer, B. (2022), 'High-resolution image synthesis with latent diffusion models', *arXiv preprint arXiv:2112.10752* .

Ronneberger, O., Fischer, P. & Brox, T. (2015), 'U-net: Convolutional networks for biomedical image segmentation', *arXiv preprint arXiv:1505.04597* .

Rouse, D. & Hemami, S. S. (2008), 'Understanding and simplifying the structural similarity metric', pp. 1188–1191.

Seymour, N. E., Gallagher, A. G., Roman, S. A., O'Brien, M. K., Bansal, V. K., Andersen, D. K. & Satava, R. M. (2002), 'Virtual reality training improves operating room performance: Results of a randomized, double-blinded study', *Annals of Surgery* **236**(4), 458–464.

Shrivastav, S. (2020), 'Tutorial — how to visualize feature maps directly from cnn layers'. Accessed: 2024-09-09.
URL: *https://www.analyticsvidhya.com/blog/2020/11/tutorial-how-to-visualize-feature-maps-directly-from-cnn-layers/*

Szarvas, D. & Pogány, D. (2024), 'Conditional molecule generation with 2d latent diffusion', *Proceedings of the 31th Minisymposium on Measurement and Information Systems* .

van den Oord, A., Vinyals, O. & Kavukcuoglu, K. (2018), 'Neural discrete representation learning', *arXiv preprint arXiv:1711.00937* .

Wang, Q., Zhang, H., Deng, C., You, Y., Dong, H., Zhu, Y. & Guibas, L. (2024), 'Sparsedff: Sparse-view feature distillation for one-shot dexterous manipulation', *arXiv preprint arXiv:2310.16838* . Last revised 18 Mar 2024.
URL: *https://doi.org/10.48550/arXiv.2310.16838*

Wang, Z., Bovik, A. C., Sheikh, H. R. & Simoncelli, E. P. (2004), 'Image quality assessment: From error visibility to structural similarity', *IEEE Transactions on Image Processing* **13**(4), 600–612.

Xiao, Z., Kreis, K. & Vahdat, A. (2022), 'Tackling the generative learning trilemma with denoising diffusion gans', *arXiv preprint arXiv:2112.07804* .

YCBtrain Dataset (2023), 'Ycb object dataset', `https://universe.roboflow.com/ycbtrain/ycb-object-dataset/browse?queryText=&pageSize=50&startingIndex=0&browseQuery=true`. Accessed: 2024-09-09. This is available via Creative Commons Attribution 4.0 International License.

YCBtrain Model (2023), 'Roboflow object detection model', `https://universe.roboflow.com/ycbtrain/ycb-object-dataset`. Accessed: 2024-09-09. This is available via Creative Commons Attribution 4.0 International License.

Zhang, R., Isola, P., Efros, A. A., Shechtman, E. & Wang, O. (2018), The unreasonable effectiveness of deep features as a perceptual metric, *in* 'IEEE Conference on Computer Vision and Pattern Recognition (CVPR)', pp. 586–595.

CHAPTER 8

Appendix:

This appendix contains the important details regarding the code repository as well as information related to the participant survey. The next section outlines the GitLab repository link, an overview of the repository's contents, and where to find step-by-step instructions on how to run the software.

8.1 GitLab Project Repository:

The project code is available at the following GitLab repository:

- **Repository URL**: `https://git.cs.bham.ac.uk/projects-2023-24/oia295.git`

8.2 Repository Contents:

The repository contains the following files and folders:

- **generation.py**: Script for generating images using the pre-trained diffusion models.
- **unconditioned.ipynb**: Script for training the unconditioned diffusion model.
- **condition_depth.ipynb**: Script for training the diffusion model conditioned on depth.
- **condition_labels.ipynb**: Script for training the diffusion model conditioned on the class labels semantic map.
- **requirements.txt**: Lists the package dependencies required to run the project.
- **README.md**: Detailed instructions on setting up the environment, installing dependencies, and running the code.

8.3 Survey

The next page shows a printout of the online survey participants filled out. Figure 8.1 below shows the raw data that was logged as participants took part in the survey.

A	B	C	D	E	F	G	H	I
Timestamp	Gender	Ethnicity	Rate the clarity of these i	Rate the clarity of these i	Rate the clarity of these i	Rate the clarity of these i	Rate the clarity of these i	Rate the clarity of these
8/26/2024 17:39:07	Male	Black, African, Caribbear	4	3	1	2	1	5
8/26/2024 17:42:48	Male	Asian	3	3	3	2	3	4
8/26/2024 17:47:15	Male	White	4	3	3	3	2	4
8/26/2024 17:55:41	Male	Hispanic or Latino	5	3	2	2	2	5
8/26/2024 17:55:50	Male	Hispanic or Latino	5	4	3	3	2	5
8/26/2024 17:57:47	Female	Asian	5	3	2	2	2	5
8/26/2024 18:19:16	Female	Asian	3	2	1	1	1	3
8/26/2024 18:21:18	Female	Asian	4	3	3	3	3	5
8/26/2024 18:31:06	Female	Black, African, Caribbear	4	3	3	4	3	4
8/26/2024 18:38:56	Male	White	3	3	3	3	3	3
8/26/2024 18:52:58	Female	Black, African, Caribbear	4	5	5	4	4	4
8/26/2024 19:16:48	Female	Black, African, Caribbear	3	3	3	2	2	3
8/26/2024 19:23:22	Male	Black, African, Caribbear	5	5	3	4	2	3
8/26/2024 21:20:09	Male	Black, African, Caribbear	3	3	3	3	2	4
8/26/2024 22:09:22	Female	Black, African, Caribbear	3	2	2	3	1	4
8/27/2024 0:39:51	Male	Asian	3	2	2	3	2	4
8/27/2024 9:41:48	Female	Asian	3	4	3	3	3	4
8/27/2024 9:53:04	Male	White	4	3	2	2	1	5
8/27/2024 20:35:02	Female	Asian	2	4	3	4	4	3
8/28/2024 4:33:51	Female	Black, African, Caribbear	2	2	2	1	1	3

Figure 8.1: Raw logged data from participants filling in the survey.

Visual Quality Survey!

This survey is to inform my computer vision dissertation.
Thank you for your help!

Rating Criteria is based on this:

1 - Very Poor Clarity
Blurry and indistinct, Major artifacts (e.g. pixelations), Low contrast, Unrecognizable

2 - Poor Clarity
Somewhat blurry, Minor artifacts, Low detail, Recognizable with effort

3 - Average Clarity
Decent but not sharp, Minimal artifacts, Moderate detail, Easily recognizable

4 - Good Clarity
Sharp and clear, No noticeable artifacts, High detail, Very recognizable

5 - Excellent Clarity
Extremely sharp, No artifacts, Perfect detail, Stunning and recognizable

1. Gender

Check all that apply.

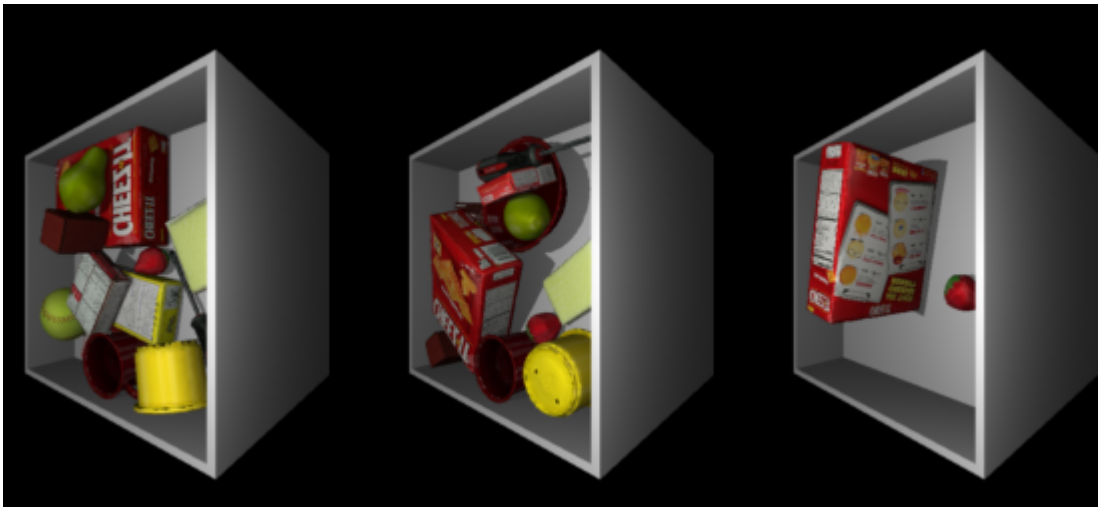
- [ ] Male
- [ ] Female
- [ ] Prefer not to say

2. Ethnicity

Mark only one oval.

- White
- Black, African, Caribbean, or Black British
- Hispanic or Latino
- Asian
- Middle Eastern
- Mixed/Multiracial
- Other
- Prefer not to say

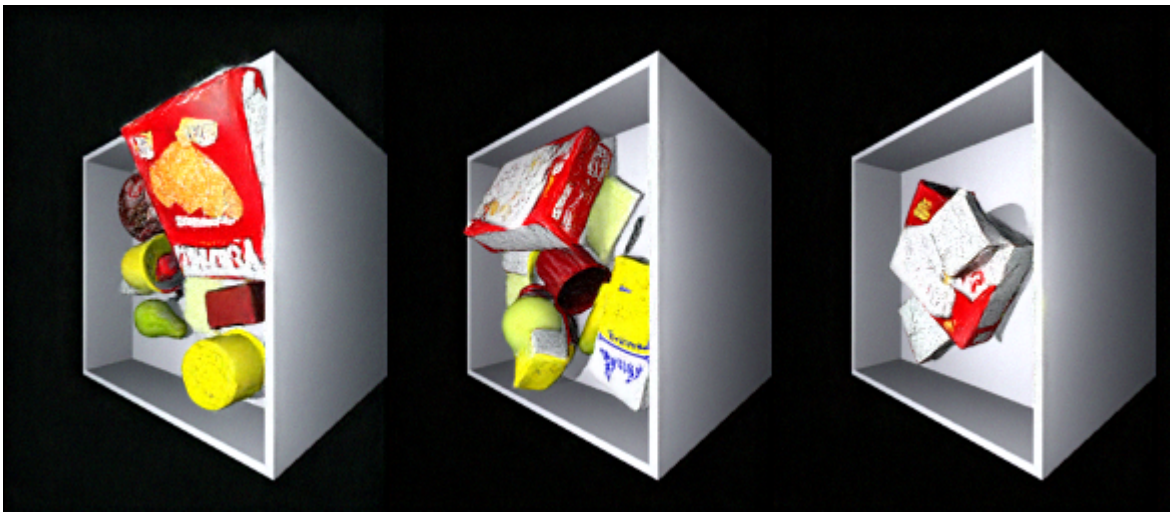
3. Rate the clarity of these image between 1-5, (please refer to ratings criteria in the description above)



Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

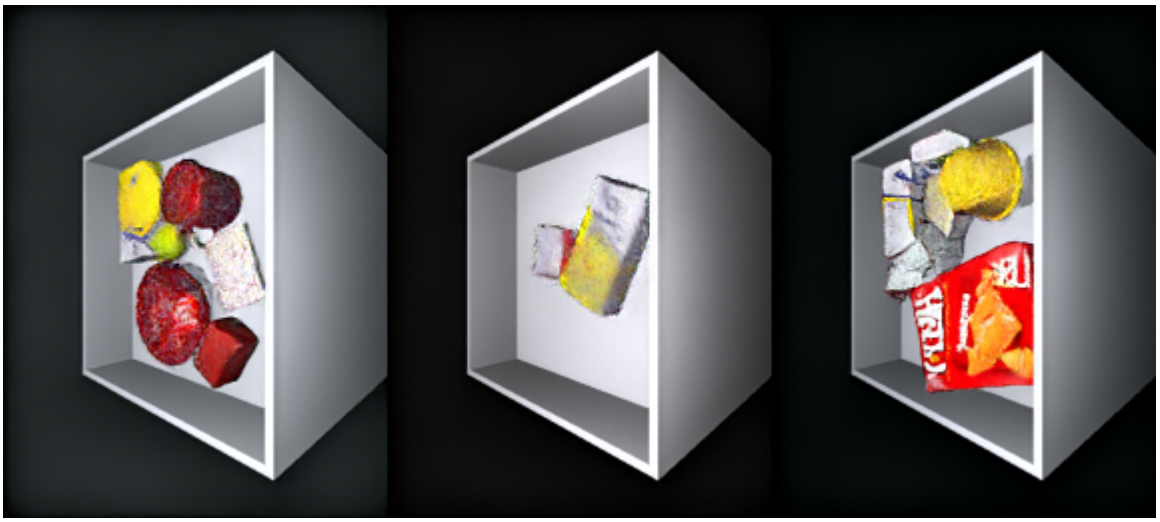
4. Rate the clarity of these image between 1-5, (please refer to ratings criteria in the description above)



Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

5. Rate the clarity of these image between 1-5, (please refer to ratings criteria in the description above)



Mark only one oval.

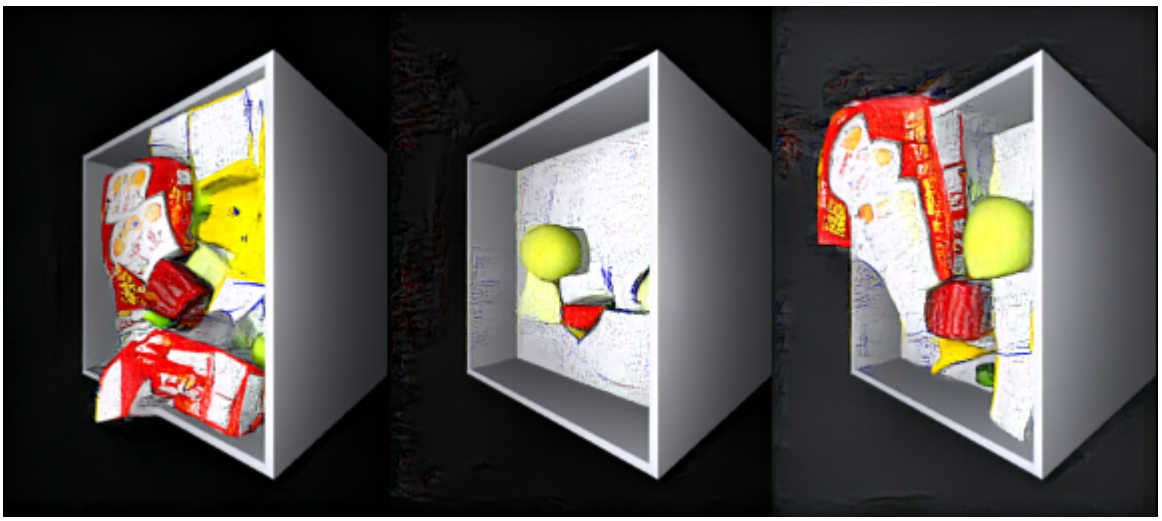
1	2	3	4	5
◯	◯	◯	◯	◯

6. Rate the clarity of these image between 1-5, (please refer to ratings criteria in the description above)

Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

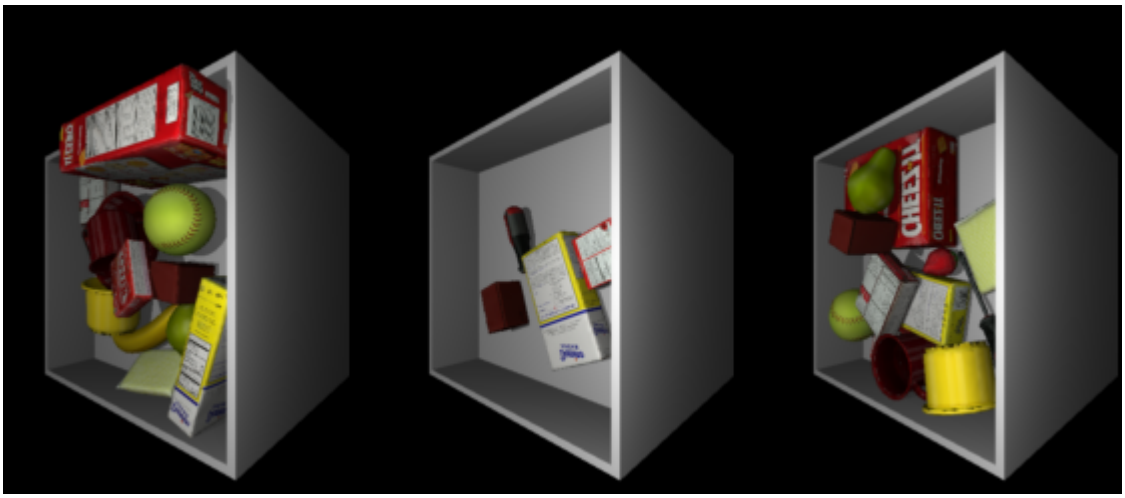
7. Rate the clarity of these image between 1-5, (please refer to ratings criteria in the description above)



Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

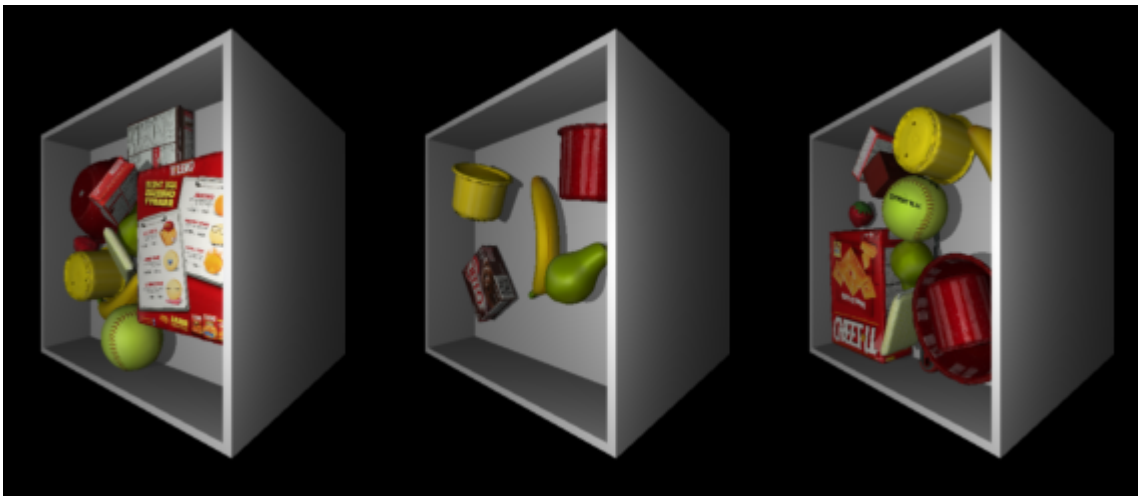
8. Rate the clarity of these image between 1-5, (please refer to ratings criteria in the description above)



Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

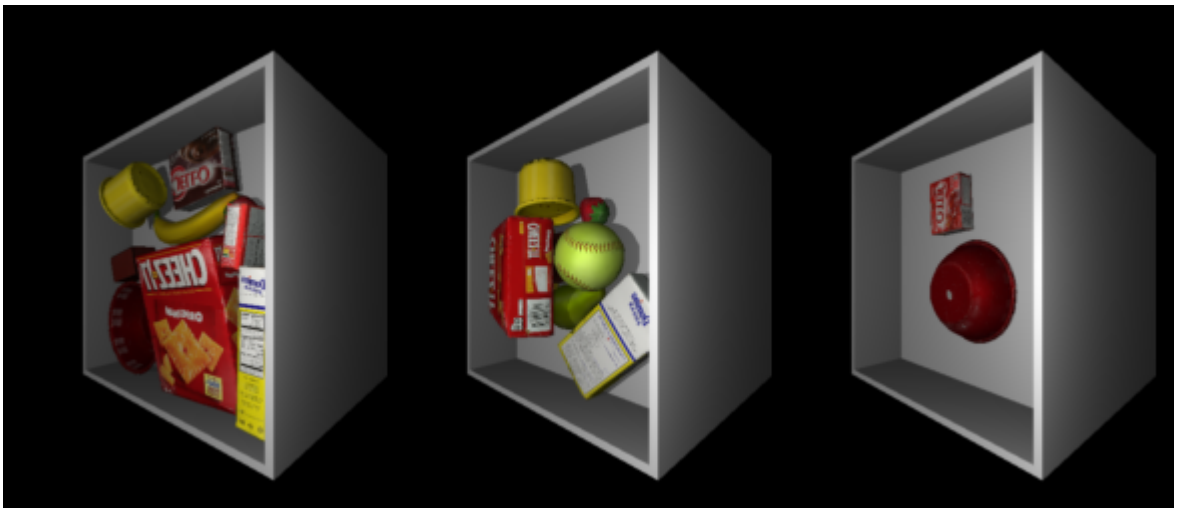
9. Rate the clarity of these image between 1-5, (please refer to ratings criteria in the description above)



Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

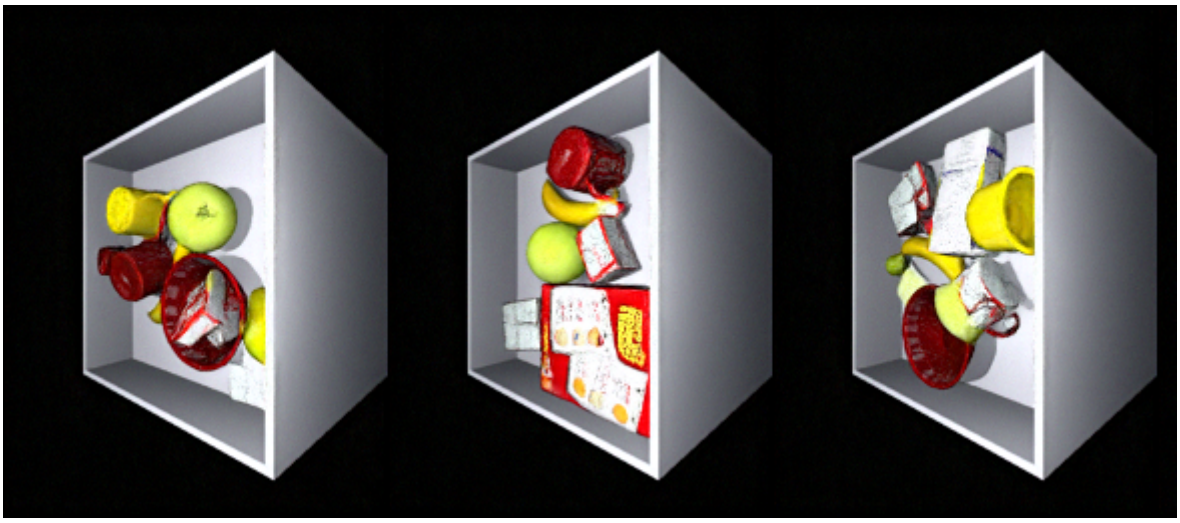
10. Rate the clarity of these image between 1-5, (please refer to ratings criteria in the description above)



Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

11. Rate the clarity of these image between 1-5, (please refer to ratings criteria in the description above)



Mark only one oval.

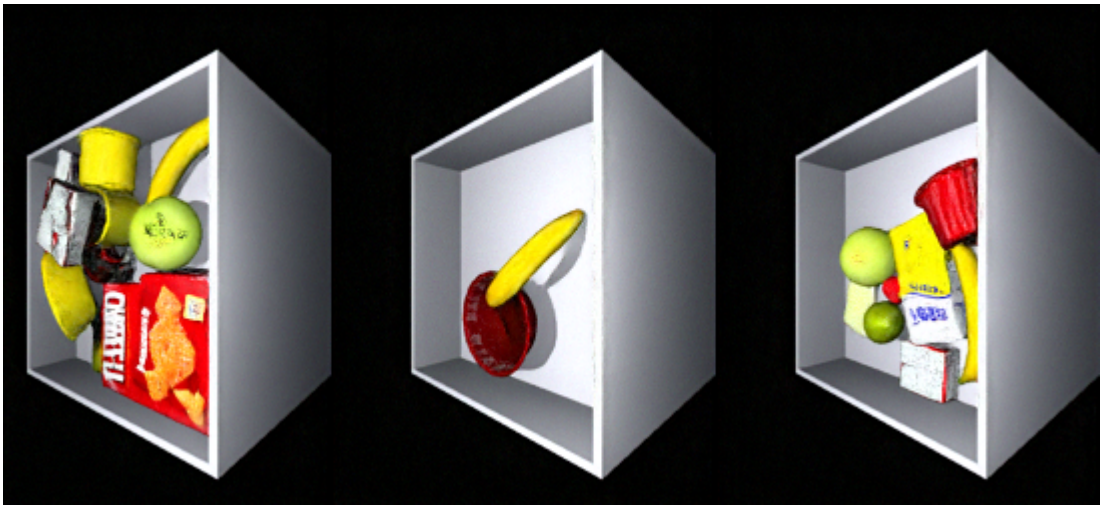
1	2	3	4	5
◯	◯	◯	◯	◯

12. Rate the clarity of these image between 1-5, (please refer to ratings criteria in the description above)

Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

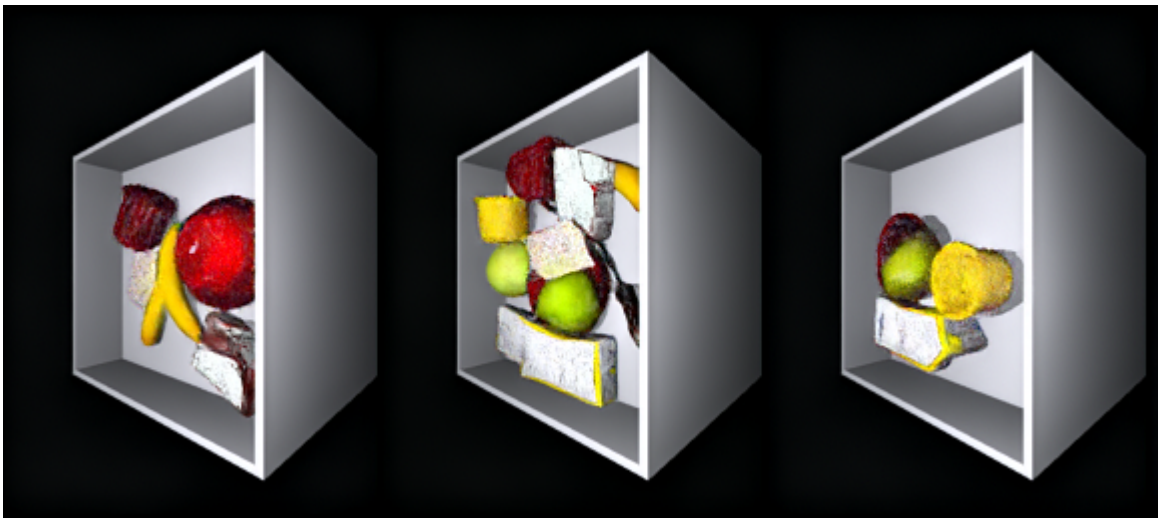
13. Rate the clarity of these image between 1-5, (please refer to ratings criteria in the description above)



Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

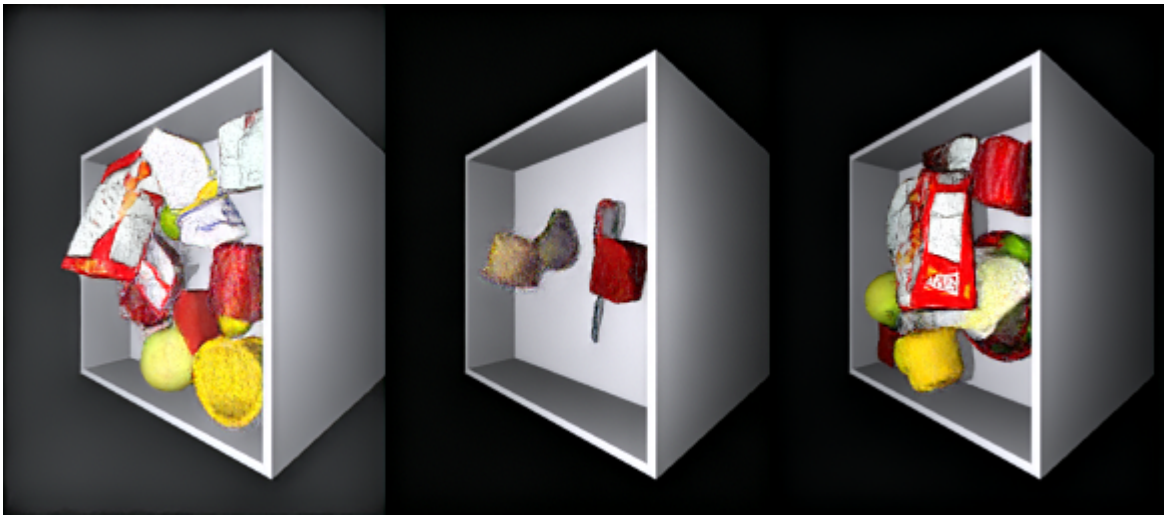
14. Rate the clarity of these image between 1-5, (please refer to ratings criteria in the description above)



Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

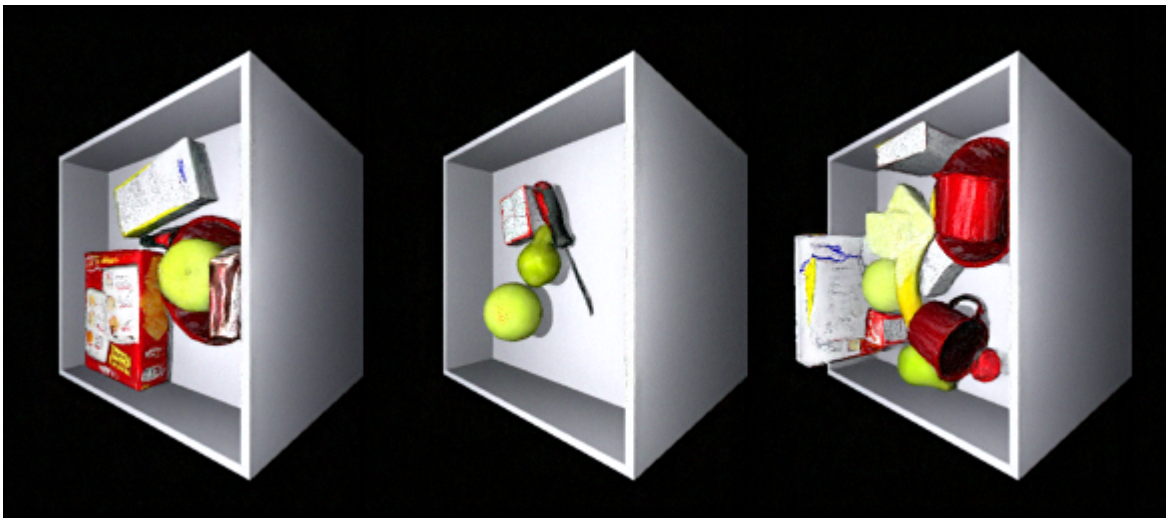
15. Rate the clarity of these image between 1-5, (please refer to ratings criteria in the description above)



Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

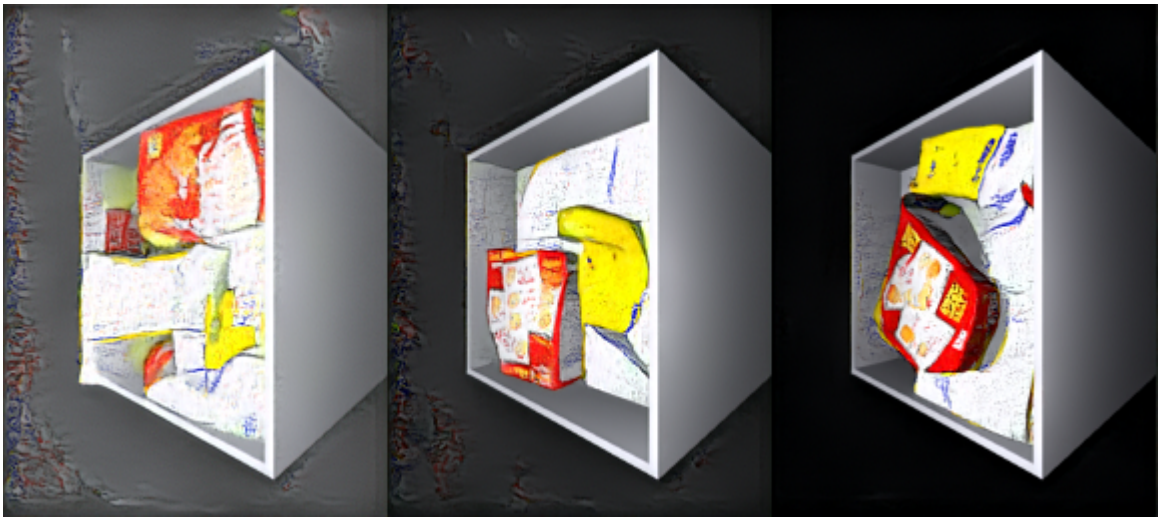
16. Rate the clarity of these image between 1-5, (please refer to ratings criteria in the description above)



Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

17. Rate the clarity of these image between 1-5, (please refer to ratings criteria in the description above)



Mark only one oval.

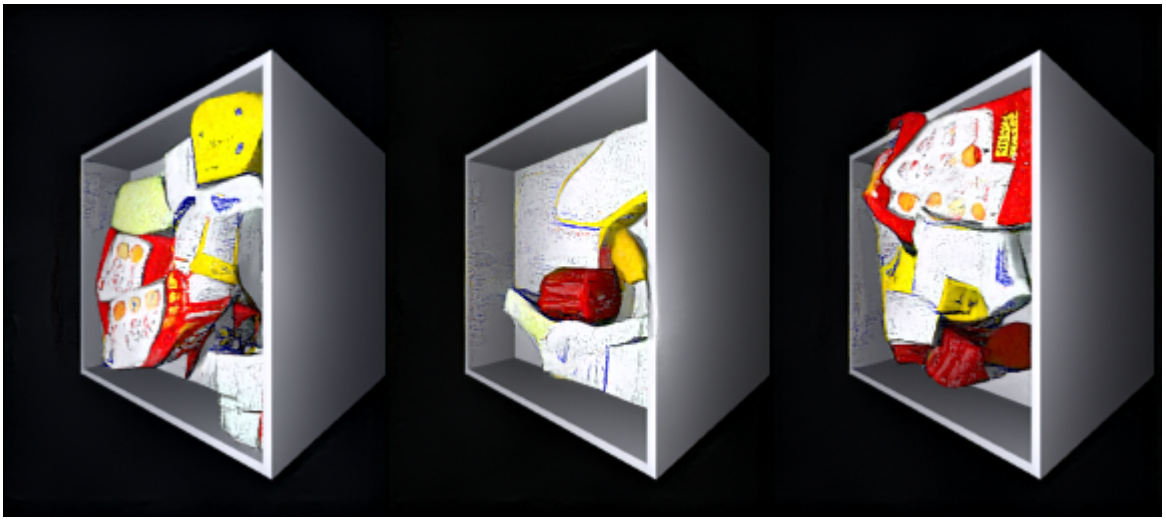
1	2	3	4	5
◯	◯	◯	◯	◯

18. Rate the clarity of these image between 1-5, (please refer to ratings criteria in the description above)

Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

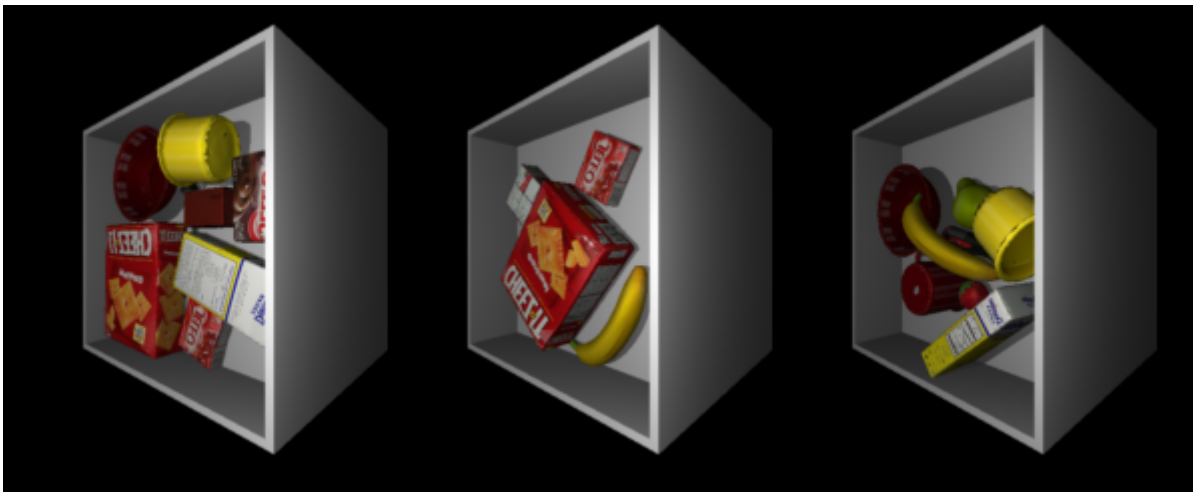
19. Rate the clarity of these image between 1-5, (please refer to ratings criteria in the description above)



Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

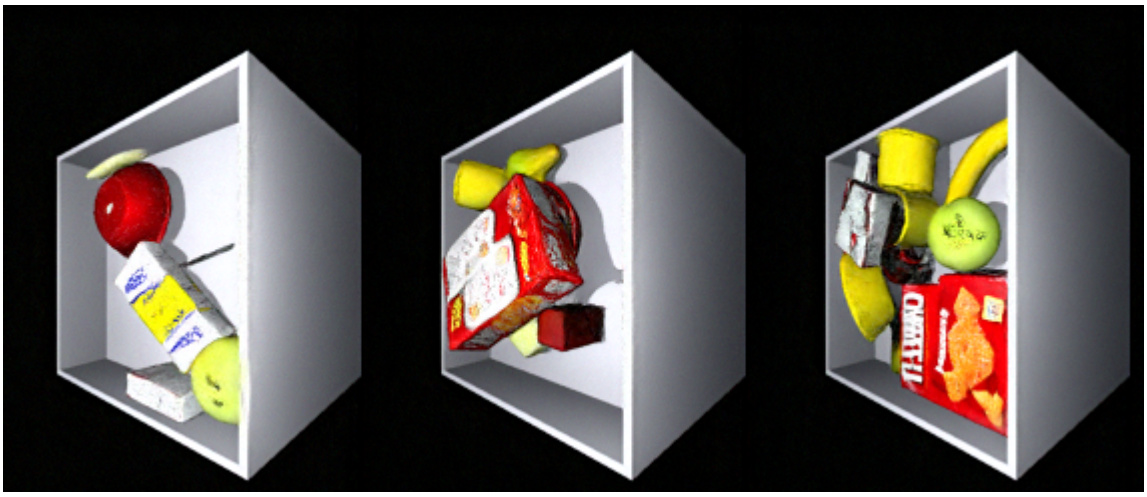
20. Rate the clarity of these image between 1-5, (please refer to ratings criteria in the description above)



Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

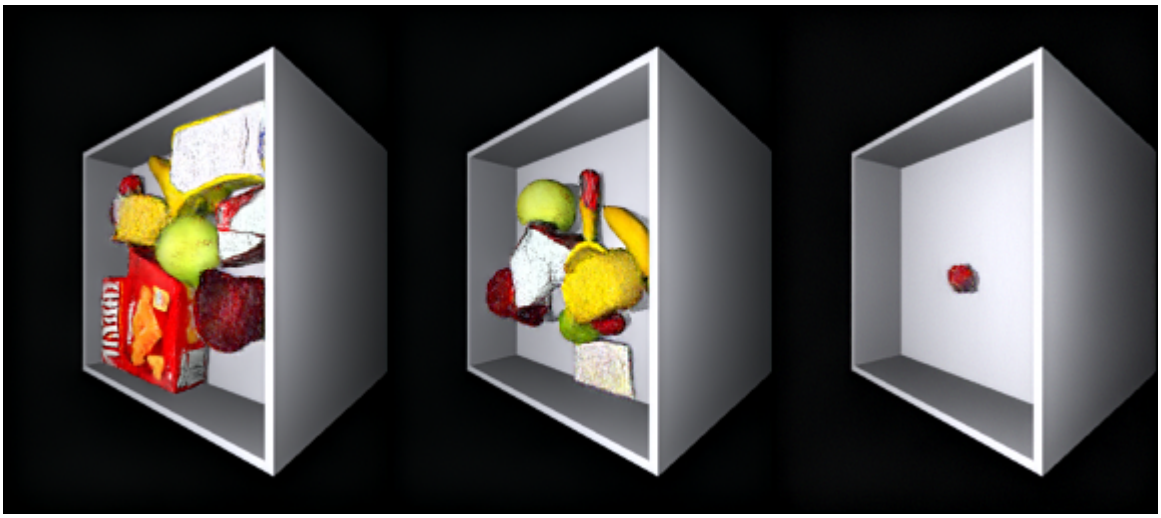
21. Rate the clarity of these image between 1-5, (please refer to ratings criteria in the description above)



Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

22. Rate the clarity of these image between 1-5, (please refer to ratings criteria in the description above)



Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯