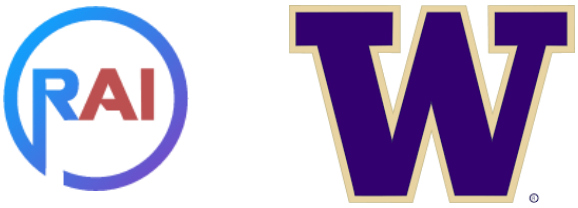
Photo Realistic Synthetic Image Generation



Tejoram Vivekanandan

Industry Mentor: Daniel King

Faculty Mentor: Prof Andrea Fanelli

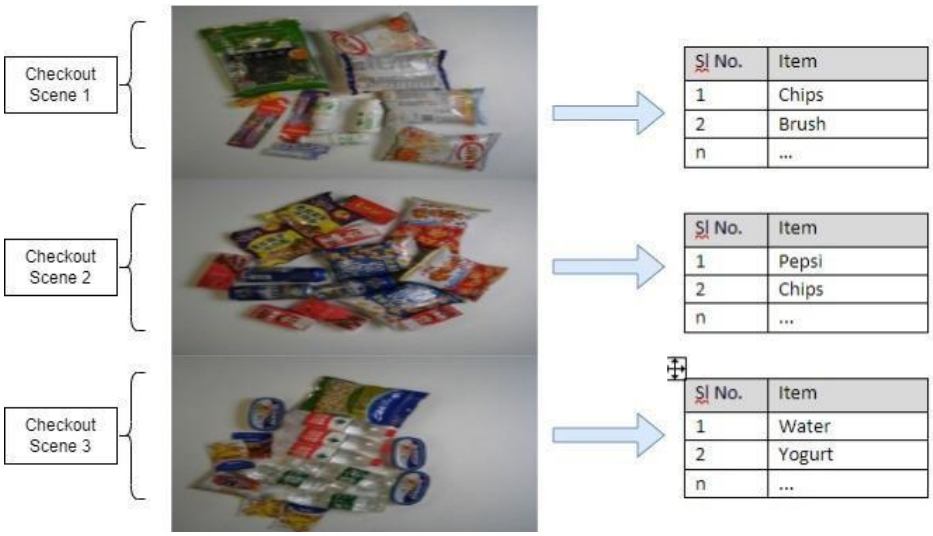
TABLE OF CONTENTS

ABSTRACT

Over the years, there has been growing interest in incorporating computer vision into the retail industry. One crucial challenge in this field is the **Automatic Checkout (ACO)** problem, which involves creating a shopping list from images of products that have been purchased. The main challenge of this problem comes from the **seasonal** and **large scale and the fine-grained nature** of the product categories as well as the difficulty for collecting training images that reflect the realistic checkout scenarios due to continuous update of the products. The **different orientations** of the objects in the checkout scene also add to the complexity of the problem.

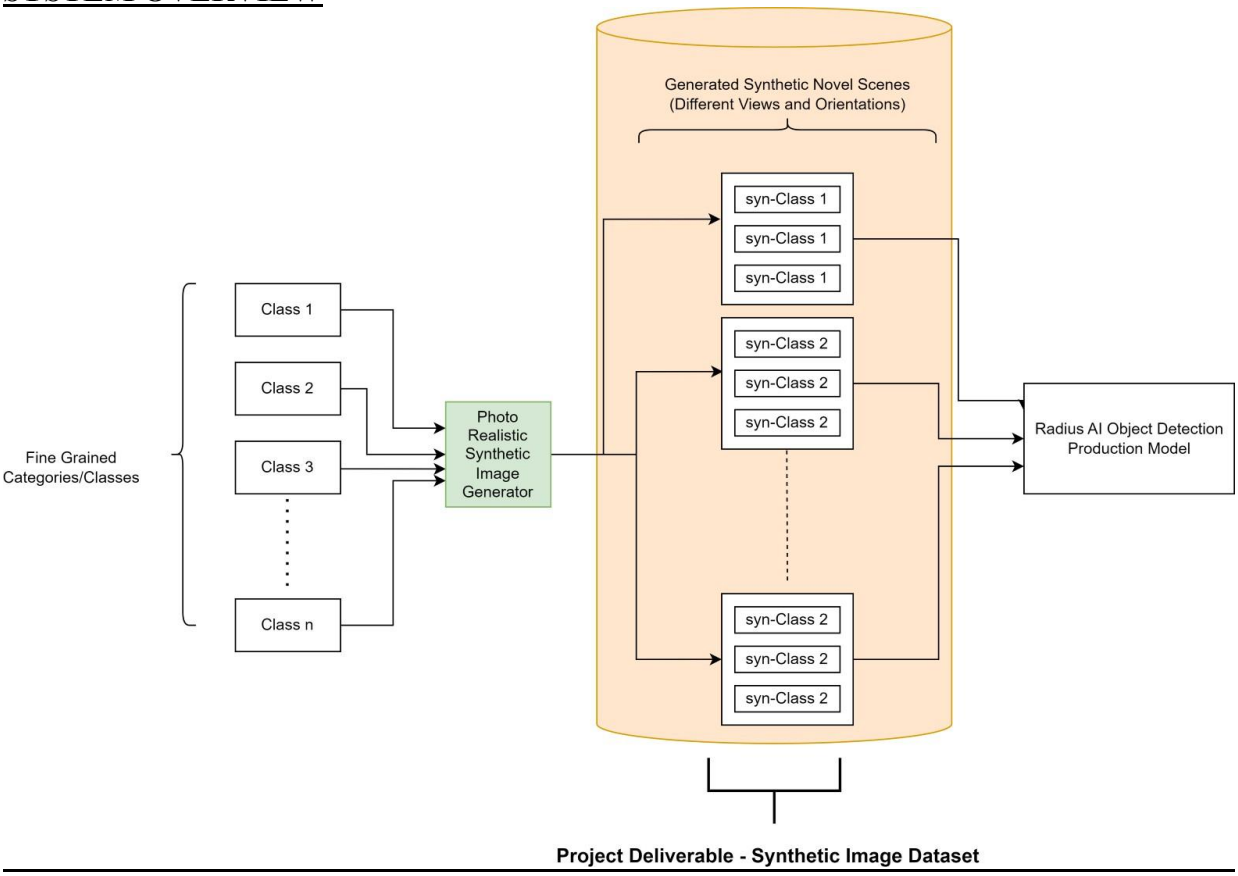
Despite its practical and research significance, this problem has not been thoroughly explored in the computer vision community, primarily due to the absence of a comprehensive data set. Through this project, we attempt to model a **photorealistic synthetic image generation architecture** pipeline that can augment the base dataset to improve the Radius AI production classifier model. This improved dataset can then also be used to solve the automatic checkout problem, whilst giving a higher performance on the Radius AI object detection models.

The augmented dataset that we curate will be able to generate novel images in the scene, and heuristically is bound to improve the accuracy of the Radius AI object detection model.



Retail Checkout Scene -> Automatic Bill Checkout

SYSTEM OVERVIEW



The system overview consists of feeding the photo-realistic synthetic image generator with fine-grained classes to generate novel views/synthetic images to augment the base dataset and further finetuning the model to improve the accuracy of the object detection model.

Fine-grained classes refer to highly specific categories within a broader class. For example, within the broad category of soda cans, fine-grained classes might include species like zenify, coke, or fanta. By generating synthetic images of these specific instances, the model can be trained to recognize and classify them more accurately.

We propose to use advanced algorithms – like GANs, NeRFs, Diffusion Models to create images that are designed to look as realistic as possible. These images are based on the characteristics of the fine-grained classes that the system is trying to generate and can be customized to meet specific needs. Once the synthetic images are generated, they can be added to the existing dataset and used to fine-tune the machine learning model. Fine-tuning involves adjusting the model's parameters based on the new data to improve its accuracy. By using synthetic images to augment the dataset, the model can be trained on a larger and more diverse set of examples, which can help it perform better on real-world data

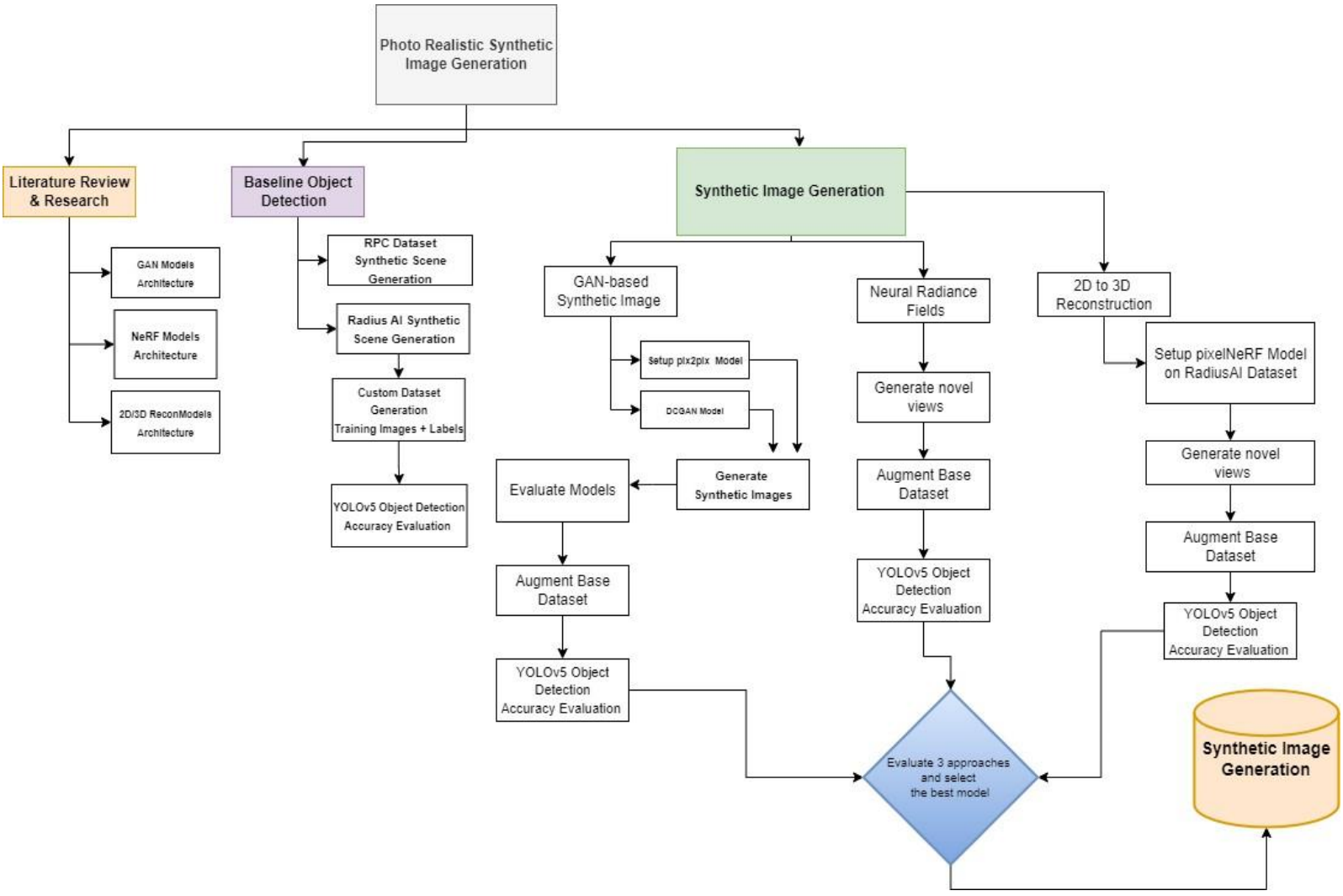
OBJECTIVES

A. Investigation, research, and development of state- of-the art methods for photo-realistic synthetic image generation.
B. Compare different model accuracies to produce the best synthetic image generation architecture
C. Investigate the generalizability to variety of target domains

DELIVERABLES

A. Develop a baseline end-to-end object detection pipeline to test the accuracy of the base dataset and the augmented dataset
B. GAN Model Deliverable: Utilize Generative Adversarial Networks and variants thereof to generate images to augment the base dataset
C. NeRF Model: Train, Test and deploy Neural Radiance Fields to generate novel views for a given scene/ objects in a scene - augmenting the base dataset
D. Utilize PixelNeRFs, training each type of objects to generate 3D images with few image – augmenting the base dataset
E. Evaluation of B, C, D Models, and curating Photo-Realistic Synthetic Image Dataset
F. Object Detection Model Accuracy and Final Report

WORK BREAKDOWN STRUCTURE



REALISTIC CONSTRAINTS/ENGINEERING STANDARDS

The constraints of this project are as follows:

1. **Accuracy**: In a photo-realistic synesthetic image generation project, the accuracy of the generated images is of utmost importance. Therefore, it is necessary to evaluate the quality of the generated images thoroughly.
2. **Validation and Verification**: The models used in the project should be validated and verified to ensure that they are functioning correctly. Validation and verification help to identify any errors or issues in the system, which can then be addressed.
3. **User Interface**: The user interface should be designed to provide a smooth and intuitive experience for the users. It should be able to handle the input data effectively and display the output results accurately.
4. **Interoperability**: The project should be designed in such a way that it can be integrated with other systems easily. This will allow for the exchange of data and information between different systems.

SYSTEM REQUIREMENTS

L1. The end-to-end image object detection pipeline should take a set of image and bounding box

information and feed a YOLOv5 object detection model

L1.1 The input image should be in the **RGB format**

L1.2 The bounding box information for each image should be associated with its respective class and bounding box information(x,y,w,h)

L2. Accuracy: The model should accurately detect objects within an image or video.

L2.1 The object detection model should have accuracy above **90%**

L2.2 The model should be optimized for speed, using techniques such as GPU acceleration and parallel processing.

L2.3 The system should produce consistent results, with a low rate of variability in image quality.

L3 Speed: The model should run in real-time, processing images and videos quickly to allow for quick and accurate object detection.

L3.1 The system should be optimized for speed, using techniques such as GPU acceleration and parallel processing.

L4 Scalability: The model should be able to handle a large number of images be able to process them efficiently.

L4.1 The model should scale well across all the images generated from the 3 different dataset augmentation techniques (NeRF's, GAN's)

L5. Robustness: The model should be able to handle diverse and challenging environments, such as images with low resolution or images with objects partially obscured.

L5.1 The model should be able to handle diverse and challenging environments, such as images with low resolution or objects partially obscured.

L6 Adaptability: The model should be able to adapt to new objects and categories over time, allowing it to continuously improve its detection capabilities.

L7 Multi-object detection: The model should be able to detect multiple objects within an image or video simultaneously.

L8 Object Classification: The model should detect objects and classify them into their respective categories.

L9. False positive reduction: The model should be able to minimize false positive detections, reducing the number of incorrect detections.

L10 Performance metrics: The model should be evaluated using performance metrics such as precision, recall, F1 score, and mean average precision (MaP).

L11. Batch processing: The model should be able to process large amounts of data in batch mode, allowing for efficient processing of large datasets.

L12 One of the methods involves generating images using Generative Adversarial Networks

L12.1 Input image should be RGB.

L12.2 The model should need pairs of inputs, real images and masks

L12.3 The model should be able to train and test different classes of objects at one time.

L12.4 The model outputs an image that is similar but not identical to the original image.

L13 One of the methods involves generating images using NeRF

L13.1 Input image should be RGB.

L13.2 The model should need to input images and input object viewing angle.

L13.3 The model should be able to train and test different classes of objects at one time.

L13.4 The model outputs images which are from a different perspective of the input object.

L14 This project attempts to generate 3D images using Pixel NeRF

L14.1 Select a model for 3D reconstruction.

L14.2 Test the model to find out how much losing the training set will crash the model to get the model limit.

L14.3 Predict and generate many images with a small number of images.

SYSTEM SPECIFICATIONS

In order to run our experiments, we used a combination of web-based resources, i.e. Google Colab and two NVIDIA 2080-Ti GPU cards. This machine set up in the capstone laboratory also had 22 GB of RAM. Certain amount of code had to also be carried out on our local machines, most of which were Macbooks with either 16 or 8 GB of RAM.

SOFTWARE DESIGN

Synthetic Scene Generation:

To train an object detection model such as YOLOv5, a dataset containing images with objects of interest and their corresponding bounding box annotations in text files is required. The larger the dataset, the better the model can be trained since it will be exposed to more examples during the training process. However, it is not enough to have a large dataset; it should also be diverse with objects of interest mixed with other objects in various environments, positions, and backgrounds. One way to create such a dataset is by manually taking a lot of photos and annotating them, which is time-consuming but yields high-quality data. Alternatively, an automatic synthetic dataset can be created by randomly scaling, rotating, and adding cropped photos of objects of interest to different backgrounds using a Python script, with corresponding annotations also generated by the script.

Fine-Grained object classes and their respective masks

(i) **(ii)** **(iii)** **(iv)** **(v)**

Image Backgrounds from Different Perspectives

Using an automated process to create a dataset is significantly faster compared to a manual process. For instance, generating 1000 synthetic images and their corresponding annotations can be completed in under an hour, whereas taking 1000 diverse photos and manually annotating them would require much more time.

Creating a retail product dataset manually would require a significant amount of time and resources, as it would involve taking pictures of various products in various locations, with diverse backgrounds, and under different lighting conditions, and then manually annotating each image with the corresponding product name and bounding box. This manual process can be time-consuming, expensive, and may not yield a sufficiently diverse and representative dataset.

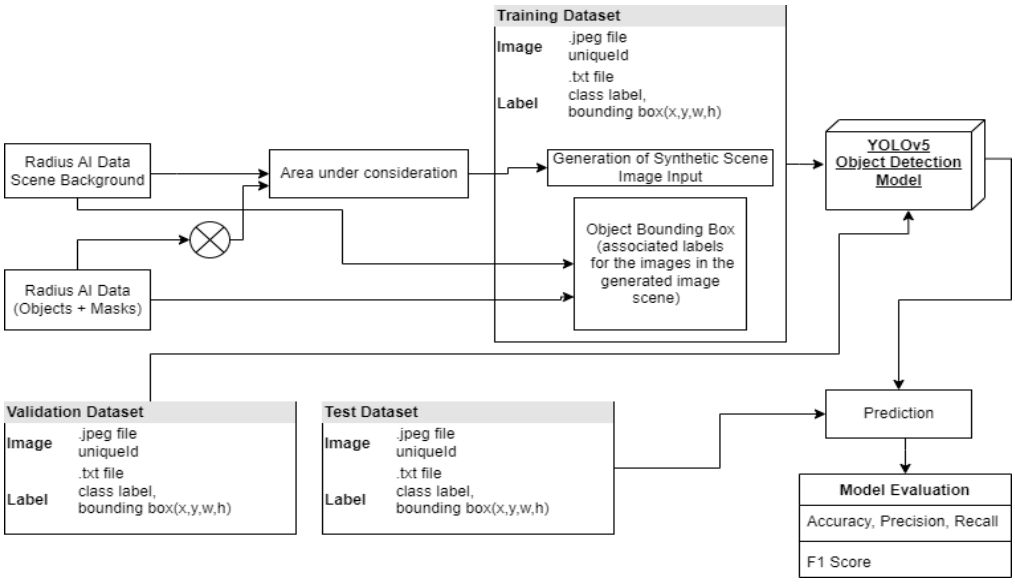
On the other hand, an automated process can be more efficient for a retail product dataset. This is because it allows for the quick and easy generation of large volumes of product images and their corresponding annotations, without the need for costly and time-consuming manual effort. Automated generation of synthetic product images can also help to ensure that the dataset is sufficiently diverse and representative, by varying product placement, lighting conditions, backgrounds, and other factors. Furthermore, this approach can be easily scalable, allowing for the generation of large datasets that can be used to train complex machine learning models for a wide range of retail product recognition and detection applications.

Features of the Synthetic Scene Generation Script:

1. **Point Generation:** Given a desired area on the image, here a table, the script generates a point on the image and places the object on the scene.
2. **Overlap Control:** Create datasets with varying levels of complexity: The script checks the degree of overlap between the objects placed on the scene. Here in the base dataset created, the degree of overlap set is 20%.
3. **Number of objects** in the Dataset: The script can also datasets of varying complexities, like the number of objects in the scene. Since the RAI dataset has 15 classes, we will be fixing the number of the objects in the scene to 15.
4. **Generate normalized YOLOv5 annotations** for all the objects in the scene.

<object-class-ID> <X center> <Y center> <Box width> <Box height>

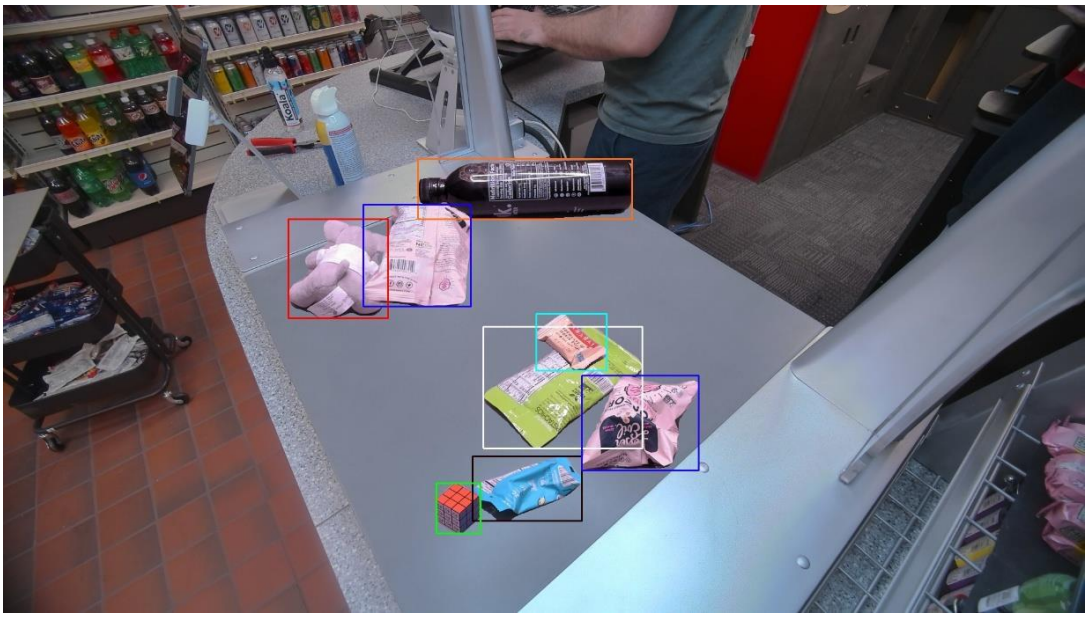
5. *[WIP]* Enhance the scale factor of the images placed in the scene to enhance the photorealism of the scene



UML Sketch of the Synthetic Scene Generation Algorithm + Detection Pipeline

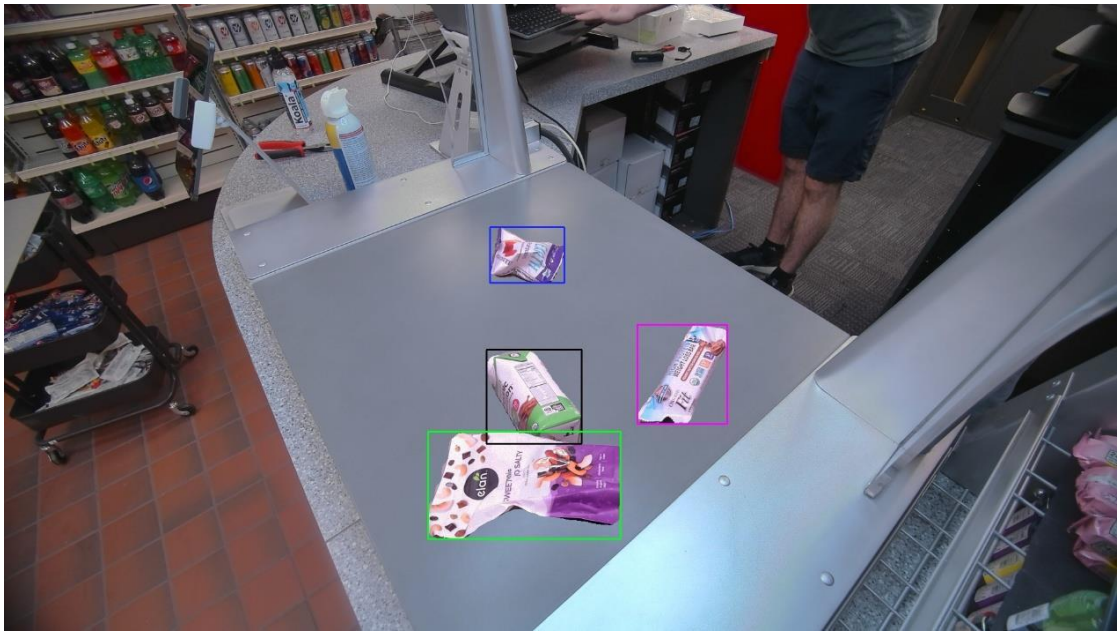
Scene Visualization with corresponding Annotations:

(i) Synthetic Scene Generation 1



(ii) Synthetic Scene with corresponding YOLOv5 Annotations

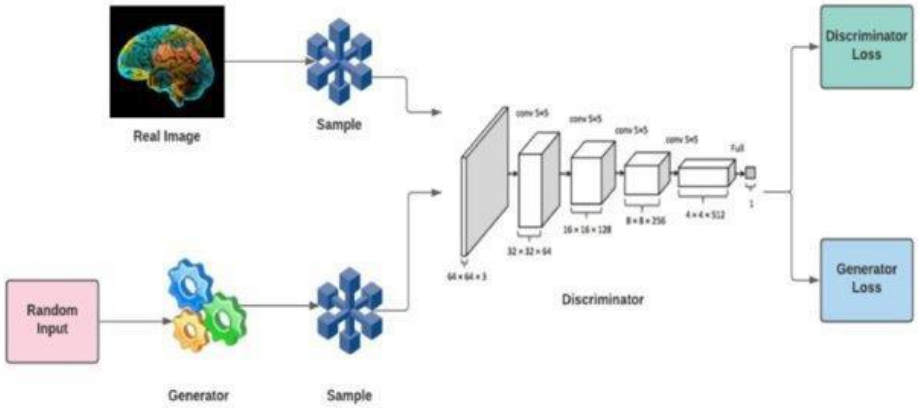
(i) Synthetic Scene Generation 2



(ii) Synthetic Scene with corresponding YOLOv5 Annotations

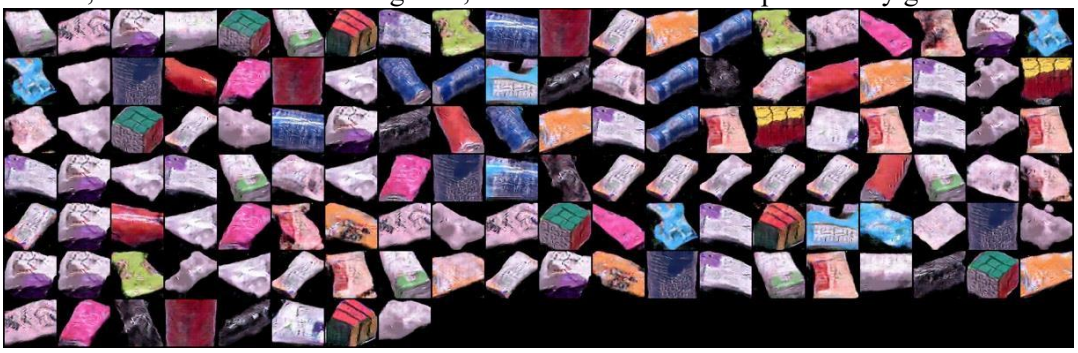
Image Generation (DCGAN):

- DCGANS are used to generate high-quality images by leveraging convolutional layers in both generator and discriminator networks. Therefore, we try to generate more realistic images by learning the underlying distribution of the input data using the dc GAN model.
- The GAN is divided into two parts, it is composed of two networks called discriminator and one called generator, which then together form the GANs network.
- For the generator, the input to the generator is some random matrix. The input to the discriminator is two images, which are the image generated by the generator and the image in the training set. Then the two images are fed in, their differences are discriminated, their losses are calculated, and finally the losses are made as small as possible. By such a mechanism, the discriminator becomes better and finally makes the result better. Because the generated image is as similar as possible to the original image, then it is equivalent to generating a realistic image.

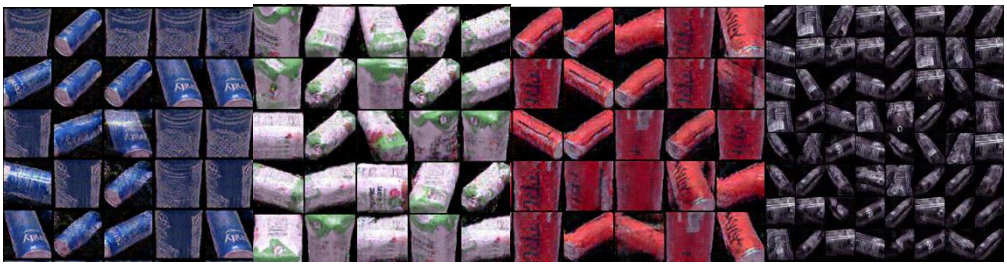


Flow Chart of the Sequence of Operations

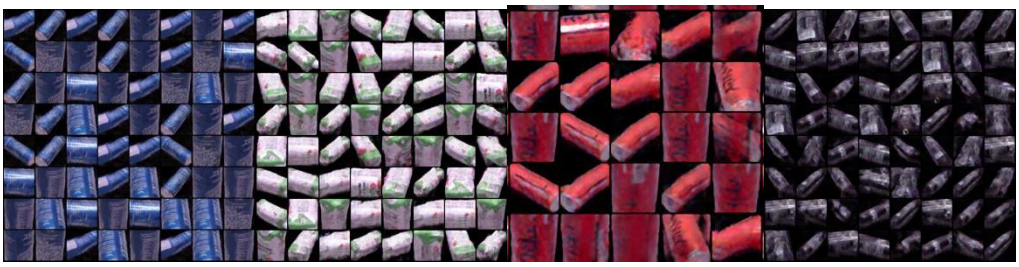
- At first, we trained all the data together, and the results were not particularly good.



- Later, we improved it by classifying the data and training them separately.



- But there is still some noise on the image, we used the median filter to remove the noise on the image, so that the results are easier to identify and verify.



- Finally, we modified the model of DCGANs: increased mask input to enhanced learning, and we also changed the generator, using the u-net architecture to make the generated images get high-frequency features, it makes the results clearer.

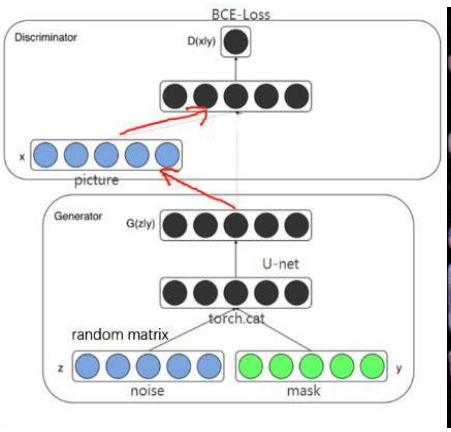
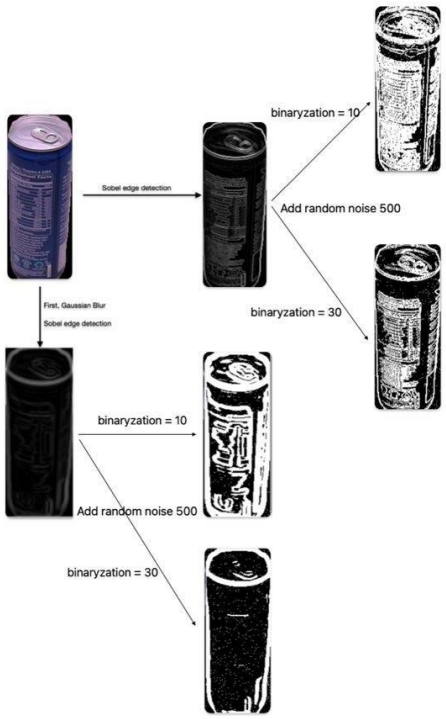


Image Generation (Pixel2Pixel GAN):

The original datasets are not large enough and increasing the diversity of datasets may potentially improve our accuracy of classification. Thus, we try to use pixel2pixel GAN to generate more diverse synthetic images using pixel2Pixel GAN. Pixel2PixelGAN requires a pair of input which is one real image and one mask.

Preprocess on masks:

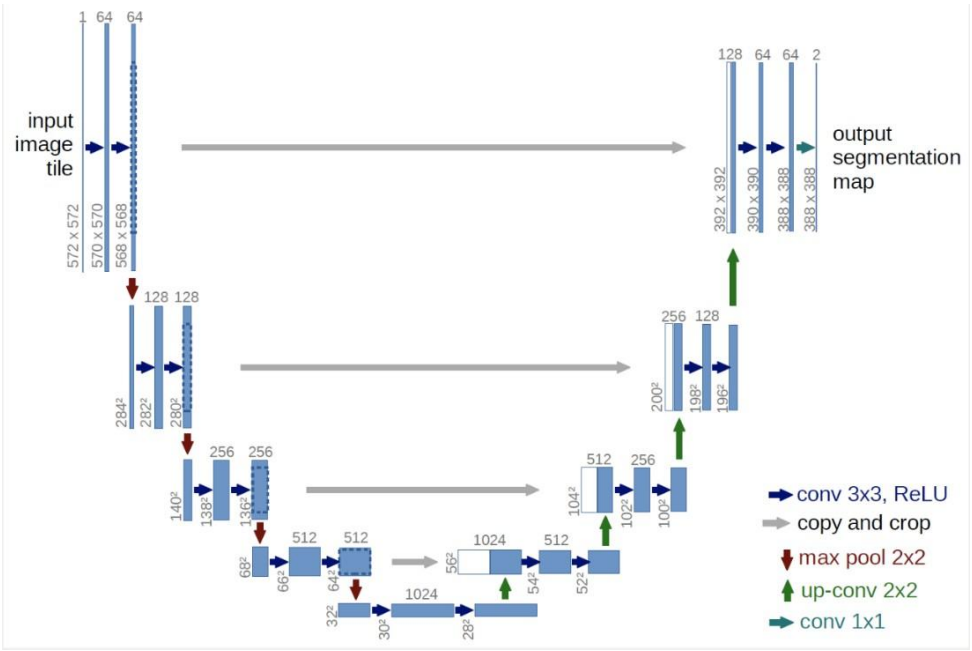
1. Use Sobel Edge Detector to generate masks with mode detailed information.
2. Use Gaussian blurring to remove noise and extract main features.
3. Add 500 random noises to increase diversity.



Process for Masks

Generator of pixel2pixelGAN

The U-Net architecture is designed to perform pixel-wise segmentation of images, where each pixel is assigned, a label based on the class it belongs to. To achieve this, U-Net uses skip connections that connect corresponding layers in the contracting and expansive paths, which help preserve spatial information and reduce the loss of information during the down-sampling and up-sampling processes. U-Net has been shown to be highly effective for a wide range of image segmentation tasks in various domains, such as medical image analysis, satellite image segmentation, and cell segmentation.



U-Net Architecture

YOLOv5 Architecture

The architecture of YOLOv5 is based on a deep convolutional neural network (CNN) that is designed to detect objects in images. The network consists of a backbone architecture, a neck architecture, and a head architecture.

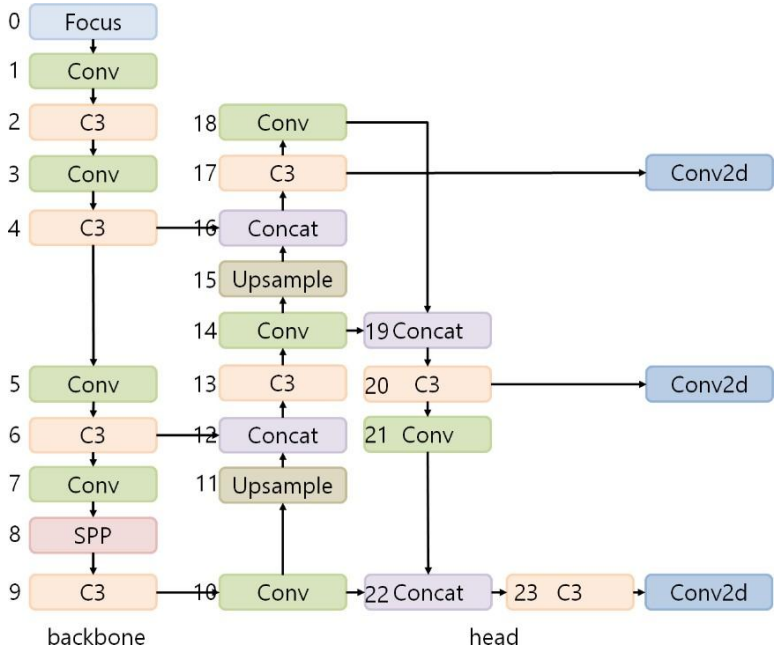
The backbone architecture is responsible for extracting features from the input image. YOLOv5 uses a variant of the *Efficient-Net* architecture called CSP (cross stage partial connections) to serve as the backbone. The CSP architecture utilizes a partial connection between the input and output of each stage, allowing for efficient feature reuse across different scales.

The neck architecture is responsible for combining the features extracted by the backbone and enhancing them further. YOLOv5 uses a spatial pyramid pooling (SPP) module, which pools features at different scales to capture multi-scale information.

The head architecture is responsible for making predictions based on the extracted features. YOLOv5 uses a set of fully connected layers followed by convolutional layers to predict bounding boxes and class

probabilities. The predictions are made at three different scales, allowing for detection of objects at assorted sizes and resolutions.

YOLOv5 also uses anchor boxes, which are pre-defined shapes used to predict the bounding boxes of objects. This allows the network to detect objects of different shapes and sizes.

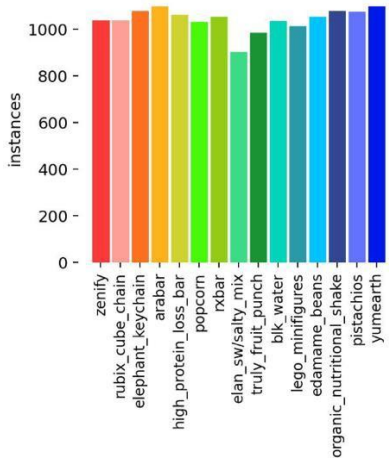


Architecture of YOLOv5

SOFTWARE IMPLEMENTATION

The base synthetic training dataset was created with 2000 training images, and a validation dataset was defined with 500 images.

The number of instances of the classes in the training dataset are displayed in the following figure:



The **YOLO5s** model was chosen as the backbone for the training for the ease of training and the hyperparameters were chosen as the following based on the initial tests performed.

1. **Number of epochs**: 100
2. **Batch Size**: 32
3. **Optimizer**: SGD (lr = 0.01)

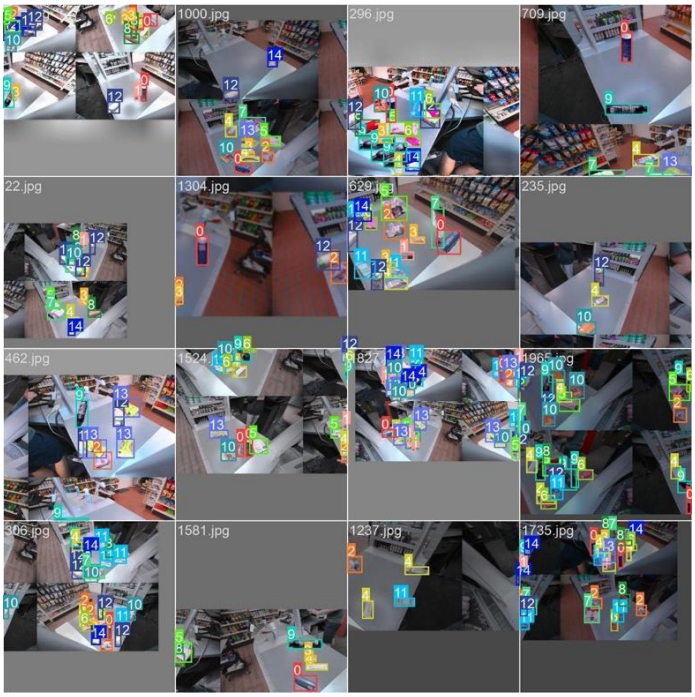


Fig. Training Batch001

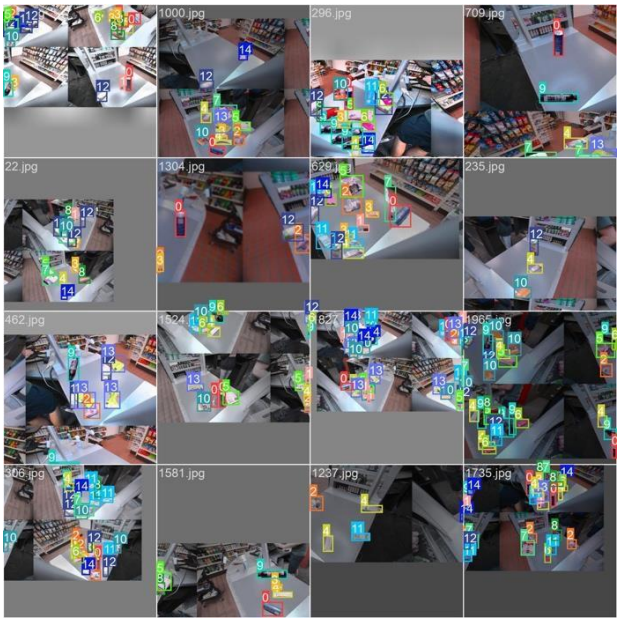
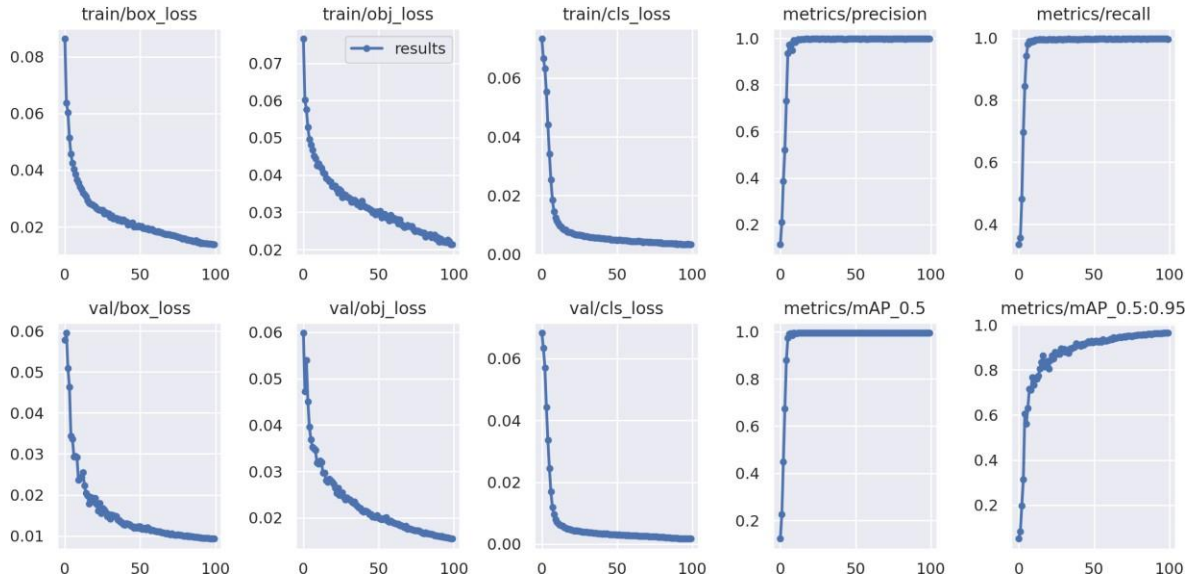


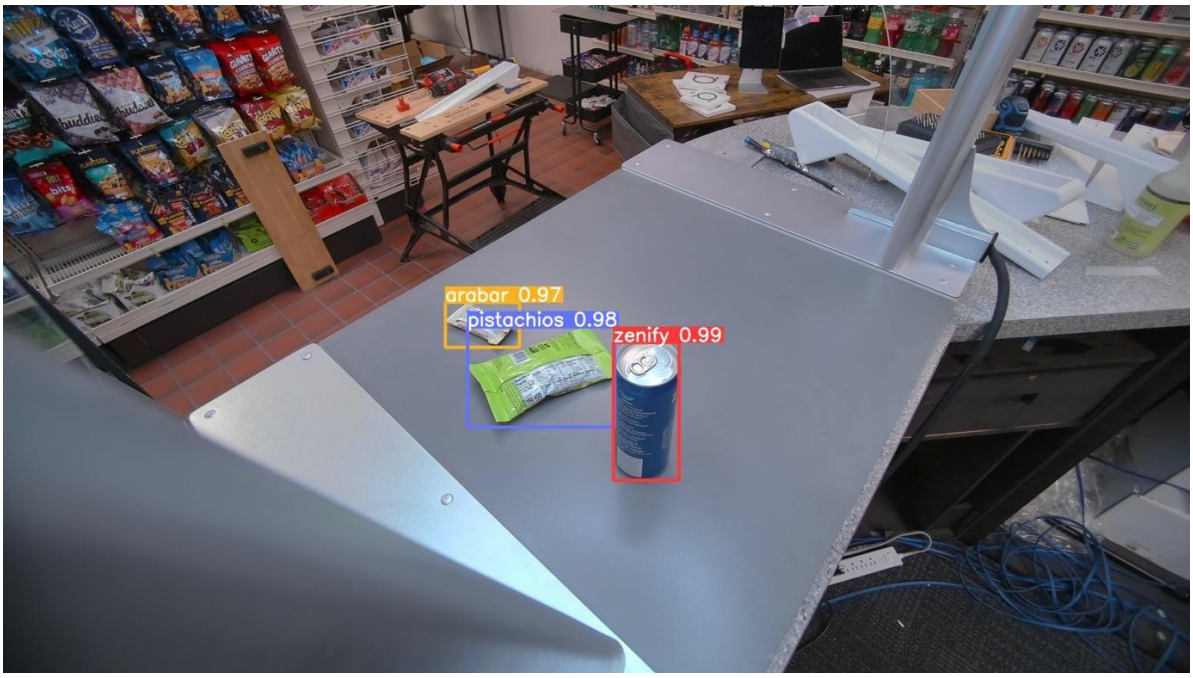
Fig. Training Batch002

Training/Validation Loss Curves *[nepochs = 100]*

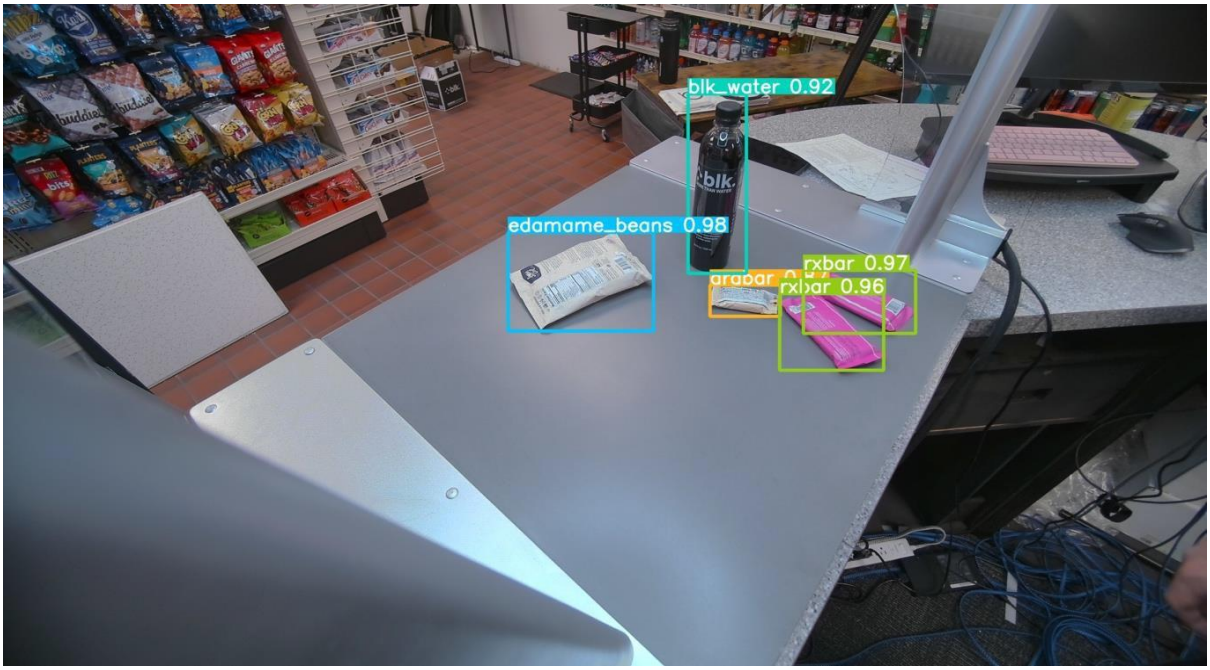


Inference on Real Data

QUALITATIVE RESULTS




organic_nutritional_shake 0.98
edamame_beans 0.97


blk_water 0.92
edamame_beans 0.98
rxbar 0.97
rxbar 0.96

QUANTITATIVE RESULTS

Test Dataset Results:

Class	Images	Instances	P	R	mAP50	mAP50-95
all	150	495	0.906	0.775	0.873	0.768
zenify	150	29	0.732	1	0.969	0.854
rubix_cube_chain	150	33	0.993	1	0.995	0.804
elephant_keychain	150	26	1	0	0.265	0.198
arabar	150	42	1	0.637	0.894	0.76
high_protein_loss_bar	150	28	0.868	0.471	0.788	0.711
popcorn	150	42	0.934	0.338	0.768	0.679
rxbar	150	42	0.994	1	0.995	0.918
elan_sw/salty_mix	150	18	1	0.894	0.995	0.872
truly_fruit_punch	150	36	0.96	0.944	0.94	0.813
blk_water	150	29	1	0.549	0.899	0.739
lego_minifigures	150	51	0.978	1	0.993	0.939
edamame_beans	150	33	0.338	1	0.703	0.635
organic_nutritional_shake	150	43	1	1	0.995	0.894
pistachios	150	9	0.797	1	0.995	0.934
yumearth	150	34	0.991	0.794	0.907	0.763

Image Generation (Neural Radiance Fields)

Neural Radiance Fields (NeRF) is a recent technique for synthesizing photo-realistic 3D models of scenes from 2D images using a deep neural network (multi-layer perceptron). The key idea is to learn a continuous representation of the scene's 3D geometry and appearance from a collection of images, which can then be used to generate novel views of the scene from any viewpoint.

1. Scene Representation

The scene is represented as a continuous 3D function that maps a 3D point in space to a radiance value, which describes the scene's color and brightness (density) at that point. This function is represented by a neural network, which takes a 3D point as input and outputs the corresponding radiance value.

Formally, let $R(x)$ denote the radiance value at point x in 3D space, and let $f(x;\ \theta)$denote the neural network function that predicts this radiance value. Then, we have, $R(x) = f(x;\ \theta)$

The neural network $f(x;\ \theta)$ considered in NeRFs is a multi-layer perceptron (MLP).

2. Training Data

To train the neural network function $f(x;\ \theta)$, we need training data consisting of images and corresponding camera poses. Specifically, let I_i denote the i^{th} image in the dataset, and let C_i denote the corresponding camera pose (i.e., the position and orientation of the camera when the image was taken). Then, the training data consists of pairs $(I_i,\ C_i)$ for $i = 1,\ \dots,\ N$.

1. Loss Function

The goal of the neural network function $f(x;\ \theta)$ is to predict the radiance value $R(x)$ at any point x in 3D space, given the training data. To achieve this, we need to optimize the parameters θ of the neural network to minimize the difference between the predicted radiance value and the ground truth radiance value.

Formally, let $R_i(x)$ denote the ground truth radiance value at point x in the i^{th} image and $p_i(x)$ denote the predicted radiance value at point x in the i^{th} image. Assuming the image has dimensions $m * n$, we define the loss function $L(\theta)$ as follows:

$$L(\theta) = \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} \, ||R_i(x_j) - p_i(x_j;\ \theta)||^2$$

where w_{ij} is a weight that depends on the distance between the camera pose C_i and the 3D point x_j in the scene. The weight is given by:

$$w_{ij} = \frac{1}{\sigma^2} * \exp\left(-\left(\frac{d_{ij}}{\sigma}\right)^2\right)$$

where σ is a constant that controls the size of the weight function and d_{ij} is the Euclidean distance between the camera pose C_i and the 3D point x_j. While this is a generalized loss function, the one we have considered in our implementation is a simple $L2$loss between the synthesized image and the ground truth image. The loss function $L(\theta)$ is optimized using stochastic gradient descent (SGD) to find the optimal parameters θ.

2. Rendering

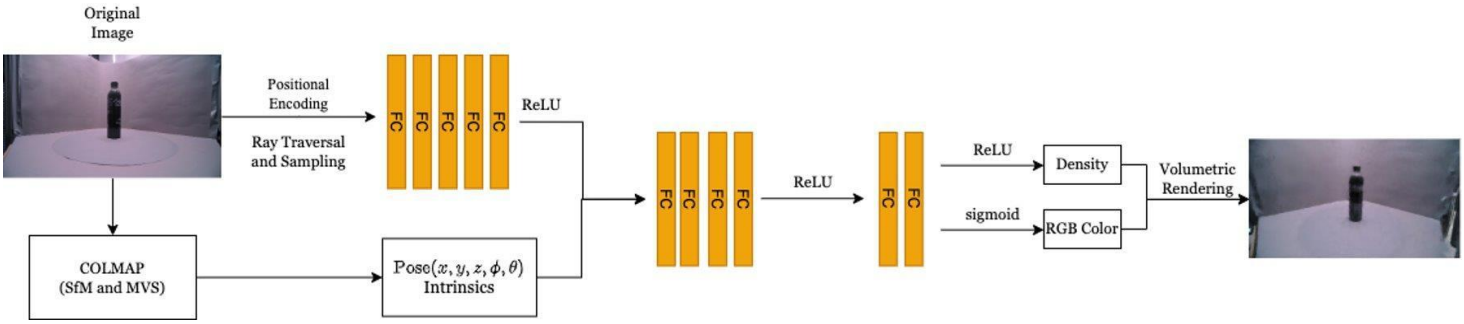
Once the neural network function $f(x;\ \theta)$ has been trained, we can use it to render novel views of the scene from any viewpoint. To do this, we first sample a set of rays that pass through the image plane at the desired viewpoint. For each ray, we use ray marching to find the 3D point where the ray intersects the scene geometry. Ray marching is a technique used to trace the path of a ray of light as it passes through a medium or interacts with an object in a scene. We then use the neural network function $f(x;\ \theta)$ to compute the radiance value at that point and use it to color the pixel corresponding to the ray in the output image.

Let P denote the output image, and let p_i denote the i^{th} pixel in P. Let $ray(p_i)$ denote the ray that passes through pixel p_i in the image plane at the desired viewpoint. Then, for each ray $ray(p_i)$, we compute the near and far bounds for the scene - t_n and t_f. Employing a uniform binning strategy between these bounds, we get the 3D points x_i, $i \in uniform(t_n,\ t_f)$ for the scene geometry and use the neural network to compute the corresponding radiance value $R(x_i)$. This radiance value is then used to color the pixel p_i in the corresponding output image.

3. Implementation

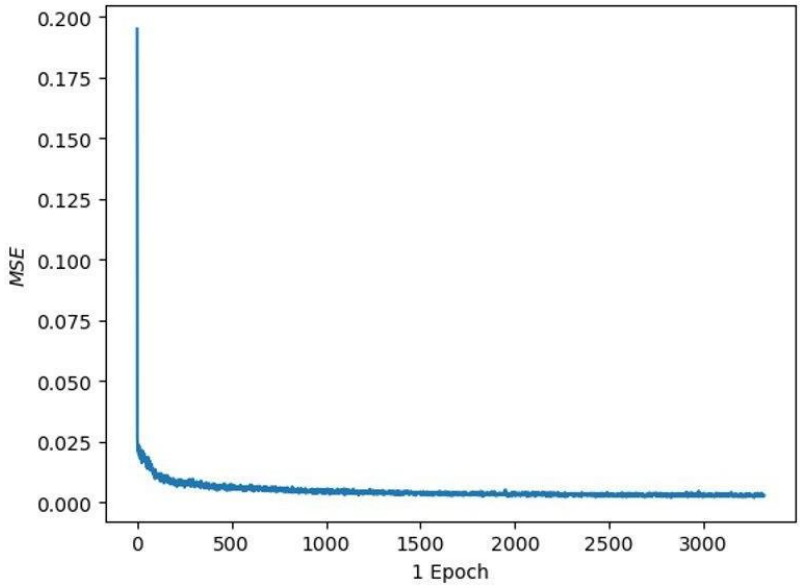
To implement Neural Radiance Fields, we have used PyTorch. The basic steps for implementing NeRF are as follows:

Step 1: Define the neural network architecture for $f(x;\ \theta)$, which takes a 3D point and camera orientation along with the camera intrinsics as input and outputs the corresponding radiance value. In our case, we have used a MLP. Given below is the architecture implemented.

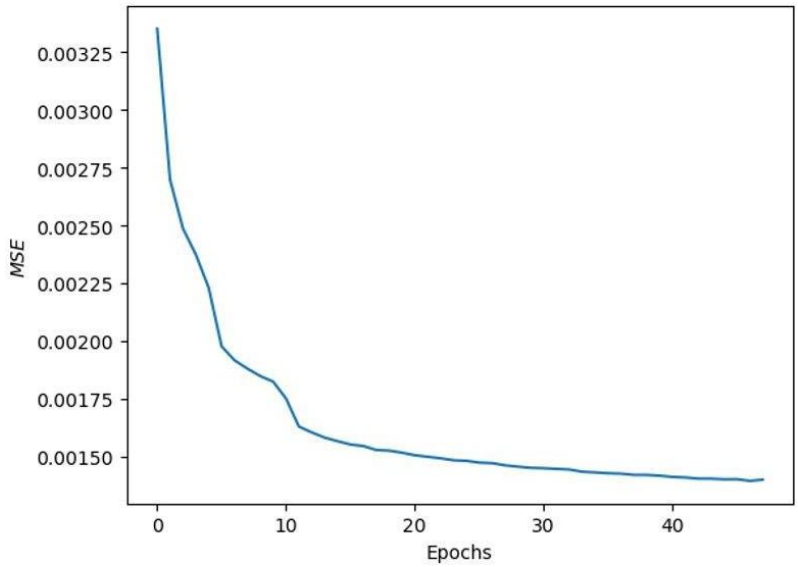


Step 2: Prepare the training data by collecting a set of images and corresponding camera poses. For this, we used the images sent to us by the RadiusAI team and manually had to estimate the camera parameters using Structure-from-motion and Multi-View-Stereo consistency from COLMAP.

Step 3: Train the network. We had conducted training on 136 Images of size $400 * 400$ pixels, with a batch size of 1024 pixels, over fifty epochs. When trained on a single GPU, it took around 10 minutes to train a single epoch. Given below is the training loss measured every epoch. For the first epoch, we make the NeRF model focus on the central portion of the image, just to warmup the neural network and get a good reconstruction of the important features present in the scene. The first training loss curve decreases drastically and then there is a gradual decrease with every epoch. This ensures that the network is training well. We had tested this on 34 images and obtained a PSNR of 28.95 and 25.94 db respectively for the training and testing data.



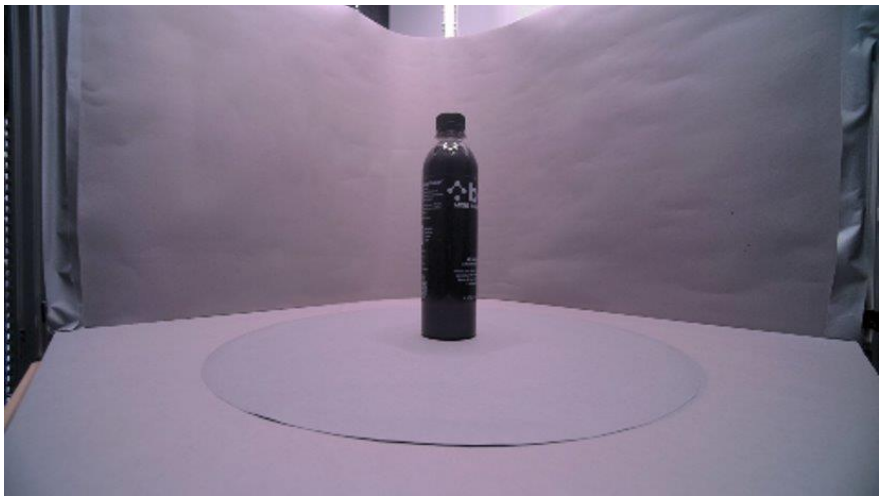
Initial Loss Curve



Training loss curve for the remaining epochs

Step 4: To render novel views of the scene, sample a set of rays that pass through the image plane at the desired viewpoint, define the bounds for the scene, perform uniform sampling to find the 3D points of interest in the scene geometry, and use the neural network function $f(x;\ \theta)$ to compute the corresponding radiance values.

Step 5: Color the pixels in the output image using the radiance values computed in Step 4. The figure below shows a resulting image from the network, before and after reconstruction.



Input ground truth image

Reconstructed Image

We also use positional encoding to improve the generated image's quality. In the context of Neural Radiance Fields, positional encoding is used to provide the neural network with further information about the 3D position of a point in space.

The basic idea of positional encoding is to add an additional input to the neural network that encodes the position of the input point. The approach considered here is to use a sinusoidal function of different frequencies and phases to encode the position along each dimension of the input space. The resulting encoding is added to the input features and passed through the neural network. When the input to the neural network is a 3D point $(x,\ y,\ z)$, we compute the positional encoding for each dimension. Each dimension of the input is encoded by a pair of values, a sine and a cosine term. The frequency of each encoding function can be defined as:

$p(x) = (\sin(\frac{2^0\pi x}{p_{max}}), \cos(\frac{2^0\pi x}{p_{max}}), \sin(\frac{2^1\pi x}{p_{max}}), \cos(\frac{2^1\pi x}{p_{max}}), \ldots, \sin(\frac{2^{L-1}\pi x}{p_{max}}), \cos(\frac{2^{L-1}\pi x}{p_{max}}))$ where i ranges from 0 to $L-1$. This means that the frequency of the encoding functions increases exponentially with i, with the first frequency being $f_0 = \frac{1}{p_{max}}$. In our experiments, we consider $L = 10$ and $p_{max} = \max_{i,j} \|x_{i,j}\|$, where $x_{i,j}$ is the 3D location of a point in the scene, represented as a vector. The final positional encoding is the concatenation of all the individual encoding vectors $p(x)$ for each point in the input set. These are then added to the input features of the neural network before passing them through the network. Positional encoding allows the neural network to distinguish between points that are close together in space but have different radiance values and is an essential component of the NeRF algorithm.

Image Generation (Pixel NeRF):

Pixel NeRF (Neural Radiance Fields) is a deep learning approach used for synthesizing 3D scenes from 2D images. It is an extension of the original NeRF algorithm that allows for high-quality rendering of complex scenes using only a single image as input.

The basic idea behind Pixel NeRF is to train a neural network to predict the radiance (or color and brightness) of each pixel in an image based on the 3D geometry and appearance of the scene. To do this, the network is trained on a dataset of images along with their corresponding 3D scene information, which includes camera poses and scene geometry.

During training, the network learns to predict the radiance of each pixel by estimating the underlying 3D scene geometry and appearance that would generate that pixel's color and brightness. Once trained, the network can be used to render new views of the scene from any camera pose by simply querying the network for the radiance of each pixel in the new image.

Pixel NeRF has demonstrated impressive results in generating photorealistic 3D scenes from single images, and has potential applications in fields such as virtual reality, gaming, and cinematography.

Pixel NeRF output for D2V dataset

Pixel NeRF output for Radius AI dataset

One approach to motion blur deblurring is to use deep learning techniques, such as Generative Adversarial Networks (GANs). A GAN is a type of neural network that consists of two components: a generator and a discriminator. The generator learns to produce realistic images, while the discriminator learns to distinguish between real and generated images.

To train a GAN for motion blur deblurring, a dataset of motion blurred images and their corresponding sharp images is required. The generator takes a motion-blurred image as input and generates a sharp image. The discriminator is then trained to distinguish between the generated sharp images and the ground-truth sharp images.

The generator is trained to produce sharp images that fool the discriminator into thinking they are real. Over time, the generator learns to produce increasingly sharp images, while the discriminator becomes more adept at distinguishing between real and generated images.

Once the GAN is trained, it can be used to deblur motion-blurred images by feeding them into the generator. The generator will then produce a sharp image that is as close as possible to the original sharp image that was blurred by motion.

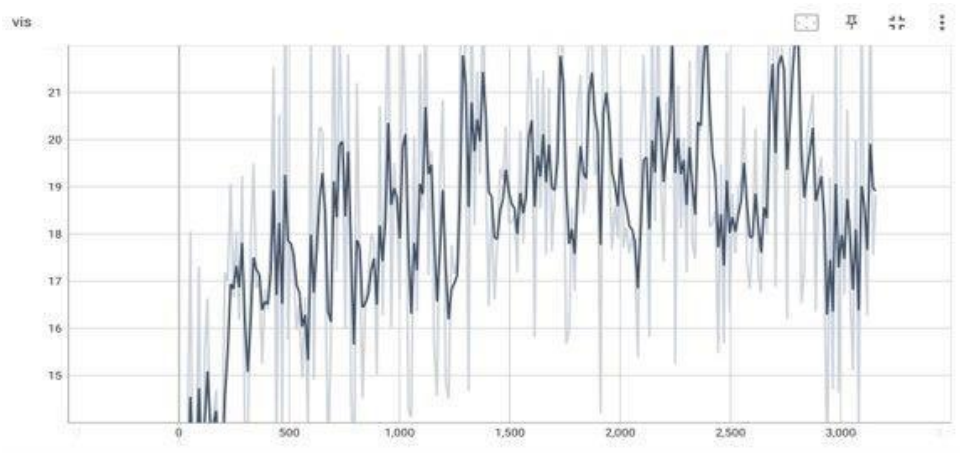
Deblur GAN output for DTU dataset

Model Testing (Pixel NeRF):

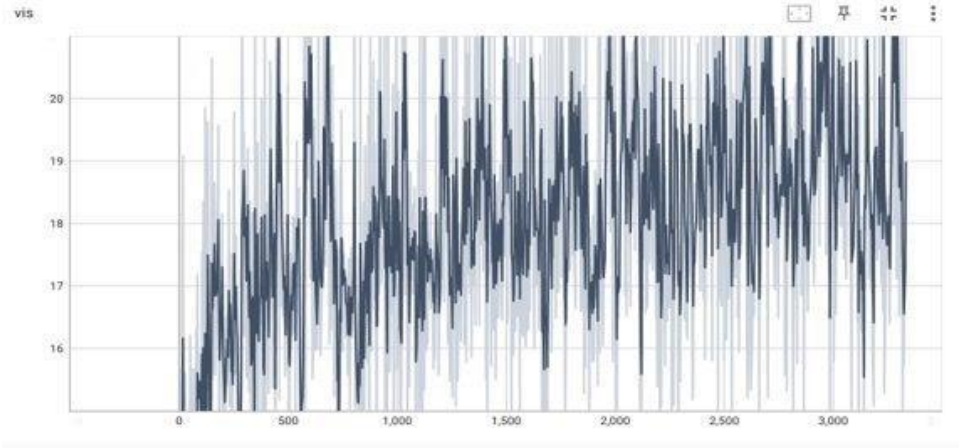
The principle of the Pixel NeRF model has been shown above. And another thing is to find out the minimum requirements for efficient and effective use of resources, since there are not so many data for us to train.

The following results show the model test results PSNR (Ratio used to show the image quality, <20 bad, 20-20 normal, >30 good). And this means that the performance of Pixel NeRF is not good enough when only a few objects are used. But the fluctuation of 50 objects result is less than 10 objects result. So, when the total amount of training set increases, the result will be relatively better.

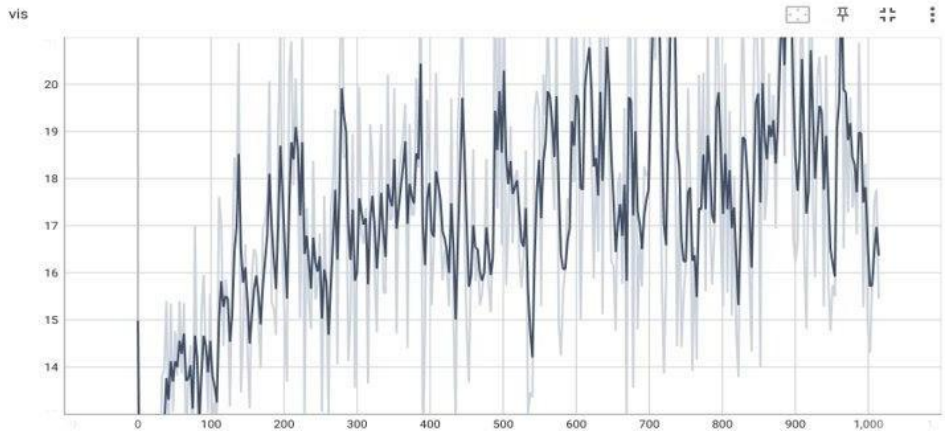
50 objects (15 pictures for each object)



10 objects (15 pictures for each object)



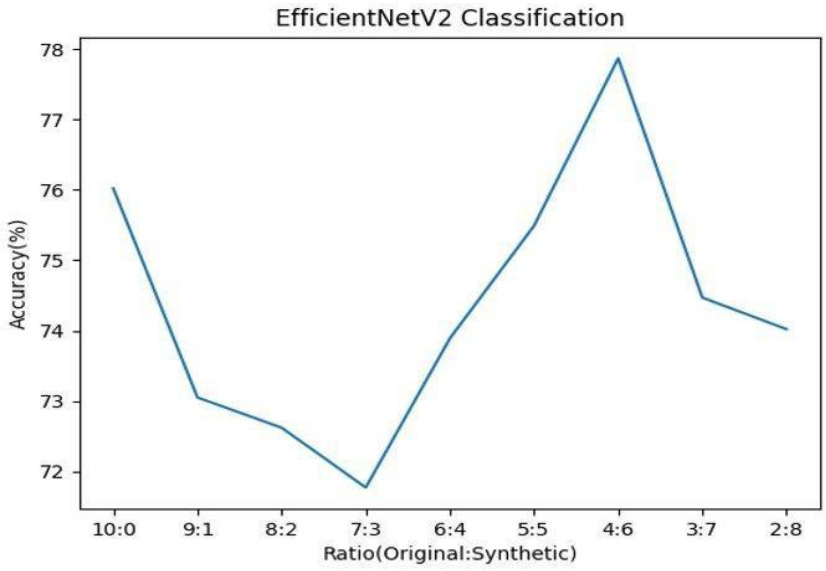
50 objects (10 pictures for each object)



However, there may be other reasons cause the model performs poorly. The next thing is to continue to finish testing Pixel NeRF and try to find out what causes the model to perform poorly in other points.

Image Testing (EfficientNetV2):

EfficientNetV2 is a state-of-the-art deep learning model designed for image classification tasks. It takes the concept of efficient architecture design a step further by incorporating various advancements and improvements. EfficientNetV2 achieves a balance between model size and performance by using a compound scaling method. This approach allows it to achieve better accuracy while maintaining a smaller model size and lower computational requirements compared to other models.



Proportion(original input : synthetic input)	original input	synthetic input	synthetic input(each class)	epochs	accuracy
10 : 0	2652	0	0	57	76.02%
9 : 1	2652	295	20	52	73.05%
8 : 2	2652	663	44	52	72.62%
7 : 3	2652	1137	76	45	71.77%
6 : 4	2652	1768	118	39	73.89%
5 : 5	2652	2652	177	37	75.49%
4 : 6	2652	3978	265	28	77.87%
3 : 7	2652	6188	413	12	74.47%
2 : 8	2652	10608	707	17	74.02%

Original input is the object image cropped from the real image, and synthetic input is the object image generated by the Gan model. Combine these input images with random backgrounds to generate a dataset and put it into EfficientNetV2 for training and classification. Vary the ratio of original and synthetic data to test the model's classification accuracy.

Each ratio was tested twice, and the average value was calculated to ensure the reliability of the results. Adding a small amount of synthetic data makes the model accuracy drop until the ratio is 7:3. Then the accuracy of the model gradually increases as more synthetic data is added. Until the ratio is 4:6 to reach the highest. When the ratio is 3:7 and 2:8, the amount of synthetic data accounts for the majority, and the accuracy rate is similar to that at the beginning. Therefore, when the ratio of original data to synthetic data is 4:6, it is most helpful for model training.

Boundary Generation (Diffusion model):

The diffusion model is a computational framework used in psychology and cognitive science to understand decision-making processes. It provides a mathematical description of how information is accumulated and integrated over time to make a choice between two or more alternatives. The model is based on the concept of diffusion, which refers to the spread or dispersion of particles or molecules from an area of high concentration to an area of low concentration.

In the context of decision-making, the diffusion model assumes that when individuals are faced with a choice, they accumulate evidence in favor of different alternatives until they reach a decision threshold. The evidence accumulation process is conceptualized as particles diffusing through a decision space. Each particle represents a unit of evidence in favor of one of the alternatives, and its movement is influenced by noisy fluctuations and external inputs.

The diffusion model incorporates several key parameters that govern the decision-making process. These include the starting point, which represents an individual's initial bias toward one alternative, the drift rate, which determines how quickly evidence accumulates for each alternative, and the decision threshold, which specifies the amount of evidence required to make a decision.

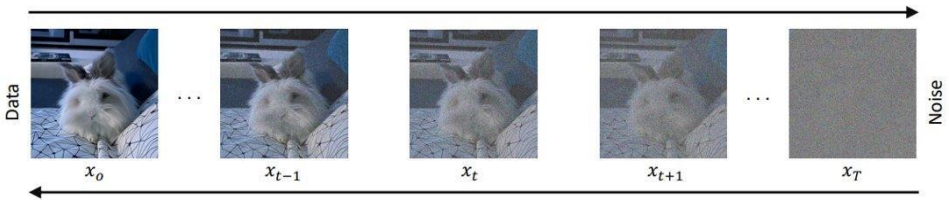
One advantage of the diffusion model is its ability to account for response time data and accuracy. By analyzing the time taken to make a decision, researchers can gain insights into the underlying cognitive processes involved. For example, faster response times may indicate more efficient evidence accumulation, while slower response times may suggest increased uncertainty or difficulty in making a decision.

The diffusion model has been applied to a wide range of decision-making tasks, including perceptual judgments, memory retrieval, and economic choices. It has been used to investigate various phenomena such as response bias, speed-accuracy trade-offs, and the effects of manipulations or interventions on decision-making behavior.

The model has also been extended and modified to capture more complex decision processes. Variations of the diffusion model, such as the drift-diffusion model and the linear ballistic accumulator model, have

been developed to account for different decision-making scenarios and provide more flexibility in modeling behavioral data.

Overall, the diffusion model offers a valuable framework for understanding decision-making dynamics by quantifying the accumulation of evidence and the time course of decision processes. Its mathematical formulation provides a precise and testable account of how individuals integrate information over time to arrive at a decision, making it a widely used tool in cognitive science research.



In-painting using diffusion is a technique employed to improve the blending of synthetic images with a background or to fill in missing portions of an image seamlessly. By leveraging the power of palette-to-palette diffusion models, one can effectively address the issue of boundary inconsistencies and create visually pleasing compositions.

Palette-to-palette diffusion models operate by transferring colors and textures from one image region (the source palette) to another (the target palette). This diffusion process ensures a smooth transition and enables the synthetic image to harmonize with the background more effectively.

When dealing with synthetic dataset generation, the challenge often lies in achieving a realistic integration of the synthetic objects with the background. Due to variations in lighting, textures, and perspectives, synthetic images can appear disjointed or out of place. Inpainting using diffusion can be a valuable tool in mitigating these issues.

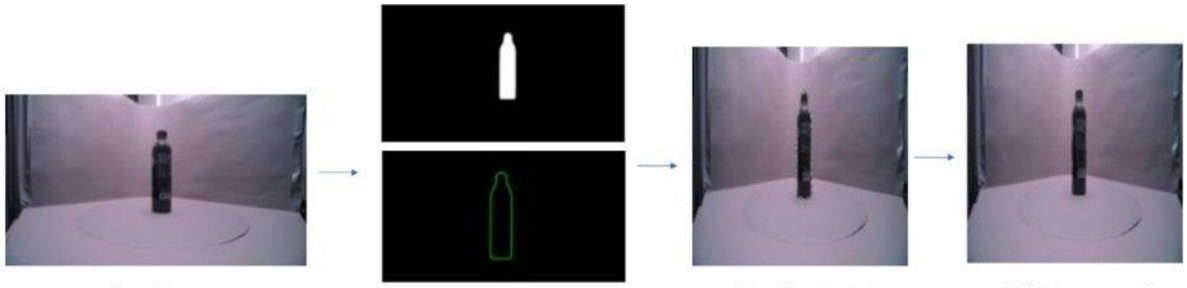
By applying a palette-to-palette diffusion model, the algorithm analyzes the boundary areas of the synthetic objects in the image. It then determines the most appropriate colors and textures from the background image to fill in these regions, making the synthetic objects appear more natural and consistent with their surroundings.

The diffusion process takes into account the color and texture similarities between the synthetic object boundaries and the surrounding areas in the background image. It ensures a seamless blend by gradually transitioning the colors and textures from the background to the synthetic object boundaries. This diffusion technique eliminates abrupt edges, smoothes out inconsistencies, and enhances the overall visual coherence of the image.

Inpainting using diffusion can significantly improve the quality of synthetic dataset generation. It allows for the creation of more realistic and visually appealing compositions, which can be beneficial in various applications such as computer vision, machine learning, and graphics.

It's worth noting that the effectiveness of inpainting using diffusion depends on the quality and diversity of the background images used, as well as the accuracy and robustness of the diffusion model. Adequate preprocessing and post-processing techniques, such as noise reduction and edge refinement, can also contribute to achieving better results.

In conclusion, inpainting using diffusion, particularly through palette-to-palette diffusion models, offers a powerful approach to blend synthetic images seamlessly with background images. By addressing the boundaries of synthetic objects and ensuring a harmonious integration, this technique enhances the realism and visual appeal of generated datasets.



PUGH MATRIX

CRITERIA [Design Concept]	GAN (DCGANs, Pix2Pix)	NeRFs
Realism: Similarity between the synthetic image and the original image	0	+1
Training time	0	+2
Training data requirements	+1	0
Model Complexity	-1	+1
Ability to generate different orientations	0	+5
Image Quality	+2	0
Memory Requirements	+1	+1
TOTAL	+3	+10

TEST DESIGN

a) To further test our photorealistic image generation task, we can consider the following plan:

a. Functional testing: This will involve testing the different approaches we have considered, including NeRFs, pixelNeRFs, DCGANs, and pix2pix, to ensure that they can generate photorealistic images that can improve the accuracy of object detection models.
b. Integration testing: We will integrate the different approaches with the baseline classifier and test their performance in terms of accuracy, speed, and image quality. This will enable us to identify the best approach for our task.
c. Usability testing: We will test the ease of use of the different approaches and how well they can be integrated with the baseline classifier. This will enable us to identify any usability issues and improve the user experience.
d. Performance testing: We will test the performance of the different approaches in terms of computational resources required, time taken to generate an image, and accuracy of object detection on generated images. This will enable us to identify the most efficient approach for our task.

To verify that our design functions are as intended, we will use several methods to verify that our design functions are as intended. First, we will visually inspect the photorealistic images generated by our approach and compare them to the ground truth images to see how closely they match. We will also evaluate the quality of the images based on criteria such as sharpness, clarity, and realism.

Second, we will use metrics such as peak signal-to-noise ratio (PSNR), structural similarity (SSIM), and Fréchet Inception Distance (FID) to compare the generated images with the ground truth images quantitatively. These metrics will provide objective measures of image quality and help us assess the performance of different approaches.

Third, we will evaluate the accuracy of object detection models using the photorealistic images generated by our approach and compare them with the results obtained using the baseline classifier. This will help us assess whether our approach improves the accuracy of object detection models.

Finally, we will conduct user testing to gather feedback on the quality of the photorealistic images generated by our approach. This will help us identify areas for improvement and ensure that the images are suitable for the intended application.

Overall, we will use a combination of qualitative and quantitative assessment methods to verify our design functions as intended and gain insights into its strengths and limitations.

RESULTS AND ANALYSIS

EVALUATION OF SYNTHETIC DATA ON BASELINE MODEL

DATASET v1

syn-GAN-DATASETv1 - *Mix of real and GAN generated crops*

syn-GAN-DATASETv2 – *All GAN generated crops*

EVALUATION MODEL WISE

Dataset -> Model	No of Images	Synthetic Crops	mAP-50	mAP-50-95
Baseline Dataset V1	1000	No	0.831	0.718
Baseline Dataset V2	2000	No	0.859	0.768
syn-GAN-DATASETv1 [pix2pix GAN]	2000	Mix [Scene-level]	0.857	0.749
syn-GAN-DATASETv2 [pix2pix GAN] [Baseline Dataset V2 + syn-GAN-DATASETv1]	**4000**	**Mix**	**0.882**	**0.765**

EVALUATION CLASS WISE PERFORMANCE

*highlighted sections show improvement in performance in mAP50 scores on syn-GAN-DATASETv2 compared to performance on DATASET v2[Baseline]

Class	**Images**	**Instances**	**mAP50** **Baseline**	**mAP50** **syn-GAN-DATASETv2**
All	150	495	0.864	**0.882** ↑
Zenify	150	29	0.984	0.979
Rubix Cube	150	33	0.995	0.995
Elephant Keychain	150	26	0.126	0.142
Arabar	150	42	0.891	0.911 ↑
High Protein Loss Bar	150	28	0.721	0.836 ↑
Popcorn	150	42	0.778	0.813 ↑
Rxbar	150	42	0.995	0.995
Elan sw/salty mix	150	18	0.995	0.995
Truly Fruit Punch	150	36	0.962	0.944
Blk water	150	29	0.97	0.927
Lego Minifigures	150	51	0.995	0.993
Edamame Beans	150	33	0.751	0.778 ↑
Organic Nutrional Shake	150	43	0.993	0.994 ↑
Pistachios	150	9	0.995	0.995
YumEarth	150	34	0.813	0.934 ↑

From the table we can visualize that for deformable shapes the augmented dataset has resulted in accuracies across the spectrum, whereas the same is not observed for solid shapes and objects. The low accuracy and mAP observed for a particular class of “elephant keychain” was further investigated and further base image augmentation were performed to improve the accuracy of the object detection model at a class-wise level.

The following table shows a comprehensive list of experiments performed to improve the accuracy of the

object detection model on the elephant keychain classifier.

Dataset -> Model	No of Images	Synthetic Crops	Validation Dataset Modality	mAP-50	mAP-50-95
Baseline Dataset/Model	1000	No	RGB	0.831	0.718
Baseline Dataset/Model	2000	No	RGB	0.859	0.768
syn-GAN-DATASETv1 [pix2pix GAN]	2000	Mix	RGB	0.857	0.749
syn-GAN-DATASETv2 [pix2pix GAN]	**4000**	**All**	**RGB**	**0.882**	**0.765**
Baseline Dataset [Base Image Augmentations]	2000	No	RGB	0.821	0.721
Baseline Dataset [Color Perturbations]	2000	No	RGB	0.802	0.713
Baseline Dataset	2000	No	Grayscale	0.821	0.727

As an ablation study, we also analysed the performance of the object detection algorithm when the input images were in grayscale. The following table summarizes our results:

Class	Images	Instances	mAP50 Baseline	mAP50 GrayScale Test Dataset
All	150	495	0.864	0.821
Zenify	150	29	0.984	0.795
Rubix Cube	150	33	0.995	0.995
Elephant Keychain	150	26	0.126	0.994
Arabar	150	42	0.891	0.82
High Protein Loss Bar	150	28	0.721	0.857
Popcorn	150	42	0.778	0.854
Rxbar	150	42	0.995	0.871
Elan sw/salty mix	150	18	0.995	0.944
Truly Fruit Punch	150	36	0.962	0.751
Blk water	150	29	0.97	0.813
Lego Minifigures	150	51	0.995	0.808
Edamame Beans	150	33	0.751	0.411
Organic Nutrional Shake	150	43	0.993	0.928
Pistachios	150	9	0.995	0.641
YumEarth	150	34	0.813	0.831

As is evident, we observe that using grayscale images, the detection accuracy for deformable images has drastically improved when compared to the baseline model!

The experiments performed on grayscale modality showed improved performance for deformable objects compared to solid shape objects. The following table shows the results for the model at a class wise level.

SUCCESS CRITERIA

Some criteria that we think need to be fulfilled for the project to be considered successful are as follows:

1. Develop and integrate multiple approaches for generating photorealistic images, including NeRFs, pixelNeRFs, DCGANs, and pix2pix, and compare their performance with the baseline classifier. (required goal - in progress)
2. Utilize diffusion models to generate higher-quality images and evaluate their performance against the other approaches. (stretch goal - pending)
3. Verify that the generated images are of high quality and suitable for improving object detection models' accuracy, using both qualitative and quantitative assessment methods. (required goal - in progress)

    a. Qualitative assessment: Visual inspection of the photorealistic images to ensure that they are realistic and of high quality
    b. Quantitative assessment: PSNR, SSIM, and FID can be used to measure the similarity between the generated images and the ground truth images. Higher values for these metrics indicate better image quality.
4. Improve the accuracy of object detection models using the photorealistic images generated by the different approaches compared to the baseline classifier. (required goal - in progress)
    - Mean Average Precision (mAP): This is a commonly used metric for evaluating the accuracy of object detection models. The mAP measures the average precision of the model across all object categories.
    - Intersection over Union (IoU): This metric measures the overlap between the predicted bounding boxes and the ground truth bounding boxes. Higher values for mAP and IoU indicate better object detection accuracy.
5. Conduct user testing to ensure the photorealistic images suit the intended application to develop robust object classifiers. (stretch goal - pending)
6. Achieve a significant improvement in the accuracy of object detection models using the generated images compared to the baseline classifier. (stretch goal - pending)
7. Ensure that the project is completed within the allocated time and budget. (required goal - in progress)

At this point, the project has made progress toward achieving the required goals. However, some stretch goals are pending and may require additional time and resources. To be considered successful, the project should significantly improve mAP and IoU using the photorealistic images generated by the different approaches compared to the baseline classifier. Additionally, the generated images should be of high quality and suitable for improving the accuracy of object detection models, as determined by both qualitative and quantitative assessment methods.

IMPACT AND CONSEQUENCES

1. Privacy: The project may involve handling and processing sensitive data, such as personal images, medical records, or identifiable information. Ensuring privacy protection and complying with applicable privacy laws and regulations are crucial to safeguarding the privacy of individuals. Unauthorized access, misuse, or improper handling of data could lead to breaches of privacy and legal consequences.
2. Data Security: Proper data security measures should be implemented to protect the data used in the project. This includes encryption, access controls, secure storage, and secure transmission protocols. Inadequate data security can lead to data breaches, identity theft, or unauthorized use of personal information.
3. Informed Consent: If the project involves using images or data from individuals, obtaining informed consent is essential. Users or participants should be informed about the purpose of data collection, how it will be used, and any potential risks or consequences. Respecting individuals' autonomy and privacy rights through informed consent processes is crucial to maintain ethical standards.
4. Patient Confidentiality: In the context of medical image generation, patient confidentiality is of utmost importance. Strict adherence to patient confidentiality regulations and guidelines, such as the Health Insurance Portability and Accountability Act (HIPAA) in the United States, is necessary to protect patients' sensitive medical information.
5. Bias and Fairness: Care should be taken to ensure that the generated images do not perpetuate biases or stereotypes, particularly regarding race, gender, or other sensitive attributes. The training data used should be diverse and representative, and algorithms should be designed to mitigate bias and promote fairness.
6. Environmental Impact: The project's computational requirements may consume significant energy resources, resulting in an environmental impact. Implementing energy-efficient hardware,

optimizing algorithms for reduced resource consumption, and considering renewable energy sources can help mitigate the environmental footprint of the project.

7. Intellectual Property: If the project involves using copyrighted images or proprietary data, respecting intellectual property rights is essential. Obtaining appropriate permissions or licenses and respecting copyright laws are necessary to avoid legal consequences.
8. Transparency and Accountability: It is important to maintain transparency throughout the project, especially in cases where synthetic images are used for deceptive or misleading purposes. Clearly communicating the nature of synthetic images and their potential limitations is crucial to avoid unethical or misleading uses.
9. Social Impact: The use of synthetic images can have social implications, such as the potential for misuse, misinformation, or deepfakes. Responsible use of technology, raising awareness about the existence of synthetic images, and promoting media literacy can help mitigate negative social impacts.

References

[1] S.Philip, ”An Introduction to Image Synthesis with Generative Adversarial Nets”,17 Nov 2018, [Online].available: https://arxiv.org/pdf/1803.04469.pdf

[2] Alex Yu, ”pixelNeRF: Neural Radiance Fields From One or Few Images”, CVPR 2021

[3] Mildenhall, B., Tancik, M., Barron, J.T., Ramamoorthi, R., & Ng, R. "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis", European Conference on Computer Vision 2020

[4] Tancik, M., Li, P., Levoy, M., Barron, J.T., & Ramamoorthi, R. (2021). "Multi-resolution NeRF: A Scalable Representation of the 3D World" Computer Vision and Pattern Recognition

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. "Generative adversarial nets" In Advances in neural information processing systems, pages 2672–2680, 2014.

[6] Isola, P., Zhu, J. Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In Proceedings of the IEEE conference on computer vision and pattern recognition

[7] Radford, A., Metz, L., & Chintala, S. (2015). "Unsupervised representation learning with deep convolutional generative adversarial networks" arXiv preprint arXiv:1511.06434

[8] C. Saharia, W. Chan, H. Chang, C. Lee, J. Ho, T. Salimans, D. Fleet, and M. Norouzi, “Palette: Image-to-image diffusion models,” in Proceedings of SIGGRAPH, pp. 1–10, 2022.