# Single and Multi-View 2D Image Processing for Enhanced 3D Object Reconstruction

by

Swapnil Ghosh
20301470
Md. Muhtasim Mahmud
20101524
Asrar Ahmed
20101522
Tashdid Al Shafi Turjo
20101311
Md. Shaeak Ibna Salim
20101044

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
Bachelor of Science in Computer Science

Department of Computer Science and Engineering
Brac University
May 2024

# Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

**Student's Full Name & Signature:**

Swapnil Ghosh
20301470

Md. Muhtasim Mahmud
20101524

Tashdid Al Shafi Turjo
20101311

Asrar Ahmed
20101522

Md. Shaeak Ibna Salim
20101044

# Approval

The thesis/project titled "Single and Multi-View 2D Image Processing for Enhanced 3D Object Reconstruction" submitted by

1. Swapnil Ghosh (20301470)
2. Md. Muhtasim Mahmud (20101524)
3. Asrar Ahmed (20101522)
4. Tashdid Al Shafi Turjo (20101311)
5. Md. Shaeak Ibna Salim (20101044)

Of Spring, 2024 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on May 20, 2024.

**Examining Committee:**

Supervisor: (Member)

Dr. Md. Ashraful Alam
Associate Professor
Department of Computer Science and Engineering
Brac University

Thesis Coordinator: (Member)

Dr. Md. Golam Rabiul Alam
Professor
Department of Computer Science and Engineering
Brac University

Head of Department: (Chair)

Dr. Sadia Hamid Kazi
Associate Professor
Department of Computer Science and Engineering
Brac University

# Abstract

This paper presents an efficient approach for 3D object reconstruction using Single and multi- view 2D image processing. In real-world scenarios, it also focuses on practical applications. Our approach is about the advanced image processing techniques as well as deep learning models which convert multiple 2D views of an object into a detailed 3D model. Our method is a novel application of convolutional neural networks that merge features from each view for ensuring consistent geometry and texture in the final model. Additionally, we introduce a robust merging module based on CNN. It improves the model's fidelity by focusing on areas with significant detail variation across different views. Our tests on lots of challenging datasets show that our method enhances computational efficiency as well as it has significant potential for practical applications in areas such as virtual reality, augmented reality, and automated quality control in manufacturing. This research marks a significant step forward in digital imaging and computer vision, offering new possibilities for industry and technology advancements.

**Keywords:** Convolutional Neural Networks, Depth Camera, three-dimensional geometry

# Acknowledgement

Firstly, we would like to express our utmost gratitude to the Almighty Allah, by the grace of Allah, our thesis was finished without any significant setbacks.

Second, we would like to express our gratitude to Dr. Md. Ashraful Alam and Hamed Efaz MD. Elahi Dad, our supervisor and mentor, for all of the helpful feedback and direction they have given us. Whenever we needed assistance, they were there to help.

Lastly, to our parents: we would not have been able to do it without their unwavering support. Thanks to their generous encouragement and prayers, we are almost ready to graduate.

# Table of Contents

# List of Figures

# Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

$3D$ Three Dimensional

$BoW$ Bag of Words

$CNN$ Convolutional Neural Networks

$CSI$ Crime scene investigation

$DCP$ Deep Closest Point

$LSS$ Latent Support Surface

$R-CNN$ Region-based Convolutional Neural Networks

$ResNet$ Residual Network

$VGG$ Visual Geometry Group

$YOLO$ You Only Look Once

# Chapter 1

# Introduction

Deep learning has spearheaded the development of 3D reconstruction and enable the development of 3D models from 2D environments such as pictures and equivalent. Though there has been significant improvements in image processing models but the task of converting from 2D to 3D model remains one of the most challenging due to inherent nature of 2D environments such as the complex shapes of the 2D objects and other associated obstacles. In order to deal with such challenge and minimize and simplify the task we introduce SMATNet, a computationally heavy yet effective framework that is successful in constructing 3D models from both single and multiple 2D models.

Our proposed SMATNet architecture has 5 modules: Feature Extractor Module, 3D Volume Generator, Context-aware Fusion Module(Contextual integrator), Refinement Network(Enhancer) and lastly Ensemble Module. The first module, Feature Extractor Module has 2 different variations, one consisting of the ResNet-50 (Encoder1) and the other one consisting one ShuffleNetV2 (Encoder 2). This encoders are used to extract 2D feature maps from inputs. The 3D Volume Generator module consist of sequences of 3D transposed convolutions to convert the feature maps into bigger yet less detailed 3D volumes. In order to integrate various 3D volumes the Context-Aware Fusion module is used, which leverages the contextual information to adjust the weights for each unit of volume and finally combining them into a united and cohesive volume. This approach puts priority on the most dependable sections for each unit volume which creates more accurate reconstructions.

The Refinement Network followed by fusion process improves the resolution of the 3D volume and also improves the level of intricate details of the 3D volumes. This module leverages 3D convolutional layers to improve the volume in each iteration which results in accurate and precise 3D model representations. The Ensemble module ensures durability and accuracy of the rebuilt models by using forecast from both the encoders. This module consist of 2 parts :Stacked encoder and Meta-Model. The Stacked Encoder takes the features from both the encoders and combines them and applies them through the Meta-Model part and finally generates the ultimate feature set.

In our SMATNet architecture, we used PyTorch as our framework which is versatile, well documented, allows abstraction of low level details. The model is trained using Adam optimized which is state of the art technique to allow accelerated convergence and improve performance. The SMATNet architecture shows incredible precision in managing the

intricate details, shapes and overcoming the obstacles possessed by 2D environment, Allowing the creation of 3D models from even a single 2D picture. SMATNet can be applied to Computer vision and related fields where there is need of 3D model reconstruction. SMATNet can be considered as a Remarkable architect for 2D to 3D regeneration model because of its modular nature which makes it adaptable. The stack ensemble along with context aware fusion module expands the limitations and beyond in the field of 3D model generation.

As a result of our research, we have made a number of significant contributions. Through the implementation of ensemble learning, we were able to successfully combine predictions from multiple models in order to improve effectiveness. Through the process of experimenting with various architectures, such as VGG, DenseNet, and Xception, we were able to determine which models were the most successful. By developing a fresh dataset that was specifically customized to a certain industry, we were able to improve the performance and relevance of our models. The ensemble model that we developed displayed higher accuracy and efficiency, outperforming individual models such as VGG, ResNet50, ShuffleNet, DenseNet, and Xception. This was accomplished despite the increased complexity and resource usage.

## 1.1 Problem Statement

Computer vision is not a new concept in the new era of technology. Our technology is advancing with the passing days, and technological agents regarding our daily lives are becoming more intelligent and robust. Nowadays, people are more and more attracted to the concept of automated, autonomous gadgets. One of the best ways to do this is through 3D reconstruction, which can be used in many different ways.

The paper [7] by Maier et al. says that autonomous robots are made to help people with delivery, healthcare, construction, and many other complex tasks. To do these tasks quickly and reliably, you need to be able to construct, find and identify objects in complex environments in real-time. An effective 3D reconstruction system is a must to solve this problem.

Besides humanoid robots, 3D reconstruction can be used for treatments, phenotyping, construction, criminal investigation, Factory-usage and aviation industry, etc. To begin with, 3D reconstruction can be applied to the health sector to increase treatment accuracy. For example, according to [16], computer vision and 3D reconstruction systems can create an accurate heart model from X-ray images of a patient's body's right and left sides. As a result, a patient's diagnosis can be easily done by limiting patient exposure to X-rays. Moreover, a detailed model of a defective organ can help physicians provide the best treatment form and bring out the most efficient diagnostic accuracy and surgical success.

Phenotyping is a process of assessing different traits of plants, including nitrogen consumption, dimensions, yield etc. According to [3], phenotyping is one of the most important aspects of agriculture and has been done manually by farmers based on "Farmer's Intuition" for hundreds of years. In this matter, 3D reconstruction and computer vision

can help the agriculture sector by accurately assessing the plants' dimensions and accurate nitrogen consumption based on the color of the leaves and fruits. As a result, the computer vision system and 3D reconstruction can open the door toward a new era of agriculture.

Construction is the base for advancement, and in the present day, construction has become more complex and becoming enormous and time-consuming day by day. Computer vision applications in this sector can add a new dimension to it. According to [24], about 745,000 employers and over 7.6 million employees construct structures worth nearly 1.4 trillion dollars annually in the United States. Moreover, construction requires various complex calculations regarding height, weight, and quantity measurements. Computer vision can quickly solve these issues through 3D reconstruction and dimension calculation. Zhou [33] suggests that a 3D reconstruction framework can create novel architectures of any kind of structure. Moreover, 3D dimension calculation can help measure height and width. Therefore, these calculations can help the architects calculate the required amount of cement, rod, and other necessary construction elements.

Criminology, the field of behavioral and social sciences, which includes the study of crime along with irrational parts of human nature is a relevant topic for research. Crime scene investigation, also known as CSI in short, is a branch of Criminology. The investigation follows a systematic process of acquiring evidence and making sound decisions. According to Gazi[11], Computer Vision techniques and 3D models can aid crime scene investigation by reconstructing a near original picture of the crime scene which can then be used for deeper analysis without missing substantial detail.

Production line or in manufacturing, 3D reconstruction methods are used in quality control by cross matching which can find defects, compare dimensions and inspect labeling. Computer Vision can monitor factory operations, detect safety hazards, and improve security in the manufacturing field. According to [31] computer Vision can also be used to detect anomalies, find potential failures, and overall make all manufacturing processes more efficient and cut a lot of costs.

The aviation sector, in which computer vision technology can be used to enhance safety mechanics, increase performance and cut costs by automating some part of the process. During flight the vital flight components can be monitored using computer vision with a potential point of failure detection mechanism which can greatly increase safety.

Nowadays, we primarily acquire 2D concepts from any vision-related technology (camera). But 3D reconstruction requires depth data that cannot be acquired from the usual image capturing devices. So, extracting 3D information from 2D images or perceptions becomes difficult. So, the concept of depth sensors was introduced to solve this issue. However, though depth sensors can provide depth data of an object, it is often quite noisy and price inefficient. Therefore, in this paper, we try to find the answers to the question- "Single and Multi-View 2D Image Processing for Enhanced 3D Object Reconstruction." Therefore, we analyze numerous research papers and try to come up with an efficient implementation of 3D reconstruction techniques using multiple good datasets and combine it with Deep Learning to make it more robust.

## 1.2 Research Objective

This research aims to develop a method that utilizes Deep Learning technology for the reconstruction of 3D objects. Therefore, we have used CNN as our primary deep learning model and depth cameras as our peripherals. The objectives of this research are:

1. To implement a minimal and efficient method for obtaining the 3D reconstruction of an object.
2. To deeply understand the mechanisms of Deep Neural Networks
3. To construct an efficient Dataset for obtaining better results in specific domains.
4. To Improve Resolution and Detail of 3D Models.
5. To Integrate Ensemble Learning for Enhanced Accuracy.

# Chapter 2

# Literature review

## Related Works

People are used to seeing depth cameras used in 3D modeling and structure reconstruction. With Kinectfusion, a user can make a geometrically accurate 3D model of an object by holding and moving the depth camera toward the object. In the paper [6], by Izadi et al., this process is explained in more detail. According to their claim, the Kinect depth camera constructs an inherently noisy depth map of the target object using a depth ray sensor, which can be modified using a novel GPU pipeline-based system. According to the authors, after generating the depth map, the GPU implementation goes through four steps: (i) depth map conversion; (ii) camera tracking; (iii) volumetric integration; and (iv) raycasting. After that, in real-world physics, ICP outliers are segmented and simulated to dynamically reconstruct the indoor scene in an AR. According to their claim, the described system can show a high level of user interaction directly in the indoor scene.

While the previous paper discussed the use of the Microsoft Kinect depth camera, Maier et al.'s paper [7], looks at the Asus Xtion depth camera. In this paper, the authors elaborate on creating a 3D map for a humanoid robot to perform collision-free movement in a complex indoor environment. In this paper, the authors mainly focused on recognizing the surroundings and localizing objects through the depth data gained from the depth camera. To do that, they discussed some procedures: i) environment representation; ii) probabilistic map update; iii) localization; and iv) path planning. Unfortunately, the proposed framework has range limitations, as it fails to recognize any object within a range of 50 cm. But moving back a little solves the issue. As a result, using the Asus Xtion RGB-D depth camera, an accurate local 3D map and localization of objects can be reconstructed for any autonomous moving agent.

The paper [5], by Geiger et al. proposes a novel real-time approach to creating 3D maps from stereo sequences. According to their claim, the reconstruction pipeline is combined with stereo matching and multi-view linking techniques and, therefore, an odometry method (an algorithm running at 25 frames per second) to generate consistent 3D cloud points (depth maps at 3–4 fps). Some of the proposed procedures are (i) feature matching; (ii) egomotion estimation; (iii) stereo matching; and (iv) 3D reconstruction. According to the authors, their work is highly suitable for any kind of computer vision for intelligent agents.

Also, the paper [8], by Baak et al. discusses efficient techniques (a generative and discriminative framework) to generate a full body pose estimation from noisy depth data image streams. Here the authors propose an efficient and robust sparse Hausdorff distance for fusing local optimization and a database lookup technique for detecting the features accurately. The pose reconstruction framework consists of some necessary procedures: (i) local optimization; (ii) feature computation; (iii) database lookup; and (iv) hypothesis voting. The authors have demonstrated significant success in entire body pose reconstruction, though there were 45-degree range limitations and capturing limitations for fast-moving objects.

Overall, these papers show different types of efficient techniques for measuring the 3D data of an object using a depth camera. Although depth data has high noise, some robust and optimized techniques are introduced and discussed. Therefore, 3D mapping and reconstruction have become significantly more efficient and accurate.

Stereo vision systems, which use two or more cameras to estimate the distance to objects in a scene, have been widely used for object distance and size measurement. Yahya et al.'s paper [9], describes one such system. The authors suggest using a stereo-vision system made up of two cameras mounted on a pan-tilt unit, to measure the distance and size of things. At first, to measure how far the object is, the triangulation system is used in this system. This system also takes different angles of a specific object by taking pictures of that object. For measuring the distance as well as measuring the size, the average error is 2 percent and 5 percent respectively. Thus, the authors claimed that it's a good result.

Image classification is used to estimate the depth of an object. This method is described in the paper [4], by Chen et al. The authors claimed that, for classifying a picture's depth, convolutional neural network (CNN) is trained. These pictures are classified into three categories, such as- near, mid-range, far. By training CNN, it can be able to classify if it is near or mid-range or far. By training, CNN's output gives a depth map of that picture. For estimating the depth, the average error is less than 2 percent. Thus, the author claims that it's a good result.

Kinect RGB-D sensor is used to estimate the object's dimension. It is described in the paper [10], by Lee et al. It is necessary to raise the accurate result of this dimension estimation problem. For that, the authors said that, using this Kinect RGB-D sensor's dynamic resolution, we can be able to increase the accurate result of this dimension estimation. As the average error for estimating the object's dimension is less than 2 percent, the authors said that it's a good result as well.

Generative adversarial networks (GANs) are used for creating 2D pictures into 3D pictures. This topic is described by Wu et al.[38]. In GAN, It is necessary to compete between these three, such as- two neural networks, a generator network and a discriminator network. Competing with these three makes them a kind of deep learning model. While the generator network makes synthetic data, the discriminator network differentiates this synthetic data. That is how GAN can be able to learn to create synthetic data by training the discriminator network and generator network. This synthetic data is almost inseparable from the actual data. That is how the authors showed how to use GAN for 2D into 3D models. The authors also claimed that this model works well.

Overall, these papers are able to show how effective it is to use different approaches for estimating an object's distance, object's size, object's dimension as well as how to create 2D pictures into 3D pictures. Thus, we can finally say that- image classification is used to estimate an object's distance and object's size while GAN is here to raise the accurate results of estimating it.

Complex-YOLO is to detect an object in a point cloud. It was described and suggested by Liu et al. in [23]. In the KITTI dataset, it shows that Complex-YOLO can be able to win against 3D object identification. When running at a quicker frame rate, Complex-YOLO's result doesn't give a bad result, rather it gives a very good result. Thus, for finding 3D objects in the point cloud, Complex-YOLO can be a dependable approach.

To display an object's three-dimensional form and to forecast its three-dimensional location and orientation, 3D Object Detection using Latent Support Surfaces use a collection of latent support surfaces. Chen et al. offer a latent support surface (LSS) model for 3D object identification in their work [21]. On the widely-used KITTI dataset—used to assess 3D object identification methods—their studies show that LSS-based models attain state-of-the-art performance. As a whole, LSS shows promise as a 3D object detection technology with many possible uses.

A 3D convolutional neural network (CNN) for real-time object recognition was proposed by Maturana and Scherer in 2015 [12]. Using a 3D CNN that learns features from 3D voxel data and a novel network architecture that is efficient for real-time object recognition, the proposed model addresses the challenges of real-time object recognition in 3D scenes. On the ModelNet40 dataset, the model outperforms cutting-edge 3D object recognition techniques while remaining computationally efficient. VoxNet is a promising method for 3D object recognition, and its architecture has inspired other research in the field.

Computer vision applications like self-driving cars, robotics, and surveillance depend on object detection. Object detection requires real-time performance and tolerance for object appearance, scale, and viewpoint. The well-known R-CNN (Regions with CNN Features) family of algorithms combines a region proposal approach with a CNN-based object classifier to solve this problem. The paper [14] by Redmon et al. introduced YOLO (You Only Look Once), which uses a single CNN to predict item positions and class probabilities in a forward pass. YOLO's architecture is optimized for real-time object identification, achieving high frame rates and great accuracy. Unfortunately, computer vision's robust real-time object detection is difficult. The R-CNN family of algorithms and YOLO, optimized for real-time detection with high frame rates and good accuracy, have been proposed to overcome this difficulty.

Computer graphics, robotics, and medical imaging require 3D shape registration to align two or more 3D shapes. ICP, feature-based, and deep learning algorithms can register 3D forms. Wang et al. [40] suggested a "Deep Closest Point" (DCP) in 2018. This deep learning-based registration method aligns shapes based on CNN predictions of dense correspondences. 3D CNN is used to find a feature representation and loss function that captures local fine-grained geometric aspects of forms. This solution outperformed cur-

rent registration methods' accuracy and processing efficiency on public datasets. DCP uses CNN to predict dense correspondences between 3D shapes and gets the best results on many datasets.

Computer vision and graphics research has extensively exploited photographs to comprehend and evaluate 3D shape features. Photometric stereo uses numerous photos of an object under different lighting conditions to recover its shape and surface reflectance."What Do Photographs Tell Us About 3D Shape?" by Srinivasan et al. (2010) reviews that, for recovering 3D shapes from a lot of pictures, the authors present a Bayesian framework. Srinivasan et al's [1] 2010 Bayesian framework is able to overcome photometric stereo's lacking.

Xie et al's[27] study says that, for reconstructing 3D objects from one or more RGB photos, Pix2Vox is a unique framework. This technology overcomes the drawbacks of previous approaches, including the inconsistent results and memory loss problems related to Recurrent Neural Networks (RNNs), such as 3D-R2N2. Pix2Vox takes input photos and uses an encoder-decoder technique to create crude 3D volumes. An important step forward is the context-aware fusion module, which merges these volumes selectively to improve reconstruction quality. The framework outperforms the state-of-the-art methods in both the ShapeNet and Pix3D benchmarks, especially when compared to the state-of-the-art methods for 3D reconstruction. As evidence of its efficacy, Pix2Vox's backward inference time is 24 times lower than 3D-R2N2. The study also demonstrates Pix2Vox's better generalization performance on ShapeNet's untested 3D categories.

Computer vision and computer graphics need to rebuild 3D bilaterally symmetric surfaces. Nature's bilateral symmetry can reveal an object's shape. There are several ways to determine bilateral symmetry in 3D shapes. T. R. S. Walsh and E. K. Wong published [2] in 2006. They suggest fitting a cylindrical or spherical model to symmetric locations along the symmetry axis to recreate a 3D, bilaterally symmetric surface. They demonstrate that the proposed method can accurately reconstruct symmetric 3D objects using a dataset.

So, shape reconstruction of 3D bilaterally symmetric surfaces is an essential problem in computer vision and computer graphics. T. R. S. Walsh and E. K. Wong's 2006 method uses cylindrical or spherical models to detect bilateral symmetry in 3D shapes.
Use LiDAR point clouds for 3D perception, autonomous driving, robotics, and other applications [17]. Real-time LiDAR point cloud multi-object classification and identification, however, is challenging. A new approach for real-time multi-object detection and classification using LiDAR point clouds is YOLO4D, a spatiotemporal technique. Zhang et al. created YOLO4D in 2021 as an improvement to the well-known YOLO object detector that makes better use of the spatial and temporal data from LiDAR point clouds. On the Waymo Open Dataset, YOLO4D outperforms cutting-edge LiDAR-based object identification algorithms in terms of accuracy and speed. So, for real-time spatiotemporal multi-object detection and classification, YOLO4D makes advantage of LiDAR point clouds. It enhances the popular YOLO object detector's speed and accuracy in the identification and categorization of objects by adding spatial and temporal information from LiDAR point clouds.

Computer vision applications like self-driving cars, robotics, and surveillance depend on object detection [13]. Deep learning-based object detection systems are new. These algorithms excel at object detection. Redmon et al.'s (2016) deep learning-based object identification algorithm, YOLO (You Only Look Once), is significant. Using a CNN, YOLO predicts item positions and class probabilities in a single forward pass. Its architecture enables real-time performance and accurate identification. Faster R-CNN, developed by Ren et al. in 2015, combines region proposal with CNN-based object classifier to improve object detection. In conclusion, YOLO and Faster-RCNN algorithms are deep learning-based object detection which are good to give accurate results.

Multi sensors help to improve 3D objects by combining data from many sensors. [39] The limitation of each and every sensor can be surpassed and we can raise the accuracy of the object recognition result by combining data from other sensor modalities. For instance, radar, LiDAR, cameras. "F- PointNet: 3D Multi-Sensor Fusion for Accurate and Efficient Object Detection" by Qi et al. is an approach which can be able to do 3D object detection. "Deep Sensor Fusion for 3D Object Detection" by Weng et al. in 2020, the authors used a feature pyramid architecture to extract features. Multi-sensor fusion is a method that can improve 3D object recognition. It takes data from lots of sensors and then combines it into a single database. F-PointNet and Deep Sensor Fusion for 3D Object identification are two of the methods to increase the accuracy of object identification. Egocentric vision is for object detection and manipulation in robotic manipulation. 3D CNN uses RGB and depth data from a multi-view method and encodes object properties. In summary, the current topic of research is revisiting 3D object detection from an egocentric perspective. The study by Noury et al.[30] in 2020 said a strategy which blends multi-view with 3D CNN.

# Chapter 3

# Methodology

The workflow of research has been divided into two parts:
(i) 3D model reconstruction using existing acclaimed and popular dataset.
(ii) Developing new datasets for domain specific research.

## Workflow

Our research workflow is elaborated in figure 3.1.

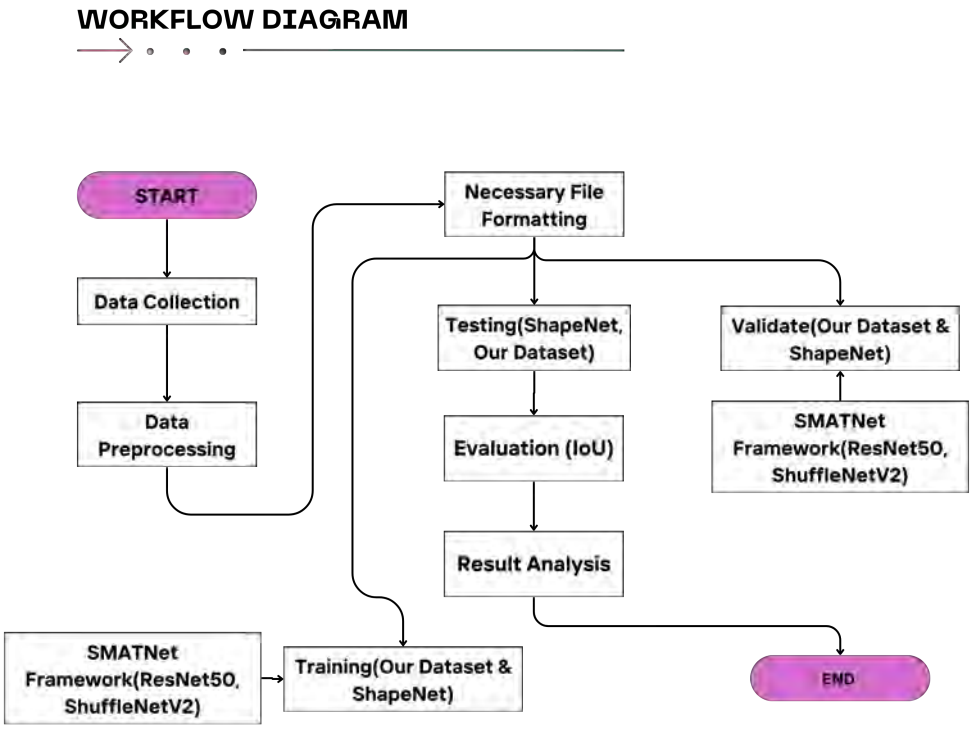



Figure 3.1: The Workflow chart of 3D model reconstruction

# Dataset

In our research purpose,We have utilized a dataset that is already extensively applied to the domain of 3D models and also developed a new one .The dataset that we used is the ShapeNet dataset that includes objects ranging from Aeroplane to sofa and many more.The dataset that we constructed is a human based dataset .A brief descriptions about the dataset is given below.

# ShapeNet dataset

The ShapeNet dataset is a large and well documented dataset containing a large collection of annotated 3D model form of various different objects and associated model. Each category is detailed with information and contains large sample size. ShapeNet is widely used in fields such as computer vision, Image processing domains etc. Due to its extensive library of 3D models, it is an invaluable tool for testing and training algorithms related to these fields. For our research purpose, we have used a specific segment of the dataset. This section includes artificial imagery of things from 13 primary categories, totaling 43,783 3D models.

# Our dataset

We tried to create a human based dataset following the formats of ShapeNet dataset, due to resource limitations, our dataset was not as precise as the existing ShapeNet dataset. Our human based 3D Dataset is a resource designed for research purposes in computer vision and 3D modeling domain. This dataset includes images of multiple human subjects along with corresponding 3D object files, encompassing a total of approximately around 1200 entries. Each entry in the dataset provides annotations of human subjects captured from various angles, ensuring a representation of 3D spatial information. The accompanying 3D object files almost accurately reflect the geometry and posture of the subjects, enabling modeling and analysis. This dataset is particularly valuable for tasks such as 3D human pose estimation, multi-view object recognition, and human-object interaction modeling.

# Dataset Construction

To construct the dataset, we employed the Intel RealSense D435 depth camera, a well known RGB-D sensor, to record pictures and their corresponding depth information. Subsequently, this data was correlated with a comprehensive three-dimensional model.

Initially, we established a connection between our depth sensor, namely the Intel RealSense D435 depth camera, and our laptop. Subsequently, we performed the necessary calibration of the sensor to guarantee the utmost precision in depth estimation. In order to initialize the Realsense depth camera we have written a custom python script to generate the point cloud files(.ply).In this python script we have used external libraries notably pyrealsense2, open3d. In a nutshell The pyrealsense2 l does the job of connecting the Realsense depth camera and open3d does the job of generating the point clouds.In

addition to this,We have also added a time delay to position the respective object properly ,In our case to stabilize the posture of the human i.e front facing, back facing, left side facing, right side facing etc. After executing the code we get the desired point cloud file(.ply).Now in order to get a perfect point cloud file we have to merge the associated point cloud files of the same object together such that for humans all the different point clouds are taken from different angles. Furthermore,We have modified the custom script such that it takes a RGB image at the same time as the depth image, the main reason for taking the RGB image is to compare with the ground truth and to generate our final output which is a .binbox file.

The final result of this workflow is a point cloud file formed by merging associated .ply files , each representing a point cloud of the human body from different angles.Now this merged point cloud file is later converted to .obj file and from there it is finally converted to a binbox file which is again done by a custom python script.

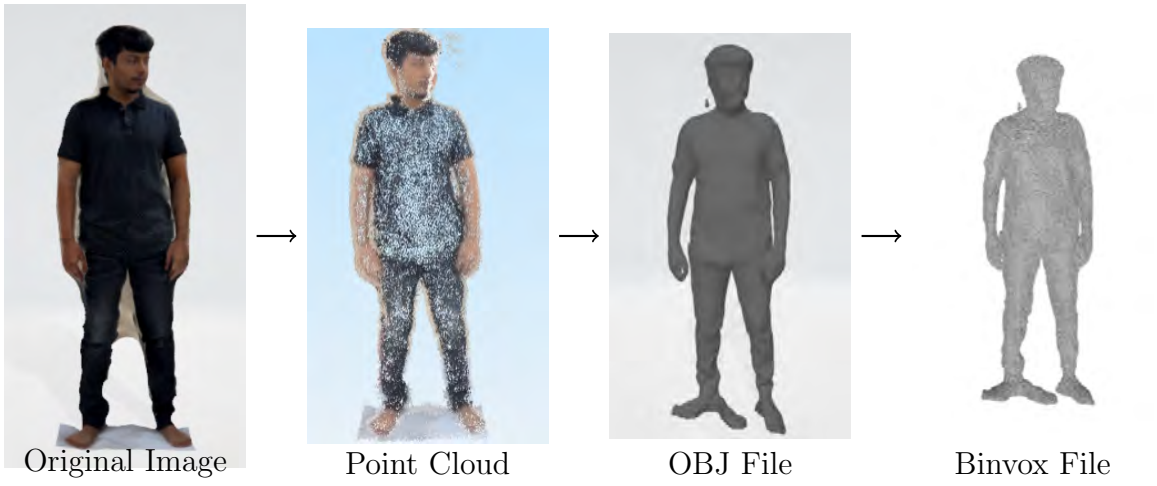



Figure 3.2: Workflow for converting an image to its corresponding binvox file. This process starts with taking a RGB depth image of a particular subject, converts it into a point cloud(PLY), then into an 3D Object(OBJ) file, and finally into a binvox file(.binvox).

# Tools and Peripherals

The mechanism of Intel RealSense depth camera has three distinct sensors: a standard RGB, an infrared (IR), and an infrared laser projector. These lenses work together to allow the cameras to determine depth by detecting infrared light that is reflected from objects in front of them. The acquired visual data, in conjunction with the RealSense software, produces a depth picture. This picture can undergo further processing to serve different purposes, like monitoring movement, developing interactive interfaces that react to movements or facial expressions, and taking colour photographs even in dim lighting situations. RealSense cameras employ stereovision to compute depth, utilising a pair of sensors (left and right cameras) and an infrared projector. The infrared projector generates imperceptible infrared photons that enhance depth accuracy in situations with low roughness. The scene is captured by both the left and right cameras, and the depth for each pixel is calculated by the processor through the correlation of points between

the two pictures. This procedure generates a depth frame, which, when merged with successive frames, produces a depth video stream. The depth of each pixel is determined with respect to a parallel plane of the recording camera, rather than the actual distance from the camera. The RealSense D4 processor is essential for processing these pictures, since it has the capability to handle 36 million depth points per second. These cameras are well-suited for devices that require fast data processing due to their high processing speed.[26]

# Preprocessing

As we mentioned earlier, two different datasets have been used in our research. Though all the datasets were regarding object images, annotations and models, some of the file formats and data were incompatible for our model training purpose. Therefore, we had to convert some of the files into required formats. In our dataset, there were model files with ".obj" extension, therefore, we have used our custom built python scripts to modify the data according to our requirements. For training purposes, our model requires a file with ".binvox" extensions in order to get the 3D model related data. So we preprocessed our datasets according to the requirements which required an extensive and time consuming computation that included heavy GPU memory consumption. It is worth mentioning that our Shapenet dataset file formats were already in the desired file formats and thus did not require any extensions related preprocessing. Later on, we performed further preprocessing on the images by adapting an efficient preprocessing method.

The adapted method systematically performs a series of alterations to pictures, with the purpose of preparing them for neural network training. This is achieved by increasing the diversity of the data and guaranteeing the resilience of the model. Firstly, the photos are transformed into PyTorch tensors, which involves modifying their format to align with PyTorch's required input. Normalisation is the subsequent stage, when pixel values are modified to match a certain mean and standard deviation, guaranteeing conformity with pre-trained models.

In order to introduce unpredictability, the preprocessing pipeline incorporates random permutations of the colour channels, central and random cropping to modify image size, and horizontal flipping. The adjustments of brightness, contrast, and saturation are applied in a random manner to imitate various lighting situations, while the addition of noise replicates real-world variances such as sensor noise or compression artefacts. Moreover, the photos' backgrounds may be substituted with arbitrary colours or designated backgrounds, which is very advantageous for segmentation assignments. These modifications jointly enhance the dataset, increasing the trained model's resilience and ability to generalise well to new, unseen pictures. This ultimately improves the model's performance and accuracy.

# Dataset Splitting

Our whole dataset consists of two different 3D model reconstruction related datasets. Between the two datasets, ShapeNet is the largest in size. So, we utilised 70 percent of the ShapeNet dataset for training, while the remaining 30 percent was allocated for

testing and validation. We followed the same splitting scheme in our newly developed dataset.

# Model Architecture

### 3.0.1 Models

In our research, we have used pretrained VGG16, DenseNet-201, Xception, MobileNet-V3, EfficientNet-B0, ResNet50 and ShuffleNet-V2 models for our Feature Extractor Module and conducted a detailed study about the computational speed and accuracy of each models. Moreover, we have used U-Net architecture in our Refinement Network Module.

### 3.0.2 VGG16

A deep convolutional neural network model called VGG16 is renowned for its depth and application of tiny convolutional filters. It has 16 layers total—3 fully linked layers at the end and 13 convolutional layers made with 3x3 filters. The model does spatial downsampling using max pooling layers. Because of its consistent design and depth, VGG16 was a breakthrough model that outperformed earlier models in image recognition tests. Particularly in computer vision applications, its architecture is now used as a benchmark for CNN systems. The model's performance in the ImageNet competition demonstrated how well it works for tasks involving picture categorization and localization.[20]

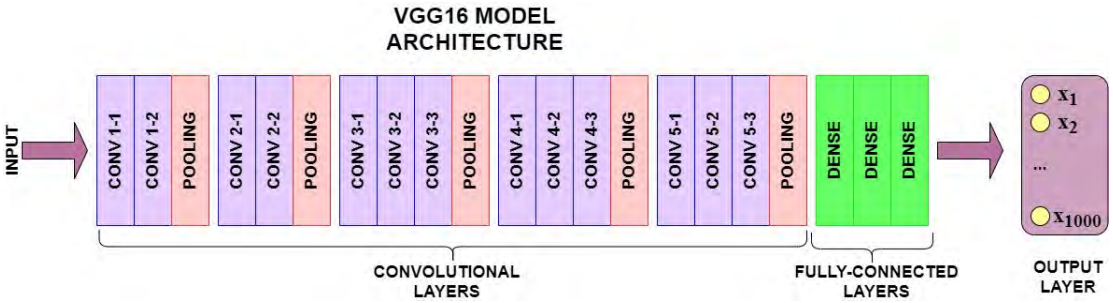



Figure 3.3: VGG16 model architecture[44]

### 3.0.3 DenseNet-201

DenseNet-201 is a deep convolutional neural network renowned for its Dense Convolutional Network design. In DenseNet-201, each layer is interconnected with all other layers in a feed-forward manner. This network comprises a total of 201 levels, which encompass many dense blocks that are joined by transition layers. DenseNet-201 is distinguished by its capacity to enhance the transmission of features, promote the reuse of features, and decrease the parameter count. DenseNet-201 achieves improved performance in picture classification tasks and optimizes the use of computing resources by establishing connections between every layer, enabling efficient gradient flow and maximizing information exchange between layers.[28]

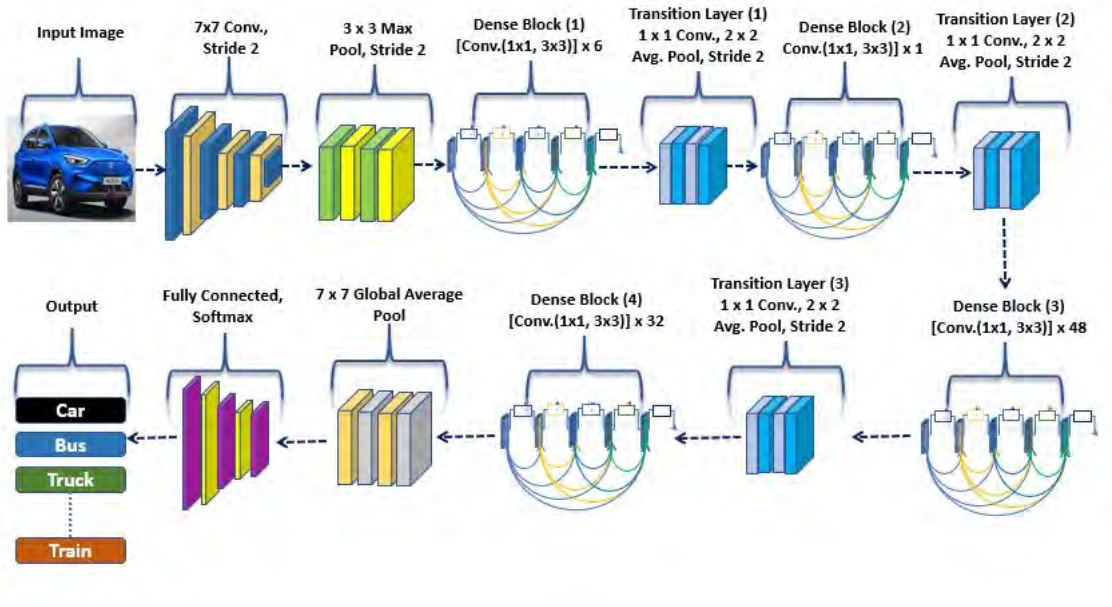



Figure 3.4: DenseNet-201 model architecture[37]

### 3.0.4 Xception

The Xception architecture, proposed by François Chollet, is a deep convolutional neural network based entirely on depthwise separable convolutions. This architecture extends the idea underlying the Inception model by completely decoupling the learning of cross-channel and spatial correlations. In Xception, each module consists of depthwise separable convolutions followed by pointwise convolutions, which significantly reduces the computational cost while maintaining high accuracy. The network has 36 convolutional layers organized into 14 modules, with residual connections around each module except the first and last. This architecture inspired from InceptionV3 outclassed its predecessor by being much more efficient to train and also showing better performance on tasks such as image segmentation,classification on ImageNet dataset.[15]

### 3.0.5 MobileNet-V3

MobileNetV3 learns from its previous versions MobileNetV2, MobileNetV1 and greatly improvises on Neural Architecture search(NAS) and platform-aware optimization.
This architecture combines hardware compatible components and also stream lined layer processes that are especially used to enhance the performance on mobile devices and also embedded devices.It also takes into consideration such h-swish function and uses squeeze-excitation modules to improve the data representation power of the architecture.This allows the architecture to provide cutting-edge precision given its embedded size and limited resources ,Makes it an ideal form of architecture for image processing and related task on mobile and embedded devices.[25]

### 3.0.6 EfficientNet-B0

EfficientNet-B0 is neural network architecture that is successful in achieving a great trade-off between accuracy and computing resource by implementing a technique called compound scaling.In this compound scaling technique a predetermined set of scaling

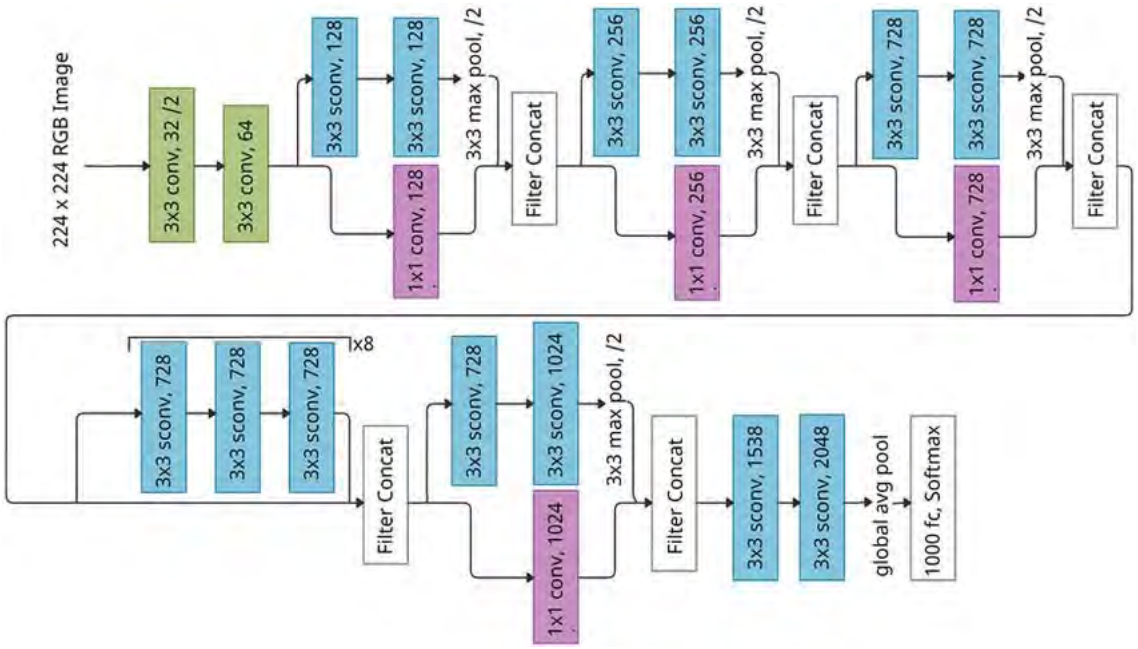



Figure 3.5: Xception model architecture[36]

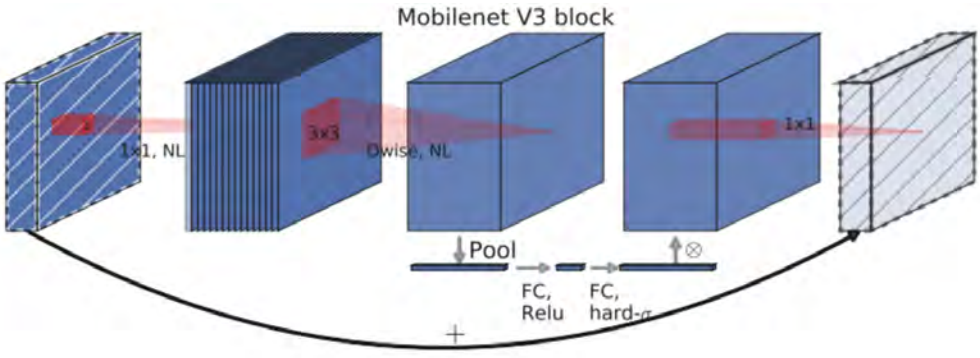



Figure 3.6: MobileNet-V3 model architecture[22]

factors is used to consistently adjust the depth,breadth and also the resolution of the network.EfficientNet-B0 is the first model in EfficientNet series and is inspired by MobileNet architecture having similarities such as squeeze and excitation optimization and also the depth wise separable convolutions.This architecture is able to leverage limited resources but maintaining an excellent accuracy compared the typical CNNs.This architecture is applicable to mobile and embedded devices having limited processing unit.[41]

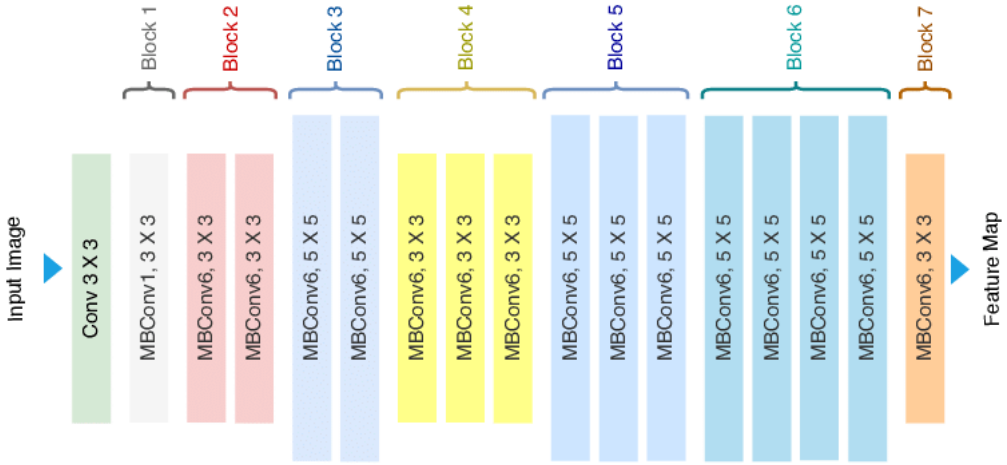



Figure 3.7: EfficientNet-B0 model architecture[29]

### 3.0.7 ResNet50

ResNet50(Residual Network 50),Is a sophisticated CNN composed of 50 layers and was specifically developed to tackle the problem of vanishing gradients in extremely deep neural networks which causes lack in performance .This architecture employs residual learning where there exists skip connections also know as shortcut connections which passes through one or more levels.The ResNet50 compromises of 48 conv-layers and one MaxPolling layer and one AveragePolling layer.This architecture also has a special block called Residual block which consist of identifying mappings to the stacked layer outputs.This clever design allows the smooth flow of gradient and improves the learning process by tackling the vanishing gradient problem. Furthermore,The ResNet50 also uses a bottleneck architecture to improve both the performance and efficiency which is combined made by 1x1 convolution layer to reduce dimension , 3x3 convolution layer and 1x1 layer to restore dimensions.[34]

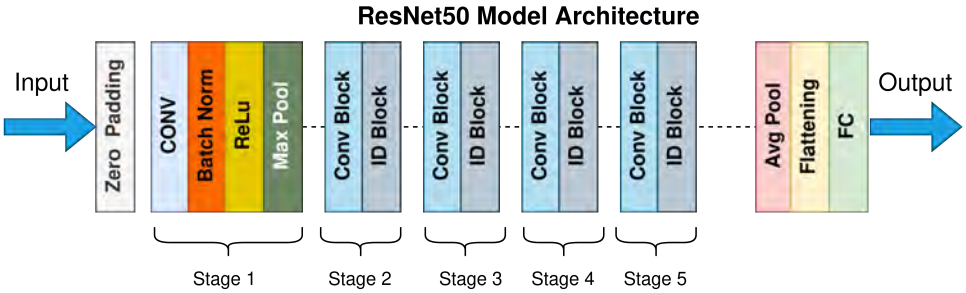



Figure 3.8: ResNet50 model architecture[42]

### 3.0.8 ShuffleNet-V2

ShuffleNet-V2 is mainly established to create realistic endorsements for building the efficient convolutional neural network (CNN) architectures. It is built on ideas of its ancestor (ShuffleNet-V1), gathering the numerous improvements to enhance the speed and accuracy for portable and low-power machines. The major modernization of SuffleNet-V2 is its own improved channel shuffle mechanism and channel split operation, which mainly efficient the computing accuracy. To reduce computational complexity, its architecture handles point by point the set of convolutions and also depth by depth dissociable convolutions. Moreover, the layers are ordered in the direction of ShuffleNet-V2 blocks, in order to improve the gradient flow and speed up merging, also combining batch normalization and residual connections. By separating the input channels within two branches and also the channel split operation allows the network to handle the limited channels at once. As a result, performance is increased and the memory access costs are reduced.[19]

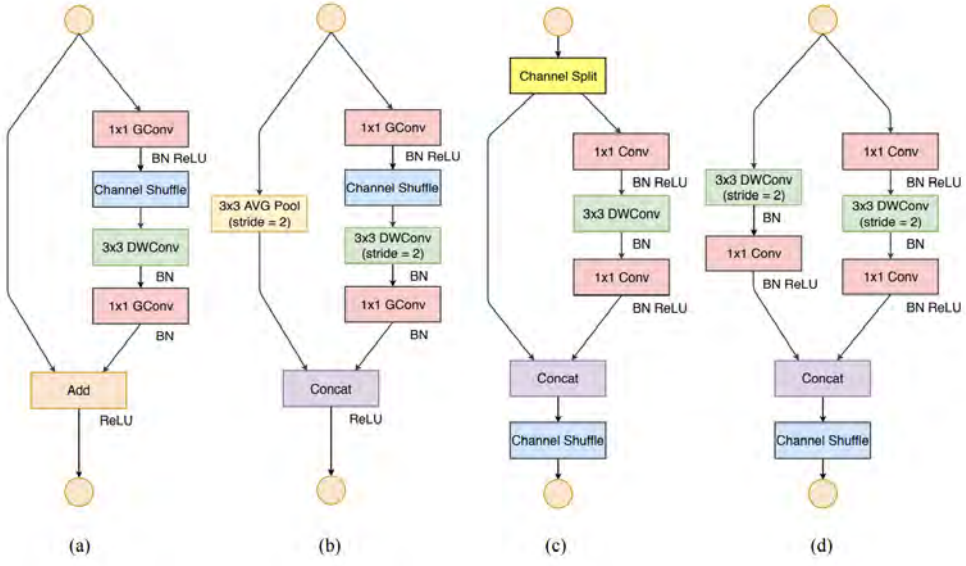



Figure 3.9: ShuffleNet-V2 model architecture[18]

### 3.0.9 U-Net

The main reason for developing the U-Net architecture was to segment biological images using cnn.The network composes of a U-shaped configuration which contains a encoder(Downsampling layer) and a decoder(Upsampling layer).In this architecture,The encoder preserves the contextual information by using combining the conv-layer and max-pooling layer ,On the other hand the decoder captures the spatial intricacies by using transposed convolution.By creating skip links between the layers in encoder and decoder pathways,This architecture is able to gain and preserve high-resolution characteristics which leads to impressive image segmentation capability.[35]

## 3.1 Framework

As PyTorch extensively utilized, thus we used it for our research work. PyTorch is a dynamic computational network. By using PyTorch, users are able to adjust models in

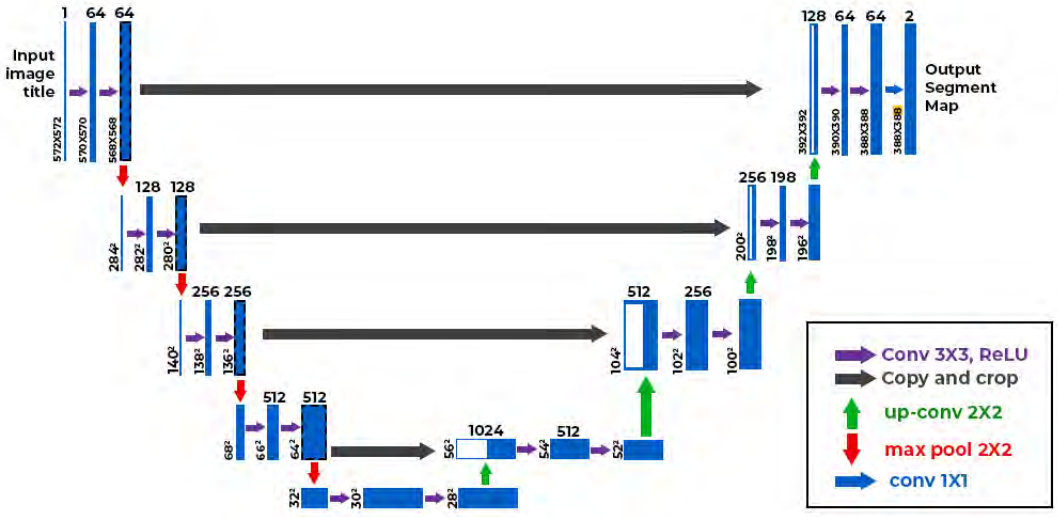



Figure 3.10: U-Net moddel architecture[43]

real time.

### 3.1.1 PyTorch

PyTorch is a deep learning framework. It improves the ease of model creation. Tensor calculation is necessary to train deep learning models. Here, Pytorch gives robust GPU acceleration. This framework gives benefits by constructing many model types. This also gives benefits to a vast community. It also provides lots of materials which are good for every person. Here are some of its features:

### 3.1.2 Dynamic Computational Graphs

In PyTorch, Dynamic graphs allow runtime alterations. For intricate topologies, it is beneficial.

### 3.1.3 Tensor Computation

Both Pytorch and NumPy have strong tensor computing capabilities. But using GPU acceleration, where substantial increase occurs in calculating the performance, is an important aspect for deep learning workloads.

### 3.1.4 Autograd

Autograd functionality is able to calculate gradients automatically. Autograd also makes the backpropagation procedure easier. It can also show a high level of efficiency.

### 3.1.5 Neural Network Library

The torch.nn module surrounds a diverse array of loss functions, and optimizers. Using this approach, to build a neural network is more simple.

### 3.1.6 GPU Acceleration

For lenient purposes, PyTorch uses GPU for tensor computations. It is necessary to decrease the computing time and it is for training large-scale models.

### 3.1.7 Community and Ecosystem

PyTorch has a diverse range of libraries and tools. Thus, it boasts an engaged community. These libraries encompass TorchVision for computer vision, TorchText for natural language processing, and TorchAudio for audio processing.

### 3.1.8 Model Deployment

PyTorch facilitates a smooth transition from research to production by offering tools such as TorchScript. This tool enables the optimisation and deployment of models in production contexts.

### 3.1.9 Interoperability

PyTorch seamlessly interfaces with popular Python libraries like NumPy, SciPy, and scikit-learn, facilitating its utilisation inside pre-existing workflows.

PyTorch is a robust and versatile framework that simplifies the whole deep learning process, including model construction, training, and deployment. It is known for its user-friendly interface and is backed by a supportive community and many resources.[32]

## 3.2 Architecture

To fulfil our research objectives, we developed SMATNet as the fundamental architecture. The architecture has five distinct components, each operating on separate principles. The five modules are as follows: i) Feature Extractor, ii) Ensembler, iii) 3D Volume Generator, iv) Context-Aware Fusion Module, and v) Refinement Network. Each of these components is built using separate Convolutional Neural Network (CNN) architectures.
The Feature Extractor is tasked with obtaining a collection of features from input photos, using either a modified ResNet-50 or ShuffleNetV2 structure, to analyse the images and generate intermediate feature tensors. The Ensembler integrates the features obtained from the several Feature Extractors. It employs a MetaModel to enhance and combine these features into a unified and complete feature tensor, hence improving the resilience and precision of the reconstruction.
The 3D Volume Generator utilises the ensembled features and employs a sequence of transposed convolutional layers to produce a rudimentary 3D volume depiction of the item. The Context-Aware Fusion Module improves these features by merging them and utilising a series of 3D convolutions and Leaky ReLU activations to enhance the accuracy and consistency of the reconstructed volumes. The Refinement Network use a mix of 3D convolutions, fully linked layers, and transposed convolutions to improve the resolution and level of detail in the 3D volumes. This process guarantees that the resulting output is a high-quality 3D reconstruction.

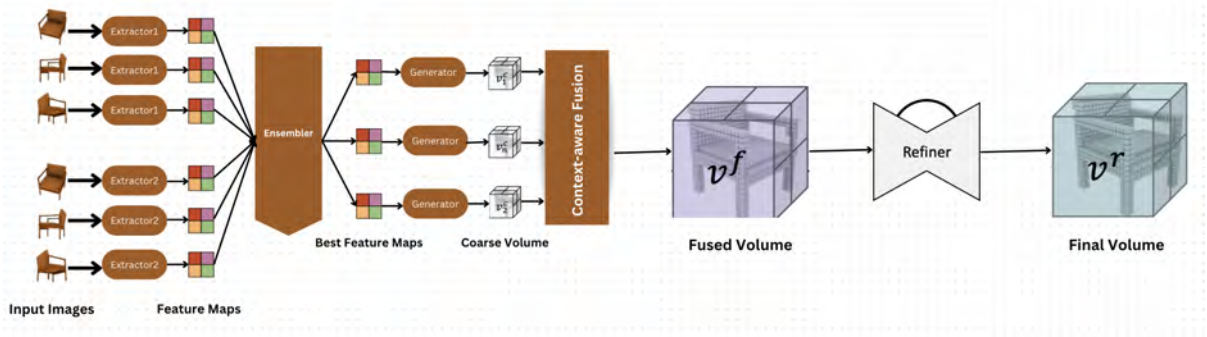



Figure 3.11: SMATNet workflow

### 3.2.1 Feature Extractor Module

The main purpose of the Feature Extractor Module is to gain a collection of features that will be given as input to the 3D volume Generator module.Through this input the 3D Volume Generator module will be able to generate 3D coarse volume.

The architecture is divided into 2 different Extractors,Extractor-1 which is developed using a customized ResNet50 architecture ,The input 2D models having dimension of 224x224x3 is handled by this ResNet50 up to layer3.This convolution layers contains batch normalization, Relu activation function and also MaxPolling layer.The output of this Extractor-1 is an intermediate feature tensor.In addition to this,Three sets of 2D convolutional layers were further used to improvise accordingly.The first set consist of 3x3 convolutional layers having an output channel 512 which is again followed by batch normalization and Relu Activation function.The second set consist of 3x3 convolutional layers with an output channel of 256 and similarly followed by batch normalization , Relu activation and a 2x2 max pooling layer.This layer is primarily used to reduce the spatial dimensions.The third layer is completely similar to the second layer.And the final output is a feature tensor having dimension of 256x7x7 which contains the semantic information.

The Extractor-2 module uses a customized ShuffleNetV2 architecture to achieve a comparable objective using a distinct approach.In the feature layers of the modified ShuffleNetV2 the 2D inputs having dimension of 224x224x3 are processed.This processing continues till the last feature layer which compromises of convolutional and batch normalization layers.Similarly,The intermediate feature tensor goes through three further iterations of 2D convolutional layers .The first set consist of 3x3 convolutional layers each having output of 512 channels followed by batch normalization including ELU activation.The second set is similar to first set,The only difference being the 256 channels instead of 512 channels. The third set replicates the second set by including a 3x3 convolutional layer, 256 output channels, batch normalization, and ELU activation. The output feature tensor obtained using Extractor2 is similarly of dimensions 256 x 7 x 7. Both Feature Extractors utilize distinct designs and processing processes to guarantee that the resulting feature vectors include abundant semantic information, which is crucial for the 3D Volume Generator to produce precise 3D reconstructions.

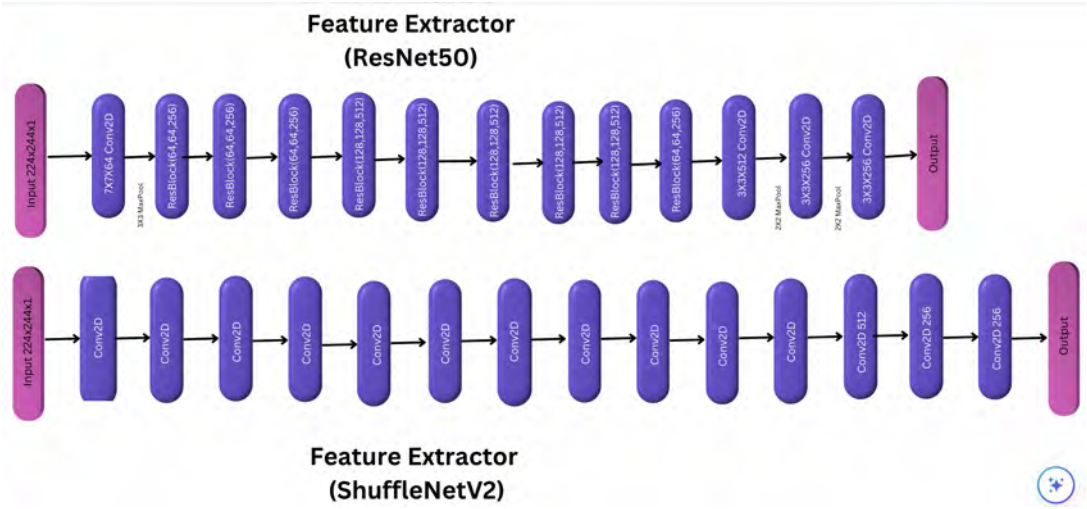



Figure 3.12: Feature Extractor Module architecture

### 3.2.2 Ensembler Module

The Ensembler module inside the SMATNet model combines two Feature Extractors (Extractor1 and Extractor2) and a MetaModel to analyse rendering pictures and produce a full feature tensor. The MetaModel is composed of a solitary linear layer, which requires the input dimension and output dimension as parameters. During the forward pass, the MetaModel performs a process where it converts the input tensor into a flat structure and then applies a linear transformation to it, resulting in the production of an output vector. Ensembler module is combined with instances of Extractor1, Extractor2, the Meta- Model, and extra parameters. This controls the size, height of the feature and width of the output tensor. While using the forward technique, processing by Extractor1 and Extractor2, input rendering pictures undergoes. After that, it generates two sets of feature tensors which are separable. Then, these tensors are combined together, these characteristics provides the ultimate feature representation. Subsequently, the resultant vector is reconfigured to its original shape, as defined by the desired feature size, height, and breadth. Furthermore, an extra singleton dimension is appended to account for the depth. This guarantees compatibility with following levels in the network. The Ensembler utilises a process that capitalises on the capabilities of both Extractors, merging and improving features to optimise the overall performance of the SMATNet model.

### 3.2.3 3D Volume Generator Module

The 3D Volume Generator module inside the SMATNet model is specifically designed to create three-dimensional volumes using the characteristics that have been retrieved by the Ensembler. The 3D Volume Generator comprises a sequence of transposed convolutional layers that increase the resolution of the input feature tensor to generate the ultimate 3D volume. The setup settings decide whether biases are utilised in the transposed convolutional layers. The initial layer of the 3D Volume Generator consists of a 3D transposed convolutional layer. It has 1568 input channels and 512 output channels. The kernel size is 4, the stride is 2, and the padding is 1. This layer is then followed by batch normalisation and a ReLU activation function. The next levels have a similar layout, except each layer decreases the number of output channels: the second layer has

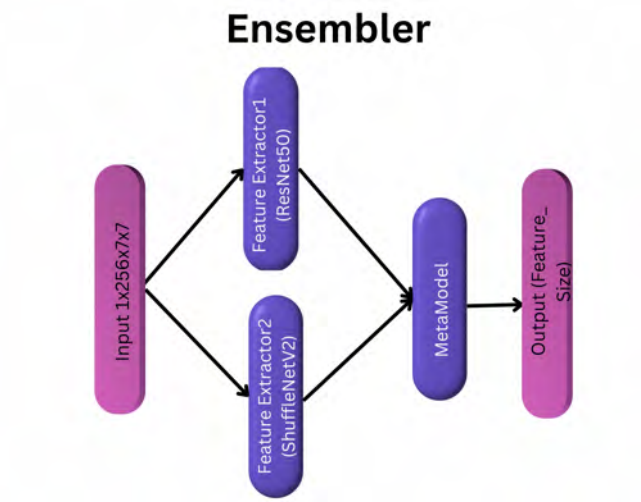



Figure 3.13: Ensembler Module architecture

512 input channels and 128 output channels, the third layer has 128 input channels and 32 output channels, and the fourth layer has 32 input channels and 8 output channels. The last layer consists of a 3D transposed convolutional layer with 8 input channels and 1 output channel. It has a kernel size of 1 and is followed by a sigmoid activation function, which guarantees that the output values are scaled between 0 and 1.

The forward technique initially permutes the input feature tensor and then divides it into separate views. Subsequently, each perspective is systematically analysed by the layers of the 3D Volume Generator. The initial view is resized to match the input size of the first layer, which is 1568 x 2 x 2 x 2. As it traverses each layer, the spatial dimensions undergo a doubling, but the number of channels is decreased based on the layer's structure. The intermediate output sizes are as follows: after the first layer, the size is 512 x 4 x 4 x 4; after the second layer, the size is 128 x 8 x 8 x 8; after the third layer, the size is 32 x 16 x 16 x 16; and after the fourth layer, the size is 8 x 32 x 32 x 32. The last layer produces a volume with dimensions of 1 x 32 x 32 x 32. This volume is then combined with the original characteristics. The processed volumes are gathered into lists and arranged in a stacked manner to create the final output tensors. The meticulous and organised methodology guarantees that the 3D Volume Generator efficiently enhances and improves the input characteristics to produce precise 3D volumes.

### 3.2.4 Context-Aware Fusion Module(Contextual Integrator)

The Context-Aware Fusion Module in the SMATNet model is tasked with enhancing the unprocessed characteristics and rough quantities produced by the 3D Volume Generator. The system employs a sequence of 3D convolutional layers to analyse and merge these characteristics into a unified volume. The configuration settings (cfg) determine the value of the negative slope for the Leaky ReLU activation function. The first four layers of the Context-Aware Fusion Module comprise 3D convolutional layers with 9 input and output channels. Each layer has a kernel size of 3 and padding of 1. Following each convolutional layer, there is batch normalisation and a Leaky ReLU activation. The layers in this process repeatedly improve the input characteristics by maintaining their spatial dimensions and performing non-linear modifications. The fifth layer distinguishes itself by combining

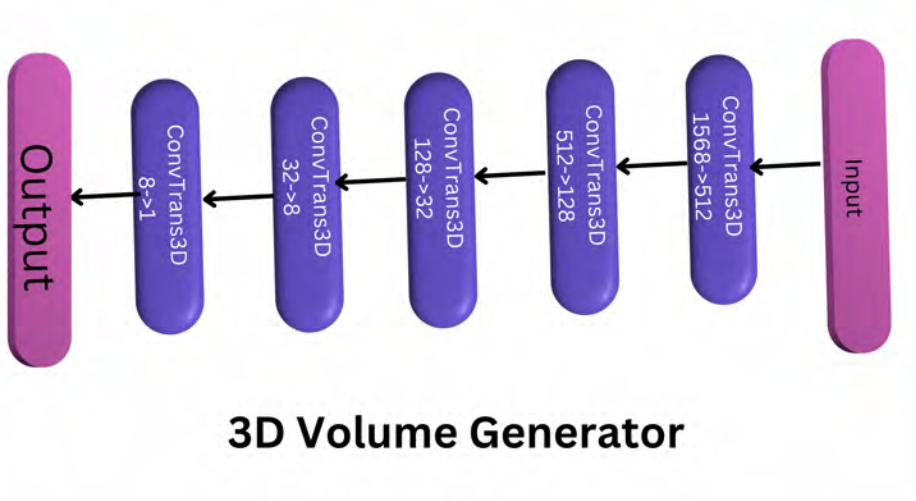



Figure 3.14: 3D Vlume Generator Module architecture

the outputs from the preceding four layers along the channel dimension, resulting in an input size of 36 channels and producing 9 channels by another series of 3D convolution, batch normalisation, and Leaky ReLU. The last layer employs a similar 3D convolutional setup to further decrease the number of channels to 1.

The forward technique involves processing the raw features and coarse volumes in order to calculate the ultimate merged volume. The features tensor in its original form is divided into separate views, and each view is processed in a sequential manner through the layers of the Context-Aware Fusion Module. At first, every view has dimensions of [batch size, 9, 32, 32, 32]. While traversing the initial four layers, the dimensions of the input stay consistent at [batch size, 9, 32, 32, 32]. However, the characteristics of the input are progressively enhanced at each stage. In the fifth layer, the concatenation process produces an input tensor with dimensions [batch size, 36, 32, 32, 32]. This tensor is subsequently reduced to [batch size, 9, 32, 32, 32] and undergoes additional refinement. The last layer decreases the number of channels to 1, resulting in a tensor of dimensions [batch size, 1, 32, 32, 32]. The improved features are compressed into a [batch size, 32, 32, 32] shape and stored in a list. The processed views are arranged and rearranged to align with the size of the coarse volumes. The volume weights are calculated by applying a softmax function to the views, which guarantees a normalised distribution of important weights. The final refined volume is obtained by calculating the weighted sum of coarse volumes. The resulting value is then restricted to a range of 0 to 1, ensuring that the output values fall within an acceptable range. The intricate process of the Context-Aware Fusion Module enables it to efficiently merge and improve the raw features and rough volumes, hence boosting the overall reconstruction quality in the SMATNet model.

$$s_r^{(i,j,k)} = \frac{\exp(m_r^{(i,j,k)})}{\sum_{p=1}^{n} \exp(m_p^{(i,j,k)})}$$

Using the above formula, the previously mentioned module calculates the score of the r-th voxel. The output of this module is the fused voxel v which is produced by the summation of the product value of coarse volume and the respective scores together.[27]

$$v^f = \sum_{r=1}^{n} s_r v_r^c$$

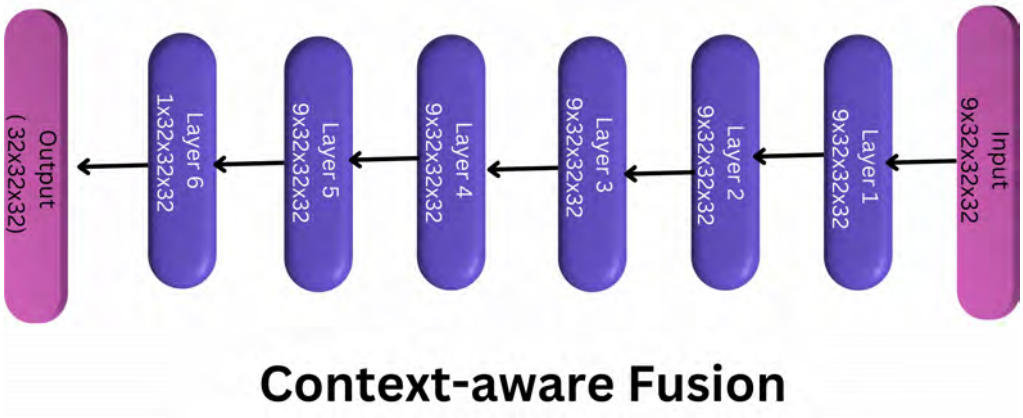



Figure 3.15: Context-Aware Fusion Module(Contextual Integrator) architecture

### 3.2.5 Refinement Network Module(Enhancer)

The Refinement Network module inside the SMATNet model is specifically engineered to augment the precision and intricacy of the 3D coarse volumes produced by the 3D Volume Generator. The model utilises a blend of 3D convolutional, fully connected, and transposed convolutional layers to iteratively enhance the input volumes. The configuration parameters (cfg) consist of the negative slope value of the Leaky ReLU activation and a flag to indicate if biases are utilised in the transposed convolutional layers. The initial three layers of the Refinement Network are comprised of 3D convolutional layers. These layers have progressively increasing output channels, starting with 32, then 64, and finally 128. The kernel size for each layer is 4, and a padding of 2 is used. After each layer, there is a sequence of batch normalisation, Leaky ReLU activation, and 3D max pooling with a kernel size of 2. This pooling operation reduces the spatial dimensions by half. The input volumes are processed by these layers, gradually decreasing their size while simultaneously increasing the depth of their features.

The forward technique begins by unsqueezing the input coarse volumes to include a channel dimension. This operation produces a tensor with dimensions [batch size, 1, 32, 32, 32]. The volumes are subsequently processed by the initial 3D convolutional layers, progressively decreasing their dimensions to [batch size, 32, 16, 16, 16], [batch size, 64, 8, 8, 8], and [batch size, 128, 4, 4, 4]. The output of the third convolutional layer is transformed into a flat shape and then fed into two fully connected layers. The first layer reduces the size to [batch size, 2048], while the second layer expands it back to [batch size, 8192]. The vector is then transformed into a shape of [batch size, 128, 4, 4, 4] and combined with the output of the third convolutional layer. The next transposed convolutional layers increase the size of the volumes. The first transposed convolutional layer increases the size to [batch size, 64, 8, 8, 8], the second layer increases it to [batch size, 32, 16, 16, 16], and the third layer restores the original size to [batch size, 1, 32, 32, 32]. Every transposed convolutional layer in the network is accompanied with batch

normalisation and ReLU activation, except for the last layer, which employs a sigmoid activation function to normalise the output values. The processed volumes are merged with the matching input volumes from previous layers, and the resulting output tensor is multiplied by 0.5 to equalise the influences from various levels. The intricate method guarantees that the Refinement Network efficiently improves the input coarse volumes, resulting in high-resolution 3D reconstructions.

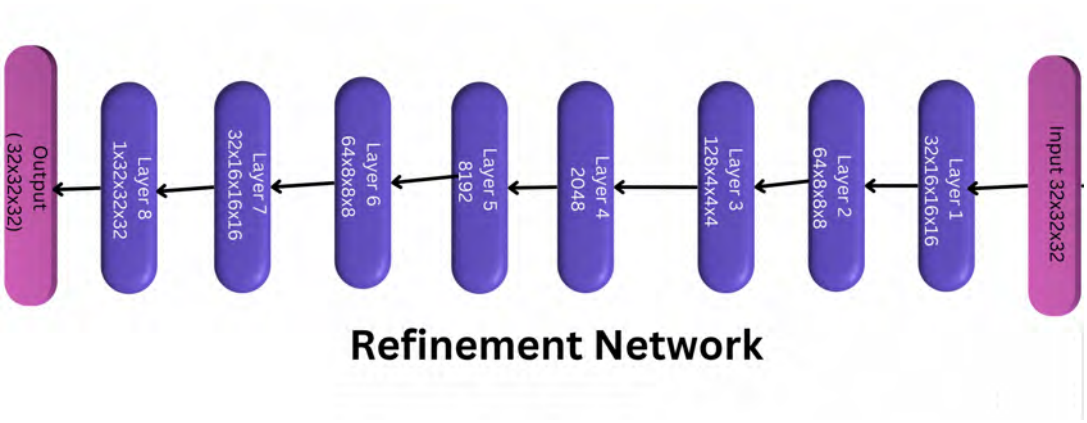



Figure 3.16: Refinement Network Module(Enhancer) architecture

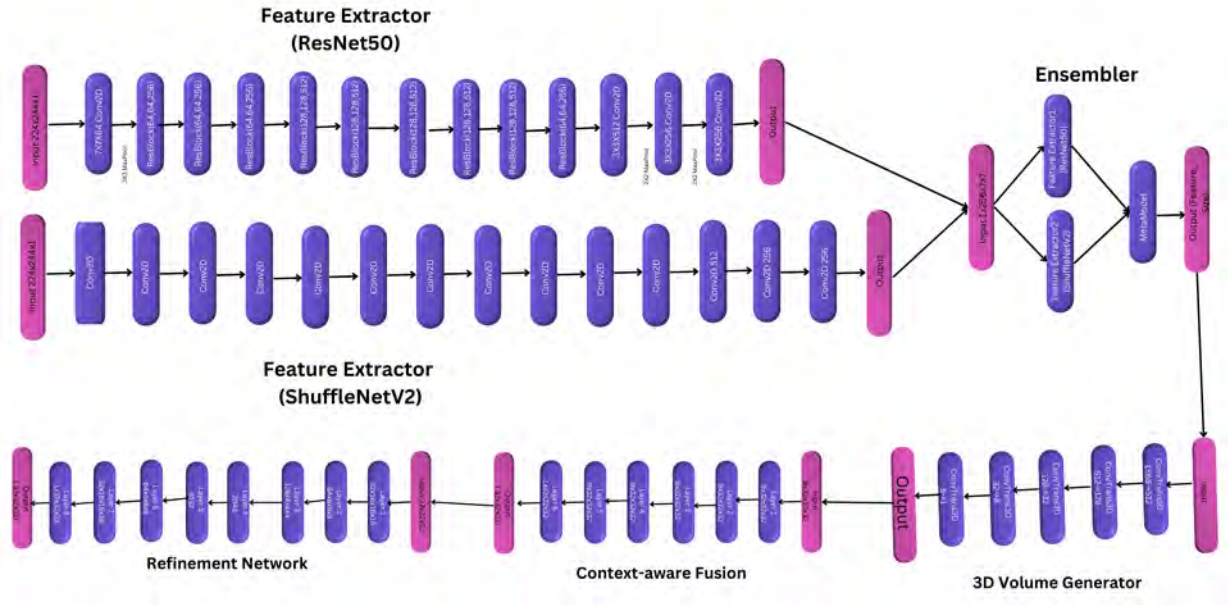



Figure 3.17: SMATNet architecture

## 3.3 Loss Function

The network's loss function is retrieved by the average value of the binary cross entropy of the constructed object and the ground truth for each of the voxels.[27]

$$\ell = \frac{1}{N}\sum_{i=1}^{N}[gt_i \log(p_i) + (1 - gt_i)\log(1 - p_i)]$$

## 3.4 Evaluation Metrics

In order to justify the output of the stated method, here the probabilities have been binarized at a fixed value of 0.3 and for similarity measure IoU(Intersection over Union) has been used.[27]

$$\text{IoU} = \frac{\sum i, j, k^I \left(P(i, j, k) > t\right) I\left(gt(i, j, k)\right)}{\sum i, j, k^I \left[I\left(P(i, j, k) > t\right) + I\left(gt_(i, j, k)\right)\right]}$$

## 3.5 Implementation Details

SMATNet utilised a batch size of 64 RGB pictures with dimensions of 224 by 224. The final output has a voxel size of $32 \times 32 \times 32$. The whole network was constructed using the Adam optimizer and PyTorch. The model started with a learning rate of 0.001 and underwent 250 epochs, with the learning rate dropping at the 150th epoch. The network is first trained for 150 epochs using just single-view photos. This allows the Refinement Network and Contextual Integrator modules to be activated from the beginning, ensuring that detailed features are enhanced and that feature merging is effective.

After this stage, the complete network goes through an extra 100 epochs of training with varied backdrop colours. This is done to simulate different environmental circumstances and enhance generalisation. The training procedure incorporates data augmentation techniques, such as modifying brightness, contrast, and saturation, along with the addition of random noise with a standard deviation of 0.1. These measures ensure that the model is capable of handling various lighting and noise circumstances with resilience. The weights are stored every 10 epochs to avoid losing progress and simplify model assessment at various phases of training.

The training setup further defines the paths for ShapeNet, Our dataset, and Pix3D datasets, encompassing taxonomy files, rendering routes, and voxel paths. The network uses LeakyReLU activation function with a negative slope of 0.2. Since, biases are not utilised in the transposed convolutional layers, overfitting probability is minimized. SMATNet efficiently generates high-resolution 3D reconstructions by using the Contextual Integrator's traits to merge features and the Refinement Network's traits to enhance details. This showcases the model's ability in handling challenging 3D reconstruction jobs.

## 3.6 Reconstruction of 3D Volume

The SMATNet's methods have been examined in various kinds of circumstances. First, start using a detailed analysis of synthetic photos using the ShapeNet testing set. SMATNet shows a very good performance. By seeing in the Intersection over Union (IoU)

graph, SMATNet shows a good resilience in both single-view and multi-view reconstruction. SMATNet has a good accuracy in capturing intricate features. Specially to replicate fragile components, such as- table legs and lamp fixtures, it also has good accuracy as well. SMATNet also shows a very good performance while it is reconstructed from many perspectives, showing robustness in various kinds of scenarios.

Keeping aside the artificial images, the research includes real-life situations using the Pix3D dataset. By utilising the RenderForCNN pipeline for data production and integrating unpredictable colour and light jittering during training, the enhanced training data improves the model's capacity to adapt to real-world RGB pictures. This is supported by the higher average Intersection over Union (IoU) scores obtained on the Pix3D testing dataset. Qualitative analysis provides further evidence to support the claim that SMATNet is highly effective in tackling the difficulties presented by real-world situations.

In order to assess the models' ability to generalise, more experiments are undertaken using Our dataset. SMATNet demonstrates exceptional generalisation capabilities, resulting in a much higher reconstruction IoU. The findings confirm that SMATNet is capable of effectively managing items that have not been encountered before, demonstrating its resilience and flexibility in a wide range of difficult situations. SMATNet architecture effectively delivers 3D vlume reconstructions by using the Contextual Integrator's traits to merge features and the Refinement Network's traits to enhance details. This shows the model's capability to handle challenging 3D reconstruction jobs.

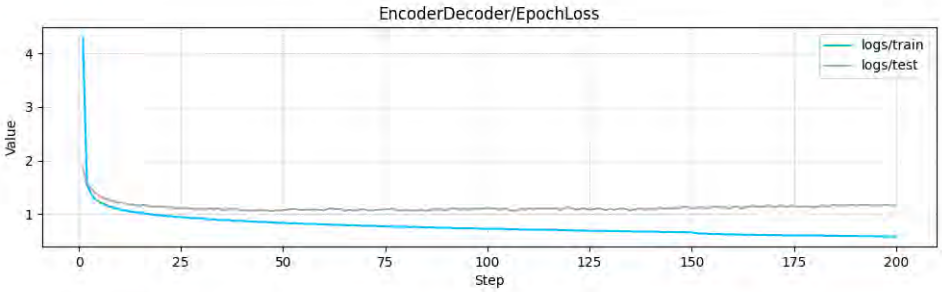
# Chapter 4

# Result Analysis

## 4.1 VGG16

### 4.1.1 Extractor-Generator Loss

In the graph 4.1, it shows the loss of the Extractor-Generator component during both testing and also training stages in a span of 200 epochs. Here, in the y-axis we see the values of the losses which is an indicator of the amount of error and x-axis shows the number of complete epochs which indicates the cycles of training completed.At the start of training, the value of loss is 4.2998 and than it goes through a very significant decline to a value of 0.5680 which is achieved at the end of the training phase. From this we can infer that such reduction in training loss hows that the model is very effective in gathering knowledge and also improvising its capacity to create very accurate predictions as time passes by. Furthermore, the starting testing loss value goes from 1.8613 to 1.1675 indicating that the model is applicable to unfamiliar and unseen data. In addition ,we can see that the both the training and testing loss curves converges which greatly shows that the model is very good at generalization and the most likely there is no form of overfitting as both value of curves gradually decreases.

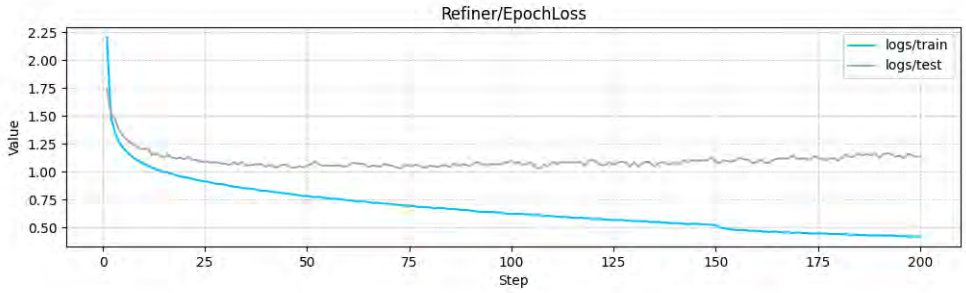



| Run | Min | Max | Start Value | End Value | Δ Value | Δ % | Start Step | End Step |
|---|---|---|---|---|---|---|---|---|
| logs/test | 1.0567 | 1.8613 | 1.8613 | 1.1675 | -0.6938 | -37.28% | 1 | 200 |
| logs/train | 0.5840 | 4.2998 | 4.2998 | 0.5860 | -3.7138 | -86.37% | 1 | 200 |

Figure 4.1: VGG16 Extractor-Generator Loss

### 4.1.2 Refiner Loss

In the graph 4.2, it shows the loss of the Refiner component during both testing and also training stages in a span of 200 epochs. Here, in the y-axis we see the values of the losses which corresponds to the amount of error and x-axis shows the number of complete epochs which indicates the cycles of training completed. At the start of training, the value of loss is 1.7427 and than it goes through a very significant decline to a value of 1.1390 which is achieved at the end of the training phase. This trend represents that the model is undergoing strong learning and the model proficiency is increasing at a good rate. Furthermore, the starting testing loss value goes from 2.2052 to 0.4198 indicating that the model is applicable to unfamiliar and unseen data. In addition ,we can see that the both the training and testing loss curves converges which greatly shows that the model is very good at generalization and the most likely there is no form of overfitting as both value of curves gradually decreases.

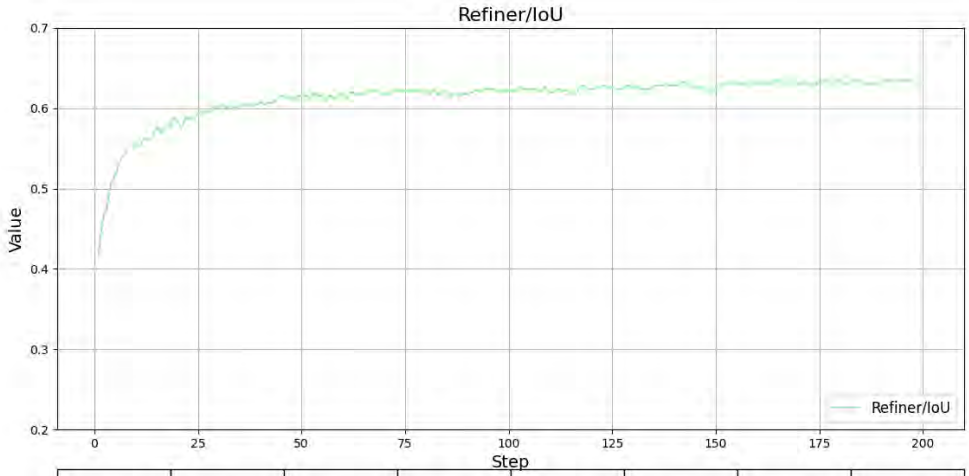



| Run | Min | Max | Start Value | End Value | Δ Value | Δ % | Start Step | End Step |
|---|---|---|---|---|---|---|---|---|
| logs/train | 1.0233 | 1.7427 | 1.7427 | 1.1390 | -0.6037 | -34.64% | 1 | 200 |
| logs/test | 0.4189 | 2.2052 | 2.2052 | 0.4198 | -1.7854 | -80.96% | 1 | 200 |

Figure 4.2: VGG16 Refinement Network Loss

### 4.1.3 IoU Score

The graph 4.3, it shows the Intersection over Union (IoU) metrics of the Refiner component in a span of 250 epochs. Here, in the y-axis we see the values of the IoU and x-axis shows the number of complete epochs which indicates the cycles of training completed. From the graph we can see the constant rise of value of IoU which is initialized at 0.4166 and finalized at 0.6315 at the end of the training session part, signifying a strong improvement of 51.57 percent. An IoU score of 0.6315 can be inferred as a significant indicator that the refiner component enhances the resolution and also the intricate details of the 3D volumes. Achieving and preserving such a value of IoU is very important for maintain the quality of the 3D volume refinement and boosting the overall proficiency of the 3D generation model.




| Run | Min | Max | End Value | ΔValue | Δ% | Start Step | End Step |
|---|---|---|---|---|---|---|---|
| Vgg16 | 0.4166 | 0.6360 | 0.6315 | 0.2149 | 51.57% | 1 | 200 |

Figure 4.3: VGG16 IoU Score

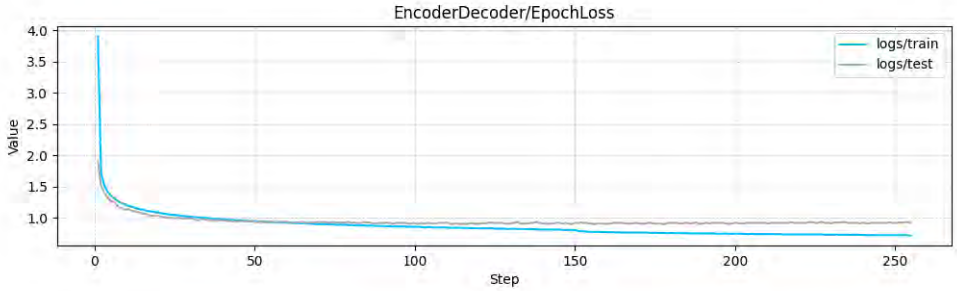
## 4.2 DenseNet-201

### 4.2.1 Extractor-Generator Loss

In the graph 4.4, it shows the loss of the Extractor-Generator component during both testing and also training stages in a span of 250 epochs.Here,In the y-axis we see the values of the losses which corresponds to the amount of error and x-axis shows the number of complete epochs which indicates the cycles of training completed.At the start of training, the value of loss is 3.9056 and than it goes through a very significant decline to a value of 0.7169 which is achieved at the end of the training phase. This trend signifies that the model is gradually becoming efficient in gaining knowledge and also improving its ability to make strong predictions as the epochs progresses. Furthermore, the starting testing loss value goes from 1.9152 to 0.9150 indicating that the model is applicable to unfamiliar and unseen data. In addition, we can see that the both the training and testing loss curves converges which greatly shows that the model is very good at generalization and the most likely there is no form of overfitting as both value of curves gradually decreases.

### 4.2.2 Refiner Loss

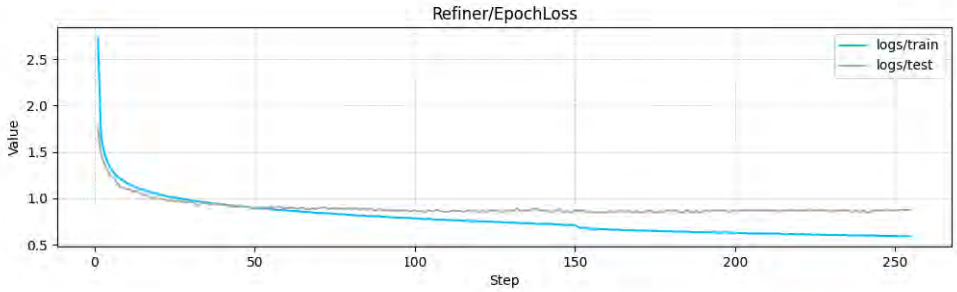
The graph 4.5, it shows the loss of the Refiner component during both testing and also training stages in a span of 250 epochs. Here, in the y-axis we see the values of the losses which corresponds to the amount of error and x-axis shows the number of complete epochs which indicates the cycles of training completed. At the start of training, the value of loss is 1.7541 and than it goes through a very significant decline to a value of 0.8761 which is achieved at the end of the training phase and a significant decrease of about 50.06 percent. This trend signifies that the model is gradually becoming efficient in gaining knowledge and also improving its ability to make strong predictions as the epochs progresses. Furthermore, the starting testing loss value goes decreases by 78.55percent




| Run | Min | Max | Start Value | End Value | Δ Value | Δ % | Start Step | End Step |
|---|---|---|---|---|---|---|---|---|
| logs/test | 0.8959 | 1.9152 | 1.9152 | 0.9150 | -1.0001 | -52.22% | 1 | 255 |
| logs/train | 0.7169 | 3.9056 | 3.9056 | 0.7169 | -3.1887 | -81.64% | 1 | 255 |

Figure 4.4: DenseNet-201 Extractor-Generator Loss

indicating that the model is applicable to unfamiliar and unseen data. In addition, we can see that the both the training and testing loss curves converges which greatly shows that the model is very good at generalization and the most likely there is no form of over-fitting as both value of curves gradually decreases.



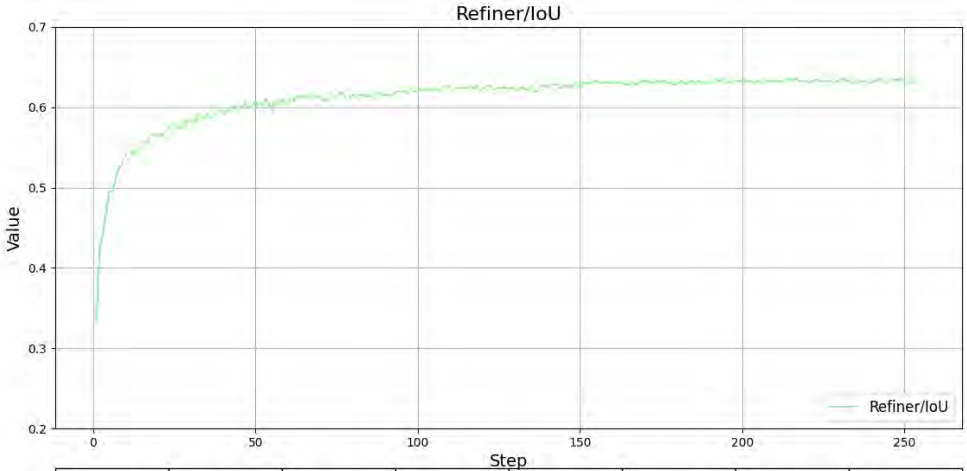

| Run | Min | Max | Start Value | End Value | Δ Value | Δ % | Start Step | End Step |
|---|---|---|---|---|---|---|---|---|
| logs/train | 0.8431 | 1.7541 | 1.7541 | 0.8761 | -0.8780 | -50.06% | 1 | 255 |
| logs/test | 0.5870 | 2.7364 | 2.7364 | 0.5870 | -2.1494 | -78.55% | 1 | 255 |

Figure 4.5: DenseNet-201 Refinement Network Loss

### 4.2.3 IoU Score

The graph 4.6, It shows the Intersection over Union (IoU) metrics of the Refiner component in a span of 250 epochs.Here, in the y-axis we see the values of the IoU and x-axis shows the number of complete epochs which indicates the cycles of training completed. From the graph we can see the constant rise of value of IoU which is initialized at 0.337 and finalized at 0.6340 at the end of the training session part. Through this trend it can

be comprehended that the model prediction increases in precision as the epochs passes by. Furthermore, the consistency and steadily increase in the values of IoU can be understood as a stable training process as there is almost no sign of abrupt changes and can further be inferred as the model is improving its ability to forecast with increasing accuracy as the number of epochs increases. Lastly, the significant rise in IoU reaching a threshold of 89.98 percent shows the notable improvement of the model to generate 3D reconstructions from 2D.



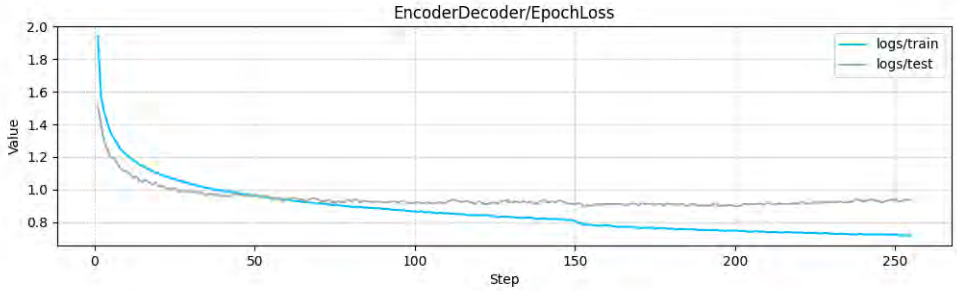

| Run | Min | Max | End Value | ΔValue | Δ% | Start Step | End Step |
|---|---|---|---|---|---|---|---|
| DenseNet201 | 0.3337 | 0.6372 | 0.6340 | 0.3003 | 89.98% | 1 | 255 |

Figure 4.6: DenseNet-201 IoU Score

## 4.3 Xception

### 4.3.1 Extractor-Generator Loss

The graph 4.7, It shows the loss of the Extractor-Generator component during both testing and also training stages in a span of 250 epochs. Here, in the y-axis we see the values of the losses which corresponds to the amount of error and the x-axis shows the number of complete epochs which indicates the cycles of training completed. At the start of training, the value of loss is 1.9396 and than it goes through a very significant decline to a value of 0.7194 which is achieved at the end of the training phase. This trend signifies that the model is gradually becoming efficient in gaining knowledge and also improving its ability to make strong predictions as the epochs progresses. Furthermore, the starting testing loss value goes from 1.5105 to 0.9377 indicating that the model is applicable to unfamiliar and unseen data. In addition, we can see that the both the training and testing loss curves converges which greatly shows that the model is very good at generalization and the most likely there is no form of overfitting as both value of curves gradually decreases.




| Run | Min | Max | Start Value | End Value | Δ Value | Δ % | Start Step | End Step |
|---|---|---|---|---|---|---|---|---|
| logs/test | 0.8934 | 1.5105 | 1.5105 | 0.9377 | -0.5728 | -37.92% | 1 | 255 |
| logs/train | 0.7194 | 1.9396 | 1.9396 | 0.7194 | -1.2203 | -62.91% | 1 | 255 |

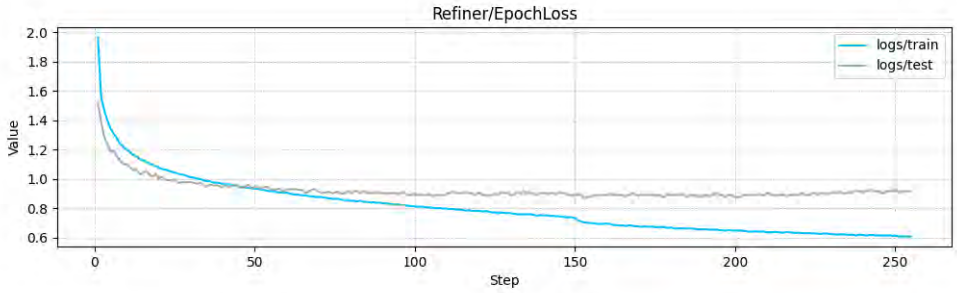
Figure 4.7: Xception Extractor-Generator Loss

### 4.3.2 Refiner Loss

The graph 4.8, It shows the loss of the Refiner component during both testing and also training stages in a span of 255 epochs. Here, in the y-axis we see the values of the losses which corresponds to the amount of error and the x-axis shows the number of complete epochs which indicates the cycles of training completed.At the start of training, the value of loss is 1.5121 and than it goes through a very significant decline to a value of 0.9162 which is achieved at the end of the training phase. This trend signifies that the model is gradually becoming efficient in gaining knowledge and also improving its ability to make strong predictions as the epochs progresses. Furthermore, the starting testing loss value goes from 1.9670 to 0.6063 indicating that the model is applicable to unfamiliar and unseen data. In addition, we can see that the both the training and testing loss curves converges which greatly shows that the model is very good at generalization and the most likely there is no form of overfitting as both value of curves gradually decreases.

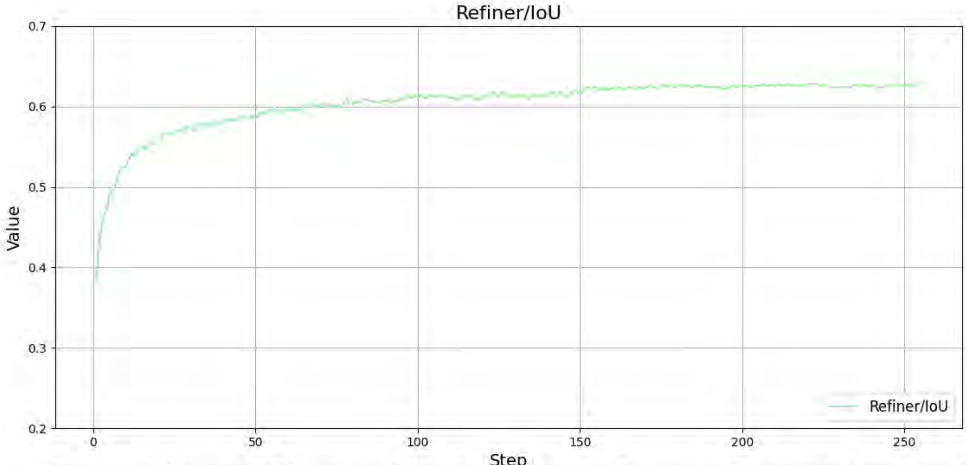
### 4.3.3 IoU Score

The graph 4.9, it shows the Intersection over Union (IoU) metrics of the Refiner component in a span of 255 epochs. Here, in the y-axis we see the values of the IoU and x-axis shows the number of complete epochs which indicates the cycles of training completed. From the graph we can see the constant rise of value of IoU which is initialized at 0.3841 and finalized at 0.6303 at the end of the training session part, signifying a strong improvement of 64.11 percent. An IoU score of 0.6303 can be inferred as a significant indicator that the refiner component enhances the resolution and also the intricate details of the 3D volumes. Achieving and preserving such a value of IoU is very important for maintain the quality of the 3D volume refinement and boosting the overall proficiency of the 3D generation model.




| Run | Min | Max | Start Value | End Value | Δ Value | Δ % | Start Step | End Step |
|---|---|---|---|---|---|---|---|---|
| logs/train | 0.8651 | 1.5121 | 1.5121 | 0.9162 | -0.5959 | -39.41% | 1 | 255 |
| logs/test | 0.6063 | 1.9670 | 1.9670 | 0.6063 | -1.3608 | -69.18% | 1 | 255 |

Figure 4.8: Xception Refinement Network Loss



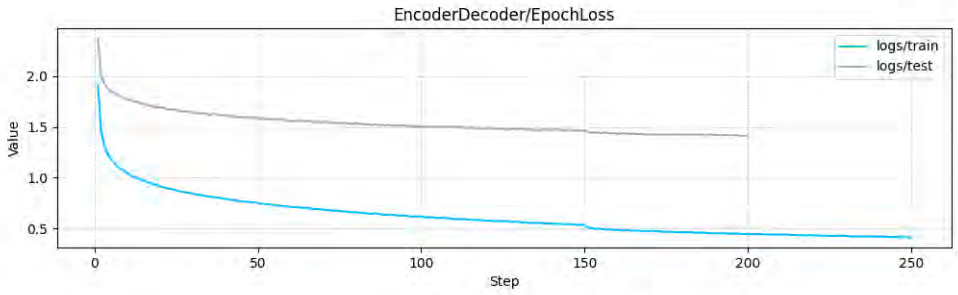

| Run | Min | Max | End Value | ΔValue | Δ% | Start Step | End Step |
|---|---|---|---|---|---|---|---|
| XceptionNet | 0.3841 | 0.6303 | 0.6303 | 0.2462 | 64.11% | 1 | 255 |

Figure 4.9: Xception IoU Score

## 4.4 MobileNet-V3

### 4.4.1 Extractor-Generator Loss

The graph 4.10, it shows the loss of the Extractor-Generator component during both testing and also training stages in a span of 200 epochs. Here, in the y-axis we see the values of the losses which corresponds to the amount of error and the x-axis shows the number of complete epochs which indicates the cycles of training completed.At the start of training, the value of loss is 1.9123 and than it goes through a very significant decline to a value of 0.4093 which is achieved at the end of the training phase. This trend signifies that the model is gradually becoming efficient in gaining knowledge and also improving its ability to make strong predictions as the epochs progresses. Furthermore, the starting testing loss value goes from 2.3756 to 1.4160 indicating that the model is applicable to unfamiliar and unseen data. In addition, we can see that the both the training and testing loss curves converges which greatly shows that the model is very good at generalization and the most likely there is no form of overfitting as both value of curves gradually decreases.



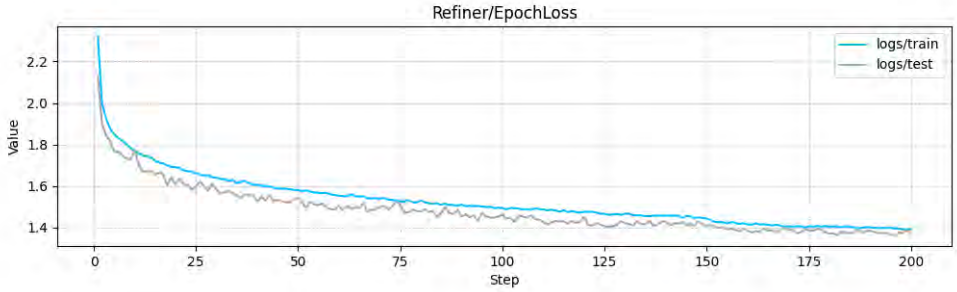

| Run | Min | Max | Start Value | End Value | Δ Value | Δ % | Start Step | End Step |
|---|---|---|---|---|---|---|---|---|
| logs/test | 1.4110 | 2.3756 | 2.3756 | 1.4160 | -0.9595 | -40.39% | 1 | 200 |
| logs/train | 0.4093 | 1.9123 | 1.9123 | 0.4093 | -1.5030 | -78.60% | 1 | 250 |

Figure 4.10: MobileNet-V3 Extractor-Generator Loss

### 4.4.2 Refiner Loss

The graph 4.11, it shows the loss of the Refiner component during both testing and also training stages in a span of 200 epochs. Here, in the y-axis we see the values of the losses which corresponds to the amount of error and the x-axis shows the number of complete epochs which indicates the cycles of training completed. At the start of training, the value of loss is 2.1293 and than it goes through a very significant decline to a value of 1.3740 which is achieved at the end of the training phase.This trend signifies that the model is gradually becoming efficient in gaining knowledge and also improving its ability to make strong predictions as the epochs progresses. Furthermore, the starting testing loss value goes from 2.3208 to 1.3949 indicating that the model is applicable to unfamiliar and unseen data. In addition, we can see that the both the training and testing loss curves

converges which greatly shows that the model is very good at generalization and the most likely there is no form of overfitting as both value of curves gradually decreases.




| Run | Min | Max | Start Value | End Value | Δ Value | Δ % | Start Step | End Step |
|---|---|---|---|---|---|---|---|---|
| logs/train | 1.3593 | 2.1293 | 2.1293 | 1.3740 | -0.7553 | -35.47% | 1 | 200 |
| logs/test | 1.3900 | 2.3208 | 2.3208 | 1.3949 | -0.9258 | -39.89% | 1 | 200 |

Figure 4.11: MobileNet-V3 Refinement Network Loss

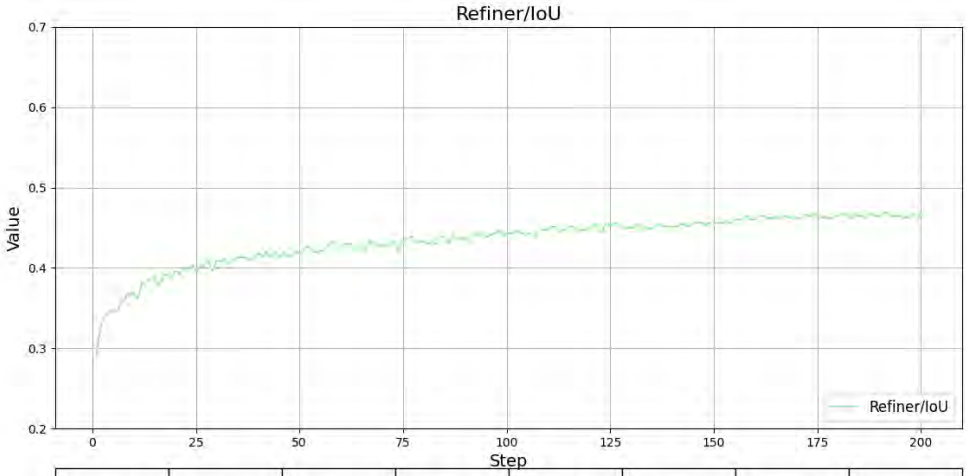
### 4.4.3 IoU Score

The graph 4.12, it shows the Intersection over Union (IoU) metrics of the Refiner component in a span of 200 epochs. Here, in the y-axis we see the values of the IoU and x-axis shows the number of complete epochs which indicates the cycles of training complete. From the graph we can see the constant rise of value of IoU which is initialized at 0.2911 and finalized a0.4661 at the end of the training session part.

## 4.5 EfficientNet-B0

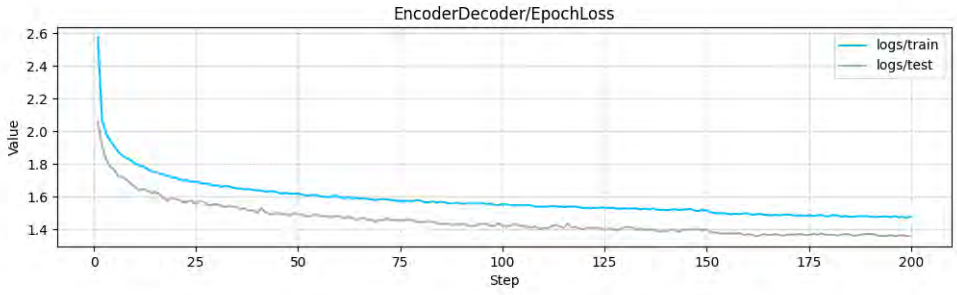
### 4.5.1 Extractor-Generator Loss

The graph 4.13, it shows the loss of the Extractor-Generator component during both testing and also training stages in a span of 200 epochs. Here, in the y-axis we see the values of the losses which corresponds to the amount of error and the x-axis shows the number of complete epochs which indicates the cycles of training completed. At the start of training, the value of loss is 2.5761 and than it goes through a very significant decline to a value of 1.4783 which is achieved at the end of the training phase. This trend signifies that the model is gradually becoming efficient in gaining knowledge and also improving its ability to make strong predictions as the epochs progresses. Furthermore, the starting testing loss value goes from 2.0586 to 1.3585 indicating that the model is applicable to unfamiliar and unseen data. In addition, we can see that the both the training and testing loss curves converges which greatly shows that the model is very good at generalization and the most likely there is no form of overfitting as both value of curves gradually decreases.




| Run | Min | Max | End Value | ΔValue | Δ% | Start Step | End Step |
|---|---|---|---|---|---|---|---|
| MobileNetV3 | 0.2911 | 0.4690 | 0.4661 | 0.1750 | 60.11% | 1 | 200 |

Figure 4.12: MobileNet-V3 IoU Score

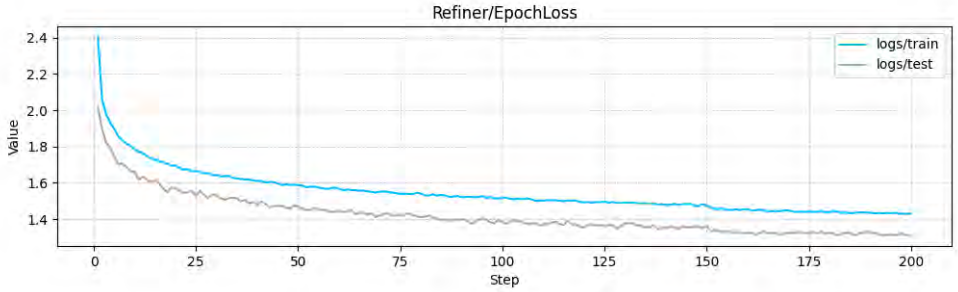



| Run | Min | Max | Start Value | End Value | Δ Value | Δ % | Start Step | End Step |
|---|---|---|---|---|---|---|---|---|
| logs/test | 1.3567 | 2.0586 | 2.0586 | 1.3585 | -0.7001 | -34.01% | 1 | 200 |
| logs/train | 1.4719 | 2.5761 | 2.5761 | 1.4783 | -1.0978 | -42.61% | 1 | 200 |

Figure 4.13: EfficientNet-B0 Extractor-Generator Loss

### 4.5.2 Refiner Loss

The graph 4.14, it shows the loss of the Refiner component during both testing and also training stages in a span of 200 epochs. Here, in the y-axis we see the values of the losses which corresponds to the amount of error and the x-axis shows the number of complete epochs which indicates the cycles of training completed. At the start of training, the value of loss is 2.0214 and than it goes through a very significant decline to a value of 1.3075 which is achieved at the end of the training phase. This trend signifies that the model is gradually becoming efficient in gaining knowledge and also improving its ability to make strong predictions as the epochs progresses.Furthermore,The starting testing loss value goes from 2.4068 to 1.4342 indicating that the model is applicable to unfamiliar and unseen data. In addition, we can see that the both the training and testing loss curves converges which greatly shows that the model is very good at generalization and the most likely there is no form of overfitting as both value of curves gradually decreases and the model is improving with time as the loss values decreases.

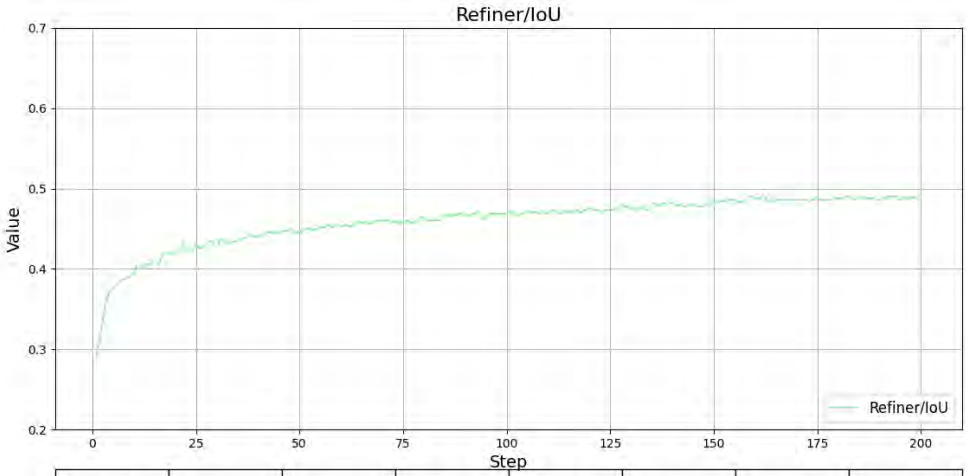



| Run | Min | Max | Start Value | End Value | Δ Value | Δ % | Start Step | End Step |
|---|---|---|---|---|---|---|---|---|
| logs/train | 1.3075 | 2.0214 | 2.0214 | 1.3075 | -0.7139 | -35.32% | 1 | 200 |
| logs/test | 1.4266 | 2.4068 | 2.4068 | 1.4342 | -0.9726 | -40.41% | 1 | 200 |

Figure 4.14: EfficientNet-B0 Refinement Network Loss

### 4.5.3 IoU Score

The graph 4.15, it shows the Intersection over Union (IoU) metrics of the Refiner component in a span of 200 epochs.Here,In the y-axis we see the values of the IoU and x-axis shows the number of complete epochs which indicates the cycles of training completed. From the graph we can see the constant rise of value of IoU which is initialized at 0.2908 and finalized at 0.4906 at the end of the training session part. Here, the value of IoU improves by 68.75 percent but the IoU value falls below the 0.5 category which indicates that this needs much more improvement. In order to get much more intricate and detailed 3D volume refinement and to improve the overall efficiency it is absolute necessary to improve the IoU above the criterion value of 0.5.




| Run | Min | Max | End Value | ΔValue | Δ% | Start Step | End Step |
|---|---|---|---|---|---|---|---|
| EfficientNetV0 | 0.2908 | 0.4925 | 0.4906 | 0.1999 | 68.75% | 1 | 200 |

Figure 4.15: EfficientNet-B0 IoU Score

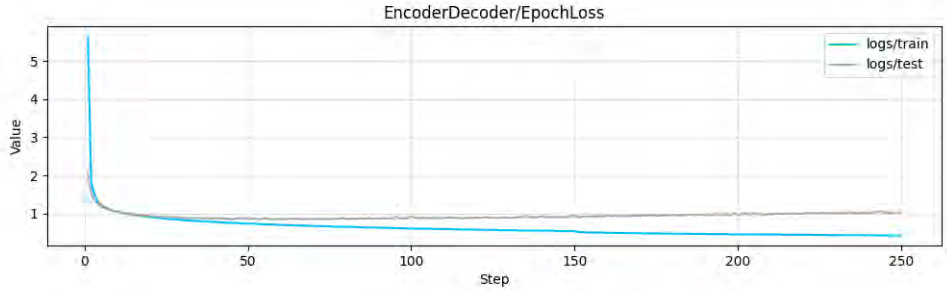
## 4.6 ResNet50

### 4.6.1 Extractor-Generator Loss

The graph 4.16, it shows the loss of the Extractor-Generator component during both testing and also training stages in a span of 250 epochs. Here, in the y-axis we see the values of the losses which corresponds to the amount of error and the x-axis shows the number of complete epochs which indicates the cycles of training completed. At the start of training, the value of loss is 5.6535 and than it goes through a very significant decline to a value of 0.4179 which is achieved at the end of the training phase. This trend signifies that the model is gradually becoming efficient in gaining knowledge and also improving its ability to make strong predictions as the epochs progresses. Furthermore, the starting testing loss value goes from 2.0506 to 1.0358 indicating that the model is applicable to unfamiliar and unseen data. In addition, we can see that the both the training and testing loss curves converges which greatly shows that the model is very good at generalization and the most likely there is no form of overfitting as both value of curves gradually decreases.

### 4.6.2 Refiner Loss

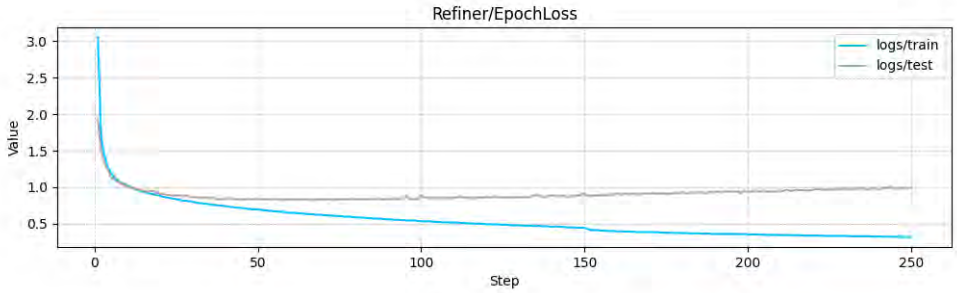
The graph 4.17, it shows the loss of the Refiner component during both testing and also training stages in a span of 250 epochs. Here, in the y-axis we see the values of the losses which corresponds to the amount of error and the x-axis shows the number of complete epochs which indicates the cycles of training completed.At the start of training, the value of loss is 1.9329 and than it goes through a very significant decline to a value of 0.9999 which is achieved at the end of the training phase. This trend signifies that the model is gradually becoming efficient in gaining knowledge and also improving its ability to make strong predictions as the epochs progresses. Furthermore, the starting testing loss




| Run | Min | Max | Start Value | End Value | Δ Value | Δ % | Start Step | End Step |
|---|---|---|---|---|---|---|---|---|
| logs/test | 0.8474 | 2.0506 | 2.0506 | 1.0358 | -1.0148 | -49.49% | 1 | 250 |
| logs/train | 0.4179 | 5.6535 | 5.6535 | 0.4179 | -5.2355 | -92.61% | 1 | 250 |

Figure 4.16: ResNet50 Extractor-Generator Loss

value goes from 3.0535 to 0.3164 indicating that the model is applicable to unfamiliar and unseen data. In addition, we can see that the both the training and testing loss curves converges which greatly shows that the model is very good at generalization and the most likely there is no form of overfitting as both value of curves gradually decreases and the model is improving with time as the loss values decreases.



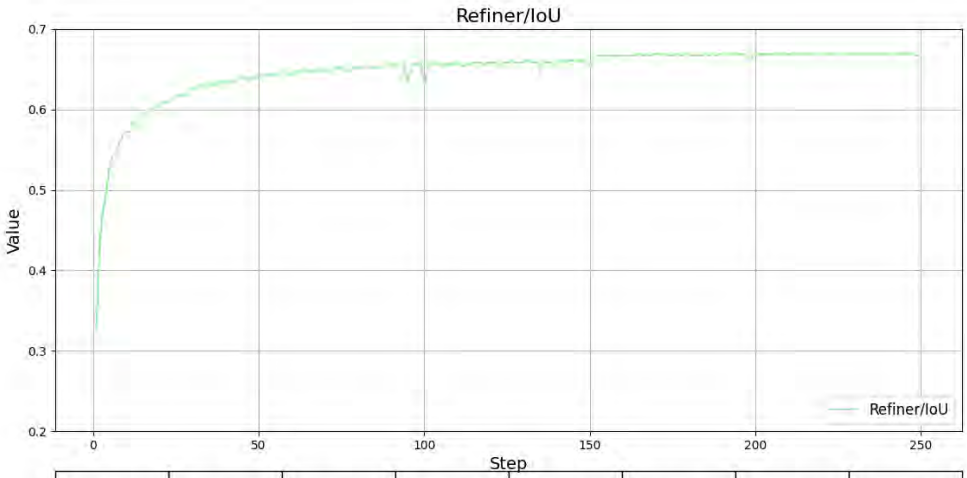

| Run | Min | Max | Start Value | End Value | Δ Value | Δ % | Start Step | End Step |
|---|---|---|---|---|---|---|---|---|
| logs/train | 0.8225 | 1.9329 | 1.9329 | 0.9999 | -0.9330 | -48.27% | 1 | 250 |
| logs/test | 0.3164 | 3.0535 | 3.0535 | 0.3164 | -2.7371 | -89.64% | 1 | 250 |

Figure 4.17: ResNet50 Refinement Network Loss

### 4.6.3 IoU Score

The graph 4.18, it shows the Intersection over Union (IoU) metrics of the Refiner component in a span of 250 epochs. Here, in the y-axis we see the values of the IoU and x-axis shows the number of complete epochs which indicates the cycles of training completed. From the graph we can see the constant rise of value of IoU which is initialized at

0.3279 and finalized at 0.6684 at the end of the training session part,Signifying a strong improvement of 103.85 percentage.
The final value of IoU which is 0.6684 gives a strong indication that the refiner component is pushing the resolution of the 3D regeneration and also improving the intricate level of details in 3D volumes.Furthermore, gaining and conserving such a high value of IoU score is of utmost importance to enhance the 3D volume refinement of intricate details and boosting the overall performance of 2D to 3D model.



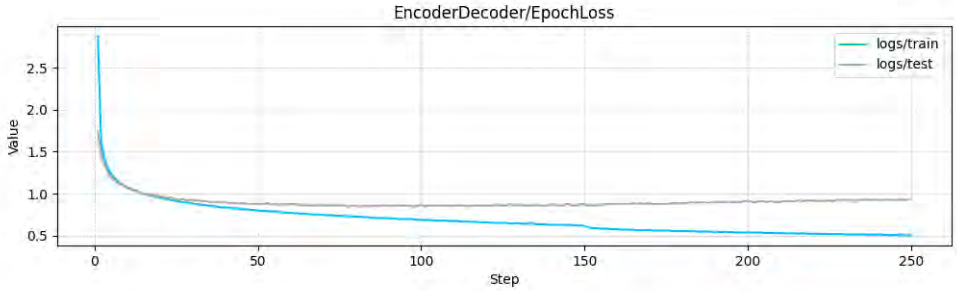

| Run | Min | Max | End Value | ΔValue | Δ% | Start Step | End Step |
|---|---|---|---|---|---|---|---|
| ResNet50 | 0.3279 | 0.6702 | 0.6684 | 0.3405 | 103.85% | 1 | 250 |

Figure 4.18: ResNet50 IoU Score

## 4.7 ShuffleNet-V2

### 4.7.1 Extractor-Generator Loss

The graph 4.19, it shows the loss of the Extractor-Generator component during both testing and also training stages in a span of 250 epochs. Here, in the y-axis we see the values of the losses which corresponds to the amount of error and the x-axis shows the number of complete epochs which indicates the cycles of training completed. At the start of training, the value of loss is 2.8771 and than it goes through a very significant decline to a value of 0.5008 which is achieved at the end of the training phase. This trend signifies that the model is gradually becoming efficient in gaining knowledge and also improving its ability to make strong predictions as the epochs progresses. Furthermore, the starting testing loss value goes from 1.7427 to 0.9360 indicating that the model is applicable to unfamiliar and unseen data. In addition, we can see that the both the training and testing loss curves converges which greatly shows that the model is very good at generalization and the most likely there is no form of overfitting as both value of curves gradually decreases.




| Run | Min | Max | Start Value | End Value | Δ Value | Δ % | Start Step | End Step |
|---|---|---|---|---|---|---|---|---|
| logs/test | 0.8425 | 1.7427 | 1.7427 | 0.9360 | -0.8067 | -46.29% | 1 | 250 |
| logs/train | 0.5008 | 2.8771 | 2.8771 | 0.5008 | -2.3763 | -82.59% | 1 | 250 |

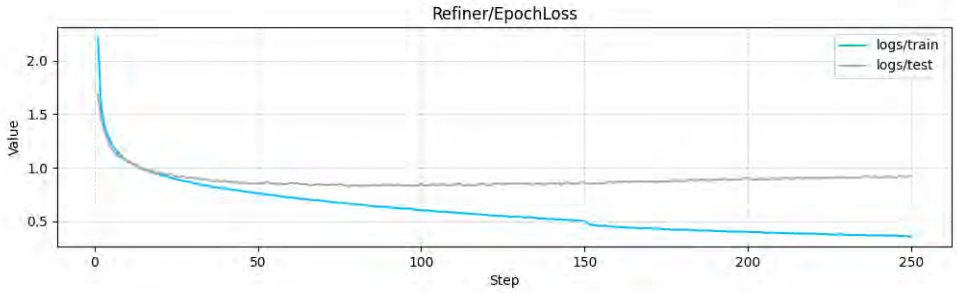
Figure 4.19: ShuffleNet-V2 Extractor-Generator Loss

### 4.7.2 Refiner Loss

The graph 4.20, it shows the loss of the Refiner component during both testing and also training stages in a span of 250 epochs. Here, in the y-axis we see the values of the losses which corresponds to the amount of error and the x-axis shows the number of complete epochs which indicates the cycles of training completed.At the start of training, the value of loss is 1.6812 and than it goes through a very significant decline to a value of 0.9239 which is achieved at the end of the training phase. This trend signifies that the model is gradually becoming efficient in gaining knowledge and also improving its ability to make strong predictions as the epochs progresses. Furthermore, the starting testing loss value goes from 2.2173 to 0.3592 indicating that the model is applicable to unfamiliar and unseen data. In addition, we can see that the both the training and testing loss curves converges which greatly shows that the model is very good at generalization and the most likely there is no form of overfitting as both value of curves gradually decreases and the model is improving with time as the loss values decreases.

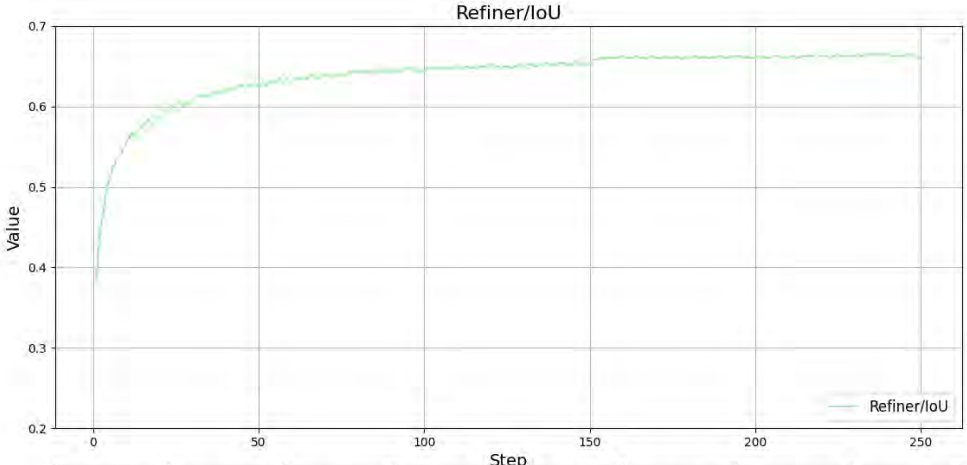
### 4.7.3 IoU Score

The graph 4.21, it shows the Intersection over Union (IoU) metrics of the Refiner component in a span of 250 epochs. Here, in the y-axis we see the values of the IoU and x-axis shows the number of complete epochs which indicates the cycles of training completed. From the graph we can see the constant rise of value of IoU which is initialized at 0.3846 and finalized at 0.6622 at the end of the training session part,Signifying a strong improvement of 72.18 percentage.
The final value of IoU which is 0.6622 gives a strong indication that the refiner component is pushing the resolution of the 3D regeneration and also improving the intricate level of details in 3D volumes.Furthermore,Gaining and Conserving such a high value of IoU score is of utmost importance to enhance the 3D volume refinement of intricate details and boosting the overall performance of 2D to 3D model.




| Run | Min | Max | Start Value | End Value | Δ Value | Δ % | Start Step | End Step |
|---|---|---|---|---|---|---|---|---|
| logs/train | 0.8238 | 1.6812 | 1.6812 | 0.9239 | -0.7574 | -45.05% | 1 | 250 |
| logs/test | 0.3592 | 2.2173 | 2.2173 | 0.3592 | -1.8582 | -83.80% | 1 | 250 |

Figure 4.20: ShuffleNet-V2 Refinement Network Loss



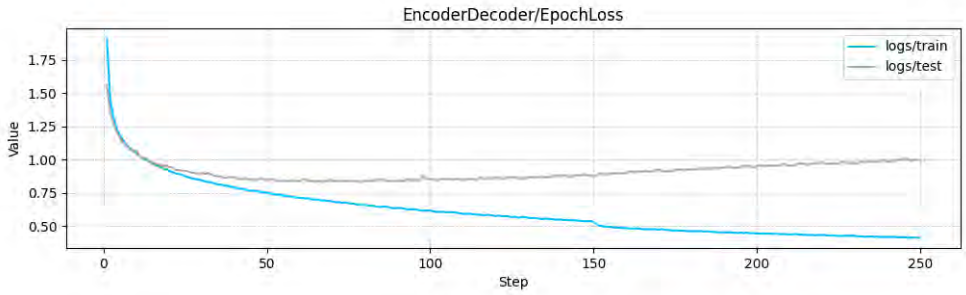

| Run | Min | Max | End Value | ΔValue | Δ% | Start Step | End Step |
|---|---|---|---|---|---|---|---|
| ShuffleNet | 0.3846 | 0.6650 | 0.6622 | 0.2776 | 72.18% | 1 | 250 |

Figure 4.21: ShuffleNet-V2 IoU Score

## 4.8 SMATNet(ShapeNet Dataset)

### 4.8.1 Extractor-Generator Loss

The graph 4.22, it shows the loss of the Extractor-Generator component during both testing and also training stages in a span of 250 epochs. Here, in the y-axis we see the values of the losses which corresponds to the amount of error and the x-axis shows the number of complete epochs which indicates the cycles of training completed. At the start of training, the value of loss is 1.9123 and than it goes through a very significant decline to a value of 0.4093 which is achieved at the end of the training phase. This trend signifies that the model is gradually becoming efficient in gaining knowledge and also improving its ability to make strong predictions as the epochs progresses. Furthermore, the starting testing loss value goes from 1.5608 to 0.9899 indicating that the model is applicable to unfamiliar and unseen data. In addition, we can see that the both the training and testing loss curves converges which greatly shows that the model is very good at generalization and the most likely there is no form of overfitting as both value of curves gradually decreases.



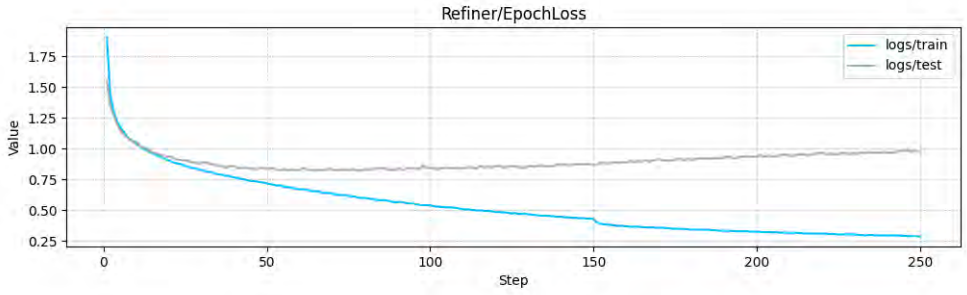

| Run | Min | Max | Start Value | End Value | Δ Value | Δ % | Start Step | End Step |
|---|---|---|---|---|---|---|---|---|
| logs/test | 0.8306 | 1.5608 | 1.5608 | 0.9899 | -0.5708 | -36.57% | 1 | 250 |
| logs/train | 0.4093 | 1.9123 | 1.9123 | 0.4093 | -1.5030 | -78.60% | 1 | 250 |

Figure 4.22: SMATNet Extractor-Generator Loss

### 4.8.2 Refiner Loss

The graph 4.23, it shows the loss of the Refiner component during both testing and also training stages in a span of 250 epochs. Here, in the y-axis we see the values of the losses which corresponds to the amount of error and the x-axis shows the number of complete epochs which indicates the cycles of training completed. At the start of training, the value of loss is 1.5502 and than it goes through a very significant decline to a value of 0.9731 which is achieved at the end of the training phase. This trend signifies that the model is gradually becoming efficient in gaining knowledge and also improving its ability to make strong predictions as the epochs progresses. Furthermore, the starting testing loss value goes from 1.9042 to 0.2861 indicating that the model is applicable to unfamiliar and unseen data. In addition, we can see that the both the training and testing loss curves

converges which greatly shows that the model is very good at generalization and the most likely there is no form of overfitting as both value of curves gradually decreases and the model is improving with time as the loss values decreases.




| Run | Min | Max | Start Value | End Value | Δ Value | Δ % | Start Step | End Step |
|---|---|---|---|---|---|---|---|---|
| logs/train | 0.8192 | 1.5502 | 1.5502 | 0.9731 | -0.5771 | -37.23% | 1 | 250 |
| logs/test | 0.2861 | 1.9042 | 1.9042 | 0.2861 | -1.6181 | -84.97% | 1 | 250 |

Figure 4.23: SMATNet Refinement Network Loss

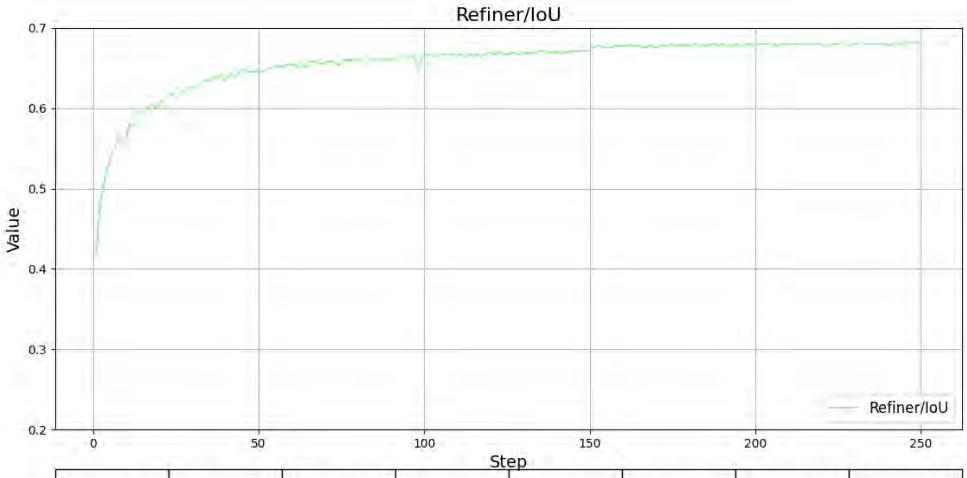
### 4.8.3 IoU Score

The graph 4.24, it shows the Intersection over Union (IoU) metrics of the Refiner component in a span of 250 epochs. Here, in the y-axis we see the values of the IoU and x-axis shows the number of complete epochs which indicates the cycles of training completed. From the graph we can see the constant rise of value of IoU which is initialized at 0.4203 and finalized at 0.6828 at the end of the training session part,Signifying a strong improvement of 62.45 percentage. The final value of IoU which is 0.6828 gives a strong indication that the refiner component is pushing the resolution of the 3D regeneration and also improving the intricate level of details in 3D volumes. Furthermore, Gaining and Conserving such a high value of IoU score is of utmost importance to enhance the 3D volume refinement of intricate details and boosting the overall performance of 2D to 3D model.

## 4.9 SMATNet(Our Dataset)

### 4.9.1 Extractor-Generator Loss

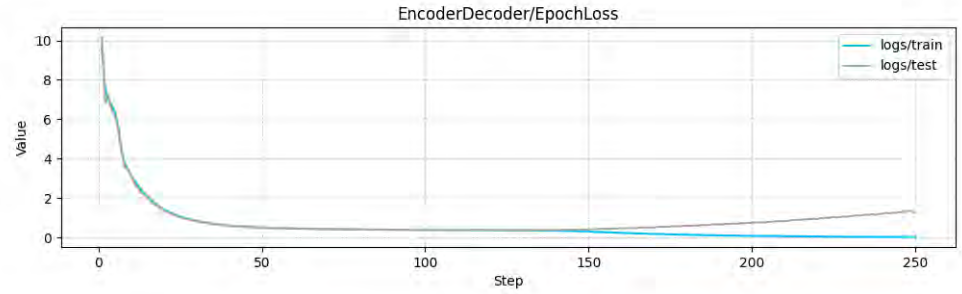
The graph 4.25, it shows the loss of the Extractor-Generator component during both testing and also training stages in a span of 250 epochs. Here, in the y-axis we see the values of the losses which corresponds to the amount of error and the x-axis shows the number of complete epochs which indicates the cycles of training completed. At the start of training, the value of loss is 9.5838 and than it goes through a very significant decline to a value of 0.0357 which is achieved at the end of the training phase. This trend signifies that the model is gradually becoming efficient in gaining knowledge and also improving




| Run | Min | Max | End Value | ΔValue | Δ% | Start Step | End Step |
|---|---|---|---|---|---|---|---|
| Ensembling | 0.4203 | 0.6828 | 0.6828 | 0.2625 | 62.45% | 1 | 250 |

Figure 4.24: SMATNet IoU Score

its ability to make strong predictions as the epochs progresses. In addition, we can see that the both the training and testing loss curves converges which greatly shows that the model is very good at generalization and the most likely there is no form of overfitting as both value of curves gradually decreases.



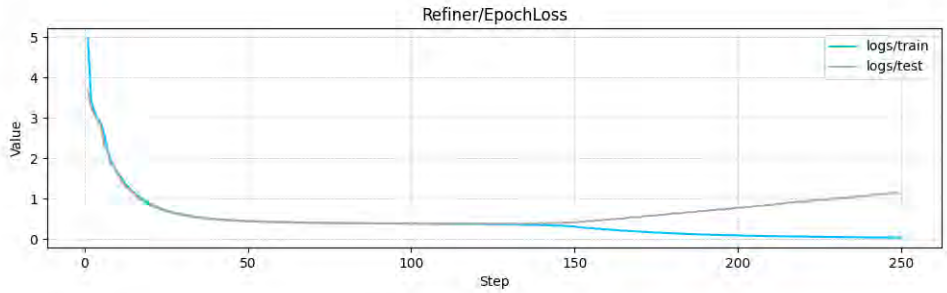

| Run | Min | Max | Start Value | End Value | Δ Value | Δ % | Start Step | End Step |
|---|---|---|---|---|---|---|---|---|
| logs/test | 0.3858 | 10.1552 | 10.1552 | 1.2521 | -8.9030 | -87.67% | 1 | 250 |
| logs/train | 0.0325 | 9.5838 | 9.5838 | 0.0357 | -9.5481 | -99.63% | 1 | 250 |

Figure 4.25: SMATNet Extractor-Generator Loss

### 4.9.2 Refiner Loss

The graph 4.26, it shows the loss of the Refiner component during both testing and also training stages in a span of 250 epochs. Here, in the y-axis we see the values of the losses

which corresponds to the amount of error and the x-axis shows the number of complete epochs which indicates the cycles of training completed. At the start of training, the value of loss is 3.6651 and than it goes through a very significant decline which is achieved at the end of the training phase. This trend signifies that the model is gradually becoming efficient in gaining knowledge and also improving its ability to make strong predictions as the epochs progresses. Furthermore, the starting testing loss value goes from 4.9723 to 0.0366, we can see that the both the training and testing loss curves converges which greatly shows that the model is very good at generalization and the most likely there is no form of overfitting as both value of curves gradually decreases and the model is improving with time as the loss values decreases.

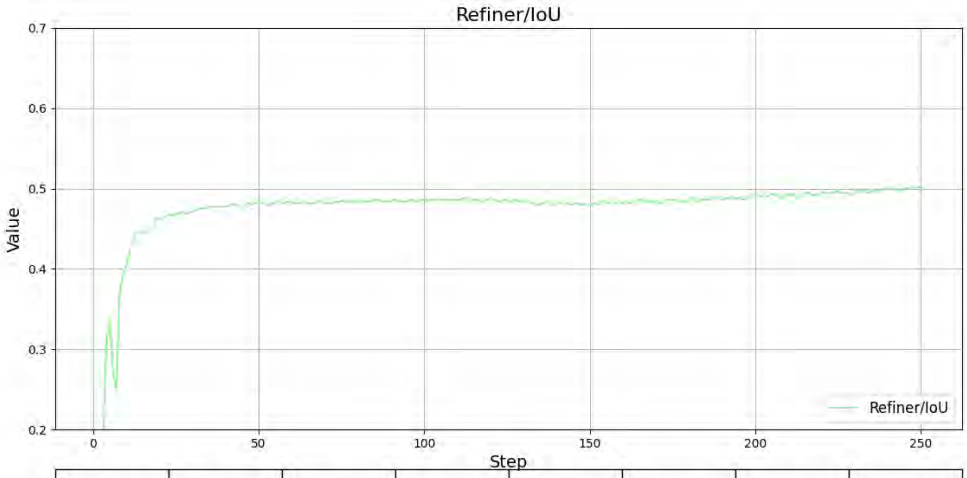



| Run | Min | Max | Start Value | End Value | Δ Value | Δ % | Start Step | End Step |
|---|---|---|---|---|---|---|---|---|
| logs/train | 0.3756 | 3.6651 | 3.6651 | 1.1166 | -2.5485 | -69.53% | 1 | 250 |
| logs/test | 0.0341 | 4.9723 | 4.9723 | 0.0366 | -4.9357 | -99.26% | 1 | 250 |

Figure 4.26: SMATNet Refinement Network Loss

### 4.9.3 IoU Score

The graph 4.27, it shows the Intersection over Union (IoU) metrics of the Refiner component in a span of 250 epochs. Here, in the y-axis we see the values of the IoU and x-axis shows the number of complete epochs which indicates the cycles of training completed. From the graph we can see the constant rise of value of IoU which is initialized at 0.0404 and finalized at 0.5016 at the end of the training session part, signifying a strong improvement of 1141.46 percentage. The final value of IoU which is 0.6828 gives a strong indication that the refiner component is pushing the resolution of the 3D regeneration and also improving the intricate level of details in 3D volumes.

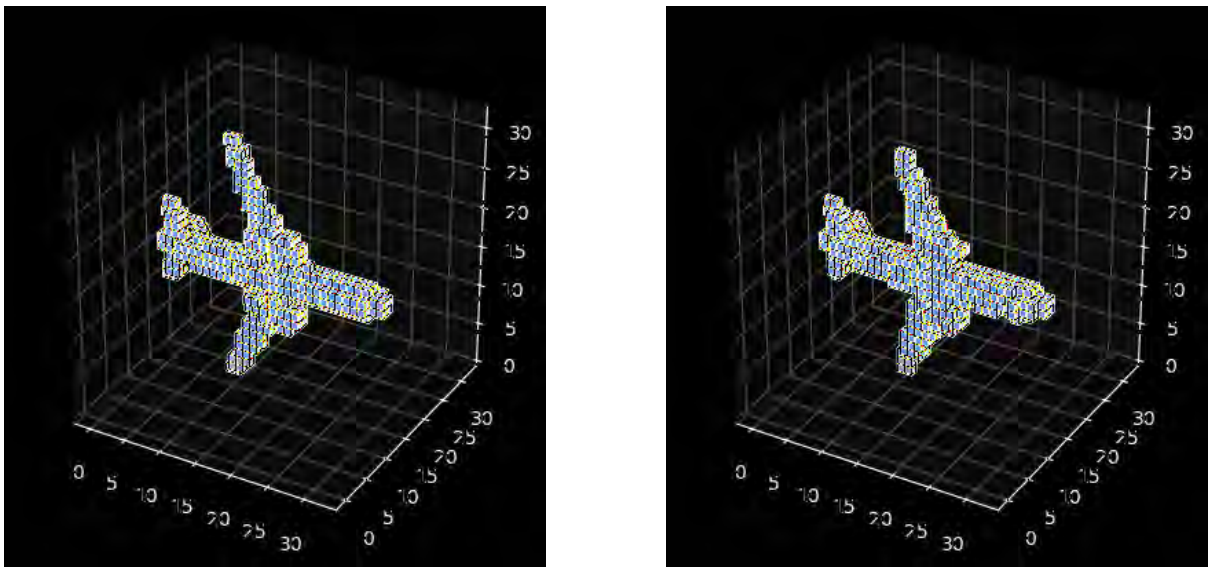



| Run | Min | Max | End Value | ΔValue | Δ% | Start Step | End Step |
|---|---|---|---|---|---|---|---|
| SMATNet | 0.0404 | 0.5017 | 0.5016 | 0.4612 | 1141.46% | 1 | 250 |

Figure 4.27: SMATNet IoU Score

## 4.10 Ground Truth vs Reconstructed Volume

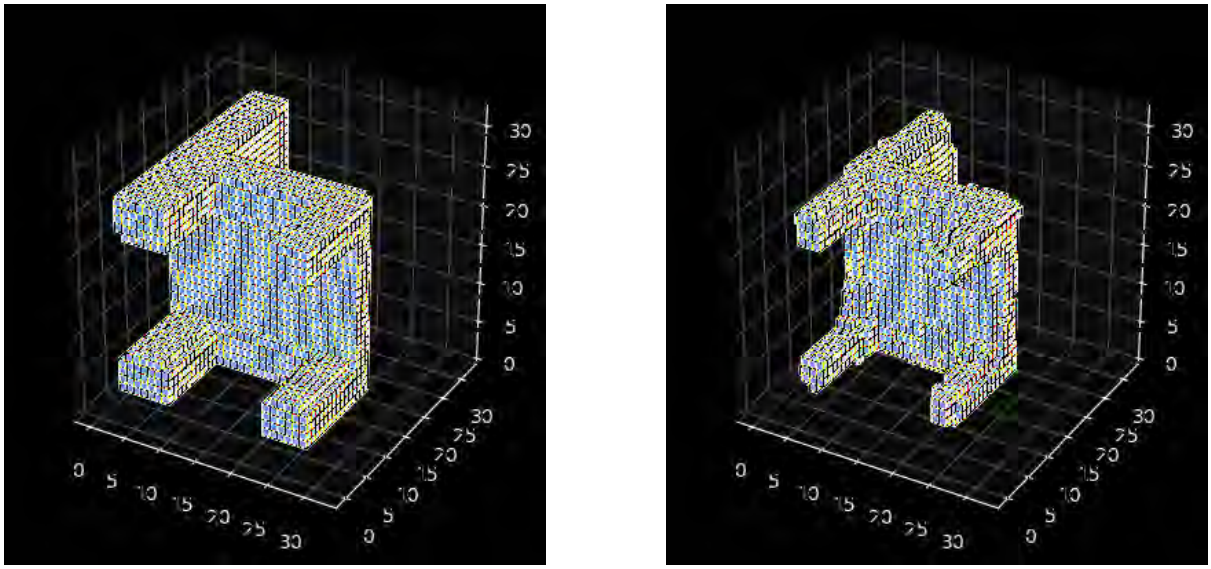
### 4.10.1 VGG16



(a) Ground Truth (b) Reconstructed Model

Figure 4.28: Comparison of Ground Truth and Reconstructed Model

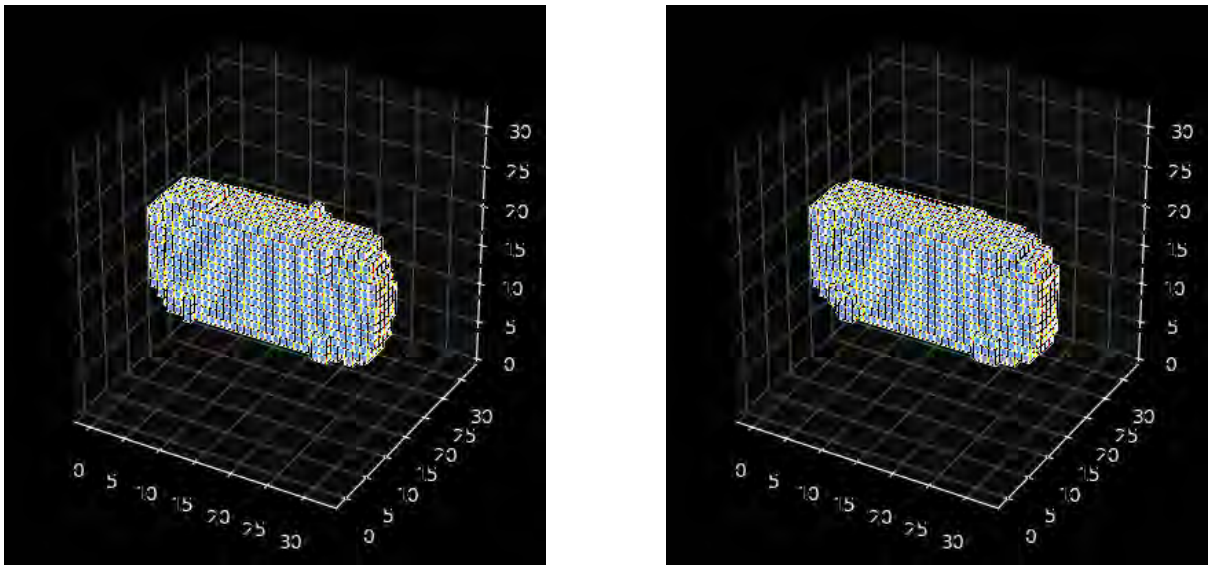
### 4.10.2 DenseNet-201



(a) Ground Truth (b) Reconstructed Model

Figure 4.29: Comparison of Ground Truth and Reconstructed Model

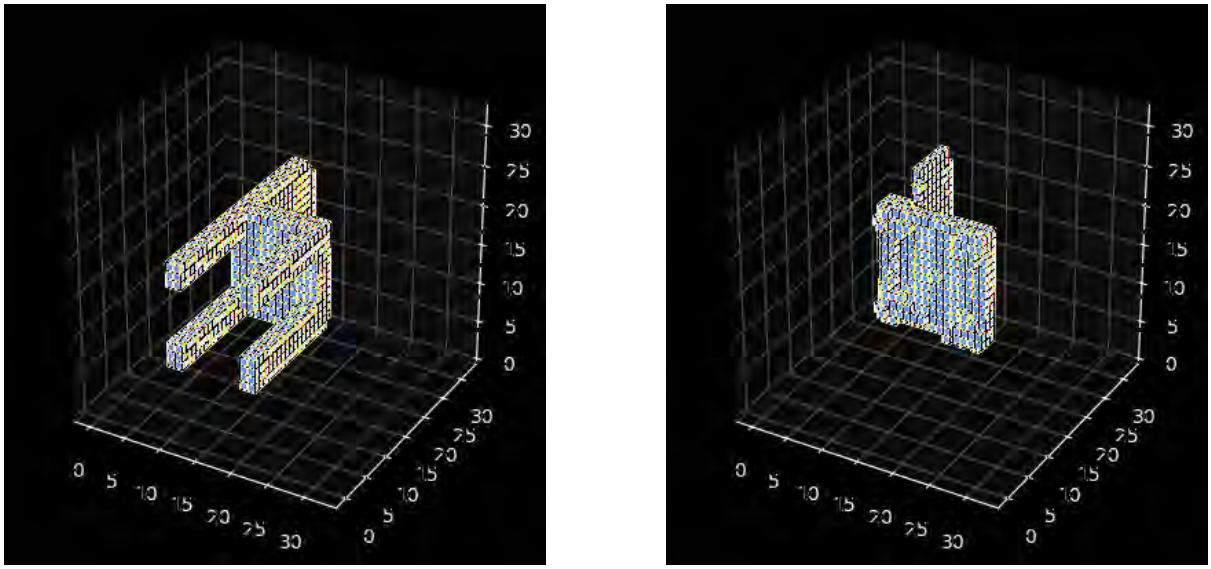
### 4.10.3 Xception



(a) Ground Truth (b) Reconstructed Model

Figure 4.30: Comparison of Ground Truth and Reconstructed Model

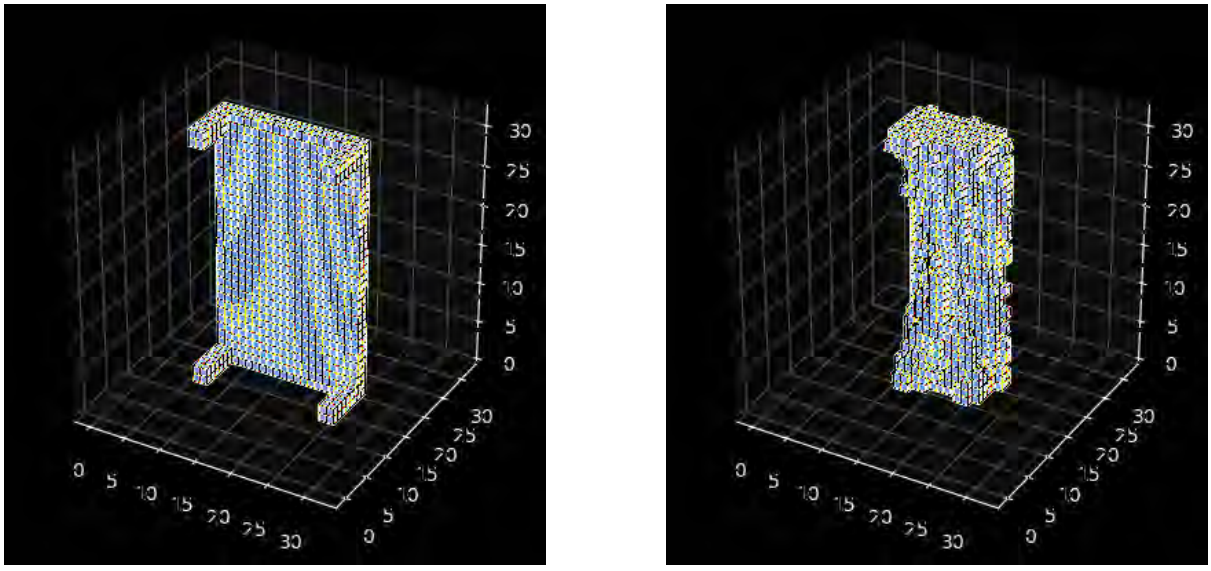
### 4.10.4 MobileNet-V3



(a) Ground Truth (b) Reconstructed Model

Figure 4.31: Comparison of Ground Truth and Reconstructed Model

### 4.10.5 EfficientNet-B0



(a) Ground Truth (b) Reconstructed Model

Figure 4.32: Comparison of Ground Truth and Reconstructed Model

### 4.10.6 ResNet50

(a) Ground Truth (b) Reconstructed Model

Figure 4.33: Comparison of Ground Truth and Reconstructed Model

### 4.10.7 ShuffleNet-V2

(a) Ground Truth (b) Reconstructed Model

Figure 4.34: Comparison of Ground Truth and Reconstructed Model

### 4.10.8 SMATNet(ShapeNet Dataset)

(a) Ground Truth (b) Reconstructed Model

Figure 4.35: Comparison of Ground Truth and Reconstructed Model

### 4.10.9 SMATNet(Our Dataset)

(a) Ground Truth (b) Reconstructed Model

Figure 4.36: Comparison of Ground Truth and Reconstructed Model

## 4.11 Comparison

The comparison of IoU scores among different models highlights significant variances in performance. The DenseNet201 model demonstrates a substantial enhancement, with the initial Intersection over Union (IoU) value of 0.3337 increasing to a final IoU of 0.6340. This is an impressive 89.98 percent improvement, indicating excellent performance. The initial performance of EfficientNetV0 was 0.2908, then it improved to 0.4906, resulting in a 68.75 percent increase, which is considered suboptimal. The MobileNetV3 model demonstrates a 60.11 percent improvement, as indicated by an initial Intersection over Union (IoU) value of 0.2911 and a final IoU value of 0.4661. However, this improvement is considered suboptimal. ShuffleNet has a significant increase of 72.18 percent, starting at 0.3846 and reaching 0.6622, which demonstrates excellent performance. The VGG16 model, starting at an initial value of 0.4166 and reaching a final value of 0.6315, demonstrates a significant improvement of 51.57 percent, which is highly commendable. The

XceptionNet model demonstrated a significant improvement, increasing from an initial accuracy of 0.3841 to 0.6303, representing a remarkable 64.11 percent gain. This performance may be considered highly commendable. SMATNet demonstrates a notable improvement, with an initial Intersection over Union (IoU) of 0.0404 and a final IoU of 0.5016. This is an outstanding increase of 1141.46 percent, which is highly commendable. The ResNet50 model demonstrates a significant increase of 103.85 percent, as it progresses from an initial value of 0.3279 to a final value of 0.6684. This development is highly commendable. SMATNet (ShapeNet) demonstrates a significant improvement, with an initial IoU (Intersection over Union) of 0.4203 and a final IoU of 0.6828, resulting in a 62.45 percent increase, which is highly commendable. Generally, models that achieve a final Intersection over Union (IoU) score of 0.60 exhibit excellent performance, whilst those with a score below 0.60 are considered to have poor performance.

Table 4.1: Comparison of Initial and Final IoU values for different models

| **Model** | **Initial IoU** | **Final IoU** | **Δ IoU** | **Δ %** | **Evaluation** |
|---|---|---|---|---|---|
| DenseNet201 | 0.3337 | 0.6340 | 0.3003 | 89.98% | Optimal |
| EfficientNetB0 | 0.2908 | 0.4906 | 0.1999 | 68.75% | Sub optimal |
| MobileNetV3 | 0.2911 | 0.4661 | 0.1750 | 60.11% | Sub optimal |
| ShuffleNet | 0.3846 | 0.6622 | 0.2776 | 72.18% | Optimal |
| VGG16 | 0.4166 | 0.6315 | 0.2149 | 51.57% | Optimal |
| XceptionNet | 0.3841 | 0.6303 | 0.2462 | 64.11% | Optimal |
| ResNet50 | 0.3279 | 0.6684 | 0.3405 | 103.85% | Optimal |
| SMATNet(ShapeNet) | 0.4203 | 0.6828 | 0.2625 | 62.45% | Optimal |
| SMATNet(Our Dataset) | 0.0404 | 0.5016 | 0.4612 | 1141.46% | Optimal |

If we categorize by the value of IoU, Resnet (IoU=0.6684), ShuffleNet(IoU=0.6622) and SMATNet(IoU=0.6828) all showing excellent capability in improvising the IoU, models such as DenseNet201(IoU=0.6340), VGG16(IoU=0.6315) and XceptionNet(IoU=0.6303) all attained finalized value of IoU ranging from 0.63 to 0.65, can be considered as noteworthy but not the best category. The rest of the models such as EfficientNetB0(IoU=0.4960) and MobileNetV3(IoU=0.4661) achieves sub-optimal IoU scores below 0.63 and can be considered worst of all the categories and thus can be inferred as not recommended and absolutely not suitable for the dataset in consideration due to their low IoU values. From this comparison we can say that, the best performing category consisting of ResNet, ShuffleNet and SMATNet are the most proficient models among all the models and there is scope for development for other models.

## 4.12 Limitations

2D to 3D model conversion is a computationally challenging task that requires a significant amount of time and resources, particularly computational resources such as GPUs and processing units. Due to constraints such as the availability of GPUs and scheduling

issues, our model produced suboptimal results. Furthermore, we faced significant challenges in scaling the dataset, as it was not feasible to collect a large number of dataset samples within such a short time window, and the quality of the dataset dipped due to not being able to collect a large number of samples of the same object. Additionally, preprocessing the collected dataset from one type to another continuously drained our already scarce time and resources.

Our Research model also faced some concerning limitations including increased number of parameters due to stacked ensembling, increased resource consumption which is the result of increase in number of parameters. On top of this, there was the issue of computationally expensive Multi-view image processing and also the need of precise preprocessing which is an precondition for 3D model generation.

# Chapter 5

# Conclusion

Since every gadget is now automated, computer vision has emerged as a priority research area for the next generation of technology. The process of converting from 2D to 3D models can be considered a fundamental component of any computer vision application that requires 3D reconstruction, as capturing 3D models directly can sometimes be expensive compared to going from 2D to 3D. The applications of our research range from medical imaging, such as prosthetics and implants, to industry-level automation and applications. Our study's ultimate goal is to provide a straightforward, accurate, and inexpensive approach to convert from 2D to 3D models, which further lays the foundation for advanced research in the future.

# Bibliography


[1] K. N. Kutulakos and S. M. Seitz, "What do n photographs tell us about 3d shape," *University of Rochester, TR680*, vol. 29, 1998.

[2] I. Shimshoni, Y. Moses, and M. Lindenbaum, "Shape reconstruction of 3d bilaterally symmetric surfaces," *International Journal of Computer Vision*, vol. 39, pp. 97–110, 2000.

[3] B. K. Hall, "Unlocking the black box between genotype and phenotype: Cell condensations as morphogenetic (modular) units," *Biology and Philosophy*, vol. 18, pp. 219–247, 2003.

[4] S. Battiato, S. Curti, M. La Cascia, M. Tortora, and E. Scordato, "Depth map generation by image classification," in *Three-Dimensional Image Capture and Applications VI*, SPIE, vol. 5302, 2004, pp. 95–104.

[5] A. Geiger, J. Ziegler, and C. Stiller, "Stereoscan: Dense 3d reconstruction in real-time," in *2011 IEEE intelligent vehicles symposium (IV)*, Ieee, 2011, pp. 963–968.

[6] S. Izadi, D. Kim, O. Hilliges, *et al.*, "Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera," in *Proceedings of the 24th annual ACM symposium on User interface software and technology*, 2011, pp. 559–568.

[7] D. Maier, A. Hornung, and M. Bennewitz, "Real-time navigation in 3d environments based on depth camera data," in *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, IEEE, 2012, pp. 692–697.

[8] A. Baak, M. M"uller, G. Bharaj, H.-P. Seidel, and C. Theobalt, "A data-driven approach for real-time full body pose reconstruction from a depth camera," *Consumer Depth Cameras for Computer Vision: Research Topics and Applications*, pp. 71–98, 2013.

[9] Y. M. Mustafah, A. W. Azman, and M. H. Ani, "Object distance and size measurement using stereo vision system," *Advanced Materials Research*, vol. 622, pp. 1373–1377, 2013.

[10] A. Anwer, A. Baig, and R. Nawaz, "Calculating real world object dimensions from kinect rgb-d image using dynamic resolution," in *2015 12th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, IEEE, 2015, pp. 198–203.

[11] E. Bostanci, "3d reconstruction of crime scenes and design considerations for an interactive investigation tool," *International Journal of Information Security Science*, vol. 4, no. 2, pp. 50–58, 2015.

[12] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, IEEE, 2015, pp. 922–928.

[13] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[14] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[15] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[16] R. A. Ralph, *Applications of 3d reconstruction in the real world by rsip vision*, RSIP Vision, Aug. 2017.

[17] A. El Sallab, I. Sobh, M. Zidan, M. Zahran, and S. Abdelkarim, "Yolo4d: A spatio-temporal approach for real-time multi-object detection and classification from lidar point clouds," 2018.

[18] N. Ma, X. Zhang, H. Zheng, and J. Sun, "Shufflenet V2: practical guidelines for efficient CNN architecture design," *CoRR*, vol. abs/1807.11164, 2018. arXiv: 1807.11164. [Online]. Available: http://arxiv.org/abs/1807.11164%7D.

[19] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "Shufflenet v2: Practical guidelines for efficient cnn architecture design," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 116–131.

[20] H. Qassim, A. Verma, and D. Feinzimer, "Compressed residual-vgg16 cnn model for big data places image recognition," in *2018 IEEE 8th annual computing and communication workshop and conference (CCWC)*, IEEE, 2018, pp. 169–175.

[21] Z. Ren and E. B. Sudderth, "3d object detection with latent support surfaces," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 937–946.

[22] A. Howard, M. Sandler, G. Chu, *et al.*, *Searching for mobilenetv3*, 2019. arXiv: 1905.02244 [cs.CV].

[23] M. Simon, K. Amende, A. Kraus, *et al.*, "Complexer-yolo: Real-time 3d object detection and tracking on semantic point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.

[24] K. Simonson, *Associated General Contractors of America*, Mar. 2019, 2019.

[25] D. Sinha and M. El-Sharkawy, "Thin mobilenet: An enhanced mobilenet architecture," in *2019 IEEE 10th annual ubiquitous computing, electronics & mobile communication conference (UEMCON)*, IEEE, 2019, pp. 0280–0285.

[26] V. Tadic, A. Odry, I. Kecskes, E. Burkus, Z. Kiraly, and P. Odry, "Application of intel realsense cameras for depth image generation in robotics," *WSEAS Transac. Comput*, vol. 18, pp. 2224–2872, 2019.

[27] H. Xie, H. Yao, X. Sun, S. Zhou, and S. Zhang, "Pix2vox: Context-aware 3d reconstruction from single and multi-view images," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 2690–2698.

[28] S.-H. Wang and Y.-D. Zhang, "Densenet-201-based deep neural network with composite learning factor and precomputation for multiple sclerosis classification," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 16, no. 2s, pp. 1–19, 2020.

[29] T. Ahmed and N. H. N. Sabab, *Classification and understanding of cloud structures via satellite images with efficientunet*, 2021. arXiv: 2009.12931 [eess.IV].

[30] B. Deng, C. R. Qi, M. Najibi, T. Funkhouser, Y. Zhou, and D. Anguelov, "Revisiting 3d object detection from an egocentric perspective," *Advances in Neural Information Processing Systems*, vol. 34, pp. 26 066–26 079, 2021.

[31] P. Gohel, P. Singh, and M. Mohanty, "Explainable ai: Current status and future directions," *arXiv preprint arXiv:2107.07045*, 2021.

[32] N. Ketkar, J. Moolayil, N. Ketkar, and J. Moolayil, "Introduction to pytorch," *Deep learning with python: learn best practices of deep learning models with PyTorch*, pp. 27–91, 2021.

[33] Z. Lu, "Construction of the 3d reconstruction system of building construction scene based on deep learning," *Scientific Programming*, vol. 2021, pp. 1–9, 2021.

[34] S. Mascarenhas and M. Agarwal, "A comparison between vgg16, vgg19 and resnet50 architecture frameworks for image classification," in *2021 International conference on disruptive technologies for multi-disciplinary research and applications (CENTCON)*, IEEE, vol. 1, 2021, pp. 96–99.

[35] N. Siddique, S. Paheding, C. P. Elkin, and V. Devabhaktuni, "U-net and its variants for medical image segmentation: A review of theory and applications," *Ieee Access*, vol. 9, pp. 82 031–82 057, 2021.

[36] K. Srinivasan, L. Garg, D. Datta, *et al.*, "Performance comparison of deep cnn models for detecting driver's distraction," *Cmc -Tech Science Press-*, vol. 68, pp. 4109–4124, May 2021. DOI: 10.32604/cmc.2021.016736.

[37] M. Akhtar, R. Mahum, F. Shafique Butt, *et al.*, "A robust framework for object detection in a traffic surveillance system," *Electronics*, vol. 11, p. 3425, Oct. 2022. DOI: 10.3390/electronics11213425.

[38] D. Alma, "Gan Architectures for 3D Models," vol. 2, no. 7, Apr. 2022.

[39] F. Drews, D. Feng, F. Faion, L. Rosenbaum, M. Ulrich, and C. Gl"aser, "Deepfusion: A robust and modular 3d object detector for lidars, cameras and radars," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2022, pp. 560–567.

[40] P. Kadam, M. Zhang, S. Liu, and C.-C. J. Kuo, "R-pointhop: A green, accurate, and unsupervised point cloud registration method," *IEEE Transactions on Image Processing*, vol. 31, pp. 2710–2725, 2022.

[41] X. Chen, X. Pu, Z. Chen, *et al.*, "Application of efficientnet-b0 and gru-based deep learning on classifying the colposcopy diagnosis of precancerous cervical lesions," *Cancer Medicine*, vol. 12, no. 7, pp. 8690–8699, 2023.

[42] Unknown Author, *ResNet50 Network Architecture Diagram*, Wikimedia Commons, [Accessed 19-May-2024], 2023. [Online]. Available: https://commons.wikimedia.org/wiki/File:ResNet50.png.

[43] GeekssforGeeks, *U-net architecture explained*, https://www.geeksforgeeks.org/u-net-architecture-explained/, Accessed: 2024-01-15.

[44] J. McDermott, *Hands-on transfer learning with keras and the vgg16 model*, https://www.learndatasci.com/tutorials/hands-on-transfer-learning-keras/, Accessed: 2024-01-15.