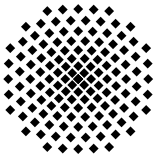

University of Stuttgart
Institute for Signal Processing and System Theory
Professor Dr.-Ing. B. Yang

Research Project S1376

LiDAR Point Cloud Up-sampling using Adversarial Networks

Author: Janaranjani Palaniswamy

Date of work begin: 14.12.2020
Date of submission: 14.06.2021

Supervisor: Mr. George Basem Fouad Eskandar

Keywords: LiDAR point cloud, Up-sampling, GAN

Abstract

Building models capable of processing the real-time sensor data to facilitate the tasks in autonomous driving is a major challenge. The denser LiDAR point cloud plays a vital role in assisting the self-driving vehicles. As the LiDAR point clouds are sparse, noisy, and irregular in nature, it is necessary to transform it by up-sampling to obtain a dense point cloud. Recent work has focused on predicting high-resolution point clouds with good perceptual and semantic information. The resulting estimates often lacks high-frequency geometric details and may not match the expected fidelity. In this project, we present a Generative Adversarial Network (GAN) based approach to generate super-resolution of point clouds. To the best of our knowledge, it is the first GAN framework for LiDAR up-sampling using geometric losses. To achieve this purpose, the point cloud is projected onto the 2D range image and its resolution is increased. The generator network is designed accordingly to increase the vertical resolution of the point cloud. In addition, we recommend using feature matching, Chamfer Distance and Virtual Normal loss as additional loss functions, to encourage the network to improve the quality of the results. Further, an improved GAN architecture with customized padding is proposed to produce sharp edges in the point cloud distribution. Finally, we recommend using the proposed model with Chamfer Distance loss to encourage the output to be more similar to the reference. In addition, Virtual Normal loss is used to drive the solution to achieve good geometric features.

In the extensive range of experiments using various loss functions based on the distance, feature maps and geometric constraints we evaluate the results in both point cloud and range image formats. The qualitative and quantitative analysis of the results shows that the geometric and perceptual quality are significantly improved compared to the state-of-the-art models. Our approach generates uniformly distributed points while preserving the geometric structure.

Contents

1. Introduction **1**
1.1. Motivation and Objective . . . 2
1.2. Contributions . . . 3
1.3. Organization . . . 3

2. Related Work **5**
2.1. Baseline Methods . . . 8

3. Method **13**
3.1. Technical Approach . . . 13
3.2. Data Processing . . . 14
3.3. Network Architecture . . . 16
3.3.1. Generator Network . . . 16
3.3.2. Discriminator Network . . . 17
3.4. Loss Functions . . . 21
3.4.1. Content Loss . . . 21
3.4.2. Perceptual Loss . . . 21
3.4.3. Total Variation Loss . . . 22
3.4.4. Adversarial Loss . . . 22
3.4.5. Discriminator Loss . . . 22
3.4.6. Proposed Loss Functions . . . 23
3.5. Improved Model Architecture . . . 25

4. Experimental Setup and Results **29**
4.1. Dataset . . . 29
4.2. Training Details . . . 29
4.3. Evaluation Metrics . . . 30
4.3.1. Earth Mover's Distance (EMD) . . . 30
4.3.2. Chamfer Distance (CD) . . . 31
4.3.3. Root Mean Squared Error (RMSE) . . . 31
4.3.4. Peak Signal to Noise Ratio (PSNR) . . . 31
4.4. Training Methodology . . . 32
4.5. Results . . . 34
4.5.1. Baseline Results . . . 34
4.5.2. Proposed Model . . . 38
4.5.3. Quantitative Results . . . 41

5. Conclusion **45**
5.1. Future Work . . . 45

A. Acronyms **47**

List of Figures **49**

List of Tables **51**

Bibliography **53**

1. Introduction

In autonomous driving, perception of the environment is a complex task, where the vehicle navigates autonomously in a highly dynamic environment. In order to achieve this goal, vehicles are equipped with various sensors to help in perceiving the low-level information from the environment in real time. Various computer vision, machine learning, and deep learning algorithms enable the high-level scene understanding. Generally, sensors such as camera, radar and LiDAR are commonly used to obtain the information about other traffic participants. The deep learning models demands large amounts of high-resolution data to effectively understand the scene.

Light Detection and Ranging (LiDAR) sensors plays a vital role in the areas of autonomous driving. While, other sensors like, camera helps in sensing the objects at a far distance only with sufficient ambient lighting conditions, LiDAR gives a 360° view of the vehicle surrounding irrespective of light conditions. LiDAR gives long-range visibility to the vehicle and it potentially acts as the eye of the self-driving vehicle.

LiDAR scan is generated by periodically emitting laser pulses while rotating around a vertical axis. The output of the LiDAR sensor is a 3D point cloud representing the topology of the vehicle's surroundings. The resulting point clouds are typically irregular and sparse in three dimensional space, and has lower resolution than the camera images. In general, the LiDAR point clouds are required to have high-resolution to provide faithful information about the environment and assist in ego-motion. A typical LiDAR sensor has multiple channels that keep rotating at various elevation angles. The density of the point clouds highly depends on the number of channels in the sensor. However, a LiDAR sensor with higher number of channels is very expensive. To solve this problem, there comes a need for up-sampling the point clouds to produce a denser data for capturing the precise details about the environment.

In practice, due to the irregularity and the sparsity of the LiDAR data, it is highly difficult in increasing the density of the points of LiDAR scans. Typically, the LiDAR processing is done with the help of camera images as reference. Having an RGB camera integrated in the sensor setup is not always possible. In addition, as LiDAR is proficient in providing the distance information about higher ranges, it makes it difficult for the camera images to aid in LiDAR data processing. To circumvent the above issues, recent research activities focuses on using only a single frame LiDAR sensors for processing the distance information.

This research project deals with a fundamental task in environment perception using LiDAR sensors, i.e. point cloud up-sampling and focuses on the utilization of adversarial learning techniques to solve the problem.

The goal of LiDAR point cloud up-sampling is to obtain an up-sampled point cloud that retains the LiDAR structure. This is widely regarded as the first step of LiDAR processing and scene understanding and has shown to contribute in improving the efficiency in many applications such as: object detection, semantic segmentation, and localization and mapping.

1.1. Motivation and Objective

Recently, deep neural networks are used to address the LiDAR point cloud up-sampling problem. Existing network architectures, including PU-Net [1], PU-GAN [2], and PC-GAN [3], have demonstrated the advantage of up-sampling point clouds through learning the point cloud distribution by various techniques. These methods are trained on point cloud data from small objects and shapes. However, these networks involve heavy processing and enforces a complete uniform distribution of points everywhere. In practice, LiDAR point clouds are not uniform in space and hence the above models fail to produce a better LiDAR up-sampling results. As a result, developing a deep learning model to perform LiDAR point cloud up-sampling with high accuracy still remains open for further research.

Processing the 3D point clouds directly with three dimensional convolutions is challenging and computationally heavy without compromising accuracy [4]. Most of the previous LiDAR processing approaches either used raw point clouds or converted the data to voxel grids [5]. Instead, in our work the LiDAR scans are projected into a 2D range map. The 2D range maps preserves the regular structure of the LiDAR point clouds. Thus, two-dimensional convolutional layers are sufficient to process the range maps. The 2D range map representation of the point clouds are more similar to 2D images. Hence, it paves way for making use of existing deep neural architectures designed for image super-resolution.

Existing CNN-based LiDAR super-resolution models use range images to up-sample LiDAR point clouds. Work carried out by Larissa et al. [6] and Shan et al. [7] shows better performance when compared with the model using the raw point cloud. These models can scale well with any range of points in the scan. However, they have limitations in generating good point cloud geometry. The convolutional layer in the network is only used to estimate the low-frequency content of the range image, while blurring or smoothing the edges and boundaries.

With the advancement of deep learning techniques, GANs has become a very powerful network that can generate plausible results from unlabelled data. GANs has been proven to perform better in image super-resolution tasks. A GAN framework has a dual neural network: a generator to learn the rich and diverse point distributions that are very similar to the ground truth, and a discriminator to implicitly evaluate the generated results. This adversarial learning strategy helps to produce more robust and natural variants resembling the ground truth data. Since GANs have adversarial loss, they are highly optimized compared to CNNs.

In order to overcome the ill-effects of convolutional layers, geometric losses can be used in the framework to capture the high-frequency intrinsic structure of the point cloud.

The data from LiDAR sensors possess high horizontal resolution depending on the number of lines in the sensor. On the other side, the vertical resolution is comparatively small and it depends on the number of lines of the sensor. The commonly available LiDAR scans are 16, 32, 64 or 128 lines. The main objective of this work is to up-sample the vertical resolution of LiDAR scans and not the horizontal resolution. This makes the resulting point cloud distribution to resemble a data obtained form a sensor with higher lines or scans.

In this work, a GAN-based network inspired by image super-resolution model [8] is proposed to up-sample the 2D range images. We employ sufficient modifications to the generator and discriminator networks to help the model implicitly evaluate high-resolution range images

against the ground truth. We define a combination of loss functions using perceptual loss, content loss, adversarial loss, Chamfer Distance (CD) loss and Virtual Normal loss (VNL) to foster solutions perceptually and geometrically hard to distinguish from the reference. We name the final model with proposed CD and VNL loss functions as LiDAR Super-Resolution GAN (LSRGAN). The network is trained using open source KITTI [9] dataset. Compared to existing deep learning models, our focus is on GANs ability to generate high-resolution perceptual point clouds. To evaluate the up-sampling quality of the model, we employ four metrics to assess its performance relative to the state-of-the-arts. Extensive experimental results show that our method has substantial performance in terms of distribution uniformity and sharp edges in the distribution.

1.2. Contributions

To the best of our knowledge, we are the first to present a work on up-sampling automotive LiDAR points using GAN-based models. This models generates an up-sampled point clouds using a low-resolution, down-sampled ranges images as input data.

The key contributions are the following:

- LiDAR scans are projected into 2D range images and are up-sampled.
- Proposed a modified super-resolution GAN model for up-sampling the range images.
- Various loss functions are proposed to train the network to learn the latent patterns in the point cloud distributions.
- Enhanced the model architecture using functional padding for mitigating overly smooth point cloud output.
- The developed model is quantitatively evaluated using metrics like EMD, CD and RMSE.
- A comparative study is made by assessing the model performance with state-of-the-arts using KITTI dataset.

1.3. Organization

The rest of this report is organized as follows:

The Chapter 2 discusses about the background about the point cloud up-sampling and point cloud processing techniques. A complete study on previously developed methods on 2D range image representations are presented in this chapter. In addition, the history about the point cloud up-sampling from conventional method to recent deep learning models are described.

Chapter 3 presents the detailed description of the approach proposed to up-sampled the LiDAR point clouds and discusses about the network architecture and various loss functions used.

Chapter 4 illustrates the experimental setup including the preparation of training dataset and implementation details. Meanwhile, it introduces various metrics used for evaluation of the

models. We also summarizes the overall methodology of training. The results obtained from baseline models and the proposed methods are discussed with visual results. Finally, a comparative study is done using quantitative analysis to show the effectiveness of our method.

Chapter 5 gives a summary and concludes the whole research project. Moreover, it gives an outlook to the future works based on the limitations of this project work.

2. Related Work

The aim of LiDAR point cloud up-sampling is to generate a high-resolution point cloud for a given low-resolution input. As we consider the range image representation of 3D LiDAR point clouds, this work is more related to image super-resolution task, which desires to improve the resolution of the images. There are a lot more research works that exists for up-sampling or enhancing the resolution of the RGB images. All the existing research works to the best our knowledge that are related to this task are discussed here starting from LiDAR data processing, point cloud up-sampling techniques and models, and the works related to image super-resolution.

In the following discussion, we first group related works into different categories covering our topic of the LiDAR range image up-sampling using GANs; we finish this section with a discussion of the state-of-the-art baseline models selected for making a comparative analysis.

Lidar Processing. Recently, the deep learning models that uses LiDAR data to extract relevant information about the vehicle environment has been greatly improved. Bo Li [10] uses 3D convolutional layers on the point cloud in voxel grid representation for vehicle detection. The problem with voxel-based LiDAR processing is they are computationally heavy, and it failed to play useful role in leveraging the sparsity of LiDAR point clouds data. Later, voxels are avoided by transforming the LiDAR from its 3D coordinates to spherical ones and projecting them into 2D domain. Bo Li et al. [11] projected the LiDAR scans into 2D range images similar to the depth channel in RGBD images. Here, the coordinates (x, y, z) of the 3D point cloud are represented as azimuth and elevation angles measured from the origin. This can also be seen as projecting the point cloud onto a 2D spherical plane. Each row on the two-dimensional map corresponds to a vertical laser scan, and each column corresponds to the horizontal field of view. Another work done by [12] also uses such mapping of point cloud to 2D spherical plane, in addition, they augment the 2D signal with extra coordinate channels. In this work, they used the method of added 2 channels to every point of the input data. They experimented the generative models like Variational Autoencoder (VAE) and generative adversarial network (GAN) using the LiDAR range images in polar and Cartesian formats and proved that such representations are robust against any input perturbations. This 2D range image approach is been adapted in our method for processing the point clouds.

An alternative approach for processing of LiDAR data is from Ondruska et al. [13]. They convert their input point cloud to an occupancy grid. They train their network to also predict future occupancy grids, thereby creating a generative model for LiDAR data. Comparatively, using 2D range image helps in reducing the preprocessing time and but also enhances the efficient representation of irregular data. Moreover, by using this we can run our model at a much higher resolution, while retaining the computational efficiency.

Optimization-based Up-sampling. Before the advancement in deep learning models,

optimization-based techniques are used for up-sampling. Alexa et al. [14] was the first to work on up-sampling a point set by adding points to the Voronoi diagram vertices. Later, Huang et al. [15] developed a edge-aware re-sampling method for point set. However, parameter tuning impacts the quality of this method. Wu et al. [16] introduced a deep points representation method to fill large holes and complete missing regions. Overall, these optimization-based methods are not data-driven; they heavily rely on shape priors which is not suitable for the recent works in deep learning methods.

Deep-Learning-based Up-sampling. In past decades, various deep neural networks have been designed to learn features point sets directly, instead of using priors for example PointNet [17]. Point cloud up-sampling commonly tries to enhance the uniformity of the point distribution. Yu et al. [1] proposed a network called PU-Net to up-sample point sets. PU-Net uses the point clouds directly for up-sampling. The up-sampling network learns multilevel features per point and it works on patches. Then it expands the point set via multi-branch convolution units by mixing-and-blending point features in the feature space. After expansion the features are split into a multitude of feature sets, which are then reconstructed using Earth Mover's Distance (EMD) loss to an up-sampled point set. The resultant point set from PU-Net is still unordered and forms a generic point cloud. For our use case, it is necessary to retain the order of point cloud sets obtained from real-world LiDAR sensors. However, this method can only perform up-sampling on the point cloud data recorded from small objects rather than the real-time LiDAR scans from autonomous vehicles. There came a need for GAN-based up-sampling framework for generating point sets with higher quality and uniformity.

GAN-based Point Cloud Processing. Goodfellow et al. [18] proposed a generative framework for unsupervised learning called Generative Adversarial Network (GAN). The GAN framework with two networks called generator and discriminator makes. Here, the output point clouds produced from generator is evaluated by the discriminator network. GANs have proven to produce a rich variety of output patterns using various datasets. Achlioptas et al. [19] was the first to adapt the GAN model to operate on raw point clouds for enhancing the representation learning. Later, PU-GAN [2] was the first model to come up with a GAN-based network for point cloud up-sampling. It uses the generator with feature extraction unit to extract the points. The extracted points are expanded by means of up-down-up expansion unit. Finally, the point set generation component improves the uniformity of the point cloud distribution. The discriminator consists of self-attention unit to enhance the feature integration and extraction ability of the model. Performance of PU-GAN is better than PU-Net in generating a uniformly distributed high-resolution point cloud output. PU-GAN model was proposed to work on raw point cloud dataset. This method has also demonstrated a better performance in up-sampling the real-world LiDAR data which is relevant to our case. However, it lacked in learning the uniformity of the 3D data structure and also have limited ability in filling the large gaps or holes in point clouds as it lacks global view of the overall shape of the 3D coordinates. This limitation of PU-GAN model leaves a room for a better GAN-based model for up-sampling any sparse and irregular point clouds from LiDAR scans of ego-motion. In our work, we tried to address these open points.

Image Super-resolution. Let's take a look at the relevant up-sampling works done using

2D images. Traditional approaches such as linear or bicubic interpolation [20], can be very fast and can be easily applied to LiDAR projections. They are more suitable for real-time applications due to their low computational complexity. However, they are not able to recover the fine details of the inputs and often yields more smoother results. With advancements in deep learning-based models, many research works were done in this area and are known for their best performances. In general, deep learning models estimate a high-resolution image from low-resolution images using a huge amount of training data with reference high-resolution images. Techniques like SR-CNN [21], trained a three-layer deep convolutional neural network to increase the image resolution. Though SR-CNN outperforms the previous traditional methods, they cannot cope with missing features in the data, and hence they demand a denser representation of input. Followed by SR-CNN, many complex neural network architectures were proposed to achieve better accuracy in image super-resolution task. One among them is SR-GAN [8] which uses a GAN based framework to achieve a remarkable performance in super-resolution of RGB images which looks more photo-realistic. The generator of SR-GAN followed the deep SRResNet architecture and used 16 residual blocks along with two up-sampling blocks. The model is experimented by reducing the resolution of high-resolution target images using bicubic kernel based down-sampling technique. Such low-resolution image is fed into the generator and up-sampling is done to get the super-resolution image output. The discriminator of SR-GAN uses [22] architecture which is feasible enough to classify generated super-resolution images from the target high-resolution image. SR-GAN model outperforms all other prior methods developed for image super-resolution.

For RGB images, the high-resolution image predictions usually do not appear visually realistic to humans. The reason is predicted high-resolution images are found to have missing high-frequency contents and hence they are perceptually unsatisfying with lack of expected fidelity. Johnson et al. [22] addressed to this issue and proposed an up-sampling network using a perceptual loss based on features extracted from VGG-16 [23]. Among the previously discussed methods, SRGAN uses a perceptual loss function which uses a compound of an adversarial loss and a content loss. The adversarial loss pushes the output to the natural image manifold using a discriminator network. The results of SRGAN for large range of up-scaling factors was proven to be more photo-realistic. The idea of SRGAN using GAN framework with perceptual loss lies at the core of our proposed approach.

LiDAR Super-resolution. Larissa et al. [6] developed a CNN-based model to synthesize a realistic high-resolution LiDAR data. They proposed to follow the image super-resolution approach to achieve an up-sampled LiDAR point cloud. They were the first to employ the perceptual losses for LiDAR applications. In their method, they adapted 2D mapping of the point clouds as discussed earlier in processing data. They employed a modified point-wise and semantic consistency loss functions to preserve the content of the LiDAR scan during up-sampling. Their results proved that perceptual loss enforced the semantic and perceptual realism in the predicted LiDAR scans. This research work showcased the benefits of perceptual loss in LiDAR world as well. Hence, it further served as a base for this work.

A recent work by Shan et al. [7] came up a dedicated framework for LiDAR super-resolution, to predict the high-resolution LiDAR with a low-resolution LiDAR as input. It again utilized

the idea of transforming the 3D LiDAR into 2D image space for improving the resolution. An architecture based on U-Net [24] is used to up-sample the low-resolution range images. The model is trained using simulated dataset and utilizes Monte-Carlo (MC) dropout as a regularization technique. This framework was able to give highly accurate point cloud up-sampling results. Their results were more convincing in comparison to super-resolution frameworks and traditional interpolation methods. In addition, their experiment also involved testing SRGAN model, but no better results were observed. In this report, we consider this problem area and develop a model based on SRGAN and train it using LiDAR range images to achieve better results.

Up-sampling Methods. The main objective of this report is to perform up-sampling. Since the up-sampling of LiDAR is to be performed with 2D range maps. A research was done to look into the up-sampling methods followed by image-based deep learning models. Most of the early works like SR-CNN puts the up-sampling stage at the front of the model. It makes the model very large and in turn costs a lot of time during testing. FSRCNN [25] inserts the up-sampling unit at the end of the model, this results in smaller input size and deeper model. FSRCNN uses deconvolution for up-sampling, while other model on image super-resolution ESRGAN [26] uses nearest interpolation. The sub-pixel method for up-sampling was proposed by ESPCN [27], it has successfully reduced the computation cost for image and video data. Whereas, RFB-ESRGAN [28] alternately uses nearest interpolation and sub-pixel convolution layer. In general, compared to nearest interpolation method, sub-pixel convolution method is more suitable for computation in depth dimensions. SRGAN [8] model also uses sub-pixel convolution method. In our case, sub-pixel convolution method might be useful, as it works on increasing the resolution of the depth map in both horizontal and vertical directions. For LiDAR up-sampling, it required to perform the up-sampling only in vertical direction but not in horizontal direction. So, a better choice for our use would be transposed convolution based on the work done by Wojna et al. [29]. The recent work on LiDAR super-resolution by Shan et al. [7] also used transposed convolutional blocks to increase the image resolution before feeding into the encoder model. Transposed convolutions have fewer and more subtle and enhances the visible artifacts of the depth maps. This preserves the compatibility of our approach in using transpose convolutional layers as an up-scaling method for the input LiDAR scans.

2.1. Baseline Methods

To make a comparative study, five models which had previously worked on point clouds and had shown strong performance are chosen as baselines. They are PU-GAN [2], PU-Net [1], conditional LiDAR generation method [12], CNN-based LiDAR up-sampling model [6], and U-Net [7]. To note that the PU-GAN and PU-Net models are designed to work with the raw point cloud data. For a fair comparison of the baselines with our method, the baseline models are trained with KITTI dataset projected in 2D range map.

PU-Net. This method employed a deep network for expanding the point set data and reconstruct the features. This network first extracts the patches of the points from various 3D models for learning the geometrical patterns of the objects. This is done by sampling patch

points on the surface and generating random points using Poisson disk sampling on each patch. Next, the point features are embedded to learn the feature of the point clouds by mapping the raw 3D coordinates to a feature space using hierarchical feature learning and multi-level feature aggregation techniques. The feature space is used to increase the point cloud resolution. The features in $N \times C_1$ dimension, where N is the number of input points and C is the dimension of the feature space, are expanded using a sub-pixel convolutional layer into $rN \times C_2$. To avoid the reconstructed point clouds being too close to each other, the network uses different convolutions for different feature sets. Finally, this model uses fully connected layers on the features to reconstruct the up-sampled 3D point coordinates. The network is trained using a union of repulsion loss to generated points more uniformly and EMD-based loss function for reconstructing the output points adhering the underlying surface. In this model, once the point features are extracted, the features are combined with nearest neighboring points to generate the up-sampled features. To find the neighboring point features, a ball query with a radius is done. We train these two networks with the reconstruction loss with EMD function and the results are listed. Though the model has better results than PointNet architecture [17], it has many limitations in reconstructing the tiny parts, and handling the irregularity and sparseness of the point cloud data. Due to duplication of point features, the expanded features are more similar to the input, which affects the overall output quality. Therefore, it is impossible for the model to produce convincing results from low-quality, sparse and irregular data.

In our work, the PU-Net is evaluated with the metrics in Section 4.3 using the pretrained model. In addition, the model is re-trained with down-sampling the input range map by removing the layers. The data are then reshaped to a dimension of $N \times 3$ to be compatible with the network architecture. For a uniform evaluation, CD loss is also added to train the model. Then the model performance is analyzed in comparison to our method both qualitatively and quantitatively using EMD and CD metrics.

PU-GAN. A GAN-based framework is used by Li et al. [2] which was shown to outperform the PU-Net model in terms of output uniformity and 3D reconstruction quality. The generator of this model has extracted the features based on the sequence of dense blocks across all the layers as in [30]. An up-down-up unit is used for expanding the extracted features. First, the features are up-sampled and self-corrected for avoiding multi-step training. And then the expanded features are down-sampled by using set of MLPs. A self-attention unit based on SAGAN [31] is used in the generator as well as in discriminator for enhancing the feature integration quality. Finally, the generator uses a point set generation component for improving the uniform distribution of the output from a global perspective. On the other hand, the discriminator uses set of MLPs and max pooling in addition to self-attention unit to produce global features of the point clouds. To improve the uniform generative ability of the network, as uniform loss is added to the generator during training. Similar to PU-Net, this model also uses EMD loss for a perfect reconstruction of the surface. This method demonstrated a better performance in up-sampling the real-scanned LiDAR inputs. However, it was not trained to fill large holes or gaps in the point cloud data due to patch level training.

In our experiment, PU-GAN is trained as baseline on KITTI dataset. Additionally CD loss is used for training. The input point clouds are converted to 2D domain and down-sampled to reduce the resolution. After that a shaping is done to make the data compatible with the PU-GAN model. The results from pretrained model based on raw 3D point cloud data

and re-implemented model with 2D range maps are evaluated using EMD and CD metrics mentioned in Section 4.3.

Deep Generative Model. Caccia et al. [12] proposed the idea of using Deep Generative Models (DGML) like variational autoencoders (VAE) and GAN for generating the LiDAR data similar to RGB images. They nominated the approach of processing the LiDAR point clouds as 2D grid maps based on Li et al. [11] work. They showed that such data representation makes it easy to use the generative model architectures which were developed for RGB images. In our work, only the VAE model is considered as one of the baselines, as it was proven to be more robust against any corruption in the input data. The Deep Convolutional GAN (DCGAN) architecture [22] was used for designing the VAE model. The encoder ϕ setup of VAE partially followed the discriminator architecture of DCGAN to encode the input data x into a latent representation $z = \phi(x; \theta_{\text{enc}})$, where θ_{enc} is the encoder parameter. The decoder ψ replicated the generator of DCGAN to reconstruct the original input conditioned on the latent encoding to obtain $\tilde{x} = \psi(\tilde{z}; \theta_{\text{dec}})$, where θ_{dec} is the decoder parameter. The variational aspect was induced my considering z as a prior distribution. In their work, Gaussian distribution was considered for z. They trained the VAE model both Cartesian and Polar representation of point clouds. The model was tested by corrupting the input data with additive noise and random removal of points from the LiDAR range images. It was able to generated the point cloud closely related with original even with missing or corrupted information.

In our experiment, we adopted their idea of training VAE with missing data in the input range image. Following that the input point cloud in range map format is down-sampled before feeding into the model. The VAE model was re-implemented to generate an up-sampled version of the low-resolution input. As this VAE-based conditional LiDAR generation model didn't use EMD and CD losses for their training, those losses are added during this experiment. The results from this baseline are compared with our method using all three metrics discussed in Section 4.3.

CNN-based High-Resolution LiDAR Synthesis. A CNN-based approach by Larissa et al. [6] used specialized loss functions like modified per-point loss and perceptual loss to synthesize a high-resolution LiDAR scan with high realism in terms of perceptual and semantic information. This CNN-based synthesis of realistic high-resolution LiDAR data method is termed as LiDAR High-Resolution Convolutional Neural Network (LHRCNN). In their method, they reused the architecture of Johnson et al. [32] along with 16 residual blocks. Strided convolutions are used for up-sampling the feature. The model takes a range image of shape $L/2 \times W$ and gives a high-resolution output image of shape $L \times W$. The perceptual loss is calculated based on the features extracted from a pretrained point-wise semantic segmentation network. L_1 loss between the features extracted from ground truth and predicted feature maps gives the perceptual loss. In addition, a semantic consistency loss is also calculated based on the output from pretrained semantic segmentation network.

In this project, this CNN-based model is used as potential baseline with 2D range image in spherical form as input data. During training, the perceptual loss and the semantic consistency loss is not used, as it depends on a pretrained semantic segmentation network. This model is predominantly trained using L_1 point-wise loss and the results are evaluated based on the metrics discussed in Section 4.3.

U-Net. As discussed earlier, the work done by Shan et al. [7] for LiDAR super-resolution is also considered as one of our baseline model. They followed an approach of enhancing the vertical resolution of LiDAR. They adapted the 2D projection of LiDAR point clouds and increased the resolution of range image to obtain a dense point cloud output. The encoder-decoder architecture of U-Net model [24] was configure to process the low-resolution range image. The input range images are up-sampled by two transposed convolutional layers. The encoder is designed with a series of convolutional and average pooling layers to down-sample the spatial resolution of the feature. Meanwhile, the kernel sizes are increased in the encoder. Whereas, the decoder has a reverse structure along with transpose convolutions. At the end, high-resolution range image is obtained which is projected back to 3D domain to obtain the dense point clouds without any information loss. Additionally, to overcome the smoothing effect on sharp and discontinuous boundaries of the range images, a Monte-Carlo dropout (MC-dropout) was applied in between the convolutional layers. During inference time, MC-dropout regularization approximates a Bayesian neural network (BNN) with numerous feed-forward passes with active dropout to produce a normalized distribution.

In our work, we refer to this method of simulation-based LiDAR super-resolution as U-Net model. The results are evaluated using metrics described in Section 4.3.

3. Method

This chapter describes the method used in this research project. The idea of LiDAR point cloud up-sampling using range images is introduced. The GAN-based model used to satisfy the objective of obtaining a high-resolution point cloud is explained elaborately. The loss functions used for training the GAN model are presented. To improvise the learning, additional loss functions and model modifications are proposed.

3.1. Technical Approach

This section describes the proposed model architecture which addresses the problem of LiDAR up-sampling by using adversarial learning techniques. The aim is to estimate an up-sampled point cloud from a sparse input point cloud. As LiDAR point clouds has 3D spatial coordinates, it requires 3D convolutional layers to solve this problem with adversarial networks. By transforming the 3D point clouds into 2D range images, we can solve this problem using image super-resolution techniques. Since the horizontal resolution of LiDAR is normally high enough, we only enhance vertical resolution throughout this work. Thus, we consider the approach of transforming the 3D point clouds into range images and increase the resolution in its height H. Our ultimate goal is to train a generator function G that estimates a high-resolution range image from a low-resolution input range image. To optimize the GAN network, a combination of loss functions are used to attain the desired uniformity and structural perception in the output point clouds.

The workflow of the proposed approach is described in figure 3.1. As shown in the figure, for a given sparse LiDAR point cloud dataset, an equivalent 2D range image is obtained by projecting the 3D point clouds. The range image is down-sampled in the vertical direction and passed into the GAN model. The GAN model is trained exclusively for up-sampling the input range images. Later, the high-resolution output range images can be projected back into 3D point clouds without any information loss.

A detailed discussion about the process of representing the 3D point clouds into 2D range images is included in Section 3.2. The network architecture of the generator and discriminator are inspired by the image super-resolution model SRGAN [8]. The necessary changes are made to the adapted architecture to suit the processing of 2D range images. The detail of the network architecture is presented in Section 3.3. To train the GAN model for this up-sampling task, specific loss functions are required to improve the learning of the generator.The individual loss functions used for this approach are discussed in detail in Section 3.4. To further improve the structural details and uniformity of the point clouds, a few more specific loss functions and a modified GAN architecture with functional pad layers are proposed, which is summarized in Section 3.5.

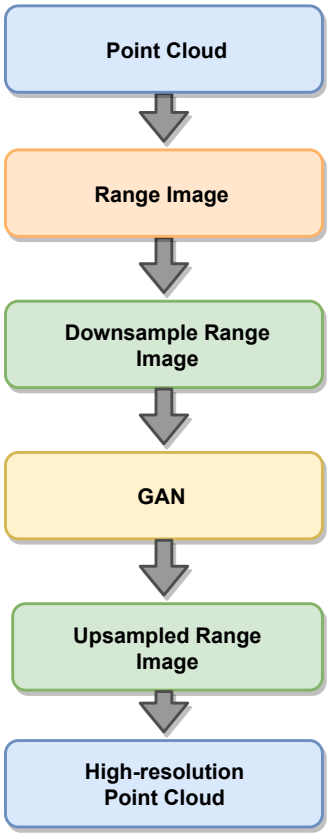

Figure 3.1.: Workflow of LiDAR point cloud up-sampling using GAN framework.

3.2. Data Processing

The resulting data from LiDAR scans are 3D point clouds with spatial coordinates. To process the 3D point clouds using 3D convolutions demands high computations. As discussed in Section 2 an image like 2D representation of the point clouds in 3D Euclidean space can be efficiently processed by 2D convolutions. Moreover, such 2D representation preserves the spatial information about the 3D points and also maintains the ordering of the points. The transformed 2D point map is referred as range image or range map through this report. The point clouds can be represented in the range image format using Cartesian, Polar or Spherical coordinates. The details about the 2D representation of the point clouds are discussed in this section.

A measurement from LiDAR sensor specifies the distance with respect to the objects in the surrounding environment. The LiDAR scanner captures the number of reflected rays from the environment at a time instance. A scan determines the points from a full 360° rotation. Each measurement is performed at certain angle of orientation called as elevation angle θ and azimuth angle ϕ. Elevation angle specifies the vertical resolution and it is determined by number of lines in the sensor module. Azimuth angle represents the horizontal field of view and it is predefined for each benchmark dataset.

For a 3D LiDAR scan containing $N(x, y, z)$ coordinates, the elevation θ and azimuth angle ϕ are represented using the function in the equation 3.1.

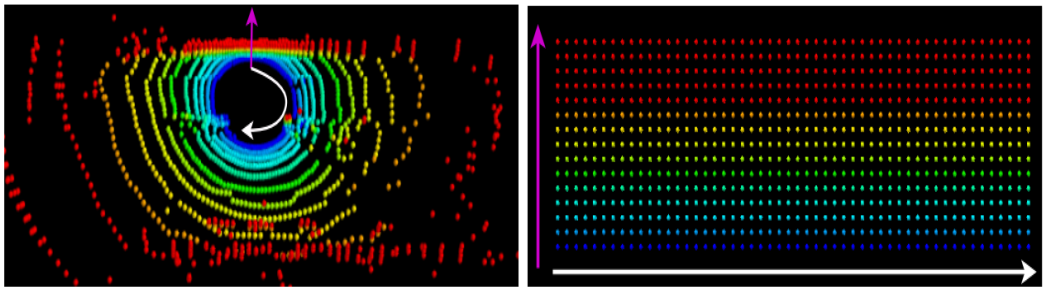

Figure 3.2.: Representation of 3D point clouds into a 2D range map.

$$\begin{aligned} \theta &= \operatorname{atan} 2(y, x) \\ \phi &= \arcsin\left(z/\sqrt{x^2 + y^2 + z^2}\right) \end{aligned} \tag{3.1}$$

Cartesian form. The process of converting a 3D point cloud into 2D range image with Cartesian coordinates is as follows.

1. All the points with same elevation angle θ are aggregated together into H bins. These points are captured from the same laser beam.
2. The points in every bin are sorted in ascending order of azimuth angle.
3. To obtain a 2D grid with rows and columns, the 360° plane is divided into W bins. By doing so, each row in the 2D map corresponds to vertical lines with fixed amount of point per row and each column represents the horizontal resolution.
4. Each cell in $H \times W$ grid map contains the average of (x, y, z) coordinates. Such representation results in a range image in Cartesian form of size $(H \times W \times 3)$.

Polar form. For Polar 2D range image representation, the function in equation 3.2 can be followed.

$$\begin{aligned} r &= \lfloor \theta / \Delta\theta \rfloor \\ c &= \lfloor \phi / \Delta\phi \rfloor \end{aligned} \tag{3.2}$$

Where $\Delta\theta$ and $\Delta\phi$ denotes the average vertical and horizontal resolution respectively. (r, c) represents the 2D range image with each element corresponding to (d, z) where (x, y) are compressed as $d = \sqrt{x^2 + y^2}$ as stated in [11]. In such a way, the number of channels in polar form can be reduced when compared to Cartesian form. The resulting 2D range image will be in the dimension of $(H \times W \times 2)$.

Spherical form. Each points of the point cloud is transformed from 3D Euclidean coordinates to Spherical ones, and project those coordinates to a 2D map based on the idea shown in [33]. The spherical coordinates are $sph = \{\phi, \theta, \rho\}$. The elevation angle θ represents H vector with a resolution $\Delta\theta$. The azimuth angle ϕ represents the W of the range image with a resolution $\Delta\phi$. Each (H, W) pair of the resulting range image encode the measured range (ρ) and reflectively of each point.

A visual representation of the point cloud mapping from 3D to 2D space referred from [12] is shown in figure 3.2, where the points collected from the same elevation angle have the same color. Every row is stacked based on the points in the increasing order of azimuth angle.

In our work, the range images with Cartesian and Polar coordinates are predominately used for our proposed model.

3.3. Network Architecture

The first GAN framework was formulated by Goodfellow et al. [18]. The framework consists of two convolutional neural networks, a generator G and a discriminator D. The main purpose of the generator is to learn the data distribution p_g, over a data x from a noise with prior distribution p_{noise} with an aim to fool the discriminator. The discriminator network D tries to distinguish between the fake generated samples form the real samples $x \sim p_{real}$. The goal of discriminator D is to maximize the estimated difference between the fake and real samples. Whereas, the generator G is trained to fool D by estimating the generated fake as real. The generator and discriminator networks are trained alternatively in each epoch. This min-max problem of GAN has an objective as mentioned in equation 3.3.

$$\min_G \max_D \mathop{\mathbb{E}}_{x \sim p_{\text{real}}} \log(D(x)) + \mathop{\mathbb{E}}_{z \sim p_{\text{noise}}} \log(1 - D(G(z))) \tag{3.3}$$

GANs have proven the ability to generate more realistic outputs. However, stabilizing GAN training is still an open problem. In practice, GANs suffer from serious mode collapse problem [34]. which happens when the generator overlooks certain modes of the target distribution and the discriminator gets stuck in the local minimum.

In our approach, the generator G is trained to produce an up-sampled range image output that fools the discriminator. On the other side, the discriminator D is trained for pushing G to reach its goal by distinguishing the generated range images from the target real ones. In this work, the generator and discriminator networks are designed to suit the particularity of range images; refer figures 3.3 and 3.4 for an overview.

3.3.1. Generator Network

As shown in figure 3.3, the generator network G takes a down-sampled range image as input and outputs a high-resolution range image. Point cloud up-sampling is produced once the model is trained. The network roughly follows the architectural guidelines set by Ledig et al. [8] for photo-realistic image super-resolution.

The architecture of G begins with 9×9 flat convolutional stage at the input followed by B residual blocks and an up-sampling block with two trained transposed convolution layers for in-network up-sampling as per Wojna et al. [29].

The main part of the deep generator network G is the B residual blocks with identity mapping from its inputs. The deep neural networks tend to suffer from the vanishing gradient problem. Beyond a certain depth, the network has more parameters and adding more layers will cause performance degradation of the network. He et al. [35] suggested that adding residual

shortcuts can speed up the convergence and enable efficient training of deep neural networks. The identity path circumvents the vanishing gradient problem, by allowing the output of one of the layers to flow directly to the input of a much deeper layer in the network. It is also mentioned that the residual connections make it easy for the network to learn the identify function; this is an appealing property for image transformation networks. The same idea is employed for range images since the output image should share the same structure with the input image.

The residual block design is inspired by Ledig et al. [8] which employed the design proposed by Gross and Wilber [36]. In specific, the residual blocks contains two 3×3 convolutional layers and 64 feature maps, followed by batch normalization (BN) layers [37] along with ParametricReLU (PReLU) [38] as the activation function for speeding up network convergence.

In contrast to SRGAN [8], transposed convolution layers are proposed to perform up-sampling instead of sub-pixel convolutional layers. For super-resolution with an up-sampling factor of r, two 4×3 transposed convolution layers followed by PReLU are used to increase the resolution of the input range image in a single direction (H) to a desired level. Finally, an up-sampled range image is produced by using a single convolutional layer with 9×9 kernel without batch normalization at the output layer. From figure 3.3, 'ConvTranspose' block indicates transposed convolutional layer and 'BN' indicates batch normalization layer.

3.3.2. Discriminator Network

Complementary to the generator network, the goal of the Discriminator D is to judge whether the input range maps are produced by the generator. The basic architecture is adopted from SRGAN model [8]. The discriminator is trained to solve the minimization problem in equation 3.3. As mentioned earlier, the alternating training of generator and discriminator leads to instability of GANs. Hence, in this task, it is essential to design a discriminator D that doesn't fail due to hyperparamater changes.

The D network contains nine 3×3 convolutional layers with an increasing kernel by a factor of 2 from 64 to 512 kernels similar to the VGG network [23]. As the number of features are doubled, it is compensated by strided convolutions to reduce the range image resolution to a desired level. In addition, Leaky ReLU (LReLU) is a widely used activation function in deep networks for efficient gradient propagation to increase the performance of the networks by allowing a small, non-zero gradient when the unit is not active. LeakyReLU [39] with $\alpha = 0.2$ is used after each normalization layer. LeakyReLU is suitable for higher resolution modelling.

Spectral Normalization (SN) [40, 41] is a widely used regularization technique to stabilize the GAN training. Spectral norm on GANs produces more diverse samples and also improves fidelity than conventional weight normalization. To achieve this, each layer of D is forced to have a SN function during training. In addition, the benefit of using SN is that it requires no hyperparameter tuning [41].

The discriminator D network also uses batch normalization [37] in the intermediate layers following the spectral normalization units. Batch normalization helps to deal with the training problems that arises due to poor initialization. Batch normalization solves it by normalizing

the input to each layer with zero mean and unit variance. It also enhances the gradient flow in the deeper networks. The model instability is avoided by not applying the batch normalization to the input layers of the discriminator.

At the end, the resulting 512 feature maps are flattened by using AdaptiveAvgPooling layer as per [42] followed by flat convolutional layer and a sigmoid activation function to obtain a probability of the classification. The architecture of the discriminator is shown in figure 3.4, where 'SN' indicates spectral normalization and 'BN' indicates batch normalization layer.

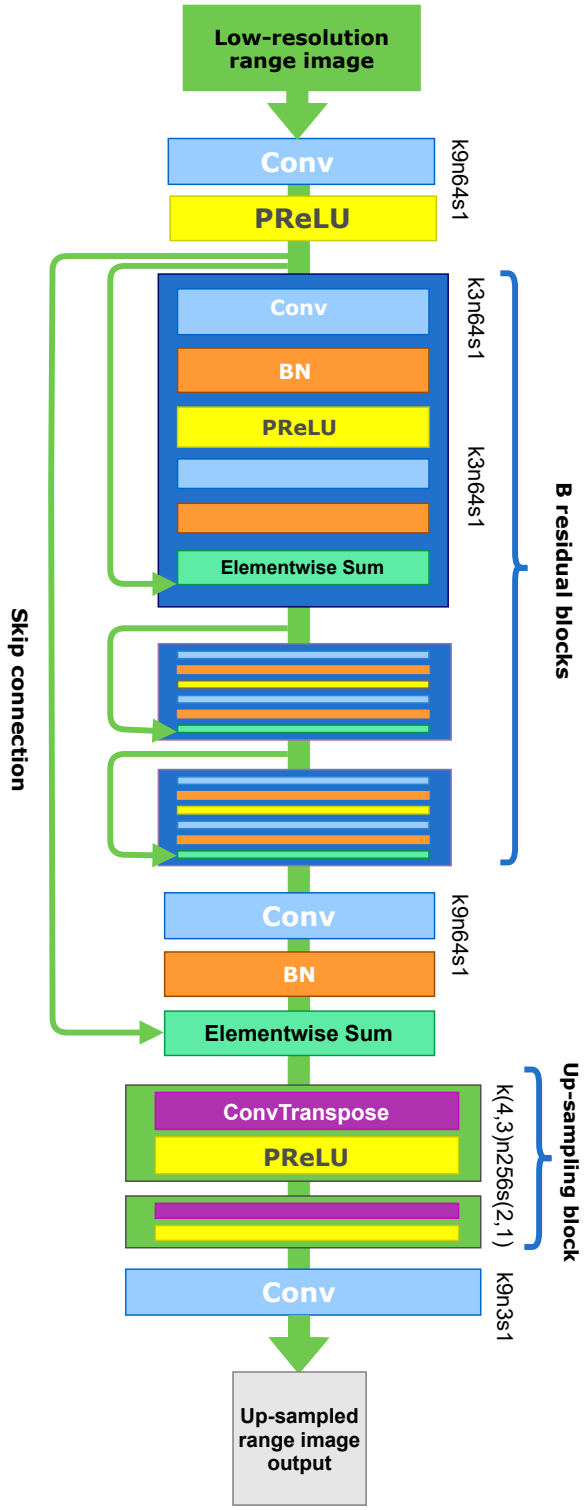

Figure 3.3.: Architecture of generator network with corresponding kernel size (k), number of feature maps (n) and stride (s) indicated for each convolutional layer.

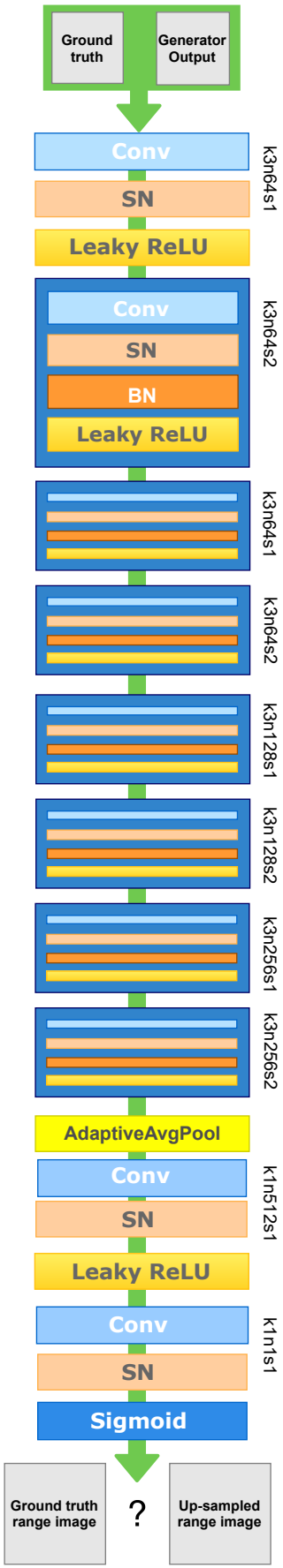

Figure 3.4.: Architecture of discriminator network with corresponding kernel size (k), number of feature maps (n) and stride (s) indicated for each convolutional layer.

3.4. Loss Functions

In order to train the generator and discriminator network simultaneously a combination of loss functions are used. The details of different loss functions used for generator loss L_G and discriminator loss L_D are presented in this section.

The generator loss function is a combination of four parts: (1) the content loss L_{MSE} which restricts the the generation of too much high-frequency content. (2) perceptual loss L_{VGG} based on feature extracted from pretrained VGG network, (3) adversarial loss L_{adv} which drives the generator to achieve the required transformation, and (4) total variation loss L_{TV} to enhance the desired textures. The total generator loss function is defined as follows:

$$L_G = \lambda_{\text{cont}} L_{\text{MSE}} + \lambda_{\text{percep}} L_{\text{VGG}} + \lambda_{\text{TV}} L_{\text{TV}} + \lambda_{\text{adv}} L_{\text{adv}} \tag{3.4}$$

where λ_{cont} , λ_{percep}, λ_{TV} and λ_{adv} are the weights used to balance the contribution of each loss term.

3.4.1. Content Loss

In this up-sampling task, the loss is calculated by comparing the high-resolution range images (d^{pred}) from G network and its corresponding ground truth to obtain the point-wise difference. This point-wise difference is referred as content loss, which can be used to encourage the network to generate high-resolution LiDAR range images. Point-wise content loss used to restrict the generation of very high-frequency content and it depends only on the low-level point-wise information. In our case, mean square error (MSE) loss is used as the content loss function.

The content loss is defined as:

$$L_{MSE} = \frac{1}{r^2 WH} \sum_{x=1}^{rW} \sum_{y=1}^{rH} \left(d_{x,y}^{gt} - G_{\theta_G} \left(d^{LR} \right)_{x,y} \right)^2 \tag{3.5}$$

Here d^{LR} is the low-resolution input range image to the generator network G and d^{gt} is the reference range image.

3.4.2. Perceptual Loss

The MSE-based content loss fails to encourage the network to generate realistic point clouds in real time scenarios. This in turn produces smooth surfaces which are perceptually unsatisfying. As per Johnson et al. [32], content loss is generally combined with a perceptual loss for training the models to achieve a high-resolution output. It allows the loss function to measure the perceptual differences between the LiDAR scans.

In case of transformation tasks like estimating the up-sampled output of LiDAR point clouds, perceptual loss plays a vital role in training the generator network. It gives the model an ability to restore high-level features of the range images and provides perceptual satisfaction.

Perceptual loss function utilizes a pretrained network to extract the features from the range images. The perceptual loss is defined based on the ReLU activation layers of the pretrained 16 layer VGG network as described by Simonyan and Zisserman [23]. This loss is formulated based on the Euclidean distance between feature maps extracted from up-sampled and ground truth range images. This extracted feature map Φ is used to compare the point clouds on a more abstract level. The perceptual loss between the generated high-resolution range images and the reference range image is defined as:

$$L_{VGG/i.j} = \frac{1}{W_{i,j}H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} \left(\phi_{i,j}\left(d^{gt}\right)_{x,y} - \phi_{i,j}\left(G_{\theta_G}\left(d^{LR}\right)\right)_{x,y}\right)^2 \tag{3.6}$$

Here $G_{\theta_G}\left(d^{LR}\right)$ is the probability of generated high-resolution output and d^{gt} is the reference range image. Whereas, $W_{i,j}$ and $H_{i,j}$ describes the dimensions of the respective feature maps within the VGG network.

3.4.3. Total Variation Loss

Total variational (TV) loss is one of the richest regularizer which is used to encourage spatial smoothness in the output image d^{pred} as described in prior work [43, 44, 45]. Due to up-sampling operation the boundaries and textures in the point clouds suffers from blurring effect and the areas nearby the edges may get overly smoothed. TV loss (L_{TV}) helps in preserving the sharpness of the borders of each object in the LiDAR scan.

3.4.4. Adversarial Loss

Adversarial loss is used for encouraging our network to output solutions that reside on the manifold of natural data. It is also known as generator loss which is helpful in trying to fool the discriminator network. Its value indicates to what extent the output range image from the generator G resembles like the high-resolution reference range image. The generative loss L_{adv} is defined based on the probabilities of the discriminator $D_{\theta_D}\left(G_{\theta_G}\left(d^{LR}\right)\right)$ over all the training samples as:

$$L_{adv} = \sum_{n=1}^{N} -\log D_{\theta_D}\left(G_{\theta_G}\left(d^{LR}\right)\right) \tag{3.7}$$

Here, $D_{\theta_D}\left(G_{\theta_G}\left(d^{LR}\right)\right)$ is the probability of reconstructed high-resolution range map $G_{\theta_G}\left(d^{LR}\right)$ is the actual target range image. For better gradient behavior we minimize $-\log D_{\theta_D}\left(G_{\theta_G}\left(d^{LR}\right)\right)$ instead of $\log\left[1 - D_{\theta_D}\left(G_{\theta_G}\left(d^{LR}\right)\right)\right]$. In our training, binary cross entropy (BCE) loss shown in equation 3.11 is used as the adversarial loss function.

3.4.5. Discriminator Loss

The discriminator loss function contains two terms: First is real loss L_{real} for encouraging the target range image more realistic than the up-sampled output image, shown in equation 3.8.

Second is the fake loss L_{rake} term for making the up-sampled output less realistic than target range image, shown as equation 3.9.

$$L_{\text{Real}} = -E\left[\log\left(\Delta_{\text{real}}\right)\right] \tag{3.8}$$

$$L_{Fake} = -E\left[1 - \log\left(\Delta_{\text{fake}}\right)\right] \tag{3.9}$$

With the real loss L_{Real} and fake loss L_{Fake}, the loss function of discriminator can be formulated as equation 3.10.

$$L_D = L_{\text{real}} + L_{\text{fake}} \tag{3.10}$$

In our work, binary cross entropy (BCE) loss function is used to calculate the real and fake terms of the discriminator loss L_D.

$$L_{BCE} = -\frac{1}{\text{output}} \sum_{i=1}^{\text{output}} d^{gt} \cdot \log d^{pred} + (1 - d^{gt}) \cdot \log\left(1 - d^{pred}\right) \tag{3.11}$$

3.4.6. Proposed Loss Functions

The choice of additional loss functions for training generator depends mainly on the underlying application. In our case, apart from the mainstream loss functions discussed in the previous section, we came up with few distance, geometric and feature-based loss metrics to improve the generator learning. Inspired by the idea of [46], two distance metrics for point clouds – the Chamfer distance (CD) and the Earth Mover's distance (EMD) are to be used as the additional generator loss functions. The feature matching (FM) loss is also adapted to further increase the stability and performance of the model by setting a new statistic as a target for the generator. Moreover, a Virtual Normal Loss (VNL) based on geometric function is added to the training of the generator network to achieve high quality geometric shapes in the output point cloud. The functionalities of each loss functions are discussed in detail in this section.

Earth Mover's Distance Loss

Earth Mover's distance (EMD) can be considered as a loss function to evaluate the similarity between the generated point cloud $S_{pred} \subseteq \mathbb{R}^3$ and the ground truth point cloud data $S_{gt} \subseteq \mathbb{R}^3$ distributions. This distance metric can be used as the additional loss function to train the generator network.

$$L_{EMD}\left(S_{pred}, S_{gt}\right) = \min_{\phi: S_{pred} \to S_{gt}} \sum_{x \in S_{pred}} \|x - \phi(x)\|_2 \tag{3.12}$$

where $\phi : S_{pred} \to S_{gt}$ is a bijection mapping.

The EMD distance solves an optimization problem, like transformation problem. It is robust against few outliers in the point cloud set. EMD is a measure of the distance of two point clouds based on the unique bijection ϕ. EMD can easily reconstruct the output, by enabling the points to capture the shape of the underlying surface. In practice, the computation of EMD is too expensive for deep neural networks.

Chamfer Distance Loss

Chamfer distance (CD) loss function is another candidate used in training networks relevant to point clouds. CD is designed to measure the similarity of two point clouds [47]. The CD loss L_{CD} is defined as:

$$L_{CD}\left(S_{pred}, S_{gt}\right) = \sum_{x \in S_{pred}} \min_{y \in S_{gt}} \|x - y\|_2^2 + \sum_{y \in S_{gt}} \min_{x \in S_{pred}} \|x - y\|_2^2 \tag{3.13}$$

Where S_{pred} and S_{gt} are two sets of (x,y,z) coordinates of the LiDAR scan. CD loss is invariant to ordering of the points in dataset. For each point, the algorithm of CD finds the nearest neighbor in the other set and sums up the squared distances between the two point sets. Compared to EMD loss, CD loss function is more suitable for point clouds due to its simplicity and better reconstruction ability shown in [48]. CD accomplishes higher accuracy and has little loss in coverage. CD in turn produces a high quality results in practice as mentioned in [46].

Feature Matching Loss

The common GAN training strategy focuses on maximizing the discriminator output. This strategy easily leads to instability of GAN. To overcome this problem, feature matching (FM) loss can be used as proposed by Salimans et al. [34]. It helps in extending the goal of the generator to match the feature vector from the discriminator. The feature vectors are extracted from the intermediate layers of the discriminator and hence they are more discriminative between the real and the generated data of the model. Let $f(x)$ denote the activation's or features on an intermediate layer of the discriminator, the new objective for the generator can be defined as:

$$\left\|\mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} \mathbf{f}(\boldsymbol{x}) - \mathbb{E}_{\boldsymbol{z} \sim p_z(z)} \mathbf{f}(G(\boldsymbol{z}))\right\|_2^2 \tag{3.14}$$

This new objective for the generator prevents it from over training on the discriminator network. With this objective, the discriminator is trained in a usual manner and $f(x)$ is extracted as an output in addition to the classification probability. In this work, FM loss is calculated using mean absolute error (MAE) function, and it can be defined as:

$$L_{FM}(y, \hat{y}) = \frac{1}{N} \sum_{i=0}^{N} \left| \hat{f(x)}_{real} - \hat{f(x)}_{fake} \right| \tag{3.15}$$

Where, $\hat{f(x)}_{real}$ and $\hat{f(x)}_{fake}$ are the average of feature vectors with real and fake samples as input to the discriminator.

Virtual Normal Loss

Yin et al. [49] shown that enforcing a loss based on virtual normal, we can obtain a good generalization and higher quality geometric shape in the output point clouds. As the network

architecture is based on convolutional layers, there are chances that they can incorporate smoothness and blurriness on the output LiDAR point clouds. This is mainly because, the CNNs focuses on producing low frequency features. In order to obtain a high-resolution output, it is essential to use higher-order geometric loss based on virtual normal (VN). It is also said that VNs are robust against noise, for example, the blurring effect caused by convolutional operations. It is worth noting that high quality geometric output can be achieved without injecting additional network parameters.

In this case, each pixel in the range image $p_i\,(u_i, v_i)$ is mapped to a 3D coordinate $P_i\,(x_i, y_i, z_i)$ by perspective projection. The 3D coordinate is denoted using the depth, 2D optical centers, and focal lengths along x and y coordinate axis. N groups of random points are selected from the range image, each group of three points. The points has to be non-colinear and long-range. These three points in each group forms a virtual plane in the 3D space, and a normal vector to this plane is calculated. This normal vector is called as Virtual Normal (VN) of the particular plane. Similarly, a large number of points are grouped in triplets and corresponding VNs are calculated. The direction divergence between the ground truth and predicted VNs is denoted as Virtual Normal Loss (VNL). The mathematical representation is as follows:

$$L_{\mathrm{VN}} = \frac{1}{N}\sum_{i=0}^{N}\left\|\boldsymbol{n}_i^{pred} - \boldsymbol{n}_i^{gt}\right\|_1 \tag{3.16}$$

Where, N is number of valid triplet groups satisfying the non-colinear and long-range conditions, n_i^{gt} is the normal vector of the ground truth from virtual plane i, and n_i^{pred} is the normal vector of the predicted output from the virtual plane i.

3.5. Improved Model Architecture

The results from the approach discussed in previous sections ends up in a smooth texture of point cloud distribution. In order to enhance the sharpness of the edges a modification to the generator and discriminator architecture is required. Since point cloud data has circular boundary conditions along the horizontal axis, the convolutional layers in both generator and discriminator uses circular padding horizontally and regular zero-padding vertically same as [50]. By enforcing circular padding in horizontal direction results in the dual benefit of improving the quality and halving the computational cost of output range images. The details of the improvements made in the generator and discriminator network using circular and constant padding are shown in the figures 3.5 and 3.6.

Bot the generator and discriminator networks follows the same architecture as in Section 3.3 except the additional functional padding layers. Two functional pad layers in constant and circular mode are added before the 2D convolutional layers as described in figure 3.7. Constant padding is applied on top and bottom (vertical direction) with a vector [0, 0, 1, 1] and circular padding is applied on left and right (horizontal direction) of the feature maps. Where the 'Func Pad Conv' block in the figures indicates the functional padding convolutional unit which is elaborated in figure 3.7.

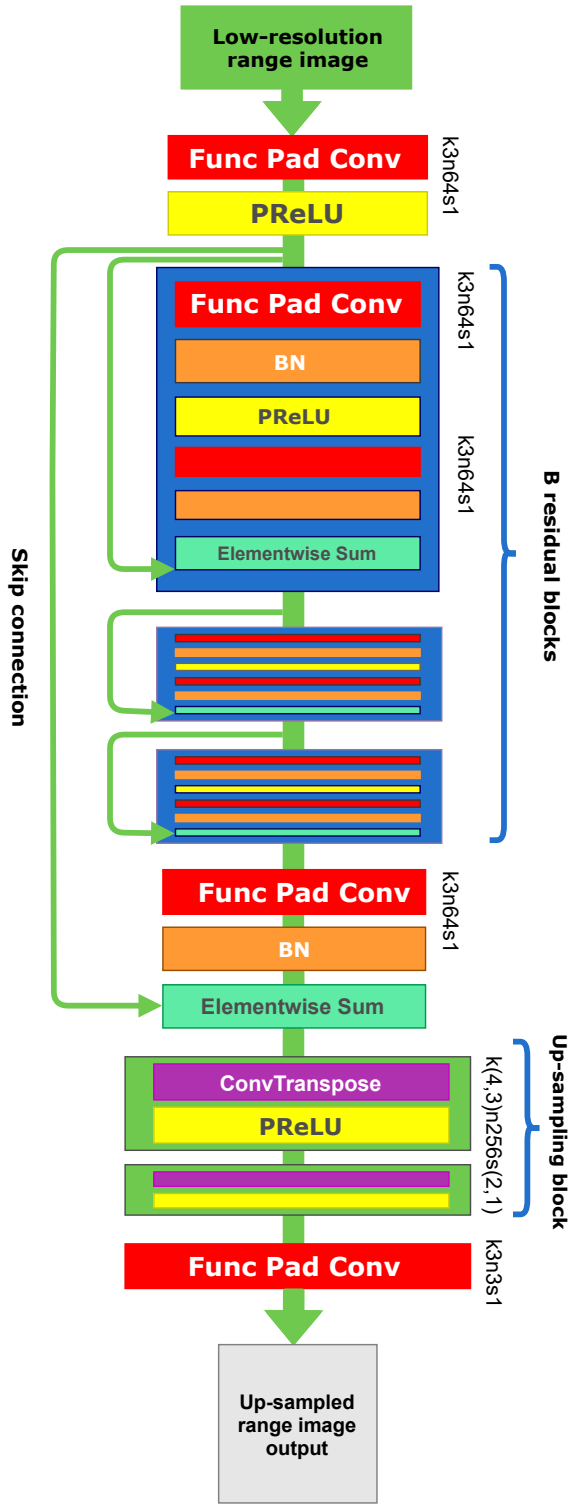

Figure 3.5.: Generator architecture with functional padding layers, where (k) is the kernel size, (n) is the number of feature maps and (s) is the stride in each convolutional layer.

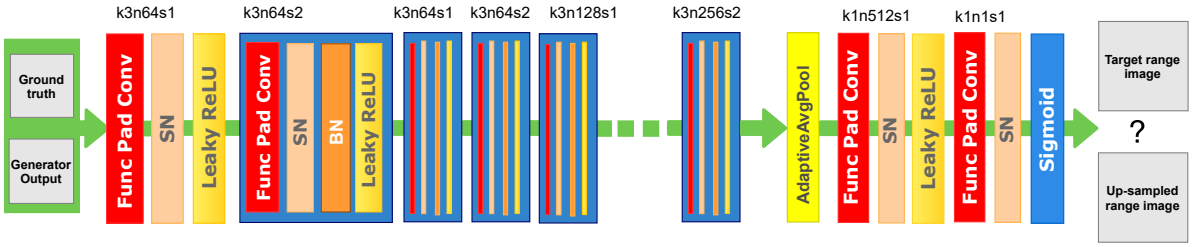

Figure 3.6.: Architecture of the discriminator with functional padding layers, where (k) is the kernel size, (n) is the number of feature maps and (s) is the stride in each convolutional layer.

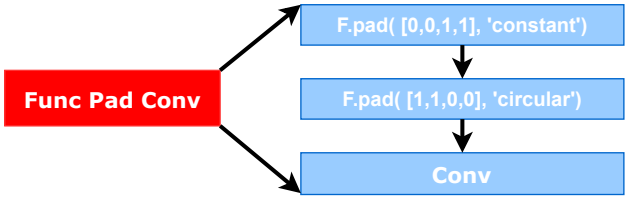

Figure 3.7.: Schematic of the functional padding convolutional unit.

4. Experimental Setup and Results

The implementation of LiDAR up-sampling methodology discussed in Section 3.1 involves a chain of experiments. This section covers in detail about the training dataset, details of the implementation and presents suitable metrics to evaluate the models. In addition, an overall training methodology that describes each training experiments in detail is also outlines in this section. Finally, the qualitatively and quantitatively of the results are presented.

4.1. Dataset

To train our networks the large-scale KITTI dataset is used. KITTI [9] is a publicly available raw dataset consists of sequenced frames. The dataset consists of 36 different annotated point cloud scenarios of variable length and a total of 12919 frames. These scenarios have diverse driving environments such as highway, roads, traffic, city and residential areas. They are also rich with dynamic and static objects. Most significantly, the KITTI used a Velodyne HDL64 sensor. The number in the name corresponds to the layer count in the LiDAR scan. For HDL64, those layers have an equidistant spacing. The HDL64 is limited to a distance lower than 80 meters.

The dataset split into training and validation set making sure that frames from the same sequence are not used in different splits. For the application of point cloud up-sampling it is straight-forward to generate the training input and the corresponding ground truth. The data is preprocessed as described in Section 3.2 and it serves as our high-resolution target data with a shape of 40×256 grid. The input low-resolution range maps are obtained by down-sampling the input scan with random removal of layers, resulting in a size of $(H/r \times W)$. This procedure is different from image-based up-sampling applications, where a bicubic down-sampling is used to generate the low-resolution data. For LiDAR point clouds, this method generates unrealistic results due to the large vertical spacing between the layers as stated in [6].

4.2. Training Details

The train and validation datasets are transformed to 2D range images. The datasets are encountered with further preprocessing to remove the outliers. The processed LiDAR range images are used as the ground truth. The low-resolution LiDAR range maps are obtained after down-sampling the target ground truth in the direction of H by an up-sampling factor r as mentioned in Section 4.1. As KITTI dataset is predominantly used for training the model, the shape of input and ground truth range maps are 10×256 and 40×256, respectively for an up-sampling factor of $r = 4$. All the training experiments are performed for an up-scaling

factor of $r = 4$. The model is trained with mini-batch stochastic gradient descent (SGD) with varying batch size specific to each loss function. All the training's are executed for 100 epochs. The perceptual loss is re-scaled by a factor of $\lambda_{\text{percep}} \approx 0.006$ to obtain a loss in the scale that is comparable to the MSE loss. Similarly the adversarial loss is weighted by a factor of $\lambda_{\text{adv}} \approx 0.001$. Whereas, the weightage of total variation loss is assigned with $\lambda_{TV} \approx$ 2e-8. The LeakyReLU activation function used in discriminator network has the slope of the leak set to 0.2. For optimization, Adam [51] with tuned hyperparameters are used. The learning rate of generator and discriminator networks are assigned as 0.001 and 0.00001, respectively. During training, the generator and discriminator are updated in an alternate manner, which is equivalent to $k = 1$ as used in Goodfellow et al. [18]. The models are implemented with PyTorch framework and trained using NVIDIA GeForce GTX 1080 Ti GPU.

4.3. Evaluation Metrics

A definite quantitative evaluation criterion for GAN is still an open research topic. There exists no standardized metric for evaluating the LiDAR point clouds up-sampling. Qualitative assessment depends on the visual inspection of samples. To quantitatively evaluate the quality of the generated up-sampled range image, we compare it with the ground truth. As the actual model output is a range image, it is essential to use metrics to evaluate the output in range image as well as in the point cloud format to comment on the effectiveness of our model. Four evaluation metrics are chosen, namely: (1) Earth-Mover's Distance (EMD), (2) Chamfer Distance (CD), (3) Root Mean Square Error (RMSE), and (4) Peak Signal to Noise Ratio (PSNR).

4.3.1. Earth Mover's Distance (EMD)

EMD is used to measure how close the generated up-sampled point cloud S_{pred} is to the ground truth S_{gt}, it is defined as

$$d_{\text{EMD}}\left(S_{pred}, S_{gt}\right) = \min_{\gamma: S_{pred} \longrightarrow S_{gt}} \sum_{x \in S_{pred}} \|x - \gamma(x)\|_2 \tag{4.1}$$

where γ is a bijection between the two point cloud datasets.

The Earth Mover's Distance (EMD) gives the solution to the optimal transportation problem, which attempts to transform one point cloud into the other. A recent work [19] shown that this metric correlates well with human inspection. Moreover, the EMD is sensitive to both global and local structures, and does not require points to be ordered. Ideally, the EMD metric for a model is expected to be equal to 1, meaning that the distribution of generated output and the ground truth are very close to each other. The ideal value is hard to achieve in practice. A lower EMD value means a better model performance.

4.3.2. Chamfer Distance (CD)

Chamfer Distance (CD) is often used to measure how far two sets of points are from each other. For each point, the algorithm of CD finds the nearest neighbour in the other set of coordinates and sums the squared distances up. Intuitively, CD finds the average of the distances of all nearest point pairs for each adversarial point [46]. CD is invariant to the ordering of the output points. Though simple, CD produces reasonable high-quality results in practice. The formal definition is as follows:

$$d_{\mathrm{CH}}\left(S_{pred}, S_{gt}\right) = \sum_{x \in S_{pred}} \min_{y \in S_{gt}} \|x - y\|_2^2 + \sum_{y \in S_{gt}} \min_{x \in S_{pred}} \|x - y\|_2^2 \tag{4.2}$$

Where, S_{pred} and S_{gt} are the two sets of (x, y, z) coordinates. Lower the value of CD, better is the model performance.

4.3.3. Root Mean Squared Error (RMSE)

The Root Mean Squared Error (RMSE) value is obtained from the Euclidean distance between each points in the range maps. RMSE measures the amount of change per point in the range image due to the processing. RMSE is directly proportional to the square root of MSE [52]. It is sensitive to outliers and can magnify the results if there are outliers in the dataset. The RMSE gives a relatively high weightage to large errors. The RMSE between the reference or ground truth range image, $d_{gt}(i, j)$ and the output range image, $d_{pred}(i, j)$ is given by

$$\mathrm{RMSE} = \sqrt{\frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [d_{gt}(i, j) - d_{pred}(i, j)]^2} \tag{4.3}$$

RMSE can be used to evaluate the quality of the generated range image. Lower RMSE value indicates that the output is very close to the reference range image.

4.3.4. Peak Signal to Noise Ratio (PSNR)

For the models that operate directly on range images, a measure that determines the similarity between two images can be used to measure the quality of the up-scaling. The PSNR is used as a measure of the quality of image reconstruction. PSNR is commonly used to measure the similarity between up-scaled and ground truth images [52]. PSNR determines the quality of high-resolution range image output from the down-sampled range image as input. PSNR calculates the Mean Square Error (MSE) of each point in the range image. For the $m \times n$ ground truth range image d_{gt} and its corresponding high-resolution image d_{pred}, the definition of MSE is shown in equation 4.4. Therefore, PSNR can be defined as shown in equation 4.5.

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [d_{gt}(i, j) - d_{pred}(i, j)]^2 \tag{4.4}$$

$$PSNR = 10 \cdot \log\left(\frac{MAX_{d_{gt}}^2}{MSE}\right) \tag{4.5}$$

Where, MAX_{gt} is the maximum possible point value of the ground truth range image. A higher PSNR value corresponds to a strong similarity between the up-sampled and the ground truth range images. However, PSNR was found to be inconsistent with the human visual system as reported in [53]. It takes into account the luminance, contrast, and structure of the two images. Therefore, in this work, we recommend using PSNR as an additional metric in conjunction with the previously discussed metrics to evaluate the quality of up-sampling. By doing this, we can bypass the poor PSNR evaluation and get an accurate assessment of the result.

4.4. Training Methodology

The main goal in this work is to develop a best performing model for LiDAR point cloud up-sampling by using adversarial learning. An extensive range of experiments are performed using our proposed model and the baseline models.

An overview of various training experiments performed are presented here. As PU-Net and PU-GAN models are trained on an object and shapes based dataset, their pretrained models were evaluated on KITTI dataset to check their up-sampling ability. Then all the five baseline models discussed in Section 2.1 are re-trained using KITTI dataset with down-sampled range images as input. DGML-VAE model takes the range image in Polar form as input. Whereas, LHRCNN and U-Net baselines models as supplied with input range images in Spherical coordinates format. For a fair comparison, EMD and CD losses were added to the training of the models to enhance the point cloud reconstruction capability. For every model considered, we select the learning rate, the latent dimension and the batch size from the predetermined set of values.

The proposed model is implemented using the training details mentioned in Section 4.2. The dataset is processed and represented as per the discussion in Section 4.1. Predominantly, range image with Cartesian and Polar coordinates are used for training the proposed model. Initially the proposed model is trained using the four default loss functions specified in Section 3.4.1 - 3.4.4. Based on the evaluation of the results, the model is trained with different configurations as discussed in Section 3.4.6 and 3.5. Experiments are done with an aim of achieving a uniform distributed dense output with realistic quality and sharp edges to define the shapes in the scene. In specific, the line structure of the LiDAR has to be retained without any smoothing or blurring effect.

The proposed model is trained by adding each proposed loss functions to the total generator loss. In addition, an experiment with the modified network architecture is also conducted on an intention to sharpen the edges. Finally, based on the results from different configurations, a combination of loss functions which gives best possible metrics values are proposed and the resultant model is named as LSRGAN. The batch size for training is varied for each configuration based on the memory capacity in the GPU.

The six different training configurations used for an ablation study of the proposed model are mentioned below.

- First the perceptual loss L_{VGG} is removed from the total generator loss function and the whole training is repeated with MSE-based content loss and adversarial loss using same hyperparameters expect the batch size. In this case, the total generator function from equation 3.4 is reformulated as:

$$L_G^{'} = \lambda_{\text{cont}} L_{\text{MSE}} + \lambda_{\text{TV}} L_{\text{TV}} + \lambda_{\text{adv}} L_{\text{adv}} \tag{4.6}$$

- The second configuration is using Feature Matching (FM) loss by extracting features from every layers of the discriminator. The loss between the features obtained by training discriminator with generated and ground truth range images are added to the total generator loss function obtained from 3.4. FM loss is calculated using mean absolute error (MAE). The updated loss function L_G^{FM} for this configuration is as follows:

$$L_G^{FM} = L_G + L_{FM} \tag{4.7}$$

- Third configuration is using EMD loss L_{EMD} instead of adversarial loss L_{adv} to calculate the generator loss function as mentioned in Section 3.4.6. As EMD loss L_{EMD} has the properties to capture the shape of the underlying surface it is expected to provide better perceptual results. In this experiment, the content loss is assigned a weight of $\lambda_{\text{cont}} = 0.1$. The generator loss function is redefined as:

$$L_G^{EMD} = \lambda_{\text{cont}} L_{\text{MSE}} + \lambda_{\text{percep}} L_{\text{VGG}} + \lambda_{\text{TV}} L_{\text{TV}} + L_{\text{EMD}} \tag{4.8}$$

- Fourth configuration is similar to the previous one. The EMD loss function in equation 4.8 is replaced with CD loss L_{CD} function from equation 3.13.
- The model architecture is modified by adding padding layers as described in Section 3.5. The training is repeated with default generator loss function as specified in equation 3.4.
- Finally, the model is training using a combination of Chamfer Distance loss L_{CD} and Virtual Normal loss L_{VN} in addition to the default loss functions. Here, MAE loss function is used to calculate the content loss. The total generator loss is reformulated as:

$$L_G = \lambda_{\text{cont}} L_{\text{MAE}} + \lambda_{\text{percep}} L_{\text{VGG}} + \lambda_{\text{TV}} L_{\text{TV}} + \lambda_{\text{adv}} L_{\text{adv}} + L_{\text{CD}} + L_{\text{VN}} \tag{4.9}$$

All models are trained end-to-end on the same KITTI dataset with 4× up-scaling factor. The results from all the baselines and our proposed method, are compared both qualitatively and quantitatively. Specifically, the baselines using raw point cloud data like PU-GAN and PU-Net are evaluated using EMD and CD metrics. Meanwhile, architectures using range images are evaluated and compared using EMD, CD and PSNR metric values. The ablation study of our proposed method with different configurations are evaluated using EMD, CD and RMSE values.

4.5. Results

In this section, we discuss the detailed analysis of the results obtained from training and validating the proposed model and the baseline models using the methodology 4.4.

A series of experiments are conducted to analyze the performance of our method. All the experiments are performed using 4× up-scaling factor. Based on the proposed method, the 3D data is represented in 2D range image format with each cell corresponding to spatial coordinates. In our implementation range images with Cartesian, Polar and Spherical coordinates are used for training and evaluation of the different models. We qualitatively and quantitatively compare proposed GAN model with five state-of-the-art point cloud up- sampling methods: PU-Net [1], PU-GAN [2], DGML-VAE [12], LHRCNN [6], and U-Net [7]. Note that PU-Net and PU-GAN performs LiDAR up-sampling using raw point cloud data as input. The DGML with VAE network architecture is capable of handling range images with Cartesian and Polar coordinates. Whereas, LHRCNN and U-Net can as also handle the range images in Spherical form as input. For fair comparison, we train them using EMD loss and CD loss in addition to the model specific loss functions. The results obtained from all the configurations and baselines are quantitatively compared using the evaluation metrics mentioned in Section 4.3.

4.5.1. Baseline Results

The qualitative analysis of the results from both point cloud based and range image based state-of-the-art models, and our proposed architecture are presented below. All the output results are visualized with corresponding input and ground truth data.

PU-Net

As a first step, the pretrained PU-Net model is evaluated with KITTI dataset using the metrics in Section 4.3. Next, the PU-Net model is re-trained using KITTI dataset. During training, the point clouds are converted to range images and then layers are removed based on the up-sampling factor r. Later, the down-sampled range image is reshaped to a dimension of $N \times 3$ to be compatible with the network architecture and then fed into the network as input. For a uniform evaluation, CD loss is also added to the model training in addition to EMD-based reconstruction loss. Compared to pretrained PU-Net the newly implemented PU-Net performs better. The former failed to handle large gaps between the layers and resulted in a denser, non-uniform LiDAR point cloud structure. Meanwhile the re-trained PU-Net was able to produced better up-sampling with retaining the structure of the underlying surface. This in turn gave a uniformly distributed high-resolution point cloud output. The visual illustration of the down-sampled input, corresponding ground truth and the outputs predicted form PU-Net is shown in figure 4.1. However, it shows that it leaves the gaps unfilled where the points are sparser and also causes blurring effect in some areas. The main drawback is, the model cannot scale well with number of points and it is computationally heavy.

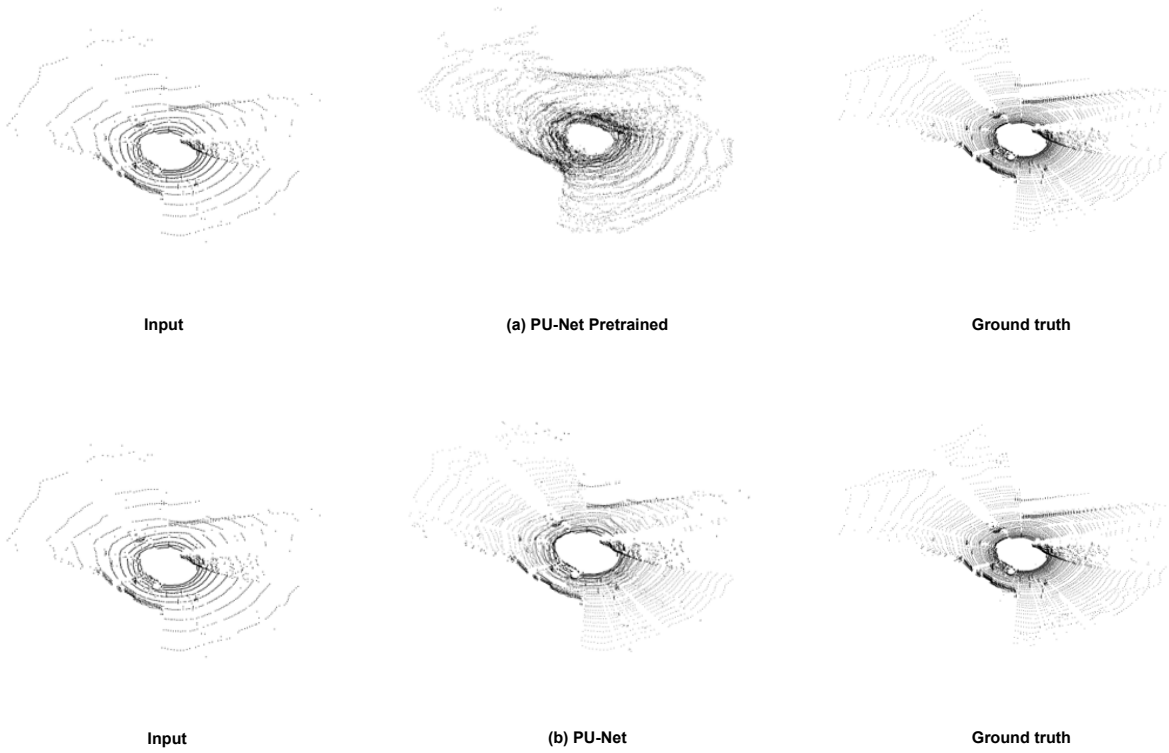

Figure 4.1.: Point cloud up-sampling results from PU-Net architecture.

PU-GAN

In our experiment, the pretrained PU-GAN models are first evaluated on KITTI dataset. There are two pretrained models for PU-GAN, one uses the patches of points from the point clouds to up-sample and other one uses the whole point clouds to up-sample. The PU-GAN pretrained on PU-Net patches did not converge and leads to poor up-sampling results as shown in figure 4.2. Following this results, PU-GAN model is re-trained following the proposed data processing approach. After preprocessing the point clouds, they are reshaped to be compatible with the model architecture. The down-sampled low-resolution data is passed as input to the model. CD loss is also added to the total loss function during the training. Figure 4.2 visualizes the low-resolution input, up-sampled output and ground truth point clouds of the PU-GAN model. The results are not satisfactory when compared to the corresponding ground truth. This model is optimized to produce a uniform point cloud with homogeneous distribution. In practice, the LiDAR point clouds has lines structure which is not uniform in space. Thus, it cannot preserve the line structure of the LiDAR scans. Hence, the figure 4.2 and 4.1 demonstrates the incapability of raw point cloud based methods in up-sampling the LiDAR scans.

DGML-VAE

In this experiment, the DGML-VAE model is fed with down-sampled range images inspired from the idea of training VAE with missing data in the point cloud proposed by Caccia et al. [12]. The model is trained with range maps in Polar form as input. The output high-resolution

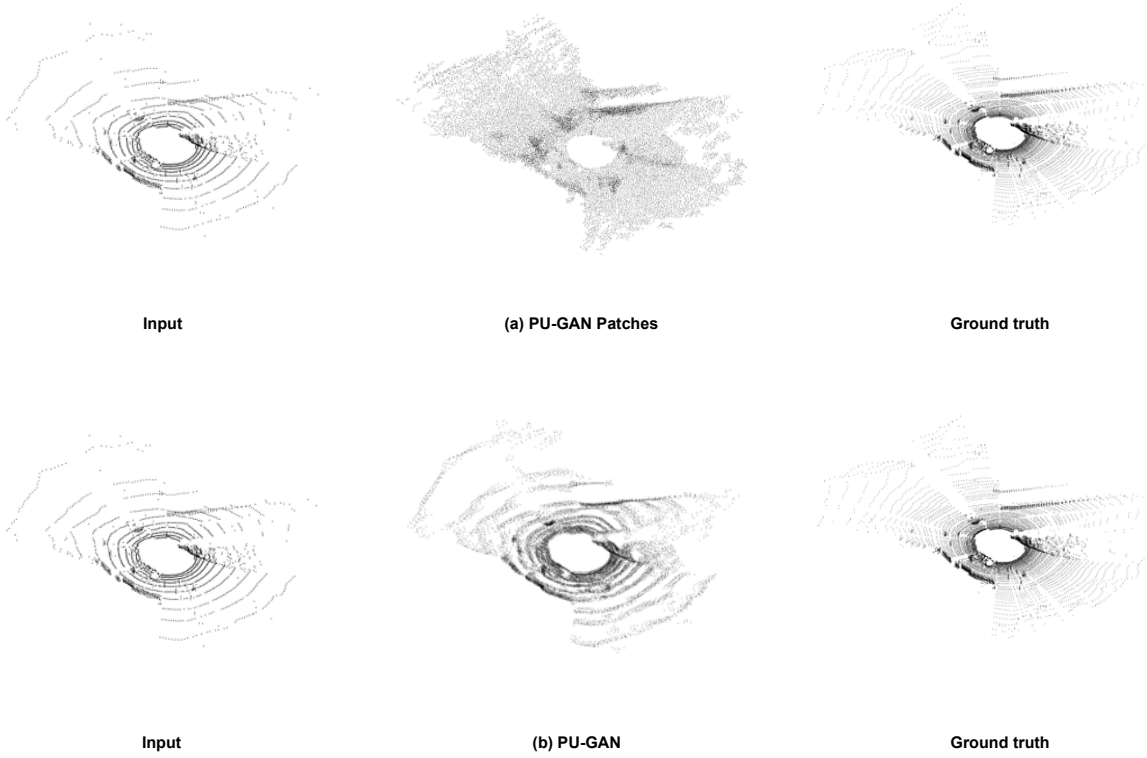

Figure 4.2.: Point cloud up-sampling results of PU-GAN model with corresponding input and ground truth.

range can be projected back to 3D domain to obtain an up-sampled dense point cloud. In addition to the reconstruction loss, EMD and CD losses are added to the training. DGML is specifically trained to solve the reconstruction problem with noisy input data. Based on the results shown in figure 4.3, it was able to handle the gaps in the input data gracefully. Moreover, with a low-resolution input, DGML-VAE model was able to up-sample while preserving the point cloud structure. However, the resultant output was overly smooth textured with no sharp edges. As it is trained for denoising, it cannot retain the spatial details of the data. Thus, DGML failed to retain the perceptual quality of the LiDAR point clouds.

LHRCNN

The CNN-based LiDAR high-resolution synthesis model (LHRCNN) is used as one of the baseline model. This model is re-trained with KITTI dataset. The range images in Spherical form are used as input to the network. As discussed in already, the input range images are down-sampled in vertical direction and then used for training. The perceptual loss which is the specialized loss function of this method is not used due to the absence of pretrained point-wise semantic segmentation network. Instead, the model is completely trained with L_1 loss. Figure 4.4 depicts the input and output range images along with the corresponding point cloud representation. The model is able to up-sample the point clouds while retaining the structure of LiDAR scans. On comparison with the ground truth, the output point cloud fails to retain the few finer details of the structure. From the range images in figure 4.4 (a)

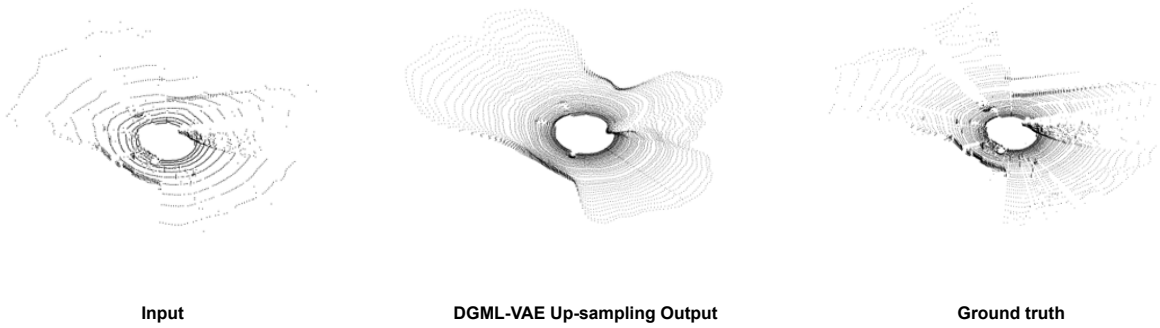

Figure 4.3.: Up-sampling results of DGML-VAE model.

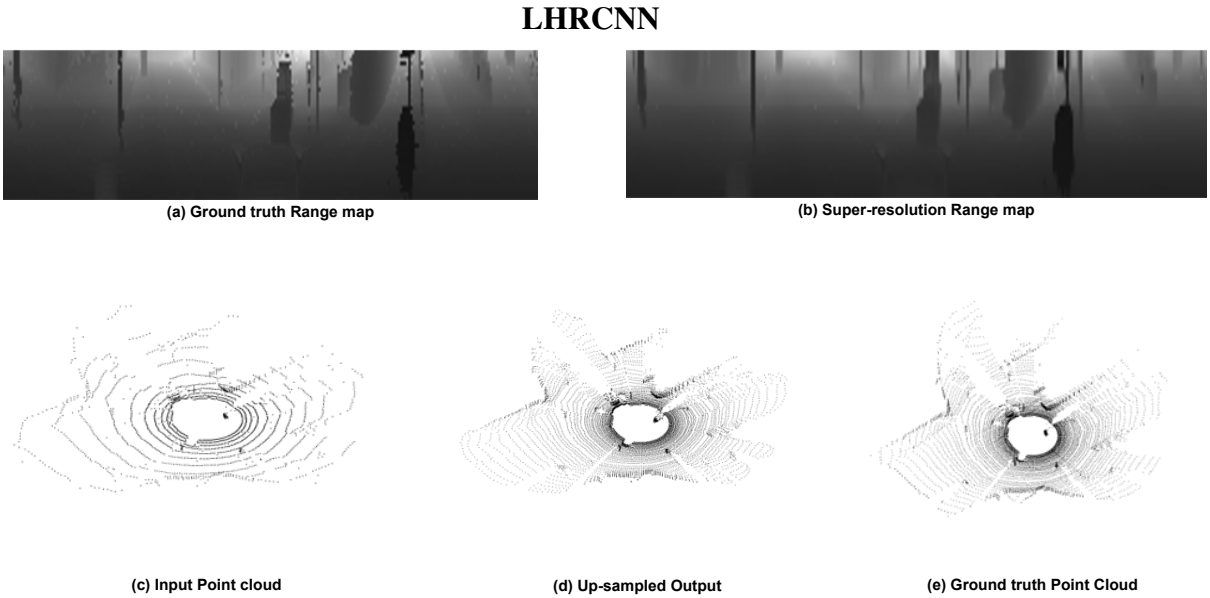

Figure 4.4.: Results from CNN-based LiDAR up-sampling model.

and (b), it is evident that small details from the ground truth are missing in the output range map. However, it is very difficult to precisely assess the visual results. Hence, evaluation of this results mainly depends on the performance metrics.

U-Net

U-Net based LiDAR super-resolution model is re-trained with KITTI dataset. The point clouds in 2D representation with spherical coordinates are used for training and evaluation of the model. Training is executed using L_1 loss for penalizing the deviation between the predicted output and the ground truth. The ground truth and high-resolution output range images are shown in figure 4.5. The super-resolution range image looks bit blurry and smoothed due to the convolutional operations. The same effect is seen in the corresponding projected point clouds. The output point clouds looks blurry and do not posses the required perceptual realism. However, performances metrics are considered more reliable in this case.

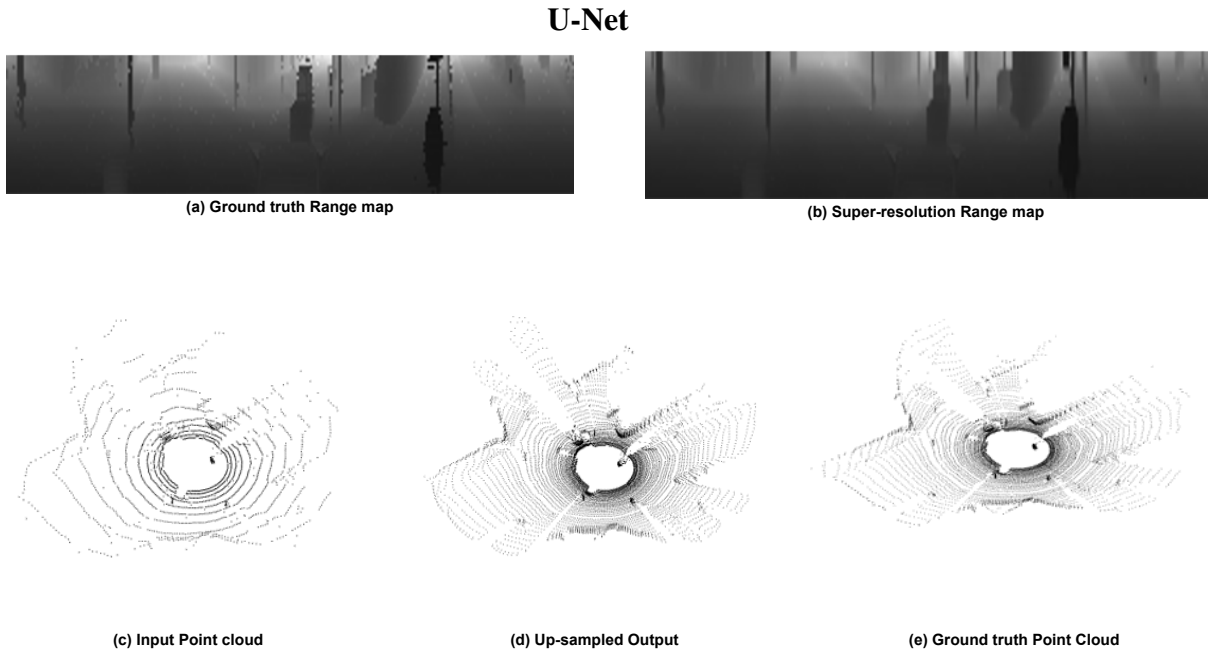

Figure 4.5.: Up-sampling results from U-Net model with corresponding input and ground truth.

4.5.2. Proposed Model

The qualitative results of our proposed model is presented here. The proposed model based on SRGAN architecture as discussed in Section 3.3.1 and 3.3.2 are implemented to demonstrate that it is capable of generating better up-sampling results than the baselines. The model is trained using the training parameters discussed in Section 4.2. The input data is processed as specified in Section 3.2. The model takes a down-sampled range images in Cartesian or Polar form as input. The training data for the neural network is gathered from 64-channel KITTI dataset and processed to obtain a 2D range image.

Firstly, the model is trained using a conjunction of content L_{MSE}, perceptual L_{VGG}, total variational L_{TV}, and adversarial L_{adv} loss functions as discussed in Section 3.4. The generated high-resolution output is shown in figure 4.6. Our proposed model can produce a better up-sampling results compared to the previous baselines. It tries to reconstruct the underlying structure of the point cloud and resembles like the corresponding ground truth shown in the figure. We can observe that, the point clouds generated by our model matches the ground truth with lower deviations. Though the results are better compared to PU-Net, PU-GAN and DGML-VAE models, the geometric shapes in the output are not evident and the model has blurring effect in the areas of minimal points. Compared to LHRCNN and U-Net models, the results of this training are not satisfactory.

Result from Modified Network Architecture

Based on the results from previous section, a need for improved model architecture was risen to reduce the smoothness in the point cloud texture and improve the realistic quality of the

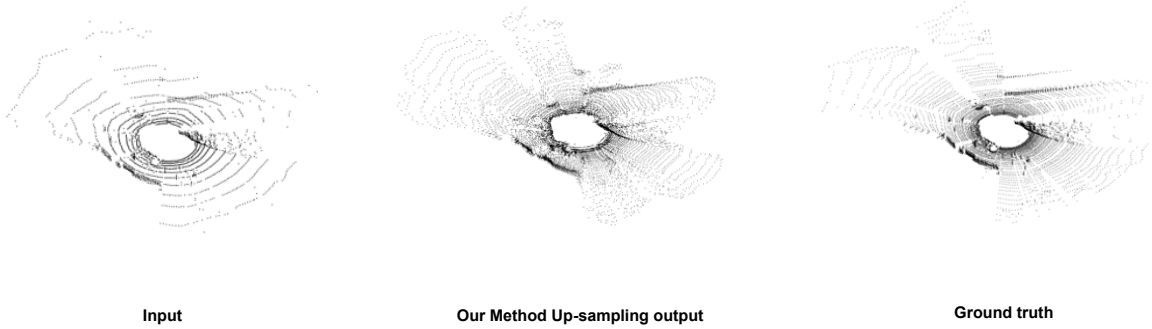

Figure 4.6.: Point cloud up-sampling results from our proposed model.

output. A circular padded architecture discussed in Section 3.5 is used for generator and discriminator of the model. This model is trained and evaluated with default loss functions as described in equation 3.4. As per this new architecture, two functional padding layers in constant and circular padding mode are added prior to the 2D convolutional layer.
The results from this training is shown in figure 4.7. It is inferred that the modified model architecture resulted in low training duration and thus has low computational cost. In addition, the smoothness in the output is reduced and edges are highlighted in contrast to other outputs. However, the expected uniformity in the output is not achieved as the points are scattered. Thus, this particular configuration failed to meet the expected line structure uniformity with perfect reconstruction of shapes.

Results from Various Loss Configurations

In an attempt to further improve the results of the proposed model shown in figure 4.6, different configuration are used to train the model. The proposed model is trained using different loss functions and a comparative study on different configuration is made. This is more similar to an ablation study by adding each loss function to the total generator loss.

The results from all the loss configurations of our model with corresponding input and ground truth point clouds are shown in figure 4.7. The output with MSE and adversarial loss ($L_G^{'}$) yields an up-sampling results with non-uniform distribution of the points in the outer region. Whereas, the model with EMD loss (L_G^{EMD}) was computationally heavy and it failed to converge. The results with feature matching loss using the loss function L_G^{FM} are highly satisfactory among all other configurations. Though it could produce a high-resolution point cloud with uniform distribution of points, the distribution gets smoother textures and this could be a problem for further processing of the point clouds like objection detection and semantic segmentation tasks. The output of the model using CD loss to calculate the generator loss (L_G^{CD}) ended up giving smooth textures to the layers. It is also observed that network with CD loss tends to scatter the points in its uncertain area where there are sparse points. Despite, it is able to reconstruct the global structure of the environment better than other configurations.

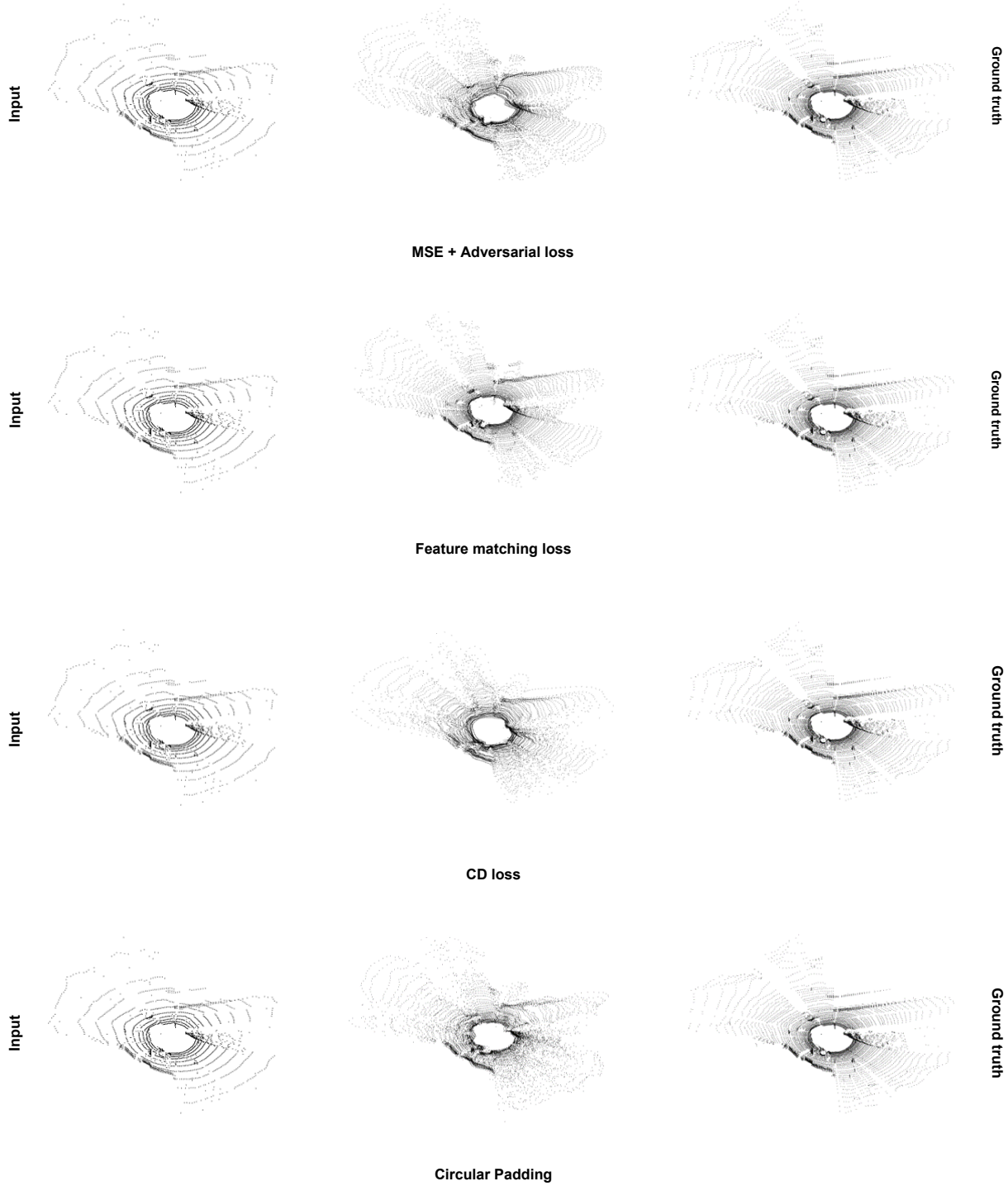

Figure 4.7.: Up-sampling results from various model configurations.

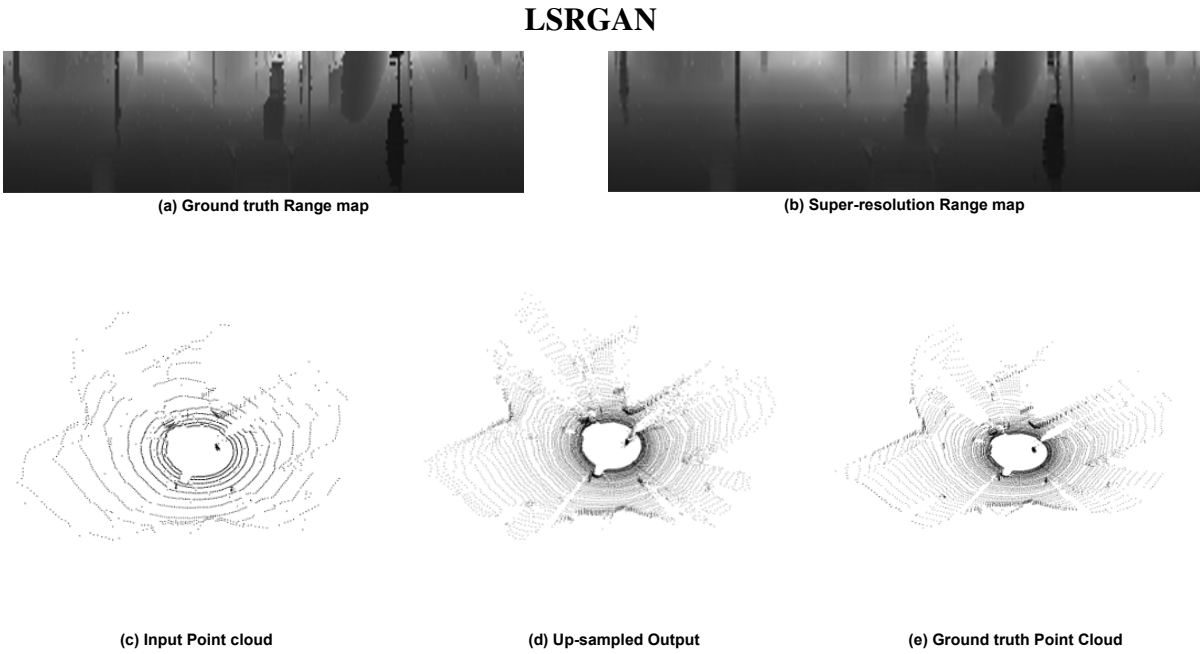

Figure 4.8.: LiDAR up-sampling results of LSRGAN with range map and point cloud representation.

LSRGAN

As an extension of the previous experiments, a final experiment is performed in narrowing down a definite configuration for the proposed model. This training is performed based on CD loss L_{CD} and the VNL loss L_{VN} along with default loss functions defined in equation 3.4. Note that, instead of MSE loss function to calculate the content loss, MSE (L_1) loss function is used. The visual results are shown in figure 4.8. The high-resolution range map looks very similar to the ground truth, with no big visual deviation. The generated output point clouds accurately resemble the geometric shapes due to the use of VNL loss. In this case, the VNL is used to produce the high-frequency content by eliminating the smoothing effect caused by convolutional layers. In addition, with the help of CD loss the global structure of LiDAR scans are perfectly reconstructed same as the ground truth.

4.5.3. Quantitative Results

All the training experiments on baseline and proposed models are evaluated using the metrics described in Section 4.3. Table 4.1 summarizes the complete quantitative comparison of results from raw point cloud based architectures. Among the baselines, pretrained PU-GAN on patches failed to converge, resulting in higher EMD metric value compared to all other results. On the other hand, the re-implemented PU-GAN, has slightly lesser EMD value as the model is slightly able to learn the structure. The CD metric in this case is very poor, showing that the network has poor reconstruction capability. PU-Net gives better results compared to other raw point clouds based models. However, all the raw point clouds based architectures are incapable of scaling with large number of points. Regardless, our method

with proposed CD and VNL loss functions as in equation 4.9 outperforms the other baseline models.

Table 4.1.: Quantitative comparison with point cloud based architectures

Framework	EMD	CD x10^(-4)
Pretrained PU-GAN (patches)	1330	1.51
PU-GAN	866	121.6
Pretrained PU-GAN	385	1.34
Pretrained PU-Net	371	2.66
Pretrained AR-GCN	265	1.14
PU-Net	242	1.70
LSRGAN (ours)	**116.5**	**0.45**

A complete ablation study of our proposed model with different loss configurations for each input form is represented in table 4.2. Following the results from qualitative comparison, the training with EMD loss produces poor metrics compared to all other configurations. In this case, the training was not converged. The training with CD loss leads to increase in the metric values compared to the default loss configuration of our method in table 4.1. The reason could be due to the absence of adversarial loss function the network has failed to achieve realistic outputs. From this analysis, the importance of adversarial loss function in training GAN network is inferred. The configuration MSE + adversarial loss and the circular padding architecture also failed to further improve the model performance with respect to all three metrics. Moreover, our method with feature matching (FM) loss achieves better performance indicating a better reconstruction. However, our LSRGAN model with CD, VNL loss functions as stated in equation 3.16, gives the best results consistently compared to all other configurations. Finally, it is observed that the model trained with range images in Polar form yields better up-sampling results than the Cartesian form, except the configuration with FM loss.

Table 4.2.: Quantitative comparison of our model with different configurations

Configuration	EMD	CD x10^(-4)	RMSE
EMD loss	855.6	179.4	19.5
Cartesian + CD loss	171.8	1.01	2.48
MSE + Adversarial loss	146.6	1.11	1.80
Circular padding	144.7	0.94	1.80
Cartesian	135	0.98	1.80
Polar	124	0.71	0.1
Polar + CD	120	0.59	0.86
Cartesian + FM	119	0.63	1.61
Polar + CD + VNL (LSRGAN)	**116.5**	**0.45**	**0.46**

The results from models based on range images are compared using EMD, CD and PSNR metric values. The results are tabulated in table 4.3. LHRCNN and U-Net models are per-

forming better than DGML-VAE model. But still their performance is below than our proposed LSRGAN with CD and VNL loss added to the default loss generator loss function. Particularly, metrics indicates that the geometric reconstruction of our proposed model is better. Following the qualitative results, the quantitative analysis also proves the effectiveness of the our LSRGAN model.

Table 4.3.: Quantitative comparison with range image based architectures

Framework	EMD	CD x10^(-4)	PSNR
DGML -VAE	272	1.70	20.9
U-Net	163	0.81	30.5
LHRCNN	153	0.84	30.5
LSRGAN (ours)	**116.5**	**0.45**	**31.22**

5. Conclusion

In this project, we presented the LSRGAN, a GAN-based LiDAR point cloud up-sampling network. This method uses 2D projection of point clouds and uses image super-resolution techniques to solve this problem. The input range image is first down-sampled and then up-sampled by means of the proposed network architecture. The generator network uses skip connections and deconvolutional layers to produce diverse and realistic up-sampled output. The discriminator is designed to handle the high-resolution range images and effectively penalize the outputs that deviates from the ground truth. To facilitate the training, a combination of content, perceptual and adversarial loss functions are used. Additionally, CD and VNL looses are used in training GAN to improve the reconstruction ability and geometric shapes with sharp edges. The output 2D range image when projected back gives the up-sampled point cloud data without any information loss. For a detailed study, different loss configurations are used to train the generator model. Through extensive experimental analysis with $\times 4$ up-sampling rate, it is demonstrated that our method with CD and VNL loss functions outperforms all the state-of-the-art baseline models for all the evaluation metrics. The results demonstrate that the proposed LSRGAN method can generate high-resolution point clouds with high fidelity and good geometric reconstruction.

5.1. Future Work

With increasing in the use of LiDAR sensors for autonomous driving, developing a better performing model for point cloud up-sampling will remain a potential topic of interest. This work of GAN based up-sampling method using range images can be extended in future for further improvements.

Possible future scope of this work is as follows:

- The model can be trained with other large scale and diverse autonomous driving datasets like nuScenes [54] and Waymo [55]. By this the performance our method with proposed configurations can be precisely evaluated based on the results obtained from other datasets. As the number of channels for nuScenes and Waymo differs from that of KITTI, the model has to be adapted to handle the spherical coordinates of the range images as well.
- Steps can be taken to improve the hyperparameter optimization to obtain even better performances.
- The generated up-sampled point clouds can be evaluated with the downstream tasks like 3D object detection to estimate the accuracy with dense point clouds.

A. Acronyms

LiDAR: Light Detection and Ranging

GAN: Generative Adversarial Network

VAE: Variational Autoencoders

SRGAN: Super-Resolution Generative Adversarial Network

DGML: Deep Generative Modeling

LHRCNN: LiDAR High-Resolution Convolutional Neural Network

LSRGAN: LiDAR Super-Resolution GAN

EMD: Earth-Mover's Distance

CD: Chamfer Distance

RMSE: Root Mean Square Error

PSNR: Peak Signal to Noise Ratio

FM: Feature Matching

VNL: Virtual Normal Loss

List of Figures

3.1. Workflow of LiDAR point cloud up-sampling using GAN framework. 14
3.2. Representation of 3D point clouds into a 2D range map. 15
3.3. Architecture of generator network with corresponding kernel size (k), number of feature maps (n) and stride (s) indicated for each convolutional layer. . 19
3.4. Architecture of discriminator network with corresponding kernel size (k), number of feature maps (n) and stride (s) indicated for each convolutional layer. 20
3.5. Generator architecture with functional padding layers, where (k) is the kernel size, (n) is the number of feature maps and (s) is the stride in each convolutional layer. 26
3.6. Architecture of the discriminator with functional padding layers, where (k) is the kernel size, (n) is the number of feature maps and (s) is the stride in each convolutional layer. 27
3.7. Schematic of the functional padding convolutional unit. 27

4.1. Point cloud up-sampling results from PU-Net architecture. 35
4.2. Point cloud up-sampling results of PU-GAN model with corresponding input and ground truth. 36
4.3. Up-sampling results of DGML-VAE model. 37
4.4. Results from CNN-based LiDAR up-sampling model. 37
4.5. Up-sampling results from U-Net model with corresponding input and ground truth. 38
4.6. Point cloud up-sampling results from our proposed model. 39
4.7. Up-sampling results from various model configurations. 40
4.8. LiDAR up-sampling results of LSRGAN with range map and point cloud representation. 41

List of Tables

4.1. Quantitative comparison with point cloud based architectures 42
4.2. Quantitative comparison of our model with different configurations 42
4.3. Quantitative comparison with range image based architectures 43

Bibliography

[1] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or and P.-A. Heng, "Pu-net: Point cloud upsampling network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2790–2799.

[2] R. Li, X. Li, C.-W. Fu, D. Cohen-Or and P.-A. Heng, "Pu-gan: a point cloud upsampling adversarial network," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 7203–7212.

[3] C.-L. Li, M. Zaheer, Y. Zhang, B. Poczos and R. Salakhutdinov, "Point cloud gan," *arXiv preprint arXiv:1810.05795*, 2018.

[4] G. Riegler, A. Osman Ulusoy and A. Geiger, "Octnet: Learning deep 3d representations at high resolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3577–3586.

[5] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4490–4499.

[6] L. T. Triess, D. Peter, C. B. Rist, M. Enzweiler and J. M. Zöllner, "Cnn-based synthesis of realistic high-resolution lidar data," in *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 1512–1519.

[7] T. Shan, J. Wang, F. Chen, P. Szenher and B. Englot, "Simulation-based lidar super-resolution for ground vehicles," *Robotics and Autonomous Systems*, vol. 134, p. 103647, 2020.

[8] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4681–4690.

[9] C. S. A. Geiger, P. Lenz and R. Urtasun, "Vision meets robotics: The kitti dataset," in *The International Journal of Robotics Research*, vol. 32, no. 11, 2013, pp. 1231–1237.

[10] B. Li, "3d fully convolutional network for vehicle detection in point cloud," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1513–1518.

[11] B. Li, T. Zhang and T. Xia, "Vehicle detection from 3d lidar using fully convolutional network," *arXiv preprint arXiv:1608.07916*, 2016.

[12] L. Caccia, H. Van Hoof, A. Courville and J. Pineau, "Deep generative modeling of lidar data," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 5034–5040.

[13] P. Ondruska, J. Dequaire, D. Z. Wang and I. Posner, "End-to-end tracking and semantic segmentation using recurrent neural networks," *arXiv preprint arXiv:1604.05091*, 2016.

[14] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin and C. T. Silva, "Computing and rendering point set surfaces," *IEEE Transactions on visualization and computer graphics*, vol. 9, no. 1, pp. 3–15, 2003.

[15] H. Huang, S. Wu, M. Gong, D. Cohen-Or, U. Ascher and H. Zhang, "Edge-aware point set resampling," *ACM transactions on graphics (TOG)*, vol. 32, no. 1, pp. 1–12, 2013.

[16] S. Wu, H. Huang, M. Gong, M. Zwicker and D. Cohen-Or, "Deep points consolidation," *ACM Transactions on Graphics (ToG)*, vol. 34, no. 6, pp. 1–13, 2015.

[17] R. Q. Charles, H. Su, M. Kaichun and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 77–85.

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

[19] P. Achlioptas, O. Diamanti, I. Mitliagkas and L. Guibas, "Learning representations and generative models for 3d point clouds," in *International conference on machine learning*. PMLR, 2018, pp. 40–49.

[20] R. Keys, "Cubic convolution interpolation for digital image processing," *IEEE transactions on acoustics, speech, and signal processing*, vol. 29, no. 6, pp. 1153–1160, 1981.

[21] C. Dong, C. C. Loy, K. He and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 2, pp. 295–307, 2015.

[22] A. Radford, L. Metz and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[23] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *In International Conference on Learning Representations (ICLR)*, 2015.

[24] O. Ronneberger, P. Fischer and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[25] C. Dong, C. C. Loy and X. Tang, "Accelerating the super-resolution convolutional neural network," in *European conference on computer vision*. Springer, 2016, pp. 391–407.

[26] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao and C. Change Loy, "Esrgan: Enhanced super-resolution generative adversarial networks," in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018.

[27] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert and Z. Wang, "Real-time single image and video super-resolution using an efficient subpixel convolutional neural network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1874–1883.

[28] T. Shang, Q. Dai, S. Zhu, T. Yang and Y. Guo, "Perceptual extreme super resolution network with receptive field block," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, 2020, pp. 1778–1787.

[29] Z. Wojna, V. Ferrari, S. Guadarrama, N. Silberman, L.-C. Chen, A. Fathi and J. Uijlings, "The devil is in the decoder: Classification, regression and gans," in *International Journal of Computer Vision*, vol. 127. Springer, 2019, pp. 1694–1706.

[30] W. Yifan, S. Wu, H. Huang, D. Cohen-Or and O. Sorkine-Hornung, "Patch-based progressive 3d point set upsampling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5958–5967.

[31] H. Zhang, I. Goodfellow, D. Metaxas and A. Odena, "Self-attention generative adversarial networks," in *International conference on machine learning*. PMLR, 2019, pp. 7354–7363.

[32] J. Johnson, A. Alahi and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European conference on computer vision*. Springer, 2016, pp. 694–711.

[33] V. Vaquero, I. Del Pino, F. Moreno-Noguer, J. Sola, A. Sanfeliu and J. Andrade-Cetto, "Deconvolutional networks for point-cloud vehicle detection and tracking in driving scenarios," in *2017 European Conference on Mobile Robots (ECMR)*. IEEE, 2017, pp. 1–7.

[34] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford and X. Chen, "Improved techniques for training gans," *arXiv preprint arXiv:1606.03498*, 2016.

[35] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016, pp. 770–778.

[36] S. Gross and M. Wilber, "Training and investigating residual nets," http://torch.ch/blog/2016/02/04/resnets.html, 2016.

[37] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning*, vol. 37, 2015, pp. 448–456.

[38] K. He, X. Zhang, S. Ren and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2015, pp. 1026–1034.

[39] A. L. Maas, A. Y. Hannun and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, vol. 30. Citeseer, 2013, p. 3.

[40] T. Miyato, T. Kataoka, M. Koyama and Y. Yoshida, "Spectral normalization for generative adversarial networks," *arXiv preprint arXiv:1802.05957*, 2018.

[41] Z. Lin, V. Sekar and G. Fanti, "Why spectral normalization stabilizes gans: Analysis and improvements," *arXiv e-prints*, 2020.

[42] Mordvintsev, Alexander, C. Olah and M. Tyka, "Inceptionism : Going deeper into neural networks," http://googleresearch.blogspot.com/2015/06/inceptionism-going-deeper-into-neural.html, 2015.

[43] H. A. Aly and E. Dubois, "Image up-sampling using total-variation regularization with a new observation model," *IEEE Transactions on Image Processing*, vol. 14, no. 10, pp. 1647–1659, 2005.

[44] E. d'Angelo, A. Alahi and P. Vandergheynst, "Beyond bits: Reconstructing images from local binary descriptors," in *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*. IEEE, 2012, pp. 935–938.

[45] A. Mahendran and A. Vedaldi, "Understanding deep image representations by inverting them," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 5188–5196.

[46] H. Fan, H. Su and L. Guibas, "A point set generation network for 3d object reconstruction from a single image," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 2463–2471.

[47] W. Zhang, H. Jiang, Z. Yang, S. Yamakawa, K. Shimada and L. B. Kara, "Data-driven upsampling of point clouds," *Computer-Aided Design*, vol. 112, pp. 1–13, 2019.

[48] P. Achlioptas, O. Diamanti, I. Mitliagkas and L. Guibas, "Learning representations and generative models for 3d point clouds," in *International conference on machine learning*. PMLR, 2018, pp. 40–49.

[49] W. Yin, Y. Liu and C. Shen, "Virtual normal: Enforcing geometric constraints for accurate and robust depth prediction," *arXiv preprint arXiv:2103.04216*, 2021.

[50] D. Pavllo, G. Spinks, T. Hofmann, M.-F. Moens and A. Lucchi, "Convolutional generation of textured 3d meshes," *arXiv preprint arXiv:2006.07660*, 2020.

[51] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[52] D. Asamoah, E. Ofori, S. Opoku and J. Danso, "Measuring the performance of image contrast enhancement technique," *International Journal of Computer Applications*, vol. 181, no. 22, pp. 6–13, 2018.

[53] J. Guo, S. Ma and S. Guo, "Maanet: Multi-view aware attention networks for image super-resolution," *arXiv preprint arXiv:1904.06252*, 2019.

[54] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 621–11 631.

[55] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine *et al.*, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2446–2454.

Declaration

Herewith, I declare that I have developed and written the enclosed thesis entirely by myself and that I have not used sources or means except those declared.

This thesis has not been submitted to any other authority to achieve an academic grading and has not been published elsewhere.

Stuttgart, 14.06.2021. Janaranjani Palaniswamy