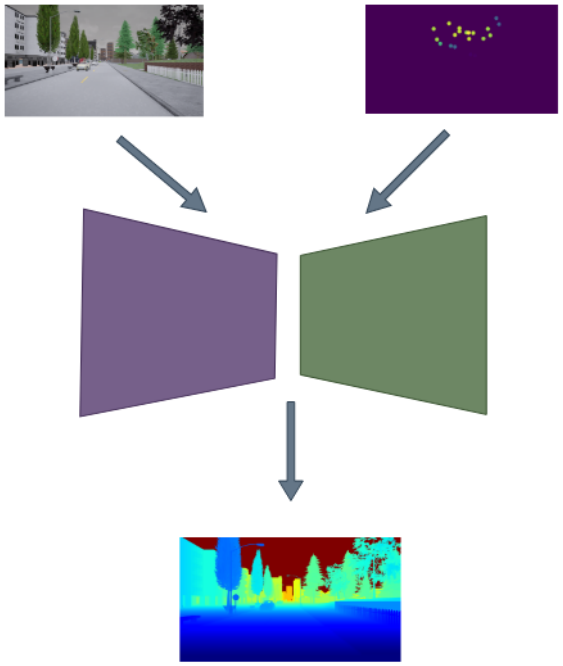
Robotics Research Lab
Department of Computer Science
University of Kaiserslautern

Master Thesis



Multi-Modal Depth Estimation Using Convolutional Neural Networks

Sadique Adnan Siddiqui

June 29, 2020

Multi-Modal Depth Estimation Using Convolutional Neural Networks

Robotics Research Lab
Department of Computer Science
University of Kaiserslautern

Sadique Adnan Siddiqui

Day of issue : 11.10.2019
Day of release : 22.06.2020

First Reviewer : Prof. Dr. Karsten Berns
Supervisor : M.Sc. Axel Vierling

Hereby I declare that I have self-dependently composed the at hand. The sources and additives used have been marked in the text and are exhaustively given in the bibliography.

June 29, 2020 – Kaiserslautern

(Sadique Adnan Siddiqui)

Acknowledgment

I would like to thank Prof. Dr. Karsten Berns for providing this opportunity to write my thesis at Robotics Research Lab. I am also deeply indebted to my supervisor Axel Vierling for his immeasurable support throughout my thesis. I would like to thank him for giving me the freedom to explore my topic and implement the ideas. This thesis would have been practically impossible without his interminable support.
I would also like to thank my peers at RR Lab and Mindgarage for all the technical assistance during these times.
I would like to thank Naveed Akram and Akash for their valuable help in creating the unreal dataset. I want to thank Ashutosh, Saurabh, Shridhar, Venky, Ritu, and Shekhar for all the help and knowledgable discussions.
Lastly, I'd like to thank my parents and my siblings. This would not have been possible without their constant support and encouragement.

Acronyms

SOTA	State-of-the-Art
ML	Machine Learning
AI	Artificial Intelligence
DL	Deep Learning
CNN	Convolutional Neural Network
Radar	Radio Detection and Ranging
Lidar	Light Detection and Ranging
NLP	Natural Language Processing
MRF	Markov Random Field
CRF	Conditional Random Field
GPS	Global Positioning System
IMU	Inertial Measurement Unit
KITTI	KITTI Vision Benchmark
VGG	Visual Geometry Group

Abstract

This thesis deals with multi-modal depth estimation using Convolutional Neural Networks. To understand how far things are located relative to the camera is a challenging task. However, it is vital for scene reconstruction which is needed for various applications such as robotics, autonomous driving, and augmented reality. The thesis explores the significance of different sensor modalities such as camera, Radar, and Lidar for estimating depth by applying deep learning approaches. CNN's perform exceptionally well on dense data, but it poses several challenges on sparse data such as Radar and fewer channel Lidar. A deep CNN architecture is proposed to investigate the impact of sparse data such as Radar point clouds on fusion with camera images for estimating depth. The network consists of an encoder where high performing pre-trained model has been used to initialize it for extracting dense features and a decoder for upsampling and predicting desired depth. The network is trained on the Nuscenes dataset [Caesar 19], which is the only multimodal dataset available that provides Radar data to the author's knowledge. A synthetic dataset has been created comprising RGB images, depth images, and Radar point cloud using Carla simulator to compensate the problem of unavailability of Radar data and perfect groundtruth depth map. The impact of Lidar point clouds on depth estimation, when fused with RGB images, has also been examined. Several experiments are conducted and inferences have been taken on a different dataset to demonstrate its generalization capability.

Contents

1. Introduction

Intelligence can be defined as a general mental ability for reasoning, problem-solving, and learning [Educba 17]. Human intelligence is capable of learning from its past experiences and can adapt to the unseen environment. They can effortlessly perform complex tasks such as recognizing, detecting objects and perceiving the scenes. However, in the case of machines to interpret what is going on in the scene is an arduous task. To develop an intelligent system that could perceive a scene as perfect as the human brain has potential applications in the field of robotics, surveillance, sports, medicines and entertainment. Recent years have witnessed ameliorating growth in the field of autonomous driving, mainly because of advancement in the areas of deep learning and artificial intelligence. After the year 2012, when the world witnessed the introduction of CNN's that gave the state-of-the-art result in the Imagenet image classification challenge, and after Google DeepMind's AlphaGo defeated the South Korean master in the board game Go, AI gained much public attention. There has been a sudden surge in almost every domain where machine learning can be possibly explored. As compared to other domains, research in the field of robotics and autonomous driving stands out among the rest. From the perception of the environment to controlling steering and planning of the paths are carried by deep learning approaches in self-driving vehicles.

The meteoric rise of autonomous driving technologies is not a recent development, but its journey has been long and winding. The first autonomous car came into existence in 1925 when the inventor Francis Houdini demonstrated a radio-controlled car, driven without anyone at the steering. In the early 1990s, researchers from Carnegie Melon were successful in demonstrating the use of neural networks to control steering in real-time. The major milestone in autonomous driving came through DARPA challenges held in 2004, where research institutions and companies competed to navigate through 142 miles circuit with their driverless vehicles. Unfortunately, none of the competitors were able to complete the challenge. However, in 2007, the DARPA urban challenge was organized where autonomous vehicles competed in a mock city environment with traffic rules, blocked routes, and hurdles similar to real-life driving challenges [Bacha 04]. In this tournament, around six teams were able to finish the race completing approximately 55 miles. Several factors, such as better collision-avoidance software and amalgamation of sensors such as Lidars, Radars, GPS was responsible for the successful demonstration of capabilities of

autonomous vehicles. From developing and testing prototypes in the research laboratory, the automotive industries have started rolling out the vehicles to be successfully driven on public roads. One of the surveys shows that almost 90 percent of car crashes in the US occurred due to negligence from the drivers. Companies such as Waymo, Tesla, Lyft nowadays are relying more on autonomous vehicles and they are making suitable utilization of sensors such as Radars, Lidars, cameras, ultrasonic sensors, GPS that could process the observations better than a human brain. Not only the automobile companies, even tech giants like Nvidia, Google, and startups such as Zoox, Cruise, are also actively engaged in developing sustainable self-driving cars by leveraging machine learning. The autonomous vehicles could help in traffic control, and that could result in reducing air pollution. The use of autonomous electric vehicles could surely help in reducing CO2 emissions. One of the primary reasons for autonomous driving research thriving so rapidly is due to the recent success of deep learning and machine learning methods. In this work, one of the specific domains of this technology is addressed, to obtain a representation of the spatial structure of the scene so that one could get the sense of distance of each pixel of the particular scene.

1.1 Motivation

The primary requirement to understand the scene is to infer the semantics and geometric structure of the scene from a single image. For computer vision in the autonomous driving field, the sensors continuously capture the information either in the form of images or in the form of point clouds, which provides distance data of the target object to build a picture that could be understandable to the vehicle. Intelligent systems do not have insights that a human has; they need to assign specific features for every object so that they could recognize them and understand what's happening in a scene. This process of scene understanding and recognizing objects can be divided into various tasks such as depth estimation, semantic segmentation, image segmentation, and object detection. This work focuses on the depth estimation task. The estimation of depth is helpful to infer the three-dimensional structure of the scene, which subsequently helps in other vision tasks such as estimation of semantic labels and pose estimation. The undeniable success of deep learning and the availability of better computation powers have invigorated the research communities to invest in this booming technology. There has been some breakthrough research for depth estimation problems from stereo image pairs, monocular images, and different sensing modalities. Depth estimation using rectified stereo image pairs is one of the reliable traditional methods, but its reliance on accurate image correspondence caused problems in areas of low texture and occlusions. Deep learning algorithms are being used to tackle the correspondence problems [Ridgeway 15] and pose estimation problem, where networks are trained to regress the 6-DOF camera pose from an RGB image [Kendall 15]. Different supervised and unsupervised techniques for depth estimation have been explored in recent years. The motivation behind this work is to explore the significance of sparse data on estimating depth when trained along with other modality using any one of the recent state-of-the-art network. As most of the fusion strategies propose to fuse Lidar point cloud and RGB images, the fusion of radar point cloud with different sensing modalities has not been explored much. The positive aspect of using Radars is its robustness to adverse weather conditions. A Radar sensor is cheap, lightweight, small in size, and is capable of measuring a more considerable distance than Lidars. As camera images provide

information about appearance and texture, distance data provides information about object shape, and it is invariant to lighting or color variation. The fusion model with both the modality could result in the enhancement of the performance of the network and could provide a better alternative than Lidars, which is quite expensive in comparison to other sensors. Another scope of this work is to estimate the depth of images captured at a different field of view. There are some real-life applications where zoom cameras play a vital role, and estimating depth using the stereo zoom camera is challenging as it requires careful calibration of cameras for determining the depth. For such cases, depth estimation on monocular zoomed image or fusion of single image and distance sensing modality using deep learning approaches can prove to be advantageous.

1.2 Tasks

In the aforementioned sections, the significance of depth estimation in autonomous vehicles is discussed. The goal of this work is to incorporate the Radar data with RGB images to showcase its significance on depth estimation task.
Basic overview of tasks are:

- Creation of groundtruth depth maps for Nuscenes dataset and creation of unreal dataset comprising of camera images, depth images, and Radar distance values using Carla simulator.
- Implementation of encoder-decoder architecture inspired by [Alhashim 18] and [Laina 16] that needs to be trained on different modality inputs.
- Evaluation of the proposed model.

2. Background

This chapter deals with the theoretical background necessary to understand the work done during the thesis. Brief details about machine learning techniques, followed by the working mechanism of CNN. A relevant literature survey has been done on traditional methods of depth estimation.

2.1 Machine Learning

2.1.1 Supervised Learning

It is a machine learning task of teaching a model by feeding input data along with its correct output. For an input variable (X) and an output(Y), the learning is said to be supervised if it emphasizes learning the mapping function from the input to the output i.e., Y = f(X). This technique aims to use that approximated mapping function on unseen data to predict the output of the given input. In this thesis, a supervised learning approach has been used where the input provided can be RGB images, or it can be a combination of sparse point cloud fused with the RGB, and the target is a depth map of the provided input. Supervised learning can be classified into two types of an algorithm: Classification and Regression. Classification aims at predicting output in the form of category whereas regression results in real value predictions.

Few of the important supervised learning algorithms are:

- Support Vector Machines
- K-Nearest Neigbours
- Linear Regression
- Logistic Regression
- Neural Networks
- Naive Bayes
- Decision Trees

2.1.2 Unsupervised Learning

In unsupervised learning, the training data consists of unlabelled targets i.e., the data contains only the inputs without any desired output assigned to them. The major focus in such an algorithm is to infer the commonalities in the datasets and group the unsorted data points according to their similarities or patterns and classify the raw data without any supervision. For example, when unsupervised learning techniques are applied to a set of images depicting different fruits, it categorizes them according to its similarities and differences so that it can group them into a specific category.

Few of the important unsupervised Learning techniques are:

- Clustering
  - K Means
  - Heirarchical
- Neural Networks
  - Generative adversarial networks
  - Autoencoders

2.1.3 Semi Supervised Learning

Both supervised and unsupervised techniques have their own disadvantages. Preparing labels for supervised learning is quite expensive in the case of regression-based problems such as semantic segmentation and depth estimation. Whereas in the case of unsupervised learning, although acquiring dataset is relatively cheap, its scope is limited. To counter the disadvantages of both the approaches, semi-supervised learning was introduced in which the training of the dataset is done by a combination of labeled and unlabeled data.

2.1.4 Reinforcement Learning

Reinforcement learning is a form of machine learning that is based on acquiring cumulative rewards based on the sequence of decisions. Unlike supervised learning, where input data is associated with its target, reinforcement learning gets some supervision in the form of feedback from the environment. It is more a game-like situation where the agent takes actions and interact with the environment, and it can either get rewards or penalty for the action it performs. The agent aims to maximize the rewards by taking sequential actions based on its observations. Markov Decision Process usually models reinforcement learning.

2.2 Convolutional Neural Network

Over the past decades, neural networks have performed tremendously well on structured data such as images and text. One of the most successful variants of these networks is a Convolutional Neural Network (CNN) because of its substantial contribution to benchmarked results on problems involving computer vision, NLP, and robotics. The architecture of CNN's is inspired by the organization of the visual cortex in the brain.

Although the working of CNN's is similar to the other neural networks, the significant differences are that unlike feed-forward neural networks, where inputs are vectors; here, the inputs are multi-channeled images. Secondly, the connection between the neurons of previous layers and subsequent layers is in a specific manner i.e. they follow hierarchical modeling where each neuron is connected to only a small chunk of the input image. The part in the input volume that a particular neuron is looking at is called a receptive field. The information from these local regions of input space with a limited receptive field is accumulated along with the depth of the input and thereby gives the effective receptive field over the inputs. One of the powerful features of CNN's is its parameter sharing ability where weights are shared by the neurons in a particular feature map. It helps in a considerable reduction of parameters and makes computation in CNN's more efficient as compared to fully connected neural networks. CNN's are comprised of mainly three types of layers, convolution layer, pooling layer, and fully connected layer. Using a stacked combination of these layers forms a complete CNN network. A simplified architecture of CNN is shown in Fig 2.1.

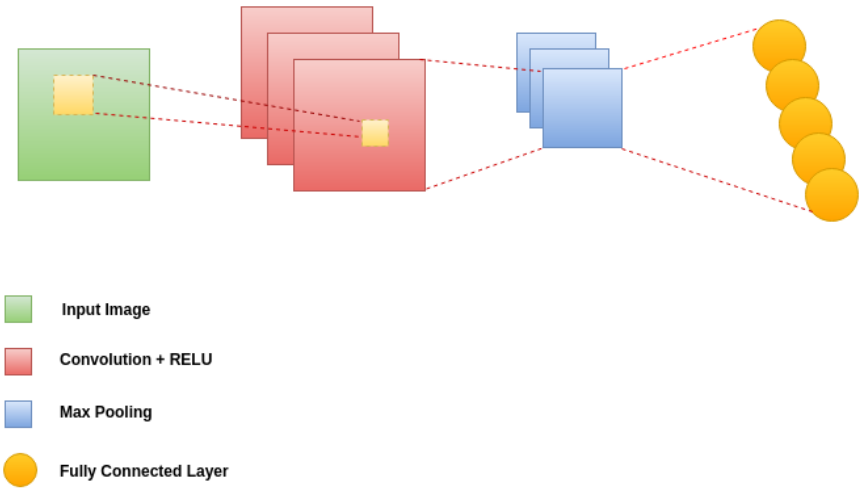


Figure 2.1: A general architecture of Convolutional Neural Networks specifying different layers.

- **Convolution Layer** - These are based on convolution operation where the layers consisting of kernels or filters convolve with the input to generate feature maps. The filters slide through the image by an amount called stride. As the filter slides through the image, it multiplies the values in the filter with the original pixel values of the input. The resultants are summed together to get the output.

- **Pooling Layer** - The pooling layer aims to reduce the dimensionality of the feature map, which decreases the parameter and complexity of the model. Pooling also helps in providing translation invariance. Some of the popular pooling techniques are max pooling, average pooling, and global pooling layer.

- **Fully Connected Layer** - Similar to feed forward neural network, neurons in this layers are connected to every neuron in preceeding and succeeding layer. Therefore, they have full access to the input volume as compared to convolution layers.

2.2.1 Activation Function

As convolution operation is linear, a nonlinear activation function is placed after the convolution layer to add nonlinearity to the feature map. ReLu and leaky ReLu activation functions have been used in this work.

- **Rectified Linear Unit** - It applies the function f(x) = max(0,x) on the activation map, therefore, restricting the negative component of the output to propagate through the network and activations are threshold at zero.
- **Leaky ReLu** - In contrast to ReLu, leaky ReLu does not vanish the negative part of the input but instead, it allows a small gradient for negative input.

$$f(x) = 1(x < 0)(\alpha x) + 1(x >= 0)(x) \tag{2.1}$$

Figure 2.2 shows the graphical comparison of ReLu and Leaky ReLu.

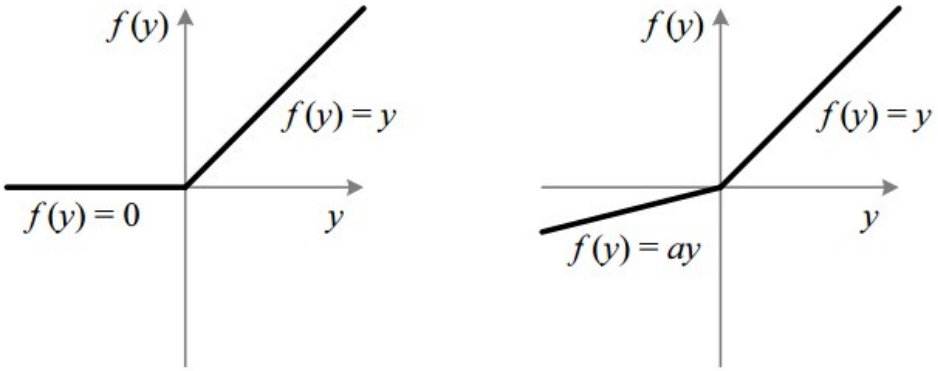


Figure 2.2: Comparison between ReLu and Leaky ReLu. [ReLu 18]

2.2.2 Batch Normalization

Batch Normalization was proposed by [Ioffe 15], it is generally applied before the activation function and resulted in better convergence of the model i.e it drastically reduces the training time. Deep neural networks are trained in mini-batches, and the distribution of these batches usually differs from each other resulting in a covariate shift. Due to the covariate shift, layer activations have to change quickly to compensate for the shifting distribution of the mini-batch. This complicates the training procedure and enforces to use small learning rates and precise weight initialization to prevent the network from diverging during the training. Batch normalization algorithm helps in alleviating this problem by normalizing the activations of each layer by zero mean and unit variance thus providing a more stable input. Batch normalization helps to reduce the training time of the model, enables a higher learning rate, prevents saturation of activation functions, and avoids the use of the Dropout layer used for regularization.

The recent advancement in computation power and ample availability of datasets has contributed immensely to CNN research. After the immense success of AlexNet in the 2012-ILSVRC challenge, a plethora of novel CNN architectures that could outperform

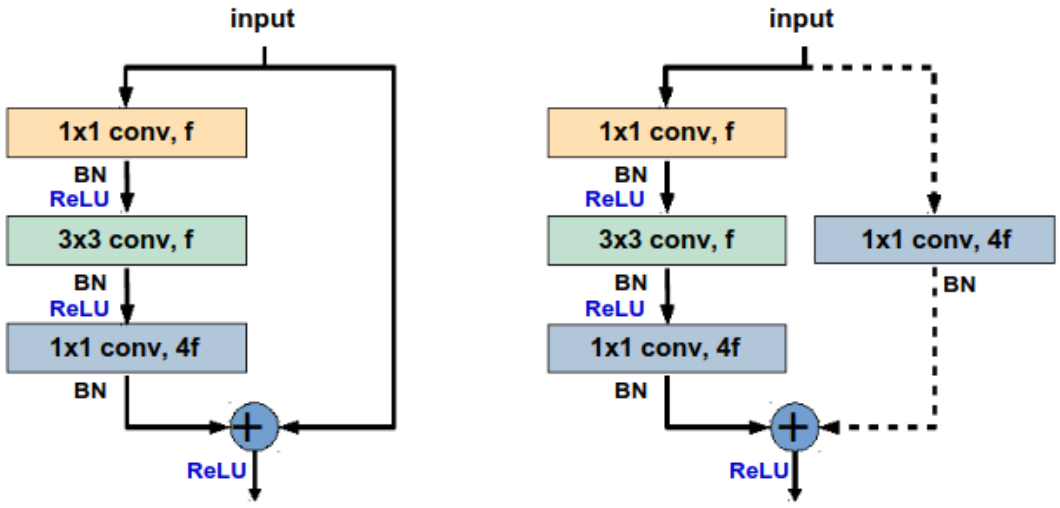


Figure 2.3: A Residual block for ResNet50. (left: identity shortcut, right: projection shortcut) Source: [Rezende 17]

humans in classification tasks has been proposed. The two architectures which are used in this work are ResNet50 and DenseNet-169. Brief working of mentioned architectures is explained in the next section.

2.2.3 ResNet

The problem with the CNN arises when more layers are added to make it deeper networks, it tends to lose the generalization capability. This was one of the drawbacks of VGG networks as their performances started to degrade on adding more convolution layer due to vanishing gradient problem. To overcome this problem of vanishing gradient, [He 15] introduced the idea of residual blocks. With the help of a residual block, the number of layers in the network can be increased without worrying about the vanishing gradient problem. The core concept of the residual block is to introduce an identity shortcut connection that skips one or more layers. This skip connection allows the gradient to pass through directly from later layers to initial layers. Such skip connections help to achieve faster convergence with better performance. The residual block can be formalized as :

$$y_l = f(h(x) + F(x, W)) \tag{2.2}$$

where x is the input of the block, and h is the skip connection. F defines the mapping between the consecutive convolution neural network and W is the weights of the filters of the convolution layers. f denotes the rectified linear unit y_l defines the mapping between residual block as a function of input and weights of the convolution layers.

The identity mapping does not have any parameters and is just added from the output of the previous layer to the next layer. But, sometimes the dimension between the input and the true output (y_l) does not match then a convolution layer of dimension 1x1 is needed to match the dimensions as shown in Fig 2.3(right)

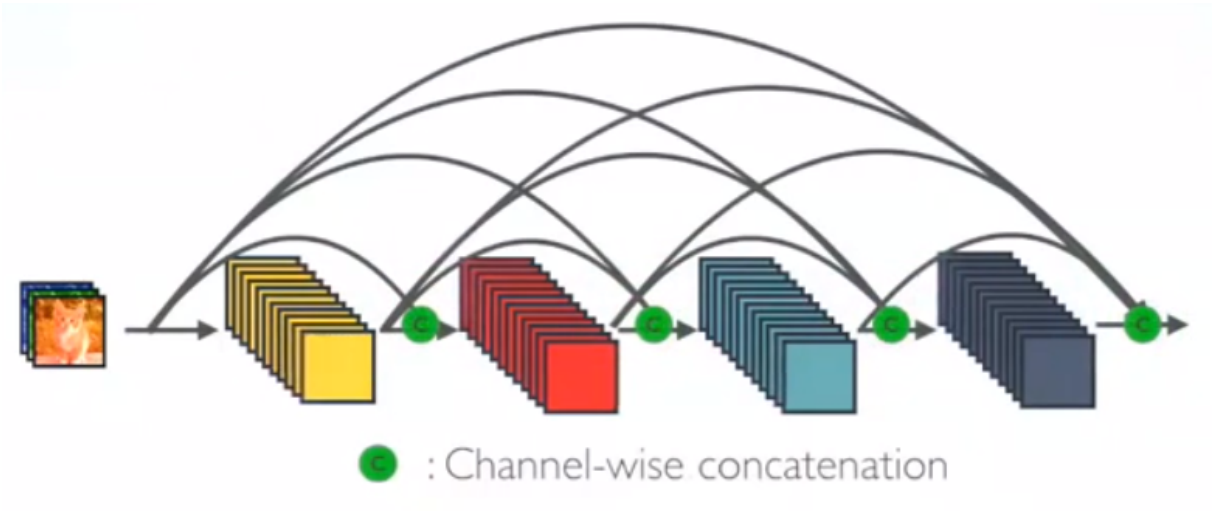


Figure 2.4: A Dense block with k channels. Source: [Sik-Ho Tsang 18]

2.2.4 DenseNet

Dense connected convolutional networks are composed of dense blocks. The reason they are called so because they are densely connected i.e. feature maps from all the previous layers forms the input of the subsequent layers. The major difference between residual block and dense block is that instead of summing outputs with the identity function in ResNet, dense blocks exploits the potential of the network through feature reuse, therefore it concatenates all the features maps from the previous layers as shown in Fig 2.4.

Consider input x, passing through a CNN comprising L layers. Let H_l be the output after nonlinear transformation, where index l denotes the particular layer. H_l consists of operations that includes batch normalization [Ioffe 15], ReLu [Nair 10] and 3x3 convolutional layer.

The equation for fully feedforward neural network connecting output of the lth layer as the input to (l+1)th layer after applying series of operation is given by :

$$\mathbf{x}_\ell = H_\ell \left(\mathbf{x}_\ell - 1 \right) \tag{2.3}$$

ResNets extended this functioning and introduced skip connection that further modified the above equation as

$$\mathbf{x}_\ell = H_\ell \left(\mathbf{x}_{\ell-1} \right) + \mathbf{x}_{\ell-1} \tag{2.4}$$

As DenseNets concatenates the feature maps coming from the previous layers the equation reformulates to

$$\mathbf{x}_\ell = H_\ell \left([\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{\ell-1}] \right) \tag{2.5}$$

where $|x_0, x_1, \ldots, x_{\ell-1}|$ denotes the concatenation of feature maps produced from layers 0.....l-1

One important characteristic of DenseNet is that the feature maps have the same dimension throughout the dense block. The number of channels increases after every layer and

transition blocks consisting of 1X1 convolution layers are used for downsampling the feature maps before passing through another dense block. The number of output feature maps of a layer is defined by the hyperparameter growth rate (k) that regulates how much information is added to the network after each layer. Let H_l produces k feature maps, then for lth layer, the total number of feature maps is given by equation 2.6, where k_0 denotes the number of channels in input layers.

$$k_l = k_0 + k * (l - 1) \tag{2.6}$$

2.3 Depth Estimation

Depth estimation is a computer vision task that is designed to estimate depth from a 2D image. 2D representation of the scene is essential for most of the applications. However, few applications require information in three dimensions, such as robotics, where 3D information is necessary for actuators to initiate actions to understand 3D geometry of a scene. The simplest example of how depth could be perceived is our eyes. Each eye captures its view of the surroundings and sends it to the brain for processing. The brain compares both left and right images and minor displacement between two viewpoints called binocular disparity helps the brain to estimate depth. The ability of the brain to extrapolate depth based on the binocular disparity between the two images is called as stereo vision [Depth-Estimation 18]. Researchers and scientists have tried to use this concept to compute depth from the environment. In computer vision, depth estimation is commonly achieved using stereo cameras. Depth estimation could be achieved by hardware as well as different algorithms that could be useful to process the depth information. First, let's get an insight into different hardware from which depth can be estimated.

- Dual Camera Technologies - Cameras in electronic devices have small distances between them to capture images from two different viewpoints that could emulate stereo vision to extract depth information of the scene.
- Motion Sensors - These are advanced sensing devices consisting of RGB sensors, infrared sensors that map depth through structured light or time of flight algorithms. Infrared dots are projected on the environment, and CMOS sensors are used to receive the reflected dots. The difference between the position of infrared dots and the received reflected dots gives the depth information. Fig 2.5 shows a Kinect motion sensor.
- LIDAR - Light Detection and Ranging is an active remote sensing device that is widely used to estimate the depth of the scene in self-driving cars. Lidar emits rapid pulses of lights on the surface from the lasers. When it hits the target, it gets reflected to a sensor which, measures the time taken for the pulse to return after bouncing off from the target. The distance of the object is estimated from the time taken by the pulse and speed of light. Lidar point cloud for a scene is shown in Fig 2.6

Depth estimation using novel algorithms is an active research topic nowadays. Deep learning methods have pushed the limits of various approaches that were possible in the

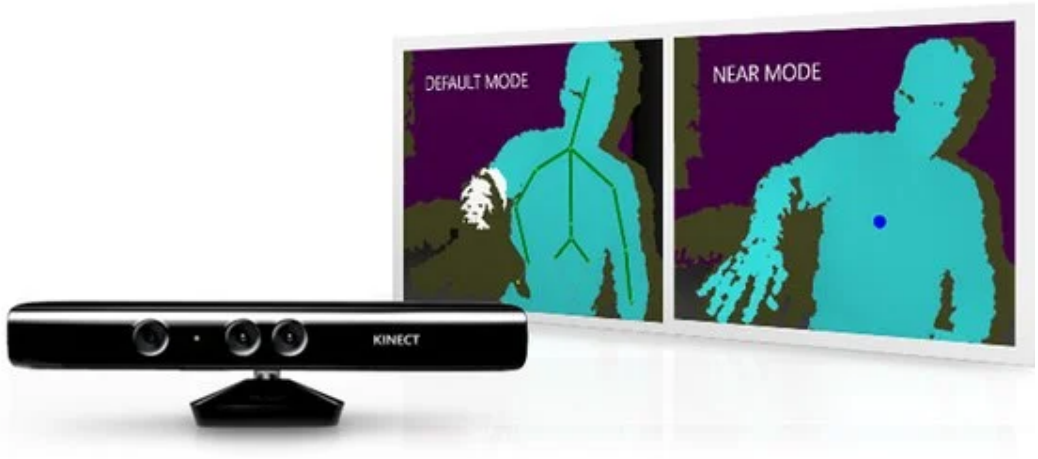


Figure 2.5: A Kinect sensor. [Microsoft Kinect 12]

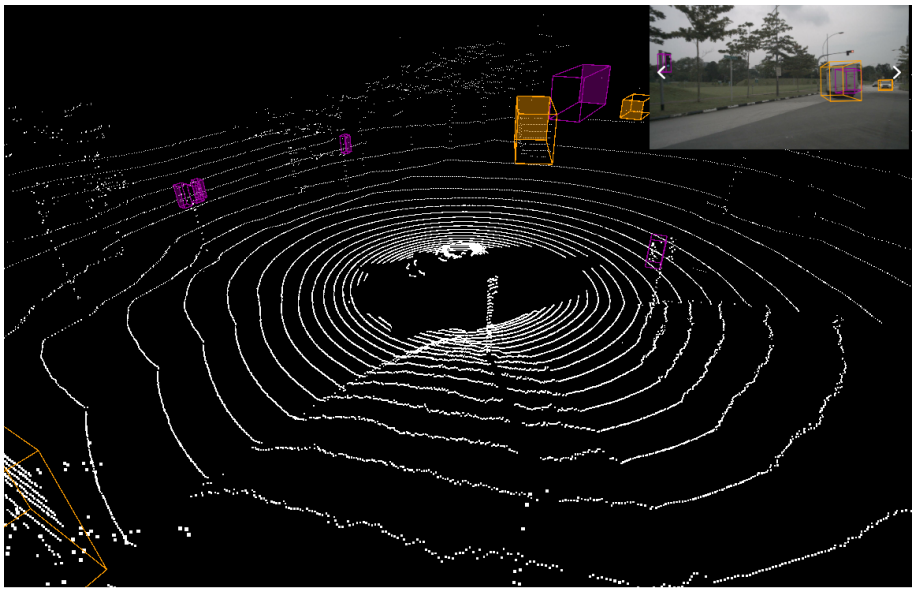


Figure 2.6: LiDAR point cloud of an example scene in nuScenes. [Caesar 19]

image processing domain. Estimating depth from single image is an ill-posed problem because many scenes in 3D planes correspond to the same 2D plane. Another problem is the ambiguity in recovering the exact scale of the actual scene from the image alone. On the other hand, using a stereo camera requires a careful calibration of cameras for accurate triangulation that requires an extensive amount of computation. Illumination problems such as reflective surface, repetitive patterns, and occlusions further aggravate the problem of estimating the actual depth of the scene. Due to these limitations, researchers are much interested in estimating depth by using a single camera, which is small and cost-efficient. Traditional and deep learning approaches for depth estimation using monocular and stereo images are discussed in the related work section.

3. Related Works

There has been a lot of prior works for depth estimation from monocular images as well as multimodal data. Few of the works fuse Lidar points and camera features extracted from a CNN by applying different fusion techniques to fuse them. In our work, sparse data from Radar has been fused along with RGB images to predict the pixel-wise depth. First, traditional computer vision techniques would be highlighted, then recent supervised and unsupervised deep learning techniques that have shown unprecedented success on various depth estimation benchmarks will be discussed.

3.1 Traditional Computer Vision techniques

Depth extraction algorithms aim at obtaining a representation of the spatial structure of a scene. The projection of the image on the scene results in the loss of depth information. When a real scene is projected, for each pixel, only one point of the real scene is projected that results in loss of the depth during the projection process into the image plane. One standard method to estimate depth is a stereo vision, where two images are acquired using two cameras placed at a distance. By matching the corresponding points on the two images and applying triangulation, one can extract depth from a pair of images, as shown in Fig 3.1. Apart from this method, depth information could be calculated using structure from motion, which requires a moving camera and sequence of static scenes and indirectly from 2-D image cues such as texture variations and shading of the image. Given rectified stereo cameras with focal length f, baseline b and corresponding image points (x_l, y_l) and (x_r, y_r), depth could be calculated as (3.1) :

$$\hat{d} = b * f/(x_l - x_r) \tag{3.1}$$

where $(x_l - x_r)$ is the disparity of the given 3D point.

Classic methods for monocular depth estimation are usually dependent on handcraft features, and the use of graphical probabilistic models such as Markov Random Field (MRF) [MRF 12] or Conditional Random Field(CRF) [CRF 12]that make strong assumptions

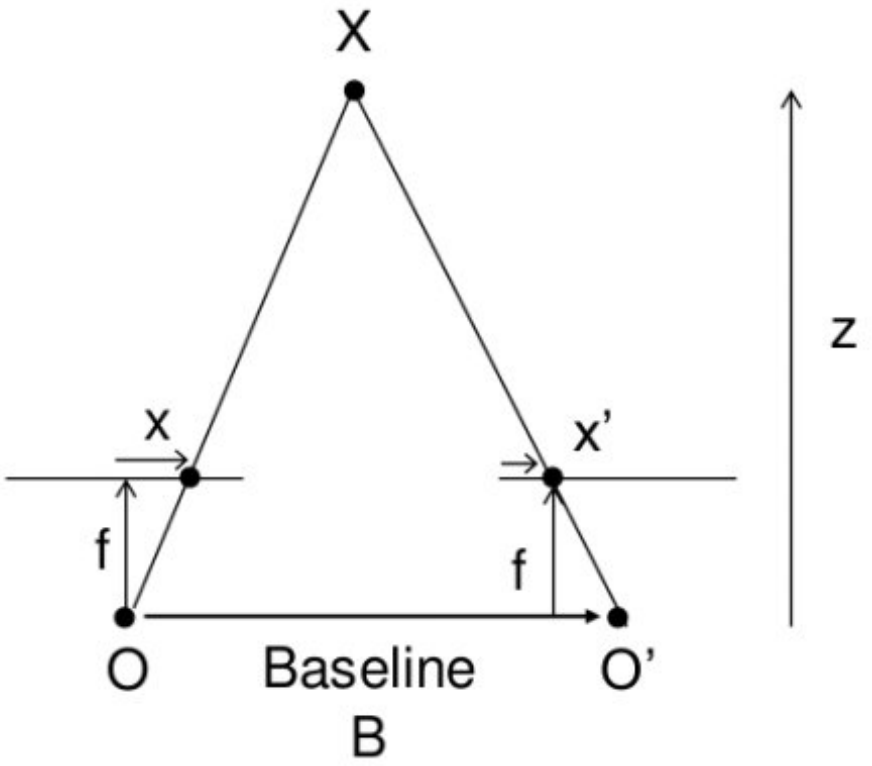


Figure 3.1: Disparity calculation from equivalent triangles formed by stereo images . [openCV 12]

about scene geometry. The first major contribution for estimating depth maps from single images was done by [A. Saxena 05] where they have used MRF to infer depth from local and global features extracted from the RGB image. They used a supervised approach where they collected data from a 3D laser scanner consisting of images and corresponding depth maps. They used MRF to trained depth rather than modeling the joint distributions of image features and depths. The image was divided into small patches, and depth is estimated for each small patch, and two types of features have been extracted, absolute features for determining the absolute depth and relative features for determining relative depths by calculating the magnitude of difference between neighborhood patches. The features have been chosen on the basis of local cues such as texture gradient, texture variation, and colors by convolving different edge and color filters over the patches. Gaussian and Laplacian MRF is used to model the posterior distribution of the depth. The following work was later extended for 3D scene reconstruction [Saxena 07]. In the next section, evolution of depth estimation architectures using Deep Learning is discussed.

3.2 Supervised Depth Estimation

In supervised approaches for depth estimation, a single image from the camera is used as an input, and depth data measured from range sensors such as RGB-D camera and laser scanners is used as ground truth for supervision in training. Few of benchmarked supervised learning techniques for depth estimation on monocular and stereo images are briefly summarized below.

Depth Map Prediction from a Single Image using a Multi-Scale Deep Network [Eigen 14]
One of the earliest work for prediction of depth using CNN's was done by Eigen et al. Most of the recent deep learning models used for depth estimation requires the usage of multi-scale features. The concept of using multi-scale features and directly regressing pixels for estimation of depth was first introduced in this paper. They used two scale network architecture consisting of a global coarse scale network and local fine scale network. Global coarse scale networks predict the overall depth of the image using a global view of the scene, where a local fine scale network refines the coarse predictions from the above network. Coarse scale networks consist of five convolution layers used for feature extractions along with max-pooling layers, followed by two fully connected layers to cover the entire image in their field of view. The network depth map output is one-fourth size of the input. The purpose of the local refine network is to make local refinements of the output from the coarse network so that it could align with local details such as edges and objects. The local refine network consists of three convolution layers with coarse features concatenated with the second convolution layer. The Fig 3.2 shows both the coarse and fine scale network. This network was trained using scale-invariant loss to measure the relationship of points in the scene irrespective of the global view of the scene.

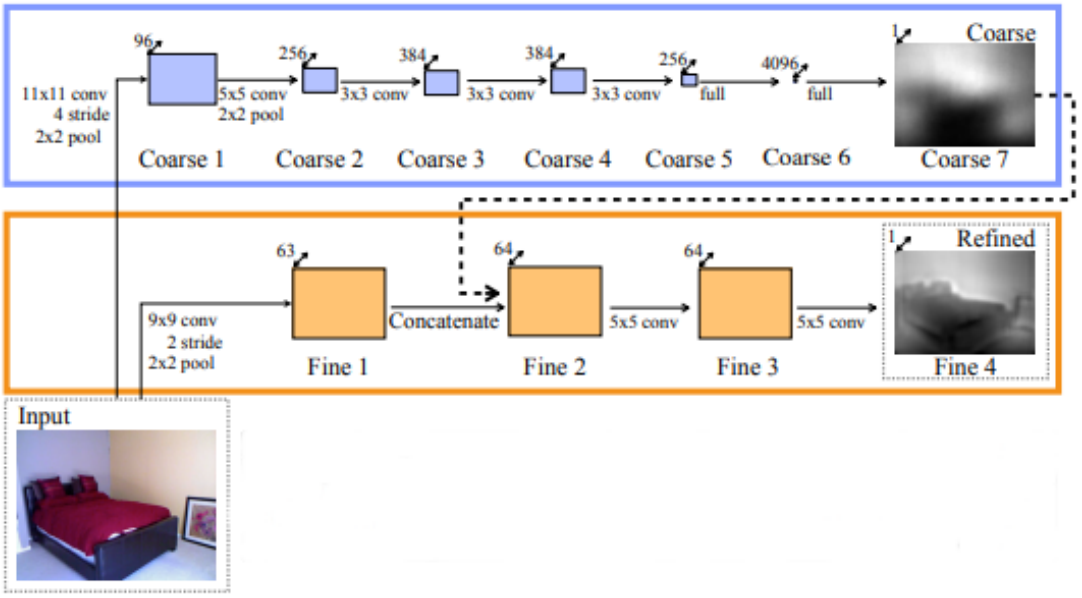


Figure 3.2: Multi scale deep network. Source: [Eigen 14]

Deeper Depth Prediction with Fully Convolutional Residual Networks [Laina 16]
This was another benchmarked supervised approach for estimating depth from a single image. They used encoder-decoder architecture, with AlexNet, VGG and ResNet were experimented to be used as an encoder, and custom decoder architecture called upprojection and fast up convolution which is described in the later section.

A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation [N. Mayer 16]
Inspired from [Fischer 15], they presented a CNN for real-time disparity estimation that gave the state of the art result. The architecture is similar to Flownet, and they have

extended that concept for prediction of scene flow and disparity. The network consists of contracting and expanding layers with long range interconnection between both. It takes stereo images as input that passes through a series of convolution layer to extract features from the images. The feature representation is then passed through a series of upconvolution layer, convolution layer, and alternate loss layer with features from contracting parts are concatenated with higher layer features to get the disparity prediction. Apart from these techniques, [Xu 18] proposed a deep CNN model with CRF to improve accuracy. The model has an encoder-decoder architecture experimented with AlexNet [Krizhevsky 12], VGG-16 [Simonyan 14], and ResNet-50 encoders respectively and fusion module consisting of continuous CRF's fusing with complementary information derived from the side output's of the networks decoder at multiple scales. They proposed two different fusion methodology based on CRF called as multi-scale CRF and cascade of specific scale CRF to implement a deep sequential network by stacking several blocks named as C-MF blocks. [Fu 18] proposed an architecture that achieved a state of the art results for depth estimation on KITTI [Geiger 12] and NYU dataset [Nathan Silberman 12]. They modified the depth learning problem from the regression problem to the ordinal quantized regression problem. Previous approaches used several upsampling layers to get a high-resolution feature map that complicated architecture, so to overcome that they introduced a novel space increasing discretization strategy to discretized continuous depth map into a number.

3.3 Unsupervised and Self Supervised Depth Estimation

Unsupervised and self-supervised approaches do not require ground truth for prediction, but it requires rectified stereo image pairs or sequence of images to train the depth estimation networks. Few of the recent unsupervised techniques that are used for depth estimation are briefly summarised below:

Unsupervised CNN for Single View Depth Estimation [Garg 16] They proposed an unsupervised approach for depth estimation by training a deep convolutional network similar to autoencoder, where stereo images pairs act as input and target. They trained the autoencoder for predicting scaled depth maps from the source image and generate an inverse warp of the target image by moving pixels from the right image along the scan-line to match the source input through reconstruction loss.

Monodepth [Godard 17]
Inspired by the work of [Garg 16] Godard et al. presented an unsupervised approach that estimates disparity as image reconstruction problem by exploiting epipolar geometry constraint and training the deep learning model to warp the left images to match the right images. The idea is to train a model that reconstructs the right image from the left image by using the right and left images from calibrated stereo cameras but at the time of inference, only a single image is used. Instead of directly predicting depth, the focus is to find a dense correspondence field so that when applied on the left image, the right image could be reconstructed. From the dense correspondence field, the disparity is predicted for

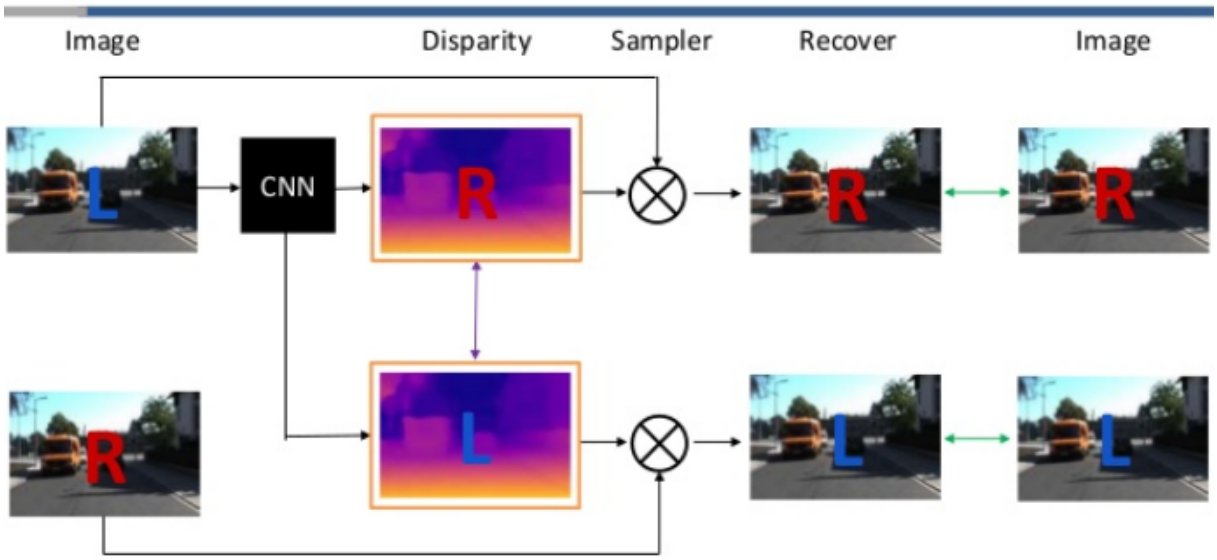


Figure 3.3: Monodepth architecture. Source: [Godard 17]

each pixel. Depth can be estimated from disparity (d) by given baseline (b) between the stereo cameras and focal length (f) of the camera using (3.1)

The architecture is loosely inspired by DispNet [N. Mayer 16], consisting of an encoder-decoder network with VGG-16 and ResNet-50 as an encoder, and decoder consisting of upsampling layers with supervision from the encoders using skip connections. The major insight of the model is that it can predict disparities for both left and right images by giving left image input to the network. The disparities are predicted using backward mapping using a bilinear sampler by enforcing the left-right consistency. The model architecture is shown in Fig 3.3. The network has been trained with a combination of loss functions consisting of structured similarity (SSIM) loss, and L1 loss forming image reconstruction loss. For disparities to be smooth, L1 penalty on disparity gradient is added to form disparity smoothness loss and Lastly, L1 left-right disparity consistency penalty has also been included in the loss function to produce more accurate disparities.

[Godard 18] proposed a self-supervised technique inspired by [Zhou 17a] for estimating depth in stereo image pairs and monocular videos. But training monocular images is slightly challenging as it requires ego-motion between temporal images pairs during training. Therefore, they proposed a pose estimation network apart from depth estimation to give camera transformation for sequences of images trained by different appearance matching loss function to avoid occluded pixel problem that occurs while training monocular images.

3.4 Multi-modal Depth Estimation

Some deep learning models, when trained with data from different modality performs exceptionally well for tasks such as semantic segmentation and object detection. Although most of the works propose to fuse camera features extracted from a deep convolutional network and Lidar point cloud for depth estimation and object detection tasks. For the job of depth completion where dense depth is estimated from sparse depth maps, Lidar proved

to be a vital modality along with RGB. [Ma 18b] proposed a supervised learning approach to train an encoder-decoder model by fusing sparse data with RGB sampled randomly from the ground truth depth image on the fly. The core idea was to explore how the injection of random noise would affect the performance of the network. Following their previous work,[Ma 18a] trained a supervised learning network with a semi-dense groundtruth depth map with a fusion of sparse lidar input and RGB images as inputs. They also proposed a self-supervised strategy with the fusion of sequences of RGB and Lidar point cloud as input. Training on series of images requires to estimate pose between two sequences, they adopted a model-based approach solving perspective n points problems to get the transformation between two frames using matching corresponding points. [Jaritz 18] proposed a supervised learning approach by utilizing a late fusion technique. They used NasNet [Zoph 17] and custom decoder to train data from both the modalities (RGB and sparse depth) for depth estimation and semantic segmentation tasks by replacing the last layer that gave the state of the art results on KITTI Dataset.

Our work is based on the incorporation of a radar point cloud with an RGB image to inspect the significance of distance data on depth estimation using deep learning models. Although the impact of radar point clouds by fusing with RGB images has been explored in object detection tasks such as [Chadwick 19], but in depth estimation tasks, it has not been explored much. The reason being the non-availability of publicly available datasets that provide data from different modalities and lack of ground-truth depth map that is quite expensive to generate for large scale datasets. Most of the previous works that are done for depth estimation using multi-modal inputs involved camera images and Lidar point clouds [Gansbeke 19]. The combination of both the sensors is complimentary and has benchmarked results in depth estimation tasks as the camera outperforms Lidar in capturing a denser and richer representation of the scene and Lidar excels in capturing distance information and it can paint a detailed 3D picture of the scene, which is difficult in case of the camera. On the other hand, Radar provides rich environment information based on received amplitudes, ranges, and the doppler spectrum. It captures larger distance information as compare to Lidar and is robust to varying weather conditions. Radar sensors can supplement camera sensors in low visibility areas by providing distance data that could help in achieving low-level autonomy in the vehicles. Through this work, the impact of Radar distance data along with RGB is explored. Further, it lay the scope for the exploration of better sensor fusions algorithms involving different sensors along with Radar and camera. A synthetic dataset consisting of Radar data with zoomed and standard camera images has been created that has been trained using the CNN's to study the impact of distance data on estimating depth on both types of images. A supervised learning approach is used for training the encoder-encoder network. The approach, implementation, and network design are explained in the next chapter.

4. Approach and Implementation

This chapter deals with the methodology, architecture design, and datasets used for this thesis. The richness, different sensing modalities, and training, validations, and testing distribution of the datasets are discussed. In the further section, the backbone network used for estimating depth by fusing sparse data such as Radar with RGB images is explained.

4.1 Datasets

With the increasing adoption of AI in all sectors, choosing the right datasets in the right format is the key to outstanding solutions. It is known that training a deep learning network requires an enormous amount of data. Availability and selection of correct datasets for the tasks involving machine learning is a challenging job. For the scope of this thesis, multi-modal datasets are required comprising of RGB images and sensing modalities such as Lidars and Radars. Using large multi-modal datasets with the diverse driving environment, contrasting weather, data from different sensors can significantly improve the network's accuracy and generalizing ability. For autonomous vehicle driving, few of the benchmarked datasets are available such as KITTI [Geiger 12], Waymo open dataset [Sun 19], Oxford Robot Car [Maddern 17]. It is a strenuous task to find the dataset containing the sensing modalities as there is only a single publicly available dataset (Nuscenes) to the author's knowledge that comprised data of both the range sensors (Radars and Lidars). Apart from using Nuscenes as a real dataset, an unreal dataset using an autonomous driving simulator Carla [Dosovitskiy 17] consisting of Radar sensing modalities is created. For the Nuscenes dataset, the ground truth depth map is created using multiple Lidar scans, and it has the problem of overlapping scenes and ambiguous image regions. A synthetic dataset overcomes this problem as it simultaneously captures images and a depth map of the scene. Thus, it provides a nearly perfect groundtruth depths which is suitable to train the model.

Details of the datasets are described below:

- **Nuscenes Dataset** - Nuscenes dataset [Caesar 19] is a publicly available large scale dataset for autonomous driving by Aptiv Autonomous Mobility. The dataset is made

up of 1000 scenes collected from Boston and Singapore, two cities are known for their dense traffic situations and highly arduous driving conditions. It consists of around 1.4 million camera images, Lidar sweeps, Radar sweeps, and object bounding boxes captured from 6 cameras, 1 Lidar, 5 Radars, GPS, IMU. Nuscenes is the only real recorded dataset available that provides radar data. Data from 85 scenes consisting of approximately 3376 images, Lidar point clouds, and Radar point clouds has been used. Out of the images and point clouds mentioned above, 80-10-10 split for training, validation, and testing are created, respectively. Fig 4.1 shows different modalities of the Nuscenes dataset.

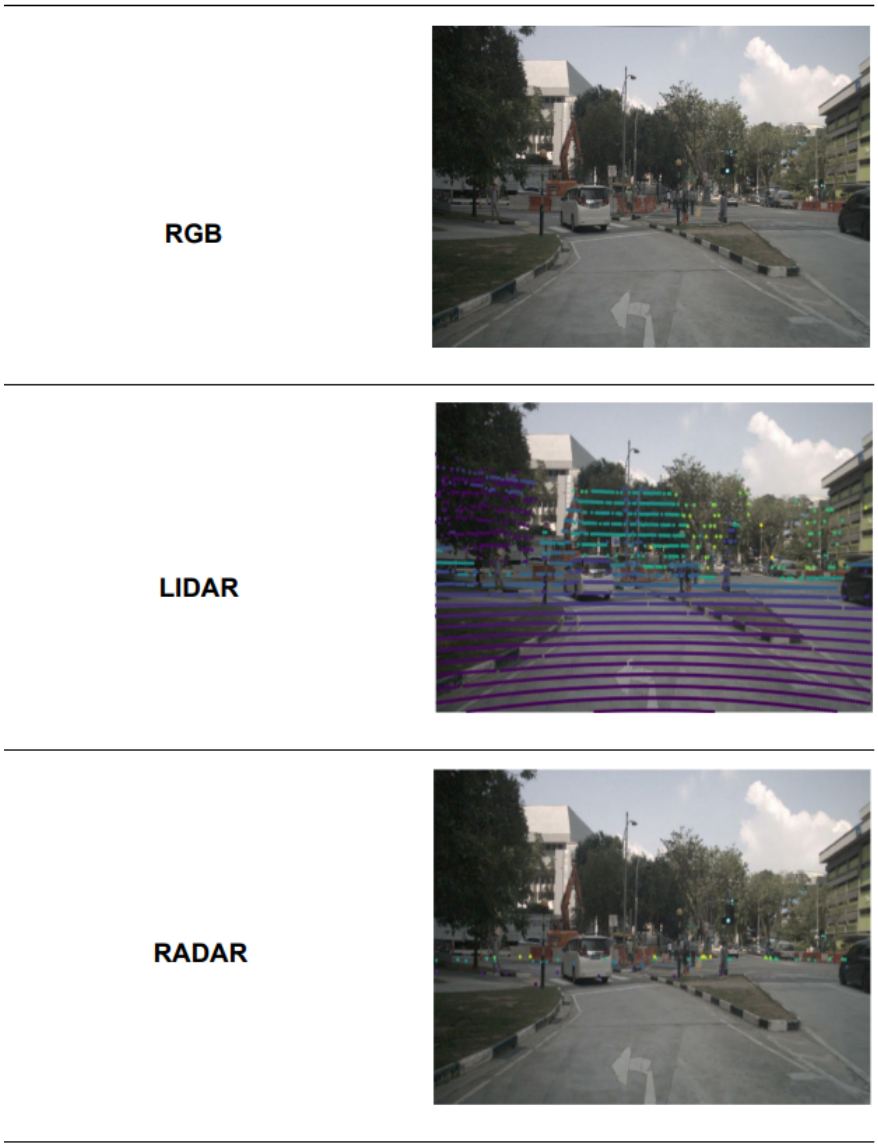


Figure 4.1: Sample images from the Nuscenes dataset comprising RGB image, Lidar and Radar point cloud projected on the image.

- **KITTI Dataset** - Kitti dataset [Geiger 12] is one of the best autonomous driving datasets available for benchmarking algorithm of depth completion and estimation,

3D object detection, and scene flow estimation. The KITTI dataset has been recorded from a moving Volkswagen wagon equipped with stereo color and grayscale cameras, 64 channel rotating Velodyne lidar scanner, with combined GPS/IMU inertial navigation system. One of the significant advantages of using the Kitti dataset for depth estimation is the availability of semi-dense depth map groundtruth, which is obtained by registering 11 consecutive lidar point cloud using Iterative Closest Point Algorithm [Uhrig 17]. Although the thesis focuses on incorporating Radar data with RGB images for depth estimation, the impact of the Lidar cloud point in enhancing the accuracy of depth estimation using the same network when fused with RGB images is also investigated. Fig 4.2 shows different modalities and groundtruth of the Kitti dataset.

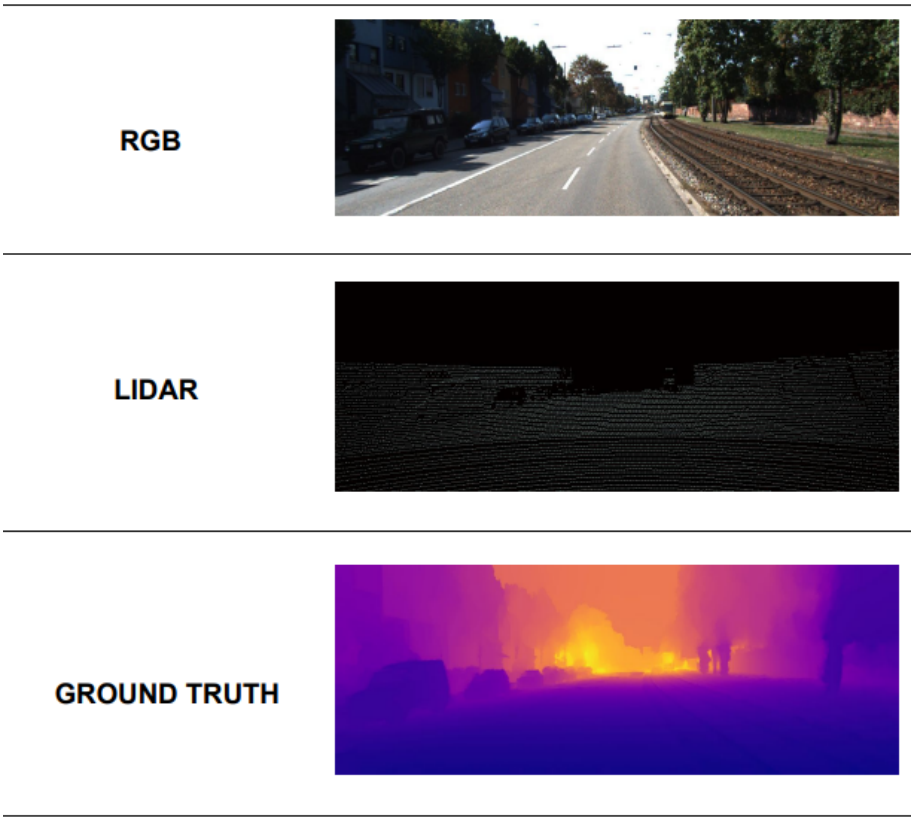
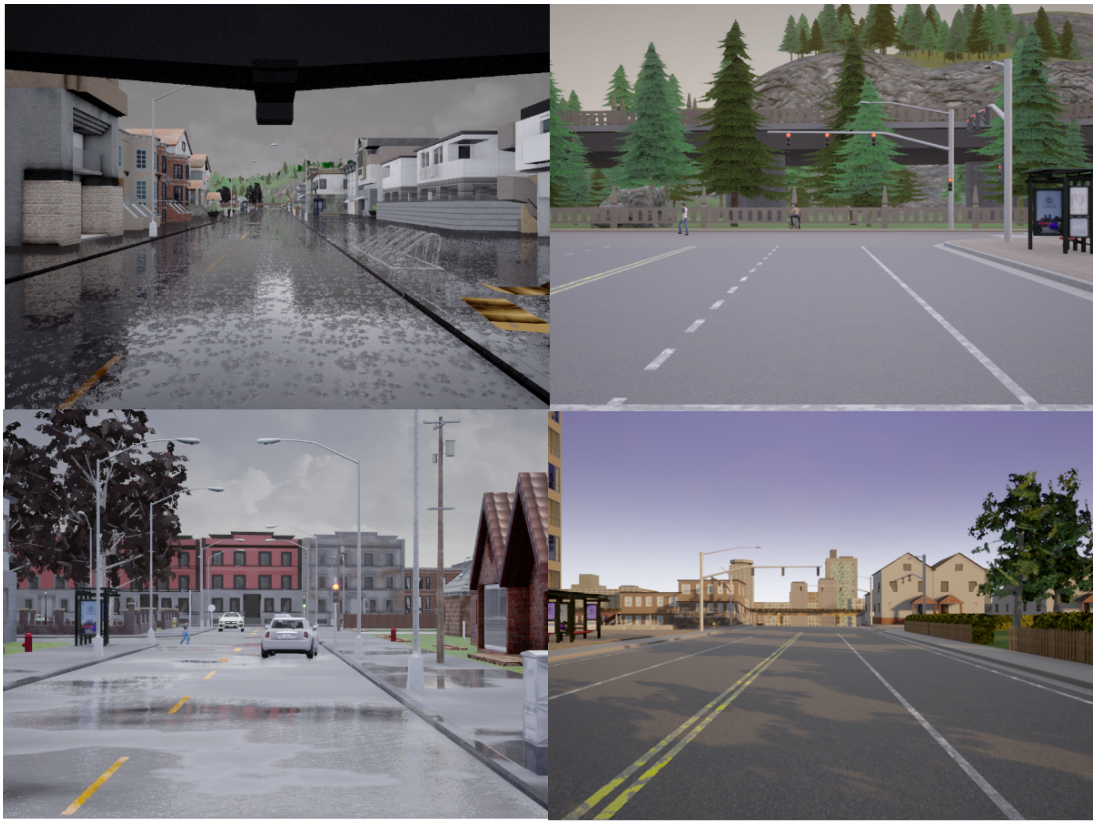


Figure 4.2: Sample images from the Kitti dataset comprising RGB image, Lidar point cloud projected on image and groundtruth depth map

- **Unreal Dataset** Although the nuscenes dataset is the only publicly available dataset that has radar point clouds, there is no official groundtruth depth map available for the same. A synthetic dataset has been created through the Carla simulator with nearly perfect groundtruth depths. Carla is an open-source emulator built on top of the Unreal Engine 4 (UE4) gaming engine for autonomous driving research [Dosovitskiy 17]. Fig 4.3a shows the GUI for the Carla simulator engine. It simulates a dynamic environment and provides an interface for the interaction of the world with the agent. It is a server-client simulation where the server runs the simulation and displays the scene. Carla provides a different driving environment composed

(a)



(b)

Figure 4.3: (a) Screen-capture of the Unreal Engine. (b) Images showing streets in Carla simulator with different weather conditions. Clockwise from top left: hard rain, cloudy noon, cloudy noon after rain, and clear noon.

of buildings, traffic lights, vegetations, terrains, vehicles, and pedestrians. Carla comprises RGB sensors, Lidar sensors, pseudo sensors that provide groundtruth for depth maps, and semantic segmentation. With the recent release of CARLA in December 2019, it has added a low fidelity Radar sensor to complete its sensor suites. Carla consists of the predefined environment of 5 towns of almost 2 Kms each with 10 predefined different weather conditions. Fig 4.3b depicts different weather conditions and streets in different towns in the Carla simulator. It can add custom weather depending on the user's requirement and can spawn dynamic objects such as different vehicles, peoples as per the requirement. The data is recorded on multiple scenes with different field of view, thus providing a versatile range of images and its corresponding depth map. Fig 4.4 shows different modalities and groundtruth of the Kitti dataset.

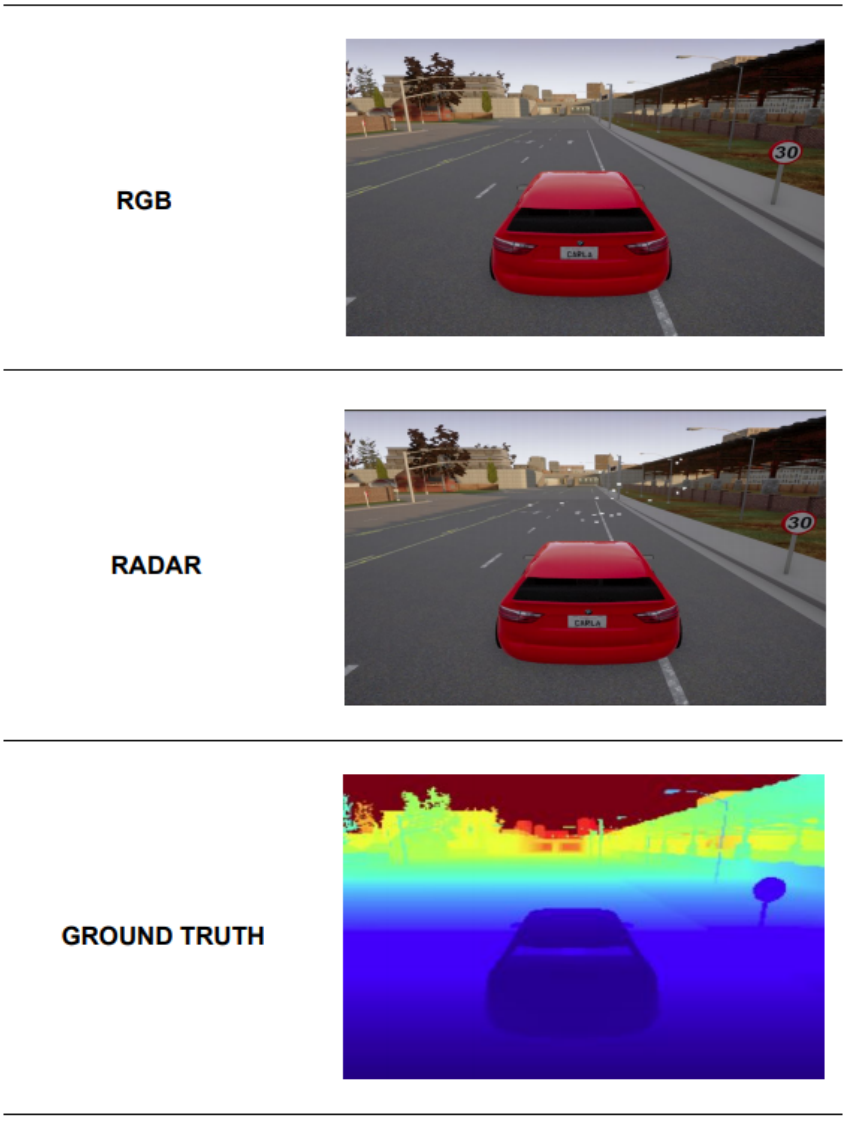


Figure 4.4: Images from the synthetic dataset comprising RGB image, Radar point cloud projected on image and depth map generated by Carla simulator.

Figure 4.5: Sample images from the university dataset which is used for inference. It includes top view images as well.

- **University Dataset** This dataset is captured at the university campus, which will be used for inference. It also contains some of the top view images taken from the crane at a construction site at Trier. Although the model has not been trained for top view images, it is used to check the generalization capability on unknown datasets. Fig 4.5 shows some of the images of this dataset.

Two different colormaps as shown in Fig 4.6 are used for interpretation of depth for groundtruths. Jet colormap is used for representing groundtruths in the Synthetic dataset, and plasma colormap is used for representing depths in KITTI and Nuscenes dataset.

Figure 4.6: Colormaps used for depth representation of groundtruths. For jet colormap, the distance of the object from the camera is represented from color ranging from blue to red whereas in plasma colormap the distance of the object from the camera is represented from color ranging from purple to yellow. The pixels represented by the colormaps encode information of its relative distance from the camera and not the absolute distance.

4.2 GroundTruth

Nuscenes dataset does not have their official depth map groundtruth, so groundtruth is created similar to recently released KITTI dataset groundtruth for depth estimation [Uhrig 17]. Lidar point clouds provide depth and reflectance information of the environment. For a Lidar point cloud, the depth information are stored by its cartesian coordinate (x,y,z), or distance $\sqrt{x^2 + y^2 + z^2}$. Besides that, its reflectance is determined by the intensity of the reflected light. There are a few ways to process the point clouds. One method was used in VoxelNet [Zhou 17b] where 3D point clouds get divided into 3D voxels, and each voxel gets assigned with the points so that it could be transformed into a unified feature representation. Therefore, point cloud gets encoded as a detailed volumetric representation, which was further connected to the region proposal network for object detection algorithms. However, the main problem with this method was that some of the voxels remained empty as Lidar points are sparse and irregularly spaced. Then, researchers from Stanford introduced Pointnet [Qi 16] and Pointnet++ [Qi 17] network which focuses on each point being processed identically and independently. Afterward, it aggregates the features extracted from several points via max pooling. Though it was tested in semantic segmentation or Lidar motion estimation tasks, in the case of processing Radar data, it has not been explored due to its excessive sparsity. A third method to represent point cloud is to project the point clouds into a 2D image so that they could be easily processed by the 2D Convolutional layer. For generating a dense depth map, a set of point clouds (10 before and 10 after frame of interest) has been accumulated and then it was projected onto the image. For every multi-modal dataset, the calibration of sensors plays a vital role. Based on nuscenes documentation, all important metadata, including calibration, annotations, are assembled in the form of a relational database. Calibration of sensors(Lidar, camera, Radar) is stored in calibrated sensors table, which stores the extrinsic parameters that are expressed relative to the ego frame, i.e., the midpoint of the rear vehicle axle. Ego vehicle pose can be accessed through egopose table. It is expressed with respect to the global coordinate frame. The ego pose is the output of the Lidar map-based localization algorithm. [Caesar 19] In the Nuscenes dataset, each sensor has its extrinsic coordinates relative to the ego frame, which is parsed in the form of JSON. For concatenation of Lidar point clouds, every point clouds need to map into a common reference frame as all the Lidar sweeps are in different coordinates frame by transforming the point clouds into the global frame. The point clouds are transformed by applying transformation matrix created by fusing transformation from vehicle's ego frame to Lidar reference frame, global frame to vehicle's ego frame, vehicle's ego frame to global frame, and sensor coordinate frame to ego vehicle frame. All the relevant equations respect to the transformation is mentioned below.

Following notations have been used in the below equations:
Vehicle's Ego frame - **ego** , Sensor Frame - **velo** , Global Frame - **global**

For transformation of vehicle's ego frame to sensor frame

$$\mathbf{T}_{\text{ego}}^{\text{velo}} = \begin{pmatrix} \mathbf{R}_{\text{ego}}^{\text{velo}} & \mathbf{t}_{\text{ego}}^{\text{velo}} \\ 0 & 1 \end{pmatrix} \tag{4.1}$$

For transformation of global frame to vehicle's ego frame using pose from egopose tables

$$\mathbf{T}_{\text{global}}^{\text{ego}} = \begin{pmatrix} \mathbf{R}_{\text{global}}^{\text{ego}} & \mathbf{t}_{\text{global}}^{\text{ego}} \\ 0 & 1 \end{pmatrix} \tag{4.2}$$

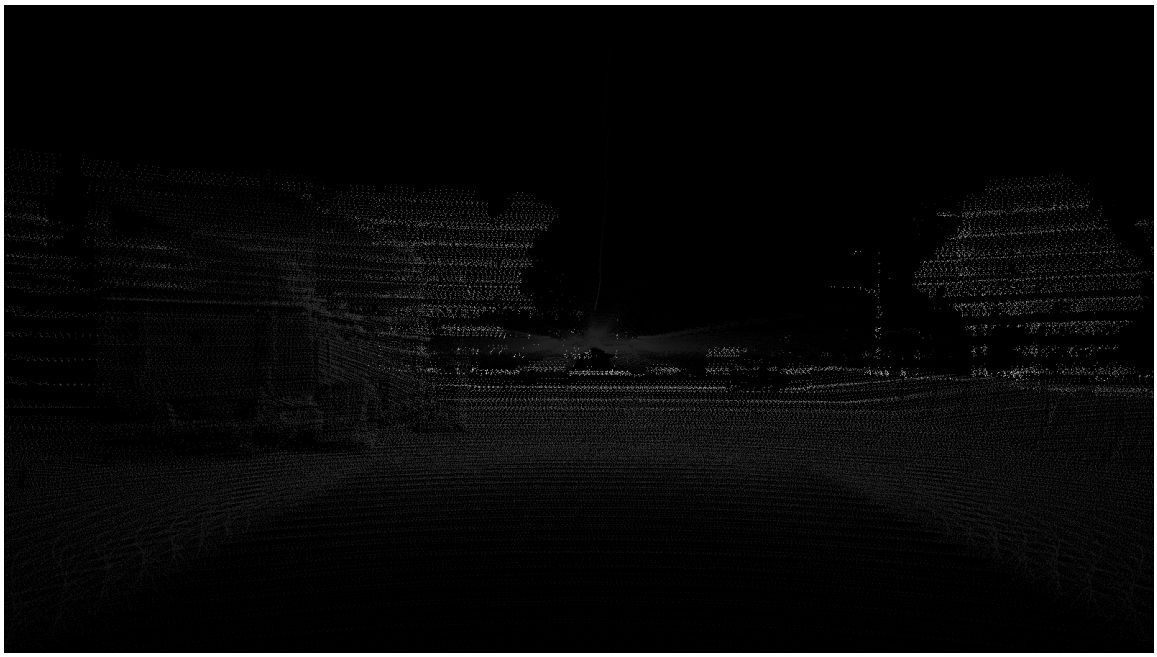


Figure 4.7: Sparse groundtruth generated by accumulating point clouds.

For transformation from global frame to vehicle's ego frame using pose from egopose tables

$$\mathbf{T}_{\text{ego}}^{\text{global}} = \begin{pmatrix} \mathbf{R}_{\text{ego}}^{\text{global}} & \mathbf{t}_{\text{ego}}^{\text{global}} \\ 0 & 1 \end{pmatrix} \tag{4.3}$$

For transformation from sensor frame to vehicle's ego frame using pose from calibrationsensors tables

$$\mathbf{T}_{\text{ego}}^{\text{current}} = \begin{pmatrix} \mathbf{R}_{\text{ego}}^{\text{current}} & \mathbf{t}_{\text{ego}}^{\text{current}} \\ 0 & 1 \end{pmatrix} \tag{4.4}$$

Transformation matrix by fusing all the transformations

$$\mathbf{T}_{\text{current}}^{\text{velo}} = \mathbf{T}_{\text{ego}}^{\text{velo}} \cdot \mathbf{T}_{\text{global}}^{\text{ego}} \cdot \mathbf{T}_{\text{ego}}^{\text{global}} \cdot \mathbf{T}_{\text{ego}}^{\text{current}} \tag{4.5}$$

A 3D point x in current sensor's frame to get converted into the reference frame where first point cloud is mapped can be transformed by fusion of all the above transformation :

$$\mathbf{y} = \mathbf{P}_{\text{r}}\, \mathbf{T}_{\text{current}}^{\text{velo}}\, \mathbf{x} \tag{4.6}$$

where P_r is the projection matrix to convert sensor frame to image frame. Fig 4.7 shows the sparse groundtruth generated from Lidar point clouds.

This groundtruth has been used for training the network, but it resulted in disappointing results. So a different approach has been tried inspired by [Levin 04] used for filling missing depth values. It is based on the principle that neighboring pixels that have similar intensities should have a similar color. The inpainting method was used by [Alhashim 18] for filling the Lidar generated depth map of the KITTI dataset by using a grayscale version of the RGB image as a weighting for smoothing the depth map. The Lidar point cloud is projected into the image, and then the inpainting algorithm was applied by the RGB

image on the projected point cloud, as shown in Fig 4.8. Human eyes are more sensitive to observe the changes in the colors, so the depth map is encoded with colormap to enhance the visualization. Colorizing images helps to pick out details and helps in noticing patterns in data in a more intuitive way. The pixels near to the camera are coded with blue color, and the color changes from blue to yellow as the pixels values differ when the object moves farther from the camera.

Figure 4.8: Inpainted groundtruth.

4.3 Proposed Architecture

In this section, the overall architecture of the proposed network used for depth estimation is discussed. In this work, two different architectures have been experimented to have a better comparison between different modalities. The subsections consist of a brief explanation of encoder-decoder architectures and different input modalities used in training the network, followed by the description of loss functions used in training these networks. In further sections, image augmentation techniques and fusion methods used for concatenation of (RGB and the sparse data) are explained.

4.3.1 Encoder-Decoder Architecture

In computer vision, dense problems that require per pixel predictions such as depth estimation and semantic segmentation are usually resolved by using encoder-decoder architecture. Encoder maps raw inputs into feature representation and downsamples the spatial resolution of the inputs. In contrast, the decoder takes those feature representations as inputs, then processes it and gives output that is the closest match of the original input. The working of the decoder is opposite to that of encoders. Decoders consist of one or more layers that increase the spatial resolution of the downsampled feature representations from the encoder. Few of the upsampling layers that form the decoder include transposed

convolution, Decomposed transposed convolution, Bilinear upsampling followed by the convolution layer. An encoder-decoder architecture is shown in Fig 4.9

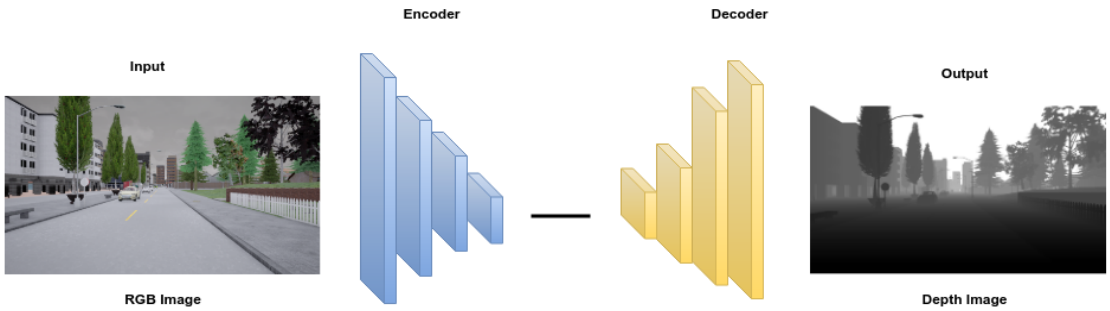


Figure 4.9: A simple encoder-decoder architecture.

For depth estimation tasks using two different modalities, it requires particular attention to the fusion techniques for fusing the modality so one could extract robust features from both different modalities. Different ways of interaction among multimodal features are explored to check whether or not it leads to the optimal performance gain. Early fusion technique has been used to fuse both the modalities as Radar data is too sparse. Channel-wise concatenation and element-wise addition of RGB and Radar data have been used to merge both the input modalities. In channel-wise concatenation, both the modalities are fused to form a four-channel input, whereas element-wise addition forms a three-channel input similar to RGB modality. Both the networks Dense Depth and Upproj that is used in this work have a universal encoder consisting of a benchmarked pre-trained network with a variation in the implementation of the decoder.

4.3.2 Dense Depth Architecture

The architecture used for depth estimation in this thesis is inspired by work done by [Alhashim 18]. The encoder and decoder layers are kept similar to the original architecture, but the network is trained on multi-modal input with different loss functions and hyperparameters in contrast to the original work. The model is trained with DenseNet-169 [Huang 16] and ResNet-50 [He 15].

The encoders used are DenseNet-169 and ResNet-50 networks that are pre-trained on Imagenet [Deng 09]. As the pre-trained networks are restricted to 3 channel inputs, a 1x1 convolution layer has been used to truncate four-channel input to three-channel before passing the input to the network. A 1x1 filter only has a single parameter for each channel of the input, so it is widely considered for dimensionality reduction. For the decoder, a 1x1 convolution layer is used with the same number of output channels as the output of the encoder by removing its last layer. Afterward, the resultant output is passed through four upsampling blocks, comprising of a bilinear upsampling layer, followed by two standard convolution layers of 3x3 filters with output channels set to half the channels of the input and the result from the upsampling layer is concatenated with the output of the same spatial dimension from the encoder. A 3x3 convolution filter is favored as it can look few pixels at a time extracting the vast amount of local features that are highly useful in later layers. Each of the upsampling blocks is followed by leaky Relu [A. L. Maas 13]

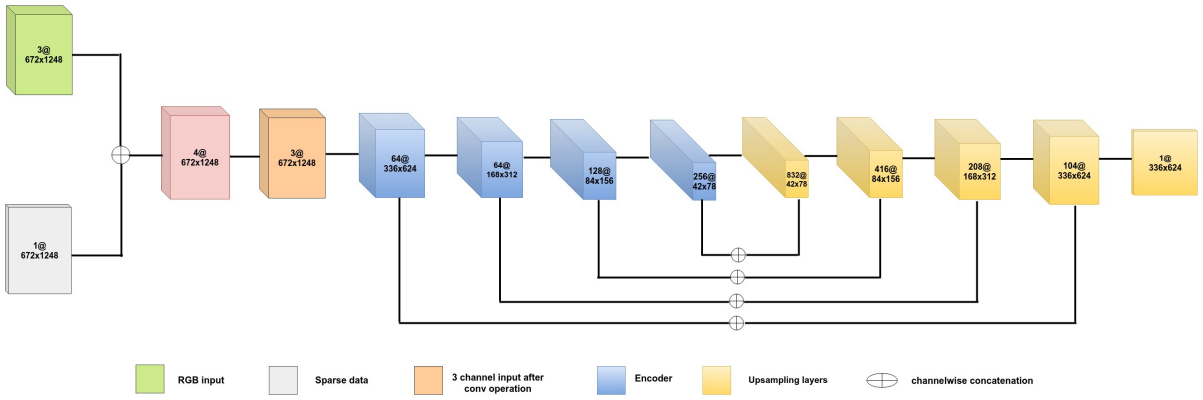


Figure 4.10: Encoder-Decoder architecture with RGB and sparse data fusion as input. For networks using only 3 channel input sparse data branch should be removed.

except for the last one. At last, a convolution layer of 3x3 filters is used as an output layer that results in the depth image of dimension half of the original image. Fig 4.10 gives an overview of the architecture. Table A.1 in the Appendix section describes different layers of the network with DenseNet-169 as an encoder.

4.3.3 UpProj Architecture

The UpProj architecture is inspired by [Laina 16]. This is a benchmarked network, and it has achieved the state of the art result for depth prediction on monocular images. In contrast to the original architecture only the upprojection layer has been used for this work. The network is trained with multi-modal inputs and inverse depth groundtruth. The motive to use this network is to compare the performance of two different decoders when trained with the same encoder. The encoder used is Resnet 50, with average pooling and fully connected layer removed from the network. The decoder is composed of 4 upsampling layers without any cross or skip connections from the encoder. The decoder has been incorporated with efficient upsampling blocks, which is called as upprojection blocks, as mentioned in [Laina 16], which proves to be more worthy appropriate for tackling high dimensional regression problem. The core idea of upprojection is to create an inverse residual block similar to Resnet that could be used for upsampling. After the feature

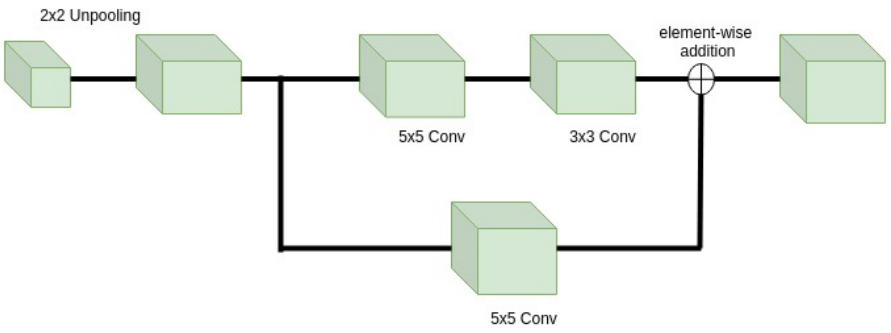


Figure 4.11: UpProjection convolution block

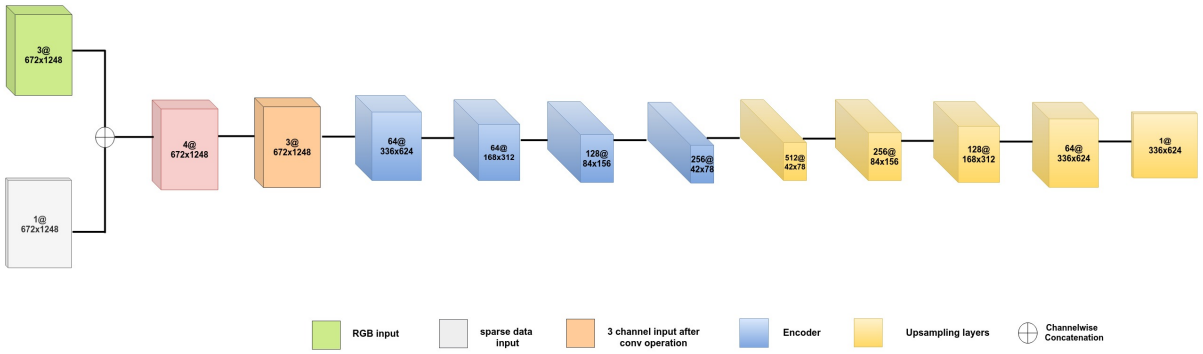


Figure 4.12: UpProj architecture with RGB and sparse data fusion as input. For networks using only 3 channel input sparse data branch should be removed.

extraction process of the encoder, the feature representation is passed to upprojection blocks consisting of transpose convolution that upsamples the inputs followed by 5x5 and 3x3 convolution layers on one branch and adding a projection connection followed by 5x5 convolution layer. The outputs of both the branches are added, similar to residual block [He 15], as shown in Fig 4.11. The consecutive upprojection blocks allow high-level information to be more efficiently passed forward in the network while increasing feature map sizes. Fig 4.12 gives an overview of the architecture. Table A.2 in the Appendix section describes different layers of the UpProj network with ResNet-50 as an encoder.

Different combinations of architectures and modalities trained using DenseNet and ResNet as encoders with different loss functions are described in Table 4.1.

4.3.4 Training

The network is trained on the Nuscenes, Kitti, and Synthetic datasets. The experiments conducted for this work include steps of preprocessing, including resizing the images and creating the train-val-test split. The input pipeline is designed using TensorFlow Data API, which is efficient and flexible. For the Nuscenes dataset , the images are resized to a resolution of 672 x 1248 and are randomly shuffled. For the synthetic dataset, the same procedure is followed, images are resized to a resolution of 672 x 1248. An unseen test set is created for evaluation, comprising of 500 images. The images are recorded on scenes that are completely different from the scenes used for recording the training data. For the Kitti dataset, the images are resized to a resolution of 384 x 1248 and are randomly shuffled. For evaluation on KITTI dataset, test images are sampled from KITTI Eigen split as used in [Eigen 14]. The Table 4.2 shows the statistics of images constituting the training, validation and test set for the above mentioned datasets.

The network is trained using inverse depth and also on original depth to compare the improvements on the predictions. When the model is trained using inverse depth, the final depth value is obtained by taking the inverse of the obtained prediction. The target depth map is defined as $d = m/d^*$ where d^* is the original groundtruth, and m is the hyperparameter, which has been initialized as 10 for the synthetic dataset and 50 for the real datasets. To determine the best value of hyperparameter m the experiment is done with different values (15,20,25,50) for both the synthetic and the real dataset but the minimum RMSE error and absolute relative error is achieved during evaluation with the

Architecture	DenseNet	ResNet	RGB	RGB (Fusion)	RGB (Add)	L1 loss	L2 loss	Berhu loss	SSIM loss	Image gradient
DenseDepth	✓	x	✓	x	x	✓	x	x	✓	✓
	✓	x	✓	x	x	x	✓	x	✓	✓
	✓	x	✓	x	x	x	x	✓	✓	✓
	✓	x	x	✓	x	✓	x	x	✓	✓
	✓	x	x	✓	x	x	✓	x	✓	✓
	✓	x	x	✓	x	x	x	✓	✓	✓
	✓	x	x	x	✓	✓	x	x	✓	✓
	✓	x	x	x	✓	x	✓	x	✓	✓
	✓	x	x	x	✓	x	x	✓	✓	✓
	x	✓	✓	x	x	✓	x	x	✓	✓
	x	✓	✓	x	x	x	✓	x	✓	✓
	x	✓	✓	x	x	x	x	✓	✓	✓
	x	✓	x	✓	x	✓	x	x	✓	✓
	x	✓	x	✓	x	x	✓	x	✓	✓
	x	✓	x	✓	x	x	x	✓	✓	✓
	x	✓	x	x	✓	✓	x	x	✓	✓
	x	✓	x	x	✓	x	✓	x	✓	✓
	x	✓	x	x	✓	x	x	✓	✓	✓
UpProj	x	✓	✓	x	x	✓	x	x	✓	x
	x	✓	x	✓	x	✓	x	x	✓	x
	x	✓	x	x	✓	✓	x	x	✓	x

Table 4.1: Different combinations of architectures along with the modalities, encoders, and loss functions used for training. RGB (Fusion) corresponds to the concatenation of Radar and RGB, whereas RGB (Add) corresponds to the elementwise addition of Radar and RGB.

Dataset	Total Images	Training Set	Validation Set	Test Set
Nuscenes	3376	2704	336	336
Synthetic	4645	3730	415	500
KITTI	10397	9000	1100	297

Table 4.2: Different datasets with number of images constituting training, validation and testing set.

above-mentioned value. One of the reasons to use this method is to reduce the loss term, as loss term tends to higher when the value of groundtruth is high. In this work, the batch size of 2 is used throughout all the experiments. The networks are trained for 20 epochs using Adam Optimizer [Kingma 14] with exponential decay rate for the first moment and the second moment estimates $\beta_1 = 0.9$, $\beta_2 = 0.999$, which is a default hyperparameter used in Adam optimizer. A constant learning rate of 0.0001 was experimented for training the network but the best performing results were achieved by reducing the learning rate to 20 percent after every 7 epochs. So, for all the networks used in this work, learning rate of 0.0001 which is reduced to 20 percent of its previous value at the eighth epoch is used for training. The learning rate vs. Epoch curve is plotted as depicted in Fig 4.13 that shows the possible learning rates used at particular epochs. All the experiments with their evaluations were carried out using Tensorflow 2.0 deep learning framework on a single machine with 64 bit Ubuntu 16.04 LTS operating system with 32GB internal memory, to have a fair comparison of all our methods.

4.3.5 Data Augmentation

Data Augmentation is a strategy of increasing the amount and induce diversity in the datasets for training Deep learning models. Few of the data augmentation techniques include rotation, flipping, zooming, sheering, cropping, and changing the brightness level of the image. These transformations on the datasets help to reduce overfitting and helps in the better generalization of the model. Augmentation of data could be classified as online and offline, where online augmentation is done on the fly during training. In the latter, the data is transformed beforehand and stored in memory. For this work, augmentation is done in an online manner with transformations, such as random horizontal flipping, random contrast, and brightness in the range of [0.9, 1.1] and [0.75, 1.25], respectively, are used with fifty percent chance.

4.3.6 Tensorflow

Tensorflow is an open-source library used for numerical computation and large scale machine learning tasks [Abadi 16]. There has been significant cleanup in API as compared to TensorFlow 1.x. In TensorFlow 1.x, computations are described in terms of a data structure called a computation graph that is composed of a set of nodes. To make use of the computation graph, sessions need to be invoked. Session handles the memory allocation and optimization that feeds the data into a graph and allows us to perform computation and generate the output. But in Tensorflow 2.0, tf.session is deprecated, and it uses tf.keras API for eager execution similar to python. Tensorflow data API is used for efficient and flexible designing of the image input pipeline, where each element corresponds to a pair of tensors consisting of an image and its ground truth.

4.3.7 Early Fusion

In deep neural networks, there are various ways to combine sensing modalities in early, middle, and late stages. Early fusion technique has been used to fuse the data from two

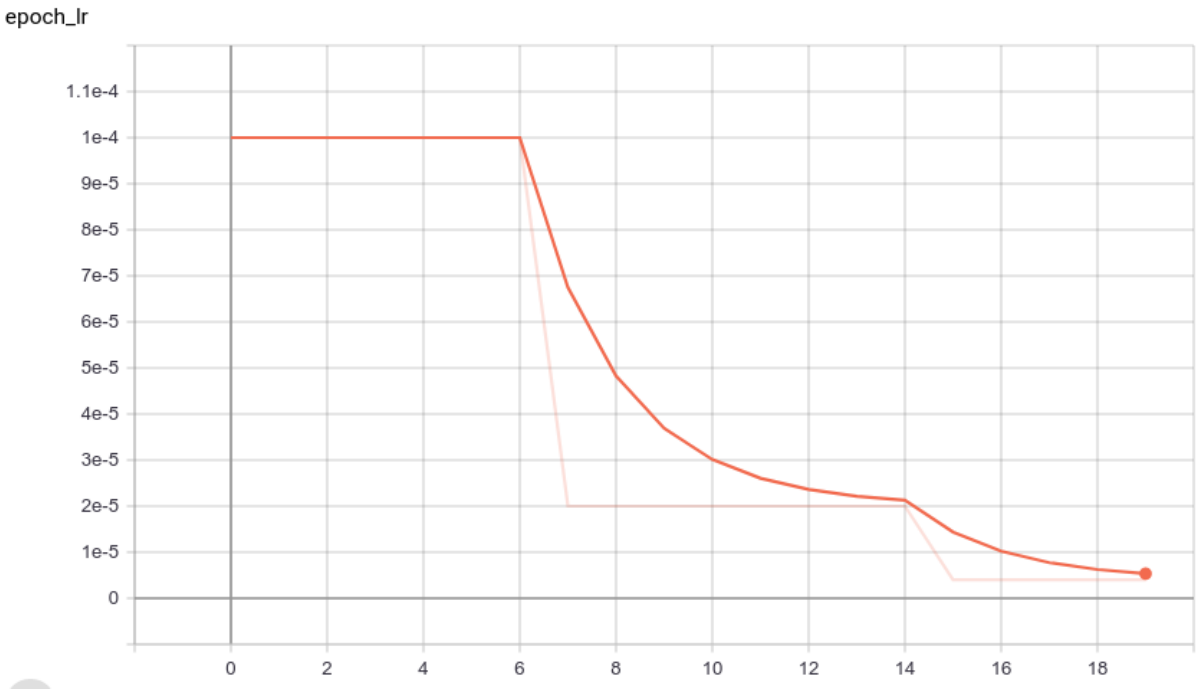


Figure 4.13: Learning rate vs Epoch curve

different modalities for our network as data from the radar is too sparse, training another expert, and then fusing the feature maps looked ineffective. The advantage of early fusion is that the network learns the joint features of multiple modalities at an early stage; thus, it fully exploits the information from the raw data. In our case, depth information from radar

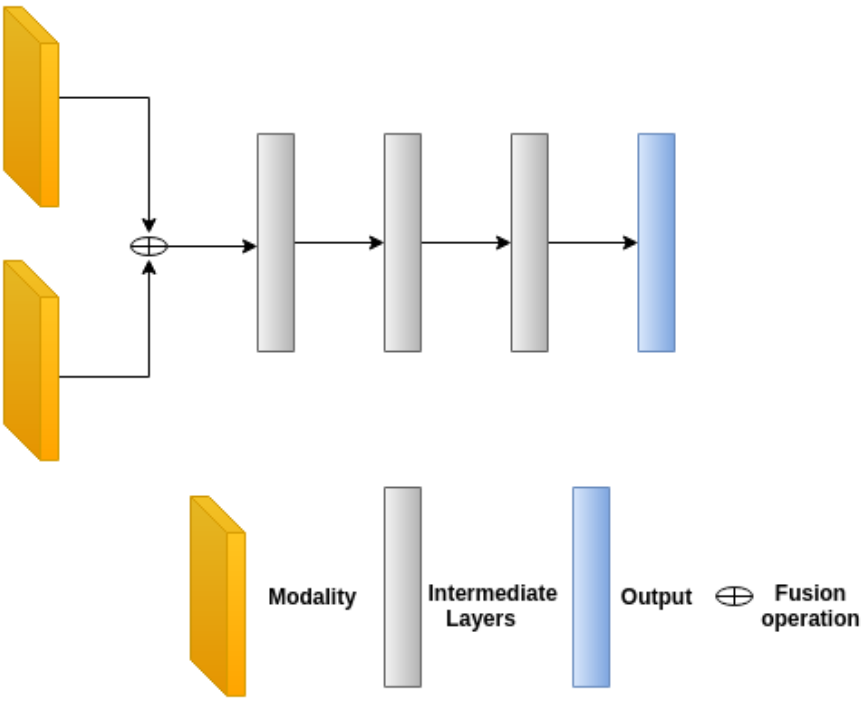


Figure 4.14: Illustration of early fusion method.

and geometrical information such as edges, corners, color information such as highlights, shading from RGB images could be utilized. Moreover, early fusion techniques require low memory as compare to middle or late fusion because it processes both sensing modalities simultaneously. One of the drawbacks of using early fusion technique is if it's needed to change one of the modality, then it requires the network to be retrained completely. Fig 4.14 shows the illustration of early fusion methods.

4.3.8 Loss Function

Deep learning models are trained using optimization algorithms that seek to minimize loss functions. A standard loss function for regression tasks considers the difference between ground truth depth map and prediction of the depth map from the network. The common loss function for regression is L1 and L2 loss. Different variants of loss functions such as scale-invariant loss [Eigen 14], inverse Huber loss [Laina 16] , the combination of smoothness and reconstruction loss [Godard 17] can be found in depth estimation literature. For encoder-decoder architecture, a loss function that considers the reconstruction of depth images is used by minimizing the difference of ground truth and predictions from the network. While each loss functions has its advantages and drawbacks, a linear combination of three different loss functions is used in this work as mentioned in (4.7)

if d_p^* is the pixel in predicted depth d* and d_p is the pixel in groundtruth depth d, and n is the total number of pixels for each depth image, loss functions is defined as

$$L(d, d^*) = \lambda L_{depth}(d, d^*) + L_{grad}(d, d^*) + L_{ssim}(d, d^*) \tag{4.7}$$

For the first loss term L_{depth} L1, L2, and Berhu loss [Zwald 12] has been examined along with the other two loss terms. The difference between the L1 and L2 loss functions is that L2 penalizes larger errors, but is more tolerant of small errors, whereas L1 loss does not produce a high error for large predictions error. L1 loss tends to make the reconstructed image less blurry as compared to L2 loss. The Berhu loss or inverse Huber loss is a combination of L1 and L2 loss as described in the equation (4.8) it is equal to the $\mathcal{L}_1(x) = |x|$ norm when $x \in [-c, c]$ and equal to $\mathcal{L}_2$ outside this range. It is more sensitive to small errors. The comparison between the networks trained with different loss functions is mentioned in Chapter 5.

$$\mathcal{B}(x) = \begin{cases} |x| & |x| \leq c \\ \frac{x^2+c^2}{2c} & |x| > c \end{cases} \tag{4.8}$$

$$L_{depth}(d, d^*) = \frac{1}{n} \sum_{p}^{n} \left| d_p - d_p^* \right| \tag{4.9}$$

The second loss term L_{grad} is the loss defined over the image gradients of the depth image and it returns gradients for each color channel.

$$L_{grad}(d, d^*) = \frac{1}{n} \sum_{p}^{n} \left| g_x \left(d_p, d_p^* \right) \right| + \left| g_y \left(d_p, d_p^* \right) \right| \tag{4.10}$$

The third loss is Structural Similarity Loss (SSIM). SSIM has been used commonly for image reconstruction tasks, which is based on the Structural Similarity (SSIM) index used

for measuring the similarity between two images. The SSIM index can be viewed as a quality measure for comparison between the two images provided [SSIM 04]. SSIM loss compares the neighborhood of target pixel between reconstructed and original images, whereas L1 and L2 loss is pointwise loss and compares loss pixel by pixel. SSIM compares corresponding pixels and their neighborhoods in two images that have a strong inter-dependencies. The three fundamental measurements used as a weighted combination in SSIM are luminance (I), contrast (C), and structure (S) as mentioned in [Ridgeway 15],

$$I(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \tag{4.11}$$

$$C(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \tag{4.12}$$

$$S(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \tag{4.13}$$

where μ_x, μ_y, σ_x, σ_y denotes mean pixel intensity and standard deviation of pixel intensity in image patch centered at x,y.

The loss function is defined as

$$L_{SSIM}(y,\hat{y}) = \frac{1 - SSIM(y,\hat{y})}{2} \tag{4.14}$$

The only weight parameter used is for the depth loss, where it has experimented with different weights parameter, and 0.1 is found to be the best parameter as threshold accuracy is higher in comparison to other weight parameters.

For UpProj architecture a combination of L1 and SSIM loss has been used to train the network. Although it was also experimented with L2 loss and Berhu loss, the best results are obtained using L1 and SSIM loss.

4.3.9 Evaluation Metrics

To evaluate the quantitative performance of the networks, standard metrics for depth estimation is used as mentioned in various depth estimation literature. [Eigen 14]. Each of the used metrics are explained below. d_i is the original depth of the pixel in depth image d and d_i^* is the predicted depth of the pixel. N is the total number of pixels for each depth image.

- Absolute Relative Error (abs rel)
  The absolute relative error is the most straightforward metric used for estimating depth per pixel-wise. The absolute relative error gives the measurement of how far the predicted value is from the target measurement relative to the target image. The absolute relative error is useful when the model is not concerned about the outliers. The large errors from the outliers contribute linearly to the overall error that makes

them more robust.

$$\frac{1}{N}\sum_{i \in N}\frac{|d_i - d_i^*|}{d} \tag{4.15}$$

- Square Relative Error (sq rel)
  The squared relative error gives the difference of squares between each pixel of groundtruth and predicted value relative to the target image. The squared relative error is useful when large errors need to be penalized.

$$\frac{1}{N}\sum_{i \in N}\frac{|d_i - d_i^*|^2}{d} \tag{4.16}$$

- Root mean squared error (rmse)
  RMS error is the measure that shows how spread out the residuals are, where residuals are the difference between the actual values and the predicted values. In RMS error as the errors are squared before they are averaged, it gives a relatively high weight to large errors. So, it is most useful when large errors are particularly undesirable.

$$\sqrt{\frac{1}{N}\sum_{i \in N}|d_i - d_i^*|^2} \tag{4.17}$$

- Threshold Accuracy

$$\begin{aligned} &\%\text{ of } d_i \text{ s.t. } \max\left(\frac{d_i}{d_i^*}, \frac{d_i^*}{d_i}\right) < thr, \text{ where} \\ &\text{thr} \in \left\{1.25, 1.25^2, 1.25^3\right\} \end{aligned} \tag{4.18}$$

The accuracy measurement, divides the pixels into error intervals governed by the threshold values. [Eigen 14] proposed these values that correspond to a prediction accuracy of {±25%, ±56.25%, ±95.31%} from the ground truth depth. For every metric except threshold accuracy lower values indicates better performance.

5. Experiments and Results

In this chapter, all the experiments performed using the specified architectures with their empirical evaluations and discussion about the results are described. The experiments are supported by their qualitative and quantitative results. Evaluation is performed for the Nuscenes dataset and Synthetic dataset. The inference is taken on the University dataset by the best performing model trained on both the datasets. The performance of the network trained by fusion of Lidar point cloud and camera images on the Kitti dataset and the Nuscenes dataset is also evaluated which is mentioned in the chapter 6

Hyperparameters are kept same throughout the experiments. For training, Adam Optimizer is used [Kingma 14] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, learning rate 0.0001 which gets reduced to 20 percent after every 7 epochs. The total number of epochs is set to 20 with a batch size of 2.

5.1 Evaluation on Synthetic dataset

For the Synthetic dataset training and evaluation is done on two separate class of networks using different encoders and loss functions as mentioned in Table 4.1
δ_1, δ_2, δ_3 are the threshold accuracy.
RGB Fusion [1] modality refers to channel-wise concatenation of RGB and Radar data, whereas RGB addition modality refers to the element-wise addition of RGB and Radar data. For Dense Depth architecture, the experiment has been done with L1, L2 and Berhu loss while keeping SSIM loss and image gradient unchanged in all the combination of loss functions [2].

[1] As the Radar point cloud is too sparse, it is not clearly visible when projected on the image. So, it is not been shown as an input modality along with RGB images when qualitative results are reported.

[2] In the following sections, if only L1, L2 and Berhu loss is mentioned it corresponds to the combination of that loss with SSIM and image gradient.

5.1.1 Model trained with original depth using Dense Depth

The model is trained with original depth using a combination of L1 loss, image gradient and SSIM loss. The fusion modality of Radar and RGB is the best performing model when trained with DenseNet. The absolute relative error and square relative error are decreased by 10% and 20% respectively as compared to the network trained only on RGB modality. All the modalities trained on DenseNet performs better than modalities trained on ResNet.The networks performed poorly as compared to networks trained using inverse depth so it was not trained further with other loss functions. Quantitative results are mentioned in Table 5.1.

Network	**Modality**	δ_1	δ_2	δ_3	**Abs Rel**	**Sq Rel**	**RMSE**
DenseNet-169	RGB	0.602	0.814	0.895	1.007	1.126	0.874
	RGB Fusion	0.626	0.844	0.893	**0.910**	**0.930**	**0.870**
	RGB Addition	0.571	0.779	0.875	1.107	1.16	0.898
ResNet-50	RGB	0.587	0.784	0.872	1.24	1.47	0.903
	RGB Fusion	0.565	0.782	0.872	1.175	1.36	0.979
	RGB Addition	0.595	0.800	0.881	1.158	1.38	0.924

Table 5.1: Comparison of different modalities trained with original depth using DenseNet and ResNet as encoders. The highlighted results shows that fusion modality trained on DenseNet performs better as compared to other modalities when trained with original depth.

5.1.2 Model trained with inverse depth using Dense Depth

The network is trained with inverse depth and final depth is obtained by taking the inverse of the obtained prediction.

Training with the combination of L1+Image gradient+SSIM Loss

The fusion modality trained on DenseNet has threshold accuracy (δ_1), (δ_2) better than other modalities. The model boosts δ_1 score from 63.4% when trained on RGB to 68% when trained along with the fusion of Radar data. The RMSE value of fusion modality is 0.820, which is lower than the model trained on RGB. The absolute and square relative error is decreased by 1.7% and 0.9% respectively. The RGB addition modality has RMSE value almost equal to fusion modality but has a higher absolute and square relative error. The models trained on DenseNet performed better on all the modalities as compared to models trained on ResNet as Densenet is a much more efficient feature extractor as it encourages reuse of features i.e same features are connected to other blocks. This allows gradients to flow directly to input, thus diminishes the chances of vanishing gradients problem. The addition modality also performed better in the case of DenseNet than RGB, the reason being that the Radar point cloud is sparse, in addition with RGB, it may act as a noise that can have a regularizing effect and reduces overfitting which results in better performance of the model. On visualizing the predictions, the fusion modality trained on DenseNet looks much smoother and better as depicted in Fig 5.1. The scene reconstruction is almost the same for all the predictions but the prediction of the distance of the mountain from the camera as shown in the rectangular box marked on the image for fusion modality is very much similar to the groundtruth as compared to predictions from other modalities. For the predictions from the model trained on ResNet, the fusion modality performs worst according to the metrics used for the evaluation as compared to other modalities. It can be depicted from the predictions denoted by a box on the image below. Quantitative results shown in Table 5.2 and Qualitative results shown in Fig 5.1.

Network	**Modality**	δ_1	δ_2	δ_3	**Abs Rel**	**Sq Rel**	**RMSE**
DenseNet-169	RGB	0.634	0.875	0.976	0.199	0.216	0.850
	RGB Fusion	0.680	0.886	0.976	0.182	0.205	0.820
	RGB Addition	0.662	0.881	0.974	0.193	0.209	0.822
ResNet-50	RGB	0.663	0.870	0.971	0.192	0.216	0.837
	RGB Fusion	0.645	0.853	0.967	0.201	0.233	0.880
	RGB Addition	0.656	0.859	0.972	0.197	0.221	0.850

Table 5.2: Comparison of different modalities trained with inverse depth using L1 loss. The highlighted results shows the best performing model in terms of RMSE and absolute relative error when trained with L1, Berhu and L2 loss along with image gradient and SSIM Loss.

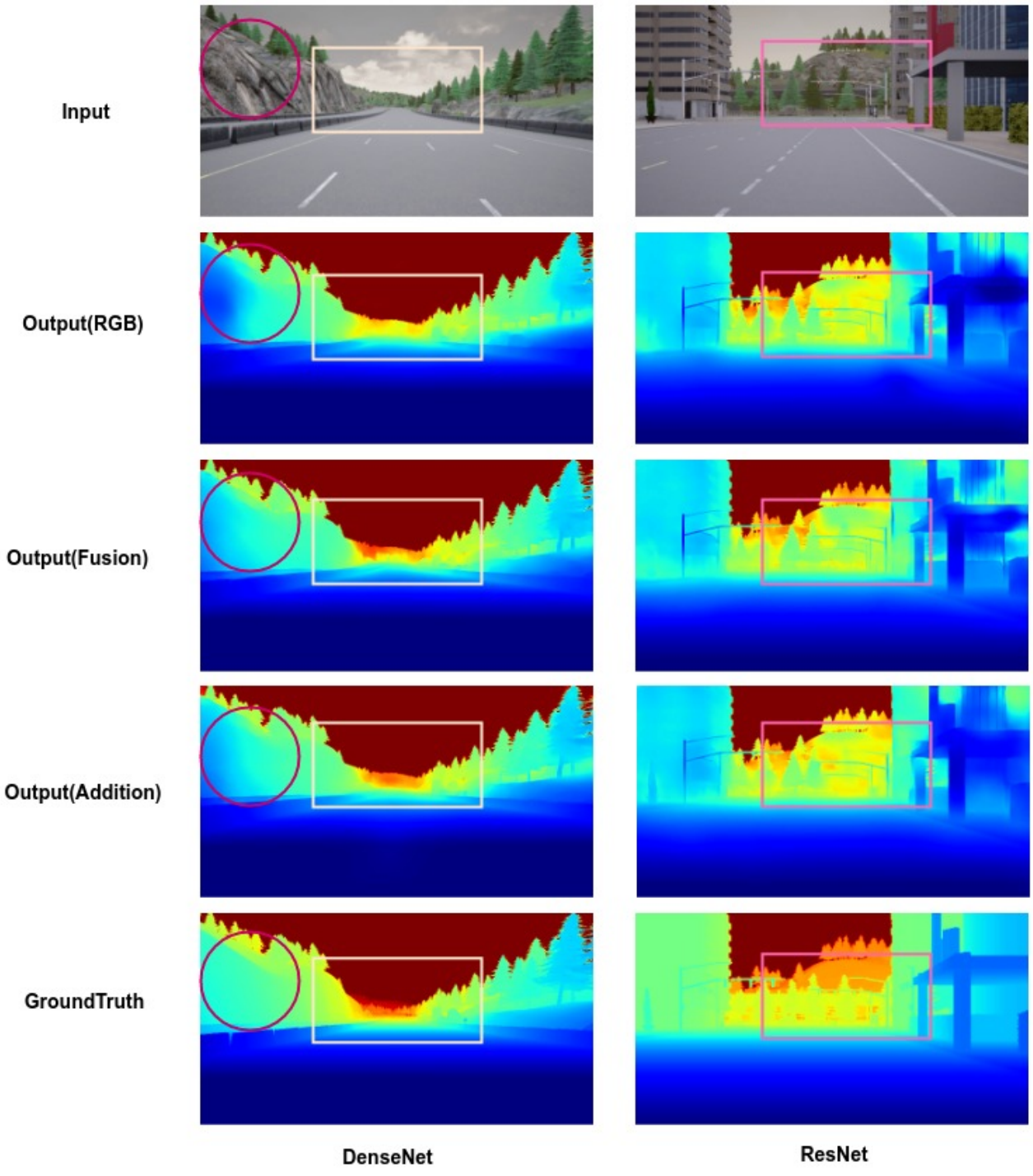


Figure 5.1: Qualitative results of different modalities trained with inverse depth using L1 loss. First column refers to predictions from the model trained on DenseNet. Second column refers to predictions from the model trained on ResNet. The images are denoted with boxes and circles wherever there seem to be variations in predictions among different modalities as compared to the groundtruth.

Training with the combination of Berhu+Image gradient+SSIM Loss

The fusion modality trained with Berhu loss using ResNet results in the decrement of absolute relative error by 0.8% as compared to the model trained on RGB images. The performance of ResNet trained model using Berhu loss is slightly better when compared to model trained with other loss. The reason may me that the Berhu loss takes advantage of L1 loss for very small errors and use L2 otherwise. The model does not perform better with DenseNet as compared to the model trained on L1 loss. Quantitative results are mentioned Table 5.3

Network	**Modality**	δ_1	δ_2	δ_3	**Abs Rel**	**Sq Rel**	**RMSE**
DenseNet-169	RGB	0.655	0.879	0.97	0.194	0.212	0.843
	RGB Fusion	0.661	0.892	0.97	0.190	0.216	0.862
	RGB Addition	0.634	0.887	0.97	0.200	0.217	0.847
ResNet-50	RGB	0.670	0.860	0.97	0.192	0.213	0.834
	RGB Fusion	0.711	0.883	0.97	0.184	0.213	0.839
	RGB Addition	0.663	0.862	0.97	0.196	0.222	0.852

Table 5.3: Comparison of different modalities trained with inverse depth using Berhu loss

Training with the combination of L2+Image gradient+SSIM Loss

The network trained on DenseNet with L2 loss has the best thereshold accuracy(δ_3) of 98% for all the modalities as compared to the network trained on L1 loss and Berhu Loss. The modality, with the element-wise addition of Radar and RGB performs best among all other modalities with a decrement of 0.9% in average relative error, 0.8% decrement in square relative error and 0.13 decrement in RMSE score as compared to the model trained only on RGB.
The addition modality trained with DenseNet performs better as compared to other modalities and could be supported by the quantitative results as shown in Fig 5.2 where the depth predictions are quite similar to the groundtruth. The fusion modality when trained with ResNet performs worst as compared to RGB and addition modality as there is a high error in prediction of depth for the mountain present on the scene that is denoted by the box in the image below. Quantitative results are mentioned in Table 5.4 and Qualitative results are shown in Fig 5.2

Network	Modality	δ_1	δ_2	δ_3	Abs Rel	Sq Rel	RMSE
DenseNet-169	RGB	0.643	0.912	0.980	0.191	0.202	0.844
	RGB Fusion	0.666	0.911	0.980	0.184	0.194	0.835
	RGB Addition	0.660	0.914	0.982	0.182	0.194	0.831
ResNet-50	RGB	0.674	0.877	0.977	0.188	0.205	0.835
	RGB Fusion	0.643	0.867	0.970	0.200	0.223	0.862
	RGB Addition	0.669	0.898	0.978	0.184	0.204	0.841

Table 5.4: Comparison of different modalities trained with inverse depth using L2 loss.The highlighted results shows the best performing model in terms of square relative error when trained with L1, Berhu and L2 loss along with image gradient and SSIM Loss.

5.1.3 Model trained with inverse depth using UpProj Network

Training with combination of L1+SSIM Loss

The network is trained using the combination of mentioned losses. Although it is also trained individually with L1, L2, and Berhu Loss, the best performing result is obtained using a combination of L1 and SSIM loss. The UpProj networks failed to accomplish the results similar to Dense Depth network. The prediction fails to reconstruct some of the objects present on the scene. For instance, the prediction of traffic light and buildings as denoted by box on the images are coarse and slightly incorrect. The main reason is that the decoder is not being supervised with the features extracted from the encoders via skip connections. For all the modalities trained on ResNet, the lowest RMSE value resulted from the UpProj network is 0.942 i.e 0.11 value higher than the best performing Dense Depth network. Quantitative results and Qualitative results are mentioned in Table 5.5 and Table 5.3 respectively.

Network	Modality	δ_1	δ_2	δ_3	Abs Rel	Sq Rel	RMSE
ResNet-50	RGB	0.664	0.870	0.967	0.196	0.242	0.961
	RGB Fusion	0.640	0.868	0.967	0.200	0.243	0.953
	RGB Addition	0.636	0.849	0.968	0.204	0.247	0.942

Table 5.5: Quantitative results of different modalities trained with ResNet using UpProj architecture.

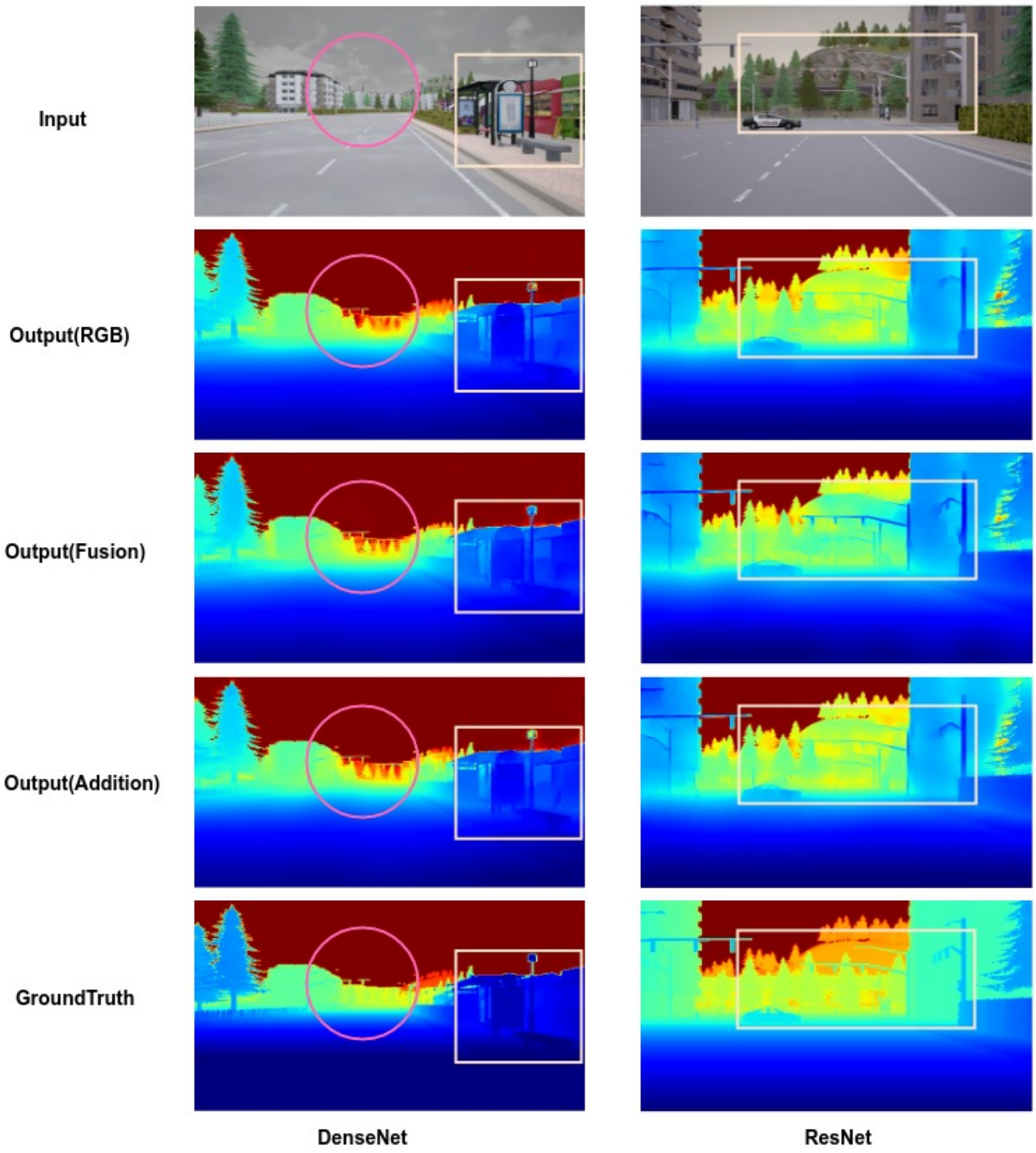


Figure 5.2: Qualitative results of different modalities trained with inverse depth using L2 loss. First column refers to predictions from the model trained on DenseNet. Second column refer to predictions from the model trained on ResNet. The images are denoted with boxes and circles wherever there seem to be variation in the predictions among different modalities.

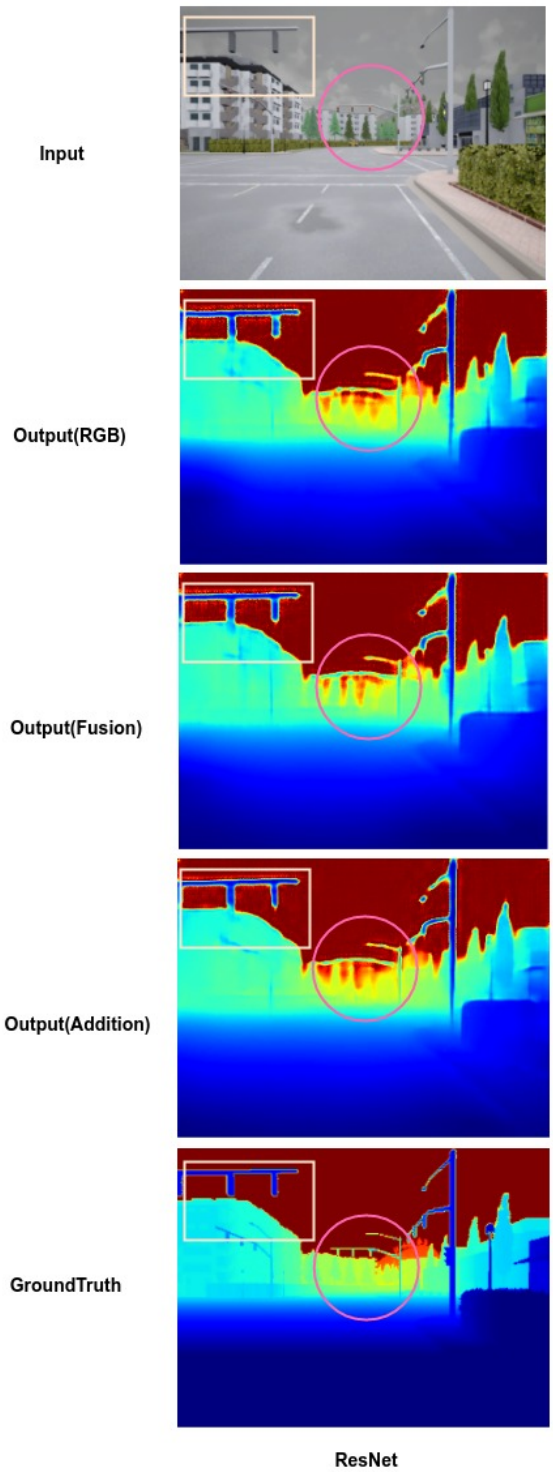


Figure 5.3: Qualitative results of different modalities trained with ResNet using UpProj architecture. The images are denoted with boxes and circles wherever there seem to be variation in predictions among different modalities.

5.2 Evaluation on Nuscenes Dataset

For the Nuscenes dataset, training is done on the DenseDepth network with DenseNet and ResNet as encoders, and UpProj network with only ResNet as encoder. The network is trained with inverse depth with a combination of different loss terms as used in synthetic dataset.

5.2.1 Model trained with inverse depth using Dense Depth

Training with the combination of L1+Image gradient+SSIM Loss

As groundtruth for Nuscenes is generated using Lidar point clouds and RGB images, the modality of RGB is much more important as compared to Radar point clouds. In comparison to all the modalities, the network trained with only RGB modality performed better. The metric results for Radar fusion and addition modality are slightly substandard as compared to a network trained only on RGB but on the visualization of predictions, the results are much more comprehensible as compared to predictions from a network trained only on RGB images and can be seen in Table 5.10. There may be some effects of Radar points while training the model, as the predictions from fusion and addition modality trained on DenseNet and ResNet are much smoother as compared to predictions from a model trained on RGB despite groundtruth being constructed from RGB and Lidar. There seem to be variations in the predictions from different modalities which is denoted with a rectangular box as shown in Fig 5.4

Quantitative results are mentioned in Table 5.6 and Qualitative results are shown in Fig 5.4

Network	**Modality**	δ_1	δ_2	δ_3	**Abs Rel**	**Sq Rel**	**RMSE**
DenseNet-169	RGB	0.903	0.985	0.996	**0.089**	**0.309**	**2.27**
	RGB Fusion	0.895	0.983	0.996	0.093	0.335	2.343
	RGB Addition	0.903	0.985	0.996	0.091	0.317	2.27
ResNet-50	RGB	0.886	0.978	0.994	0.098	0.372	2.41
	RGB Fusion	0.879	0.977	0.994	0.099	0.384	2.469
	RGB Addition	0.879	0.978	0.995	0.098	0.37	2.47

Table 5.6: Comparison of different modalities trained with inverse depth using L1 loss on Nuscenes dataset. The highlighted results shows the best performing model in terms of RMSE and absolute relative error, and square relative error when trained with L1, Berhu and L2 loss along with image gradient and SSIM Loss.

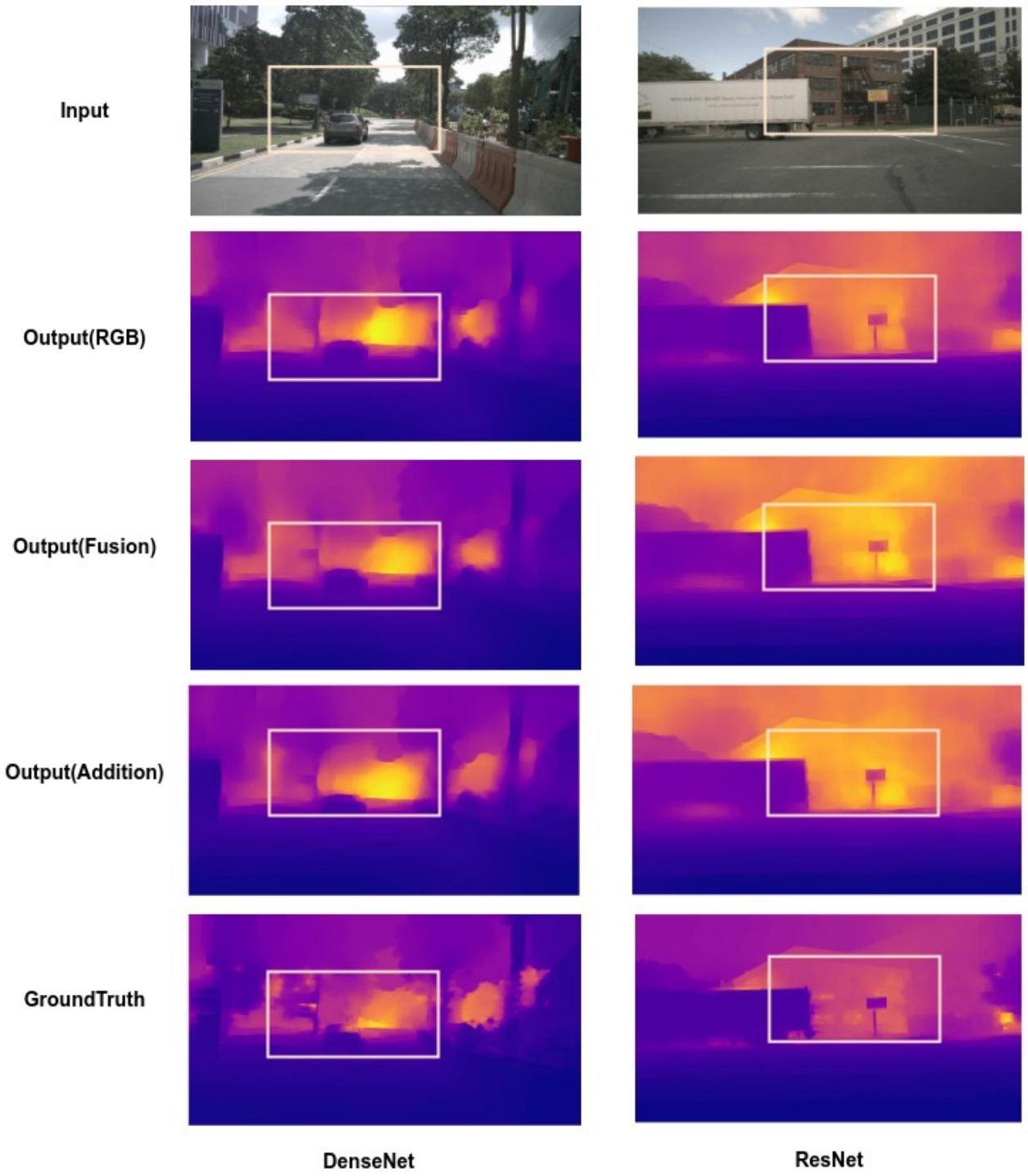


Figure 5.4: Qualitative results of different modalities trained with inverse depth using L1 loss. First column refers to predictions from the model trained on DenseNet. Second column refer to predictions from the model trained on ResNet. The images are marked with boxes wherever there seem to be variations in predictions among different modalities.

Training with the combination of L2+Image gradient+SSIM Loss

The network trained on all the modalities with L2 loss performed poorly on every metric used for calculation of prediction error as compared to networks trained on L1 loss. As the groundtruth is not perfect as compared to Synthetic dataset, the model gets penalizes for large errors when trained with L2 loss that seems to degrade the estimations. Quantitative results are mentioned in Table 5.7

Network	Modality	δ_1	δ_2	δ_3	Abs Rel	Sq Rel	RMSE
DenseNet-169	RGB	0.891	0.984	0.996	0.094	0.332	2.383
	RGB Fusion	0.891	0.982	0.996	0.095	0.344	2.41
	RGB Addition	0.890	0.982	0.996	0.094	0.340	2.42
ResNet-50	RGB	0.870	0.977	0.995	0.103	0.399	2.569
	RGB Fusion	0.867	0.967	0.995	0.103	0.400	2.583
	RGB Addition	0.872	0.977	0.995	0.103	0.397	2.550

Table 5.7: Comparison of different modalities trained with inverse depth using L2 loss on Nuscenes dataset.

Training with the combination of Berhu+Image gradient+SSIM Loss

The network trained with Berhu loss on DenseNet and ResNet fails to replicate the results of networks trained using L1 loss. Networks trained using Berhu Loss has poor results in all the metric except RMSE, in comparison to network trained with L2 loss. For small error difference between the predictions and the groundtruth, Berhu loss might have benefited from L1 but the estimations would have degraded because of L2 loss. The prediction from fusion and addition modality is more dense and smooth as compared to the prediction from the model trained only on RGB images. Quantitative results are mentioned in Table 5.8 and Qualitative results are shown in Fig 5.5

Network	Modality	δ_1	δ_2	δ_3	Abs Rel	Sq Rel	RMSE
DenseNet-169	RGB	0.895	0.981	0.995	0.093	0.339	2.356
	RGB Fusion	0.889	0.981	0.995	0.102	0.374	2.416
	RGB Addition	0.893	0.981	0.995	0.096	0.353	2.372
ResNet-50	RGB	0.874	0.976	0.994	0.104	0.416	2.52
	RGB Fusion	0.872	0.975	0.993	0.104	0.417	2.558
	RGB Addition	0.875	0.976	0.994	0.103	0.404	2.536

Table 5.8: Comparison of different modalities trained with inverse depth using Berhu loss on Nuscenes dataset.

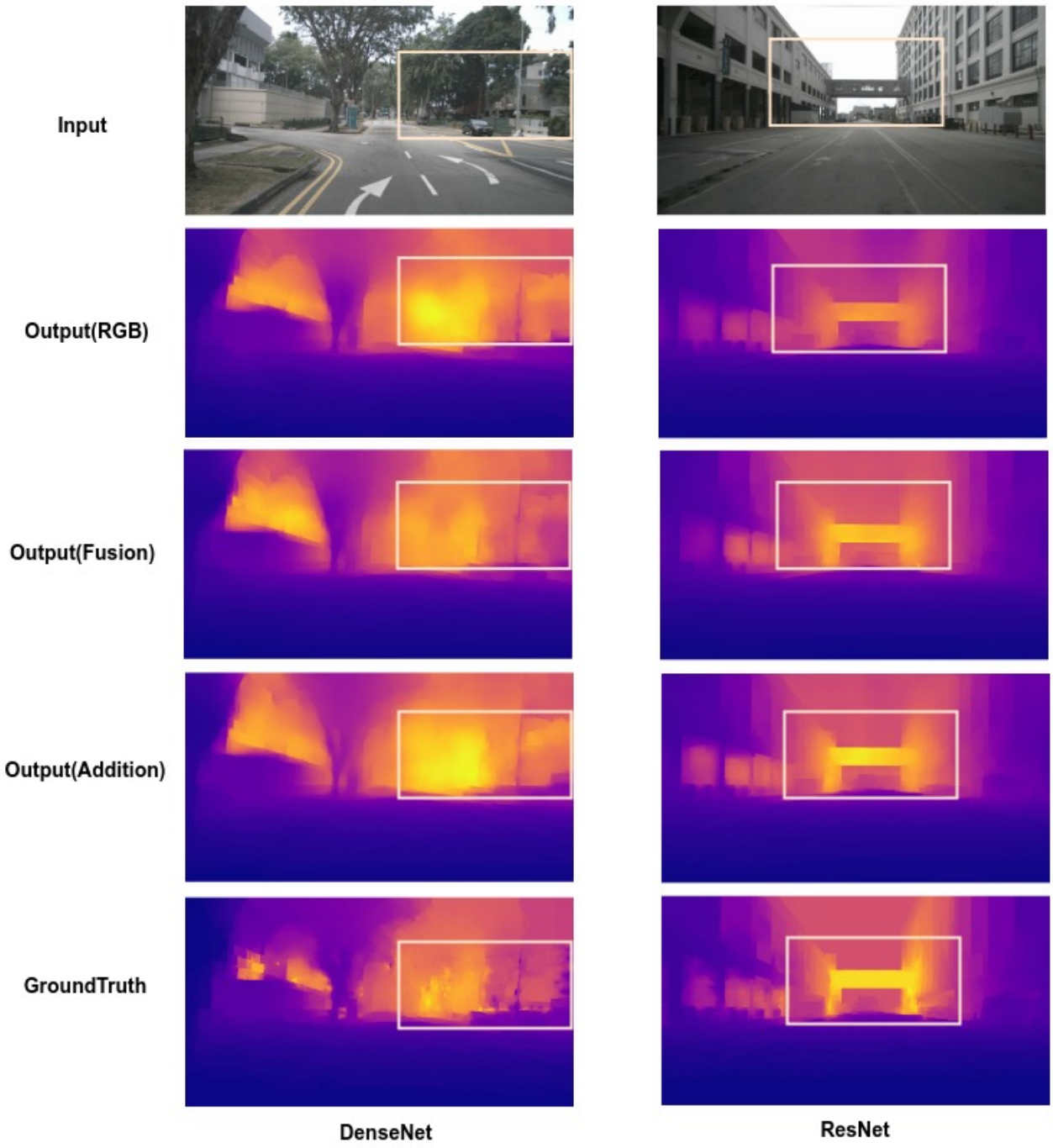


Figure 5.5: Qualitative results of different modalities trained with inverse depth using Berhu loss. First column refers to predictions from the model trained on DenseNet. Second column refer to predictions from the model trained on ResNet. The images are denoted with boxes wherever there seem to be variations in predictions among different modalities.

5.2.2 Model trained with inverse depth using UpProj Network

Training with the combination of L1+SSIM Loss

The network is trained using the combination of mentioned losses. The prediction error of the network trained only on RGB and the network trained on RGB addition modality is substantially the same. The quality of results is not at par with results compared to Dense Depth networks. The predictions from UpProj network are blurry in comparison of depth map predicted from Dense Depth networks. Quantitative results are mentioned in Table 5.9 and Qualitative results are shown in Fig 5.6

Network	**Modality**	δ_1	δ_2	δ_3	**Abs Rel**	**Sq Rel**	**RMSE**
ResNet-50	RGB	0.874	0.972	0.994	0.105	0.42	2.53
	RGB Fusion	0.871	0.967	0.993	0.106	0.434	2.56
	RGB Addition	0.874	0.972	0.994	0.105	0.42	2.53

Table 5.9: Comparison of different modalities trained with ResNet using UpProj architecture.

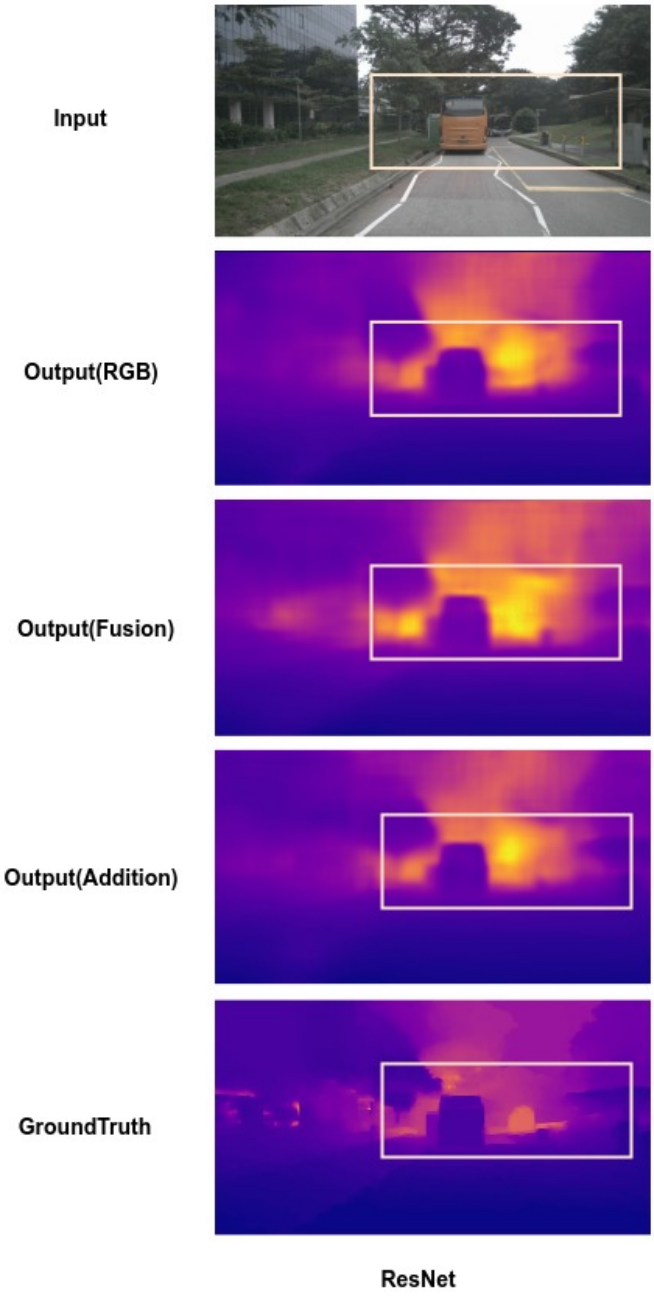


Figure 5.6: Qualitative results of different modalities trained with ResNet using UpProj architecture. The images are denoted with boxes wherever there seem to be variations in predictions among different modalities.

5.3 Inference Time

Table 5.10 contains the inference time of both the networks on all three different modalities. ResNet is faster as compared to DenseNet but it often has higher prediction error and lower threshold accuracy.

Architecture	Network	Modality	Inference Time (seconds)
Dense Depth	DenseNet	RGB	0.10
		RGB Fusion	0.11
		RGB Addition	0.10
	ResNet	RGB	0.076
		RGB Fusion	0.079
		RGB Addition	0.078
UpProj	ResNet	RGB	0.087
		RGB Fusion	0.089
		RGB Addition	0.088

Table 5.10: Comparison of different modalities across networks in terms of inference time

5.4 Summarization of Results

An exhaustive set of experiments is conducted on Dense Depth networks and UpProj networks using DenseNet and ResNet as encoders trained on different loss functions. From the results, it is clear that the Dense Depth network trained with inverse depth has a vast improvement in the predictions as compared to UpProj networks. For the Synthetic dataset, the fusion of RGB and Radar modality trained using DenseNet with a combination of L1, image gradient and SSIM loss surpasses every other model in terms of RMSE with a value of 0.820. The element-wise addition of Radar and RGB trained using DenseNet with a combination of L2, image gradient, and SSIM loss has the best threshold accuracy (δ_3) and lowest square relative error among all the networks. In the case of Synthetic dataset, the network trained with either L1 or L2 loss in combination with image gradient and SSIM using DenseNet has almost identical result. The networks trained with ResNet for all the modalities have moderate predictions as compared to DenseNet, but they have a better inference time which is quite beneficial when used in real-time applications.

For the Nuscenes dataset, as groundtruth is generated using Lidar point clouds with inpainting of RGB, the modality of RGB plays a vital role when model trained with concatenation and addition of Radar point clouds with RGB images. When the network trained only on RGB images, the predictions have lower RMSE value as compared to the model trained on other modalities. Overall, the performance of all the modalities trained using DenseNet with the combination of L1, image gradient and SSIM loss outperformed all other different combinations.

Inference on University dataset has been taken using the best performing model on the Synthetic dataset and Nuscenes dataset.

5.5 Inference on University Dataset

The best performing network from both the datasets has been used for inference on University dataset by combining Radar data and RGB images.

Inference using model trained on Nuscenes dataset

The model trained on the Nuscenes dataset fails to reconstruct the scene perfectly as compared to the model trained on the Synthetic dataset. The depth for the scene as shown in Fig 5.7 can be interpreted by the different colors of the plasma colormap that gives the idea about how far the objects are from the camera. As in the second image, the lower half of the building is shown with yellow color that signifies the area with maximum depth on the scene. The images are marked with boxes wherever the model can reconstruct the objects of the scene and estimate the depth correctly.

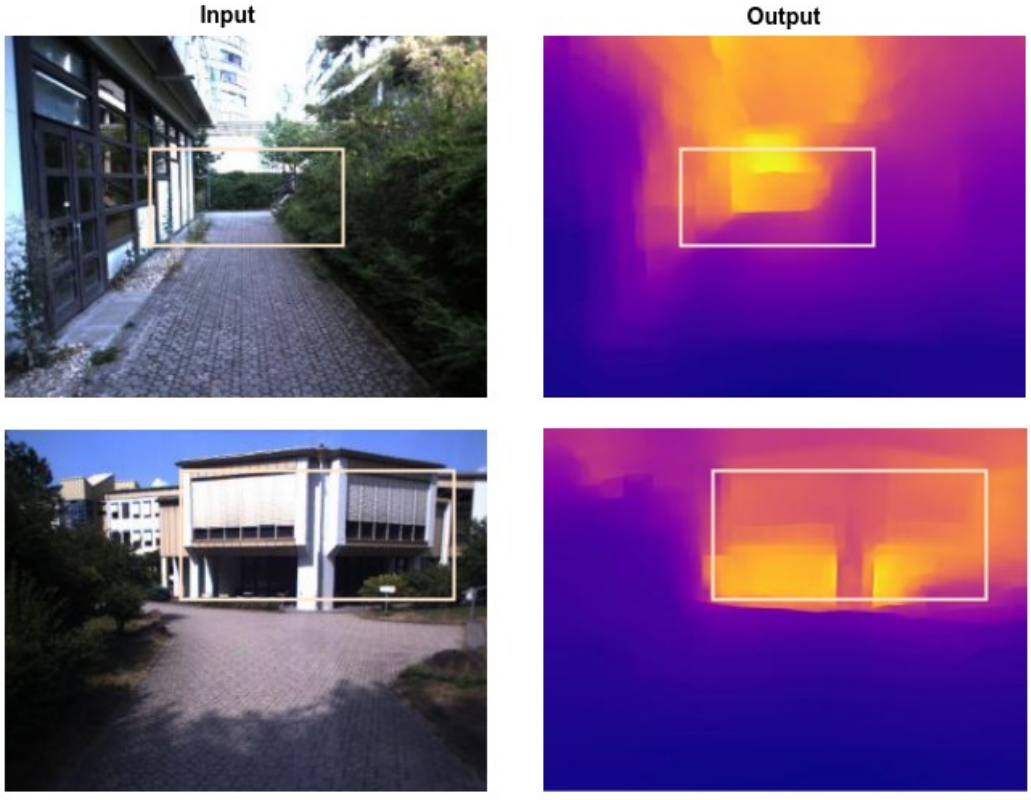


Figure 5.7: Inference on University dataset with the model trained on Nuscenes dataset. The images are denoted with boxes to represent the reconstruction of the objects on the scene and its depth from the camera.

Inference using model trained on Synthetic dataset

The model trained on the Synthetic dataset performs extremely well in recreating the objects on the scene for the University dataset despite being trained on a completely different dataset. The depth for the scene as shown in Fig 5.8 can be interpreted by the different colors of the jet colormap that gives the idea about how far the objects (such as buildings, persons, trees as represented by the boxes in the image) are from the camera.

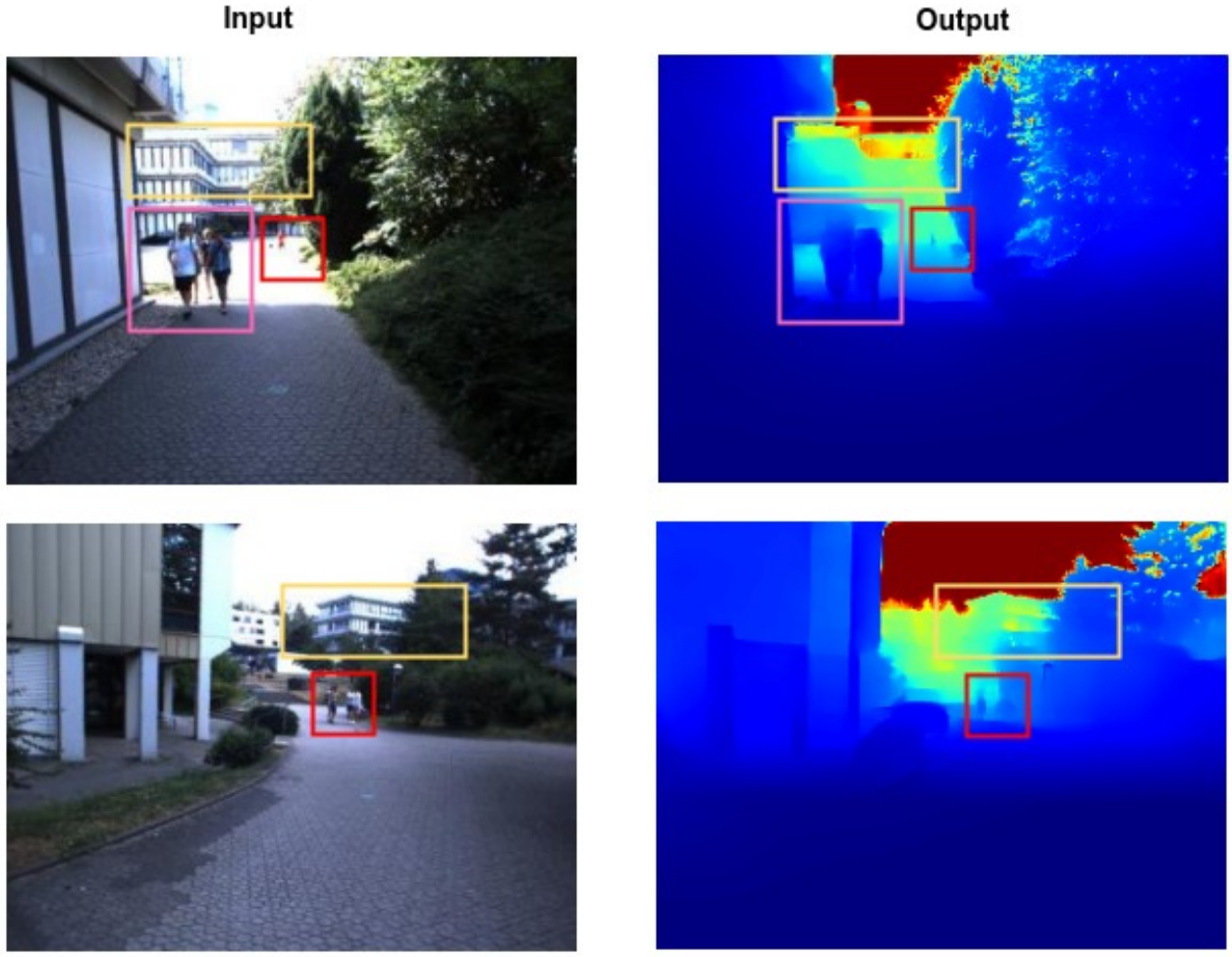


Figure 5.8: Inference on University dataset with the model trained on Synthetic data. The images are marked with boxes to represent the object on the scene and its depth from the camera.

Inference on Top-View image using model trained on Synthetic dataset

As the model is not trained on top view images, it's quite obvious that it will result in incorrect estimation of depth for the predictions. The model can recreate the objects on the scene perfectly, but the prediction of the distance of objects from the camera is estimated incorrectly at several places. The prediction is represented with the jet colormap so it should be represented with the color ranging from blue to red as the object moves away from the camera, but the predictions for the faraway objects are incorrect as shown in Fig 5.9.

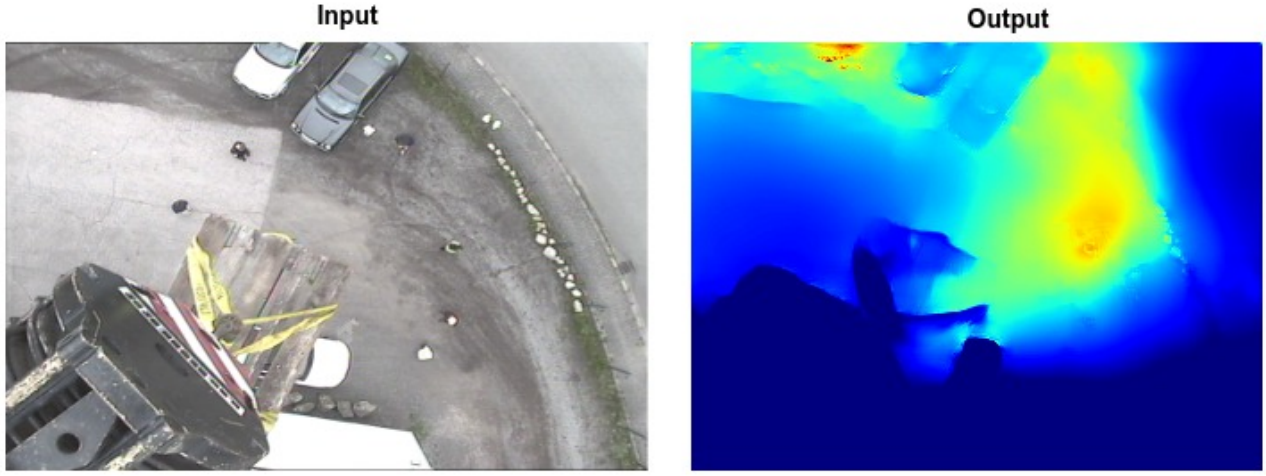


Figure 5.9: Inference on top view images with the model trained on Synthetic data.

5.6 Detection Failures

As the Synthetic dataset and Nuscenes dataset has not been trained on night scenes, the networks fail to reconstruct those scenes and estimate the depth for the same. In the case of reflection from the surfaces, the network sometimes assumes it to be the sky and predicts wrong depth measurement. Fig 5.10 shows scenes where network resulted in failed predictions.

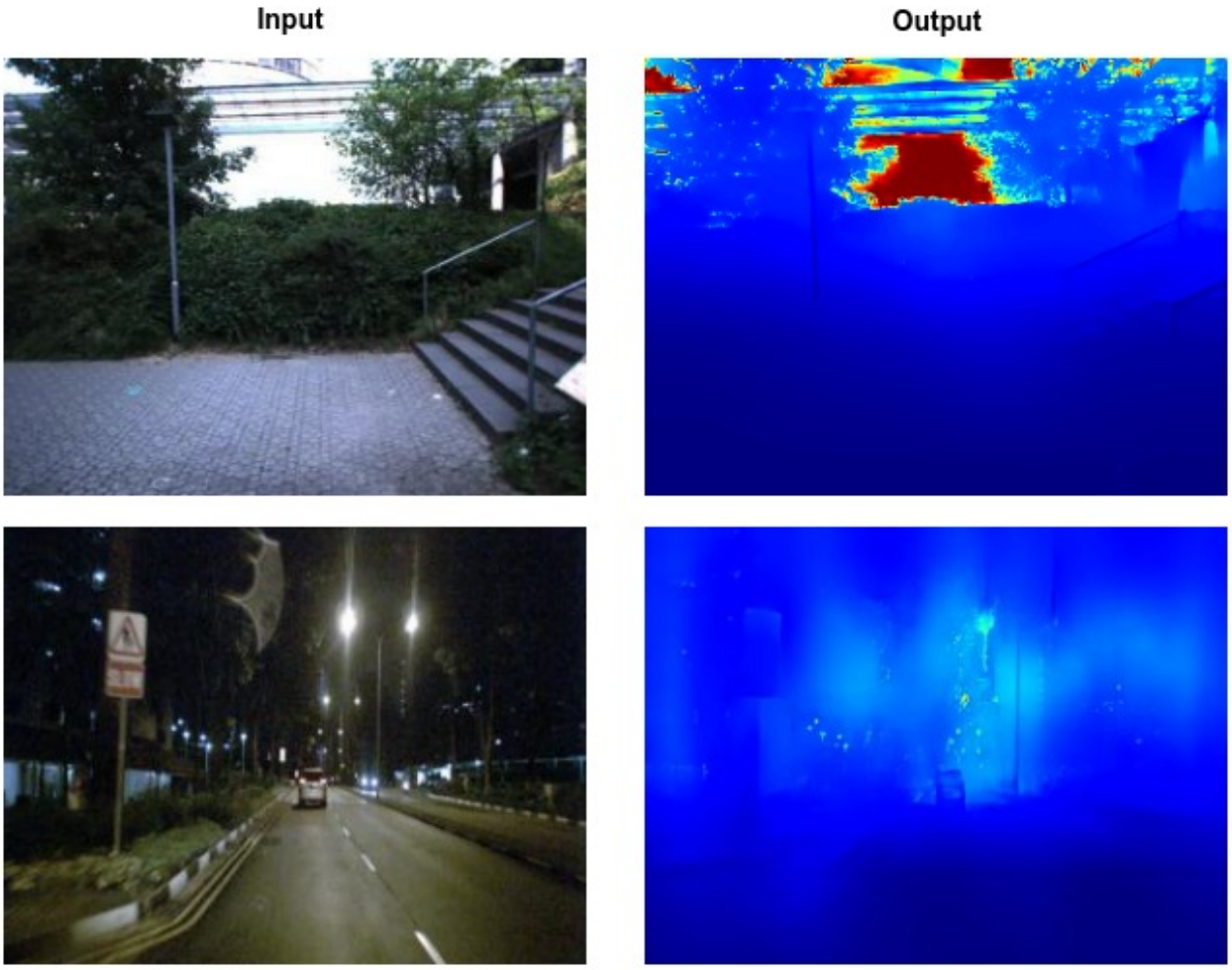


Figure 5.10: Detection failures from network trained on Synthetic dataset.

6. Ablation Studies

Ablation studies are performed to analyze the impact of Lidar point clouds on depth estimation when fused with RGB images. For this work, the network is trained on publicly available KITTI and Nuscenes datasets that provide laser and camera data. Dense Depth network with the same hyperparameters that are used to train the Radar fusion modality has been used. The network is trained with inverse depth using the combination of L1, Image gradient, and SSIM loss with DenseNet as an encoder.

6.1 Evaluation on KITTI dataset

The depth map groundtruth for the KITTI dataset is constructed from the accumulation of consecutive Lidar point clouds, and the colorization method is used to convert sparse groundtruth to the dense depth map. The network is expected to perform much better when trained with the fusion of Lidar and RGB than the network trained on only RGB images because the input modality effectively is a subset of the groundtruth. The fusion modality performed better in all metrics used for measuring prediction error. It reduces the absolute relative error and square relative error by 4.7% and 22% respectively. The RMSE is decreased by a value of 1.0. The network amplified the threshold accuracy (δ_1) by 8% as compared to the network trained only on RGB. The predictions are enhanced when the model is trained with the fusion of Lidar and RGB. For instance, the shop board denoted by box on the image as shown in Fig 6.1 is visible in the predictions obtained from model trained with Lidar and RGB fusion but it is not visible on the predictions obtained from model trained only on RGB.

Network	**Modality**	δ_1	δ_2	δ_3	**Abs Rel**	**Sq Rel**	**RMSE**
DenseNet-169	RGB	0.853	0.961	0.985	0.116	0.505	2.84
DenseNet-169	RGB Fusion	0.933	0.978	0.989	**0.079**	**0.280**	**1.84**

Table 6.1: Comparison of different modalities trained with DenseNet on Kitti dataset.

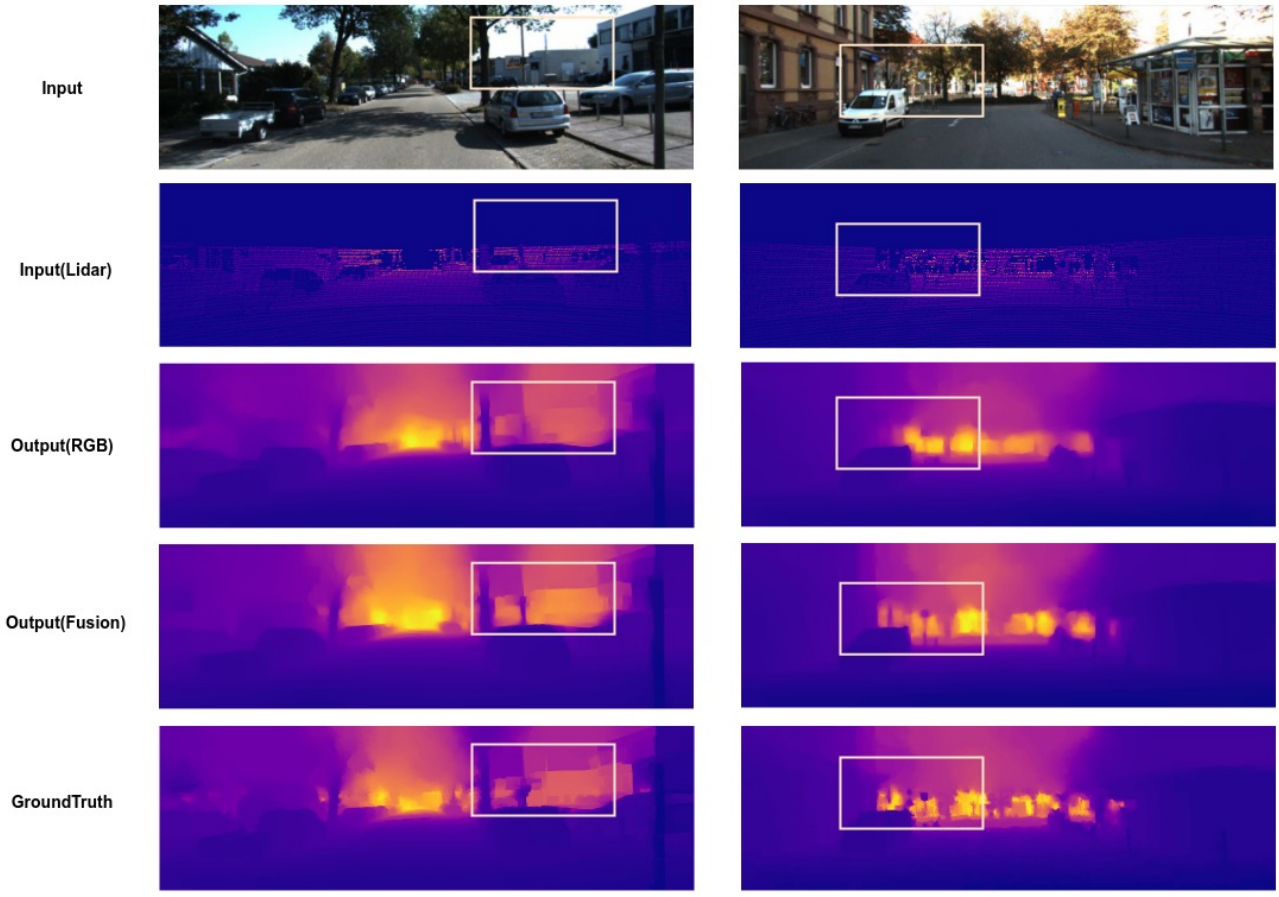


Figure 6.1: Qualitative results of different modalities trained with DenseNet on Kitti dataset. The images are denoted with boxes wherever there seem to be variations in predictions among different modalities.

6.2 Evaluation on Nuscenes Dataset

There is a 1% decrement in absolute relative error and RMSE is decreased by value of 0.25 when the network is trained using the fusion of Lidar point cloud and RGB images. There is slight enhancement in the prediction from the fusion model as compared to the model trained only on RGB. For example, the building behind the tree as denoted by box in the image as shown in Fig 6.2 is partially visible in prediction obtained from the model trained on fusion of Lidar and RGB. Quantitative results are mentioned in Table 6.2 and Qualitative results are shown in Table 6.2

Network	**Modality**	δ_1	δ_2	δ_3	**Abs Rel**	**Sq Rel**	**RMSE**
DenseNet-169	RGB	0.903	0.985	0.996	0.089	0.309	2.27
DenseNet-169	RGB Fusion	0.92	0.983	0.996	**0.079**	**0.30**	**2.02**

Table 6.2: Comparison of different modalities trained with DenseNet on Nuscenes dataset.

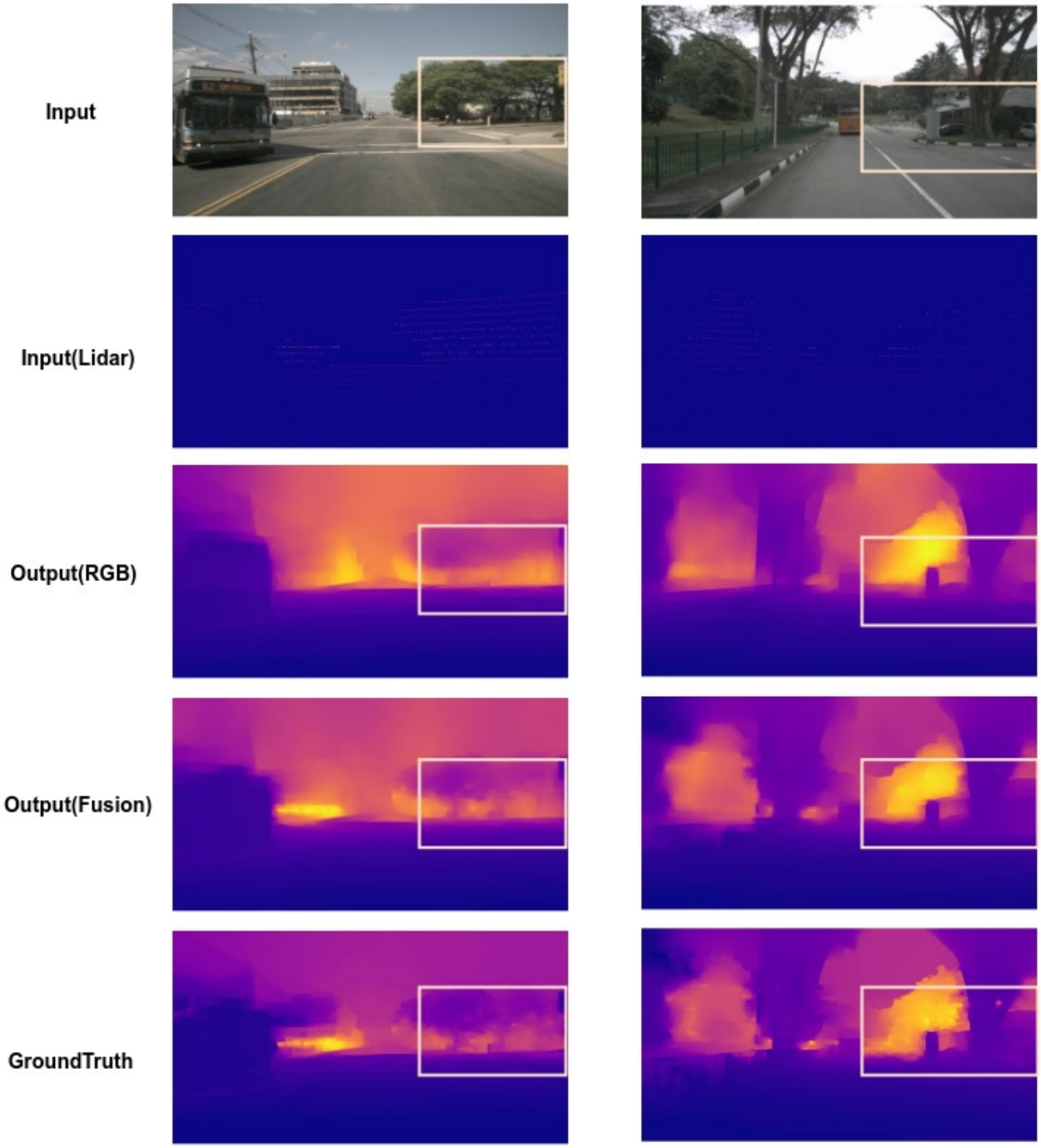


Figure 6.2: Qualitative results of different modalities trained with DenseNet on Nuscenes dataset. The images are denoted with boxes wherever there seem to be variations in predictions among different modalities.

7. Conclusions

In this chapter, the approaches and results are summarized. Some further future improvements and new open research areas related to the proposed work is suggested.

7.1 Summary and Discussions

In this work, two architectures namely Dense Depth and UpProj has been discussed. Both the networks consist of a universal encoder and differ in the implementation of the decoder. The decoder in Dense Depth networks consists of upsampling layers with supervision from the encoders using skip connections, whereas UpProj networks use upprojection residual blocks for upsampling. Dense Depth networks performed better than UpProj networks on all the metrics used for measuring prediction error for both the dataset. Several experiments were conducted to find the best performance model using a different combination of encoders and loss functions. For the synthetic dataset, the fusion modality trained using DesneNet with the combination of L1, image gradient, and SSIM loss is the best performing model in terms of RMSE. The Radar addition modality trained with DenseNet using the combination of L2, image gradient, and SSIM has the highest threshold accuracy (δ_2) and (δ_3) of 91.4% and 98% respectively. For the Nuscenes dataset, the network trained with only RGB modality has a lower absolute relative error and square relative error as compared to networks trained on the fusion of RGB and Radar data for all the loss functions. To evaluate the generalization capability of the network, the inference has been taken on the University dataset. The network is able to estimate the depth despite the images been captured with different cameras and focal lengths. The qualitative performance of the University dataset is previewed in Table 5.8 and it shows that network is capable of predicting depth for inputs on which it has not been trained. The impact of fusion of Lidar point clouds and RGB images is also investigated using the same network that is used to train the Radar fusion modality. For the KITTI dataset, the fused modality of Lidar and RGB images resulted in the decrement of square relative error by 22%.

7.2 Future Work

In this section, future improvement in the field of depth estimation is suggested. To explore the multimodal fusion approaches for depth estimation, datasets consisting of data from various sensors is an utmost necessity. There are only a few publicly available datasets that provide data corresponding to both the distance sensors such as Lidar and Radar, so there is a considerable demand for rich and large datasets to expand the boundaries of research in this field. As the creation of a groundtruth depth map is an expensive and tedious task, a self-supervised approach for multi-modal data could be explored. It has already been implemented for monocular images [Godard 18], a similar approach could prove beneficial for multi-modal data. For this work, the depth information from radar measurements is manifested in an image like representation. As Radar measurement comprises of range, radial velocity, and amplitude, there can be a future possibility to explore other components of Radar measurements for depth estimation task. A smaller network with fast response time is necessary for the deployment of Deep Learning models on embedded devices and robots. Various model optimization techniques by eliminating redundant and unnecessary training features to make the model light and efficient could be explored.

A. Appendix

LAYER	OUTPUT	FUNCTION
$INPUT_1$	672x1248x3	
$INPUT_2$	672x1248x1	
CONV	672x1248x3	CONV 1x1 on fused input
CONV1	336x624x64	DesneNet CONV1
POOL1	168x312x64	DenseNet POOL1
$POOL_2$	84x156x128	DenseNet POOL2
$POOL_3$	42x78x256	DenseNet POOL3
...	...	..
CONV2	21x39x1664	Conv 1X1 of DenseNet final block
UP1	42x78x1664	Upsample 2X2
CONCAT-1	42x78x1920	Concat
UP1-CONVA	42x78x832	Conv 3X3
UP1-CONVB	42x78x832	Conv 3X3
UP2	84x156x832	Upsample 2X2
CONCAT-2	84x156x960	Concat
UP2-CONVA	84x156x416	Conv 3X3
UP2-CONVB	84x156x416	Conv 3X3
UP3	168x312x416	Upsample 2X2
CONCAT-3	168x312x480	Concat
UP3-CONVA	168x312x208	Conv 3X3
UP3-CONVB	168x312x208	Conv 3X3
UP4	336x624x272	Upsample 2X2
CONCAT-4	336x624x104	Concat
UP4-CONVA	336x624x104	Conv 3X3
UP4-CONVB	336x624x104	Conv 3X3
CONV3	336x624x1	Conv 3X3

Table A.1: Dense Depth Architecture. Layers up to CONV2 are layers of DenseNet-169

LAYER	OUTPUT	FUNCTION
$INPUT_1$	672x1248x3	
$INPUT_2$	672x1248x1	
CONV	672x1248x3	CONV 1x1 on fused input
ResNet OUTPUT	21x39x2048	ResNet CONV1
...	...	..
CONV2	21x39x1024	Conv 1X1 of Resnet final block
UP1	42x78x512	Upsample 2X2
UP1-CONVA	42x78x256	Conv 5X5
UP1-CONVB	42x78x256	Conv 3X3
UP1_Bottom_CONV	42x78x256	Conv 3X3
ADD-1	42x78x256	Concat
UP2	84x156x256	Upsample 2X2
UP2-CONVA	84x156x128	Conv 5X5
UP2-CONVB	84x156x128	Conv 3X3
UP2_Bottom_CONV	84x156x128	Conv 3X3
ADD-2	84x156x128	Concat
UP3	168x312x128	Upsample 2X2
UP3-CONVA	168x312x64	Conv 5X5
UP3-CONVB	168x312x64	Conv 3X3
UP3_Bottom_CONV	168x312x64	Conv 3X3
ADD-3	168x312x64	Concat
UP4	336x624x64	Upsample 2X2
UP4-CONVA	336x624x32	Conv 5X5
UP4-CONVB	336x624x32	Conv 3X3
UP4_Bottom_CONV	336x624x32	Conv 3X3
ADD-4	336x624x32	Concat
CONV3	336x624x1	Conv 3X3

Table A.2: UpProj Architecture. Layers up to CONV2 are layers of ResNet-50

Bibliography

[A. L. Maas 13] A. Y. H. A L Maas, A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models. In Proc.", 2013.

[A. Saxena 05] S. H. C. A Saxena, A. Y. Ng, "Learning Depth from Single Monocular Images. In Proc.", 2005.

[Abadi 16] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. J. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Józefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. G. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. A. Tucker, V. Vanhoucke, V. Vasudevan, F. B. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems", *CoRR*, vol. abs/1603.04467, 2016.

[Alhashim 18] I. Alhashim, P. Wonka, "High Quality Monocular Depth Estimation via Transfer Learning", *arXiv e-prints*, vol. abs/1812.11941, 2018.

[Bacha 04] A. Bacha, C. Reinholtz, A. Wicks, M. Fleming, A. Naik, M. Avitabile, N. Elder, "The DARPA Grand Challenge: overview of the Virginia Tech vehicle and experience", in *Proceedings. The 7th International IEEE Conference on Intelligent Transportation Systems (IEEE Cat. No.04TH8749)*. Oct 2004, pp. 481–486.

[Caesar 19] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, O. Beijbom, "nuScenes: A multimodal dataset for autonomous driving", *arXiv preprint arXiv:1903.11027*, 2019.

[Chadwick 19] S. Chadwick, W. Maddern, P. Newman, "Distant Vehicle Detection Using Radar and Vision", *CoRR*, vol. abs/1901.10951, 2019.

[CRF 12] "Conditional Random Field", `https://en.wikipedia.org/wiki/Conditional_random_field`, 2012.

[Deng 09] J. Deng, R. Socher, L. Fei-Fei, W. Dong, K. Li, L.-J. Li, "ImageNet: A large-scale hierarchical image database", in *2009 IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, vol. 00. 06 2009, pp. 248–255.

[Depth-Estimation 18] "Depth Estimation.", `https://medium.com/beyondminds/depth-estimation-cad24b0099f`, 2018.

[Dosovitskiy 17] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, V. Koltun, "CARLA: An Open Urban Driving Simulator", in *Proceedings of the 1st Annual Conference on Robot Learning*. 2017, pp. 1–16.

[Educba 17] "Educba", `https://www.educba.com`, 2017.

[Eigen 14] D. Eigen, R. Fergus, "Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture", *CoRR*, vol. abs/1411.4734, 2014.

[Fischer 15] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, T. Brox, "FlowNet: Learning Optical Flow with Convolutional Networks", *CoRR*, vol. abs/1504.06852, 2015.

[Fu 18] H. Fu, M. Gong, C. Wang, K. Batmanghelich, D. Tao, "Deep Ordinal Regression Network for Monocular Depth Estimation", *CoRR*, vol. abs/1806.02446, 2018.

[Gansbeke 19] W. V. Gansbeke, D. Neven, B. D. Brabandere, L. V. Gool, "Sparse and noisy LiDAR completion with RGB guidance and uncertainty", *CoRR*, vol. abs/1902.05356, 2019.

[Garg 16] R. Garg, V. K. B. G, I. D. Reid, "Unsupervised CNN for Single View Depth Estimation: Geometry to the Rescue", *CoRR*, vol. abs/1603.04992, 2016.

[Geiger 12] A. Geiger, P. Lenz, R. Urtasun, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite", in *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012.

[Godard 17] C. Godard, O. Mac Aodha, G. J. Brostow, "Unsupervised Monocular Depth Estimation with Left-Right Consistency", in *CVPR*. 2017.

[Godard 18] C. Godard, O. Mac Aodha, G. J. Brostow, "Digging Into Self-Supervised Monocular Depth Estimation", *CoRR*, vol. abs/1806.01260, 2018.

[He 15] K. He, X. Zhang, S. Ren, J. Sun, "Deep Residual Learning for Image Recognition", *CoRR*, vol. abs/1512.03385, 2015.

[Huang 16] G. Huang, Z. Liu, K. Q. Weinberger, "Densely Connected Convolutional Networks", *CoRR*, vol. abs/1608.06993, 2016.

[Ioffe 15] S. Ioffe, C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.", in *ICML*, ser. JMLR Workshop and Conference Proceedings, F. R. Bach, D. M. Blei, Eds., vol. 37. JMLR.org, 2015, pp. 448–456.

[Jaritz 18] M. Jaritz, R. de Charette, É. Wirbel, X. Perrotton, F. Nashashibi, "Sparse and Dense Data with CNNs: Depth Completion and Semantic Segmentation", *CoRR*, vol. abs/1808.00769, 2018.

[Kendall 15] A. Kendall, M. Grimes, R. Cipolla, "Convolutional networks for real-time 6-DOF camera relocalization", *CoRR*, vol. abs/1505.07427, 2015.

[Kingma 14] D. P. Kingma, J. Ba, "Adam: A Method for Stochastic Optimization", *CoRR*, vol. abs/1412.6980, 2014.

[Krizhevsky 12] A. Krizhevsky, I. Sutskever, G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, K. Q. Weinberger, Eds., Curran Associates, Inc., 2012, pp. 1097–1105.

[Laina 16] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, N. Navab, "Deeper Depth Prediction with Fully Convolutional Residual Networks", *CoRR*, vol. abs/1606.00373, 2016.

[Levin 04] A. Levin, D. Lischinski, Y. Weiss, "Colorization using optimization.", *ACM Trans. Graph.*, vol. 23, no. 3, pp. 689–694, 2004.

[Ma 18a] F. Ma, G. V. Cavalheiro, S. Karaman, "Self-supervised Sparse-to-Dense: Self-supervised Depth Completion from LiDAR and Monocular Camera", *arXiv preprint arXiv:1807.00275*, 2018.

[Ma 18b] F. Ma, S. Karaman, "Sparse-to-Dense: Depth Prediction from Sparse Depth Samples and a Single Image", 2018.

[Maddern 17] W. Maddern, G. Pascoe, C. Linegar, P. Newman, "1 Year, 1000km: The Oxford RobotCar Dataset", *The International Journal of Robotics Research (IJRR)*, vol. 36, no. 1, pp. 3–15, 2017.

[Microsoft Kinect 12] "Kinect for Windows: How Gaming Tech Is Migrating to Business", `https://www.wired.com/2012/02/microsoft-kinect-for-windows//`, 2012.

[MRF 12] "Markov Random Field", `https://ermongroup.github.io/cs228-notes/representation/undirected/`, 2012.

[N. Mayer 16] P. H. P. F. D. C. A. D.-`[U+FFFD]` s. N Mayer, E Ilg, T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation", *CVPR*, 2016.

[Nair 10] V. Nair, G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines", in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, J. Fürnkranz, T. Joachims, Eds. 2010, pp. 807–814.

[Nathan Silberman 12] P. K. Nathan Silberman, Derek Hoiem, R. Fergus, "Indoor Segmentation and Support Inference from RGBD Images", in *ECCV*. 2012.

[openCV 12] "Open CV tutorials", `https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_calib3d/py_depthmap/py_depthmap.html/`, 2012.

[Qi 16] C. R. Qi, H. Su, K. Mo, L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation", 2016. cite arxiv:1612.00593.

[Qi 17] C. R. Qi, L. Yi, H. Su, L. J. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space", *CoRR*, vol. abs/1706.02413, 2017.

[ReLu 18] "Activation Function", https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6, 2018.

[Rezende 17] E. Rezende, G. Ruppert, A. Theophilo, T. Carvalho, "Exposing Computer Generated Images by Using Deep Convolutional Neural Networks", *Signal Processing: Image Communication*, 11 2017.

[Ridgeway 15] K. Ridgeway, J. Snell, B. Roads, R. S. Zemel, M. C. Mozer, "Learning to generate images with perceptual similarity metrics", *CoRR*, vol. abs/1511.06409, 2015.

[Saxena 07] A. Saxena, J. Schulte, A. Y. Ng, "Depth Estimation Using Monocular and Stereo Cues", in *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, ser. IJCAI'07. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007, p. 2197–2203.

[Sik-Ho Tsang 18] "Review: DenseNet — Dense Convolutional Network (Image Classification", https://towardsdatascience.com/review-densenet-image-classification-b6631a8ef803, 2018.

[Simonyan 14] K. Simonyan, A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", *CoRR*, vol. abs/1409.1556, 2014.

[SSIM 04] "Image quality assessment: from error visibility to structural similarity", https://ece.uwaterloo.ca/~z70wang/publications/ssim.html, 2004.

[Sun 19] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, D. Anguelov, "Scalability in Perception for Autonomous Driving: Waymo Open Dataset", 2019.

[Uhrig 17] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, A. Geiger, "Sparsity Invariant CNNs", in *International Conference on 3D Vision (3DV)*. 2017.

[Xu 18] K. Xu, S. Guo, N. Cao, D. Gotz, A. Xu, H. Qu, Z. Yao, Y. Chen, "ECGLens: Interactive Visual Exploration of Large Scale ECG Data for Arrhythmia Detection", in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, ser. CHI '18. New York, NY, USA: ACM, 2018, pp. 663:1–663:12.

[Zhou 17a] T. Zhou, M. Brown, N. Snavely, D. G. Lowe, "Unsupervised Learning of Depth and Ego-Motion from Video", in *CVPR*. 2017.

[Zhou 17b] Y. Zhou, O. Tuzel, "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection", *CoRR*, vol. abs/1711.06396, 2017.

[Zoph 17] B. Zoph, V. Vasudevan, J. Shlens, Q. V. Le, "Learning Transferable Architectures for Scalable Image Recognition", *CoRR*, vol. abs/1707.07012, 2017.

[Zwald 12] L. Zwald, S. Lambert-Lacroix, "The BerHu penalty and the grouped effect". 2012.

.