Faculty of Electrical Engineering and Information Technology

INTERNATIONAL AUTOMOTIVE ENGINEERING

Master Thesis

in collaboration with Infineon Technologies

RADAR–Monocular Camera Multiscale Fusion for Metric Depth Estimation and Object Detection

Author: **Vishal Vishnu Kagade**

Matriculation Number: **00124916**

Registered on: 25.11.2024

Submitted on: 23.05.2025

First Examiner:
Prof. Dr. Alessandro Zimmer

Company Advisor:
Mr. Vasu Tammisetti

Second Examiner:
Prof. Dr. rer. nat. Armin Arnold

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, that I have not presented it elsewhere for examination purposes, and that I have explicitly indicated all material which has been quoted either literally or by consent from the sources used. I have marked verbatim and indirect quotations as such.

Ingolstadt, 23.05.2025

Vishal Kagade

Acknowledgments

First and foremost, I would like to extend my heartfelt gratitude to my primary supervisor, **Prof.Dr.Alessandro Zimmer**, for his mentorship, patience, and invaluable guidance throughout my research journey. I especially thank Prof. Zimmer for continuously inspiring and pushing me to achieve my best.

A special thanks goes to my company supervisor, **Mr.Vasu Tammisetti**, for providing me with an incredible opportunity to work on cutting-edge research. His continuous support, insightful advice, and extensive expertise in deep learning significantly contributed to the successful completion of this thesis.

I would also like to express my deepest appreciation to **M.Eng.Joed Lopes da Silva**, who always went out of his way to support me. His creative ideas, countless brainstorming sessions, and generous nature gave me the confidence to accomplish this work successfully. Additionally, I sincerely thank **Mr.Yamen Mohsin** for sharing valuable insights and providing essential resources that greatly benefited this research.

Finally, I am profoundly grateful to my brother, **Mr.Santosh Kagade**, my family, and friends for their unwavering support, encouragement, and belief in me throughout my academic and professional journey. Their love has shaped my journey into one that is truly rewarding and fulfilling.

Vishal Kagade
Ingolstadt, Germany
May 2025

Abstract

Autonomous driving faces numerous challenges due to the dynamic and unpredictable nature of real-world environments. Accurate metric depth estimation is a critical requirement for autonomous driving and computer vision applications. While LiDAR sensors achieve high accuracy, they face limitations due to cost and integration complexity. This thesis explores a multitask, multimodal fusion of 4D RADAR and monocular RGB images, supported by monocular relative depth maps, to achieve precise metric depth estimation and object detection.

A key limitation in existing RADAR-CAMERA fusion methods is their inability to accurately assign RADAR depths to image pixels, resulting in errors in depth estimation. To address this, we propose a novel approach that introduces a depth-pixel association metric, improving the assignment of RADAR depths to corresponding image pixels. A transformer-based model is employed to enhance cross-modality association, leading to significant improvements in depth estimation accuracy. The proposed architecture utilizes a single backbone network with two task-specific heads: one for depth estimation and the other for object detection, demonstrating the multitasking capability of the framework.

The framework is evaluated on the ZDU-4D RADAR dataset using standard metrics, including MAE and RMSE for depth estimation. Experimental results show that our approach outperforms state-of-the-art RADAR-CAMERA depth estimation methods, achieving a 11.8% improvement in accuracy. Additionally, the integrated object detection head enhances scene understanding by accurately localizing relevant objects, improving perception robustness under diverse environmental conditions.This work effectively addresses the limitations of existing methods, advancing both metric depth estimation and object detection performance for autonomous driving applications.

Master Thesis GitHub Repository

Nomenclature

AI	Artificial Intelligence
D	Dimensions
DL	Deep Learning
LiDAR	Light Detection and Ranging
ML	Machine Learning
RADAR	Radio Detection and Ranging
SOTA	State of the Art

Glossary

RGB Red, Green, and Blue.

Acronyms

3D Three Dimension. 1, 8, 14, 16, 29,

34 Four Dimension. 16,

AI Artificial Intelligence. 4,

BCE Binary Cross Entropy.

BDD100K Berkeley Driving Dataset. 17, 46, 57,

CBS Convolution, Batch normalization and ReLU. 45,

CNN Convolutional Neural Network. 11, 22, 23,

D Dimensions.

DL Deep Learning. 4,

ELU Exponential Linear Unit. 6, 7,

FC Fully Connected (layer).

FN False Negative. 12,

FoV Field of View. 15,

FP False Positive. 12,

FPS Frames Per Second.

IoU Intersection over Union. 12, 13, 41,

LiDAR Light Detection and Ranging. 1, 2, 14, 15, 17, 22, 30, 32, 38, 46, 52–54, 58, III, , IX

LoFTR Local Feature Transformer. 34, 35, 44, 45, 57, 58,

MAE Mean Absolute Error. 18, 50–53, 57, III,

mAP Mean Average precision. 12, 13,

ML Machine Learning. 4,

MLP Multilayer Perceptron.

MSE Mean Squared Error.

MTL multi-task learning. 1,

NMS Non-Maximum Suppression.

RADAR Radio Detection and Ranging. 15, 16, 30–34, 44, 53, 55, 57, III, , VII, IX

ReLU Rectified Linear Unit. 6, 7,

ResNet Residual Network. 21–23, 33, 44, 48,

RGB Red, Green, Blue (color channels). 8, 13, 17, 32–34, 44, 52, 57, III,

RMSE Root Mean Square Error. 50, 51, III,

ROI Region of Interest. 32,

RPN Region Proposal Network.

SCN sparse convolution network. 33, 44,

SfM Structure from Motion.

SGD Stochastic Gradient Descent.

SOTA State of the Art. 32, 44, 50,

SOTA State of the art. 2, 3, 17, 50,

SSIM Structural Similarity Index Measure.

TN True Negative. 12,

TP True Positive. 12,

ViT Vision Transformer. 24, 25, 57,

YOLOv8 You Only Look Once version 8. 36, 45,

List of figures

List of tables

Table of contents

1 Introduction

Accurate perception of the environment is a fundamental requirement for autonomous driving systems, as it enables critical tasks such as dense 3D scene reconstruction, object detection, and obstacle avoidance. One of the most vital part of perception pipeline is depth estimation, which provides essential spatial understanding of the surrounding environment. Traditionally, LiDAR sensors have been the primary choice for achieving metric depth estimation due to their ability to produce highly accurate 3D data. However, despite their advantages, LiDAR systems face significant challenges such as high costs, power consumption, and extensive data bandwidth requirements. These factors limit their feasibility for large-scale deployment, particularly in cost-sensitive or resource-constrained applications.

To address these limitations, monocular depth estimation using cameras has emerged as a promising alternative. Monocular cameras are low-cost, lightweight, and widely available, making them a practical choice for autonomous driving. While monocular depth estimation performs well in predicting relative depth within a scene, it suffers from scale ambiguity, which prevents accurate estimation of metric depth [4]. This limitation poses challenges for applications that require precise distance measurements, such as navigation, obstacle detection, and collision avoidance.

Recently, camera-radar fusion has been explored to overcome the scale ambiguity of monocular depth estimation. Radar sensors, unlike cameras, are inherently robust to different weather conditions like fog, rain, and low light, ensuring reliable depth supervision under challenging environments [22]. Additionally, the development of Three Dimension (3D) radar and emerging 3D radar technology has further enhanced radar's capabilities by introducing an elevation dimension, offering more comprehensive spatial information. However, radar-based depth estimation also faces significant challenges. Radar data tends to be sparse and often contains substantial noise, making it difficult to achieve dense and accurate depth predictions. Furthermore, aligning radar points with image pixels in a cross-modal fusion setting can be imperfect, leading to inconsistencies in the fused data [37].

In addition to improving depth prediction accuracy, recent research has demonstrated that integrating auxiliary tasks—such as semantic segmentation, surface normals estimation, and object detection—can significantly enhance depth estimation performance [34]. These tasks provide complementary scene context, enabling a deeper understanding of the environment. For example, semantic segmentation helps distinguish between different scene components (e.g., roads, vehicles, pedestrians), while surface normals add geometric insights by capturing the orientation of surfaces [34]. Similarly, object detection directly complements depth estimation by localizing and identifying objects within the scene, associating accurate depth information with detected objects, and facilitating better spatial understanding. This synergy allows the perception system to refine depth maps and improve object localization simultaneously.

These tasks are brought together using a multi-task learning (MTL) framework, which has become increasingly popular in past years due to its functionality to improve both efficiency and robustness. MTL leverages shared feature representations between related tasks, enabling the model to understand robust and generalized features while reducing computational costs through shared computations [9]. Furthermore, MTL acts as a form of task-specific regularization, mitigating overfitting by allowing tasks to guide and complement one another during the training process [9]. For instance, in the context of depth estimation and object detection, accurate object boundaries identified through the detection task can refine depth maps, while depth information aids in improving object localization and size estimation.

This thesis explores camera-radar fusion in conjunction with a multi-task learning framework, where depth estimation serves as the primary task and object detection as an auxiliary task. By leveraging the complementary strengths of radar sensors and monocular cameras and integrating multi-task learning, the proposed approach aims to achieve accurate, efficient, and context-aware depth estimation for autonomous driving applications. The combination of these techniques addresses the limitations of individual modalities and provides a robust solution for challenging real-world perception tasks, ultimately contributing to the advancement of autonomous driving systems.

1.1 Motivation

Accurate spatial understanding of the environment, through classification and localization, is essential for autonomous perception. Among the key challenges, metric depth estimation and object localization play pivotal roles in achieving reliable scene understanding. Traditional dense LiDAR sensors provide high-accuracy depth information but are constrained by their high cost, energy consumption, and limited availability. To overcome these problems, camera-radar fusion has came as a promising alternative, leveraging the strengths of both sensors.

Radar sensors are cost-effective and robust to adverse environmental conditions, such as poor lighting and weather variations. However, radar data suffer from sparsity and lack precise elevation information, which introduces errors in depth estimation. State of the art (SOTA) models have explored various methods to overcome these limitations. For example:

1. Extending radar points through processing pipelines.
2. Two-stage approaches, where the first stage associates depth to image pixels, and the following stage trains the model to refine the metric depth estimation.

While two-stage approaches have shown significant improvements in depth estimation accuracy, they suffer from high latency, making them inefficient for real-time applications. To address this issue, a one-stage approach with monocular relative depth supervision in radar-camera fusion offers a promising solution. Monocular depth estimation provides contextual depth information, while radar provides robust and reliable depth measurements, leading to improved overall depth estimation.

During my work on the AITHENA project at Infineon,[1] one critical task was obtaining the yaw angle of objects using a monocular camera. . While we successfully achieved this using LiDAR data, we encountered significant challenges with the monocular camera due to its lack of depth perception. To overcome this limitation, a depth estimation network can be coupled with an object detection module in a multitasking model. This integrated approach provides both localization and depth estimation, facilitating yaw angle determination and improving the robustness of scene understanding.

This multitasking framework not only addresses the immediate challenge of yaw angle estimation but also opens up possibilities for 3D reconstruction, a vital component of the autonomous perception pipeline. By integrating depth estimation and object localization into a single, efficient model, this approach enhances real-time perception capabilities and paves the way for more advanced applications in autonomous driving.

[1]Funded by the European Union's Horizon Europe Research and Innovation Program through Grant Agreement No. 101076754 (AITHENA project).

1.2 Objectives

In this work, we aim to achieve the following objectives:

1. To propose a novel transformer-based association network that fuses radar and camera data using monocular depth supervision for dense metric depth estimation.
2. To evaluate the metric depth estimation model using SOTA evaluation metrics on the ZDU-4D RADAR dataset.
3. To extend the depth estimation network by incorporating an object detection head, enabling a multitasking algorithm for simultaneous depth estimation and object detection.

Throughout this report, we assume a basic understanding of deep learning-based depth estimation and object detection methods.

Artificial Intelligence (AI) enables machines to simulate human intelligence, particularly in tasks like autonomous decision-making and computer vision. Within AI,Machine Learning (ML) focuses on developing algorithms that allow systems to learn and adapt from data. ML covers a range of techniques, including supervised learning that relies on labeled data, unsupervised learning that uncovers hidden patterns in data without labels.

A key subset of ML, Deep Learning (DL), leverages multi-layered neural networks to analyze large, complex datasets. DL plays a critical role in enabling AI applications like object detection, image recognition, and scene understanding, which are essential for autonomous systems and computer vision tasks. These technologies drive innovation in areas like self-driving cars, robotics, and advanced visual perception systems.

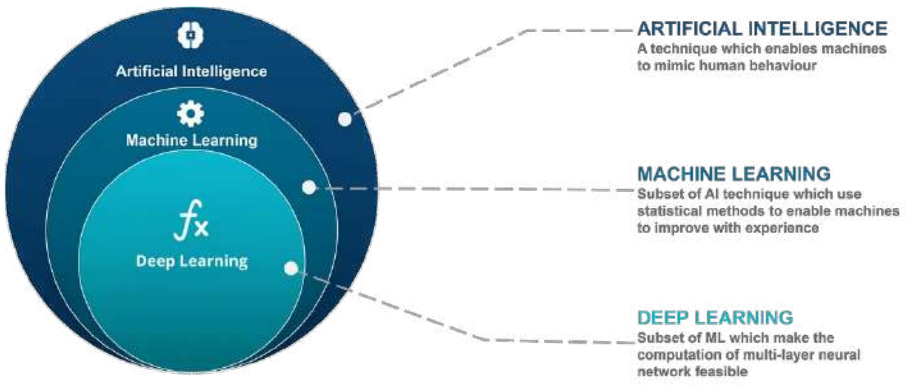

Artificial Intelligence
Machine Learning
f_x
Deep Learning
ARTIFICIAL INTELLIGENCE
A technique which enables machines to mimic human behaviour
MACHINE LEARNING
Subset of AI technique which use statistical methods to enable machines to improve with experience
DEEP LEARNING
Subset of ML which make the computation of multi-layer neural network feasible

Figure 1: Definition of Artificial Intelligence (AI) systems [11]

1.3 Deep Learning

The term "deep" refers to the use of multiple layers (hidden layers) in neural networks, enabling them to learn hierarchical and abstract representations. With the availability of large-scale data and computational power, Deep learning is a key pillar of modern AI.

1.3.1 Neural Networks: The Foundation of Deep Learning

Artificial Neural Networks (ANNs), which form the core of DL, are inspired by how the human brain processes information. They typically comprises of three different types of layers:an input layer that takes in raw data like pixel values or numerical features, one or more hidden layers where the network learns by adjusting connections between neurons, and an output layer that delivers the final prediction or classification.

In a neural network, each neuron processes its inputs by applying weights and then uses an activation function to introduce non-linearity. This non-linear transformation helps the network understand and learn more complex patterns and connections within the data. As more layers are added, the network can extract increasingly abstract features [30], making it effective for tasks such as Object detection, natural language processing, and autonomous systems.

A simple representation of a single neuron in a neural network can be written as:

$$y = f\left(\sum_{i=1}^{n} W_i X_i + b\right)$$

Here:

- X_i are the input values,
- W_i are the weights of each input,
- b is the bias,
- $f(\cdot)$ is the activation function that introduces non-linearity.

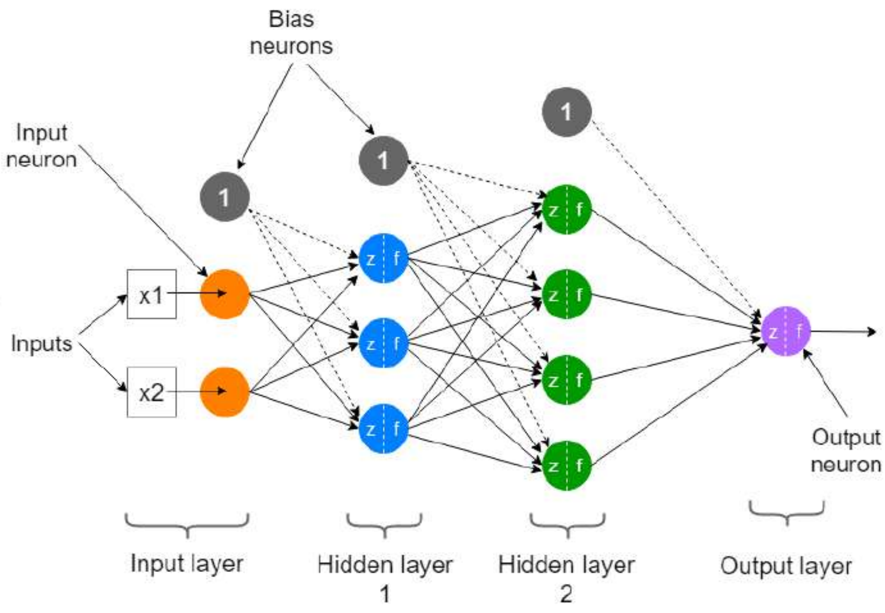


Figure 2: Structure of neural network [30]

1.3.2 Activation Functions

Activation functions are trigger that decide how neurons in a neural network should be activated, or in other words, when they "fire." They play a key role in neural networks because non-linear activation functions enable the network to learn complex patterns that can't be captured by just using a series of simple, straight-line operations [39].We are discussing some of the activation function in details which is actively used in thesis research and implementation.

Softmax:
The Softmax function is typically employed as the output activation in multi-class classification problems. It transforms a vector of real-valued inputs into a normalized probability distribution, assigning a probability to each class such that the total sums to one [28].

Mathematically, if $\mathbf{z} = (z_1, z_2, \ldots, z_K)$ is the input vector, then the Softmax is formulated as:

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}.$$

Key properties:

- **Probability Distribution:** Each output falls within the range of 0 to 1, and all the outputs together add up to 1, creating a proper probability distribution.
- **Differentiable:** The Softmax function is continuously differentiable, which makes it well-suited for gradient-based optimization techniques.
- **Amplification of Differences:** Softmax accentuates the differences between input values,larger input values result in disproportionately higher output probabilities.

Rectified Linear Unit Activation Function:

The Rectified Linear Unit (ReLU) is quite famous activation functions because it is simple to implement and works well when training deep neural networks [38]. ReLU is typically used in hidden layers.

Mathematical Definition: ReLU is defined as:

$$\text{ReLU}(z) = \max(0, z).$$

This means that for any input value:

1. If z is positive, it returns z.
2. If z is negative, it returns 0.

Key Properties:

- **Non-linearity:** ReLU introduces non-linearity, allowing neural networks to model dynamic relationships [28].
- **Efficient Gradient Calculation:** Since the derivative of ReLU is 1 for positive inputs and 0 for negative inputs, it helps with faster and more efficient training in many cases [38].
- **Sparsity:** ReLU tends to produce sparse activations because any negative input is mapped to 0. This helps in reducing overfitting.
- **Dying ReLU Problem:** A potential drawback is that neurons can get stuck and never activate if their weights result in a negative input, leading to "dead neurons."

Exponential Linear Unit Activation Function:

Exponential Linear Unit (ELU) is a improved version of ReLU designed to address some of its shortcomings, such as the dying ReLU problem [28]. It introduces a smooth exponential function for negative inputs.

Mathematical Definition: For input z, the ELU function is defined as:

$$\text{ELU}(z) = \begin{cases} z, & \text{if } z > 0, \\ \alpha(e^z - 1), & \text{if } z \leq 0, \end{cases}$$

where α is a constant that controls the value of negative inputs. Typically, $\alpha = 1$ is used, but it can be adjusted.

Key Properties:

- **Non-linearity:** Like ReLU, ELU provides a non-linear transformation that can help neural networks learn complex patterns.
- **Differentiability for Negative Inputs:** Unlike ReLU, which is not differentiable at 0, ELU is differentiable everywhere, including for negative inputs.
- **Smooth Transition:** For negative inputs, the output is not zero but follows an exponential curve, allowing the function to handle negative values in a smoother manner.
- **Reduced Risk of Dead Neurons:** The use of the exponential component ensures that ELU does not have the same "dead neuron" issue that ReLU suffers from, making it more stable in certain cases.

1.4 Depth Estimation Foundation

Depth estimation is the technique of determining the distance of objects within a scene from a particular viewpoint or camera. Unlike standard 2D images that capture only color and intensity information, depth estimation adds a third dimension i.e distance,providing insight into the spatial layout and structure of the environment. This information is critical for applications such as autonomous driving, robotic navigation and 3D reconstruction.

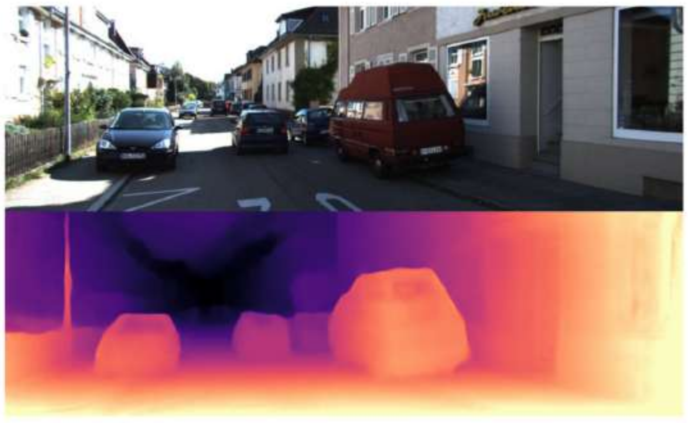


Figure 3: Depth Estimation [51]

Depth estimation is generally expressed in two different ways;

1. Relative depth: The depth is defined relative to the camera. SO the depth map can self explain that the object is far or near, however it has no absolute information [32].
2. Metric Depth Estimation: Metric depth maps provide the absolute distance of the objects from the camera reference, offering precise depth values [4].

1.4.1 Fundamental Approaches to Depth Estimation

Stereo Vision: Stereo vision mimics human binocular perception. By capturing pair of images of the same scene from slightly different viewpoints (like our eyes), depth can be inferred from the disparities (differences in pixel positions) between corresponding points in both images.

- **Working Principle**: If an object appears shifted between the left and right images, the amount of shift (disparity) can be used to compute the object's distance, given the camera parameters and the baseline (the distance between the two camera lenses).

Monocular Depth Estimation: The depth estimation using a single Red, Green, Blue (color channels) (RGB) image is challenging as it lacks direct positional cues. Instead, it relies on:

- Geometric and Perspective Cues: Objects farther away appear smaller; parallel lines converge at a vanishing point [35].
- Occlusion and Texture: The blocking of one object by another and the density of textures can offer indirect hints [51].
- Learned Priors: Modern techniques use large datasets and deep learning to learn patterns and structures that correlate image features with depth [51].

Multi-View Depth Estimation: When number of images of a scene are available from different perspective (e.g., from a moving camera), these can be combined to infer depth. The techniques like Structure-from-Motion (SfM) and Simultaneous Localization and Mapping (SLAM) rely on camera movement and tracking visual features across multiple frames to build a 3D representation of the environment. [1].

1.4.2 Evaluation Metrics

Evaluating depth estimation models involves assessing how accurately the models can predict depth from visual data, typically images or videos. There are several commonly used metrics for evaluating depth estimation performance:

1. Mean Absolute Error (MAE)

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N} |d_i - \hat{d}_i|$$

- **Description:** This measures the average absolute difference between the ground truth d_i and the prediction $\hat{d}_i$,providing a straightforward measure of prediction accuracy.

2. Mean Squared Error (MSE)

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N} (d_i - \hat{d}_i)^2$$

- **Description:** Measures the average squared difference between the ground truth and the predicted depth.

3. Root Mean Squared Error (RMSE)

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(d_i - \hat{d}_i)^2}$$

- **Description:** The square root of MSE, providing error in the same unit as the depth.

4. Root Mean Squared Logarithmic Error (RMSLE)

$$\text{RMSLE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\log(d_i + 1) - \log(\hat{d}_i + 1))^2}$$

- **Description:** Measures the logarithmic difference between the ground truth and predicted depth, which can be useful when errors in depth are proportional to the true depth.

5. Threshold Accuracy (δ)

$$\text{Accuracy:} \quad \% \text{ of } d_i \text{ s.t. } \max\left(\frac{d_i}{\hat{d}_i}, \frac{\hat{d}_i}{d_i}\right) = \delta < \text{threshold}$$

- **Common thresholds:** $\delta < 1.25$, $\delta < 1.25^2$, $\delta < 1.25^3$
- **Description:** Measures the proportion of predicted depth values that fall within a specific range relative to the ground truth.

6. Logarithmic Error (Log Error)

$$\text{Log Error} = \frac{1}{N}\sum_{i=1}^{N}|\log(d_i) - \log(\hat{d}_i)|$$

- **Description:** Measures the absolute difference between the logarithms of the ground truth and predicted depth.

7. Scale Invariant Logarithmic Error (SILog)

$$\text{SILog} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\log d_i - \log \hat{d}_i)^2 - \frac{1}{N^2}\left(\sum_{i=1}^{N}(\log d_i - \log \hat{d}_i)\right)^2}$$

- **Description:** Measures the difference between logarithmic values, removing the influence of scale differences.

8. Relative Squared Error (RelSE)

$$\text{RelSE} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{d_i - \hat{d}_i}{d_i}\right)^2$$

- **Description:** Measures the relative squared difference between the ground truth and predicted depth.

9. Relative Absolute Error (RelAE)

$$\text{RelAE} = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{d_i - \hat{d}_i}{d_i}\right|$$

- **Description:** Measures the relative absolute difference between the ground truth and predicted depth.

Choosing the Right Metric The choice of evaluation metric depends on the desired application and the nature of the depth estimation task. For instance:

- **MAE** and **MSE** are straightforward and commonly used.
- **RMSE** provides error in the same unit as depth, making interpretation easier.
- **Log-based errors (RMSLE, Log Error, SILog)** are useful for handling wide ranges of depth values.
- **Threshold Accuracy** provides an intuitive understanding of prediction quality.

Using a combination of these metrics is often recommended to get a comprehensive evaluation of depth estimation performance.

1.5 Object Detection Basics

Object detection comprises of identifying and localizing all the objects within an image using bounding boxes and assigning them the appropriate class labels [7]. As a fundamental task in computer vision, it has diverse applications across various domains, including face recognition, medical image analysis, and autonomous driving.In autonomous driving, object detection plays a vital role in the real-time recognition and localization of different road objects. This data is subsequently utilized by downstream systems such as tracking and prediction modules. There are two main tasks involved in object detection,

- Classification
- Localization

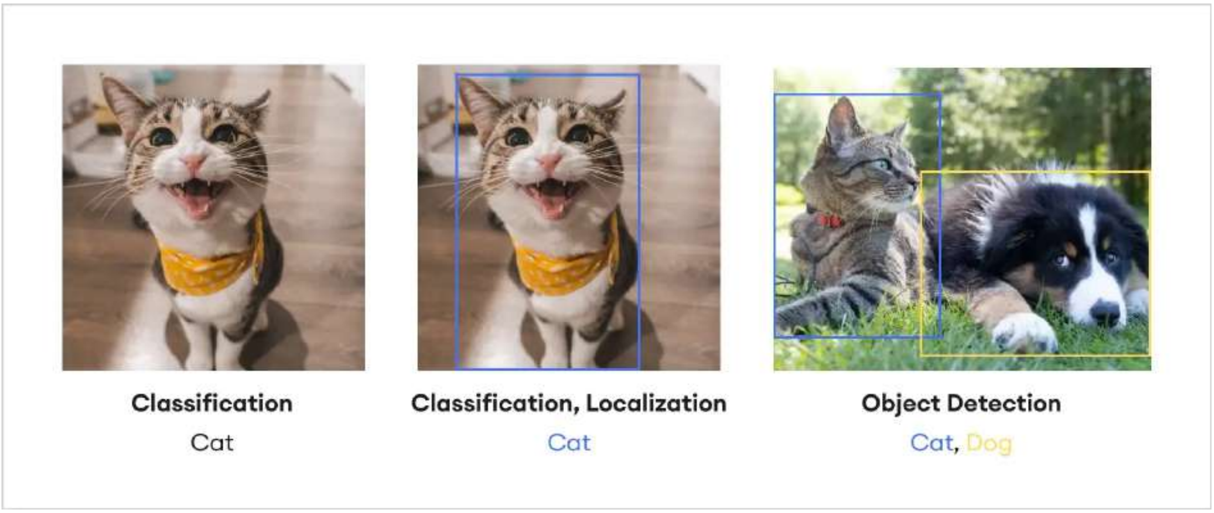


Figure 4: Illustration of the differences between classification, localization, and object detection. The first image shows simple classification (e.g., Cat), the second includes both classification and localization using a bounding box, and the third demonstrates object detection with multiple objects (Cat and Dog) each identified and localized with separate bounding boxes [41]

1.5.1 Classification

In object detection, classification refers to assigning a category label to each identified object in the input data, which may consist of images, point clouds, or their combination. Once an object is identified, the system determines its category from a predefined set of classes, such as vehicles, pedestrians, traffic signs, or other relevant objects based on the application. Convolutional Neural Network (CNN) are commonly used to extract abstract or high-level features from input data [41]. These features are then processed through fully connected layers, which output a probability distribution across the predefined multiple object classes. The detected object is ultimately classified based on the object class with the highest probability after the final activation layer.

1.5.2 Localization

Localization in object detection focuses on determining the position and boundaries of each object in an image by predicting a bounding box (bbox). A 2D bounding box is typically defined by four parameters: the coordinates of the top-left corner (x_i, y_i) and the bottom-right corner (x_j, y_j), or alternatively, the center of the box (c_x, c_y) along with its width w and height h. For a 3D oriented bounding box, seven parameters are used, including the center (c_x, c_y, c_z), length l, width w, height h, and orientation θ. The primary objective of localization is to position the bounding box as tightly as possible around the object while fully enclosing it. Bounding box predictions are achieved through regression [44], where the network learns to estimate the coordinates of the bounding box from the extracted features.

1.5.3 Object Detection Metrics

A thorough comprehension of object detection metrics is essential for effectively evaluating and comparing various models. Below is a concise overview of the different definitions and metrics employed in the assessment of object detection models.

Intersection Over Union (IoU) : The Intersection over Union (IoU) metric, based on the Jaccard Index, quantitatively evaluates the overlap between two bounding boxes by computing the ratio of their intersection to their union[5]. It specifically compares predicted bounding boxes against ground truth annotations, enabling determination of detection accuracy as either True Positive or False Positive.

The IoU is is defined as follows;

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} \tag{1}$$

True Positive, False Positive, False Negative and True Negative A clear understanding of above concepts, and threshold values is fundamental for evaluating object detection models. The following bullet points outline these key concepts:

- **True Positive (TP):** A correct detection is achieved when the predicted bounding box overlaps adequately with the actual (ground truth) bounding box. This is determined when the IoU is equal to or exceeds the predefined threshold.
- **False Positive (FP):** A correct detection is achieved when the predicted bounding box overlaps with the actual (ground truth) bounding box. This occurs when the IoU is below the set threshold, indicating that the detection is not accurate.
- **False Negative (FN):** A scenario where the model does not recognize an object that actually exists in the ground truth. This results in a missed detection, as the object is not identified by the model.
- **True Negative (TN):** Generally not applicable in object detection tasks. TN would represent the numerous potential bounding boxes that correctly do not correspond to any objects. However, due to the vast number of possible non-object bounding boxes within an image, calculating TN is impractical and thus not used in standard evaluation metrics.
- **Threshold:** The IoU value that determines whether a detection is classified as a True Positive or a False Positive. Depending on the specific metric and application, this threshold is typically set at 50%, 75%, or 95%. The chosen threshold significantly impacts the sensitivity and specificity of the detection model.

Mean Average precision (mAP): Before diving into understanding mAP we need to to understand two important terms ;

1. **Precision:** Precision a model's capability to correctly identify only the relevant objects within an image [5]. Precision is the ratio of true positive to the total number of positive detections. Mathematically, Precision is expressed as:

$$\text{Precision} = \frac{TP}{TP + FP}$$

2. **Recall:** Recall describes how well a model can detect all the relevant objects in an image. It's calculated by comparing the number of correctly identified objects (true positives) to the total number of actual objects in the ground truth [5]. The formula for Recall is:

$$\text{Recall} = \frac{TP}{TP + FN}$$

mAP is a comprehensive metric that evaluates both the precision and recall across different classes and IoU thresholds. It is the average of the Average Precision (AP) scores for all object classes. AP summarizes the precision-recall curve into a single value, reflecting the model's accuracy in both classification and localization [5]. The mAP is calculated as:

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^{N} \text{AP}_i$$

where:

- N is the number of object classes.
- AP_i is the Average Precision for the i-th class.

AP for each class is typically calculated by integrating the area under the precision-recall curve. In datasets like PASCAL VOC and MS COCO, mAP is evaluated at various IoU thresholds (e.g., 0.5, 0.75, and averaged over multiple thresholds such as 0.5:0.95 in COCO).

- **Importance of mAP:** mAP provides a single performance metric that estimates the model's capability to accurately classify and precisely localize objects across all classes. It is widely used in benchmark challenges and competitions to compare the effectiveness of different object detection models.

1.6 Autonomous sensors

Sensors play a vital role in autonomous systems and computer vision applications for environment perception. Depending on the specific application, different types of sensors are utilized. To enhance the system's robustness and address the complexities of real-world scenarios, multiple sensors are often integrated—a process known as sensor fusion. The following are some of the most commonly used sensors.

1.6.1 Camera

Cameras are one of the most commonly used sensors in autonomous systems and computer vision applications. They capture visual data in the form of images or video, providing rich information about the environment. Cameras can be RGB Monocular, stereo, or RGB-D (color and depth), enabling applications such as object detection, recognition, tracking, and depth estimation. With advancements in resolution and processing capabilities, cameras are integral to scene understanding and play a pivotal role in tasks requiring detailed visual analysis.

Camera Calibration: Camera calibration is a critical process in computer vision to accurately interpret the data captured by a camera. Two main parameters are important for camera calibration.

- **Intrinsic Parameters:** These defines the camera's internal properties, including its focal length, optical center, and any distortion coefficients resulting from lens imperfections. These parameters are expressed in the camera intrinsic matrix, often represented as:

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

  Here, f_x and f_y are the focal lengths in pixels, c_x and c_y are the coordinates of the principal point, and s is the skew coefficient (usually zero for most modern cameras).

- **Extrinsic Parameters:** It defines the camera's position and orientation in the world coordinate system, typically represented using a rotation matrix for orientation and a translation vector for position.

Calibration is performed using techniques such as the pinhole camera model and patterns like a chessboard or a dot grid. The calibration process is essential for:

- Correcting lens distortions.
- Mapping image coordinates to real-world coordinates.
- Enabling applications such as 3D reconstruction, depth perception, and sensor fusion.

Accurate camera calibration improves the reliability and accuracy of autonomous systems, ensuring precise scene interpretation and robust operation in real-world environments.

1.6.2 LiDAR

LiDAR is a sensing technology that uses laser beams to measure distance and create accurate 3D maps of the surrounding. It operates by sending out laser pulses and calculating how long they take to bounce back after hitting nearby objects. LiDAR sensors are widely used in autonomous systems for precise mapping, obstacle detection, and localization.

Key Features of LiDAR:

- **High Accuracy:** LiDAR can provide centimeter-level accuracy in distance measurements, making it ideal for applications requiring precision.
- **3D Mapping:** LiDAR generates a detailed 3D point cloud of the surrounding environment, which is useful for scene understanding and object detection.
- **Wide Field of View:** LiDAR sensors typically offer a 360-degree horizontal field of view, enabling comprehensive environment sensing.

- **Robustness:** LiDAR is less affected by lighting conditions (e.g., darkness or bright sunlight) compared to cameras, making it reliable in diverse environments.

Challenges and Limitations of LiDAR: While LiDAR is a powerful sensing technology, it has several challenges and limitations that must be addressed for optimal performance:

- **High Cost:** The good quality LiDAR with higher resolution and a wide Field of View (FoV), are quite costly, making them a costly choice for large-scale deployment in commercial applications.
- **Weather Sensitivity:** LiDAR perform badly in extreme weather conditions such as heavy rain, fog, or snow. These conditions scatter the laser pulses, reducing accuracy and effective range.
- **High Power Consumption:** LiDAR systems, particularly spinning LiDARs, consume significant power, which can be a constraint for battery-powered autonomous systems.
- **Size and Weight:** High-performance LiDAR sensors can be bulky and heavy, posing integration challenges in compact or weight-sensitive platforms like drones.

1.6.3 RADAR

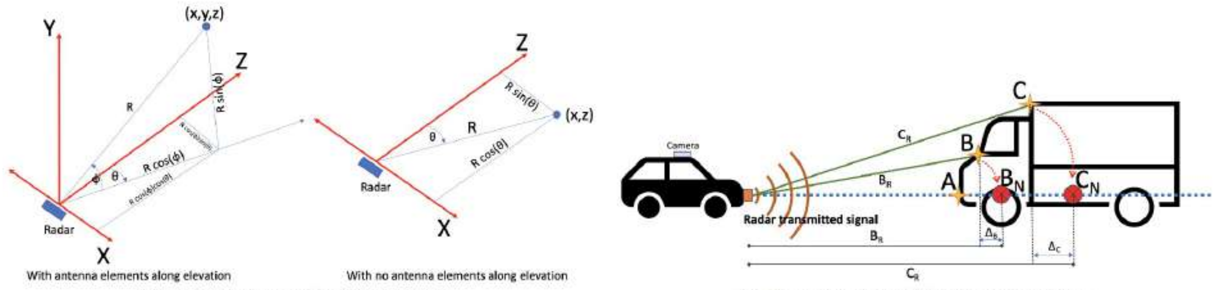
RADAR is one of the most robust sensors used in autonomous systems. Operating at high frequencies in the millimeter wavelength range, it is highly resilient to varying weather conditions which can significantly impact other sensors like cameras or LiDAR. Additionally, RADAR is cost-effective compared to high-resolution LiDAR, making it an ideal choice for applications requiring durability and affordability.

RADAR works on the principle of **time of flight** to measure the distance (range) of objects by calculating the time taken for electromagnetic waves to travel to the target and return. For velocity estimation, it utilizes the **Doppler effect**, which measures the frequency shift of the returned signal caused by the relative motion of the target.

Additional Features of Radar:

- **Long-Range Sensing:** Radar can detect objects at long distances, making it suitable for highway scenarios in autonomous vehicles.
- **Penetration Through Obstacles:** Radar waves can penetrate certain materials like fog, smoke, and even walls, enabling reliable operation in obstructed environments.
- **Wide Field of View:** Modern radar systems offer a broad coverage area, ideal for tracking multiple objects simultaneously.
- **All-Weather Reliability:** Unlike cameras and LiDAR, radar performs consistently under adverse weather and lighting conditions.

while considering all the advantages it faces lot of challenges like lower resolution, clutters, complex signal processing etc. The 3D radar are generally used for automotive application where it provides the information about range, velocity and azimuth angle[22].For the applications like depth estimation and localization, 3d RADAR introduces the measurement error due to lack of elevation information.



(a) mmWave radar point cloud geometry (b) Elevation ambiguity in radar point clouds

Figure 5: RADAR point clouds present challenges such as ambiguity in elevation (y-axis) and noise affecting the azimuth (x-axis) and depth (z-axis) components. Figure (a-left) depicts the optimal method for generating mmWave RADAR point clouds. Conversely, Figure (a-right) illustrates the geometry when the RADAR system lacks data for the elevation axis. In these scenarios, the RADAR incorrectly presumes that all returning reflections originate from a flat plane orthogonal to the sensor, leading to a consistently zero y-value that is unsuitable for practical applications. Consequently, the x and z values also become noisy, described mathematically as:

$$\Delta x = R\sin\theta(1-\cos\phi), \quad \Delta y = R\sin\phi, \quad \Delta z = R\cos\theta(1-\cos\phi)$$

Figure (b) presents a real-world example of this noise. Due to height ambiguity, points B and C exhibit depth differences, while point A is accurate because it lies within the plane perpendicular to the RADAR's plane[37].

The introduction of Four Dimension (34) RADAR, which provides elevation information in addition to the other three dimensions, presents an optimal solution to address the issues highlighted in Fig. 5. By incorporating elevation data, 34 RADAR overcomes the limitations of traditional 3D RADAR, such as elevation ambiguity and associated inaccuracies, thereby enhancing its effectiveness in autonomous systems and other applications. However 34 RADAR is still susceptible to noise and clutter from multi-path reflections, particularly in complex environments with many reflective surfaces.

1.7 Architecture Fundamentals

1.7.1 Depth Anything

Introduction: Depth anything is highly practical model. They didn't propose any novel architecture, rather they focused on different training techniques with large dataset. To achieve their goals, they developed a data engine that gathers and automatically labels a vast amount of previously unlabeled data (approximately 62 million samples). This significantly expands their dataset's scope, which helps reduce generalization errors. They explored two straightforward yet powerful approaches to make this data scaling effective:

1. The authors created a more demanding optimization target by utilizing data augmentation techniques. This approach encourages the model to actively explore more visual cues and build stronger, more reliable representations [48].
2. They implemented an supervision mechanism to encourage the model to learn rich semantic knowledge from pre-trained SOTA encoders like DINOv2 [48].

They thoroughly tested the model's zero-shot capabilities using six public datasets and a selection of randomly captured photographs [48].

Preparation of training dataset: They designed a data engine that automatically labels the unlabeled data (also called pseudo-labels). The engine uses 62 million images from eight different public datasets, such as Berkeley Driving Dataset (BDD100K) and COCO. They utilize only the images without labels. To train the annotation tool for labeling the unlabeled data, they use 1.5 million labeled datasets from six different datasets. The annotation engine assigns labels to the unlabeled data, which is then combined with the labeled data and used together for training in a self-supervised learning process.

Despite the advantages of using unlabeled images, if you have sufficient labeled data and a strong pretrained model, primary experiments found that using pseudo-labels does not improve the baseline model that is solely trained on labeled images[47].

To address this issue, instead of only providing easy guesses for the depth of unlabeled image, the model is exposed to various distortions and noise in different versions of the same image. This approach helps the model become better at understanding depth in general. Lastly, the model benefits from the excellent pretrained backbone of DINOv2 (cite ref), which has been trained on rich semantic information[48].

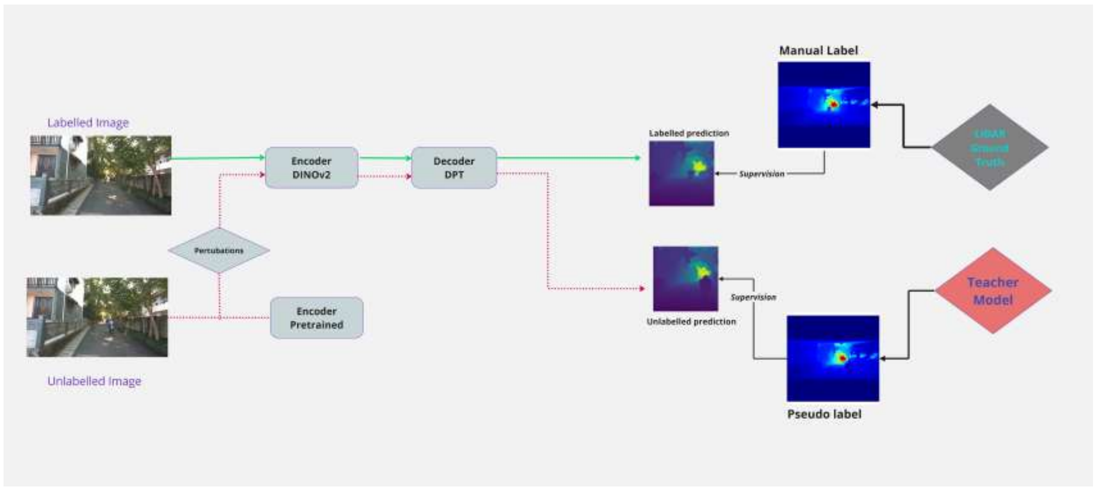


Figure 6: A labeled RGB image is passed through a DINOv2 encoder and DPT decoder and supervised directly by LiDAR ground truth. In parallel, an unlabeled image is subjected to the same perturbations which runs through a frozen, pretrained encoder and the same decoder; its prediction is then supervised using pseudo-labels generated by a teacher model. Consistency between both branches encourages robust depth learning from both labeled and unlabeled data [47].

Training Process

Learning Labelled Images: First, the depth value is transferred into disparity space as $d = \frac{1}{\text{depth}}$ and then normalized between 0 and 1. To ensure compatibility with multiple datasets, the affine invariant loss is used to account for the different sizes and shapes of each example.

The affine invariant loss L_l is defined as:

$$L_l = \frac{1}{HW} \sum_{i=1}^{HW} \rho(d_i^*, d_i),$$

where d_i^* and d_i are prediction and ground truth, respectively. The function ρ represents the affine-invariantMean Absolute Error (MAE) loss:

$$\rho(d_i^*, d_i) = \left| \hat{d}_i^* - \hat{d}_i \right|,$$

where $\hat{d}_i^*$ and $\hat{d}_i$ are the scaled and shifted versions of the prediction d_i^* and ground truth d_i. They are defined as[48]:

$$\hat{d}_i = \frac{d_i - t(d)}{s(d)},$$

where $t(d)$ and $s(d)$ are used to co-align the prediction and ground truth [48]:

$$t(d) = \text{median}(d),$$
$$s(d) = \frac{1}{HW} \sum_{i=1}^{HW} |d_i - t(d)| \, .$$

To further enhance the accuracy of the Teacher model (t), which is trained on labeled images, we initialize its encoder with pretrained weights from DINOv2.In practice, the pretrained weights are responsible for detecting the sky, which is the farthest point, and in disparity space, we set this to zero.

Why sky?

Pre-trained semantic segmentation models are often trained on large datasets where the sky is a prominent feature. Utilizing these models to detect the sky leverages their strengths and the extensive training they have undergone, ensuring high accuracy in sky detection.

Using the vast unlabeled dataset: As explained in the preparation of the dataset section, the labeled and unlabeled images are used for training the annotation engine. To robustly train the model, it is optimized with harder samples, which are obtained by applying different perturbations to the unlabeled images. This method encourages our student model to proactively gain additional visual information and learn robust representations from the unlabeled images [47].

Two main perturbations are used: the first one is strong color distortion, which includes color jittering and Gaussian blurring, and the second one is spatial distortion, known as CutMix [48].

In CutMix, two images are randomly selected, and a mask is created from them. A rectangular region from one of the unlabeled images is binary masked, where the rectangle is marked as 1. Then, using this mask, the CutMix perturbation is created as shown in the figure below:

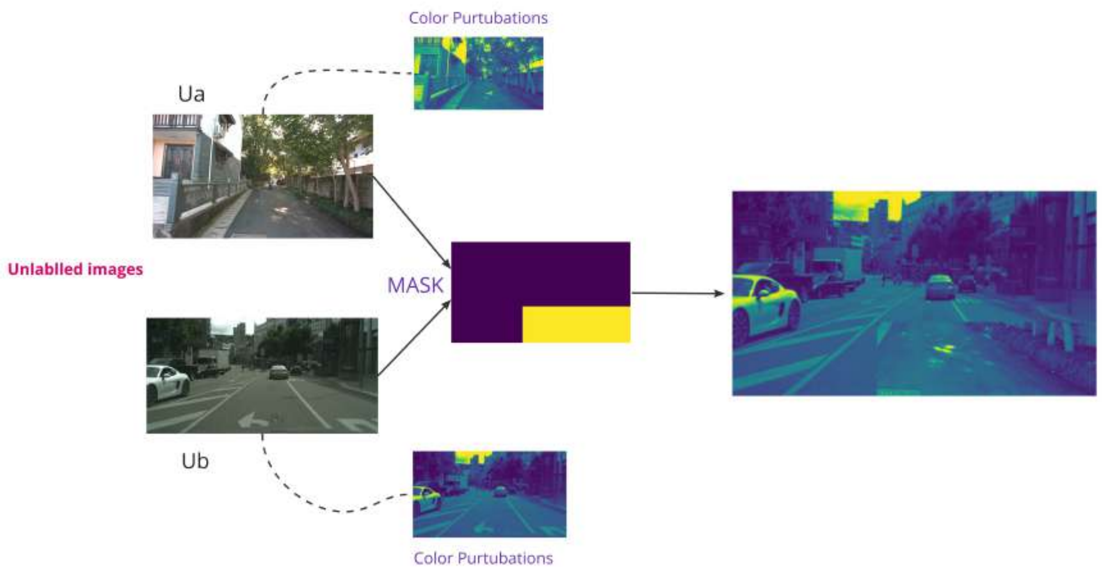


Figure 7: Perturbations to unlabeled data (adapted based on the concept from [47]).

The CutMix perturbation is defined as:

$$u_{ab} = u_a \odot M + u_b \odot (1 - M),$$

where u_{ab} is the resulting image, u_a and u_b are the original images, and M is the binary mask. The student model receives u_{ab}, while the unlabeled images are passed to the teacher model. The affine invariant loss is computed for the region M and the region $(1 - M)$ [48]:

$$L_{M_u} = \rho\left(S(u_{ab}) \odot M, T(u_a) \odot M\right),$$

$$L_{1-M_u} = \rho\left(S(u_{ab}) \odot (1 - M), T(u_b) \odot (1 - M)\right),$$

where S and T are the student and teacher models, respectively.

Finally, the weighted average of the losses is calculated as:

$$L_u = \frac{1}{HW}\left(\sum_i M_i \cdot L_{M_u} + \sum_i (1 - M_i) \cdot L_{1-M_u}\right),$$

where HW is the total number of pixels.

The CutMix perturbation is created with a 50% probability.

Semantic assistance for depth estimation: A lot of work has already been done on the improvement of multi-task learning. The authors also experimented with semantic segmentation as an auxiliary task. They created semantic segmentation labels for 4000 classes using state-of-the-art semantic segmentation models such as Grounding DINO, SAM, RAM, etc. During joint training, the model is configured to generate both depth and segmentation predictions. It achieves this by employing a shared encoder that feeds into two distinct decoders [48].

However, in evaluation, it was found that the auxiliary task did not improve the original MDE model. One reason for this was that decoding the images into a very large number of classes (4K) results in a loss of significant semantic information. The performance improved significantly when the auxiliary task was trained with fewer samples. Conversely, the depth task was already well-trained on a very good dataset, which limited the improvement[47].

Therefore, the authors decided to use more informative semantic features as complementary supervision for the depth estimation task. DINOv2, which is trained on a vast number of samples from different datasets, is a perfect choice for this supervision. They transfer the semantic features to their depth estimation model with a feature alignment loss:

$$L_{\text{feat}} = \frac{1}{HW}\sum_{i=1}^{HW}\left(1 - \cos(f_i, f_i')\right),$$

where $\cos(\cdot,\cdot)$ calculates the cosine similarity between two feature vectors. Here, f_i is the feature extracted by the depth model S, while f_i' is the feature from a frozen DINOv2 encoder[47].

An additional challenge arises because semantic encoders like DINOv2 produce the same feature for different parts of an object (e.g., the front and rear of a car), whereas every object or even pixel has different depth values. Forcing redundant and unnecessary features into the model is not useful. To overcome this challenge, a tolerance margin is applied. If the cosine similarity of f_i and f_i' exceeds a threshold α, that pixel is not considered in the feature alignment loss L_{feat}. This approach enables the method to leverage both the semantically rich representations provided by DINOv2 and the discriminative part-level representations derived from depth supervision[47].

1.7.2 ResNet Encoder

In deep learning, adding more layers to a neural network often improves performance, as seen in winning architectures after AlexNet [21], which won the ImageNet 2012 competition. However, when too many layers are added, a problem called the vanishing or exploding gradient occurs [18]. This means the gradients either become too small (close to zero) or too large, making training difficult. As a result, the training and test errors increase instead of improving.

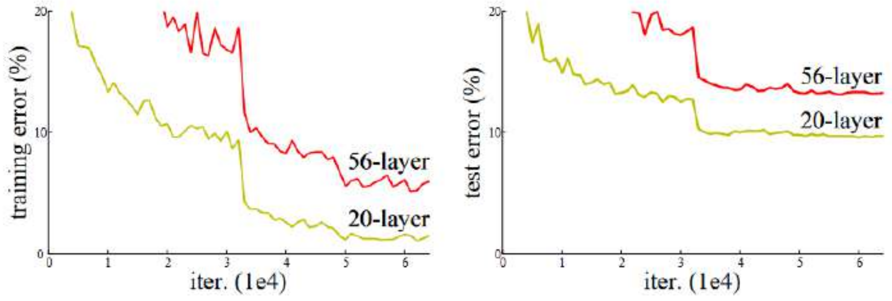


Figure 8: The graphs illustrate the training error (left) and test error (right) of 20-layer and 56-layer "plain" networks on CIFAR-10. Notably, the deeper 56-layer network shows greater error during both training and testing compared to the shallower 20-layer network [18].

To overcome this problem, Microsoft Research experts launched novel architecture Residual Network (ResNet). ResNet is architecture where number of residual blocks are stacked.

Residual Blocks: The idea of residual block was created to overcome a problem explained in 8.In this network, we use a method called as skip connections. These connections skip certain layers, directly linking the activations of one layer to a later layer. This forms a residual block, and stacking these residual blocks results in ResNet. The idea behind this approach is to allow the network to learn the residual mapping instead of having the layers directly learn the original mapping [18]. Thus, let the network fit instead of using, say, the initial mapping of $H(x)$,

$$\boxed{F(x) := H(x) - x \quad \text{which gives} \quad H(x) := F(x) + x}$$

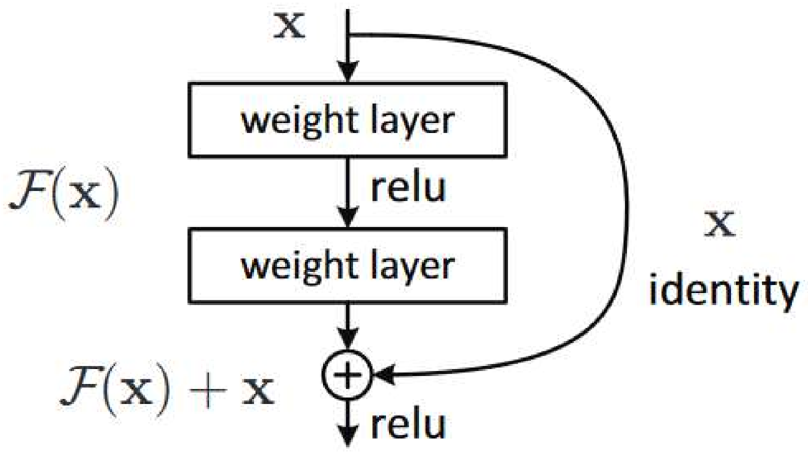


Figure 9: The input x is processed by two consecutive weight layers with a ReLU activation in between to compute the residual mapping $\mathcal{F}(x)$. An identity shortcut bypasses these layers and adds x to $\mathcal{F}(x)$, followed by a final ReLU activation on the summed output [18].

The advantage of using skip connections is that they act as a form of regularization, bypassing layers that negatively impact the architecture's performance. This enables the training of very deep neural networks without facing problems like vanishing or exploding gradients.

ResNet Architecture We increase the depth of Deep Neural Networks by stacking additional layers to improve accuracy and performance, particularly when addressing complex problems. The concept of layering is based on the idea that adding more layers enables the network to understand progressively complex non linear features.For instance, in image recognition, early layers might identify edges, subsequent layers textures, and even deeper layers complete objects. However, it has been observed that traditional CNN have a practical depth limit. Fig 8 illustrates the error percentages for training and testing data in a 20-layer network compared to a 56-layer network.

In both scenarios, the 56-layer network shows a higher error percentage than the 20-layer network. This indicates that adding more layers beyond a certain point leads to reduced performance. This degradation can be attributed to factors such as network initialization, the optimization function, the vanishing gradient problem [18]. While overfitting might seem like a potential cause, it is unlikely in this case since the 56-layer network performs poorly on both training and testing data, which is not typical of overfitting.

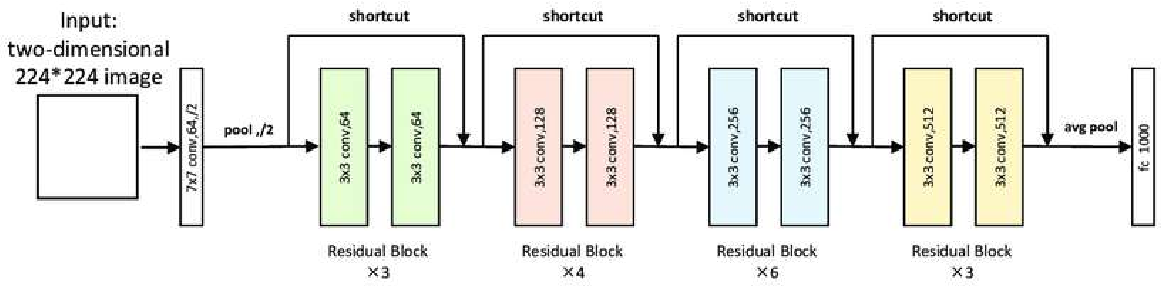


Figure 10: ResNet structure [18]

Role of ResNET in depth estimation:

- **Learning Hierarchical Features:** ResNet's residual blocks enable the network to learn both low-level (edges, textures) and high-level (objects, shapes) features effectively [18], which are crucial for understanding spatial relationships required for depth estimation.
- **Avoidance of Vanishing Gradient Problems:** Depth estimation often requires very deep networks to capture fine-grained spatial and contextual information. ResNet's skip connections ensure gradients flow smoothly through the network during backpropagation, allowing for stable training of deep architectures [18].
- **Robustness to Input Types:** ResNet is versatile and can effectively process diverse input types, such as images, radar, or LiDAR [36]. This makes it particularly useful in applications that combine multiple sensor inputs, as it can generalize well across these modalities.

- **Efficient Feature Extraction:** The encoder in depth estimation models relies heavily on feature extraction. ResNet's design allows it to extract detailed and multi-scale features efficiently [18], which are essential for accurately predicting depth maps.
- **Spatial Context Understanding:** Depth estimation requires understanding spatial relationships and context in a scene. ResNet's deep structure and ability to capture hierarchical patterns make it ideal for modeling such spatial dependencies [36].

1.7.3 Sparse Convolution Module

In the realm of computer vision,CNN typically operate on dense inputs, such as images or videos represented by pixel grids. However, challenges arise when working with sparse inputs, like 3D laser scan data, where only a fraction of the data is available. Traditional CNNs struggle with these inputs due to their irregular and sparse nature. The Sparsity Invariant CNN addresses this limitation with a novel architectural and operational framework.

Core Challenges with Sparse Inputs: Sparse data introduces two key challenges;

1. **Irregularity**: The spatial distribution of valid data points is inconsistent [42].
2. **Density Dependency**:Conventional convolutions depend on the assumption of uniform input density, which is invalid in sparse data scenarios [42].

Standard approaches to handle sparse inputs often involve filling missing data with default values (e.g., zeros) or introducing an additional channel to indicate data validity. These methods tend to under perform due to their inability to adapt to the variability in sparsity levels effectively [42].

Proposed Architecture: The Sparsity Invariant CNN introduces a sparse convolution layer, which is designed to handle sparse inputs more effectively.

Sparse Convolution Layer: Instead of using all kernel elements indiscriminately, the layer computes convolutions only on valid (observed) pixels. Each convolution operation is normalized by the number of valid pixels within the kernel area. This adjustment ensures that the output is invariant to the sparsity level of the input data [42].

The sparse convolution operation is defined as:

$$f_{u,v}(x,o) = \frac{\sum_{i,j} o_{u+i,v+j}\, x_{u+i,v+j}\, w_{i,j}}{\sum_{i,j} o_{u+i,v+j} + \epsilon} + b$$

Here, x is the input, o is the binary validity mask, w is the convolutional kernel, b is the bias, and ϵ prevents division by zero.

The main advantage of this sparse convolution operation is that it ensures the filter output remains invariant to the number of observed inputs, which can vary greatly across different filter locations in sparse and irregular data [42].

To propagate information effectively through subsequent layers, it is critical to maintain the visibility state of the data and make it available for downstream layers [42]. To achieve this,

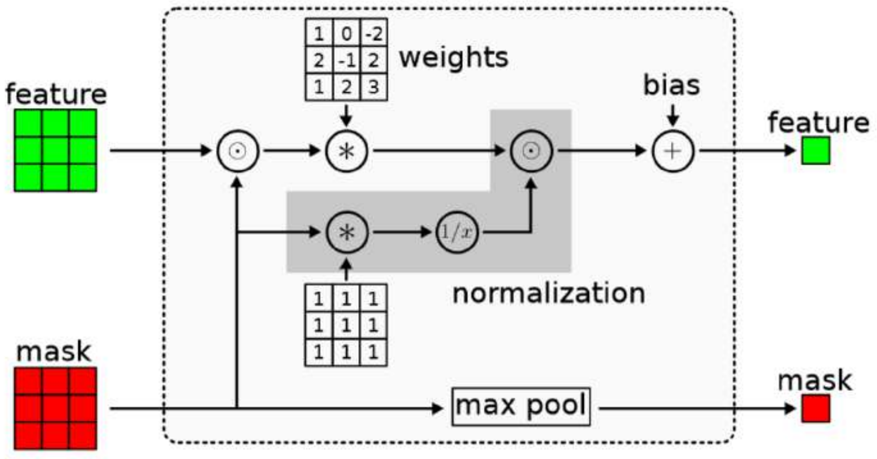


Figure 11: This diagram illustrates our sparse convolution operation. In the schematic, $(\cdot)$ signifies element-wise multiplication, $(*)$ represents convolution, $1/x$ denotes inversion, and "max pool" indicates the max-pooling operation. The incoming feature data can be either single- or multi-channel [42]

we mark output locations as "unobserved" when no input pixels have been observed. This is accomplished using a max-pooling operation:

$$f^o_{u,v}(o) = \max_{i,j=-k,\ldots,k} o_{u+i,v+j}$$

This operation assigns a value of 1 to locations where at least one observed variable is visible to the filter, and 0 otherwise. Combined with the convolution output, this updated observation mask is used as input for the next sparse convolution layer [42].The fully convolution module can be visually depicted in Fig 11.

1.8 Vision Transformer (ViT)

Transformers are encoder-decoder networks that incorporate self-attention and cross-attention mechanisms. Rich Sutton has remarked that transformers exhibit less inductive bias towards human intervention, emphasizing their data-driven nature. Transformers are primarily utilized in natural language processing (NLP) tasks but have increasingly found applications in other domains such as computer vision . Transformers can be categorized into three main types based on their architecture:

1. **Encoder-Only Models:** These are used for Image learning representations(e.g Vision Transformer (ViT)) [12].
2. **Decoder-Only Models:** These are designed for generative tasks (e.g., GPT-3) [6]
3. **Encoder-Decoder Models:** These are useful for sequence-to-sequence tasks (e.g., BERT).

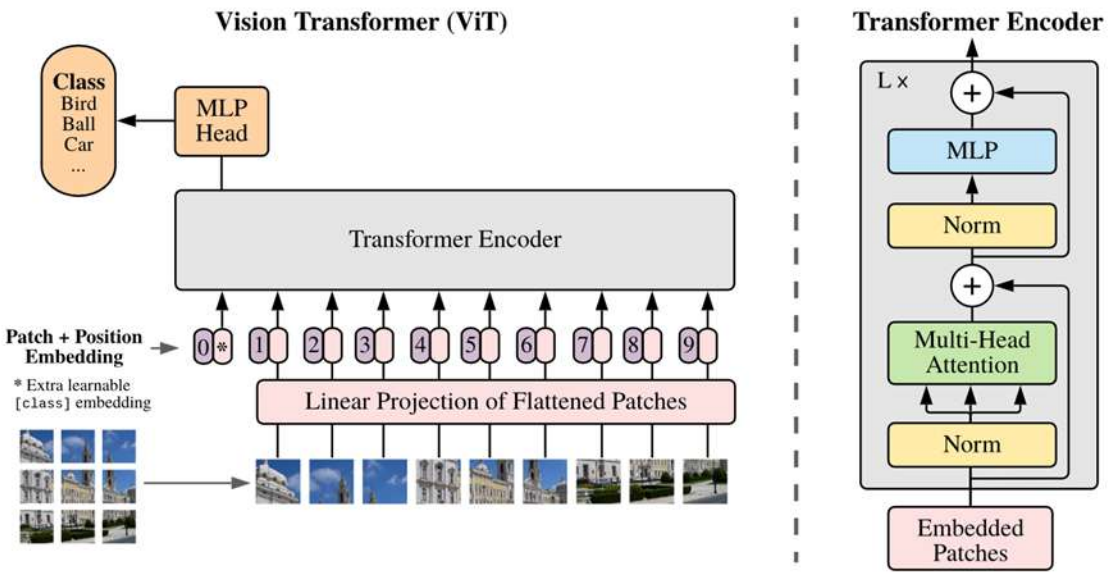


Figure 12: The Vision Transformer's architecture involves segmenting an image into fixed-size patches. These patches are individually transformed into linear embeddings, combined with positional embeddings, and then processed as a sequence by a standard Transformer encoder. For classification purposes, a common method is employed: an extra learnable "classification token" is introduced into the sequence [12].

1.8.1 ViT Architecture:

ViT adapt the transformer architecture, originally designed for NLP, to process image data.Vision transformers process images as follows [46];

1. The input image is divided into fixed-size patches, which are then flattened into a sequence of vectors, similar to tokens in NLP models.
2. These patches are linearly projected into embeddings. To account for positional information (i.e., which patch corresponds to which part of the image), positional embeddings are added.
3. A special learnable "classification token" is prepended to the sequence, which helps the model classify the input image.

Main Components of the Encoder:

1. **Embedded Patches:** Each image patch is represented by an embedding, and positional embeddings are added to indicate the origin of each patch within the image [12].
2. **Single-Head Attention:** Attention facilitates communication between patches. This involves projecting embeddings into **queries**, **keys**, and **values** [12]:

   Queries (Q): Represent "what I am looking for," computed as

$$Q = XW_Q.$$

Keys (K): Represent "what I have," computed as

$$K = XW_K.$$

Values (V): Represent "what gets communicated," computed as

$$V = XW_V.$$

The attention mechanism uses scaled dot-product attention:

$$Y = \text{Softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V.$$

The softmax function normalizes the output so that the values sum to 1. To stabilize training, the dot-product of Q and K is divided by $\sqrt{d_k}$, where d_k is the dimension of the key vectors.

3. **Multi-Head Attention:**Expanding upon the single-head attention, Multi-head attention operates by running multiple parallel attention "heads." This design allows each head to specialize in attending to different aspects or regions of the input features, leading to the model's ability to grasp a more comprehensive array of relationships and dependencies in the data [46].

   Specifically, for each head, the input embeddings are linearly projected into separate query, key, and value spaces using learned weight matrices.

$$\text{head}_i = \text{Attention}(Q_i, K_i, V_i) = \text{Softmax}\left(\frac{Q_i K_i^\top}{\sqrt{d_k}}\right)V_i,$$

   where Q_i, K_i, and V_i are the projected queries, keys, and values for the i-th head.

   The outputs of all heads are then added and passed through a final linear projection layer:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O,$$

   where h is the number of attention heads and W^O is a learned weight matrix.

   This parallel attention approach enables the model to parellely process features from various representation and different positions [46]. For example, one head might focus on local patterns, while another might capture long-range dependencies or semantic relationships.

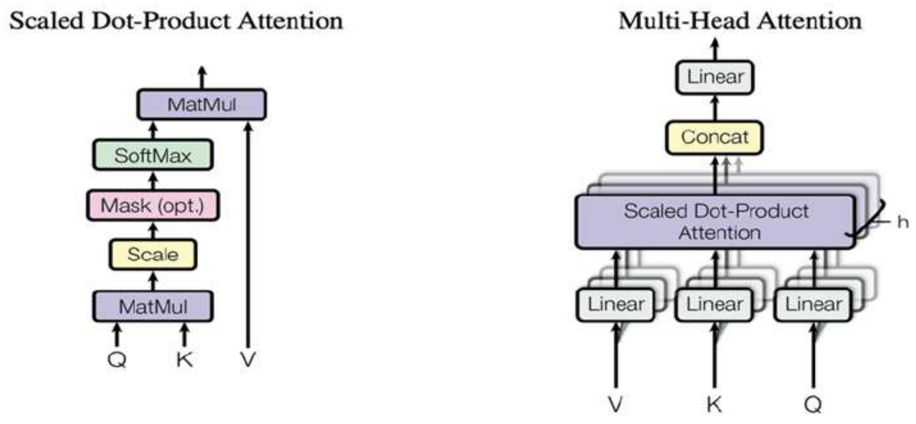


Figure 13: Attention Mechanism:(left) scaled dot-product attention, which computes a scaled similarity between queries and keys, applies an optional mask, then softmax and a weighted sum with values; (right) multi-head attention, which runs h parallel scaled dot-product attention heads, concatenates their outputs, and applies a final linear projection [46].

2 Related Work

2.1 Monocular Depth Estimation

Early research in monocular depth estimation primarily utilized supervised learning methods requiring labeled depth data, which was often scarce and expensive to collect. Garg et al. [14] pioneered an approach using calibrated stereo cameras for self-supervised training, significantly easing data collection. However, this method was constrained to stereo camera setups and often suffered from matching inaccuracies due to geometric constraints. Subsequent research explored optical flow and stereo-based self-supervision but still struggled with dynamic scenes.

To overcome dependence on stereo cameras, Godard et al. [16] proposed utilizing consecutive video frames from monocular cameras for self-supervised depth estimation, leveraging epipolar geometry. Initially, this approach resulted in poor performance, prompting further refinements such as feature matching loss, disparity smoothness loss, and left-right disparity consistency loss [16]. Despite these improvements, the requirement of future frames introduced latency, hindering real-time applications.

Transformer-based architectures later emerged, demonstrating significant improvements in monocular depth estimation performance. Ranftl et al. [31] leveraged transformers for multi-scale feature extraction, effectively handling long-range dependencies and enhancing the generalization capability of models trained on large datasets. This approach has proven robust in complex and diverse scenarios.

However, accurate generalization across diverse environments remains challenging. Researchers have addressed this by expanding dataset diversity: Chen et al. [8] used sparsely annotated ordinal relationships from internet-sourced images; Li et al. [25] created datasets using Structure-from-Motion (SfM) and Multi-View Stereo (MVS) from videos featuring static human poses; and Gordon et al. [17] estimated camera intrinsics from YouTube videos. However, data sourced from the web often requires extensive preprocessing and verification, limiting its practical usability.

To enhance model robustness, Ranftl et al. introduced MiDaS (Mixed Depth and Scale) [32], trained using multi-objective optimization across multiple heterogeneous datasets. MiDaS effectively managed scale variance through specialized loss functions and transformer-based encoders, thus improving performance across diverse scenarios. Nevertheless, MiDaS still faced challenges in accurately estimating depth on unseen datasets and under challenging conditions.

Addressing these limitations, Yang et al. introduced Depth Anything [47], designed specifically for improved generalization. Depth Anything employed a self-generated annotation engine, diverse data augmentation techniques (e.g., CutMix), and optimization across several public datasets and randomly captured images, resulting in enhanced depth estimation performance. For further details, refer to Section 1.7.1.

2.2 Metric Monocular depth estimation

Recent research has increasingly focused on enhancing model generalization for relative depth estimation, which is important for understanding spatial relationships in scenes. This approach is valuable for applications like 3D reconstruction, object recognition, and autonomous navigation, where understanding relative depth is crucial for effective spatial reasoning.

However, many real-world applications require not just relative depth but accurate metric depth values for optimal performance. For instance, in robotics, augmented reality, and autonomous vehicles, precise absolute depth measurements are essential to ensure accurate navigation, safe interaction with the environment, and the reliable overlay of digital information onto the physical world. Therefore, while relative depth estimation provides significant insights, there is a growing emphasis on developing methods that can deliver precise metric depth values to meet the demands of these critical applications [4].

Earlier work on metric depth estimation primarily focused on using encoder-decoder architectures, with LiDAR point clouds serving as ground truth. However, this approach faced significant limitations due to the scarcity of training data, as training for metric depth estimation requires all data to have consistent intrinsic and extrinsic parameters. Haoyu Ren addressed this challenge by recovering 3D scene shapes from a single image through a two-stage framework [19].This framework integrates monocular depth estimation with 3D point cloud, which were trained to forecast absent depth shifts and focal lengths [19].

Despite these advancements, treating metric depth estimation as a regression problem proved suboptimal. To improve accuracy, researchers began considering the problem as a combination of classification and regression. In response to this, AdaBins [2] introduced a transformer-based module that divides the predicted depth into bins, where the bin widths are adaptively determined across the image. The final depth is calculated by taking a weighted average of the bin centers.

Building on this, LocalBins [3] introduced a method that divides the metric depth into normalized depth and scale bins at the pixel level, rather than across the entire image. It then calculates bin centers using multiscale features from the decoder layers [3]. PixelBins simplified this approach by replacing the transformer with convolutional layers, but neither LocalBins nor PixelBins addressed the challenges of scale invariance and dissimilar training data.

To address these issues, Shariq Farooq proposed ZoeDepth, a modified version of LocalBins that leverages the robust relative depth estimation capabilities of the MiDaS model and an improved LocalBins module [32]. ZoeDepth predicts bin centers at each scale of the decoder layer. However, unlike LocalBins, which doubles the number of bin centers at each scale, ZoeDepth predicts all the bin centers at the bottleneck layer. These centers are then adaptively adjusted using attractors, making the model more robust to scale and size variations [4].During training, ZoeDepth prefers using a log-binomial distribution for probability prediction rather than softmax, as softmax is more suited to tasks with unordered classes. This choice further enhances the model's ability to generalize across different scales and data conditions.Recently Depth anything adapted the zoedepth for metric depth estimation, the excellent generalization of depth anything over relative depth estimation, provides excellent result when fine tuned with zoedepth model.

Some work has also focused on reconstructing metric 3D scenes using depth estimation models, which typically require precise intrinsic and extrinsic camera parameters. UniDepth takes an innovative approach by directly predicting 3D points from a monocular image without relying on any additional camera information [29]. UniDepth achieves this through a transformer-based, self-promptable camera module that utilizes simple harmonic encoding to learn the camera's intrinsic and extrinsic parameters [29]. The image embedding is represented in a pseudo-spherical format, effectively disentangling camera information from depth. This enables more flexible and robust 3D scene reconstruction.However, when testing the UniDepth model on the KITTI dataset, it was observed that the predicted intrinsic parameters were quite inaccurate. This highlights a significant challenge: for a metric depth estimation model to be robust, it needs to be trained on a variety of datasets. However, obtaining a large dataset with consistent single-camera parameters is difficult and time-consuming [29].

To address this challenge, Wei Yin proposed Metric3D, a method that transforms datasets captured from different cameras into a canonical camera space, drawing inspiration from techniques used in human body reconstruction [49]. In this approach, images from different datasets are coarsely standardized as if they were captured by the same camera. This is achieved by first adjusting the image appearance to simulate the conditions of a canonical camera, ensuring consistency across varying datasets. Additionally, the method involves transforming the ground truth depth labels to align with this canonical camera space, which provides consistent supervision during training.During inference, a decanonicalization transformation [49] is applied to retrieve the final metric depth, ensuring that the depth estimations are accurate and relevant to the original camera settings. To further improve training, Random Proposal Normalization Loss is employed, which is a scale-invariant loss function. This loss function enhances the model's robustness to varying scales and camera parameters, ultimately leading to more accurate metric depth estimation across diverse datasets.

2.3 Camera-RADAR Depth estimation

RADAR and camera sensors offer complementary advantages for depth estimation tasks. While cameras provide high-resolution color and texture information, they lack direct depth perception, especially in challenging lighting conditions. RADAR, on the other hand, provides robust depth measurements regardless of lighting but suffers from low resolution, sparsity, and ambiguity, particularly in the elevation angle. Combining the strengths of these two modalities has emerged as a promising approach for accurate and reliable depth estimation.

Jaun-Ting Lin describes a two-stage depth estimation model designed to reduce noise in RADAR data. The model combines RGB images with 3D RADAR points, which are projected onto an image using extrinsic and intrinsic parameters [25]. Their pilot study found that mid-fusion and late-fusion techniques outperform early fusion. In the first stage, the model generates a coarse dense depth map of the scene. This map is then compared to the RADAR depth map, and a noise filtering module is used to reject outliers. In the second stage, the RGB image, the noise-filtered depth map [25], and the initial output are processed through ResNet to produce the final depth estimation. Author doesn't address issue of error due ambiguous elevation angle.

Chen-Chou Lo proposes a novel preprocessing method to reduce the sparsity and limited field of view inherent to RADAR sensors, resulting in a denser RADAR map with reduced errors. The authors designed a model to estimate dense depth from a monocular camera and sparse RADAR data using a deep ordinal network [26].They use ordinal regression loss to turn depth estimation from a typical regression task into a classification-based approach. They preprocess the RADAR 2D points into height-extended 3D measurements.

Yunfei Long proposes a depth completion network that improves the association between RADAR data and image pixels. Automotive RADAR beams are wider than camera pixels, and the distance between the RADAR and camera makes it challenging to match RADAR data accurately with image pixels [27]. This mismatch leads to poor depth estimation when combining RADAR with video, unlike the better results seen with LiDAR and video.

To address this, the author introduces RC-PDA (RADAR Camera Pixel Depth Association), a method that matches raw RADAR depth data with a patch of nearby pixels. If any pixel within this patch has a depth value that closely matches the RADAR depth (within a certain threshold), that pixel is assigned the RADAR depth [27]. The model is then trained using binary classification with binary cross-entropy loss to improve this association.

Stefano Gasperini introduces a novel approach to improving depth estimation in dynamic driving scenarios by leveraging RADAR data using self-supervised learning. Traditional self-supervised monocular depth estimation methods often struggle with accurately predicting the depth of moving objects due to their reliance on a static scene assumption [15]. Author addresses this limitation by using RADAR data as a weak supervision signal during training and as an optional input during inference.

Shuguang Li proposes a dual-branch network that fuses RADAR and camera images to predict depth in driving scenarios. The method divides the driving scene into three parts i.e. road, trees, and sky each handled separately by different branches of the network [22]. After predicting the depth map for each part, they are combined at the final stage to create a complete depth map of the scene. However, this method has some limitations. It does not account for errors that occur when RADAR points are projected onto the image, and some information may be lost when merging the depth predictions from the different branches.

Akashdeep Singh describes the mechanics of RADAR depth formation and the errors that occur due to ambiguous elevation angles. When these angles are projected onto an image, they can lead to incorrect position associations [37]. To address this issue, the author proposes the design of an association network that links individual raw RADAR points with corresponding pixels in an image, producing a semi-dense RADAR depth map. To ultimately fuse the RADAR and image depth, they propose an encoder-decoder based fused gate network with skip connections [37].

3 Methodology

In this section we introduce the problem statement and novel architecture information.

3.1 Problem Statement

The aim of this thesis is to recover the dense depth $d \in \mathbb{R}^{H_0 \times W_0}$ through a multi-scale fusion of an RGB image $I \in \mathbb{R}^{H_0 \times W_0 \times 3}$, a RADAR-projected image $R \in \mathbb{R}^{H_0 \times W_0 \times 1}$, and monocular relative depth $d_r \in \mathbb{R}^{H_0 \times W_0 \times 1}$ as inputs. Here, H_0 and W_0 denote the height and width of the image, respectively. The model is cross-tested and regressed using ground-truth LiDAR depth maps $d_g \in \mathbb{R}^{H_0 \times W_0 \times 1}$. Depth maps are cropped to the Region of Interest (ROI) to focus on areas of interest and achieve optimal results.

An additional aim of the thesis is 2D object detection on the radar-projected image $R \in \mathbb{R}^{H_0 \times W_0 \times 1}$. Each image consists of N objects, indexed by i, and is represented by ground-truth 2D object detection boxes $Y = \{y_{\text{cls}}, y_{\text{bbox}}\}$, where y_{cls} belongs to one of K classes, and y_{bbox} represents the geometric bounding box parameters. The bounding box parameters are given as $y_{\text{bbox}} = (c_x, c_y, h_0, w_0)$, where c_x and c_y represent the center coordinates of the bounding box, and h_0 and w_0 represent the height and width, respectively.

To address this issue, we propose a transformer-based association architecture, supervised using monocular depth images. This approach imparts relative scene understanding to the fusion process. After processing through the association network, the decoder outputs the dense depth of the scene. Furthermore, the learned features from the decoder are utilized to regress the object detection head.

3.2 Model Architecture

3.2.1 Monocular depth Prediction

We utilize pre-existing networks to generate reliable and precise scaleless depth d_r estimations from a single-view image. The exceptional quality of these monocular depth predictions provides a strong basis for scale-aware learning. In this research, we incorporated State of the Art (SOTA) monocular depth models, such as MiDaS v3.1 [32] and Depth Anything [47], with pretrained weights trained on extensive datasets. Specifically, the Depth Anything model with its pretrained weights is used for predicting relative depth in this study.Both the architecture are built on transformer architecture with strong generalization.They infer the relative depth of the pixels, producing the depth map.Notably, our framework is versatile and compatible with arbitrary mono-depth prediction networks that predict depth d_r, inverse depth Z_r or others.A detailed explanation of the model can be found in Section 1.7.1.

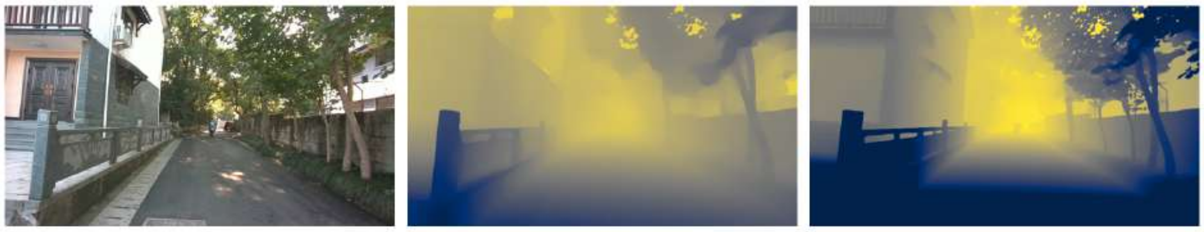


Figure 14: Left: the input image. Middle: Mono-pred from MiDaS [32]. Right: Mono-pred from Depth Anything [47].Depth anything certainly outperforms the MiDaS due to its robust generalization

The monocular depth estimation model has been trained on more than 12 million images, making it robust for predicting scaleless depth on unseen datasets [48].

3.2.2 Encoder

The process of extracting meaningful features from input data is fundamentally influenced by the type of modality. Encoders serve as the core components for deriving high-level representations tailored to each modality. This thesis focuses on three modalities: RADAR, images, and depth maps, with specific encoders employed to effectively capture the unique features of each.The following encoders are considered for these modalities;

- **RGB Image Modalities:** The image encoder employs ResNet-34 with channel configurations of 64, 64, 128, 256 and 512 across its layers.ResNet-34 offers several advantages when used for image feature extraction, particularly in tasks requiring a balance between performance and computational efficiency.By incorporating residual connections, the system helps alleviate the vanishing gradient problem, ensuring effective back-propagation even as the network depth increases [18]. The architecture is well-suited for transfer learning, allowing pretrained weights to be fine-tuned for specific tasks, saving both time and resources.We are using the pretrained ResNet network from torchvision library.
- **RADAR Modalities:** The RADAR encoder leverages ResNet-18 with channel configurations of 64, 64, 128, 256, and 512 across its layers. However, the direct application of standard convolutional operations to RADAR data is suboptimal due to the high sparsity of the input, where a significant proportion of the data contains null or zero values. To mitigate this limitation, the sparse convolution network (SCN) framework is employed, incorporating a sparse convolution module to efficiently process non-zero elements by eliminating redundant zero-value computations [42]. Following this sparse pre-processing, feature extraction is performed using the ResNet-18 architecture [18], enabling the encoder to capture meaningful high-level representations from the radar data.
- **Monocular Depth Modalities:** To effectively encode depth information, a pretrained ResNet-34 architecture is employed as the backbone, featuring channel configurations of 64, 64, 128, 256, and 512 across its layers. The ResNet-34 architecture is particularly well-suited for this task due to its residual connections, which facilitate efficient training of deep networks and mitigate the vanishing gradient problem [18].

3.2.3 Local Feature Transformer Module (LoFTR)

The local features F_A and F_B, extracted from the encoder, are passed through the Local Feature Transformer (LoFTR) module to learn the co-dependent association of depth between the RADAR and RGB images. Conceptually, the LoFTR module converts features into representations that facilitate easier matching. [40].

Preliminaries: Transformer [12] We have briefly introduce the Transformer in subsection 1.8.A Transformer encoder consists of a series of interconnected encoder layers arranged sequentially. The architecture of an individual encoder layer is illustrated in Fig.12.

The attention layer is the central component of the encoder layer. Its input vectors are traditionally referred to as the query, key, and value. Similar to information extraction, the query vector Q extracts information from the value vector V based on attention weights [12]. These weights are determined by the dot product between Q and the key vector K associated with each value V [46]. The computation graph of the attention layer is presented in Fig.15. In Fig.15 suppose the goal is to connect the L and R elements to derive their combined feature representations. This task highlights a challenge with convolutions: their inherent local connectivity means many layers must be stacked to establish broader connections between distant elements. In contrast, the global receptive field of Transformers achieves this connection using only a single attention layer [12] .

The attention layer is formally represented as:

$$\text{Attention}(Q, K, V) = \text{Softmax}(QK^T)V$$

At its core, the attention mechanism works by comparing each query with a set of key elements to identify the most relevant information. Based on these similarity scores, it assigns weights to the value vectors and combines them into a single output [45]. When a query closely matches a key, the corresponding value has a stronger influence on the final output. In the context of Graph Neural Networks, this process is often referred to as “message passing.”

Linear Transformer: In Transformers, the length of Q and K is denoted as N, and their feature dimension is D. Calculating the dot product between Q and K leads to a computational cost that grows quadratically ($O(N^2)$) with the input sequence length [40]. Using the standard Transformer for local feature matching becomes impractical, even if the input length is shortened by a local feature CNN [40]. To address this issue, we propose using a more efficient version of the standard attention layer in the Transformer.Linear Transformer [20] proposes to reduce the computational complexity of the Transformer to $O(N)$, We substitute the exponential kernel used in the original attention layer with a different kernel function to explore alternative ways of computing attention [20], defined as:

$$\text{sim}(Q, K) = \phi(Q) \cdot \phi(K)^T,$$

where $\phi(\cdot) = \text{elu}(\cdot) + 1$.The operation is illustrated in Fig.16(C).By leveraging the associativity property of matrix multiplication, the product between $\phi(K)^T$ and V is computed first. As $D \ll N$, this reduces the computational cost to $O(N)$ [20].

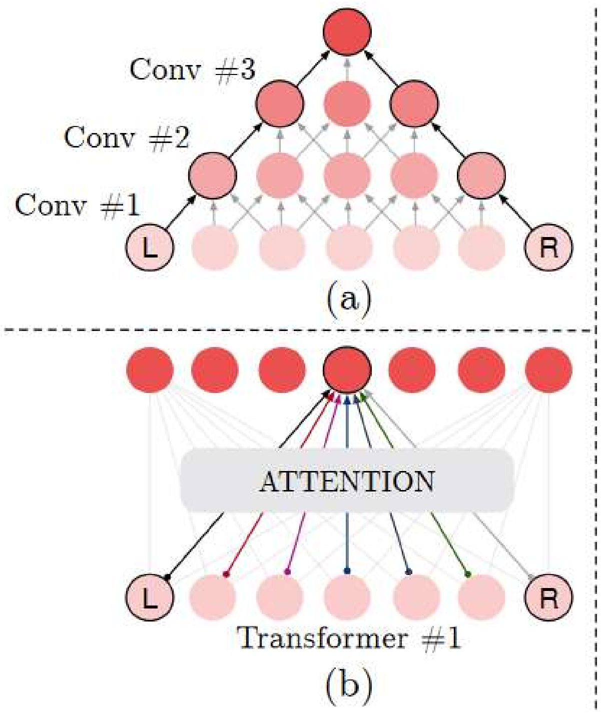


Figure 15: Comparison of receptive fields in (a) stacked convolution layers and (b) a transformer attention layer. In (a), successive convolution layers (Conv #1, #2, #3) gradually expand a local, triangular receptive field from input positions L to R. In (b), a single self-attention layer attends globally, allowing every output token (top row) to directly incorporate information from all input positions (bottom row) via learned attention weights. [12]

Self-Attention and Cross-Attention Layers: The self-attention layer uses the query, key, and value from the same input. Self-attention enables learning features directly from the input. In contrast, cross-attention takes the key and value from one input and the query from another, allowing it to learn associations between different inputs.

In the self-attention blocks, both f_i and f_j (cf. Fig. 16) originate from the same feature set—either F_A or F_B. In cross-attention blocks, one of the pair comes from F_A while the other comes from F_B (or vice versa), depending on the attention direction. This self- and cross-attention process is applied N_c times within the LoFTR module.

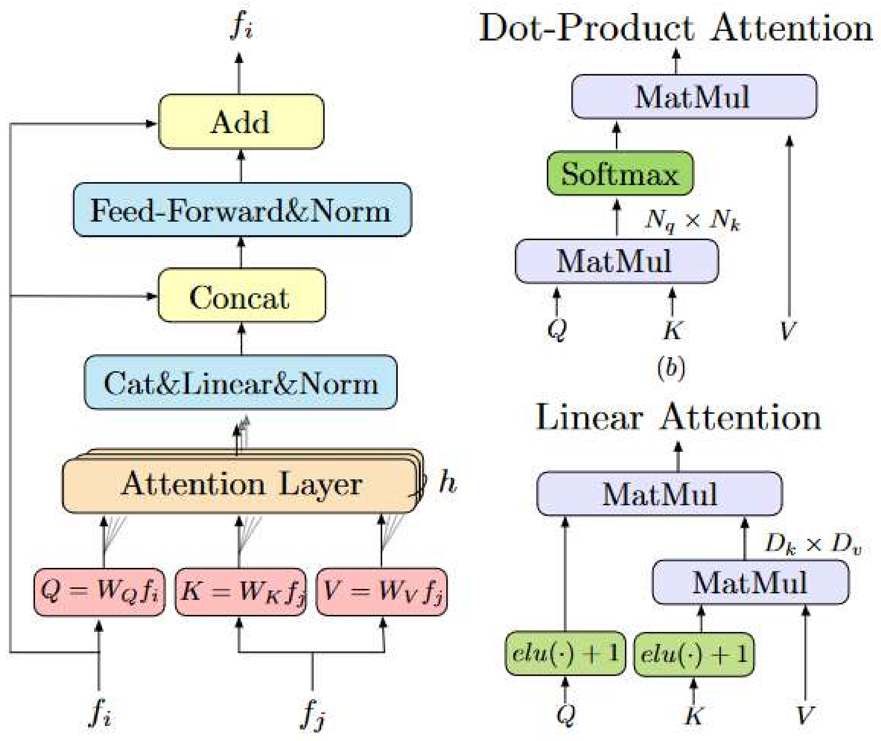


Figure 16: (a) A transformer encoder block with h parallel attention heads. (b) Standard dot-product attention, which has $O(N^2)$ computational cost. (c) A linearized attention mechanism reducing complexity to $O(N)$; the scaling factor is omitted for clarity [40].

3.3 Object Detection Head

Object detection head is extended to current depth estimation network. Object detection head uses learned latent feature from depth estimation model decoder.We are using the detection head from ultralytics [43] for efficient implementation of model YOLOv8.

You Only Look Once version 8 (YOLOv8) [44] is the latest iteration in the YOLO family of deep learning models, designed for fast and efficient object detection, segmentation, and classification. It incorporates significant improvements in model architecture, performance, and usability over previous versions like YOLOv5 and YOLOv7.

We have selected the YOLOv8 over YOLOv5 because of the following improvements:

Among all the improvements, the anchor-free method introduces a significant enhancement. Unlike YOLOv5, which relies on anchor-based detection, YOLOv8 adopts an anchor-free detection approach. This allows the model to directly predict the center of an object rather than calculating offsets from predefined anchor boxes [44]. Additionally, YOLOv8 employs a decoupled head architecture, where each head is dedicated to a specific task—bounding box regression or classification—enabling more precise and task-optimized predictions.

Feature	YOLOv5	YOLOv8
Architecture	Standard backbone and neck design. Unified head for classification and regression.	Redesigned backbone and neck for improved feature extraction and accuracy. Decoupled head for task-specific optimization.
Anchor Mechanism	Anchor-based detection with predefined anchor boxes.	Anchor-free detection, eliminating anchor boxes for simplicity.
Speed	Fast training and inference but with higher overhead due to anchors.	Faster training and inference with reduced computational requirements.
Segmentation Support	Limited to detection and classification tasks.	Supports detection, segmentation, and classification tasks seamlessly.
Input Resolution	Fixed input resolution requires retraining for changes.	Dynamic input resolution adapts without retraining.
Augmentation	Basic augmentations like Mosaic.	Advanced augmentations like Mosaic and MixUp for better generalization.
Performance	Good mAP and inference speed.	Higher mAP and faster inference, outperforming YOLOv5 across benchmarks.
Export Options	Limited to ONNX, TorchScript, and TensorRT.	Enhanced export options including CoreML, ONNX, TensorRT, and more.
Adaptability	Less adaptable to irregular object shapes and sizes.	More adaptable to varying object shapes and aspect ratios (scale-invariant).
Handling Overlaps	Struggles with overlapping objects due to anchor alignment issues.	Handles overlapping objects better due to anchor-free design.
Customization	Moderate ease for custom dataset training.	Simplified custom dataset training and fine-tuning with built-in utilities.

Table 1: Comparison of YOLOv5 and YOLOv8 [44] [43]

The anchor-free method offers several advantages over anchor-based detection [43]:

1. **Simplified Detection Pipeline:** By eliminating the need for anchor box design and tuning, anchor-free methods streamline the detection process. This reduces the dependency on hyper-parameter tuning, such as selecting anchor aspect ratios and scales, making the model more robust and easier to implement.

2. **Improved Adaptability:** Anchor-free detectors are more flexible and can better handle irregular object shapes, as they do not rely on predefined anchor boxes. This flexibility makes them inherently scale-invariant and well-suited for tasks involving objects with varying shapes or aspect ratios.

3. **Enhanced Localization Accuracy**: By directly predicting bounding box coordinates without the constraints imposed by anchor box parameters, anchor-free methods can achieve more precise object localization.
4. **Better Handling of Overlapping Objects:** Anchor-based methods often struggle with overlapping objects, as anchor boxes may not align well with multiple instances. Anchor-free methods, by design, are better equipped to handle such scenarios.
5. **Reduced Computational Cost:** Removing the need for anchor boxes can lower the computational burden during both training and inference, making anchor-free approaches more efficient in certain applications.

3.3.1 Loss Functions

The choice of loss function in a depth estimation model is crucial, as it emphasizes different aspects of the training process. In selecting the most appropriate loss function, three primary factors must be considered [26, 19]:

- **Regression Loss:** The model is trained against LiDAR ground-truth depth data. During training, the two-dimensional pixel values are regressed to match these ground-truth depths.
- **Scale-Invariant Loss:** In real-world scenarios, the same object may appear at different scales. Consequently, the model must be robust to scale variations. A scale-invariant loss can help the model adapt to objects of varying sizes.
- **Edge-Aware Loss:** Depth predictions near object boundaries often incur higher errors. However, accurately capturing edges is essential for distinguishing closely situated objects. An edge-aware loss specifically addresses this issue by emphasizing accurate depth estimation at object edges.

Depth Estimation Loss Functions : The masked L1 loss function,specifically designed for tasks where only certain parts of the input data should contribute to the loss. This type of loss is particularly useful for pixel-wise prediction tasks such as depth estimation, where certain regions of the data may be invalid or irrelevant. Below is the theoretical explanation of the components and purpose of this loss function:

1. **Regression Loss:**
   - **Standard L1 Loss:**

     The L1 loss, also known as the absolute error loss, measures the absolute difference between the predicted values (y_{pred}) and the ground truth (y_{true}). Mathematically, it is defined as:

     $$L_1 = \frac{1}{n}\sum_{i=1}^{n} |y_{\text{pred},i} - y_{\text{true},i}|$$

     – n is the total number of samples (or elements, in the case of tensors).

     – $|y_{\text{pred},i} - y_{\text{true},i}|$ is the absolute difference between the predicted and actual value for the i-th element.

This loss is relatively robust to outliers compared to the L2 loss because it penalizes large errors linearly rather than quadratically. However, for fair training in depth estimation tasks, pixels with invalid or out-of-range depth (e.g., below zero or above a certain threshold) are typically excluded. This exclusion is achieved through masking, as described below.

- **Masked L1 Loss:**

  Masking is a technique used to selectively include or exclude certain elements in the loss calculation. A binary mask M is applied, where:

  $$M[i] = \begin{cases} 1 & \text{if the } i\text{-th element is valid,} \\ 0 & \text{if the } i\text{-th element is invalid.} \end{cases}$$

  In depth estimation, for example, some pixels may lack reliable depth values due to occlusions or sensor limitations, and thus should not contribute to the loss.

  The masked L1 loss focuses solely on valid elements and is formulated as:

  $$L_{\text{masked}} = \frac{\sum_{i=1}^{n} M[i] \cdot |y_{\text{pred},i} - y_{\text{true},i}|}{\sum_{i=1}^{n} M[i]}$$

  - The numerator accumulates absolute differences only for valid elements.
  - The denominator, $\sum_{i=1}^{n} M[i]$, normalizes by the total number of valid elements.

2. **Scale-Invariant Loss:** In real-world scenarios, objects may appear at vastly different scales depending on their distance to the camera or changes in the camera's intrinsic parameters. A scale-invariant loss penalizes relative errors rather than absolute differences, making the model more robust to variations in object size and viewing distances.

   A commonly used formulation for a scale-invariant loss in depth estimation [13] involves comparing the logarithms of the predicted depth ($\log y_{\text{pred}}$) and the ground-truth depth ($\log y_{\text{true}}$). One example is:

   $$L_{\text{si}} = \frac{1}{n} \sum_{i=1}^{n} \left(\log y_{\text{pred},i} - \log y_{\text{true},i}\right)^2 - \frac{\lambda}{n^2} \left(\sum_{i=1}^{n} \left(\log y_{\text{pred},i} - \log y_{\text{true},i}\right) \right)^2$$

   where λ is a weighting factor.This term accounts for large-scale shifts in depth predictions.

   By minimizing the differences in log-space, the model emphasizes preserving relative depth relationships. Hence, even if an entire scene is scaled up or down, the loss remains relatively invariant and does not penalize proportional changes as harshly as an absolute difference measure would.

3. **Edge aware Loss:** Accurate depth estimation near object boundaries is crucial for distinguishing closely situated objects and preserving scene structure. However, these areas often exhibit higher errors due to sharp discontinuities. To address this, many depth-estimation approaches include an edge-aware smoothness term or edge-preserving regularization [16].

   The idea is to penalize depth gradients in homogeneous regions more heavily while relaxing them near strong intensity edges in the reference image.

A common formulation of edge-aware smoothness is:

$$L_{\text{edge}} = \sum_{i=1}^{n} \left(\left|\partial_x d_{\text{pred},i}\right| e^{-|\partial_x I_i|} + \left|\partial_y d_{\text{pred},i}\right| e^{-|\partial_y I_i|} \right)$$

where:

- $d_{\text{pred},i}$ is the predicted depth at pixel i,
- I_i is the intensity (or grayscale) of the corresponding reference image pixel,
- ∂_x and ∂_y denotes spatial image gradients along the x and y directions, respectively [16].

The exponential factors $e^{-|\partial_x I_i|}$ and $e^{-|\partial_y I_i|}$ reduce smoothing across edges, preserving sharp transitions in the depth map while maintaining smoothness in uniform regions [16].

Depth estimation Loss function: For training the metric depth estimation model, we use a weighted combination of the three aforementioned losses:

$$L_{\text{weightage}} = a \cdot L_{1\_loss} + b \cdot L_{\text{si}} + c \cdot L_{\text{edge}},$$

where a, b, and c are the weighting factors for the respective loss terms. Here:

$$L_{1\_loss} = \text{L1 loss}, \quad L_{\text{si}} = \text{scale-invariant loss}, \quad L_{\text{edge}} = \text{edge-aware loss}.$$

Object Detection Loss Functions: Object detection is the task of localization and classification.For training object detection models, different loss functions are used for different tasks. Following is a brief description of different loss functions used in object detection;

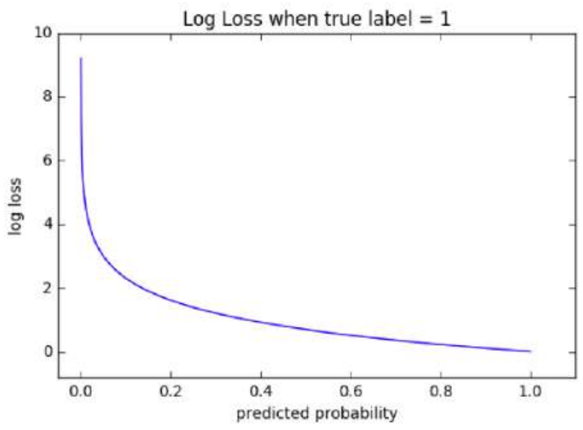


Figure 17: Cross Entropy Loss [10]

Binary Cross Entropy (BCE): For classification problems, the model's performance is often gauged by calculating the cross-entropy, as shown in Fig.17 between the model's predicted class and the actual class in so called one-hot encoding, given by:

$$L(x_i, y_i) = -\sum_{t=1}^{K} y_{it} \cdot \log(\hat{y}_{it}),$$

where,

$$\hat{y}_{it} = \begin{cases} 1, & \text{if } \hat{y}_i = y_t, \\ 0, & \text{otherwise,} \end{cases}$$

here, y_{it} is the one-hot encoding of the true class y_i, which is 1 for the correct class and 0 otherwise. $\hat{y}_{it}$ is the predicted probability that the model assigns to class t, i.e., $\hat{y}_i = f(x_i)$ is the model's output in the form of a probability vector [10].

The loss function is then summed and averaged over all observations. As a result, the loss function is minimized by adjusting the model parameters to concentrate most of the probability mass on the correct class label. The loss function equals 0 if and only if all observational inputs are correctly classified with a probability of 1.

Complete Intersection over Union (IoU) : The CIoU loss function is an enhanced loss metric employed in object detection tasks. Its purpose is to comprehensively evaluate the spatial discrepancy between a predicted bounding box and its respective ground truth bounding box. It extends the IoU loss [33] by considering not only the overlap between the boxes but also their distances and aspect ratios, leading to more precise predictions. CIoU comprises of three geometric consideration for modeling regression relationships [33].

$$CIoU = S(A, B) + D(A, B) + V(A, B)$$

where,

- S - Overlap Area
- D - Normalized central point distance
- V - Aspect Ratio

The previous IoU loss [33] has proven overlap area calculation. The overlap region is thus evaluated using the identical IoU loss.

As you can see in Fig.18 (left).The distance may vary across different training samples, so it is normalized to address this variability. The normalized center point distance and the aspect ratio need to remain unaffected by changes in regression scale [33]. Therefore, a normalized central point distance is used to effectively measure the separation between the two boxes as you can see in Fig.18

The formula for V is given as:

$$V = \alpha \left(4\pi^2 \left(\arctan\left(\frac{w_B}{h_B}\right) - \arctan\left(\frac{w}{h}\right) \right)^2 \right),$$

where:

- w_B: Width of GT box,
- h_B: Height of GT box.

The value of α is determined as follows [33]:

$$\alpha = \begin{cases} 0, & \text{if IoU} < 0.5, \\ V \cdot (1 - \text{IoU}) + V, & \text{if IoU} \geq 0.5. \end{cases}$$

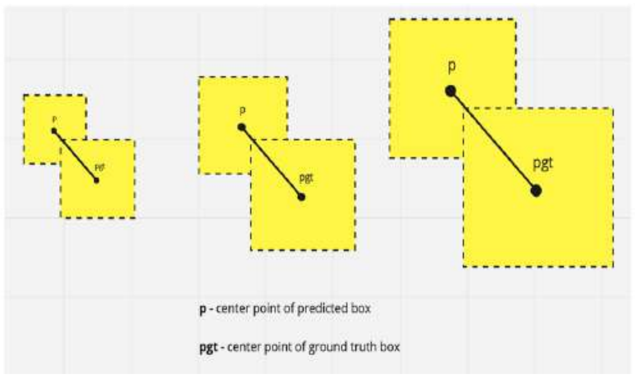
Here, α serves as a balancing parameter. When the IoU falls below 0.5, the boxes are considered poorly aligned, and enforcing aspect ratio consistency is given lower priority.

Finally, CIoU loss provides a moving direction for bounding boxes even when they do not overlap with the target box.

Distribution Focal Loss (DFL): Distribution Focal Loss (DFL) is a loss function primarily used in tasks involving bounding box regression, particularly in object detection systems [23]. It is designed to enhance the precision of bounding box localization by addressing the discretization errors introduced during the representation of continuous bounding box coordinates.

Key Ideas Behind DFL:

1. Discretization of Continuous Values:
   - Bounding box regression often involves predicting continuous coordinates. To simplify the process, these continuous values are typically discretized into a set of bins.
   - However, discretization can lead to errors due to the loss of granularity in representing precise locations.
2. Modeling as a Classification Task:


p
pgt
p
pgt
p
pgt
p - center point of predicted box
pgt - center point of ground truth box

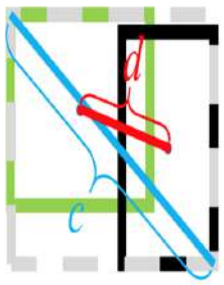


Figure 18: Left: Normalized central point distance. Right: Representation of Normalized central point distance[52].

- Instead of directly predicting the bounding box coordinates, DFL transforms the problem into a classification task.
- Each bounding box coordinate is represented as a probability distribution over the predefined discrete bins.

3. Weighted Average Prediction:
   - After classification, the final coordinate is computed as a weighted average of the bins, where the weights are determined by the predicted probabilities.

The Distribution Focal Loss is formulated as:

$$\text{DFL} = -\sum_i \left(y_i \log\left(\hat{y}_i\right) + (1 - y_i) \log\left(1 - \hat{y}_i\right) \right),$$

where:

- y_i: The true probability distribution (often represented as one-hot encoding),
- $\hat{y}_i$: The predicted probability distribution over the discrete bins.

This formula is a variant of the standard focal loss, designed to penalize incorrect predictions more heavily when the model is uncertain [23].

Advantages of DFL:

1. Better Localization Precision:By predicting a distribution over bins and using a weighted average for final outputs, DFL achieves finer granularity in bounding box localization compared to directly regressing coordinates.
2. Smooth Predictions:The approach avoids abrupt changes in bounding box predictions during training, leading to smoother and more stable outputs.
3. Compatibility:DFL can be easily integrated into existing object detection frameworks and works well with anchor-free models.

3.4 Proposed Model

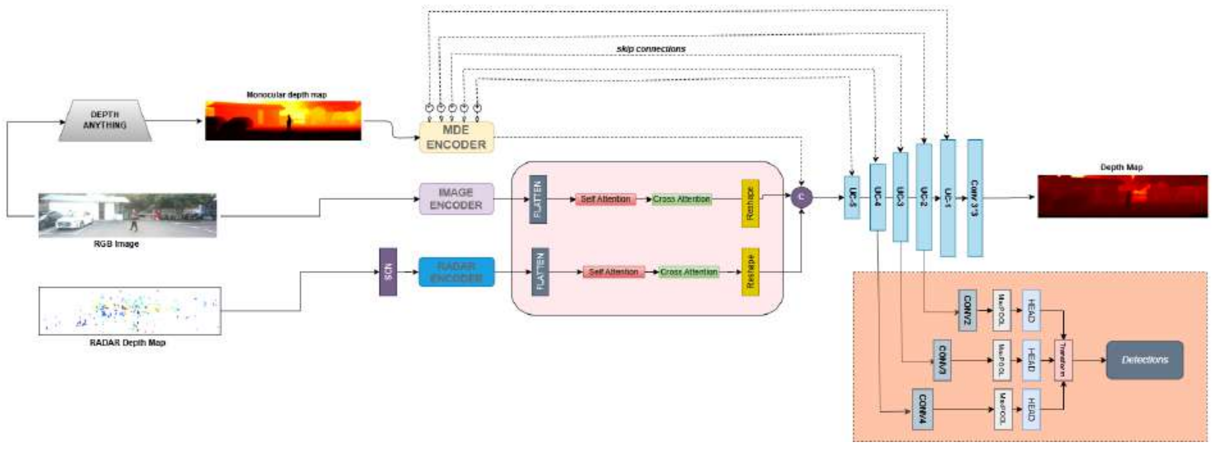


Figure 19: Model Architecture

As illustrated in fig 19, our model takes RGB image $I \in \mathbb{R}^{H_0 \times W_0 \times 3}$, a RADAR-projected image $R \in \mathbb{R}^{H_0 \times W_0 \times 1}$, and monocular relative depth $d_r \in \mathbb{R}^{H_0 \times W_0 \times 1}$ as inputs. The processing of the data to achieve the desired result, is divided into parts as follows.

1. The monocular depth of the RGB image is recovered using SOTA monocular depth estimation model.
2. Radar maps are processed through SCN which eliminates the non zero RADAR depth.
3. All the three modalities are processed through association network and the depth is predicted using decoder.
4. Latent feature from the different scale of decoder is extracted to regress the object detection HEAD

Features from an image are extracted using an image encoder (ResNet-34), which reduces the input image dimensions by a factor of 1/32. The ResNet-34 encoder is designed to capture high-level features from the input image, as explained in paragraph 1.7.2.For the radar map, feature extraction begins with the elimination of invalid pixels using the Sparse Convolution Module (SCM). The processed radar map is then passed through a ResNet-18 encoder to extract meaningful features.In addition, the RGB image is processed through a monocular depth estimation model to generate a relative depth map of the scene. The monocular depth model outputs the inverse relative depth which is first inverted to normal depth for linear generalization.These depth maps are subsequently fed into the ResNet-34 encoder for feature extraction.As ResNet is trained on ImageNet dataset, which all are three channel samples, so to adopt the ResNet for a single channel samples, the 3 channel weights are averaged for better utilization of the pretrained weights. Detailed information about the radar encoder and the monocular depth estimation model can be found in paragraph 1.7.3 and section 1.7.1, respectively.

Simultaneously, features extracted at each layer and at different scales from the encoders for all three inputs are stored. These features are later used as "**skip connections**" in the decoder to enhance the reconstruction process in subsequent stages.

The latent features extracted from the image and radar encoders are processed through the LoFTR module. LoFTR, a transformer-based architecture, is particularly effective at learning associations between different modalities. To meet the requirements of linear transformers, the latent features are flattened into one-dimensional vector representations before being passed through the attention layers. As detailed in paragraph 3.2.3, the self-attention layers operate by taking a query, key, and value derived from their respective inputs—radar features and image features. These are subsequently processed by cross-attention layers, which take the query from one modality and the key and value from the other. This interaction facilitates robust cross-modal learning as it involves a larger receptive field for the cross-modal association , enhancing the overall feature representation.

The features extracted from the LoFTR module are added along the channel (c) dimension with the features obtained from the monocular depth encoder. The combined features are then passed to the decoder, which consists of six successive up-sampling layers. These layers progressively up-sample the latent features to match the original input dimensions. At each up-sampling stage, skip connections from the encoder are concatenated along the channel dimension. These skip connections play a crucial role in enhancing the reconstruction quality of the depth map by preserving fine-grained details from earlier stages.

The latent features $L1$, $L2$, and $L3$ extracted from the model decoder are utilized for object detection regression. However, the dimensionality of these features does not directly conform to the original architecture of YOLOv8. To resolve this, a Convolution, Batch normalization and ReLU (CBS) block with a residual connection is employed to align the feature dimensions.

Once the dimensions are matched, the features are processed through a $4{\times}4$ max-pooling operation before being passed to the object detection head. The outputs from the three detection heads are subsequently concatenated and fed into the Transform block. The Transform block is responsible for generating anchors required for bounding box regression during training. Furthermore, classification is integrated into the Transform block, leveraging a softmax activation function to predict class probabilities.

4 Experiments

4.1 Datasets

ZJU-4DRadarCam Dataset [22]: We utilized the ZJU-4DRadarCam dataset to train our model. This dataset was collected on a university campus using a ground robot equipped with Oculii's EAGLE 4D Radar, a RealSense D455 camera, and RoboSense M1 LiDAR sensors. It includes diverse driving scenarios, covering both urban and wilderness environments. The ZJU-4DRadarCam dataset contains a total of 33,409 synchronized Radar-Camera key-frames, divided into 29,312 frames for training and validation and 4,097 frames for testing.

Berkeley Driving Dataset (BDD100k) [50]: We utilized the BDD100K dataset for training and evaluation of our object detection model. BDD100K is a comprehensive and large-scale dataset designed for autonomous driving tasks, encompassing over 100,000 diverse images. It includes detailed annotations for object detection, instance segmentation, lane detection, and other tasks. The dataset captures a wide array of driving scenarios, such as varying conditions (rain, snow, fog), various times of the day (daytime, nighttime, dusk), and geographic diversity across urban, suburban, and rural settings. Its object detection annotations include a broad range of classes, such as vehicles, pedestrians, cyclists, traffic lights, and traffic signs, making it an excellent choice for developing robust models capable of performing in real-world driving environments. This diversity ensures that models trained on BDD100K are well-equipped to handle the complexities of real-world scenes.

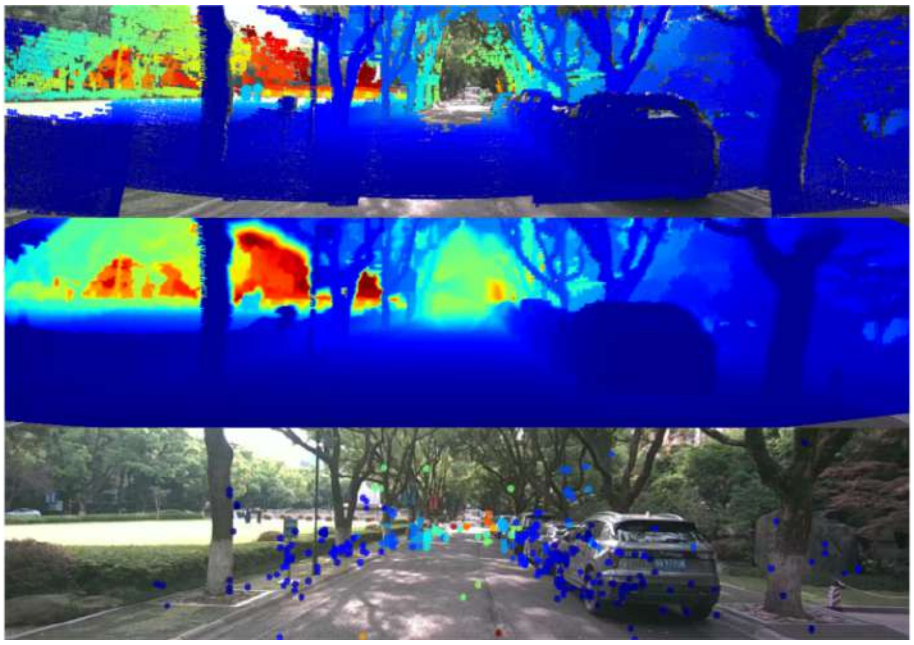


Figure 20: ZJU-4DRadarCam dataset with LiDAR depth d_{gt}, interpolated LiDAR depth d_{int}, and depth from 4D Radar point cloud P shown from top to bottom [22].

4.2 Data Prepossessing

The ZJU-4DRadarCam dataset was adapted to train the metric depth estimation model. The initial dataset processing steps are illustrated in Fig. ??. The input image $I \in \mathbb{R}^{H_0 \times W_0 \times 3}$ has dimensions of $720 \times 1280 \times 3$. The ground truth and radar-projected image maps have dimensions of $720 \times 1280 \times 1$.

During data collection, the authors observed that depth information is most critical in the central region of the autonomous environment's perception field. Additionally, they noted a significant increase in error as the elevation angle increases, highlighting the challenges associated with depth estimation at higher elevation angles. Consequently, the dataset design prioritizes accurate depth representation in these regions as shown in Fig.21.

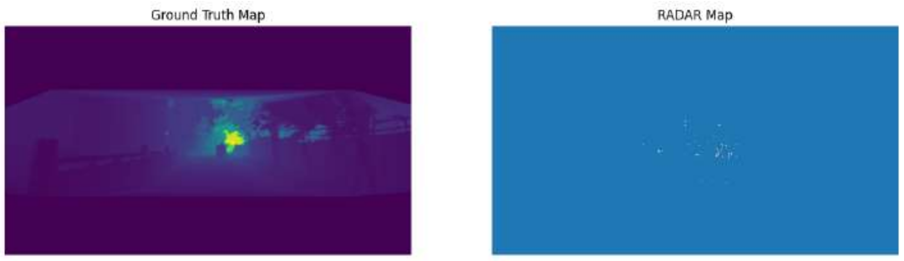


Figure 21: LiDAR and RADAR sample

To address this, the inputs are cropped to a size of ($320 \times 1280 \times$ channels), removing irrelevant regions from the top and bottom of the input. Similarly, the monocular depth maps are cropped to the same dimensions after being converted to normal relative depth.

To improve the training data's diversity and robustness, a range of data augmentation methods are utilized. These include random and center cropping to focus on relevant regions of the input, resizing to maintain uniform dimensions, and horizontal flipping to introduce variability in object orientation. Brightness and contrast adjustments simulate different lighting conditions, while sharpening enhances image details. Gaussian noise is added to mimic sensor imperfections and improve model resilience to noisy inputs. Additionally, Cut-mix augmentation is employed, blending patches of different images and labels to encourage the model to generalize better by learning object-level features across varying spatial contexts. These enhancements collectively ensure the model's adaptability to diverse real-world scenarios and can be visualized in 22.

4.3 Training Details

The training of the multitask algorithm is divided into two stages. First, the depth estimation model is trained, followed by the addition of the object detection head.

4.3.1 Metric depth estimation model training

The metric depth estimation model trained on the ZJU-4DRadarCam dataset [22] for 85 epochs with variable learning rate. To stabilize training, a OneCycle learning rate scheduler is used. A batch size of 4 is maintained throughout training, and the Adam optimizer is employed. The model training utilizes parallel data processing with the PyTorch Lightning Fabric module on

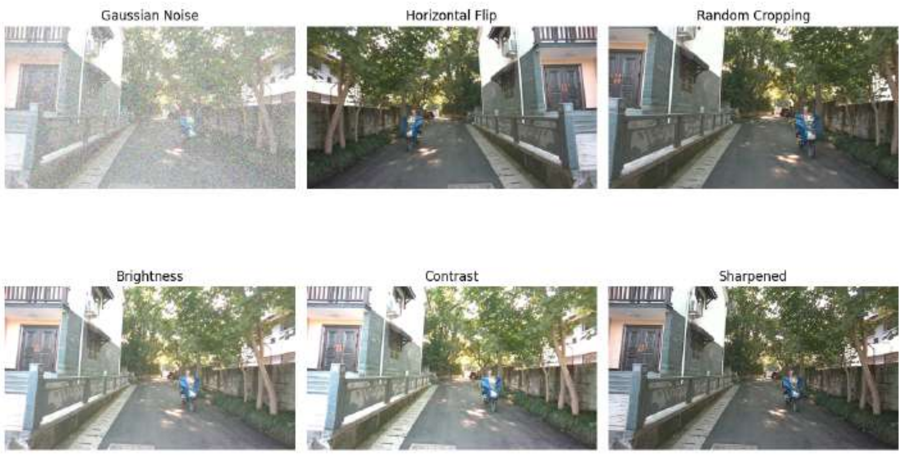


Figure 22: Color and Geometric Augmentations (own illustration)

three A100 GPUs each with 80 GB RAM, enabling end-to-end training.The training runs for 24 hours. We are following the default saved weights used in pretrained ResNet architecture.We are also applying the early stopping with patience of 10 epochs with the help of validation loss.

4.3.2 Object detection head training

The object detection model is trained on the BDD100K dataset, with input images resized to $(320 \times 1280 \times 3)$ to align with the architecture of the metric depth estimation model. Due to the absence and irrelevance of radar point clouds and monocular depth maps, these inputs are replaced with zero-filled tensors. The weights of the depth estimation layers are frozen, and only the parameters of the object detection head are updated during training.

The loss function is a weighted combination of Binary Cross-Entropy (BCE) loss, Distribution Focal Loss (DFL), and Complete Intersection over Union (CIoU) loss, with respective weights of 0.5, 1.5, and 7.5. The object detection head is trained for 100 epochs using a learning rate of 10^{-3} and a weight decay of 1×10^{-2}. A OneCycle learning rate scheduler is employed to stabilize training. A batch size of 16 is utilized, and weight optimization is done using the Adam optimizer. Training is executed on three NVIDIA A100 GPUs, each with 80 GB of VRAM, using PyTorch Lightning Fabric for efficient multi-GPU parallelism. The total training duration is approximately 7 hours.

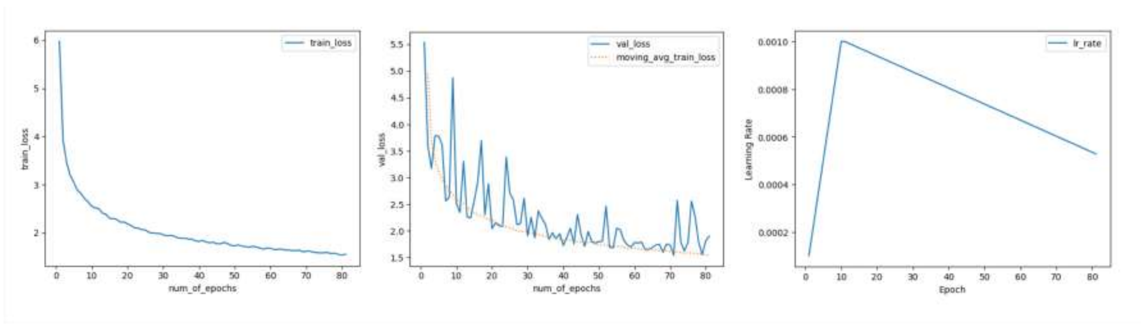


Figure 23: Training dynamics of the depth estimation model: (left) the declining training loss; (center) the validation loss (solid) alongside the moving average of the training loss (dashed); and (right) the learning-rate schedule showing a warmup to peak and subsequent linear decay

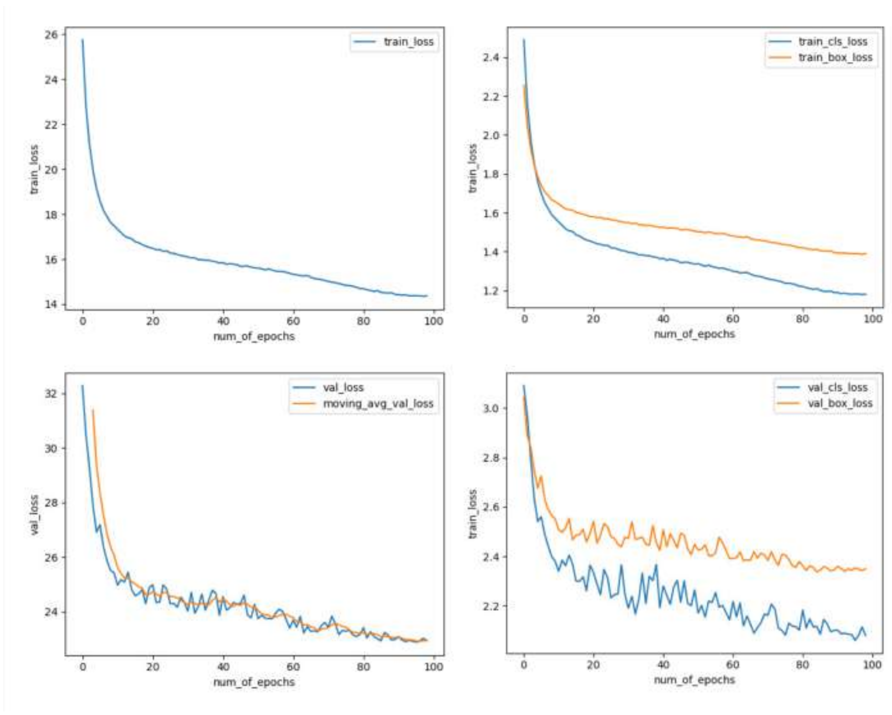


Figure 24: Training and validation loss curves for the object detection head: (a) overall training loss; (b) training classification loss (blue) and bounding-box regression loss (orange); (c) validation loss (solid blue) with its moving average (dashed orange); (d) validation classification loss (blue) and bounding-box regression loss (orange).

4.4 Evaluation and Results

4.4.1 Test Dataset and Preprocessing

The evaluation of the depth estimation model is performed using state-of-the-art (SOTA) performance metrics on a held-out test set, which constitutes 10% of the original dataset. This test set is kept in its original, unaugmented form to preserve the integrity of the data. For preprocessing, the test data follows the same deterministic procedures applied to the training data, including operations such as normalization and resizing. However, it excludes any augmentation techniques,typically employed during training to enhance data diversity,ensuring a consistent and unbiased assessment of the model's performance.

4.4.2 Comparative studies with State of the art

The evaluation metrics are adapted from SOTA methods and include key performance indicators such as MAE, Root Mean Square Error (RMSE), Log10 Error, and Threshold Errors (d_1, d_2, d_3).The brief explanation of the evaluation metrics can be found in section 1.4.2.The metric depth estimation is evaluated against the ground truth depth (d_{gt}) for ranges of 20m and 40m.

Distance (m)	Method	MAE	RMSE	δ_1	δ_2	δ_3
50	DORN [26]	2210.171	4129.691	0.783	–	–
50	Singh [37]	1785.391	3704.636	0.831	–	–
50	RadCam [22]	1067.531	2817.362	0.922	–	–
50	Ours	**895.23**	**1763.43**	**0.9430**	0.98	0.9930
70	DORN [26]	2402.180	4625.231	0.777	–	–
70	Singh [37]	1932.690	4137.143	0.828	–	–
70	RadCam [22]	1157.014	3117.721	0.921	–	–
70	Ours	**1131.2**	**2134.34**	**0.9330**	0.9703	0.9813
80	DORN [26]	2447.571	4760.016	0.776	–	–
80	Singh [37]	1979.459	4309.314	0.828	–	–
80	RadCam [22]	**1183.471**	3228.999	**0.920**	–	–
80	Ours	1338.8	**2776.0**	0.9120	0.9629	0.9811

Table 2: Comparison of methods for different distances.

The table presents key evaluation metrics i.e. MAE, RMSE and threshold errors (δ_1, δ_2, δ_3),for depth estimation methods evaluated at distances of 50m, 70m, and 80m, with the metrics adapted from SOTA approaches; notably, at 50m and 70m the proposed method ("Ours") achieves considerably lower MAE and RMSE values along with higher δ_1 scores compared to DORN [26], Singh [37], and RadCam [22], indicating more accurate depth predictions, while at 80m the proposed approach remains competitive with RadCam [22] by delivering a lower RMSE and comparable threshold accuracy.

Table 3: Model Complexity Overview

Metric	**Value**
Total Parameters	72 M
Multiply-Accumulate (MACs)	48.84 GMac
Approx. GFLOPs	98
Frames per Second (FPS)	37
FPS with Monocular Depth Inference	
Depth Anything Small	25
Depth Anything Medium	20
Depth Anything Large	12

Model Complexity: Our proposed network comprises approximately 72 million parameters and requires about 98 GFLOPs for each forward pass, achieving a speed of 37 frames per second on typical hardware. The high parameter count enables the model to learn rich, detailed features, while the GFLOPs value indicates the computational effort per inference. Despite these requirements, the model's runtime performance remains practical for real-world applications at 37 FPS.

4.4.3 Distance-Based Performance Analysis

The model can infer depths from 0 to 90 meters. Because short- to medium-range accuracy is particularly important for most autonomous applications, we evaluated the model by dividing the depth range into four bins at 20, 40, 60, and 80 meters.

Table 4: Performance at Different Distances

Distance (meters)	**MaE**	**RMSE**	δ_1	δ_2	δ_3
20	655.6	1446.3	0.9720	0.9840	0.9980
40	895.0	1763.0	0.9430	0.9800	0.9930
60	1131.2	2134.0	0.9330	0.9703	0.9813
80	1338.8	2776.0	0.9120	0.9629	0.9811

As the table indicates, the model yields lower errors and higher accuracy fractions ($\delta_1, \delta_2, \delta_3$) at shorter distances (20 m). Both MAE and RMSE increase as the prediction distance grows, reflecting a decline in performance over longer ranges. Likewise, the accuracy metrics ($\delta_1, \delta_2, \delta_3$) show a consistent drop with increasing distance, demonstrating that while the model is most reliable at short to medium ranges, its predictive capabilities diminish as the target depth range extends.

4.4.4 Ablation Studies on effect of different modalities

To gauge how each modality affects the output of our depth estimation model, we inferred the model under three conditions—using all modalities (image + radar + monocular depth map), without radar (image + monocular depth map), and without the depth map (image + radar). The results clearly show that using all modalities achieves the lowest MAE and RMSE at every

Method	**Range (m)**	**MAE (mm)**	**RMS (mm)**	$\boldsymbol{\delta_1}$
Image + Radar + Monocular Depth Map	20	655.6	1446.3	0.9720
Image + Radar + Monocular Depth Map	40	895.0	1763.0	0.9430
Image + Radar + Monocular Depth Map	60	1131.2	2134.0	0.9330
Image + Monocular Depth Map	20	1000.4	2180.4	0.8663
Image + Monocular Depth Map	40	1518.8	2925.5	0.8470
Image + Monocular Depth Map	60	1779.5	3344.0	0.8417
Image + Radar	20	2168.3	3442.3	0.6080
Image + Radar	40	3453.1	5175.6	0.4975
Image + Radar	60	4039.5	6137.3	0.4704

Table 5: Evaluation metrics for different modality ablations

range, accompanied by the highest δ_1 values. When radar is removed, performance remains moderate (MAE around 1000 mm at 20 m) but is still less accurate than the full setup. Finally, removing the depth map (leaving only image + radar) leads to the largest errors, particularly at longer ranges. Overall, these findings highlight the importance of each sensing input,especially the learned depth map and confirm that combining all available modalities yields the most robust depth estimates.

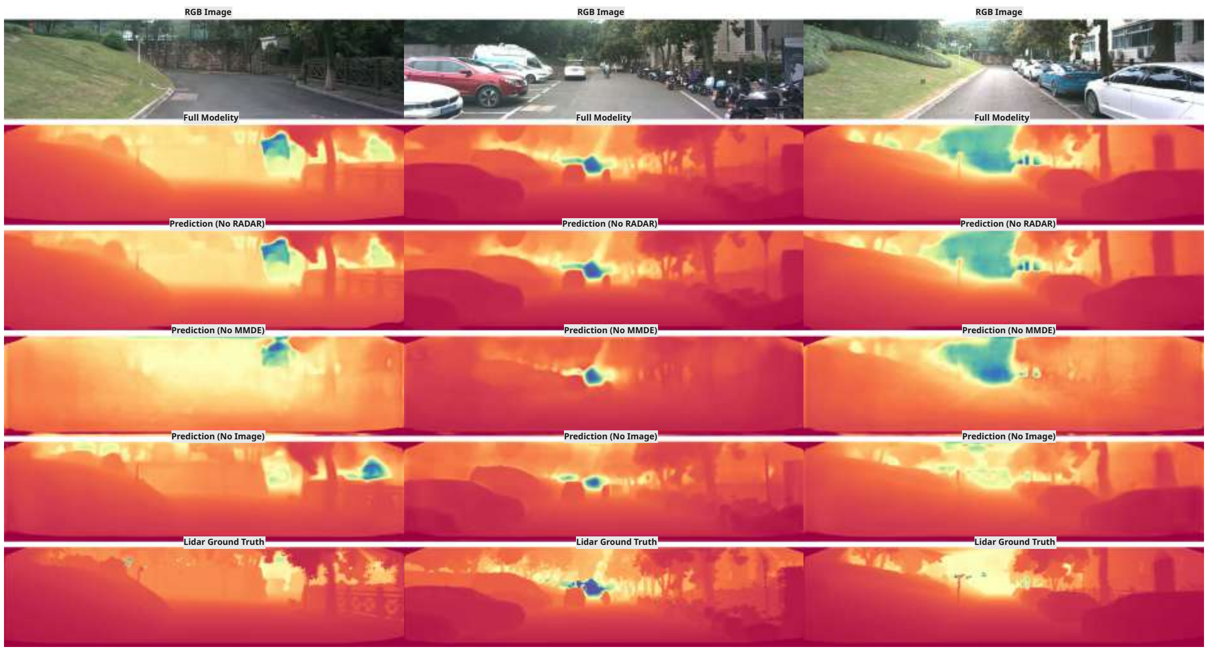


Figure 25: Ablation study on modality contributions for metric depth estimation. Each column presents a different scene. From top to bottom: input RGB image; prediction with full modalities; prediction without LiDAR; prediction without monocular depth map; prediction without the RGB image; and LiDAR ground truth.

4.4.5 Grid Error Analysis

Traditional evaluation metrics computed over the entire depth map often fail to reflect the spatial variability in model performance across different regions of the image. This limitation is particularly relevant in applications such as autonomous technology, where accurate depth estimation is critical within the forward-facing field of view. To address this, we incorporate a grid-based evaluation strategy that enables localized analysis of depth prediction accuracy.

Specifically, both the predicted metric depth map and the LiDAR ground truth are divided into non-overlapping patches of size 40×40 pixels. For each patch, the MAE is calculated, and the results are aggregated into an 8×32 grid representation. This approach allows for a more granular assessment of the model's performance across the image for the threshold of 60m.

Furthermore, to assess the contribution of RADAR data to depth estimation, we also compute the grid-based MAE using depth predictions generated without incorporating RADAR input. This comparative analysis provides deeper insights into the influence of multi-sensor fusion on depth prediction quality.

Following are main observation made from the grid map showed in Figure 26;

- The MAE in the top–middle region is comparatively high because that area is largely occupied by the sky. Across the rest of the scene, however, the MAE remains below the 1 m threshold, demonstrating good overall model accuracy.
- Figure 26 presents a depth map predicted without radar input. In this case, the MAE in the frontal view deteriorates because no radar data are available. As Figure **??** shows, radar returns are concentrated near the center of the scene; their absence therefore causes a significant drop in accuracy.
- Both heat-maps exhibit a clear vertical gradient: errors drop sharply below patch-row 2 and stay well under 1,m for almost all ground-level patches, indicating that the network is consistently reliable at near- to mid-range distances.
- The error surface is largely laterally symmetric, with the highest values centred on the optical axis and lower errors toward the image edges. This pattern suggests that strong parallax cues at the periphery compensate for the lack of radar priors, while the texture-poor sky region remains challenging.

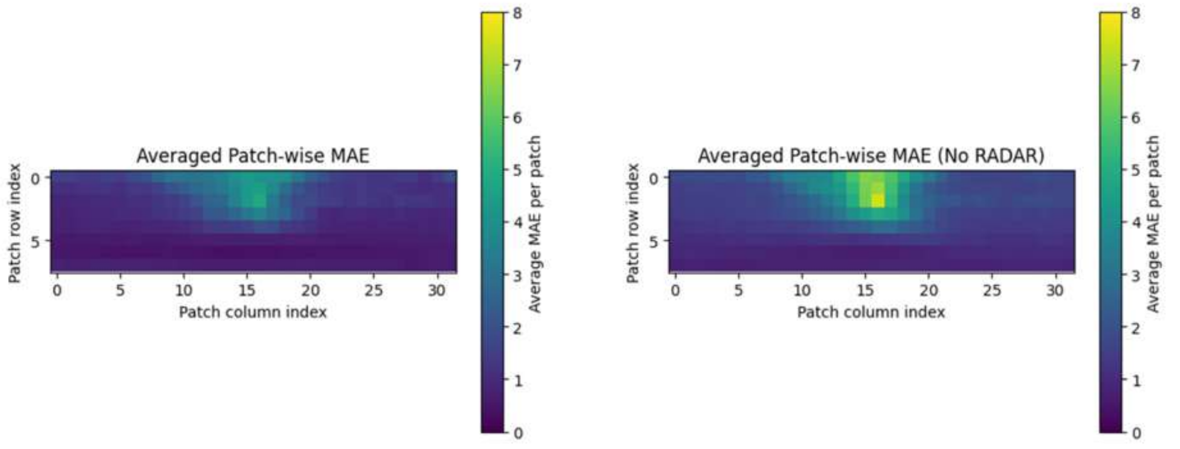


Figure 26: Grid error analysis of patch-wise MAE over the image: left, the full model's averaged MAE per patch; right, the model's MAE distribution when RADAR input is removed. Elevated errors in the central region without RADAR indicate its strong contribution to depth accuracy

4.5 Qualitative Results

4.5.1 Metric Depth Estimation

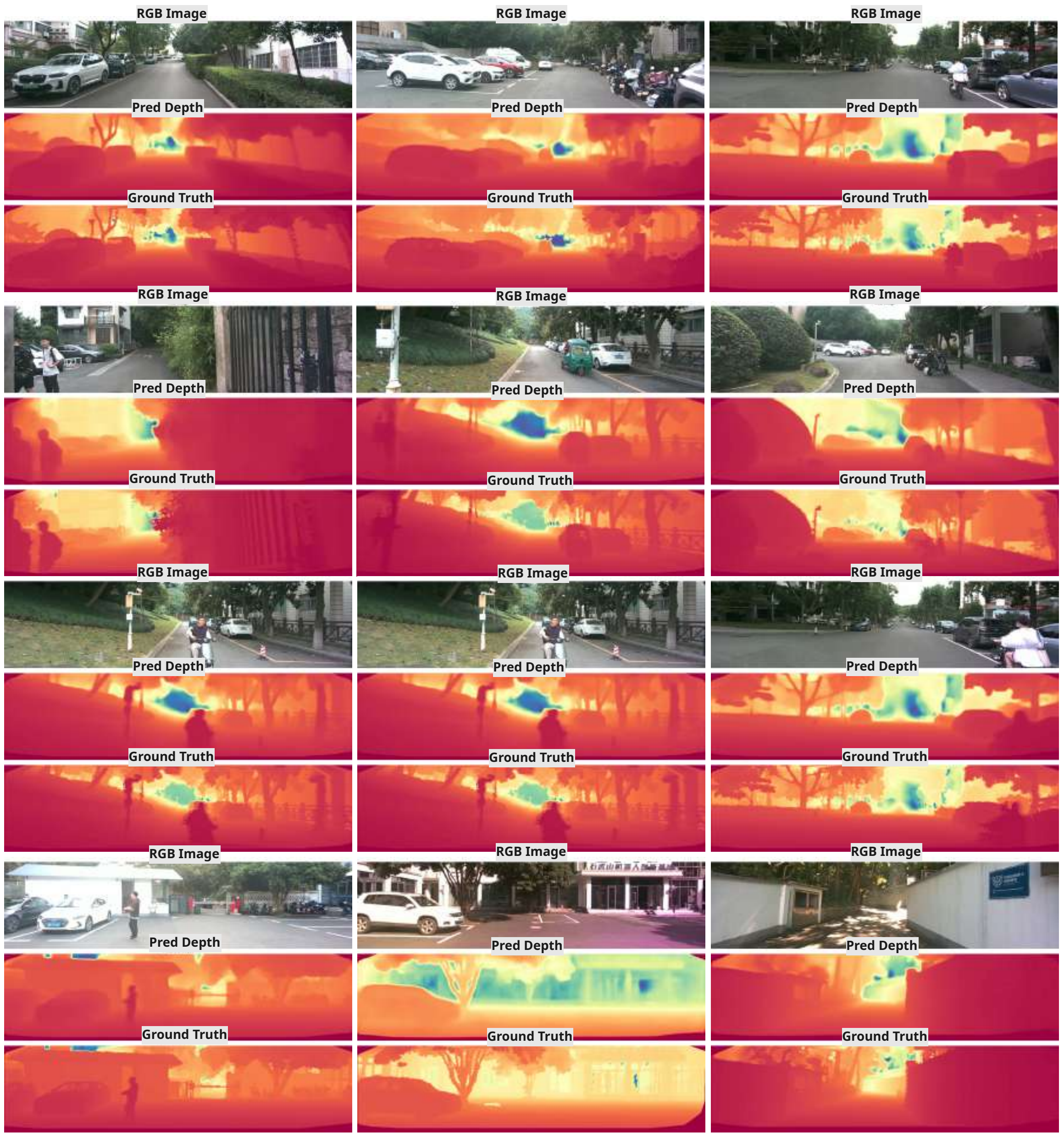


Figure 27: The figure represents the predictions of the depth estimation model for different scenarios. Top: RGB image, Middle: Metric dense depth prediction, Bottom: LiDAR ground truth.

4.5.2 2D Object Detection

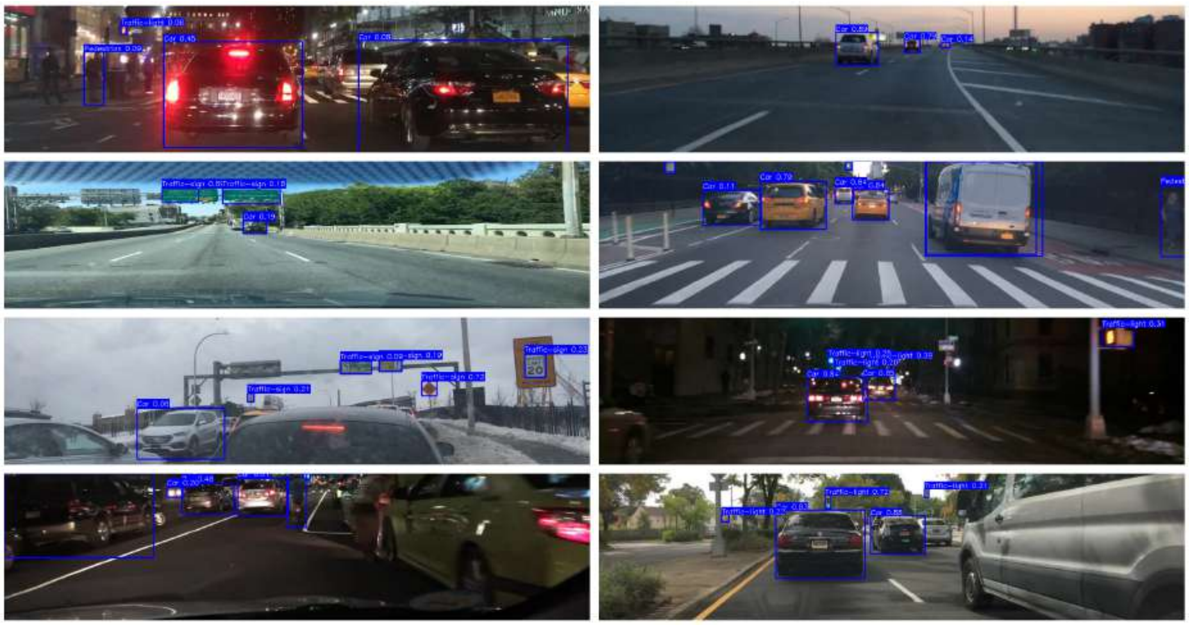


Figure 28: Multi-scene object detection results

The focus of this thesis is on estimating depth by fusing radar, camera, and monocular depth map data. Spatial information and localization are very important for autonomous driving scenarios. As shown in Figure 28, object detection is performed solely using the camera, without the integration of RADAR or monocular depth maps.

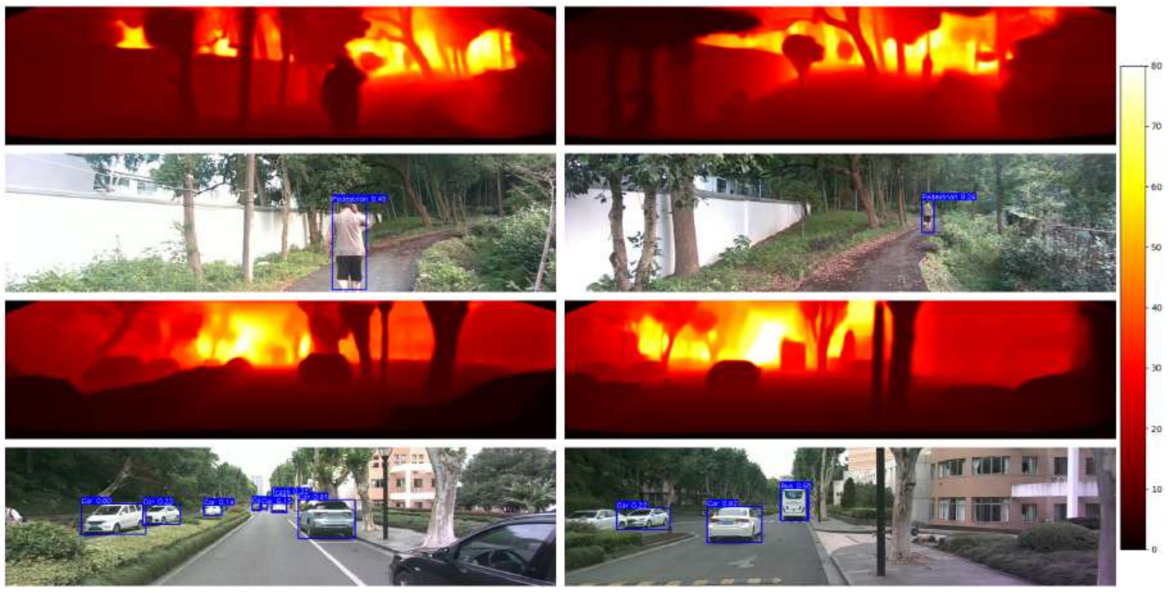


Figure 29: Multi-tasking model inferencing results :: Upper Row-Metric Depth Map; Bottom Row- Object detection results

As shown in Figure 29, the results were obtained by simultaneously performing inference on both tasks. This figure highlights the outcomes of a multitasking algorithm designed to process multiple tasks concurrently rather than in sequence. By executing inference on both tasks at the same time, the approach optimizes the use of computational resources, potentially taking advantage of overlapping data dependencies between tasks.

5 Conclusion

Depth estimation using radar and camera data presents significant challenges due to the inherent sparsity and ambiguity associated with radar measurements. To effectively address these limitations, this thesis proposed a novel depth estimation model designed to integrate multi-scale features from RGB images and radar data, further leveraging monocular relative depth information to enhance estimation accuracy.

Central to our method is the incorporation of the LoFTR module, specifically developed to learn cross-modal features effectively. Leveraging the global receptive field of the Vision Transformer (ViT), the LoFTR architecture achieves robust associations between radar measurements and corresponding pixels in RGB images. Additionally, by incorporating position-based contextual information derived from monocular depth estimation, the proposed model mitigates common radar-related errors, significantly improving overall depth accuracy.

The model was thoroughly evaluated on the ZDU-4DRadarCAM dataset. Notably, our approach achieved an improvement of 11.8% in MAE, confirming its effectiveness in handling the complexities inherent to radar-camera fusion. Moreover, our model consistently outperformed all benchmarked methods across various evaluation metrics.

Further analysis revealed that our model exhibits optimal performance up to distances of 60 meters, achieving an MAE of 1.13 meters. This demonstrates suitability for autonomous driving applications, which frequently require accurate depth measurements within this operational range. Additionally, the model achieves real-time inference speeds of 37 frames per second (FPS) without quantization, and 25 FPS when integrated with the monocular depth estimation component. These performance metrics highlight the substantial potential for real-time deployment, particularly after further optimization through quantization.

Additionally, an auxiliary object detection head was introduced to complement depth estimation by providing object localization capabilities. The object detection head was independently trained on the BDD100K dataset, with depth estimation module weights frozen and without the benefit of radar and monocular depth information. Although primarily evaluated visually—due to the thesis's focus on depth estimation—the preliminary results indicated promising localization accuracy. However, the unavailability of radar and monocular depth maps during object detection training introduced performance limitations, which represents a valuable opportunity for future work.

In summary, this thesis presents a novel RADAR-Camera fusion method that significantly advances depth estimation accuracy and demonstrates robust performance suitable for real-time autonomous applications. The findings and methods outlined not only improve upon existing methodologies but also suggest promising directions for continued research, particularly in enhancing and integrating complementary detection capabilities.

6 Future Scope

The next phase of research should concentrate on enriching supervision and improving efficiency so that metric depth estimation and object detection can operate robustly in real-world settings. Recent use of foundation models such as Prompt-Depth-Anything [24] encapsulate powerful geometric priors. Conditioning the LoFTR feature extractor on prompts to guide the foundation model for accurate depth estimation by leveraging the weights learned by foundation model.

Accurate metric depth depends on reliable ground truth, yet dense, well-aligned LiDAR scans remain expensive to acquire. Modern simulators and game engines like CARLA, AirSim, or Unreal Engine, for example—can now generate photorealistic scenes together with perfect depth and point-cloud annotations. A systematic program of synthetic-to-real training that combines domain randomization, mixed-reality fine-tuning, and curriculum learning could close the supervision gap when real LiDAR data are scarce.

Multitask learning studies report that adding a semantic-segmentation branch stabilizes shared representations and often boosts the accuracy of companion tasks. Incorporating segmentation as an auxiliary loss or additional output therefore offers a promising route to faster convergence and greater robustness in cluttered scenes.

Real-time performance matters as much as accuracy when models are deployed on edge devices. Layer-wise structured pruning, post-training integer quantization, quantization-aware training should be explored to reach sub-30-millisecond inference on embedded hardware without sacrificing precision.

In summary,the next research stage aims to make the multitask pipeline both smarter and faster: leverage foundation-model priors to guide depth features, fill LiDAR gaps with realistic synthetic data, add a segmentation branch to steady shared representations, and compress the network with pruning and quantization for real-time performance on edge devices.

References

[1] Gwangbin Bae, Ignas Budvytis, and Roberto Cipolla. Multi-View Depth Estimation by Fusing Single-View Depth Probability with Multi-View Geometry. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2832–2841, New Orleans, LA, USA, June 2022. IEEE.

[2] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. AdaBins: Depth Estimation using Adaptive Bins. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4008–4017, June 2021. arXiv:2011.14141 [cs].

[3] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. LocalBins: Improving Depth Estimation by Learning Local Distributions, March 2022. arXiv:2203.15132 [cs].

[4] Shariq Farooq Bhat, Reiner Birkl, Diana Wofk, Peter Wonka, and Matthias Müller. ZoeDepth: Zero-shot Transfer by Combining Relative and Metric Depth, February 2023. arXiv:2302.12288 [cs].

[5] Prayash Bohra. Understanding IoU, Precision, Recall, and mAP for Object Detection Models, 2025.

[6] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, and Tom Henighan. Language Models are Few-Shot Learners.

[7] Wei Chen, Yan Li, Zijian Tian, and Fan Zhang. 2D and 3D object detection algorithms from images: A Survey. *Array*, 19:100305, September 2023.

[8] Weifeng Chen, Zhao Fu, Dawei Yang, and Jia Deng. Single-Image Depth Perception in the Wild.

[9] Michael Crawshaw. Multi-Task Learning with Deep Neural Networks: A Survey, September 2020. arXiv:2009.09796 [cs].

[10] Wizard Data Science. Evaluation Metrics for Machine Learning or Data Models, 2022.

[11] Dattuthunuguntla. Artificial Intelligence Vs Machine Learning Vs Deep Learning.

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, June 2021. arXiv:2010.11929 [cs].

[13] David Eigen, Christian Puhrsch, and Rob Fergus. Depth Map Prediction from a Single Image using a Multi-Scale Deep Network, June 2014. arXiv:1406.2283 [cs].

[14] Ravi Garg, Vijay Kumar BG, Gustavo Carneiro, and Ian Reid. Unsupervised CNN for Single View Depth Estimation: Geometry to the Rescue, July 2016. arXiv:1603.04992 [cs].

[15] Stefano Gasperini, Patrick Koch, Vinzenz Dallabetta, Nassir Navab, Benjamin Busam, and Federico Tombari. R4Dyn: Exploring Radar for Self-Supervised Monocular Depth Estimation of Dynamic Scenes. In *2021 International Conference on 3D Vision (3DV)*, pages 751–760, December 2021. arXiv:2108.04814 [cs].

[16] Clement Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised Monocular Depth Estimation with Left-Right Consistency. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6602–6611, Honolulu, HI, July 2017. IEEE.

[17] Ariel Gordon, Hanhan Li, Rico Jonschkowski, and Anelia Angelova. Depth from Videos in the Wild: Unsupervised Monocular Depth Learning from Unknown Cameras, April 2019. arXiv:1904.04998 [cs].

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition, December 2015. arXiv:1512.03385 [cs].

[19] Mu Hu, Wei Yin, Chi Zhang, Zhipeng Cai, Xiaoxiao Long, Hao Chen, Kaixuan Wang, Gang Yu, Chunhua Shen, and Shaojie Shen. Metric3D v2: A Versatile Monocular Geometric Foundation Model for Zero-shot Metric Depth and Surface Normal Estimation, March 2024. arXiv:2404.15506 [cs].

[20] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention, August 2020. arXiv:2006.16236 [cs, stat].

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, May 2017.

[22] Shuguang Li, Jiafu Yan, Haoran Chen, and Ke Zheng. Radar-Camera Fusion Network for Depth Estimation in Structured Driving Scenes. 2023.

[23] Xiang Li, Wenhai Wang, Lijun Wu, Shuo Chen, Xiaolin Hu, Jun Li, Jinhui Tang, and Jian Yang. Generalized Focal Loss: Learning Qualified and Distributed Bounding Boxes for Dense Object Detection, June 2020. arXiv:2006.04388 [cs].

[24] Haotong Lin, Sida Peng, Jingxiao Chen, Songyou Peng, Jiaming Sun, Minghuan Liu, Jiashi Feng, Xiaowei Zhou, and Bingyi Kang. Prompting Depth Anything for 4K Resolution Accurate Metric Depth Estimation.

[25] Juan-Ting Lin, Dengxin Dai, and Luc Van Gool. Depth Estimation from Monocular Images and Sparse Radar Data, September 2020. arXiv:2010.00058 [cs].

[26] Chen-Chou Lo and Patrick Vandewalle. Depth Estimation from Monocular Images and Sparse radar using Deep Ordinal Regression Network. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 3343–3347, September 2021. arXiv:2107.07596 [cs, eess].

[27] Yunfei Long, Daniel Morris, Xiaoming Liu, Marcos Castro, Punarjay Chakravarty, and Praveen Narayanan. Radar-Camera Pixel Depth Association for Depth Completion, June 2021. arXiv:2106.02778 [cs].

[28] Abrar Muhammad Abdullah. Activation Function: Formulas, Explanation, Usage, Pros and Cons with its Types, 2023.

[29] Luigi Piccinelli, Yung-Hsu Yang, Christos Sakaridis, Mattia Segu, Siyuan Li, Luc Van Gool, and Fisher Yu. UniDepth: Universal Monocular Metric Depth Estimation, March 2024. arXiv:2403.18913 [cs].

[30] Rukshan Pramoditha. Two or More Hidden Layers (Deep) Neural Network Architecture.

[31] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision Transformers for Dense Prediction, March 2021. arXiv:2103.13413 [cs].

[32] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Midas, August 2020. arXiv:1907.01341 [cs].

[33] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 658–666, Long Beach, CA, USA, June 2019. IEEE.

[34] Florian Sauerbeck, Dan Halperin, Lukas Connert, and Johannes Betz. CamRaDepth: Semantic Guided Depth Estimation Using Monocular Camera and Sparse Radar for Automotive Perception. *IEEE Sensors Journal*, 23(22):28442–28453, November 2023.

[35] Kieran Saunders, George Vogiatzis, and Luis Manso. Self-supervised Monocular Depth Estimation: Let's Talk About The Weather, July 2023. arXiv:2307.08357 [cs].

[36] Sadique Adnan Siddiqui, Axel Vierling, and Karsten Berns. Multi-Modal Depth Estimation Using Convolutional Neural Networks, December 2020. arXiv:2012.09667 [cs].

[37] Akash Deep Singh, Yunhao Ba, Ankur Sarker, Howard Zhang, Achuta Kadambi, Stefano Soatto, Mani Srivastava, and Alex Wong. Depth Estimation from Camera Image and mmWave Radar Point Cloud. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9275–9285, Vancouver, BC, Canada, June 2023. IEEE.

[38] Pradeep Singh and Balasubramanian Raman. *Deep Learning Through the Prism of Tensors*, volume 162 of *Studies in Big Data*. Springer Nature Singapore, Singapore, 2024.

[39] Student, Department of Information Technology, Mahatma Gandhi Institute of Technology, Hyderabad and Brihat Kaleru. Prediction of on-Base Percentage (OBP) of Baseball Players Using Neural Networks. *INTERANTIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT*, 08(06):1–5, June 2024.

[40] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. LoFTR: Detector-Free Local Feature Matching with Transformers, April 2021. arXiv:2104.00680 [cs].

[41] SuperAnnoatte. What is image classification? Basics you need to know.

[42] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. Sparsity Invariant CNNs, August 2017. arXiv:1708.06500 [cs].

[43] Ultralytics. Ultralytics YOLO Docs.

[44] Rejin Varghese and Sambath M. YOLOv8: A Novel Object Detection Algorithm with Enhanced Performance and Robustness. In *2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS)*, pages 1–6, Chennai, India, April 2024. IEEE.

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need, August 2017. arXiv:1706.03762 [cs].

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need, August 2023. arXiv:1706.03762 [cs].

[47] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth Anything.

[48] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth Anything: Unleashing the Power of Large-Scale Unlabeled Data. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10371–10381, Seattle, WA, USA, June 2024. IEEE.

[49] Wei Yin, Chi Zhang, Hao Chen, Zhipeng Cai, Gang Yu, Kaixuan Wang, Xiaozhi Chen, and Chunhua Shen. Metric3D: Towards Zero-shot Metric 3D Prediction from A Single Image, July 2023. arXiv:2307.10984 [cs].

[50] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning, April 2020. arXiv:1805.04687 [cs].

[51] Chaoqiang Zhao, Qiyu Sun, Chongzhen Zhang, Yang Tang, and Feng Qian. Monocular Depth Estimation Based On Deep Learning: An Overview. *Science China Technological Sciences*, 63(9):1612–1627, September 2020. arXiv:2003.06620 [cs].

[52] Karthik Ziffer. Complete IoU, 2020. Accessed: 2025-06-28.