UNIVERSITY OF COPENHAGEN

# Investigating Monocular Depth Estimation Methods

Bachelor Thesis

Christian Shaffer-Lassen (mkv334)

January 13, 2025

Advisor: Stefan Sommer

Department of Computer Science, University of Copenhagen

**Abstract**

Monocular Depth Estimation (MDE) is a critical computer vision task of predicting the pixel-wise depth in a scene from two-dimensional images. Despite its potential, MDE faces challenges such as scale ambiguity, data scarcity, and computational trade-offs that impact real-world applicability. This thesis systematically investigates various MDE approaches, with a focus on encoder architectures, including Convolutional Neural Networks, Vision Transformers, and foundation models like DINOv2, while emphasizing the role of transfer learning. By analyzing accuracy, efficiency, and generalization, the study identifies strengths, weaknesses, and practical trade-offs between these architectures.

The results demonstrate that models initialized with pretrained weights consistently outperformed their non-pretrained counterparts, with significant improvements in threshold accuracy and RMSE, highlighting the importance of leveraging transfer learning. The DINOv2 foundation model achieved the best performance across all metrics, showcasing its robustness for downstream tasks. A custom lightweight encoder architecture was proposed, but achieved underwhelming results, due to lack of depth and design choices. To address the limited ground truth depth data, a pseudo-labeling experiment using DepthAnythingV2 revealed qualitative improvements in domain generalization, though quantitative results were underwhelming. Ablation studies underscored the critical role of skip connections for sharp edge predictions and the trade-offs between encoder depth and computational efficiency. Future directions include optimizing lightweight designs and scaling pseudo-labeling to improve performance in diverse scenarios.[1]

[1] Models and training code available at: https://github.com/Shaffer-Lassen/bachelor

# Contents

**Contents** ii

**1 Introduction** 1

**2 Background** 4
2.1 Depth Estimation . . . 4
2.1.1 Traditional Depth Estimation . . . 4
2.1.2 Monocular Depth Estimation . . . 5
2.2 Convolutional Neural Networks (CNN's) . . . 5
2.2.1 Convolutional Layers . . . 5
2.2.2 Activation functions . . . 6
2.2.3 Pooling Layers, Batch Normalization . . . 7
2.3 Vision Transformers . . . 7
2.3.1 Patch Embeddings . . . 8
2.3.2 Self-Attention Mechanism, Multi-Head Attention & Transformer Blocks . . . 8
2.4 Related work . . . 9
2.4.1 CNN-based Approaches . . . 9
2.4.2 Transformer-based Approaches . . . 10
2.4.3 Foundation Models for Depth . . . 10
2.4.4 Self- and Semi-supervised Training . . . 10
2.5 Ongoing Challenges in Monocular Depth Estimation . . . 11
2.6 Motivation . . . 11

**3 Methodology** 13
3.1 Architectures in this work . . . 13
3.2 CNN as Encoder . . . 14
3.2.1 ResNet . . . 14
3.2.2 DenseNet . . . 16
3.3 Transformer as Encoder (Swin-Transformer) . . . 17

3.4 Foundation Model as Encoder (DINOv2) . . . . . 18
3.5 Custom Method - Lightweight CNN with Spatial Attention) . 19
3.6 Decoder . . . . . 20
3.7 Data and Training . . . . . 21
3.8 Evaluation . . . . . 22

**4 Experiments** **24**
4.1 Experimental Setup . . . . . 24
4.2 Architectural Experiment . . . . . 25
4.3 Ablation Study . . . . . 26
4.4 Pseudo-Label Experiment . . . . . 27
4.5 Reproducibility . . . . . 28

**5 Results & Discussion** **29**

**6 Conclusion & Future Work** **36**

**Bibliography** **38**

**A Ablation Study** **43**
A.1 Encoder Depth . . . . . 43
A.2 Skip Connections . . . . . 44
A.3 Loss Functions . . . . . 44
A.4 Decoder Architecture. . . . . 45

**B Pseudo Labels** **46**

Chapter 1

# Introduction

When analyzing images, a central objective is to recognize how objects are arranged in space; particularly, how far they are from the camera or from each other [33, 4]. This corresponds to a spatial understanding, so to speak, of the images, also referred to as *depth estimation*. The task of depth estimation is important in many computer vision applications, such as autonomous navigation [12], robotics [39], and augmented reality [3, 13]. Information about the depth of a scene allows systems to perceive three-dimensional relationships from inherently two-dimensional images, enabling tasks like obstacle avoidance, object manipulation, and human–machine interaction [5].

Traditional methods for extracting depth information rely on specialized hardware, including stereo camera setups or laser-based sensors (LiDAR) [12], which are expensive and difficult to work with. Consequently, deployment of these systems and large-scale data collection becomes impractical. Monocular Depth Estimation (MDE) addresses this challenge by predicting depth from a single 2-dimensional RGB image [13], eliminating the need for multi-camera setups or active sensors at inference time. Mathematically, one can view MDE as the problem of learning a mapping

$$f : \mathbb{R}^{3\times H\times W} \to \mathbb{R}^{H\times W} \tag{1.1}$$

where $\mathbf{I} \in \mathbb{R}^{3\times H\times W}$ is the input image, and $f(\mathbf{I})$ is a dense depth map assigning a depth value to each pixel [33].

Despite remarkable progress, MDE remains a difficult problem [9, 34]. Images recorded by a single camera compress the 3D scene into a 2D projection, discarding explicit information about scale, which arguably makes the task ill-posed [24, 4]. Consequently, MDE forces models to rely on learned priors or contextual cues, such as textures, lighting conditions, and familiarity with objects and shapes, to approximate distances [11, 16]. Moreover, the shortage of extensive ground-truth datasets with accurate pixel-level depth annotations

imposes additional challenges. Producing dense ground truth depth typically involves scanning devices or complex stereo calibration pipelines, which may be infeasible on a large scale [39, 49]. As a result, MDE models can suffer from limited training data, making them vulnerable to domain overfitting and diminished generalization when encountering previously unseen contexts [53, 14].

Early research in MDE embraced deep learning paradigms, predominantly using convolutional neural networks (CNNs) [22, 26, 9] to learn robust representations from raw image data. More recently, vision transformers [8, 27] and large-scale foundation models [31, 6] have pushed performance boundaries, leveraging self-attention mechanisms, self-supervision, and massive pre-training on large scale datasets. However, transformer architectures are often memory-intensive and computationally expensive [45, 49]. Real-time applications or hardware-constrained devices require more efficient models without incurring substantial drops in accuracy [40, 18].

The objective of this thesis is to systematically investigate and evaluate different architectural approaches to MDE, including CNNs, Vision Transformers, and foundation models, as well as to cover the importance of transfer learning. We identify strengths and weaknesses in terms of different accuracy metrics, efficiency, and generalization capabilities. Additionally, the thesis aims to address the challenge of data scarcity in MDE, by evaluating the feasibility of using pseudo-labeled data to improve domain generalization.

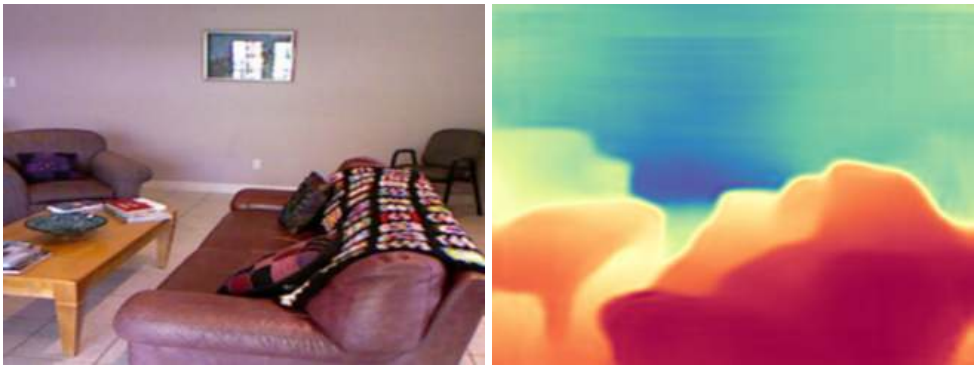

**Figure 1.1:** Illustration of monocular depth estimation, where a single RGB image (left) is mapped to a dense depth prediction (right). Prediction with the DINOv2 encoder.

In the following chapters, we first present the fundamental concepts of depth estimation and demonstrate why recovering three dimensional information from two-dimensional images is nontrivial. Chapter 2 formally defines the problem of depth estimation as well as motives the necessity of a continuous development in the field of MDE. It presents fundamental concepts of CNNs and Transformers and surveys the evolution of MDE approaches, beginning

with early CNN-based methods, transitioning into more modern transformer-based architectures, and lastly review the recent breakthroughs in computer vision foundation models. Chapter 3 outlines the architectures of the models employed in our experiments in detail. This chapter also highlights supporting methods relevant for training process, such as data pre-processing and augmentation. The experimental work is described in Chapter 4, which presents how the full range of experiments is conducted – from encoder architecture comparisons to ablation experiments on encoder depth, skip connections, and loss functions. Additionally, this chapter describes our investigation into leveraging pseudo-labels from DepthAnythingV2 to annotate unlabeled COCO images and train a student model. Chapter 5 presents both quantitative and qualitative findings from experiments, assesses whether they align with expectations, and identifies limitations of conducted experiments. Lastly, the thesis concludes the main contribution in Chapter 6, references the initial research questions, and outlines promising directions for further exploration of efficient and accurate MDE models.

Chapter 2

# Background

To fully grasp the methodology adopted in this thesis, we start by presenting a background chapter. The chapter covers the fundamentals of depth estimation, as well as formally define monocular depth estimation. Furthermore, the chapter introduces the fundamentals of CNNs and Vision Transformers, two widely adopted methods for solving MDE. Finally, we survey different existing approaches to solving the MDE task in Section 2.4.

## 2.1 Depth Estimation

Depth estimation is the task of predicting how far every given object in a scene is from the camera and other objects in the same scene. Monocular depth estimation recovers the three-dimensional scene of objects from a single RGB image [33, 4]. Unlike methods that rely on multiple cameras or specialized sensors, monocular approaches estimate how far each pixel is from the camera using visual cues in a single view [13] learning via deep learning methods. Visual cues could be various properties connected to depth in an image, such as parallel lines that converge towards a vanishing point or objects that does not vary a lot in size.

### 2.1.1 Traditional Depth Estimation

Traditional solutions avoid scale ambiguity by measuring 3D information more directly and computing a geometric estimate [23]. Stereo vision uses two horizontally separated cameras, inspired by the human eyes, to measure the *disparity* between each pair of matched pixels [12], leading to a depth estimate. Another technique uses LiDAR sensors that emit laser pulses and time their return, constructing accurate 3D maps [46]. Both approaches achieve high depth accuracy but require specialized hardware and calibration.

MDE removes these hardware constraints and infers depth from a single RGB image [13, 4, 9]. This is advantageous because, unlike traditional depth sensing, MDE can theoretically be applied in many more contexts.

### 2.1.2 Monocular Depth Estimation

Formally, monocular depth estimation can be framed as learning a function

$$f : \mathbb{R}^{3\times H\times W} \to \mathbb{R}^{H\times W}, \tag{2.1}$$

which predicts a pixel-wise depth map $\mathbf{D} \in \mathbb{R}^{H\times W}$ from a single input image $\mathbf{I} \in \mathbb{R}^{3\times H\times W}$.

Unlike stereo-based depth estimation methods, MDE relies solely on learning abstract features that capture pixel-level patterns relevant for depth. Learning these abstract features in the diversity of real world settings requires a huge amount of training data [32]. However, obtaining quality depth annotations is expensive and can be incomplete or noisy, especially in reflective regions [11, 14]. Therefore, ground truth depth data are limited and approaches often rely on supporting techniques to produce quality depth maps. Models trained on mostly indoor data often struggle in outdoor environments with different scales and lighting [53]. Even if the relative depth ordering of a prediction is correct, the global scale can be off, highlighting the complexity of MDE [9]. The scale ambiguity has led to several approaches for solving MDE for relative depth rather than absolute depth [35, 49].

As earlier stated, most successful approaches to the MDE problem have involved deep learning architectures such as CNNs and vision transformers. In the following sections, we cover the fundamentals of such networks and the mechanisms that make them feasible for MDE.

## 2.2 Convolutional Neural Networks (CNN's)

A convolutional neural network (CNN) is a class of deep learning models designed to process data with grid-like structures, such as images. This section covers the main methodology and concept of CNN's in order to motivate the use on computer vision tasks like MDE.

### 2.2.1 Convolutional Layers

The main building blocks of CNN's are the convolutional layers. Convolutional layers apply a filter that slides across the input data to learn the relational structures and patterns of an image. Formally, given an input tensor $\mathbf{x} \in \mathbb{R}^{C_{\text{in}}\times H\times W}$ and a filter $\mathbf{W} \in \mathbb{R}^{C_{\text{out}}\times C_{\text{in}}\times K\times K}$, the convolutional

operation outputs [15]:

$$\mathbf{y} = \sum_{c'=1}^{C_{\text{in}}} \sum_{k_1=1}^{K} \sum_{k_2=1}^{K} \mathbf{W}_{c,c',k_1,k_2} \cdot \mathbf{x}_{c',h+k_1,w+k_2}, \tag{2.2}$$

where $\mathbf{y} \in \mathbb{R}^{C_{\text{out}} \times H_{\text{out}} \times W_{\text{out}}}$ is the output feature map. Each filter applied captures features within their receptive field of $K \times K$ pixels. One of the key features of convolutional layers is the spatial weight sharing, since the same filter is applied to all of the input data [15]. In comparison, fully connected layers in neural networks have a unique weight for every connection. Suppose that an input image **I** with dimensions $32 \times 32 \times 3$ (**H** $\times$ **W** $\times$ **Channels**) is passed through a convolutional layer and a fully connected layer for comparison. Number of output channels is set to 64.

A standard convolutional $3 \times 3$ filter is applied. The total number of parameters is then:

$$P_{\text{conv}} = C_{\text{out}} \cdot C_{\text{in}} \cdot K \cdot K + C_{\text{out}} = 64 \cdot 3 \cdot 3 \cdot 3 + 64 = 1,792.$$

For the fully connected layer, with an output size of 64 neurons, the total number of parameters is:

$$P_{\text{fc}} = C_{\text{in}} \cdot C_{\text{out}} + C_{\text{out}} = 3,072 \cdot 64 + 64 = 196,608.$$

This comparison illustrates that the fully connected layer requires significantly more parameters than the convolutional layer due to the absence of weight sharing. However, fully connected layers are often implemented at the end of a CNN in order to flatten feature maps and produce predictions at a meaningful channel dimension.

### 2.2.2 Activation functions

Activation functions introduce non-linearity to neural networks by only activating pre-defined outputs, hence the name [15]. Activation function are usually applied on the output one or more layers to guide the gradient to focus on correct predictions. This work will use two different varitions of the ReLU activation function, which are often seen in MDE networks. ReLU is defined as [36]:

$$\text{ReLU}(x) = (x)^+ = \max(0, x) \tag{2.3}$$

The simple definition os the ReLu activation function describes that only positive inputs are activated throughout the network. Negative prediction will always output 0. This loss function is widely used in MDE models, as the objective is to estimate the Euclidean distance between the camera and each pixel in the input image. A variation LeakyReLU is defined as [25]:

$$\text{LeakyReLU}(x) = \begin{cases} x, & \text{if } x \geq 0, \\ \text{negative slope} \cdot x, & \text{otherwise.} \end{cases} \tag{2.4}$$

LeakyReLU takes a negative slope (float) as input an applies it to all $x < 0$. This addresses a common issue with the ReLU function, where many neurons may become inactive when their outputs are set to zero. This can hinder the optimizer's ability to effectively update weights during backpropagation.

Finally, we introduce the sigmoid activation function, which is used sparingly in this thesis. The sigmoid function is defined as [38]:

$$\text{Sigmoid}(x) = \sigma(x) = \frac{1}{1 + \exp(-x)} \tag{2.5}$$

The sigmoid activation function normalizes all outputs to a value between $[0; 1]$. This is particularly useful in MDE models that are trained on inverse normalized depth maps for relative depth estimation as done as a part of the experimental work in this thesis.

### 2.2.3 Pooling Layers, Batch Normalization

*Pooling layers* are a class of layers responsible for reducing the spatial dimensions of the feature maps of the network, while preserving relevant information. The pooling operations used in this work include maximum pooling or average pooling in CNN networks. Max-pooling takes some window size $K \times K$ and a stride $s$, reducing a window of pixels to its maximum value, while average-pooling computes the mean value within the window.

For example, applying with a $2 \times 2$ window size and a stride of two, will reduce spatial dimensions by 2, as it reduces a 2×2 pixel window to a single. *Batch Normalization* is a technique widely used in deep learning that stabilizes and accelerates deep neural network training [20, 15]. The process involves a normalization, where the mean and variance of are computed across the batch. This is followed by rescaling and shifting, to preserve the layer's ability to learn complex features.

## 2.3 Vision Transformers

CNN based architectures have in many years been the standard for solving the MDE task. However, more recently, researchers have increasingly turned to Vision Transformers as a key architectural component in their solutions. ViT-based MDE models have actually achieved better performance results than CNNs primarily due to their less restrictive receptive field [33]. Despite their advantages, ViTs have limitations. Their larger number of learnable parameters makes them more computationally expensive compared to CNN [33]. To address this challenge, researchers have developed hybrid approaches that leverage both architectures' strengths. These hybrid models typically combine a ViT encoder, which excels at capturing global context,

with a CNN decoder [34, 47, 51]. This design pattern allows the model to benefit from the ViT's broad receptive field while maintaining computational efficiency through CNN components. Given the growing significance of Vision Transformers in monocular depth estimation, this section explores their core principles and mechanisms.

### 2.3.1 Patch Embeddings

The first step of any vision transformer is to decompose the image $I \in \mathbb{R}^{C\times H\times W}$ into a sequence of patches $x_p \in \mathbb{R}^{N\times(P^2\times C)}$, where P is the patch size and N is the number of patches[8]. Each patch is then projected into a D-dimensional space using a learnable projection matrix $E$:

$$E \in \mathbb{R}^{(P^2\times C)\times D} \tag{2.6}$$

The resulting patch embeddings from this proccess can be described as:

$$z_0 = [x_p E; x_p E; ...; x_p E] + E_{pos}, \tag{2.7}$$

where $E_{pos} \in \mathbb{R}^{N\times D}$ learnable positional matrix, both for absolute and relative positions of patches, ; is the concatenation along the sequence dimension and $z_0$ is the initial transformer input.

### 2.3.2 Self-Attention Mechanism, Multi-Head Attention & Transformer Blocks

The self-attention mechanisms are the innovation that enables ViT's to capture global dependencies. First patch embeddings is transformed into three seperate vectors. Formally, each input vector $z$ is transformed into:

$$Q = zW_q, \quad K = zW_k, \quad V = zW_v, \tag{2.8}$$

where $Q$ represent queries, $K$ represent keys and $V$ represents values. From these vectors, an attention score is computed [42]:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{2.9}$$

where $QK$ is the dot product between $K$ and $Q$, softmax ensures that attention scores sum to 1, and the multiplication with vector $V$ computes the weighted sum of values.

Multi-head attention is an attention mechanism that is repeated several times in parallel with different learnable projections, creating multiple attention heads. This ability allows the model to capture different relational features simultaneously [42].

Analogous to convolutional blocks in CNN, transformer blocks serve as the primary processing component in Vision Transformers [8]. Each transformer block usually consists of a multi-head attention layer followed by a multi-layer perceptron layer [30]. Layer normalization before each component and residual connections between them are a common accepted practice.

Despite the constraints of higher computational cost, vision transformers demonstrate highly usable abilities of learning relational features through attention mechanisms. Their ability to capture global dependencies from the first forward, due to their global receptive field, is essential for its use in MDE [34, 47].

## 2.4 Related work

Monocular depth estimation has attracted considerable interest in computer vision, leading to various strategies for turning a single RGB image into a dense depth map [33]. Early methods relied heavily on *Convolutional Neural Networks* (CNNs) to learn how image features relate to depth [9, 24]. More recent approaches incorporate *transformers* to capture broader context [34, 47]. Another emerging path involves large-scale *foundation models*, such as DINOv2, which leverage extensive pretraining on unlabeled data [31]. Each category addresses different aspects of the depth estimation challenge but also faces notable drawbacks, especially with regard to domain shifts, data limitations, and the inherent ambiguity of predicting distance from a lone view.

### 2.4.1 CNN-based Approaches

The general pipeline of MDE models include a *encoder–decoder* design. This is a design, where the data is first encoded into some feature representation and then decoded into to a depth map. This is also the case for CNN-based approaches. The encoder applies layers of convolutions and nonlinear activations, gradually reducing the resolution of the image while learning higher-level features. If we denote the input image by $\mathbf{I} \in \mathbb{R}^{3\times H\times W}$, the encoder produces a series of feature maps at various spatial scales, capturing information from edges up to more abstract shapes. The decoder then upsamples these feature maps back to the original spatial dimensions, combining them with intermediate outputs (often called *skip connections*) so that important details lost during downsampling are recovered for the final depth map $\hat{\mathbf{D}} \in \mathbb{R}^{H\times W}$ [9, 17, 24].

Architectures such as ResNet [17] and DenseNet [19] are widely used as CNN backbones, while more compact variants exist for faster inference on embedded hardware [18, 40]. By backbone, we mean pretrained networks acting as encoder of a model. Some CNN-based models emphasize *scale-invariant* or

*ordinal* loss functions to handle the difficulty of predicting absolute depth. Others include multi-scale feature fusion, allowing the network to combine fine details from early layers with broader context from deeper layers [2, 10]. These designs achieve strong performance on datasets similar to the training distribution but can struggle when scenes differ significantly in lighting or geometry [53].

### 2.4.2 Transformer-based Approaches

Transformers address one of the main weaknesses of CNNs: limited ability to model long-range relationships in a single pass. Rather than convolving over local regions, transformers use self-attention across the entire image to decide which patches of pixels should focus on one another [8]. An image is split into patches, each turned into a vector token. Multiple tokens form a sequence, and the model learns attention weights indicating how strongly each token should influence every other token. This approach can help the network infer depth more accurately in scenes with large or widely separated objects [52, 27].

Pure transformers sometimes require massive memory and computation when dealing with high-resolution images [34]. To address this, researchers explore hybrid CNN–transformer architectures, which rely on convolutional blocks for the early layers and transformer blocks to capture global context [51, 41]. Even with these optimizations, transformer-based models often pose practical challenges in resource contrained settings.

### 2.4.3 Foundation Models for Depth

Foundation models like DINOv2 [31] are pretrained on large-scale image collections without specialized depth annotations [6]. They learn general visual features from very large and diverse data sources that can later be adapted for depth prediction or other downstream tasks. One might take a pretrained backbone that was trained on millions of unlabeled images, then attach a task-specific head to produce a depth map.

Because foundation models are often large, they may be expensive to run in real time. They can also be difficult to fine-tune if the pretrained objective differs too strongly from the depth estimation goal. However, in many cases, they transfer well across computer vision domains, handling indoor and outdoor scenes more robustly than models trained solely on a single dataset [50].

### 2.4.4 Self- and Semi-supervised Training

Both self- and semi-supervised MDE has emerged as a significant research field to address the reliance on large-scale datasets with ground truth depth

maps to improve generalization [13] and the data scarcity. In deep learning, self-supervised learning is the training of a model completely without labeled data. Self-supervised methods that estimate depth labels with photometric consistency between left and right image pairs [13] showed how the reliance on labeled data could be minimized and were later followed by approaches that use sequential frames from monocular video to learning.

Semi-supervised learning typically combines traditional supervised learning, with methods such as pseudo-labeling [48] to improve robustness in various domains. By pseudo-labeling, we mean training teacher model to annotate unlabeled data with predictions to improve the diversity of e.g. real work scenes. The annotated data is then used to train a student model. Synthetic data has also proven effective in making smoother predictions, as the labeled data quality is extremely high. One recent approach trains a teacher model solely on synthetic data, followed by a large-scale pseudo-labeling process to incorporate the complexity of real world images [49].

## 2.5 Ongoing Challenges in Monocular Depth Estimation

Regardless of architecture, the monocular depth estimation must deal with *scale ambiguity*, since the image alone does not reveal how large or distant objects are [9, 3]. Although networks often learn good relative ordering, predicting which objects are in front or behind, the absolute scale can be off by a constant factor in novel settings. Data availability also remains an obstacle. Collecting ground-truth depth typically requires stereo rigs, LiDAR sensors, or other hardware, making comprehensive, accurate datasets difficult and expensive to obtain [39, 12, 35]. Domain gaps between indoor and outdoor scenes, or between synthetic and real imagery, can degrade performance unless models are carefully adapted or regularized [53].

Transformer-based or foundation models often handle domain shifts better, but can be large and slow. CNN-based methods run efficiently on smaller devices but may overfit to specific scene types and fail to capture global context well. Thus, current research seeks to combine ideas from each branch, developing solutions that preserve much of the flexibility and robustness of high-capacity architectures while remaining deployable on resource-limited platforms [45, 40]. These efforts continue to shape the evolution of monocular depth estimation, with the aim of finding solutions that are accurate on absolute depth and practical in real-world contexts.

## 2.6 Motivation

Despite these constraints, single view depth estimation remains crucial in scenarios where multi-camera setups or specialized sensors are infeasible

[45, 5]. Its reduced hardware demands make it suitable for mobile devices or devices limited in resources, though purely lightweight models underperform more computationally intensive approaches. Current research aims to fuse the strengths of high-capacity architectures with efficiency-focused designs [40, 18], seeking an optimal balance between precision and practicality. The motivation of this thesis is to map the impact of different encoder architectures and other network components, in order to make better design choices, when building an MDE model with the accuracy-efficiency trade-off in mind. Furthermore, the data scarcity described motivates the search for complementary techniques, such as leveraging pseudo-labeled data, if feasible.

Chapter 3

# Methodology

This chapter outlines the methodology adopted in this work, focusing on different encoder architectures and training practice. The primary focus of this work is to compare multiple architectures. The encoder architectures uses in this work includes CNNs, Vision Transformers, a foundation model and a lightweight custom design. The following sections describe these existing approaches in detail, as well as presents some of the foundational theory behind the approaches.

## 3.1 Architectures in this work

All models trained in the experimental work of this thesis take an RGB image $\mathbf{I} \in \mathbf{R}^{3\times224\times224}$ as input and produce a predicted depth map $\mathbf{D} \in \mathbf{R}^{1\times112\times112}$ - that is, a prediction of the input resolution $\frac{1}{2}$. This is done through a U-net architecture [37], where the encoder is responsible for downsampling the spatial resolution and extracting abstract features. The decoder receives the deepest layer (with smallest resolution) as input and progressively upsamples it. Skip connections are implemented between the encoder and the decoder, where the information is transferred directly, see Fig. 3.1. The main architectural focus of the thesis is the encoder, which is why the decoder remains the same in all models. The following sections describes the different architectures, that we adopt for our encoders – two CNN, a Transformer network, a foundation model, and a custom more lightweight network.

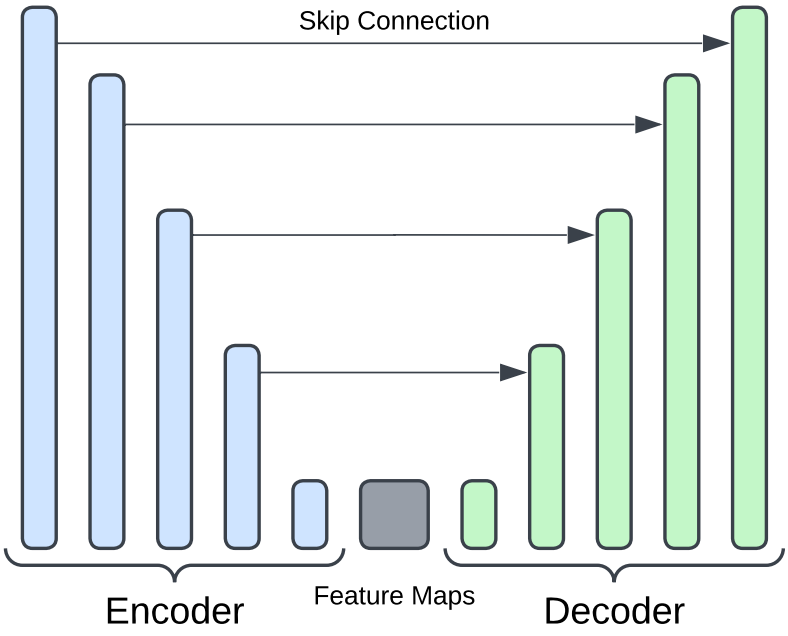


**Figure 3.1:** Illutstration of the U-net architecture, that all models of this thesis implements. Input is progressively downsampled by the encoder, while the decoder upsamples a final prediction to 1 channel of $\frac{1}{2}$ input resolution. Skip connections are adopted to preserve details.

## 3.2 CNN as Encoder

CNN is chosen as the primary baseline for its strong track record in monocular depth estimation [22, 17]. For experimental work, we have selected ResNet and DenseNet[17, 19], originally designed for classification, to act as our encoder networks in two different models inspired by [24, 2]. Both backbones will be trained with and without pretrained weight from ImageNet-1K [7]. The following describes the architecture of ResNet and DenseNet, respectively, and highlights what differentiates them.

### 3.2.1 ResNet

Resnet, introduced by He et al. [17], is a well-known CNN that tackles the vanishing gradient problem by implementing residual connections. The introduction of residual connections made it possible to train deeper network, without losing features learned early in the network. The core of ResNet is the bottleneck blocks, that consists of a 1x1 convolution that reduces

channel dimensions, a 3x3 convolution to capture features, and another 1x1 convolution that either restores or expands the channel dimensions. The residual connection is made between the input and output for each bottleneck block, adding the input to the output of the final convolution through a simple, yet effective element-wise addition. ResNet has several implementation with varying depth. The baseline implementation for our experimental work is ResNet50 [43]. ResNet50 consists of bottleneck blocks grouped into four different stages. Each stage contains 3, 4, 6, and 3 blocks, respectively. Before entering the bottleneck stages, the input resolution is downsampled to $\frac{1}{4}$ by an initial 7x7 convolution with a stride of 2 followed by batch normalization, ReLU activation, and a max-pooling operation. In this process, the channel size is increased from 3 (RGB) to 64.

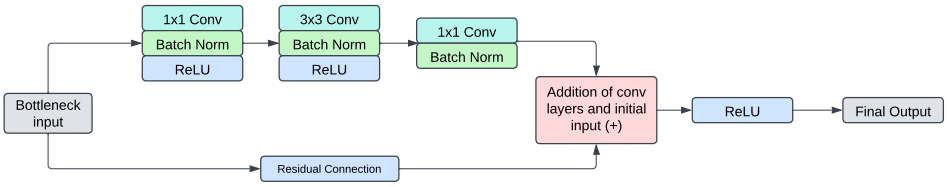


**Figure 3.2:** Illustration of a single Bottleneck from the ResNet CNN

As the network progresses through the bottleneck stages, the features maps will adopt channel dimensions of 256, 512, 1024, and 2048, while the spatial resolution of each feature map is $56 \times 56$, $28 \times 28$, $14 \times 14$, and $7 \times 7$. For our MDE model, output of each of the four stages is preserved in order to implement a U-net architecture with direct skip connection between encoder and decoder at matching spatial resolution. In the experimental work, one model will be initialized with pretrained ImageNet-1K classification weights [7], ensuring that the encoder already captures a wide set of visual patterns before fine-tuning the depth estimation task, while another model will be trained without pre-initialized weight.

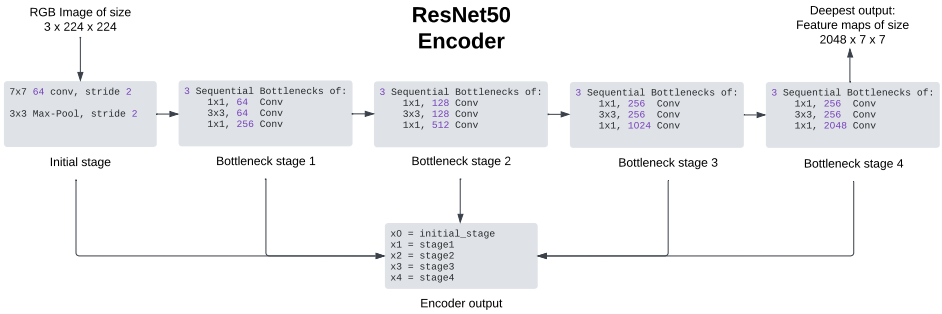


**Figure 3.3:** Illustration of how the ResNet50 is used as encoder in our MDE model. Outputs of all visualized stages are given as input to the decoder, that is the output of the encoder $x_0 - x_4$.

### 3.2.2 DenseNet

DenseNet, introduced by Huang et al. [19], is another approach that addresses vanishing gradients. The key innovation of DenseNet is how each layer, within a block of layers, is connected to all prior layers within the same block. The design fosters efficient feature preservation in deep network, making it feasible for complex tasks like depth estimation. As ResNet, the architectural design begins with an initial 7x7 convolution followed by a max-pooling operation to reduce spatial dimensions. The encoder implementation used in this work is DenseNet169, as done by Alhashim et al. [2]. DenseNet169 has four blocks with 6, 12, 32, and 32 densely connected layers, respectively. Between each of the dense blocks, there is a transition layer that reduces channel dimensionality through $1 \times 1$ convolutions and average pooling operations.

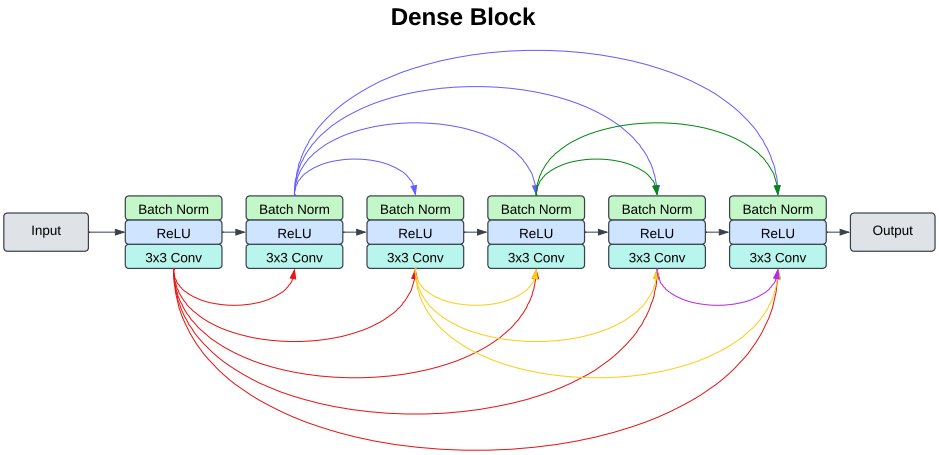


**Figure 3.4:** Illustration of a dense block, where every convolutional layer is connected to all prior within that same block.

As with ResNet, the channel dimensions of DenseNet are progressive upsampled, while the spatial dimensions are downsampled. DenseNet169 offers a powerful feature extractor that facilitates robustness and generalization through feature preservation. As with the ResNet50, one model will feature a DenseNet169 encoder trained from scratch, while another will be initialized with pretrained ImageNet-1K classification weights [7].

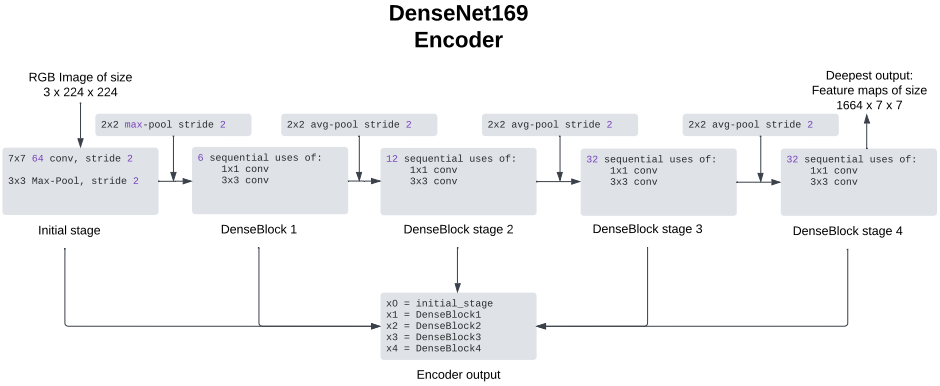


**Figure 3.5:** Illustration of DenseNet169 encoder used in our model

## 3.3 Transformer as Encoder (Swin-Transformer)

Another MDE approach that we implement for our MDE decoder is the transformers. Transformers replace localized convolutional kernals with

self-attention layers that assess relationships among all patches of the input image as described in Section **??**. We use a specific transformer, Swin, as encoder in this thesis as in [30].

The Swin-Transformer is a vision transformer developed to serve as a general-purpose backbone for computer vision tasks. The Swin transformer implements a hierarchical design that makes it more efficient than traditional transformers. This is done by computing with what they call Shifted Windows [30]. The key innovation of the approach with shifted windows is, that self-attention computation, which are in general heavy, are limited to non-overlapping local windows. This method makes the model flexible for adopting different sizes. The base model is the version used in the experimental work of this thesis.

At the initial stage, input images are divided into patches of size $4 \times 4$. With each pixel having 3 channel, a patch will contain 48 pixel values, that are now treated as a token. The patch splitting is followed by a linear embedding layer that projects the raw pixel values into a higher-dimensional space of size $C = 128$ for the base version. These two steps are what forms the tokens for the first stage.

As mentioned, the Swin Transformer is hierarichal. It progressively reduces spatial resolution while increasing feature dimensions (just like CNN's) through four stages. In the first stage, the spatial resolution is $\frac{H}{4} \times \frac{W}{4}$. Self-attention layers are applied based on windows in Swin blocks. The output token of the first stage had dimension $C = 128$. Stage 2 reduces spatial resolution to $\frac{H}{8} \times \frac{W}{8}$, stage 3 $\frac{H}{16} \times \frac{W}{16}$ and stage 4 to $\frac{H}{32} \times \frac{W}{32}$. In each stage, several blocks are applied to capture and transform features. The dimension $C$ is 128, 256, 512, and 1024, respectively, in each of the stages.

## 3.4 Foundation Model as Encoder (DINOv2)

In order to cover the different approaches of solving the MDE task, we have included a model, that leverages the foundation model, DINOv2 [31], as a frozen backbone. A frozen backbone is a pretrained encoder, where no weights are updated during training. By freezing the encoder, training time will be reduced heavily, as back-propagation is only necessary in the decoder. The DINOv2 is a computer vision foundation model capable of multiple tasks, such as image classification, object detection and depth estimation. It is built with a refined Vision Transformer [34] architecture as a backbone and is designed to extract general applicable features from images, that is, the *foundation* part.

DINOv2 is trained on more than 140 million images, a large-scale dataset to ensure robust generalization capabilities [31]. Aditionally, the training is

self-supervised. The model learns image representations by aligning patches within and across images without explicit labels [31, 6].

In order to use DINOv2 as our encoder, we process an image $\mathbf{I} \in \mathbf{R}^{3\times\mathbf{H}\times\mathbf{W}}$ through the model and output several hidden layers to obtain features of different spatial dimensions for skip connections and progressive upsampling similar to the models presented previously.

## 3.5 Custom Method - Lightweight CNN with Spatial Attention)

In our experimental work, we propose a CNN based encoder, that incorporates spatial attention on convolutions. The encoder design primarily follows a CNN-based hierarichial structure, while incooporating a light-weight multi-head spatial attention module. The core of the proposed model, is the ConvolutionalAttentionBlocks. The blocks adopt an initial convolutional block with a $3 \times 3$ convolution of stride 2, followed by batch normalization and ReLU activation, which is a generally accepted pattern. Subsequently, a multi-head attention module, inspired by transformers, computes the spatial attention of the feature maps from the convolution. The attention mechanism projects the input into query, key and value tensors, splits them into multiple heads, and finally computes the scaled dot product across spatial locations. The output is then projected back to the original feature space. Inspired by ResNet, the blocks also apply a residual connection between the initial input and the produced output.

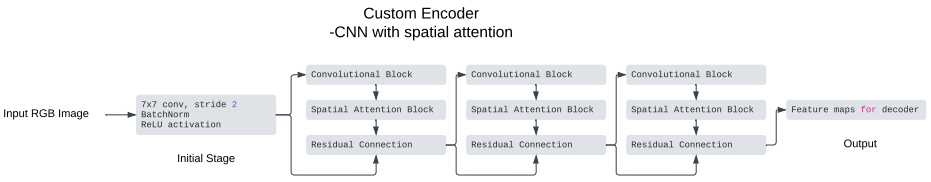


**Figure 3.6:** Illustration of custom encoder implementing a CNN with spatial attention module

This custom encoder has an initial stage and 3 convolutional attention stages, which is similar to the design of the other models, except that it only has four total stages. The decoder remains the same and will be detailed in the next section.

## 3.6 Decoder

We implement all models with similar decoders, placing emphasis on the encoder's role. This section describes two different approaches to decoder architectures that are quite straight forward, both implementing skip connections from the encoder.

**Interpolation.** The first decoder is adopted as default on all of our models and is inspired by the simple architecture of Alhashim et al. [2]. This decoder takes five multi-cale feature maps $x_0..x_4$, produced by the encoder, each one corresponding to different spatial resolutions, with $x_0$ being the largest and $x_4$ being the smallest. The decoder then upsampled the features from the deepest layer, $x_4$, to the next spatial resolution, such that it matches $x_4$. At this stage, the upsampled output is concatenated with $x_3$, that is the skip connection, as $x_3$ is directly inputted without further processing. This upsampling process is done all the way from $x_4$ to $x_0$, accompanied by a series of convolutional layers with batch normalization and ReLu activation functions. The final upsampling block passes the output through a convolutional layer $3 \times 3$, followed by a convolution $1 \times 1$ that reduces the channel dimension to 1, which is the pixel-wise depth prediction.

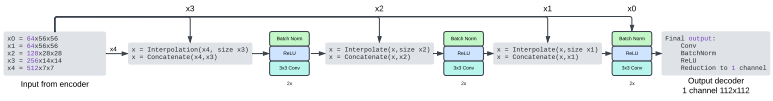


**Figure 3.7:** Illustration of decoder used in our models based on interpolation

**Transposed Convolution.** The second decoder utilizes transposed convolutions to progressively upsample the feature maps. It is inspired by fully convolutional networks. This decoder takes five feature map inputs, $x_0...x_4$ as our other decoder. Starting with the deepest feature map $x4$ a transposed convolution is applied to double the spatial resolution. We again deploy skip connections through concatenations as earlier stated. The skip connection is followed by a $3 \times 3$ convolution with batch normalization and ReLU activation to refine the combined features. This process is repeated until $\frac{1}{2}$ encoder input resolution is achieved. The main difference is the use of transposed convolutions for upsamplings instead of interpolation. The characteristics of each decoder is covered in 4.3.

## 3.7 Data and Training

Datasets relevant for supervised MDE training contain pairwise RGB images and ground truth depth maps. Most datasets for MDE are obtained by methods such as LiDAR or stereo vision camera setups. Some of the most prominent datasets in the research field of MDE are the NYUv2 and KITTI datasets:

- **NYUv2 [39]**: The NYUv2 data set consists of 1,449 densely labeled images and depth pairs, as well as a raw dataset of more than 400k images and depth maps. Depths and RGB images are captured as video sequences using the Microsoft's Kinect. The dataset contains only indoor scenes. The densely labeled dataset of 1,449 samples has been used to benchmark MDE model performance on indoor scenes for years. The raw part of the dataset has been used to train a wide range of MDE models throughout the years. Depth maps of the raw dataset often have missing depth values due to noise during depth measurements, such as shadows. The original resolution of NYUv2 data points are $640 \times 320$
- **KITTI [12]**: The KITTI dataset is a well-known and recognized dataset in the field of computer vision, including depth estimation tasks. The dataset is obtained by cars equipped with high-resolution cameras and LiDAR sensors to capture RGB images and depths simultaneously. Since it is captures from cars, the dataset consists of a diverse set of outdoor only scenes. The large scale of the dataset and quality ground truth depth maps have made it almost standard practice to use as benchmark.

NYUv2 and KITTI is the two most used dataset in MDE research. The two datasets have since served as the standard evaluation datasets for indoor and outdoor depth performance, respectively. Some recent studies have highlighted limitations in using these as evaluation when assessing the capabilities of modern depth estimation models [49]. However, NYUv2 serves as the main training and evaluation dataset of the experimental work carried out in the project.

The level of supervision, in training the presented MDE models, is strictly supervised. The training relies on image and ground truth depth pairs from the NYUv2 dataset, where a loss is computed between the predicted depth and ground truth [39]. Before feeding the RGB images and the corresponding ground truth to the models described in this thesis, both the images and ground truth depth maps are scaled to 256 on the shortest side, followed by a random crop of $224 \times 224$. During training a horizontal flip is performed on the input data with a probability of 0.5. This is the input resolution of all models trained in this thesis. The predicted depth maps are $\frac{1}{2}$ input

resolution $112 \times 112$.

In addition to the NYUv2 dataset, a subset of Microsoft's COCO [29] is also used for separate experiment. These images does not have depth labeled. We use a state-of-the-art model, DepthAnythingV2 to annotate images with pseudo-labels. These annotations are relative, meaning that depth labels are scaled between 0 and 1. Ultimately, models trained on these data can only predict the relative depth. The methodology of using a large-scale model to produce pseudo-labels comes from the training of the DepthAnythingV2 model [49].

## 3.8 Evaluation

Evaluation of depth estimation models cannot be done with a single metric, due to the complex nature of 3-dimensional spaces. For instance, if we only look at the relationship between pixel, we cannot evaluate absolute scale of the scene. On the other hand, if we only evaluate a summed average of the error, we cannot fully grasp, if predictions are correct relative to others. Thus, the evaluation of MDE models is done with several complementary metrics. While some errors focus on absolute errors (RMSE), others access relative depth (AbsRel) or look at the consistency of the predictions (threshold accuracy). Evaluation depth estimation with a variety of these metrics gives a comprehensive insight to model performance aspects, such as global scale- and relational accuracy. This section introduces four standard metrics widely adopted for evaluating monocular depth estimation[3, 2, 24, 49, 28, 13, 45, 13]: root mean squared error, absolute relative error, logarithmic error, and threshold accuracy.

The first metric, Root Mean Squared Error (RMSE), is defined as:

$$\text{RMSE} = \sqrt{\frac{1}{|T|} \sum_{i \in T} (d_i - d_i^*)^2}, \tag{3.1}$$

, where $d_i$ is the depth prediction for pixel $i$, $d_i^*$ is the ground truth depth of corresponding pixel and $|T|$ is the number of valid pixels in ground truth. RMSE provides an intuitive metric great for accessing the absolute depth difference. It penalizes large errors heavily due to its squared-error computation. RMSE does not access the relative depth well, as the error does not penalize relational errors. The next evaluation metric, threshold accuracy is defined as:

$$\text{Threshold}(\delta) = \frac{1}{|T|} \sum_{i \in T} 1 \left[ \max \left( \frac{d_i}{d_i^*}, \frac{d_i^*}{d_i} \right) < \delta \right], \tag{3.2}$$

, where $\delta$ is a predefined threshold. Threshold accuracy provides a way of accessing how many predictions are done "well". Generally, 1.25, $1.25^2$, and

$1.25^3$ are used as three levels of thresholds. Threshold accuracy is strong because it is scale invariant. Although intuitive, the binary nature of threshold accuracy limits the metric from penalizing large errors and rewarding very accurate predictions. Another widely adopted metric is the absolute relative error (AbsRel) defined as:

$$\text{AbsRel} = \frac{1}{|T|} \sum_{i \in T} \frac{|d_i - d_i^*|}{d_i^*}, \tag{3.3}$$

AbsRel is very strong in evaluating scenarios, where relative depth and scene understanding, as it is scale invariant. This makes it a well-suited metric for comparison between scenes of different depth ranges, e.g. indoor and outdoor. However, it can be disproportionately affected by errors in predictions, where the ground truth is close to 0. Finally, we will use the logarithmic error defined as:

$$\text{Log}_{10} = \frac{1}{|T|} \sum_{i \in T} |\log_{10}(d_i) - \log_{10}(d_i^*)|, \tag{3.4}$$

Log10 error particularly useful as it better handles the nonlinear nature of depth values than prior presented metrics. The metrics presented in this section, will provide a strong base for evaluation and understanding how different models perform in certain aspect of depth prediction. This will be particularly relevant in Chapter 5, where the results of the experiments are presented.

Chapter 4

# Experiments

In this section, we presents our experimental investigations aimed at evaluating monocular depth estimation methods. First, we compare several different encoder architectures, such as convolutional neural network (CNN), transformer-based models, a foundation model, and a lightweight custom encoder, establishing baseline performance metrics on the NYUv2 dataset. In doing so, we highlight both the differences in architectural design and the influence of transfer learning on model performance. Second, through a series of ablation studies, we systematically examine the impact of various architectural components, including encoder depth, skip connections, loss functions, and decoder designs. Finally, we explore a custom approach using pseudo-labels generated by DepthAnythingV2 on COCO dataset images, investigating the potential of leveraging unlabeled data for depth estimation. Each experiment is designed to address specific research questions about model architecture, component effectiveness, and training methodology, with results evaluated using standard metrics in the NYUv2 test set.

## 4.1 Experimental Setup

All experiments were conducted on a Linux server located at the AI Pioneer Center Copenhagen. The server was equipped with 2 Intel Xeon Gold 5420+ CPUs, each with 28 cores, and 2 NVIDIA RTX 6000 Ada Generation GPUs, each with 49 GB of memory. However, the model training did not include distribution on both GPUs. The software environment for all experiments included Ubuntu 22.04.5 LTS as the operating system. GPU acceleration was enabled using CUDA 12.2 and NVIDIA driver version 535.183.01. Python 3.10.12 was used to execute the experiments within a virtual environment created using `venv`. The virtual environment ensured isolated dependencies, with all required library versions specified in a `requirements.txt` file for reproducibility. The monitoring tool Weights & Biases [44] was used to plot

training curves compare performance during training.

## 4.2 Architectural Experiment

The first experiment compares the models presented in Section 3. That is, encoders based on ResNet [17], SwinTransformer [30], DenseNet [19], DINOv2 [31] as well as an approach from scratch. The experiment focuses on different methods in designing an encoder/backbone, while the decoder remains naive and unchanged. The Swin-, DenseNet- and ResNet-encoder will be trained both with an without pretrained weights from ImageNet-1K [7] in order to determine the influence of transfer learning.

| **Encoder Type** | **Pretrained Weights** |
|---|---|
| ResNet50 | Yes (ImageNet-1K) |
| Swin Transformer | Yes (ImageNet-1K) |
| DenseNet169 | Yes (ImageNet-1K) |
| DINOv2 | Yes |
| ResNet50 | No |
| Swin Transformer | No |
| DenseNet169 | No |
| Custom | No |

**Table 4.1:** Models Trained for the Architectural Encoder Experiment

The training process utilized a preprocessed subset of the NYUv2 dataset [39], consisting of approximately 50k paired RGB images and ground truth depth maps. This subset was obtained from the `FastDepth` GitHub repository by Lee et al. [45]. The preprocessed dataset was chosen for practicality, as it includes only relevant RGB images and depth maps. Unlike the original dataset, which exceeds 400GB and contains unrelated data, such as object labels, the subset is conveniently formatted with RGB images and depth maps stored pairwise in `.h5` files. Data preprocessing and augmentation techniques used for this experiment are outlined in Section 3.7.

Training was conducted on the hardware setup detailed in Section 4.1 for approximately 1,5 million iterations (30 epochs) with a batch size of 8. The PyTorch stochastic gradient-based optimizer, AdamW [1], implementing the algorithm proposed in [21], was used with a learning rate of $1 \cdot 10^{-4}$ and weigth decay. This learning rate was selected based on its widespread adoption in the literature, as demonstrated in works such as [2, 3, 35]. The loss function employed in this experiment was the standard L1 loss. For comparisons and insights into the impact of different loss functions, refer to Section 4.3.

The accuracy of the model is evaluated on the densely labeled official test set provided in [39], using the following metrics, as detailed in Section 3.8:

- Root Mean Square Error (RMSE),
- Mean Absolute Relative Error (AbsRel),
- Logarithmic Error (log10), and
- $\delta$ thresholds ($\delta_1$, $\delta_2$, $\delta_3$).

Additionally, qualitative evaluation will be conducted using examples of model predictions performed on the official test split.

## 4.3 Ablation Study

The ablation study aims to evaluate the contribution of the individual components of the model, within the architectural designs tested in the previous experiment. The data set, the training process, and hyperparameters remain unchanged from 3.8, unless otherwise specified.

Firstly, a comparison of different encoder depths will be made. Specifically, three models are trained using pre-trained encoders. These models utilize pre-trained ResNet50, ResNet101, and ResNet152 backbones, respectively. We expect deeper models to enhance performance. However, we aim to determine whether the performance gains justify the additional parameters and the longer inference time. The models of these experiments were trained for 20 epochs.

Aditionally, the ablation study investigates the impact of skip connections between the encoder and decoder in a U-net architecture. This experiment is performed on both the scratch-trained model and the ResNet50-based model, while retaining skip connections within the encoder network. This involves training each model both with and without skip connections between the encoder and decoder. Based on prior research, it is hypothesized that skip connections will improve model performance, both quantitatively and qualitatively, based on the theoretical information preservation they provide. The models of the ablation are trained for 20 epochs.

Furthermore, the ablation study covers the impact of loss function. The loss functions covered are $L1$, $L1Smooth$, $L2$ and Huber loss. The comparison will be made on the model trained from scratch. Models of this experiment was only trained for 10 epochs.

The final ablation covers two different decoder architecture and the impact on the final upsampled depth prediction. The comparison will be done between the architecture adopted in 4.2, upsampling by interpolation, and an unprecedented design utilizing transposed convolutions.

| Study Component | Configurations Tested |
|---|---|
| **Encoder Depth** | ResNet50, ResNet101, ResNet152 (pretrained) |
| **Skip Connections** | With and without encoder-decoder skip connections for ResNet50 and scratch-trained models |
| **Loss Functions** | L1, L1Smooth, L2, Huber |
| **Decoder Design** | Interpolation-based upsampling vs. transposed convolution |

**Table 4.2:** Summary of Ablation Studies

All ablation studies are evaluated with the same metrics and test set described in 4.2, and will be evaluated both quantitatively and qualitatively.

## 4.4 Pseudo-Label Experiment

Finally, we want to explore the feasibility of training MDE models using pseudo-labeled data. For this experiment, the largest model of DepthAnythingV2 [49] has been used as a teacher model to annotate unlabeled image data from Microsoft's Common Objects in Context (COCO) [29]. 83k RGB images were annotated with pseudo-labels. The annotations represent the relative inverse depth predictions within the interval [0, 1], where values closer to 0 correspond to greater distances.

**Figure 4.1:** Example of a pseudo-label from the COCO dataset annotated by DepthAnythingV2. The left image shows the original COCO RGB image, and the right image shows the corresponding pseudo-label with relative inverse depth values.

A Resnet based student model is trained solely via supervised learning, accepting the pseudo-labels as ground-truth data. Model performance will

be accessed quantitatively on a test subset of pseudolabeled.

## 4.5 Reproducibility

For reproducibility purposes, all code, configurations, and training scripts used in these experiments are publicly available on my GitHub repository. The repository includes a `readme.md` file with instructions for setting up the environment, downloading the data set and training the model. The implementation uses PyTorch and maintains consistent random seeds(10) across experiments for reproducible results. The `requirements.txt` specifies all necessary dependencies and their versions.

Chapter 5

# Results & Discussion

The primary objective of this thesis is to investigate and evaluate various MDE approaches with respect to accuracy, efficiency, and generalization. This includes comparison of architectures based on CNN, Vision Transformers, and foundation models, such as DINOv2. Furthermore, the study examines the role of transfer learning in developing strong performing MDE models, by leveraging pre-trained networks from other vision tasks, such as classification. In addition to the architectural focus, we also aim to address the challenge of data scarcity in MDE by experimenting with models trained solely on pseudo-labels data generated by a large-scale state-of-the-art model.

**Accuracy of different architectures.** The quantitative performance of the models was evaluated using standard metrics for monocular depth estimation, including the root mean square error (RMSE), the absolute relative error (AbsRel), the logarithmic error (Log10) and the thresholds $\delta$. The evaluation of both models initialized with and without pre-trained weights can be seen in Table 5.

One of the most significance results is the huge accuracy difference between pre-trained and non-pre-trained models. All models initialized with pre-trained weight demonstrate a clear advantage across all metrics. The models with ResNet, DenseNet and Swin encoders was all initialized with weights from image classification training. Comparing the tightest threshold accuracy of 25%, the Swin-, DenseNet- and ResNet-based models improved with 31%, 25%, 24%, respectively, while the RMSE was reduced by approximately 25% across all three models. This significant accuracy boost, utilizing weights from another task, highlights the importance of leveraging transfer learning in computer vision task, when they share a common domain. Not only does it enhance final performance, but it also reduces training time, as encoders are already capable of extracting meaningful features at initialization, unlike models initialized with random weights. The idea of feature sharing between

different tasks within the same domain is further enhanced by the performance of the model with DINOv2 encoder. This model performs best across all metrics, showing that the foundation model is capable of performing extremely well on downstream tasks, by having been trained to learn features that are generally important.

Based on literature surveyed for this thesis, we would expect a transformer-based encoder to outperform a solely CNN encoder. However, the Swin transformer did not perform to the expected extent compared to the CNN-based models. When expecting the results of the Swin-encoder trained without pretrained weights, it perform better than the CNN-based models on threshold $1.25^2$ and $1.25^3$, which highlights its ability to capture global context better. However, it has fewer precise predictions ($\delta$ 1) and worse RMSE. An resonable explanation for the lack of performance boost, might be the architecture of the Swin-Transformer. It reduces computational needs by computing self-attention on local windows, instead of capturing global context from the first forward, as transformers are designed to do. For a future project, it would be interesting to compare the Swin-Transformer with another transformer backbone trained from scratch.

**Table 5.1:** Performance Comparison of Different Encoders on the NYUv2 Test Set. Models are grouped according to initialization (pretrained vs. from scratch). Evaluation is done on **scaled** depth maps at prediction size

| *Encoders initialized with pretrained weights* | | | | | | |
|---|---|---|---|---|---|---|
| **Model** | $\delta_1 \uparrow$ | $\delta_2 \uparrow$ | $\delta_3 \uparrow$ | **RMSE**$\downarrow$ | **AbsRel**$\downarrow$ | **log10**$\downarrow$ |
| ResNet50 | 0.801 | 0.950 | 0.983 | 0.648 | 0.147 | 0.064 |
| SwinTransformer | 0.804 | 0.947 | 0.980 | 0.661 | 0.150 | 0.065 |
| DenseNet169 | 0.808 | 0.947 | 0.980 | 0.662 | 0.146 | 0.067 |
| DINOv2 | **0.903** | **0.983** | **0.996** | **0.480** | **0.103** | **0.045** |
| *Encoders trained from scratch* | | | | | | |
| **Model** | $\delta_1 \uparrow$ | $\delta_2 \uparrow$ | $\delta_3 \uparrow$ | **RMSE**$\downarrow$ | **AbsRel**$\downarrow$ | **log10**$\downarrow$ |
| ResNet50 | 0.644 | 0.873 | 0.955 | 0.877 | 0.221 | **0.094** |
| SwinTransformer | 0.616 | **0.878** | **0.961** | 0.896 | 0.229 | 0.097 |
| DenseNet169 | **0.647** | 0.874 | 0.955 | **0.867** | **0.210** | **0.094** |
| Custom | 0.524 | 0.812 | 0.934 | 1.048 | 0.276 | 0.117 |

The custom designed encoder model, which combined a lightweight CNN with a spatial attention module, did not perform particularly well. Inspecting the $\delta_1$ threshold, it performed 23% worse than the best-performing trained model from scratch. The intuition of the design was to utilize the spatial attention mechanism, proven effective for MDE, with the lightweight nature of a small CNN network. If we had more time, we would have experimented

with more complex hybrid architectures for the encoder, searching for the sweet spot between current hardware and model complexity. In general, models that leverage pre-trained weight perform reasonably well, although slightly worse, than comparable models from the literature [9, 24, 2, 10].

This comparison of pre-trained and scratch-trained models reinforces the importance of transfer learning in achieving high-performance MDE systems. The superior performance of the DINOv2 encoder provides insight into how foundation models, trained for general tasks, can be effectively adapted for depth estimation, advancing the understanding of architecture selection for MDE. These findings advance our understanding of how architectural design impacts the accuracy of models, but also emphasize that training strategies are just as important.

**Encoder Depth.** In the ablation study of the experimental work, we aimed to cover the influence of making the following changes on our model – increasing encoder depth, removing skip connections, adopting different loss functions, and changing the architecture of the decoder. Following results in Appendix A.1 increasing encoder depth improved performance slightly. Examining RMSE performance, ResNet101 and ResNet152 obtained a performance boost of approximately 2% and 4.5%, respectively. While accuracy is improved, the number of parameters also increased from ResNet50's 28 million, to 47 million for ResNet101 and 62 million for ResNet152, resulting in a vastly increase in inference time, see Chapter 5. Following the motivation for MDE of making precise depth estimation feasible for smaller devices, the modest performance boost must be weighted against the additional computational requirements. From a practical perspective, the trade-off between performance and computational requirements becomes critical, when considering one of the key motivation of MDE – availability. In scenarios such as smaller real time systems such as drones, where computational resources is constrained, the accuracy boost might not justify the increased inference time. This finding highlights the importance of prioritizing based on application purpose and not just pure accuracy performance.

**Skip Connections.** The removal of skip connections between the encoder and decoder led to a slight decrease in quantitative performance Table A.2. However, examining the qualitative results revealed a much more significant impact; see Fig. 5.2. The model *with* skip connections was shown to produce depth maps with sharper edges around the object boundaries. This improvement can be attributed to the impact of skip connections in transferring high-resolution spatial details directly from the encoder to the decoder in all upsampling layers. The model *without* skip connection relies only on upsampling the depth map from the deepest encoder layer, resulting in more blurry depth transitions between objects.

| Model | Total Parameters | Trainable Parameters | Avg. Inference Time (ms) |
|---|---|---|---|
| custom | 5,434,049 | 5,434,049 | 4.02 |
| dino | 95,558,593 | 8,978,113 | 10.31 |
| res50 | 28,076,801 | 28,076,801 | 4.64 |
| res101 | 47,068,929 | 47,068,929 | 8.05 |
| res152 | 62,712,577 | 62,712,577 | 11.44 |
| dense | 44,322,689 | 44,322,689 | 13.54 |
| swin | 96,839,801 | 96,839,801 | 13.54 |

**Table 5.2:** Comparison of MDE Models: total parameters, trainable parameters, and average inference time. Inference time is computed on a single Nvidia RTX 6000 GPU, which is an enterprise GPU.

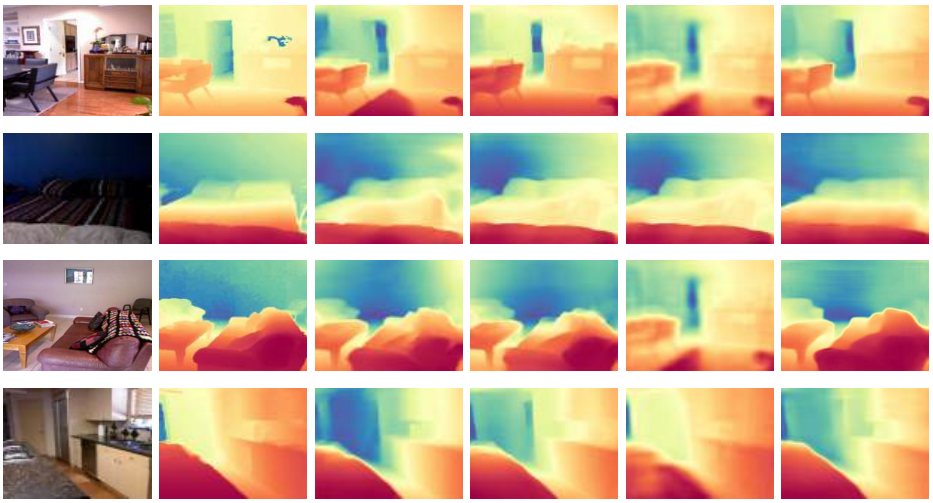

**Figure 5.1:** Qualitative results on models with pretrained encoder. **a)** Input RGB image. **b)** Ground truth depth map, **c)** ResNet50, **d)** DenseNet169 **e)** Swin, **f)** DINOv2

.

**Loss functions.** The choice of loss function had a very modest impact on the performance on our lightweight scratch model, see Table A.3. However, **L1** and **L1smooth** yielded the best results overall. The results of this ablation, does not provide a huge insight into the field of loss functions in MDE, as many papers experiment with more advanced combinations of loss function, especially when dealing with mixed datasets.

**Decoder Architecture.** This ablation covered the substitution of the decoder based on interpolation, used in all other experimental work, with another design using transposed convolutions to upsample the encoders output. Quantitatively, the interpolation decoder demonstrates a slightly superior performance, see Appendix A.4. In addition to its quantitative superiority,

the interpolation based decoder avoided the checkerboard pattern observed in predictions from the decoder with transposed convolutions, see Fig. A.3. These artifacts, a known issue with transposed convolutions, highlight the practical challenges of using this design despite its theoretical potential for richer feature reconstruction.

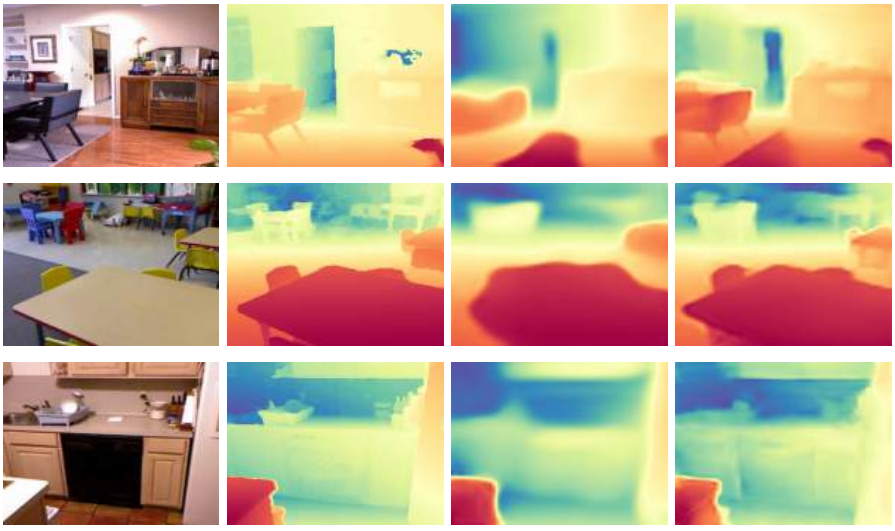

**Figure 5.2:** Qualitative results on the same ResNet50 model with and without skip connections between encoder and decoder. **a)** RGB input, **b)** ground truth depth map, **c)** without skip connections, **d)** with skip connections.

**Pseudo Labeling Experiment.** In this experiment, we addressed the challenge of data scarcity in monocular depth estimation (MDE) utilizing pseudo-labeled data generated by the DepthAnythingV2 model in the COCO dataset. Unlike standard test sets with predefined ground truth, the COCO test set in this study was defined and pseudo-labeled by us, providing a unique opportunity to evaluate the impact of noisy labels on model performance.

Quantitatively, models trained on pseudolabeled data did not perform particularly well when evaluated on traditional metrics such as threshold ($\delta$). The diversity of the COCO dataset, has a very wide range of scenes, object categories, and spatial configurations, probably made it difficult for the model to excel in qualitative metrics.

Qualitative results suggest that the model is generalizing well to different domains. For example, depth maps generated on COCO validation images retained the sharpness observed in pseudo-labeled training data, as shown in the Appendix B. This generalization indicates that pseudolabeling can effectively transfer knowledge to unseen environments, even with the limitations posed by label noise.

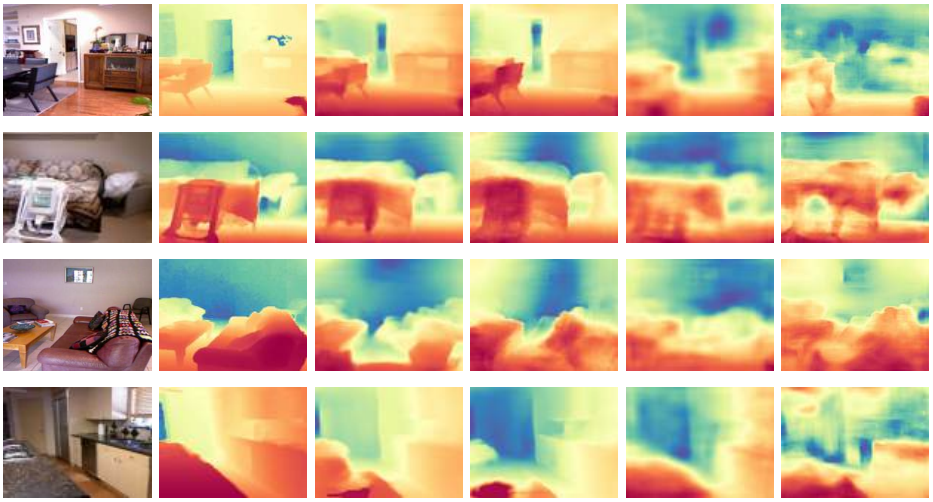

**Figure 5.3:** Qualitative results on models with non-pretrained encoder. **a)** Input RGB image. **b)** Ground truth depth map, **c)** ResNet50, **d)** DenseNet169 **e)** Swin, **f)** Custom

.

These findings highlight the potential of pseudolabeling as a scalable approach to training MDE models when ground-truth depth data is unavailable or impractical to obtain. While quantitative performance remains an area for improvement, the qualitative results demonstrate that pseudo-labeling can successfully teach models key spatial relationships and structural consistency, offering a promising direction for future research.

**Limitations.** The following highlights and discusses the limitations encountered during the development and experimentation of this project. All models in this work were trained on data scaled down to a relatively low resolution of $224 \times 224$ as well as producing depth maps of half the resolution. This was an intentional choice made at the beginning of the experimental work to reduce computational demands and training times. When investigating many different model architectures, long training times can reduce the coverage of different approaches. Consequently, this limits the models ability to produce high-resolution depth predictions.

The experimental work of this project relies on a relatively small dataset compared to the vast datasets used in recent state-of-the-art models. For example, the DepthAnythingv2 [49] model was trained on more than 62 million images, enabling greater generalization and performance. In contrast, the limited data set size in this project inherently restricts the ability of the models to generalize to diverse and unseen scenarios. This constraint serves as a reminder of the critical role of dataset size and diversity in model development. The pseudo-label experiment served as a small-scale experiment

on how to address the data scarcity, but scaling that experiment up would require extreme computational power and time.

The data augmentation techniques used in this project were limited to horizontal flipping and random cropping. In contrast, many modern MDE models adopt more extensive augmentation policies, such as brightness adjustments, channel swaps, or noise injection [14, 35, 5, 2]. These additional augmentations help improve robustness and generalization. The absence of such techniques in this work likely constrained the models' ability to perform effectively across a wide range of lighting and environmental conditions.

The CNN architectures employed in this project are relatively straightforward, without incorporating recent advancements such as more complex hybrid architectures. Although these CNN models are robust and grounded in foundational research, they may not fully leverage modern innovations essential for achieving state-of-the-art performance. This limitation arose due to time constraints and the lack of knowledge of advanced neural networks at the beginning of the project. However, these architectures provided a solid foundation for experimental exploration and understanding.

The investigation of loss functions in this project was limited to a small number of frequently used ones. More complex loss functions, often required for tasks such as mixing datasets or unsupervised learning, were not explored. This limitation reflects the strict supervised learning approach adopted in this project. Future investigations could expand into advanced loss functions to improve model adaptability and performance in more challenging scenarios.

Finally, the project did not produce experimental results on the inference time of the models on smaller devices. Given more time, experiments could have been conducted on various devices, such as Nvidia's Jetson Nano, to evaluate the feasibility of implementing real-time solutions on smaller, resource-constrained devices and to contextualize the work. This limitation leaves open the question of how these methods could perform in practical real-world applications requiring real-time inference.

In summary, while these limitations define the boundaries of the project, they also point to opportunities for future exploration and improvement. By addressing these aspects, future work can build upon the foundations laid by this project and push toward more robust and generalizable MDE models.

Chapter 6

# Conclusion & Future Work

After working with the objectives of systematically investigating MDE architectures and experimenting with pseudo-labled data to address data scarcity, we con conclude the following - Firstly, the evaluation of CNN-based models, Vision Transformers, and foundation models demonstrated clear differences in performance. Models leveraging initilization of pretrained encoder weights consistently outperformed those trained from scratch, with the DINOv2 foundation model achieving the highest accuracy despite frozen weights, highlighting the importance of robust general feature extraction and the power of transfer learning. A custom, more lightweight model, was proposed to lower complexity and inference time, but it's accuracy was modest compared to proven architectures such as ResNet, DenseNet and Swin-transformer.

Secondly, ablation studies highlighted the significance of architectural decisions in MDE models. Increasing encoder depth marginally improved accuracy but significantly increased inference time, making it less practical for resource-constrained scenarios. Skip connections were found to be crucial for preserving local information and producing sharper depth maps. These findings reinforce the importance of fine-tuning architectural and training choices based on application-specific constraints.

Finally, the pseudo-labeling experiment addressed the challenge of data scarcity by utilizing a teacher model to generate pseudo-labeled data from the COCO dataset. This experiment, although a very small scale, gave some promising results for the feasibility of training on pseudo-labeled data.

In conclusion, this thesis achieves its objective by providing a detailed investigation of the design, training, and evaluation of MDE models, identifying the strengths and weaknesses of various architectures, and exploring practical solutions to data scarcity. The results emphasize that achieving accurate, efficient, and generalizable MDE requires careful consideration of architectural

trade-offs, data- and training strategies.

Future work can build upon this work, by exploring alternative foundation models for pseudo-labeling, optimizing more lightweight architectures for greater efficiency, and extend evaluations to other more diverse dataset or benchmark suites. Exploring more complex decoding is also a subject for future work.

# Bibliography


[1] *Adamw pytorch 2.5 documentation*, https://pytorch.org/docs/stable/generated/torch.optim.AdamW.html, Accessed: 2024-12-29, 2024.

[2] I. Alhashim and P. Wonka, "High quality monocular depth estimation via transfer learning," *CoRR*, vol. abs/1812.11941, 2018. arXiv: 1812.11941. [Online]. Available: http://arxiv.org/abs/1812.11941.

[3] S. F. Bhat, I. Alhashim, and P. Wonka, "Adabins: Depth estimation using adaptive bins," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[4] A. Bhoi, "Monocular depth estimation: A survey," *CoRR*, vol. abs/1901.09402, 2019. arXiv: 1901.09402. [Online]. Available: http://arxiv.org/abs/1901.09402.

[5] A. Bochkovskii *et al.*, *Depth pro: Sharp monocular metric depth in less than a second*, 2024. arXiv: 2410.02073 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2410.02073.

[6] R. Bommasani *et al.*, *On the opportunities and risks of foundation models*, 2022. arXiv: 2108.07258 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2108.07258.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and F.-F. Li, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[8] A. Dosovitskiy *et al.*, *An image is worth 16x16 words: Transformers for image recognition at scale*, 2021. arXiv: 2010.11929 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2010.11929.

[9] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[10] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, "Deep ordinal regression network for monocular depth estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[11] S. Gasperini, N. Morbitzer, H. Jung, N. Navab, and F. Tombari, "Robust monocular depth estimation under challenging conditions," in *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, IEEE, Oct. 2023. DOI: 10.1109/iccv51070.2023.00751. [Online]. Available: http://dx.doi.org/10.1109/ICCV51070.2023.00751.

[12] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[13] C. Godard, O. Mac Aodha, and G. J. Brostow, *Unsupervised monocular depth estimation with left-right consistency*, 2017. arXiv: 1609.03677 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1609.03677.

[14] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow, "Digging into self-supervised monocular depth estimation," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019.

[15] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[16] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[17] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, 2015. arXiv: 1512.03385 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1512.03385.

[18] A. G. Howard *et al.*, *Mobilenets: Efficient convolutional neural networks for mobile vision applications*, 2017. arXiv: 1704.04861 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1704.04861.

[19] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, *Densely connected convolutional networks*, 2018. arXiv: 1608.06993 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1608.06993.

[20] S. Ioffe and C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, 2015. arXiv: 1502.03167 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1502.03167.

[21] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017. arXiv: 1412.6980 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1412.6980.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, vol. 25, 2012, pp. 1097–1105.

[23] H. Laga, L. V. Jospin, F. Boussaid, and M. Bennamoun, "A survey on deep learning techniques for stereo-based depth estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 4, pp. 1738–1764, Apr. 2022, ISSN: 1939-3539. DOI: 10.1109/tpami.2020.3032602. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2020.3032602.

[24] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks," in *3D Vision (3DV), 2016 Fourth International Conference on*, 2016.

[25] *Leakyrelu - pytorch 2.5 documentation*, https://pytorch.org/docs/stable/generated/torch.nn.LeakyReLU.html, Accessed: 2025-01-06, 2025.

[26] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[27] D.-J. Lee *et al.*, *Lightweight monocular depth estimation via token-sharing transformer*, 2023. arXiv: 2306.05682 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2306.05682.

[28] J. H. Lee, M.-K. Han, D. W. Ko, and I. H. Suh, "From big to small: Multi-scale local planar guidance for monocular depth estimation," *CoRR*, vol. abs/1907.10326, 2019. arXiv: 1907.10326. [Online]. Available: http://arxiv.org/abs/1907.10326.

[29] T.-Y. Lin *et al.*, *Microsoft coco: Common objects in context*, 2015. arXiv: 1405.0312 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1405.0312.

[30] Z. Liu *et al.*, *Swin transformer: Hierarchical vision transformer using shifted windows*, 2021. arXiv: 2103.14030 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2103.14030.

[31] M. Oquab *et al.*, *Dinov2: Learning robust visual features without supervision*, 2024. arXiv: 2304.07193 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2304.07193.

[32] N. Padkan, P. Trybala, R. Battisti, F. Remondino, and C. Bergeret, "EVALUATING MONOCULAR DEPTH ESTIMATION METHODS," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLVIII-1/W3-2023, pp. 137–144, 2023. DOI: 10.5194/isprs-archives-XLVIII-1-W3-2023-137-2023. [Online]. Available: https://doi.org/10.5194/isprs-archives-XLVIII-1-W3-2023-137-2023.

[33] U. Rajapaksha, F. Sohel, H. Laga, D. Diepeveen, and M. Bennamoun, "Deep learning-based depth estimation methods from monocular image and videos: A comprehensive survey," *ACM Computing Surveys*, vol. 56, no. 12, pp. 1–51, Oct. 2024, ISSN: 1557-7341. DOI: 10.1145/3677327. [Online]. Available: http://dx.doi.org/10.1145/3677327.

[34] R. Ranftl, A. Bochkovskiy, and V. Koltun, "Vision transformers for dense prediction," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2021.

[35] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, *Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer*, 2020. arXiv: 1907.01341 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1907.01341.

[36] *Relu - pytorch 2.5 documentation*, https://pytorch.org/docs/stable/generated/torch.nn.ReLU.html, Accessed: 2025-01-06, 2025.

[37] O. Ronneberger, P. Fischer, and T. Brox, *U-net: Convolutional networks for biomedical image segmentation*, 2015. arXiv: 1505.04597 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1505.04597.

[38] *Sigmoid - pytorch 2.5 documentation*, https://pytorch.org/docs/stable/generated/torch.nn.Sigmoid.html, Accessed: 2025-01-06, 2025.

[39] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, Springer, Berlin, Heidelberg, 2012, pp. 746–760. DOI: 10.1007/978-3-642-33715-4_54.

[40] M. Tan and Q. V. Le, *Efficientnet: Rethinking model scaling for convolutional neural networks*, 2020. arXiv: 1905.11946 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1905.11946.

[41] A. Varma, H. Chawla, B. Zonooz, and E. Arani, "Transformers in self-supervised monocular depth estimation with unknown camera intrinsics," in *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, SCITEPRESS - Science and Technology Publications, 2022. DOI: 10.5220/0010884000003124. [Online]. Available: http://dx.doi.org/10.5220/0010884000003124.

[42] A. Vaswani *et al.*, *Attention is all you need*, 2023. arXiv: 1706.03762 [cs.CL]. [Online]. Available: https://arxiv.org/abs/1706.03762.

[43] *Vision/torchvision/models/resnet.py at main - pytorch/vision*, https://github.com/pytorch/vision/blob/main/torchvision/models/resnet.py, Accessed: 2025-01-04, 2025.

[44] *Weights  biases: The ai developer platform*, https://wandb.ai/site/, Accessed: 2025-01-08, 2025.

[45] Wofk, Diana and Ma, Fangchang and Yang, Tien-Ju and Karaman, Sertac and Sze, Vivienne, "FastDepth: Fast Monocular Depth Estimation on Embedded Systems," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[46] F. Wu and L. Chen, *Depth estimation maps of lidar and stereo images*, 2022. arXiv: 2212.11741 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2212.11741.

[47] G. Yang, H. Tang, M. Ding, N. Sebe, and E. Ricci, *Transformer-based attention networks for continuous pixel-wise prediction*, 2021. arXiv: 2103.12091 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2103.12091.

[48] L. Yang, B. Kang, Z. Huang, X. Xu, J. Feng, and H. Zhao, *Depth anything: Unleashing the power of large-scale unlabeled data*, 2024. arXiv: 2401.10891 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2401.10891.

[49] L. Yang *et al.*, *Depth anything v2*, 2024. arXiv: 2406.09414 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2406.09414.

[50] J. Zhang and Y. Ma, "Knowledge distillation: A survey," *International Journal of Automation and Computing*, vol. 18, no. 3, pp. 1–17, 2021.

[51] N. Zhang, F. Nex, G. Vosselman, and N. Kerle, *Lite-mono: A lightweight cnn and transformer architecture for self-supervised monocular depth estimation*, 2023. arXiv: 2211.13202 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2211.13202.

[52] C. Zhao *et al.*, "Monovit: Self-supervised monocular depth estimation with a vision transformer," in *2022 International Conference on 3D Vision (3DV)*, IEEE, Sep. 2022, pp. 668–678. DOI: 10.1109/3dv57658.2022.00077. [Online]. Available: http://dx.doi.org/10.1109/3DV57658.2022.00077.

[53] W. Zhao, Z. Gong, C. Wang, and P. Feng, "Domain adaptation for monocular depth estimation using semantic information," in *International Conference on Pattern Recognition (ICPR)*, 2020.

Appendix A

# Ablation Study

## A.1 Encoder Depth

**Table A.1:** Performance Comparison of Different ResNet Depths on the NYUv2 Test Set.

| *Comparison of Encoder Depth – Training through 10 epochs* | | | | | | |
|---|---|---|---|---|---|---|
| **Model** | $\delta_1 \uparrow$ | $\delta_2 \uparrow$ | $\delta_3 \uparrow$ | **RMSE**$\downarrow$ | **AbsRel**$\downarrow$ | **log10**$\downarrow$ |
| ResNet50 | 0.8009 | 0.9486 | 0.9814 | 0.6433 | 0.1516 | 0.0642 |
| ResNet101 | 0.8126 | 0.9509 | 0.9837 | 0.6288 | 0.1430 | 0.0617 |
| ResNet152 | **0.8215** | **0.9588** | **0.9872** | **0.6131** | **0.1390** | **0.0597** |

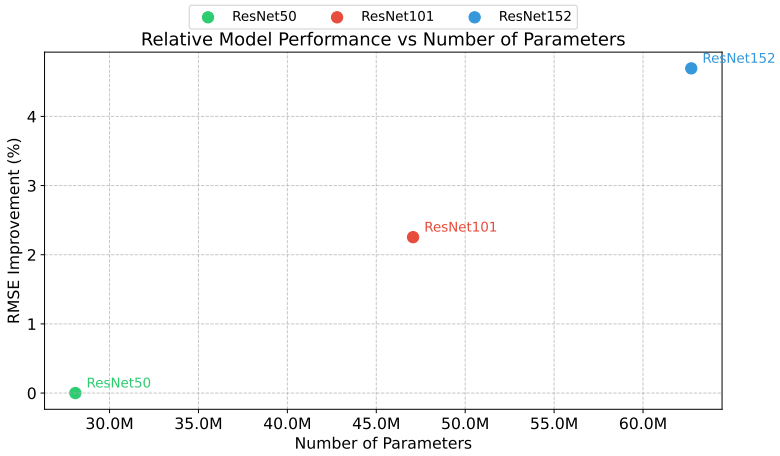


**Figure A.1:** Relative performance boost (%) as encoder depth increases on the NYUv2 test set.

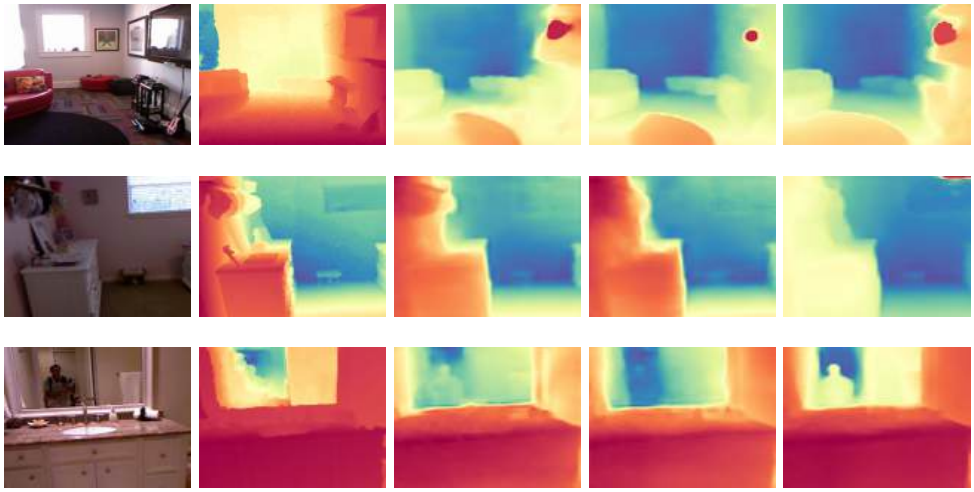

**Figure A.2:** Qualitative comparison of encoder depth for ground truth and predictions using ResNet50, ResNet101, and ResNet152.

## A.2 Skip Connections

**Table A.2:** Performance Comparison With and Without Skip Connections

| *ResNet50 Model Variants on NYUv2 Test Set* | | | | | | |
|---|---|---|---|---|---|---|
| **Variant** | $\delta_1 \uparrow$ | $\delta_2 \uparrow$ | $\delta_3 \uparrow$ | **RMSE**$\downarrow$ | **AbsRel**$\downarrow$ | **log10**$\downarrow$ |
| With Skip | 0.8009 | 0.9486 | 0.9814 | 0.6433 | 0.1516 | 0.0642 |
| Without Skip | 0.7969 | 0.9471 | 0.9840 | 0.6494 | 0.1475 | 0.0644 |

## A.3 Loss Functions

**Table A.3:** Performance Comparison of Different Loss Functions for a Scratch-Trained Model on NYUv2

| *Evaluation of Loss Functions on NYUv2 Test Set* | | | | | | |
|---|---|---|---|---|---|---|
| **Loss Function** | $\delta_1 \uparrow$ | $\delta_2 \uparrow$ | $\delta_3 \uparrow$ | **RMSE**$\downarrow$ | **AbsRel**$\downarrow$ | **log10**$\downarrow$ |
| Huber | 0.5107 | 0.8071 | 0.9368 | 1.0410 | 0.2837 | 0.1187 |
| L2 | 0.4982 | 0.8016 | 0.9391 | 1.0475 | 0.2797 | 0.1202 |
| L1smooth | 0.5112 | 0.8136 | 0.9404 | 1.0331 | 0.2764 | 0.1174 |
| L1 | 0.5126 | 0.8076 | 0.9357 | 1.0479 | 0.2872 | 0.1186 |

## A.4 Decoder Architecture.

| *Decoder Variants on NYUv2 Test Set - Scratch Model* | | | | | | |
|---|---|---|---|---|---|---|
| **Decoder** | $\delta_1 \uparrow$ | $\delta_2 \uparrow$ | $\delta_3 \uparrow$ | **RMSE**$\downarrow$ | **AbsRel**$\downarrow$ | **log10**$\downarrow$ |
| Interpolation | 0.524 | 0.812 | 0.934 | 1.048 | 0.276 | 0.117 |
| Transposed | 0.507 | 0.797 | 0.930 | 1.065 | 0.292 | 0.121 |

**Table A.4:** Performance Comparison Between Interpolation and Transposed Convolutions as Decoder Architectures

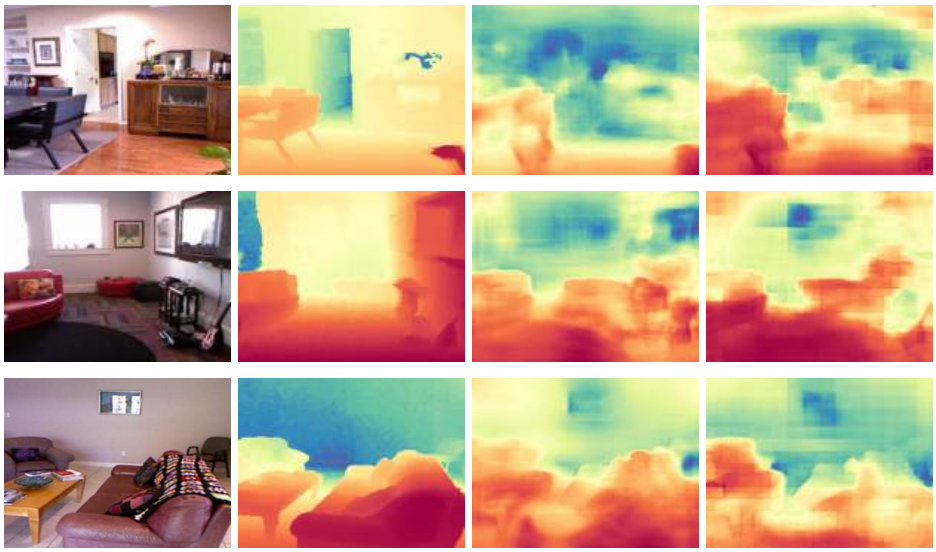

**Figure A.3:** Qualitative performance comparison of decoders. Columns represent: (a) input RGB image, (b) ground truth depth map, (c) depth predictions with interpolation decoder, and (d) depth predictions with transposed convolution decoder.

Appendix B

# Pseudo Labels

**Table B.1:** Performance - models trained on COCO. Evaluated on unseen subset of COCO

| **Model** | $\delta_1 \uparrow$ | $\delta_2 \uparrow$ | $\delta_3 \uparrow$ |
|---|---|---|---|
| DINODepth | 0.5658 | 0.7626 | 0.8534 |
| ResNet152 | 0.5015 | 0.6987 | 0.8016 |
| Swin | 0.5043 | 0.7034 | 0.8054 |
| Scratch | 0.3802 | 0.5820 | 0.7052 |

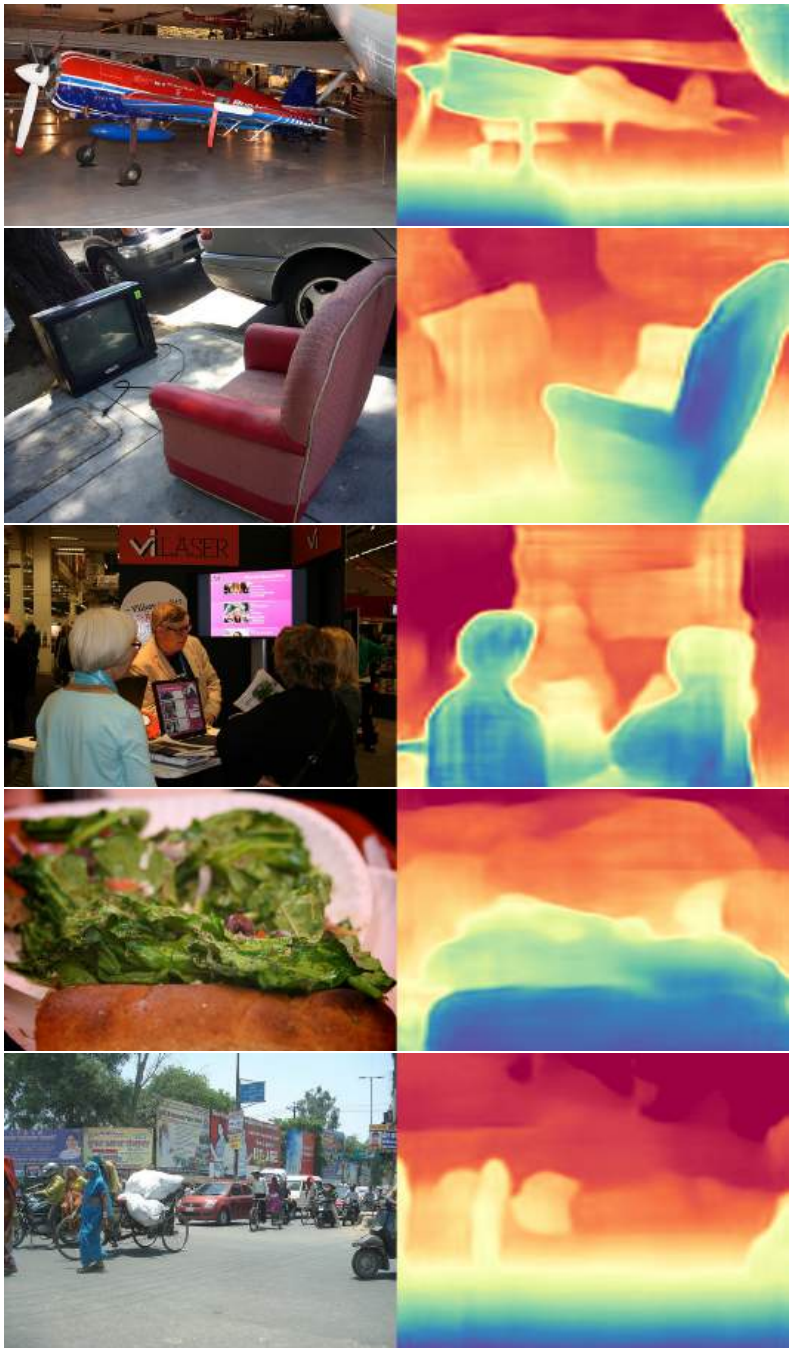

**Figure B.1:** Qualitative performance of model trained on COCO dataset with DINOv2 as encoder

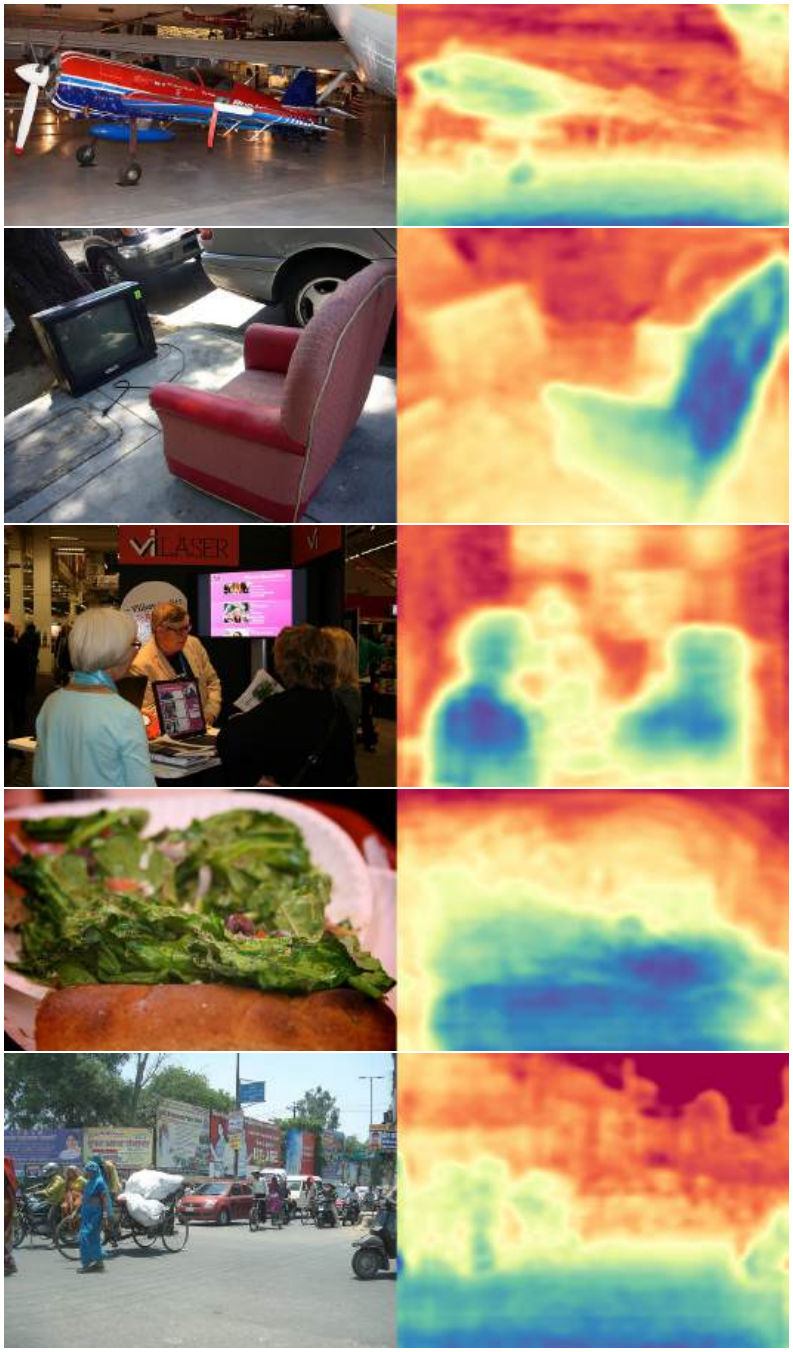

**Figure B.2:** Qualitative performance of the model trained on the COCO dataset with Swin Transformer as encoder

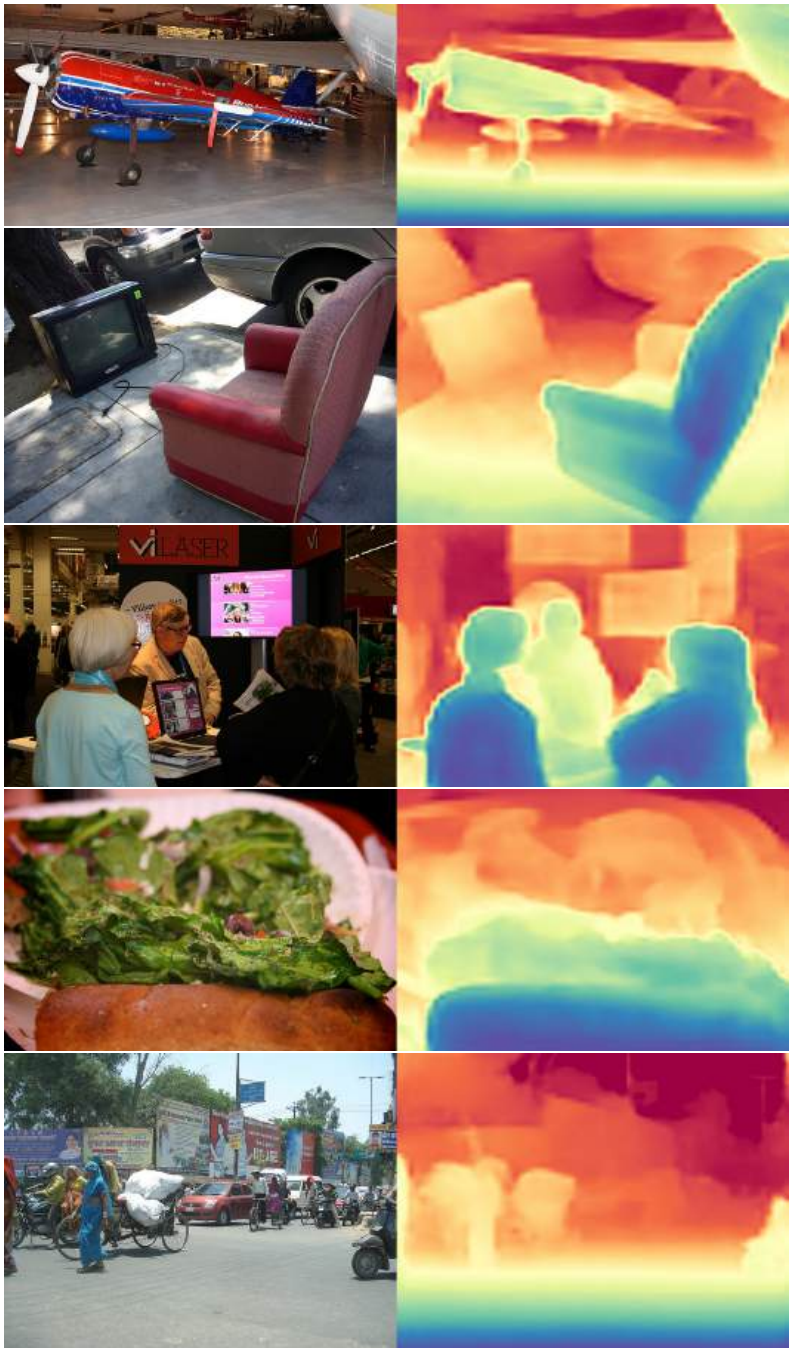

**Figure B.3:** Qualitative performance of the model trained on the COCO dataset with ResNet152 as encoder

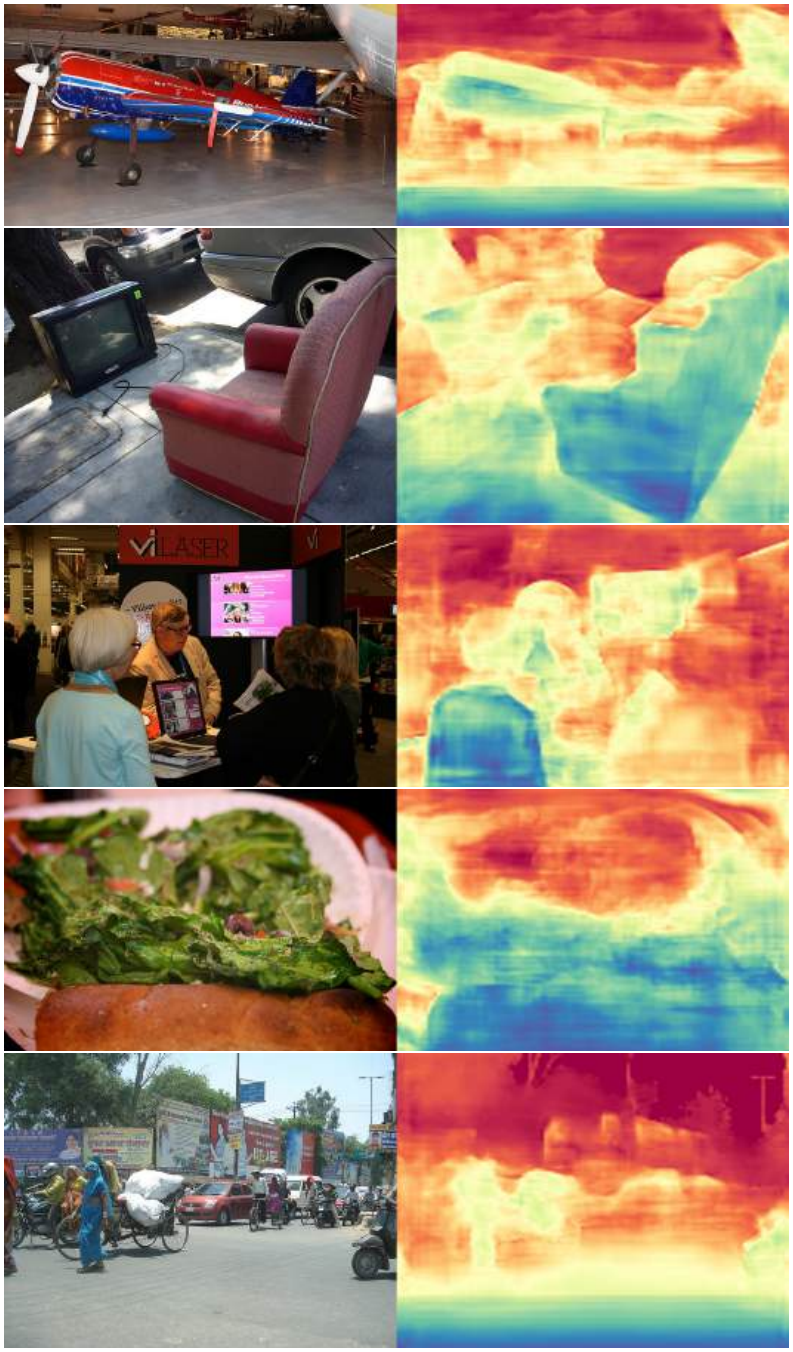

**Figure B.4:** Qualitative performance of the model trained on the COCO dataset with the custom Scratch model as encoder