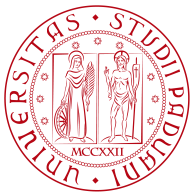

University of Padova

Department of Mathematics

Master Thesis in Data Science

Hybrid Encoder and Architectures for Advanced Monocular Depth Estimation: A Comparative Synthesis Approach

Supervisor
Professor Lamberto Ballan
University of Padova

Co-supervisor
Dott.ssa Elena Izzo
University of Padova

Master Candidate
Daniela Di Labbio

Academic Year
2024-2025

To my mother and father,
those who believed in me from the beginning to the end.
To you, who more than anyone else would have liked to be there.
To my grandfather Alessandro.

Abstract

Monocular depth estimation represents a critical capability in computer vision systems, with applications ranging from autonomous driving to augmented reality. Despite significant advances in this field, current solutions often require a trade-off between model performance and computational efficiency, limiting their practical deployment. This thesis introduces an innovative architectural integration approach that combines the complementary strengths of two leading models: Xi-Net and Lite-Mono, to address this fundamental challenge.

The methodology involves a systematic evaluation of architectural fusion strategies, carefully analyzing how components from both source models can be effectively combined while preserving their respective strengths. We conduct extensive experiments using the KITTI dataset, a comprehensive benchmark suite for autonomous driving applications, to validate our approach across diverse real-world scenarios including urban, residential, and highway environments.

The model variants we develop offer different trade-offs between performance and resource utilization, making them suitable for a range of deployment scenarios from mobile devices to more powerful computing platforms.

This work contributes to the field of computer vision by establishing a new paradigm for model integration that could serve as a blueprint for future efforts in creating efficient, high-performance vision systems. Furthermore, our findings advance the understanding of architectural design principles that enable effective model scaling across different computational constraints, potentially enabling broader adoption of advanced computer vision capabilities in resource-constrained environments.

Contents

Listing of figures

Listing of tables

Listing of acronyms

CNNs . . . . . Convolutional Neural Networks

ViTs . . . . . Vision Transformers

MSA . . . . . Multiheaded self-attention

MLP . . . . . Multilayer Perceptron

LN . . . . . Layer normalization

CDC . . . . . Consecutive Dilated Convolutions

LGFI . . . . . Local-Global Features Interaction

Abs Rel . . . . . Absolute Relative Error

Sq Rel . . . . . Square Relative Error

RMSE . . . . . Root Mean Square Error

RMSE . . . . . Root Mean Square Error

RMSE log . . . Logarithmic Root Mean Square Error

1 Introduction

Monocular depth estimation is a well-established problem in computer vision, with applications ranging from autonomous driving to augmented reality. Despite the progress made over the years, many existing models struggle to achieve a balance between accuracy and computational efficiency. This thesis focuses on exploring the integration of two models, Xi-Net and Lite-Mono, to address these limitations and improve the practicality of depth estimation systems.
The primary motivation behind this work is to investigate whether combining the strengths of these two models can lead to a more efficient solution without significantly compromising performance. Xi-Net, designed for mobile and edge devices, offers a lightweight architecture with reduced computational demands. Lite-Mono, on the other hand, is known for its high accuracy in depth estimation, particularly in self-supervised settings. However, both models have their limitations: Xi-Net sacrifices some accuracy for efficiency, while Lite-Mono, despite its high performance, is not particularly efficient in terms of computational resources.
By integrating these models, we aim to develop a system that strikes a better balance between accuracy and efficiency, making it more suitable for real-world applications where computational resources are limited. To evaluate our approach, we use the KITTI dataset, a widely used benchmark in autonomous driving research. This dataset provides a variety of real-world scenarios, allow-

ing us to test the robustness and generalization capabilities of our models.
The two variants of the integrated model developed in this work offer different trade-offs between performance and computational requirements. While they may not achieve state-of-the-art efficiency, they represent a step forward in making depth estimation more practical for deployment on resource-constrained devices. This thesis contributes to the field by exploring how the integration of different architectures can lead to more balanced solutions, even if they are not yet fully optimized for efficiency.
The thesis is organized as follows: Chapter 2 discusses the challenges and background of monocular depth estimation. Chapter 3 provides the theoretical foundations, covering convolutional neural networks, transformers, and attention mechanisms. Chapters 4 and 5 detail the architectures of Lite-Mono and Xi-Net, respectively. Chapter 6 describes the methodology, including the dataset and experimental setup. Chapter 7 presents the results of our experiments, and Chapter 8 concludes the thesis with a summary of findings and potential future directions.

2 Problem Definitions

2.1 Challenges in Monocular Depth Estimation

The ability to perceive depth from visual information represents one of the most fundamental capabilities of biological vision systems. Although humans naturally excel at inferring depth from a single eye, replicating this capability in an artificial system has proven to be a significant challenge that has evolved dramatically over the past five decades. The development of monocular depth estimation reflects not only the advancement of computer vision techniques but also our growing understanding of how to approach the complex problem if inferring three-dimensional structure from two-dimensional images.
The first attempts to estimate depth from monocular images emerged in $1970s$, with researchers focusing on understanding and replicating the monocular signals that humans use for depth perception. Horn's[1] pioneering work introduced the Shape-from-Shading technique, which attempted to recover surface orientation by analyzing the gradual variation of shading in an image. While groundbreaking, this approach struggled with real-world scenarios where illumination conditions were complex or unknown. During the same period, Gibson's work on texture gradient analysis attempted to estimate depth by analyz-

ing how texture patterns appear to compress and scale with distance, but faced significant limitations when dealing with multiple textures or irregular patterns common in natural scenes.

The $1990s$ and early $2000s$ saw the emergence of more sophisticated geometric approaches. The Make3D system, developed by Saxena et el. in 2009, [2] represented a significant advancement by combining multiple monocular cues and using a Markov Random Field model to estimate depth. However, its reliance on planar surface assumptions and high computational requirements limited its practical applications. These traditional approaches, while innovative, highlighted the fundamental challenges of using hand-crafted features and explicit geometric assumptions to capture the complexity of real-world scenes. The transition to deep learning approaches marked a revolutionary turning point in the field. The seminal work by Eigen et al. in 2014, "Depth Map Prediction from a Single Image using a Multi-Scale Deep Network"[3], demonstrated that convolutional neural networks could learn to predict depth directly from RGB images without explicit geometric reasoning. This breakthrough approach utilized a multi-scale architecture that could capture both fine details and global scene context, achieving unprecedented performance on standard benchmarks.

The success of deep learning in monocular depth estimation can be attributed to several key factors. First, the ability of neural networks to learn hierarchical representations automatically eliminated the need for hand-crafted features, allowing models to capture complex depth cues that had proven difficult to encode explicitly. Second, the availability of large-scale dataset, such as NYU Depth V2 and KITTI, provided the substantial amount of training data necessary for deep learning approaches to succeed. Third, advances in network architectures, particularly the introduction of encoder-decoder structures and skip connections, enabled models to preserve fine spatial details while capturing global context.

The rapid evolution of deep learning approaches continued with significant innovations in network architecture and training strategies. The introduction of fully convolutional networks (FCNs) by Long et al.[4] improved the spatial resolution of depth predictions, while residual networks enabled the training of deeper architectures for better feature extraction. DORN (Deep Ordinal Regression Network) by Fu et al.[5] reformulated depth estimation as an ordi-

nal regression problem, leading to more accurate predictions across different depth ranges.

Perhaps most significantly, the development of self-supervised learning techniques has begun to address one of the field's most persistent challenges: the need for extensive ground truth depth data. In particular, some work demonstrated that neural networks could learn depth estimation using only stereo image pairs as supervision, without explicit depth measurements.

While deep learning models have significantly improved in their ability to capture global scene structure, they often struggle to maintain precise depth discontinuities and fine geometric details. This limitation becomes particularly apparent in applications requiring high-precision depth maps, such as augmented reality or robotic manipulation tasks. Recent architectural innovations, including attention mechanisms and multi-scale feature fusion approaches, have shown promise in addressing this challenge, but further improvements are required.

Looking toward the future, several promising research directions emerge that may address these fundamental challenges. The integration of physical priors and geometric constraints into deep learning frameworks represents one particularly promising avenue. By incorporating explicit knowledge about scene geometry and physical constraints, models may become more robust and reliable while requiring less training data. Recent work in physics-informed neural networks has demonstrated the potential of this approach in other domains, and its application to depth estimation warrants further investigation.

The development of more efficient architectures for real-time applications represents another crucial direction for future research. While current systems can achieve impressive accuracy, many require significant computational resources that limit their practical deployment on edge devices or in resource-constrained environments. The exploration of neural architecture search techniques and hardware-aware model design may lead to more efficient solutions that maintain accuracy while reducing computational requirements.

Self-supervised and weakly-supervised learning approaches continue to show promise in reducing the dependence on large annotated datasets. The ability to learn from unlabeled video sequences or readily available stereo pairs could significantly expand the applicability of depth estimation systems. Recent work

in this direction has demonstrated increasingly sophisticated training strategies that leverage multiple self-supervision signals, including optical flow, surface normals, and semantic consistency.
The emergence of transformer-based architectures in computer vision has introduced new possibilities for monocular depth estimation. These models' ability to capture long-range dependencies and global context could address some of the limitations of traditional convolutional approaches. Early results using vision transformers for depth estimation have shown promising improvements in handling complex scenes and maintaining consistency across large spatial regions.
Integration with other computer vision tasks represents another promising direction for future research. By jointly learning depth estimation alongside tasks such as semantic segmentation, surface normal estimation, and camera pose estimation, models may develop more robust and coherent scene understanding capabilities. This multi-task learning approach could lead to more reliable systems that better mirror human visual understanding.
As we look to the future, the field of monocular depth estimation stands at an exciting junction. While significant challenges remain, the combination of new architectural approaches, improved training strategies, and integration with other computer vision tasks suggests a path toward more robust and practical depth estimation systems. The continued development of these technologies will be crucial in enabling new applications in augmented reality, autonomous navigation, and computational photography.

2.2 Aim of this work

The aim of this thesis is to explore and develop novel approaches for integrating the strengths of two distinct computer vision models: Xi-Net and Lite-Mono. Specifically, this work aims to investigate the feasibility of combining Xi-Net's mobile-oriented architecture with Lite-Mono's high-performance characteristics to create an efficient and versatile solution. Through this research, we seek to address the fundamental challenge of achieving optimal performance while maintaining computational efficiency suitable for resource-constrained envi-

ronments.

To achieve these objectives, we will develop distinct versions of an integrated model, detailed in subsequent chapters, each offering different trade-offs between performance and computational requirements. Our approach leverages the architectural advantages of both source models: Xi-Net's mobile-optimized design principles and Lite-Mono's superior performance characteristics. This integration aims to bridge the gap between high-performance computer vision systems and practical deployment constraints.

A critical aspect of this research will be the systematic evaluation of various architectural fusion strategies, examining how different components from both models can be effectively combined while preserving their respective strengths. This involves careful analysis of computational bottlenecks and performance characteristics across different operational scenarios.

To ensure robust evaluation of our proposed models, we will utilize the KITTI dataset, a comprehensive benchmark suite specifically designed for autonomous driving applications. The KITTI dataset provides diverse real-world scenarios, including urban, residential, and highway environments, making it particularly suitable for assessing the performance of our integrated models. This dataset choice enables direct comparison with existing state-of-the-art models and ensures our evaluation meets industry-standard benchmarks.

Furthermore, this work will focus on quantifying the performance-efficiency trade-offs in both proposed model variants. The evaluation metrics will align with those established by the KITTI benchmark, allowing for standardized performance assessment and meaningful comparisons with existing solutions.

The anticipated contributions of this research extend beyond the immediate technical achievements. First, our work aims to establish a new paradigm for model integration that could serve as a blueprint for future efforts in creating efficient, high-performance computer vision systems. Second, by demonstrating successful fusion of mobile-optimized and high-performance architectures, we hope to advance the field's understanding of architectural design principles that enable effective model scaling across different computational constraints.

3

Theoretical Foundations

3.1 Depth Estimation

Depth estimation is a fundamental problem in computer vision, with numerous practical applications in fields such as robotics, augmented reality, and autonomous driving. Traditionally, depth information was acquired using specialized hardware like LiDAR or stereo systems, which require expensive and complex equipment. In recent years, however, deep learning techniques, particularly convolutional neural networks, have enabled depth estimation directly from monocular or stereo images, making the process more accessible and cost-effective.

Despite the success of CNNs, challenges remain, particularly when dealing with complex scenes or ambiguous depth cues. To address these limitations, Vision Transformers (ViTs) have emerged as a promising alternative, offering advantages in capturing long-range dependencies and handling more complex depth estimation tasks.

This chapter will explore the use of CNNs and ViTs in depth estimation, and introduce a hybrid approach that combines the strengths of both models to improve accuracy and robustness in diverse real-world applications.

3.1.1 CNNs for Depth Estimation

Convolutional Neural Networks have revolutionized the field of computer vision, providing advanced solutions for a variety of applications, including the estimation of depth from monocular images. This problem, known for its complexity and inherent ambiguity, has found in CNNs a powerful tool for extracting three-dimensional information from two-dimensional data.
Depth estimation using CNNs is a field that has seen various approaches and architectures develop over time, each with distinct characteristics in terms of accuracy, efficiency, and applicability in real-world scenarios. The main approaches can be divided into two categories: depth estimation from single images (monocular) and depth estimation from stereo image pairs. Although both approaches aim to produce dense depth maps, they differ in the input data and the techniques used for training and estimation.
In the case of monocular depth estimation, the goal is to extract information about the distance of objects in a scene from a single image. The main challenge here is that a single two-dimensional image does not contain explicit depth information, necessitating a sophisticated understanding of scene geometry. The architecture based on CNN for monocular depth estimation focus on learning high-level representations that allow depth to be inferred from visual cues such as texture, shading, and perspective. A foundational example in this field is the work of Eigen et al.[3], who proposed a CNN for depth estimation from monocular images. The network, designed to be fully convolutional, produces a depth map directly from the image, but has shown some limitations in terms of accuracy, especially in complex scenarios. Subsequently, improvements such as residual networks have been introduced, allowing for refinement of depth estimation by learning the differences between real and estimated depths, thereby reducing errors and improving precision in more challenging scenes. The approach of Ma et al.[6] is an example of how residual networks have contributed to improving performance in this context, enabling better handling of ambiguities in the mapping between the image and depth.
However, depth estimation from stereo images leverages the principle of disparity between two images taken from different angles, from which depth can be derived through a process known as triangulation. Convolutional neural

networks applied to this type of data are designed to learn the correspondence between pixels in the two images and calculate the disparity, which is then transformed into a depth map. This approach, while generally more accurate than monocular estimation, requires a pair of images and can be limited in cases where only a single image is available. An example of a CNN application in this context is the work of Liu et al.[7], who developed a dual network for depth estimation from stereo images, capable of handling scenarios with multiple exposures, such as those found in HDR 3D applications. CNN-based stereo matching networks have the advantage of effectively addressing the problem of point correspondence, which is one of the main challenges in stereoscopy, and improve the quality of depth maps, particularly under difficult lighting conditions.

Another interesting development in this field is the introduction of self-supervised approaches for depth estimation. These approaches do not require annotated depth maps for training the network, but instead leverage geometric or temporal information to infer depth. An example of such an approach is provided in 2020 by Fang et al.[8], who proposed a self-supervised method for monocular depth estimation, using scene geometry to generate indirect supervision during training. This type of approach represents a significant evolution as it reduces the need for manually annotated datasets, which are time consuming and resource-intensive to create.

In a better perspective, depth estimation using CNNs has seen continuous evolution, with improvements in architectures and the introduction of new techniques leading to significant advances in accuracy and applicability. However, despite these successes, depth estimation remains a complex challenge, especially in real-world scenarios where lighting conditions, object variety, and scene complexity can negatively impact performance. In addition, computational efficiency remains a limiting factor, particularly when real-time estimation is desired.

As research progresses, CNN-based models are evolving to address these challenges, and one of the most promising directions is the adoption of Vision Transformers (ViTs) that we discuss in the following chapter.

3.2 Vision Transformers and Attention Mechanisms

Transformers represent a revolutionary encoder-decoder architecture first introduced by Vaswani et al. in their paper "Attention Is All You Need" [9]. Initially designed for sequence-to-sequence tasks in natural language processing, transformers have since become a cornerstone of modern deep learning due to their remarkable ability to capture global dependencies. This ability to model relationships across long distances, without the limitations of local receptive fields, has led to their widespread adoption across various domains, including computer vision tasks such as monocular depth estimation.
The architecture processes visual input through a mechanism fundamentally different from traditional CNNs. While CNNs operate by scanning local regions of an image with convolutional filters, transformers, specifically the Vision Transformer variant (Illustrated in Figure 3.1), divide the image into patches and model direct relationships between all spatial locations. This global ap-

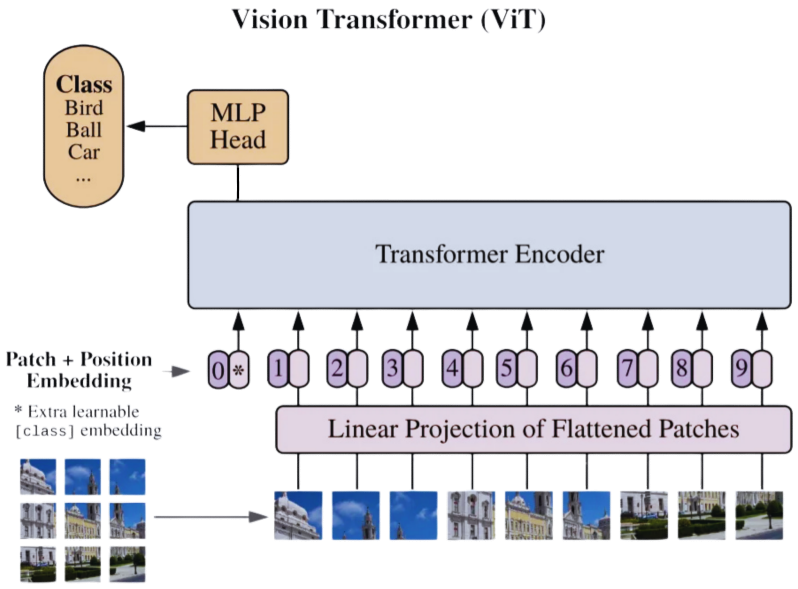

Figure 3.1: Vision Transformer

proach enables transformers to capture the complex spatial dependencies that

are crucial for tasks like depth estimation, where understanding the full spatial context and geometry of a scene is essential.
In particular, to handle 2D images, the Vision Transformer first reshapes the image $x \in \mathbb{R}^{H\times W\times C}$ into a sequence of flattened 2D patches $x_p \in \mathbb{R}^{N\times(P^2\times C)}$, where (H, W) represents the original image resolution, C is the number of channels, (P, P) is the resolution of each image patch, and $N = HW/P^2$ is the resulting number of patches, which also serves as the effective input sequence length for the Transformer. These patches undergo processing through a trainable linear projection (Eq. 3.1) to create patch embeddings of constant dimension D.
Similarly to sequence processing in Transformers, we prepend a learnable embedding to the sequence of embedded patches ($z_0^0 = x_{\text{class}}$), whose state at the output of the Transformer encoder (z_L^0) serves as the image representation y (Eq. 3.4).
During pre-training and fine-tuning, a classification head is attained to z_L^0. The classification head is implemented by an MLP with one hidden layer at the pre-training time and by a single linear layer at the fine-tuning time. To maintain crucial spatial information, the system incorporates position embeddings added to the patch embeddings. The model employs standard learnable $1D$ position embeddings, as more complex $2D$-aware position embeddings have not demonstrated significant performance advantages.
The complete input sequence can be expressed as:

$$z_0 = \left[x_{\text{class}}; x_p^1\mathbf{E}; x_p^2\mathbf{E}; \ldots; x_p^N\mathbf{E}\right] + E_{\text{pos}}, \quad \mathbf{E} \in \mathbb{R}^{(P^2\cdot C)\times D}, \mathbf{E}_{\text{pos}} \in \mathbb{R}^{(N+1)\times D} \tag{3.1}$$

While, The Transformer encoder (illustrated in Fig. 3.2 consists of alternating layers of multiheaded self-attention (MSA, Eq 3.2) and MLP blocks (Eq.3.3). Each layer applies LN before every block, and residual connections are added after every block (Wang et al., 2019[10]; Baevski Auli, 2019[11]).

$$z'_\ell = \text{MSA}(\text{LN}(z_{\ell-1})) + z_{\ell-1} \qquad \ell = 1, \ldots, L \tag{3.2}$$

$$z_\ell = \text{MLP}(\text{LN}(z'_\ell)) + z_\ell, \qquad \ell = 1, \ldots, L \tag{3.3}$$

$$y = \text{LN}(z_L^0). \tag{3.4}$$

The Vision Transformer implements minimal image-specific inductive bias

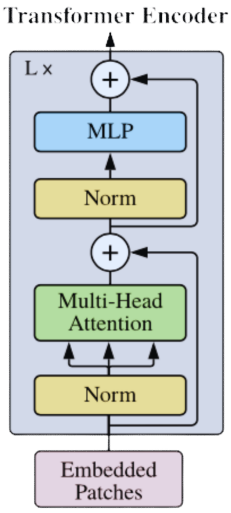

Figure 3.2: Trasformer encoder

compared to CNNs, with only MLP layers maintaining local and translational equivariance while self-attention layers operate globally. The $2D$ neighborhood structure comes into play only when initially dividing the image into patches and when adjusting position embeddings for different resolution images during fine-tuning. A hybrid architecture variant exists where input sequences can be formed from CNN feature maps, with the patch embedding projection applied to patches from a CNN feature map. In a special case, 1×1 patches can be used, effectively flattening the spatial dimensions of the feature map.The cornerstone of this architecture lies in its self-attention mechanism, which we will examine next. This mechanism enables the model to overcome the traditional limitations of CNN-based architectures by dynamically weighting the importance of different image patches. Through this approach, the model can focus on relevant spatial relationships regardless of their distance in the image, making it particularly effective for tasks requiring understanding of full scene geometry, such as monocular depth estimation.
In the following discussion of self-attention, we'll explore how this mechanism allows transformers to excel in tasks that require understanding global context and scene structure.

3.2.1 Attention Mechanisms

Self Attention

Self-attention emerged as a groundbreaking mechanism in natural language processing, fundamentally designed to uncover the intricate web of relationships between words within sentences. By assigning carefully calculated weights to each word, this sophisticated approach enables more nuanced and accurate predictions in language tasks. Building upon these foundational principles, researchers have ingeniously extended this concept to the realm of computer vision, where they discovered that similar mechanisms could effectively capture long-range dependencies between pixels across visual content. This elegant translation from language to vision introduced the concept of the non-local operator, a sophisticated computational construct that computes responses at any given position by synthesizing information from the entire input space. In the paper "Non-local Neural Networks" by Xiaolong Wang et al.[12], the self-attention mechanism is adapted to computer vision tasks by framing it as a specific instance of the non-local operation. The non-local operation is designed to capture long-range dependencies in visual data, such as relationships between distant pixels in an image or frames in a video sequence. This is achieved by computing a weighted sum of features from all positions in the input, where the weights are determined by the affinity (or similarity) between positions. The self-attention mechanism relies on three fundamental components: queries (Q), keys (K), and values (V). These components are derived from the input features through learned linear transformations and play distinct roles in the computation. Queries represent the position (or feature) for which we want to compute the output, and each query is compared against all keys to determine the relevance (or affinity) of other positions to the target position. Keys represent the positions (or features) that are compared against the query, and the similarity between a query and a key determines the weight assigned to the corresponding value. Values represent the actual features that are aggregated to produce the output, and they are weighted by the affinities computed between queries

and keys. Mathematically, the self-attention mechanism can be expressed as:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \tag{3.5}$$

Here, QK^T computes the pairwise affinities (or similarities) between queries and keys, and the scaling factor $\sqrt{d_k}$ (where d_k is the dimensionality of the keys) is used to prevent the dot products from growing too large, which can lead to unstable gradients. The softmax function normalizes the affinities into a probability distribution, ensuring that the weights sum to 1. The resulting weights are then used to compute a weighted sum of the values V, producing the final output. In the context of the non-local operation, the self-attention mechanism is implemented by first transforming the input feature map x into queries Q, keys K and values V using learned linear transformations:

$$\begin{aligned} Q &= W_Q x, \\ K &= W_K x, \\ V &= W_V x. \end{aligned} \tag{3.6}$$

where W_Q,W_K,W_V are learnable weight matrices. The affinities between all pairs of positions are computed using the dot product between queries and keys:

$$f(x_i, x_j) = Q_i^T K_j \tag{3.7}$$

which measures how much attention the model should pay to position j when processing position i.These affinities are normalized using the softmax function to produce attention weights:

$$\alpha_i j = softmax\left(\frac{Q_i^T K_j}{\sqrt{d_k}}\right). \tag{3.8}$$

Finally, the output for each position i is computed as a weighted sum of the values V:

$$y_i = \sum_j \alpha_i j V_j. \tag{3.9}$$

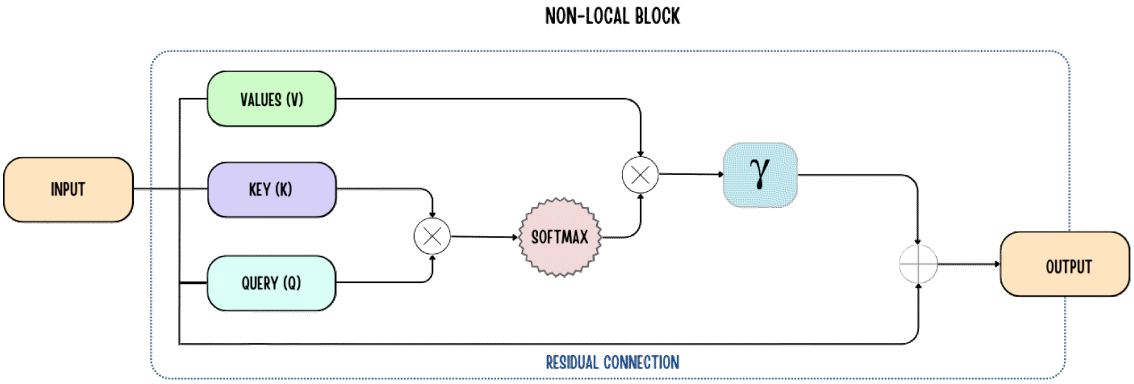

Figure 3.3: Non-Local Block

So, the first matrix multiplication $Q * K$ computes affinities, determining the attention weights that quantify the influence of each pixel (the column of K) on the pixel being analyzed (the row of Q).These affinities are normalized using softmax to obtain the final attention weights A. The subsequent multiplication $A * V$ creates a weighted aggregation of the information, resulting in the attention matrix Y. This attention matrix Y is multiplied by a learnable weight γ (equivalent to a $1*1*1$ convolution)), controlling how much of the computed attention is applied. The weight γ can start from zero, stabilizing the model's training initially. Finally, a residual connection is added by performing a tensor sum between the input (the residual) and the result of the attention computation, ensuring the model retains original features while incorporating global context. This entire flow is visually represented in the figure 3.3, illustrating how the non-local block integrates self-attention into computer vision tasks. The non-local operation (and by extension, self-attention) is applied in various computer vision tasks, including action recognition in videos, where it captures temporal dependencies between frames, object detection, where it models relationships between distant objects in a scene, and semantic segmentation, where it improves pixel-level predictions by considering global context. After introducing the Self-Attention (SA) mechanism, we now delve into its extension, Multi-Head Self-Attention (MHSA, illustrated in Figure 3.4), a key element of Vision Transformers (ViTs). MHSA represents a significant evolution that enables the model to learn diverse representations from the same input information, substantially improving its ability to capture relationships between image

patches. The fundamental principle lies in calculating the attention across multiple parallel "heads," each equipped with distinct projection parameters. From a mathematical perspective, MHSA is defined as:

$$MHSA(Q, K, V) = Concat(head_1, head_2, \ldots, head_h)W^O \tag{3.10}$$

where each attention head is computed as:

$$head_i = Softmax\left(\frac{Q_i K_i^T}{\sqrt{d_k}}\right) V_i. \tag{3.11}$$

Each attention head is calculated considering the query (Q), key (K), and value matrices (V), obtained by linear projections of the input. The key dimension (d_k) is used for normalization, while a final projection matrix (W^o) optimally combines the results of different heads. This mathematical framework supports the central idea of MHSA: Each head can develop the ability to focus on different aspects of the image, allowing the model to extract richer and more diversified representations.

The Multi-Head Self-Attention architecture demonstrates notable advantages over traditional Self-Attention. The presence of multiple heads significantly enhances the model's learning capacity, enabling simultaneous capture of various image characteristics. This parallel approach not only increases representation quality but also contributes to better generalization: combining different perspectives on the data strengthens model robustness and minimizes the risk of overfitting.

The model's expressiveness is considerably amplified through the use of multiple attention heads, enabling the learning of more complex data structures compared to what is possible with a single head. A particularly relevant aspect is the ability to capture multi-scale relationships: the MHSA architecture allows simultaneous analysis of different spatial aspects of the image, leading to a more sophisticated understanding of both local details and global features.

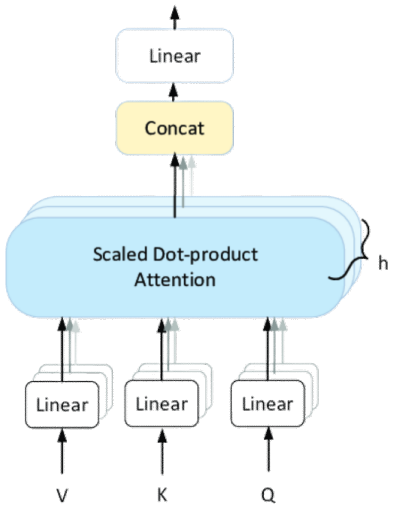

Figure 3.4: Multi-Head Self-Attention

Convolutional Block Attention Module

The Convolutional Block Attention Module (CBAM), introduced by Woo et al.[13] in their seminal paper "CBAM: Convolutional Block Attention Module", represents a significant advancement in attention mechanisms for Convolutional Neural Networks (CNNs). This innovative module combines channel attention and spatial attention, enhancing the ability of CNNs to focus on relevant features within feature maps. Unlike other attention mechanisms, CBAM stands out for its computational efficiency, adding only a limited number of parameters to the network without relying on complex matrix multiplications. To provide a clear visual understanding of the CBAM architecture, Figure 3.5 illustrates the overall structure, showing how the input feature map is sequentially processed by the channel attention module and the spatial attention module, ultimately producing a refined feature map with a greater focus on important features. CBAM builds on prior work, such as the Squeeze-and-Excitation (SE) modules proposed by Hu et al.[14], but extends them by introducing a dual attention mechanism that processes both channel-wise and spatial relationships within feature maps. This sequential approach, in which channel attention precedes spatial attention, has proven particularly effective in improving model performance on various computer vision tasks, such as im-

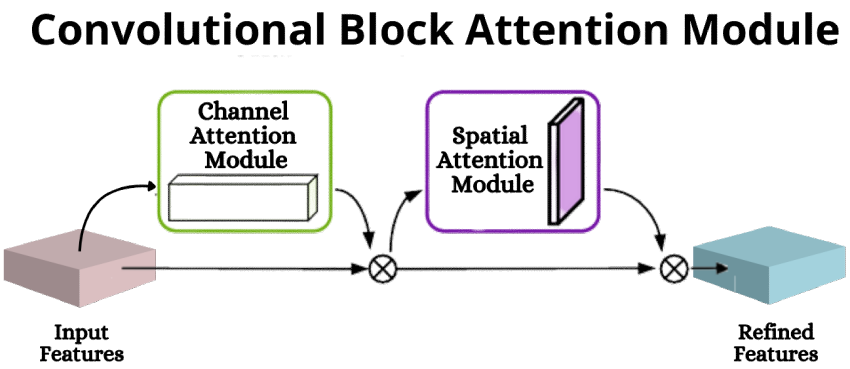

Figure 3.5: Overview of CBAM Architecture

age classification, object detection, and semantic segmentation.
The channel attention module operates on the spatial dimensions of the input tensor, using average pooling and max pooling to generate two vectors representing the average and maximum values of each channel, respectively. These vectors are then processed by a three-layer Multi-Layer Perceptron (MLP), with a compression factor applied in the intermediate layer. The results are summed and passed through a sigmoid function to generate the final channel attention vector. This process allows the network to emphasize the most informative channels while suppressing the less relevant ones. For a detailed visual breakdown of this mechanism, Figure 3.6 provides a diagram that illustrates the channel attention process, including the pooling operations, the MLP and the application of the attention vector to the input feature map. However, the spatial

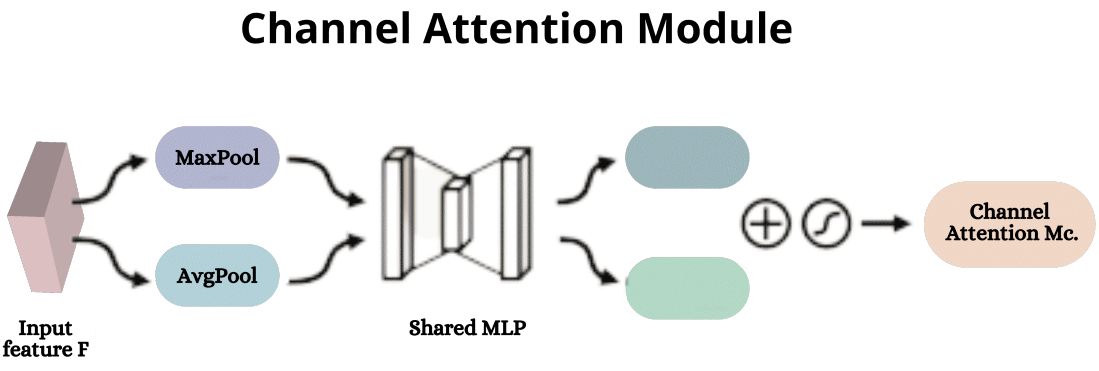

Figure 3.6: Channel Attention Mechanism

attention module operates on the depth of the tensor, using average pooling and max pooling along the channel axis to generate two matrices representing the average and maximum values for each spatial location. These matrices are concatenated and passed through a 3x3 convolution, followed by a sigmoid, to produce a spatial attention map. This mechanism enables the network to focus on the most significant regions within the feature maps. Following the channel attention mechanism, CBAM implements a spatial attention module that provides fine-grained spatial refinement of features. As illustrated in Figure 3, this module operates by analyzing relationships in the depth dimension of the feature tensor, employing a complementary approach to the channel attention. The module generates two descriptor matrices through parallel average-pooling and max-pooling operations along the channel axis. These matrices capture different aspects of spatial information: average-pooled features represent the overall distribution of spatial responses, while max-pooled features highlight the most prominently activated locations. As shown in the figure 3.7, these complementary spatial descriptors are combined through concatenation and processed using a lightweight 3x3 convolutional layer. The resulting output passes through a sigmoid activation function, producing a refined spatial attention map that enables the network to selectively emphasize or suppress different regions of the input feature map based on their relative importance. One of the

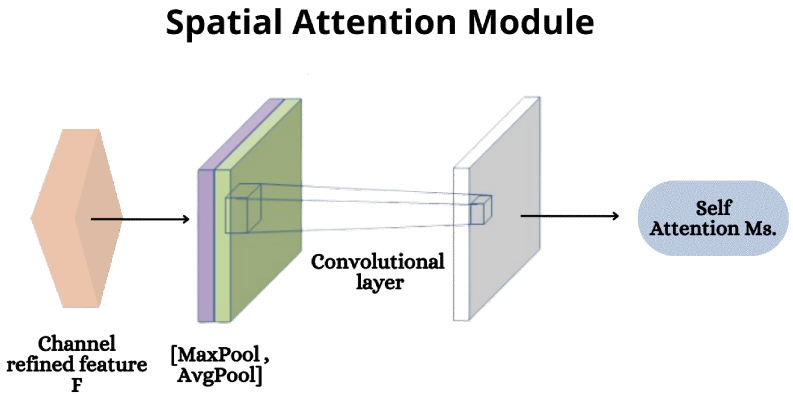

Figure 3.7: Spatial Attention Mechanism

distinguishing features of CBAM is its seamless integration with existing architectures, such as ResNet. Specifically, the module can be placed immediately

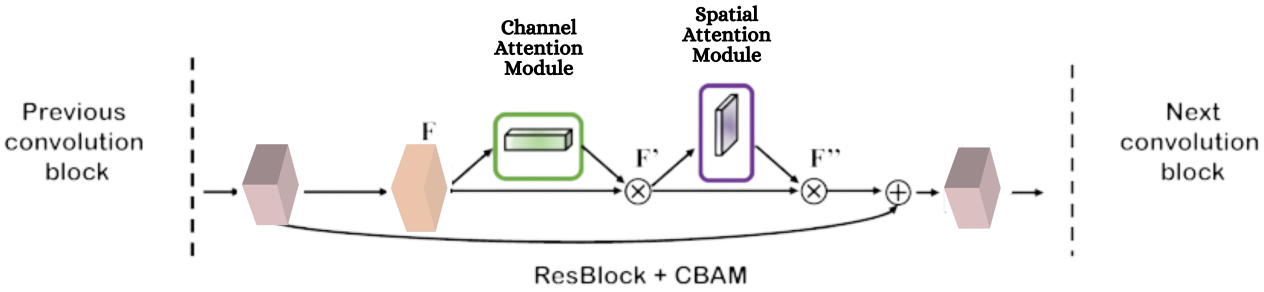

Figure 3.8: Integration between ResNet and CBAM

after a convolution, and its output can be summed with the residual of the initial input, following the typical ResNet approach. This makes CBAM particularly suitable for enhancing the performance of deep convolutional networks, stabilizing learning, and improving computational efficiency. To illustrate this integration, Figure 3.8 provides a schematic showing how CBAM is incorporated into a ResNet block, highlighting the residual connection and placement of CBAM after the convolution layer.

The effectiveness of CBAM has been demonstrated in numerous studies. For example, Park et al.[15] showed that ResNets enhanced with CBAM consistently outperform their baseline counterparts in various computer vision tasks. Furthermore, the flexibility of CBAM has led to several variants and improvements, such as the version proposed by Kim et al.[16], which incorporates temporal attention for video processing, and the lightweight variant by Chen et al.[17], which reduces computational overhead while maintaining performance benefits.

CBAM has significantly influenced the development of subsequent attention mechanisms, becoming a benchmark for more complex approaches. Its combination of channel attention and spatial attention has paved the way for new research, including integration with transformer architectures, efficient implementation for mobile devices, and extension to 3D vision tasks. These developments suggest that the principles introduced by CBAM will continue to be relevant in the design of neural architectures, particularly in contexts that require fine-grained feature refinement.

3.3 Hybrid CNN-Transformers

The evolution of computer vision architectures has witnessed a significant shift with the introduction of Vision Transformers (ViT).
Although approaches based only on transformers have shown remarkable results, they often lack the inherent inductive biases that make Convolutional Neural Networks (CNNs) so effective for visual tasks. This limitation has led to the emergence of hybrid architectures that aim to combine the strengths of both approaches. Traditional CNNs excel at capturing local patterns and hierarchical features through their inherent inductive biases, whereas transformers excel at modeling long-range dependencies through their self-attention mechanisms. The motivation behind hybrid architectures is to take advantage of these complementary strengths while mitigating their respective weaknesses.
A significant contribution to hybrid architectures came from Guo et al.[18] with their introduction of CMT (Convolutional Neural Networks Meet Vision Transformers). This innovative approach effectively combines convolutional blocks with attention mechanisms, creating a hierarchical integration of local feature extraction with global attention processing. The architecture seamlessly integrates convolutional blocks for efficient local feature extraction with transformer blocks for modeling long-range dependencies, all unified through a hierarchical feature fusion mechanism. The authors' empirical results demonstrate that this hybrid approach achieves superior performance compared to pure CNN or transformer-based approaches while maintaining computational efficiency.
In parallel development, Wu et al. [19] proposed the Convolutional Vision Transformer (CvT), taking a different approach to hybridization. Rather than treating convolutional and attention mechanisms as separate components, CvT incorporates convolutional operations directly into the transformer architecture. This integration allows for more efficient processing of visual information while maintaining the powerful modeling capabilities of the transformers.
The advantages of these hybrid approaches are substantial. They demonstrate improved feature extraction through the combination of local and global feature processing, enhanced handling of multi-scale information, and superior spatial relationship modeling. The computational efficiency is significantly bet-

ter than that of pure transformers, with reduced complexity and more efficient processing of high-resolution input. Performance benefits include better accuracy on various computer vision tasks, better generalization capabilities, and reduced dependency on large-scale training data.

However, these architectural innovations also present certain challenges. The increased complexity in design and implementation requires careful consideration during development, with more hyperparameters to tune and potentially more challenging training processes. Resource requirements remain significant, with substantial computational and memory demands that require careful optimization strategies.

Looking toward the future, the development of hybrid architectures represents a promising direction in computer vision research. The field continues to explore more efficient integration strategies, automated architecture search for optimal hybrid designs, and specialized variants for specific computer vision tasks. Researchers are actively working to reduce computational requirements while maintaining or improving performance.

Hybrid CNN-Transformer architectures represent a significant advancement in computer vision. By combining the strengths of both CNNs and transformers, these approaches achieve superior performance while addressing some of the limitations of pure approaches. Although challenges remain in terms of complexity and resource requirements, the hybrid approach appears to be a promising direction for future research and applications in computer vision.

The continued evolution of these hybrid architectures suggests a future where the boundaries between traditional CNNs and transformers become increasingly blurred, leading to more efficient and capable vision systems. As research progresses, we can expect further innovations in this space, potentially leading to even more effective solutions for complex computer vision tasks.

4
Lite-Mono

In the paper Lite-Mono: A Lightweight CNN and Transformer Architecture for Self-Supervised Monocular Depth Estimation (Zhang et al., 2023)[20], the authors propose a novel hybrid architecture called Lite-Mono represents a significant advancement in self-supervised monocular depth estimation, combining the efficiency of Convolutional Neural Networks with the ability of Transformers to capture global information. The architecture of this model is designed to balance lightness with effectiveness, reducing the number of parameters without compromising prediction quality.
At the core of the network is DepthNet (represented in Fig.4.1), an encoder-decoder that extracts and refines image features to produce inverse depth maps. The encoder consists of four stages, each incorporating two key components: the Consecutive Dilated Convolutions module and the Local-Global Features Interaction module.

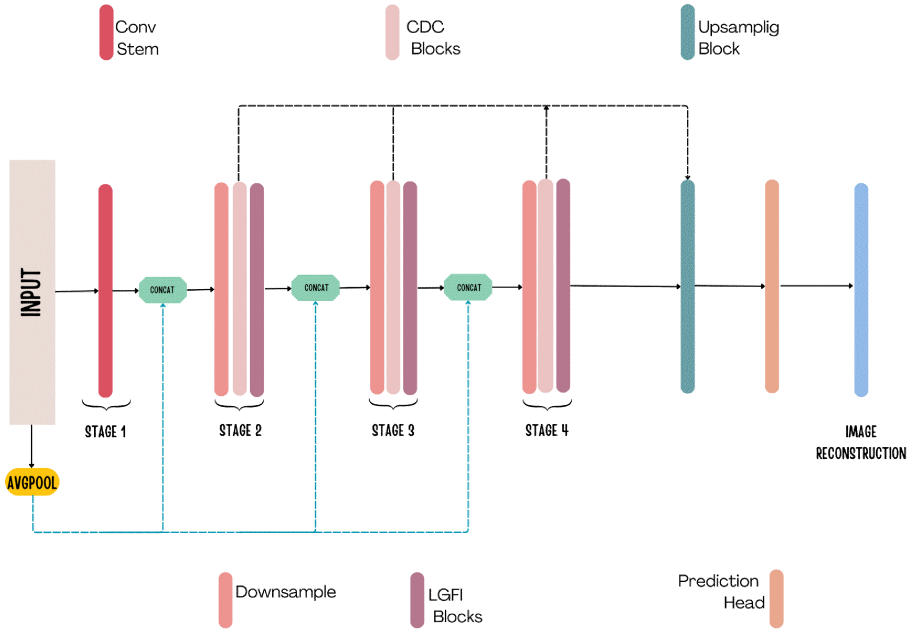

Figure 4.1: Depth Net

The former (reprepresented in Fig.4.2) expands the receptive field without increasing the number of parameters by utilizing consecutive dilated convolutions with progressive dilation rates. This allows the network to capture local information at multiple scales, enhancing its ability to identify objects and structures at different depths.
On the other hand, the LGFI module (represented in Fig.4.2) leverages cross-covariance attention to model global information along feature channels,thus reducing computational complexity compared to conventional Transformers. This approach effectively integrates contextual information from the image, addressing the limited receptive field problem of CNNs alone. In addition, DepthNet adopts cross-stage connections to improve feature propagation and minimize information loss caused by downsampling. The decoder follows a hierarchical architecture, utilizing bilinear up-sampling and convolutions to generate multi-scale depth maps. This approach improves the accuracy of depth estimation by combining low- and high-level information. The PoseNet, in contrast, leverages a ResNet-based architecture to estimate the camera motion

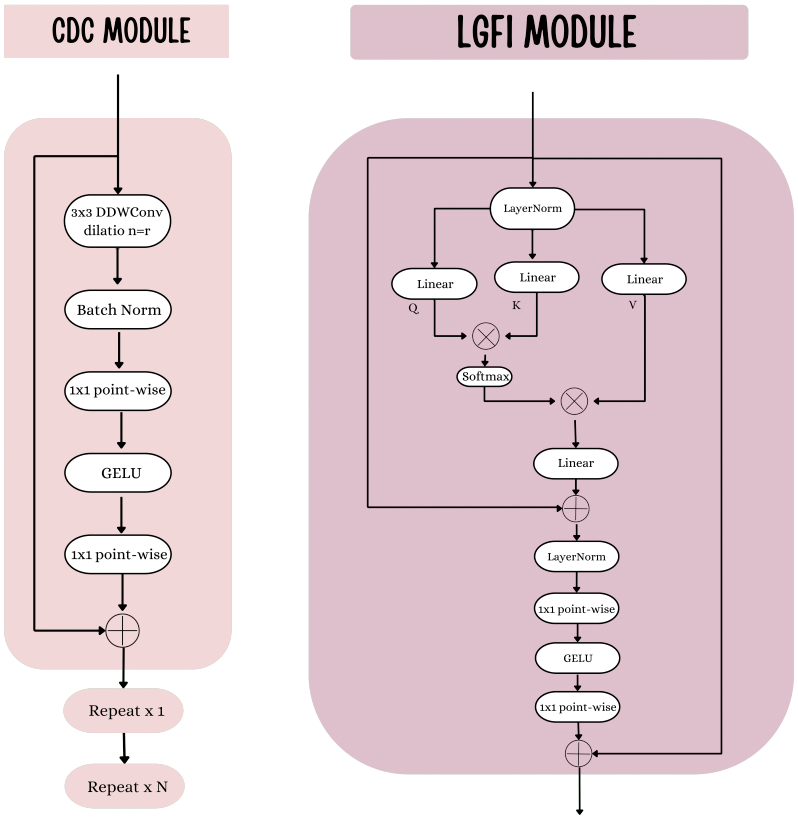

Figure 4.2: LGFI module and CDC module

between adjacent frames, a crucial component for self-supervised training.
Lite-Mono's learning process is governed by a loss function that combines image reconstruction and depth map regularization.
The photometric loss measures the similarity between the target and synthesized image using a combination of the Structural Similarity Index (SSIM) and the L1 distance:

$$L_p(I_t, \hat{I}_t) = \alpha \frac{1 - SSIM(I_t, \hat{I}_t)}{2} + (1 - \alpha)\|I_t - \hat{I}_t\| \tag{4.1}$$

where α is empirically set to 0.85.
To address occlusions and moving objects, the minimum photometric loss is computed as:

$$L_p(I_s, I_t) = \min_{I_s \in \{-1,1\}} L_p(\hat{I}_t, I_t) \tag{4.2}$$

A binary mask μ is used to remove moving pixels:

$$\mu = \min_{I_s \in \{-1,1\}} L_p(I_s, I_t) > \min_{I_s \in \{-1,1\}} L_p(\hat{I}_t, I_t) \tag{4.3}$$

Thus, the final image reconstruction loss is:

$$L_r(\hat{I}_t, I_t) = \mu \cdot L_p(I_s, I_t) \tag{4.4}$$

Additionally, an edge-aware smoothness loss is introduced to ensure spatial coherence in the depth map:

$$L_{\text{smooth}} = |\partial_x d_t^*| e^{-|\partial_x I_t|} + |\partial_y d_t^*| e^{-|\partial_y I_t|} \tag{4.5}$$

where $d_t^* = d_t / \bar{d}_t$ is the mean-normalized inverse depth.
So, the total loss function is then defined as:

$$L = \frac{1}{3} \sum_{s \in \{1, \frac{1}{2}, \frac{1}{4}\}} (L_r + \lambda L_{\text{smooth}}) \tag{4.6}$$

where λ is set to $1e^{-3}$ as in previous works.

Dataset and Experimental Results

Lite-Mono was extensively evaluated using the KITTI dataset, a widely used benchmark in the field of depth estimation. KITTI consists of high-resolution outdoor driving scenes captured using stereo cameras and LiDAR sensors. The dataset includes 39,180 monocular video sequences for training, 4,424 for validation, and 697 for testing. The evaluation metric focuses on Absolute Relative Error (Abs Rel), Root Mean Square Error (RMSE), and depth accuracy thresholds ($\delta < 1.25, \delta < 1.25^2, \delta < 1.25^3$). (For further details see Chapter 6)

Methods	Depth Error (↓)				Depth Accuracy (↑)			Model Size (↓)
	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	Params.
Lite-Mono	0.123	0.898	4.869	0.199	0.854	0.953	0.980	3.1M

Table 4.1: Lite Mono Results[1]

Compared to state-of-the-art methods, Lite-Mono reach higher performance (view Tab.4.1 for more details) with a significantly reduced number of parameters. In particular, it outperforms Monodepth2 with an 80% reduction in trainable parameters while maintaining higher accuracy.
However, despite its lightweight design, Lite-Mono still requires a GPU to function efficiently, as its architecture benefits from hardware acceleration for convolutional operations and Transformer-based processing. The model excels in depth estimation accuracy but is primarily optimized for deployment on devices with dedicated GPUs, making it particularly suitable for applications in robotics, autonomous driving, and augmented reality.

[1]These results were obtained through a personal reproduction of the Lite-mono code, trained on the university's cluster using an Nvidia V100 GPU.

5 XiNet

Building upon the LiteMono architecture discussed in the previous chapter, XiNet [21]introduces significant advancements in neural network design. Although LiteMono focused on fundamental network optimization, XiNet takes a more sophisticated approach by implementing specialized blocks of convolutional layers specifically designed to enhance model efficiency on device deployment. These convolutional blocks represent a carefully orchestrated sequence of layers, engineered to optimize the critical trade-off between performance and complexity by precisely managing the number of operations, parameters, and RAM utilization.

The cornerstone of XiNet's architecture lies in its novel XiConv block, a parameterized convolutional unit that fundamentally transforms the traditional approach to convolutional operations. The XiConv block's architecture is governed by two essential parameters that enable fine-grained control over its computational characteristics. The first parameter, α, serves as the channel reduction coefficient, providing precise control over the network's channel dimensions.

In deep neural networks, channels represent different feature maps of the data being processed. The input channels (C_{in}) correspond to the number of distinct feature maps that enter the convolutional block. For example, in the case of an RGB image, Cin would initially be 3, representing the red, green, and

blue color channels. As data progresses through the network, these channels become more abstract, representing various learned features. The output channels (C_{out}) represent the number of feature maps produced by the convolutional operation, effectively determining how many different features the layer can learn to detect.

For instance, when applying $\alpha = 0.4$ to a XiConv block with initial input channels $C_{in} = 16$ (meaning the block receives 16 different feature maps) and output channels $C_{out} = 32$ (meaning the block would normally produce 32 different feature maps), the architecture dynamically adjusts to reduced dimensions. The actual input becomes $C_{in} = \alpha \times 16 = 6$ channels, and the output becomes $C_{out} = \alpha \times 32 = 12$ channels demonstrating the block's ability to adapt its computational footprint while maintaining essential feature extraction capabilities. This reduction in channel dimensions directly translates to fewer parameters and computations, contributing to the network's overall efficiency.

The second parameter, γ, functions as the compression coefficient and introduces a sophisticated two-phase convolution process that represents a significant departure from conventional convolutional operations. The initial phase implements a strategic compression through a 1×1 pointwise convolution, reducing the channel dimension from αC_{in} to $C_{out}/$. This compression stage is followed by the main convolution operation utilizing a 3×3 kernel, which transforms the compressed representation to the final output dimension of αC_{out}. This two-phase approach, as illustrated in Figure 5.1 , enables the block to maintain feature extraction capability while significantly reducing computational overhead.

A distinctive feature of the XiConv block is its innovative handling of the input signal. Between the compression and main convolution phases, the architecture implements a tensor addition that combines the compressed features with a processed version of the original input. This processing involves a specialized block that performs both channel dimensionality adjustment through pointwise convolution and spatial adaptation via adaptive average pooling. This approach guarantees that crucial data from the input is retained throughout the processing stages, ensuring computational efficiency is upheld. The XiConv block further enhances its feature processing capabilities through the incorporation of a mixed attention mechanism following the main convolution.

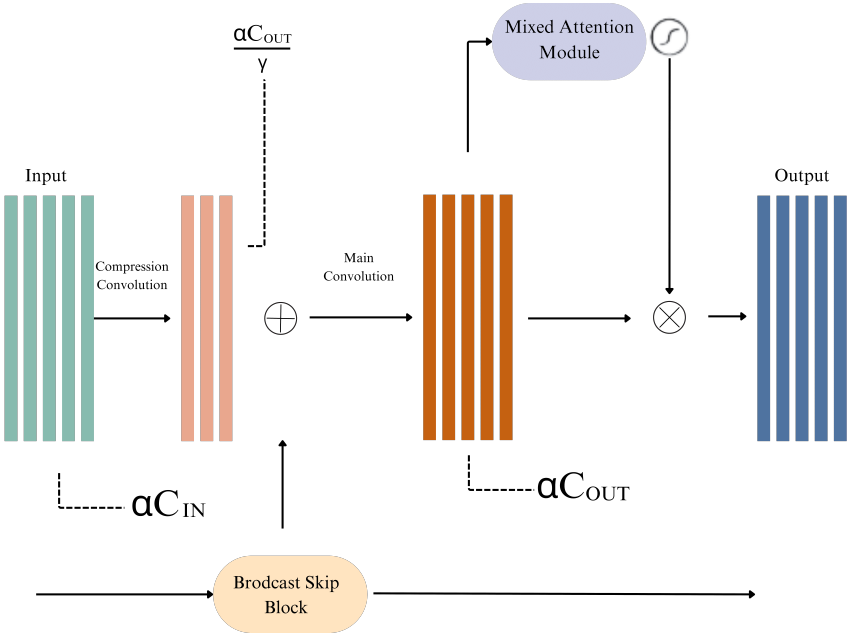

Figure 5.1: XiNet convolutional Block

This attention block implements a hybrid approach that simultaneously considers both channel and spatial attention patterns, applying the resulting attention map through Hadamard product multiplication. This sophisticated attention mechanism enables the block to capture complex dependencies in the data while maintaining computational efficiency.

Within the broader XiNet architecture, these XiConv blocks are strategically arranged to create a hierarchical feature representation through progressive spatial dimension reduction. The network implements a systematic dimensional reduction strategy through the application of stride 2 in alternating XiConv blocks. This arrangement is complemented by a global broadcasting mechanism that distributes the original input information to each XiConv block, enabling the preservation of primary input features throughout the processing pipeline.

The integration of these architectural elements creates a convolutional block design that significantly advances the state of the art in energy-efficient neural network processing. Through its careful parameter selection and innovative feature processing mechanisms, the XiConv block achieves a remarkable balance between computational efficiency and processing capability, representing

a significant improvement over traditional convolutional approaches.
The practical implementation of these blocks requires careful consideration of parameter selection to optimize the balance between computational efficiency and model performance. The selection of α and γ parameters significantly influences the network's capacity and computational requirements, while the block's broadcasting mechanism provides substantial benefits in maintaining access to original input features throughout the network. This architectural approach represents a carefully considered trade-off that becomes particularly significant in resource-constrained environments where energy efficiency is paramount.

6 Methodology

6.1 Dataset

The KITTI dataset represents one of the most relevant and widely used resources in the field of computer vision and autonomous driving. Developed by the Karlsruhe Institute of Technology in collaboration with the Toyota Technological Institute of Chicago, KITTI provides a wide range of real-world data acquired in urban and suburban environments, serving as a fundamental reference for research and the development of advanced perception algorithms.
Data acquisition was carried out using a vehicle equipped with a heterogeneous set of sensors, including stereo cameras, LiDAR scanners, inertial measurement units (IMU), and high-precision GPS systems. This configuration enables a detailed representation of the surrounding environment, providing essential information for object detection and classification, depth estimation, and three-dimensional space reconstruction. The availability of highly accurate, synchronized, and annotated data has allowed algorithm developers to test and validate their solutions in realistic scenarios, contributing to advances in computer vision and autonomous navigation technologies.
The KITTI[22] dataset contains 61 stereo road scenes intended for research in autonomous driving and robotics, and the data were collected using multiple

sensors, including cameras, 3D LiDAR, GPU/IMU units, and others. To train and evaluate proposed methods, the Eigen split [3] is used, which includes a total of $39,180$ monocular triplets for training, $4,424$ for evaluation, and 697 for testing. Self-supervised training is based on known camera intrinsics. To standardize the process, all focal length values of images in the KITTI dataset are averaged, using the same intrinsic parameters for all images during training[23]. In the evaluation phase, predicted depth is restricted to the range $[0, 80]$ meters, as is common practice.

A distinctive aspect of the KITTI dataset is the variety of tasks for which it was designed. Its annotations cover various object categories, such as vehicles, pedestrians, and cyclists, making it suitable for training and evaluating convolutional neural networks for object recognition and tracking. Furthermore, by combining LiDAR data with images, the dataset enables the study of sensor fusion, a key element for the robustness and reliability of autonomous driving systems.

Another significant contribution of the dataset is the ability to test visual odometry and SLAM (Simultaneous Localization and Mapping) techniques. The provided image sequences, combined with depth information and inertial data, allow the development and evaluation of algorithms for localization and environmental mapping. This feature is of fundamental importance for improving the performance of autonomous vehicles, as it enables more precise and safer navigation even in complex and dynamic environments.

The impact of the KITTI dataset on the scientific community is evidenced by the high number of research studies that have used it as an experimental foundation. Its well-organized structure and availability of standardized benchmarks have facilitated comparability between different approaches, fostering the continuous evolution of scene analysis and interpretation techniques. The continuous expansion of computational capabilities and the integration of new deep learning technologies are further enhancing the dataset's potential, making it an essential reference for studies on autonomous perception and mobile robotics.

6.2 Experimental Setup

This section presents the experimental methodology and implementation details of our depth estimation model. The experimental setup was designed to ensure reproducibility while maintaining optimal performance across different hardware configurations.
The implementation framework relies on a carefully configured environment based on Python 3.7, chosen specifically for its compatibility with the deep learning libraries required for this work. To maintain consistency across different systems and avoid dependency conflicts, we established the development environment using Anaconda, a comprehensive package management system. The core deep learning framework consists of PyTorch 1.13 and Torchvision 0.14.
These versions were specifically chosen to maintain compatibility with CUDA 11.7, which provides the necessary GPU acceleration capabilities for efficient model training.
The environment configuration process begins with the creation of a dedicated virtual environment through Anaconda, as shown below:

Bash

```
conda create -n <envName> python=3.7
conda activate <envName>
```

Following environment activation, we installed the comprehensive set of required dependencies:

Bash

```
pip install
linear-warmup-cosine-annealing-warm-restarts-weight-decay==1.0
matplotlib==3.3.4
numpy==1.21.6
opencv==4.5.2.54
pillow==8.3.2
scikit-image==0.15.0
six==1.15.0
tensorboardX==2.1
thop==0.1.1
timm==0.4.12
torch==1.7.1
torchvision==0.14.0 + cu117
wandb
```

Our implementation incorporates Weights & Biases (Wandb), a robust experiment tracking tool that enables comprehensive logging, management, and visualization of experimental results. This addition provides valuable capabilities for monitoring training progress, comparing different model configurations, and maintaining detailed records of our experimental outcomes.
The training procedure was executed through a carefully structured command interface, allowing for precise control over various hyperparameters:

Bash

```
python train.py
–data_path path/to/your/data
–model_name mytrain
–num_epochs 36
–batch_size 12
–mypretrain path/to/your/pretrained/weights
–lr 0.0005 5e − 6 36 0.0001 1e − 5 36
```

The batch size was optimized based on available GPU memory to maximize

computational efficiency while maintaining training stability. The learning rate schedule implemented a dual-phase approach, with an initial warm-up period followed by a gradual decay, helping to prevent convergence to suboptimal solutions.
For performance assessment, we implemented a comprehensive testing framework using the following command structure:

Bash

```
python evaluate_depth.py
–load_weights_folder path/to/your/weights/folder
–data_path path/to/your/data
–model model_name
```

This evaluation methodology provides a thorough assessment of the model's performance, enabling detailed analysis of its strengths and limitations. The combination of the metrics offers a comprehensive view of the model's depth estimation capabilities, facilitating meaningful comparisons with existing approaches in the literature.

6.3 Models

As previously mentioned, the Lite-Mono architecture, though intended to be lightweight, encounters notable hardware compatibility obstacles that constrain its practical utility. While GPU acceleration is necessary for optimal performance, the architecture is largely confined to NVIDIA GPUs due to several limiting components. These constraints arise from the use of GELU activation functions, Reflection2D padding operations, and self-attention mechanisms. To mitigate these compatibility challenges, we initially replaced the GELU activation functions with broadly supported ReLU functions and developed a custom reflection padding operation that preserves its original functionality but is hardware-neutral. In the following table 7.2, we present the derived outcome.

Methods	Depth Error (↓)				Depth Accuracy (↑)			Model Size (↓)
	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	Params.
Lite-Mono	0.123	0.898	4.869	0.199	0.854	0.953	0.980	3.1M
Lite-Mono cust	0.120	0.895	4.817	0.195	0.866	0.955	0.980	3.1M

Table 6.1: Lite Mono Custom Results

However, even with these modifications, when running on a CPU, the processing times remained too slow for real-time applications. These persistent performance constraints led to the development of a modified architecture that incorporates convolutional blocks from the mobile-oriented XiNet model.The details of these models, along with their features and how they work, will be explained and covered in the next sections.

Lite-Xinet

Lite-XiNet represents the first significant attempt to integrate XiNet as an encoder for depth estimation, marking an important step in addressing the hardware compatibility limitations of previous approaches.
In particular way, the encoder processes RGB images through a sophisticated level hierarchy, each utilizing XiEncoder instances for feature extraction.

The Lite-Xinet depth encoder is designed to aggregate multi-scale information across different stages, ensuring effective hierarchical feature extraction. This encoder employs advanced downsampling strategies, dilated convolutions, and normalization techniques to enhance spatial and contextual representation.
The encoding process begins with an input image of size , which initially passes through a downsampling layer based on average pooling with a ratio of three. This approach progressively reduces resolution while mitigating spatial information loss. The data flow then moves through a hierarchical encoding structure composed of multiple levels.
At the second level, the input is processed by the XiEncoder, which incorporates three intermediate convolutional layers. Before transitioning to the next level, a one-by-one convolution is applied to reduce the number. At the third level, the XiEncoder is employed once more which refines the extracted features to improve their representational quality. The final encoding stage, the fourth level, includes an XiEncoder and also a module based on Consecutive Dilated Convolutions, which leverages dilated convolutions to enhance the aggregation of contextual information across multiple scales.
The CDCBlock is designed to extract contextual information at different scales by utilizing dilated convolutions with varying dilation rates. Specifically, the dilation rates allowing for an increased receptive field while maintaining the feature map size. This approach prevents significant loss of spatial resolution. The feature transformation follows a structured process, where batch normalization and depthwise dilated convolutions are applied in sequence. The transformation function is expressed as , where batch normalization ensures stable feature distribution, and depthwise dilated convolutions introduce multi-scale representations without increasing computational complexity.
A crucial aspect of this architecture is the integration of inter-level connections, inspired by the residual connections found in ResNet. This design ensures efficient information flow across encoding stages. After processing at the fourth level, the CDCBlock reduces the number of channels optimizing the features for subsequent upsampling. The refined features are then concatenated with those from previous encoding levels and undergo further transformations. To improve resolution, an Upsizing Block is applied, followed by bilinear interpolation to align feature dimensions across different levels. This step ensures

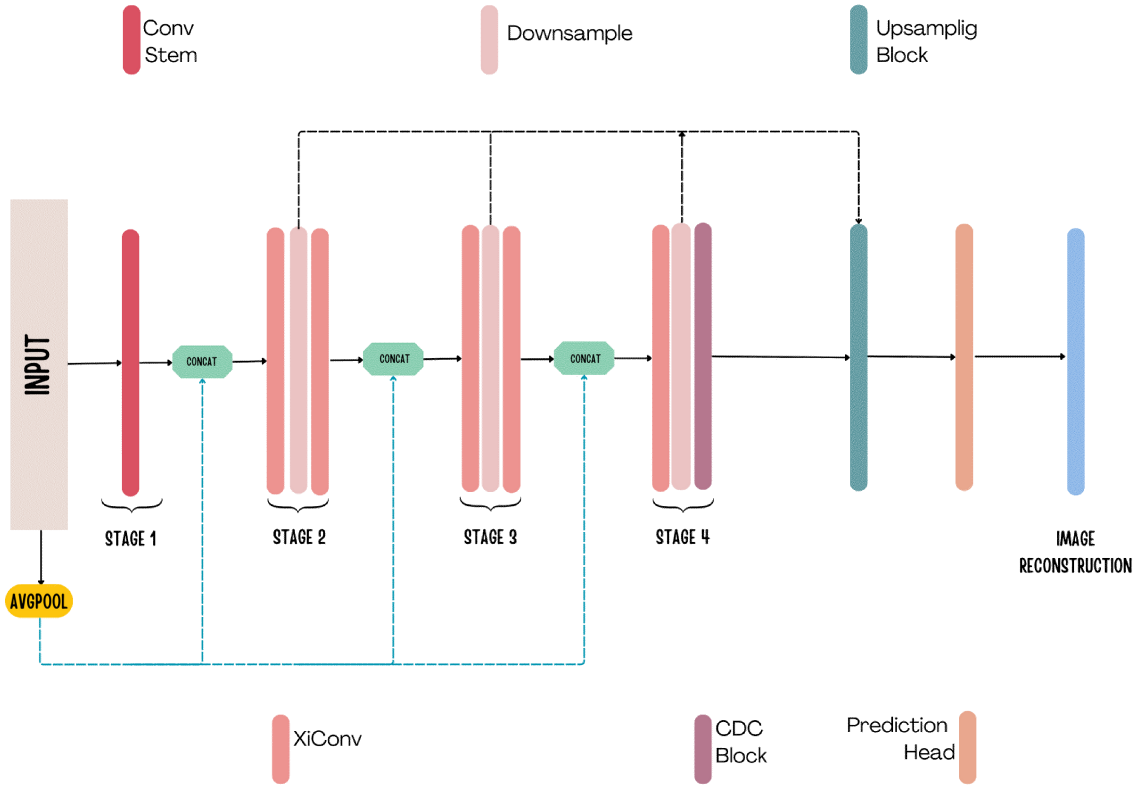

Figure 6.1: Lite-Xinet

consistency in feature representation before further processing.
As shown in Figure 6.1 , the Lite-Xinet depth encoder integrates hierarchical downsampling, dilated convolutions, and inter-level connections to achieve a robust and effective feature representation. This architectural approach enables the preservation of high spatial fidelity while enhancing the learning of contextual features. Through its design, ensures a balance between computational efficiency and representational richness, making it a suitable choice for various depth estimation tasks.

Lite-Xinet α

The integration of XiNet as an encoder for depth estimation represents a significant advancement in addressing the hardware compatibility and computational efficiency challenges present in existing architectures. This chapter examines the modifications made to the traditional pyramid structure and their impact on model performance, with particular focus on the optimization strategies employed to maintain accuracy while reducing computational overhead.
Our research led to the development of Lite-Xinet α (Fig. 6.2), a refined architecture that fundamentally reimagines the traditional pyramid structure of depth estimation encoders. The most significant modification involves the strategic reduction of pyramid levels, specifically the removal of Level 3, resulting in a more streamlined two-level architecture.
The modified architecture implements a carefully calibrated channel configuration, representing a reduction from traditional implementations while maintaining robust feature representation capacity. This optimization directly addresses the computational overhead of previous architectures while preserving the model's ability to capture essential depth information. To bridge the spatial resolution gap created by the removed intermediate level, we implemented bilinear interpolation techniques that ensure smooth feature integration between the remaining levels.
Furthermore, we introduced depthwise separable convolutions to replace standard convolutional operations at strategic points in the network, significantly reducing computational complexity while maintaining feature extraction capabilities.

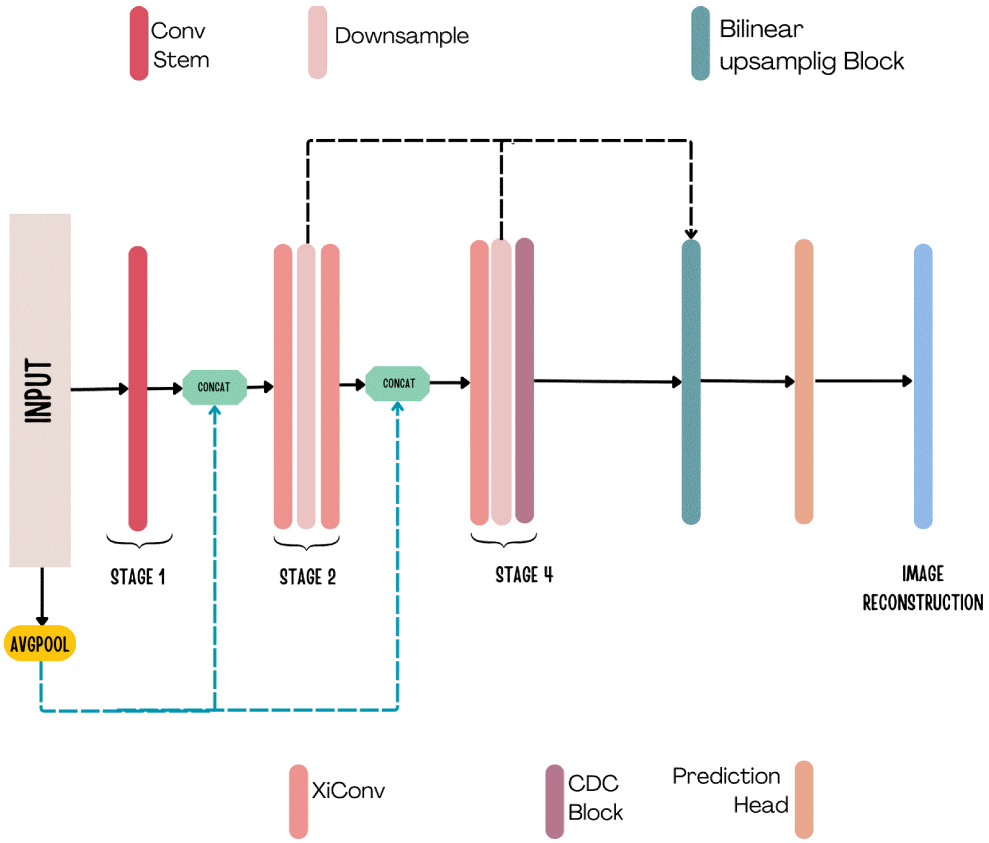

Figure 6.2: Lite-Xinet α

Lite-Xinet β

The following chapter describes the architecture of the PyXiNetBeta, designed to process images and composed of several modular blocks. The architecture leverages advanced image processing techniques, including spatial and channel attention (CBAM), dilated convolutions (CDC), and multi-scale encoding and decoding blocks, with the goal of extracting meaningful features from complex images.
The architecture of PyXiNetBeta is structured into several key components. The encoding blocks, called XiEncoder, are responsible for extracting features at various resolution levels. These blocks are designed to adapt to the input image sizes and produce feature maps that capture both local and global information at progressively lower resolutions. Each encoding level is configured

with specific parameters, such as the number of output channels, to obtain an increasingly abstract representation of the images, ranging from lower resolutions (Level 2 with 32 output channels) to higher resolutions (Level 4 with 96 output channels).
Each encoding block is followed by a convolution block, called the LevelConvolutionsBlock, which further refines the extracted features. This block includes convolution, normalization, and activation operations, all aimed at improving the quality of the feature maps.
A key element of the architecture is the CBAM (Convolutional Block Attention Module) attention module, which enhances the quality of the features through a mechanism that operates both spatially and across channels. The CBAM module calculates weights for each feature channel, emphasizing the most informative ones, and for each spatial position, focusing on the most relevant regions. The output of this module is a weighted feature map that is then processed by the dilated convolution block.
The CDCBlock, or dilated convolution block, captures contextual information at multiple scales. By using dilated convolutions, this block expands the receptive field without significantly increasing the number of parameters. It is applied only at the deepest level of the network (Level 4), where the features are more abstract and rich in contextual information.
To restore the spatial resolution of the feature maps and allow the combination of information from different levels, upsampling blocks are used. These blocks are essential for aligning features at different resolutions before proceeding with concatenation.
The network's processing flow can be broken down into three main stages: feature extraction, application of attention and dilated convolution, and multi-scale fusion with upsampling. First, the input images are processed by the encoding blocks, producing feature maps that are then refined through the convolution blocks. Next, the CBAM attention module and the CDCBlock further refine the features, while multi-scale fusion with upsampling aligns and combines the extracted information at different resolutions.

7 Experiments

7.1 Evaluation Metrics

The assessment of depth estimation models requires specialized evaluation metrics designed to quantify the accuracy of predictions against ground truth depth values. While simple numerical comparisons might seem sufficient, the complexity of depth estimation necessitates a multi-faceted approach to evaluation. This chapter presents a detailed examination of the primary metrics used in the field, including Absolute Relative Error, Square Relative Error, Root Mean Square Error, logarithmic RMSE, and accuracy metrics (threshold accuracies at different levels). Together, these metrics provide a comprehensive framework for assessing model performance, measuring both absolute errors and relative precision across varying depth ranges.

The Absolute Relative Error serves as a fundamental metric in depth estimation evaluation, providing a normalized measure of prediction accuracy. Its mathematical definition is given by

$$AbsRel = \frac{1}{N} \sum \left(\frac{|y_i - \hat{y}_i|}{y_i} \right) \tag{7.1}$$

where y_i represents the ground truth depth, $\hat{y}_i$ represents the estimated depth, and N is the total number of pixels. By dividing the absolute error by the true depth value, Abs Rel effectively normalizes errors across different depth ranges. This metric places particular emphasis on accuracy in close-range predictions, where even small absolute errors can result in significant relative differences. Values are expressed as a proportion of the true depth, making the results readily interpretable in different scenarios.
The Square Relative Error introduces a quadratic penalty for prediction errors, offering several distinct advantages. Its mathematical definition is:

$$SqRel = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{(y_i - \hat{y}_i)^2}{y_i}\right) \tag{7.2}$$

The squared term makes this metric particularly sensitive to large prediction errors, and by maintaining the normalization by true depth, Sq Rel preserves depth-awareness while emphasizing accuracy in challenging cases. Higher Sq Rel values often indicate the presence of significant prediction failures that might be masked by simpler metrics.
The root mean square error provides an absolute measure of the prediction error in the original depth units. It is possible to define the RMSE from a mathematical point of view as:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{7.3}$$

Unlike relative metrics, RMSE maintains the original scale of measurement, offering a comprehensive view of model performance across all depth ranges. As a standard statistical measure, RMSE facilitates comparison with other types of regression models.
The logarithmic variant of RMSE, known as the RMSE log, offers several unique advantages. It is defined as:

$$RMSElog = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(log(y_i) - log(\hat{y}_i))^2} \tag{7.4}$$

By operating in log space, this metric reduces the dominance of errors in far-field predictions. Logarithmic transformation helps achieve a more balanced error distribution across depth ranges, and log space errors often correlate better with human perception of depth differences.
Threshold accuracy metrics ($\delta < t$) provide a different perspective on model performance. In particular, it is provided by the following equation:

$$\delta < t = \frac{1}{N} \sum \left(max \left(\frac{y_i}{\hat{y}_i}, \frac{\hat{y}_i}{y_i} \right) \right) \tag{7.5}$$

where $t \in 1.25, 1.25^2, 1.25^3$. Expressed as percentages, these metrics directly indicate how often predictions fall within acceptable bounds. The use of three increasingly relaxed thresholds provides a nuanced view of prediction accuracy, and these metrics often align well with application-specific requirements for depth estimation accuracy.
An effective evaluation of depth estimation models requires careful consideration of how these metrics complement each other. Using all metrics in combination provides the most complete picture of model performance, as each metric highlights different aspects of prediction quality. Different applications may prioritize certain metrics: robotics applications might emphasize Abs Rel for near-field accuracy, environmental scanning might prioritize RMSE for absolute accuracy, and general-purpose systems might focus on threshold accuracy metrics. When comparing models, it is essential to use consistent metric implementations and evaluation procedures across all comparisons.
In conclusion, the evaluation of depth estimation models requires a sophisticated approach using multiple complementary metrics. Each metric provides unique insights into model performance: relative metrics (Abs Rel, Sq Rel) illuminate the model's ability to handle varying depth scales, absolute metrics (RMSE, RMSE log) provide concrete error measurements in real-world units, and threshold metrics offer intuitive accuracy assessments for practical applications. Understanding these metrics' characteristics and interrelationships is crucial for meaningful model evaluation and comparison in the field of depth estimation.

7.2 Results

In this chapter, we present the results obtained for the different models explored during the experimentation phase. For each model, various configuration parameters were tested to determine the optimal settings. Specifically, the effects of three key hyperparameters were examined: drop path, weight decay, and the number of training epochs.
Drop path is a regularization technique that randomly deactivates entire paths within the neural network during training. This helps reduce dependency on specific paths, improving the model's robustness and making it less prone to overfitting. Unlike the more common dropout, which acts on individual connections, the drop path introduces greater variability by deactivating entire "subnetworks" within the network.
On the other hand, weight decay is a regularization technique that penalizes the model's weights, encouraging them to remain small. This is done by adding a term to the loss function that depends on the L2 norm of the weights, preventing the model from becoming too complex and reducing the risk of overfitting. This approach improves the generalizability of the model, helping it to perform well on unseen data during training.
Each configuration was chosen following a methodical approach, which included a series of preliminary studies aimed at identifying the most promising values for each parameter. The drop path was varied to explore the influence of regularization on model robustness, while weight decay was modified to test the model's generalization ability and prevent overfitting. Finally, the number of epochs was adjusted to assess the impact of the duration of the training on the final performance.
The results presented here were obtained by comparing the different configurations for each model, in order to highlight which combination of hyperparameters offers the best trade-offs in terms of accuracy, generalization, and training time.

7.2.1 Lite-Xinet

We begin the discussion of the results starting with Lite-XiNet, focusing on how modifying the weight decay hyperparameter impacts the model's performance. In particular, beyond the standard configuration, which uses a weight decay of $1e^{-2}$, we have tested several alternative values, including $1e^{-3}$, $5e^{-3}$, and $3e^{-3}$.

The choice to explore these additional values stems from the need to better understand how different magnitudes of regularization affect the model's ability to generalize. A weight decay of $1e^{-2}$ is typically chosen as a reasonable default, balancing the regularization effect while allowing the model to learn from the data. However, experimenting with smaller values can help assess whether a lower regularization strength might lead to better generalization, especially in cases where the model is underfitting. On the other hand, values like $5e^{-3}$ and $3e^{-3}$ were tested to explore the impact of slightly stronger regularization, which could help prevent overfitting in more complex models or in settings where the model is prone to memorizing the training data.

In addition to modifying the weight decay, we also decided to explore different drop path rates, starting from the default value of 0.2. We tested higher values, such as 0.3 and 0.5, to better understand the impact of stronger regularization. A drop path of 0.3 was tested to see if a moderate increase in regularization could improve generalization, while a value of 0.5 was chosen to test the effect of stronger regularization, where more paths are dropped during training.

Furthermore, we also decided to test the impact of increasing the number of epochs, raising it to 50, in order to evaluate whether longer training times would result in better model performance or if the model would simply overfit the training data with extended training.

The results of these experiments, including the variations in weight decay, drop path, and number of epochs, are presented in the table below. These tables highlight the performance of the model for each configuration, allowing us to identify the optimal settings for Lite-XiNet.

Methods	HPs	Depth Error (↓)				Depth Accuracy (↑)			Model Size (↓)
		Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	Params.
Lite-Xinet	default	0.131	0.989	5.196	0.212	0.838	0.947	0.977	6.3M
Lite-Xinet	Weight Decay = $1e^{-3}$	0.133	0.946	5.125	0.212	0.833	**0.978**	0.980	6.3M
Lite-Xinet	Weight Decay = $3e^{-3}$	**0.127**	**0.937**	**5.000**	**0.205**	**0.850**	0.951	0.978	6.3M
Lite-Xinet	Weight Decay = $5e^{-3}$	0.319	2.988	9.977	0.444	0.726	0.843	**0.981**	6.3M
Lite-Xinet	Drop Path = 0.3	0.131	1.007	5.087	0.207	0.845	0.950	0.978	6.3M
Lite-Xinet	Drop Path = 0.5	0.127	0.946	5.023	0.206	0.850	0.951	0.978	6.3 M
Lite-Xinet	Epochs = 50	0.284	2.796	10.040	0.421	0.744	0.891	0.973	6.3M

Table 7.1: Lite-Xinet results

The experiments carried out on the Lite-Xinet model have highlighted the crucial importance of proper hyperparameter configuration to optimize performance in monocular depth estimation. Comparison of different configurations represented in the graph below revealed interesting patterns in how parameters influence model performance. The variant that showed the most significant performance was configured with a weight decrease of $3e^{-}3$, achieving superior results in most evaluation metrics. The experiment with training epochs 50 produced the least satisfactory results, demonstrating the importance of an adequate training period to allow the model to converge to an optimal solution.

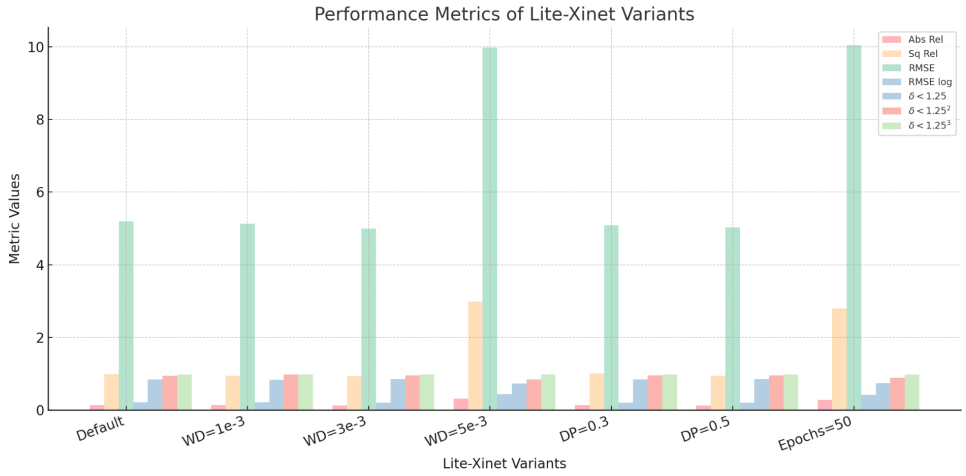

Figure 7.1: Performance Metrics of Lite-Xinet

7.2.2 Lite-Xinet α

Building upon the experimental approach used for Lite-Xinet, we conducted a similar series of hyperparameter investigations for Lite-Xinet α. Our objective was to systematically explore the impact of weight decay, drop path, and training epochs on the model's performance.
The following table presents the comprehensive results of these experiments, highlighting the performance metrics for each configuration of Lite-Xinet α.

Methods	HPs	Depth Error (↓)				Depth Accuracy (↑)			Model Size (↓)
		Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	Params.
Lite-Xinet α	default	0.150	1.163	5.476	0.227	0.813	0.933	0.972	69.02K
Lite-Xinet α	Weight Decay = $1e^{-3}$	0.149	1.176	5.405	0.224	0.814	0.937	0.974	69.02K
Lite-Xinet α	Weight Decay = $3e^{-3}$	0.150	1.190	5.435	0.225	0.813	0.936	0.973	69.02K
Lite-Xinet α	Weight Decay = $5e^{-3}$	0.151	1.226	5.474	0.228	0.811	0.935	0.972	69.02K
Lite-Xinet α	Drop Path = 0.3	0.146	1.148	5.443	0.224	0.816	0.937	0.974	69.02K
Lite-Xinet α	Drop Path = 0.5	**0.136**	**0.956**	**5.265**	**0.222**	**0.817**	**0.938**	**0.975**	69.02K

Table 7.2: Lite-Xinet α results

As shown in Figure 7.2, the experimental results for the Lite-Xinet α demonstrate that the drop path with value configuration 0.5 achieves optimal performance in most metrics. The histogram highlights this configuration's superior error rates and accuracy metrics while maintaining model efficiency at 69.02K parameters. Although the drop path equal to 0.3 marginally outperforms in < 1.25, the general superiority of the drop path $(= 0.5)$ on other metrics establishes it as the most effective variant, significantly outperforming the Weight Decay configurations.

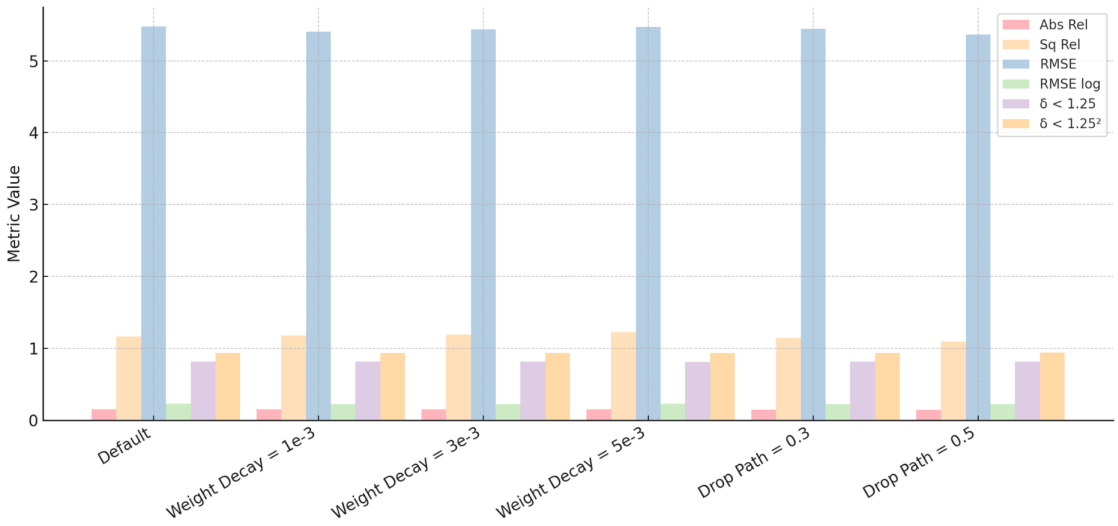

Figure 7.2: Comparison of Depth Estimation Metrics for Lite-Xinet α

7.2.3 Lite-Xinet β

For Lite-Xinet β, we applied a similar experimental methodology.
The experiments systematically varied key parameters to assess their impact on model performance, generalization ability, and potential overfitting risks.

Methods	HPs	Depth Error (↓)				Depth Accuracy (↑)			Model Size (↓)
		Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	Params.
Lite-Xinet β	default	0.130	0.964	**5.054**	0.207	0.847	0.949	0.978	6.03M
Lite-Xinet β	Weight Decay = $1e^{-3}$	0.134	1.037	5.272	0.212	0.839	0.944	0.976	6.03M
Lite-Xinet β	Weight Decay = $3e^{-3}$	0.133	1.085	5.179	0.214	0.837	0.942	0.975	6.03M
Lite-Xinet β	Weight Decay = $5e^{-3}$	0.132	0.973	5.087	0.211	0.840	0.947	0.977	6.03M
Lite-Xinet β	Drop Path = 0.3	**0.127**	**0.954**	5.056	**0.205**	**0.851**	**0.951**	**0.978**	6.03M
Lite-Xinet β	Drop Path = 0.5	0.135	1.023	5.263	0.213	0.835	0.944	0.976	6.03M

Table 7.3: Lite-Xinet β results

Based on the results shown in table 7.3, the Lite-Xinet β configuration with Drop Path $= 0.3$ emerges as the best performing variant, demonstrating superior performance in almost all evaluation metrics.
This configuration achieves the lowest depth error values and the highest depth accuracy scores among all variants tested. The only trade-off is the significantly larger model size compared to other configurations, representing a roughly tenfold increase in the number of parameters.
To facilitate improved visualization, the figure includes a histogram that illustrates the variation in metrics across various configurations. This presentation aids in comprehending the performance disparities between each setup.

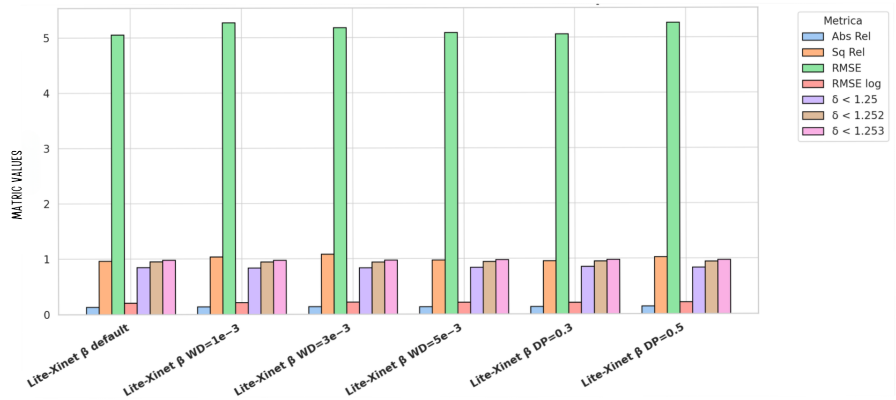

Figure 7.3: Comparison of Depth Estimation Metrics for Lite-Xinet β

7.3 Results Overview

In the following chapter, a brief summary of the quantitative results obtained will be provided to define the best model, followed by the presentation of qualitative results.

7.3.1 Quantitative Results

Our comprehensive evaluation encompasses three distinct variants of the Lite-Xinet architecture, each designed to explore different trade-offs between computational efficiency and depth estimation performance. As we can see in the table below, we have identified the standard Lite-Xinet architecture as the model that achieves the optimal balance between computational resources and performance metrics. The standard Lite-Xinet demonstrates remarkable effective-

Methods	**Depth Error (↓)**				**Depth Accuracy (↑)**			**Model Size** (↓)
	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	Params.
Lite-Xinet	**0.127**	**0.937**	**5.000**	**0.205**	**0.850**	**0.951**	**0.978**	6.3M
Lite-Xinet α	0.136	0.956	5.265	0.222	0.817	0.938	0.975	69.02K
Lite-Xinet β	0.127	0.954	5.056	0.205	0.851	0.951	0.978	6.03M

Table 7.4: Best model results

ness in all evaluation criteria. In terms of depth error metrics, it achieves the lowest values in both Absolute and Square Relative errors among all tested variants. The model's excellence extends to depth accuracy measurements, where it maintains exceptional precision with scores that match or slightly exceed those of other variants.
Among the variants tested, Lite-Xinet α emerges as the best compromise between accuracy and efficiency. Although its performance is slightly lower than that of the standard model, the drastic reduction in model size makes it a highly attractive option for mobile devices and edge devices with limited computational resources. The trade-off in precision may be acceptable in scenarios where efficiency is the priority.
Lite-Xinet β, despite its similar performance to the standard model, does not offer sufficient improvements to justify its implementation. The marginal differences in performance metrics, coupled with its comparable model size, position it as a viable but not superior alternative to the standard architecture.
Given these considerations, Lite-Xinet α emerges as the best, maintaining a well-balanced model size while achieving quite good performance metrics. This optimal positioning makes it the most practical choice for real-world applications where both computational efficiency and accuracy are critical considerations.

7.3.2 Qualitative Results

To provide a more comprehensive understanding of these findings, subsequent sections will present detailed visual comparisons through various qualitative examples (Figure 7.4). These visualizations will help illustrate the nuanced performance differences between model variants, including depth maps. This additional visual context will further reinforce our conclusion about the best model optimal positioning and provide stakeholders with a clear understanding of the trade-offs involved in model selection.

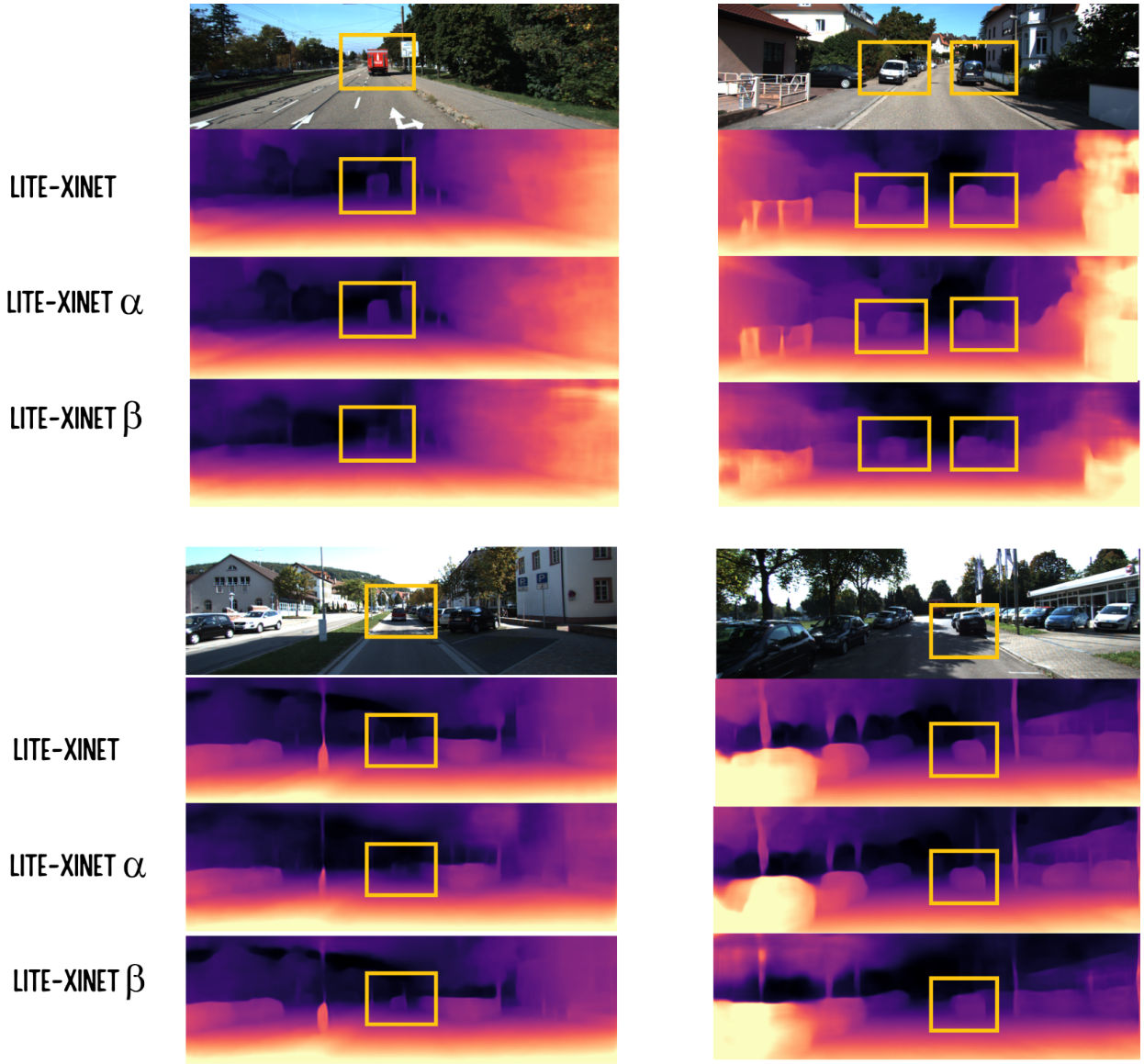

Figure 7.4: Qualitative results

8 Conclusion

This thesis has undertaken an in-depth analysis of the potential offered by architectures based on monocular depth estimation. Research has focused particularly on optimizing computational resources, a crucial aspect for implementation on devices with limited processing capabilities, such as embedded systems and mobile platforms.

The experimental investigation has identified the Lite-Xinet α variant as a solution that provides an acceptable trade-off between estimation accuracy and computational efficiency. Although this implementation presents some performance limitations compared to the full-scale architecture, it is important to emphasize that the advantages in terms of model size reduction and hardware requirements represent a significant step toward the practical applicability of these technologies in resource-constrained environments.

Regarding the Lite-Xinet β variant, the analysis has shown that, despite maintaining performance levels comparable to the standard model, it has not demonstrated sufficiently significant improvements over the α version to justify its adoption. This observation suggests the need to explore further architectural optimization strategies to achieve a more effective balance between estimation accuracy and system efficiency.

The practical implications of the obtained results extend to various application domains, where the ability to estimate depth in real time is a fundamental re-

quirement.
Looking ahead, several promising research directions emerge. One particularly interesting area involves the exploration other learning techniques which are notoriously expensive and time-consuming to produce.
Furthermore, the conducted research highlights that the field of hybrid CNN-Transformer models for depth estimation is still in a phase of rapid evolution, with many open questions that warrant further investigation. Among these, particular importance is given to the optimization of attention mechanisms, improvements in training strategies, and the identification of even more computationally efficient architectures.
In conclusion, this study contributes to a better understanding of the potential and limitations of the models in the specific context of depth estimation, while also providing methodological and practical insights for future research in the field. The collected evidence suggests that, despite existing challenges, this research direction holds promising prospects for the development of increasingly efficient solutions.

References

[1] M. J. Brooks and B. K. P. Horn, "Shape and source from shading," in *International Joint Conference on Artificial Intelligence*, 1985. [Online]. Available: https://api.semanticscholar.org/CorpusID:16497656

[2] A. Saxena, M. Sun, and A. Y. Ng, "Make3d: Learning 3d scene structure from a single still image," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 824–840, 2009. [Online]. Available: https://cs.stanford.edu/~asaxena/reconstruction3d/saxena_make3d_learning3dstructure.pdf

[3] 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 (NeurIPS)*, vol. abs/1406.2283, 2014. [Online]. Available: https://arxiv.org/pdf/1406.2283.pdf

[4] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *CVPR*, vol. abs/1411.4038, 2015. [Online]. Available: https://arxiv.org/pdf/1411.4038

[5] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, "Deep ordinal regression network for monocular depth estimation," in *CVPR*, vol. abs/1806.02446, 2018. [Online]. Available: https://arxiv.org/pdf/1806.02446.pdf

[6] J. Ma, D. Zhang, and Y. Wu, "Monocular depth estimation using cnn-residual network," in *Proceedings of the IEEE International Conference on Computer Vision*, Venice, Italy, Oct. 2017.

[7] Y. Liu, Y. Zhang, and J. He, "Mestereo-du2cnn: A novel dual channel cnn for learning robust depth estimates from multi-exposure stereo images for hdr 3d applications," *arXiv preprint arXiv:2206.10375*, 2022. [Online]. Available: https://arxiv.org/abs/2206.10375

[8] L. Fang, W. Xie, W. Liu, and Z. He, "Towards good practice for cnn-based monocular depth estimation," *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2020. [Online]. Available: https://openaccess.thecvf.com/content_WACV_2020/papers/Fang_Towards_Good_Practice_for_CNN-Based_Monocular_Depth_Estimation_WACV_2020_paper.pdf

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Łukasz Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[10] Q. Wang, B. Li, T. Xiao, J. Zhu, C. Li, D. F. Wong, and L. S. Chao, "Learning deep transformer models for machine translation," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019.

[11] A. Baevski and M. Auli, "Adaptive input representations for neural language modeling," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[12] X. Wang, R. B. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7794–7803. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2018/papers/Wang_Non-Local_Neural_Networks_CVPR_2018_paper.pdf

[13] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "Cbam: Convolutional block attention module," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 3–19.

[14] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7132–7141.

[15] J. Park, S. Woo, J.-Y. Lee, and I. S. Kweon, "Bam: Bottleneck attention module," *arXiv preprint arXiv:1807.06514*, 2020.

[16] D. Kim, S. Woo, J.-Y. Lee, and I. S. Kweon, "Temporal attention mechanism for video processing," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019, pp. 123–132.

[17] Y. Chen, Z. Wang, Y. Peng, Z. Zhang, G. Yu, and J. Sun, "Lightweight cbam: Efficient attention for mobile devices," *arXiv preprint arXiv:2101.06250*, 2021.

[18] J. Guo, K. Han, H. Wu, Y. Tang, X. Chen, Y. Wang, and C. Xu, "Cmt: Convolutional neural networks meet vision transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 12 175–12 185. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2022/papers/Guo_CMT_Convolutional_Neural_Networks_Meet_Vision_Transformers_CVPR_2022_paper.pdf

[19] H. Wu, B. Xiao, N. Codella, M. Liu, X. Dai, L. Yuan, and L. Zhang, "Cvt: Introducing convolutions to vision transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 22–31. [Online]. Available: https://openaccess.thecvf.com/content/ICCV2021/papers/Wu_CvT_Introducing_Convolutions_to_Vision_Transformers_ICCV_2021_paper.pdf

[20] N. Zhang, F. Nex, G. Vosselman, and N. Kerle, "Lite-mono: A lightweight cnn and transformer architecture for self-supervised monocular depth estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2211.13202

[21] A. et al., "Xinet: Efficient neural networks for tinyml," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, accepted to ICCV 2023. [Online]. Available: https://openaccess.thecvf.com/content/ICCV2023/papers/Ancilotto_XiNet_Efficient_Neural_Networks_for_tinyML_ICCV_2023_paper.pdf

[22] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research*,

vol. 32, no. 11, pp. 1231–1237, 2013. [Online]. Available: https://doi.org/10.1177/0278364913491297

[23] C. Godard, O. M. 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, pp. 1951–1960. [Online]. Available: https://doi.org/10.48550/arXiv.1806.01260

Acknowledgments

I wish to convey my profound appreciation to Professor Ballan for affording me the chance to engage in this research endeavor. His guidance and unwavering support have served as a pivotal reference throughout the entire process.
A deep gratitude is also extended to Elena, whose accessibility, expertise, and patience facilitated my substantial development during this project. She willingly bestowed her time and insights, assisting me at every phase and aiding in overcoming the challenges encountered.
I am particularly grateful to my family, whose continuous support, love, understanding, and trust have been invaluable, especially during the most challenging periods. Their presence has profoundly enhanced this journey.
Additionally, I am immensely thankful to my friends for their encouragement and steadfast support at every juncture of this experience.