Friedrich-Alexander-Universität Erlangen-Nürnberg

Chair of Multimedia Communication and Signal Processing

Master Thesis

The Light Weight Monocular Depth Estimation

Mehmet Ömer Eyi

Jun. 2024

Supervisor: Prof. Vasileios Belagiannis

Declaration

I confirm that I have written this thesis unaided and without using sources other than those listed and that this thesis has never been submitted to another examination authority and accepted as part of an examination achievement, neither in this form nor in a similar form. All content that was taken from a third party either verbatim or in substance has been acknowledged as such.

Place, Date	Mehmet Ömer Eyi Spießstraße 29, Nürnberg

Contents

Kurzfassung

Monokulare Tiefenschätzungsnetzwerke werden in erster Linie zur Bereitstellung von Tiefeninformationen in Situationen verwendet, in denen eine direkte Tiefenerfassung entweder nicht verfügbar, unpraktisch oder zu teuer ist. Darüber hinaus kann das Vorhandensein hinreichend genauer Tiefeninformationen mehrere Computer-Vision-Aufgaben im Vergleich zu reinen RGB-Ansätzen verbessern, z. B. die Schätzung der menschlichen Pose und die Objekterkennung. Das Ziel dieser Arbeit ist es, eine leichtgewichtige Encoder-Decoder-Architektur für die Tiefenschätzung auf eingebetteten Geräten vorzustellen. Die jüngste Literatur legt nahe, dass auf Selbstaufmerksamkeit basierende Architekturen, wie der Vision Transformer, das Potenzial haben, Faltungsneuronale Netze zu übertreffen, wenn eine ausreichende Menge an Trainingsdaten vorhanden ist. Im Gegensatz zu Methoden, die auf schwerere Backbones zur Verbesserung der Genauigkeit setzen, konzentriert sich diese Arbeit auf Einfachheit und die Reduzierung der Modellkomplexität. In diesem Zusammenhang wird untersucht, wie man den neuesten Stand der Technik mit einem begrenzten Berechnungsbudget kombinieren kann. Die entwickelten Ansätze werden anhand von Standard-Benchmarks evaluiert und mit verwandten Baselines und bestehenden Ansätzen verglichen.

Abstract

Monocular depth estimation networks are primarily used to provide depth information in situations where direct depth sensing is either unavailable, impractical, or cost-prohibitive. Furthermore, the presence of reasonably accurate depth information can improve several computer vision tasks compared to RGB-only approaches, e.g. human pose estimation and object recognition. The aim of this thesis is to present a lightweight encoder-decoder architecture for depth estimation on embedded devices. Recent literature suggests that self-attention-based architectures, such as the Vision Transformer, have the potential to outperform convolutional neural networks given a sufficient amount of training data. In contrast to methods that rely on heavier backbones to improve accuracy, this work focuses on simplicity and reducing model complexity. In this context, it will be investigated how to combine the latest state of the art with a limited computational budget. The developed approaches will be evaluated on standard benchmarks and compared with related baselines and existing approaches.

Symbols and Notations

σ	Non-linear Function
$\sum$	Sum
Q	Query
K	Key
V	Value
T	Transpose
PE	Positional Encoder
$\odot$	Element-wise multiplication
$\in$	Element of
δ	Threshold Accuracy
L_p	Photometric Loss
P	Patch
μ	Mean
σ_x	Standard Deviation

Abbreviations and Acronyms

CDC Consecutive Dilated Convolution

CNN Convolutional Neural Network

DoF Degrees of Freedom

FLOPS Floating-Point Operations per Second

GELU Gaussian Error Linear Unit

LGFI Local Global Feature Interaction

MHSA Multi Head Self Attention

MLP Multi Layer Perceptron

PATM Phase-Aware Token Mixing

SSIM Structural Similarity Index Measure

Chapter 1

Introduction

Monocular and binocular depth estimation are the two main approaches for depth estimation. The binocular depth estimation is the standard approach for generating depth maps of scenes. By using two cameras, the depth map of a scene can be generated by utilizing triangulation, matching and epipolar geometry [1] in this approach. The epipolar-geometry is used to increase the speed of matching feature pairs on the cameras, whereas triangulation is utilized to extract the depth information of the features. This is the main strategy in binocular depth estimation, and deep learning models are employed to enhance the quality of the binocular depth maps as in [2, 3]. Unlike binocular depth estimation, depth maps of monocular depth estimation cannot be generated by this approach, as illustrated in Fig 1.1. In a monocular depth estimation, triangulation and epipolar geometry cannot be used since disparity information between the cameras is lost. In addition to this, the case where infinitely many 3D scenes having the same 2-D depth map arises. This case creates a projection ambiguity, and this leads to the main challenge in monocular depth estimation. Due to this phenomenon, the monocular depth estimation is an ill-posed problem. However, humans acquire depth perception through navigation and interaction in the real world, which allows them to form plausible depth estimates for unfamiliar scenes[4]. Mimicking this depth perception on the deep learning models may reduce the complexity of models utilized in 3-D tasks in terms of the number of required camera sensors.

The monocular depth estimation can enhance performance in scenarios where crucial depth information is unavailable due to a limited number of cameras. Talking head generation for ultra-low bitrate video conferencing can be given as an example of such a case. Since teleconferencing devices typically have only one camera, utilizing depth information from a single camera enhances the reconstructed video quality at the decoder, as demonstrated by Hong et al [5]. Apart from this, monocular depth estimation is also used in the field of autonomous driving. In this field, the depth map generated from the monocular depth estimation network is fused with the LiDAR data to enhance the performance in object tracking and prediction of the objects' 3D bounding boxes

[6]. Another application field of monocular depth estimation is monocular endoscopy, and the reason behind using monocular depth estimation networks in this field is the limited space in the human anatomical cavities and the cost of the cameras utilized in the endoscopy [7]. Apart from these fields, depth maps also enhance the performance of the models in recognition tasks [8], 3D modeling [9, 10], physics and support models [8], and robotics [11, 12]. These improvements in the performance of the models utilizing the depth networks come from exploiting the geometric relationships between objects of interest and their surrounding scenes.

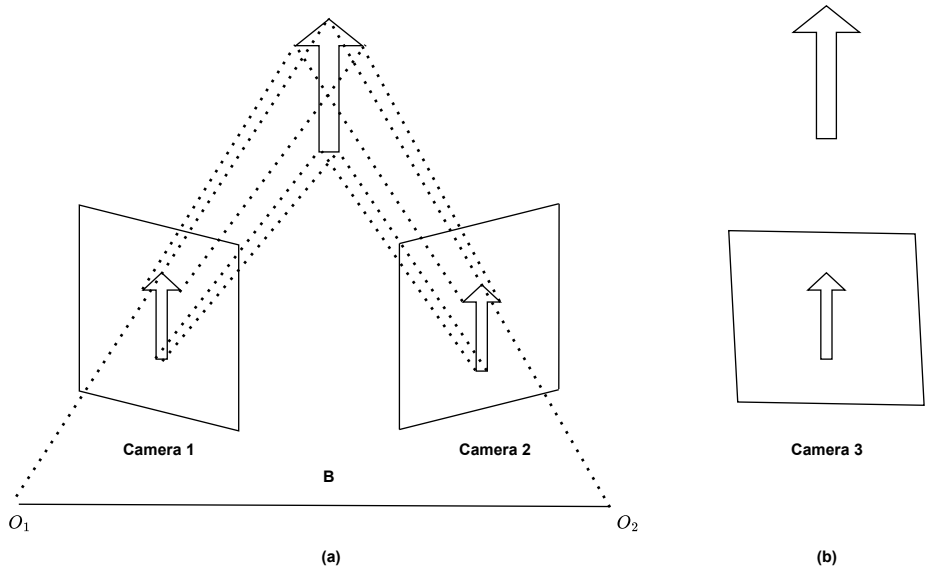


Figure 1.1: The scheme of the (a) binocular and (b) monocular depth estimation. The terms B, O_1, and O_2 denote the baseline between the camera centers, the center of Camera 1 and the center of Camera 2 respectively.

Using edge devices instead of high-powered devices reduces expenses associated with data gathering, processing, and actuation [13]. Considering this, many researchers started to work on lightweight algorithms for object detection, depth estimation and semantic segmentation instead of working on the best-performing ones needing high computation power [14, 15, 16]. Lightweight monocular depth estimation is a commonly used solution for tackling this issue since it uses a single camera and can be run on real-time applications [14, 17, 18]. Due to this issue, the model's parameter size and Floating-Point Operations per Second (FLOPS) employed by the model have also been considered criteria for the performance evaluation. Apart from the performance and complexity concerns, many researchers have adopted encoder-decoder monocular depth estimation schemes to facilitate the modifications in their framework [14, 18, 19].

The encoder in their scheme mainly extracts the features from the RGB input image, whereas the decoder in their scheme mainly creates the final depth map by interpolating the features. Considering this scheme illustrated in Fig. 1.2, vision transformer-based monocular depth estimation [19, 20, 21], and a hybrid architecture combining Convolutional Neural Networks (CNNs) and Transformers [14] are state-of-the-art solutions for lightweight monocular depth estimation. The vision transformer-based monocular depth estimation models have higher complexity than the hybrid architectures since the complexity in vision transformers is much higher than the lightweight attention mechanisms in hybrid models. These lightweight mechanisms make the hybrid models more applicable to real-time applications due to their low complexities. Therefore, this hybrid framework is selected as the baseline architecture while designing the lightweight monocular depth estimation in this thesis.

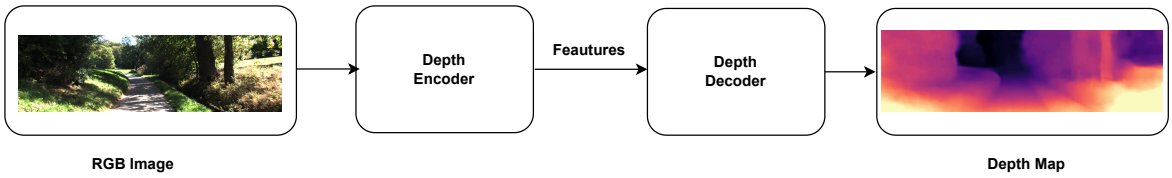


Figure 1.2: The encoder-decoder scheme of the monocular depth estimation

In this thesis, an attention-based up-sampling having a low complexity scheme is proposed while designing the decoder of the monocular depth estimation. Instead of using solely bi-linear up-sampling and 2-D convolutional blocks at the decoder, lightweight attention architectures are utilized for the up-sampling of the outputs of the encoder. Moreover, pixel shuffling is replaced with bi-linear up-sampling to reduce the memory demands and complexity of the model. After designing this up-sampling scheme, different attention-based deep learning architectures and MLP mixer architectures, lightweight substitutes for transformers, are replaced with cross-covariance attention. The performance and complexity of these modified monocular depth estimation networks are evaluated and compared with the state-of-the-art monocular depth estimation networks.

1.1 Problem Description and Objective

The main challenge in designing a monocular depth estimation is providing a low-complex model while maintaining its performance in generating depth maps. Introducing Multi Head Self Attention (MHSA) and vision transformer modules to the

depth model is not desired due to their high complexities. Instead of using these modules, Zhang et al. use cross-covariance attention [14] to alleviate the computation burden in attention layers, whereas Ning et al. use trap attention techniques to tackle this complexity issue [17]. Apart from these recent attempts to solve the complexity problem, novel ways to solve this problem will be investigated in this thesis. In this thesis, Zhang et al.'s lightweight monocular depth estimation network is selected as the baseline model while comparing the performance and complexity of the proposed models. This thesis proposes novel approaches in the encoder and decoder of the depth model. In the encoder of the depth model, Zhang at al.'s attention modules are replaced with lighter-attention and MLP-based modules. The complexity and performance of the modified depth estimation networks are evaluated and compared with this baseline. Apart from these modifications in the depth model's encoder, an up-sampling technique having a low complex attention module is designed for the decoder of the depth model, and the bi-linear interpolation operations in the decoder are replaced with this attention-based up-sampling technique. Pixel shuffle operation is employed to alleviate the complexity of the attention module introduced by this up-sampling technique. This up-sampling technique's impacts on model complexity and performance are investigated.

1.2 Organization of the Thesis

This thesis is composed of 6 chapters. In Chapter 1, a brief overview of the monocular depth estimation is introduced along with the motivation behind using monocular depth estimation. Apart from this, this chapter also introduces the application fields of monocular depth estimation. After this introductory chapter, Chapter 2 focuses on the related works on monocular depth estimation. In this chapter, the recent works on the architectures of the monocular depth estimation networks are presented and explained briefly. After this chapter, Chapter 3 mainly explains the theoretical background needed for designing monocular depth estimation and the lighter attention modules in detail. This chapter begins with the fundamentals of deep learning blocks and continues with the theoretical foundations behind designing a monocular depth estimation network. After this chapter, Chapter 4 demonstrates the architectures of the depth models utilizing the proposed strategies in this thesis. In addition, this chapter introduces pixel shuffle operation to reduce the model complexity. Apart from these, this chapter gives a brief explanation of the datasets utilized in the training and evaluation of the depth models. After introducing the architectures of the modified

depth models, Chapter 5 demonstrates the performance and complexity gap between the modified and baseline depth models. Moreover, the depth maps generated by depth models are visualized. In Chapter 6, future works related to monocular depth estimation and final remarks on the modified networks in the thesis are presented.

Chapter 2

Related Works

In this chapter, the recent works related to monocular depth estimation are presented under two groups which are supervised and self-supervised approaches. This chapter begins with the supervised depth models and continues with the self-supervised depth models. After introducing the recent advances in these approaches, the recent works based on the light attention modules are presented.

Supervised and self-supervised depth models are the two main deep learning models utilized in monocular depth estimation in the literature. The supervised monocular depth estimation models use the ground truth depth maps in the training, whereas the self-supervised depth models do not use these ground truths since these ground truth depth maps are not present. Instead of using these ground truths, the self-supervised deep learning frameworks in monocular depth estimation use the reprojection principle that reconstructs the input image using the depth map and camera pose matrix. Moreover, the pose estimation network estimating the camera pose matrix is used only in the training. The main distinctions between these two approaches are the presence of the ground truth and the loss function. In supervised depth models, the loss function in the training uses the disparity between the generated and ground truth depth map, whereas the loss function utilized in self-supervised models employs the disparity between the original and reconstructed RGB images.

2.1 Supervised Monocular Depth Estimation Models

The loss function in supervised depth models compares the models' depth map prediction with the ground truth of the depth maps in the training. The encoder of these supervised monocular depth models extracts the features from an RGB input image, whereas the decoder reconstructs the depth maps based on these features. While designing the encoder of the model, single-layer and multi-layer feature extraction are used architectures in the supervised monocular depth estimation. In general, VGG [22] and DDRNet-23-slim [23] models pre-trained on the ImageNet dataset are the

encoders used in the single-layer feature extraction [18, 24]. The output of these encoders passed through bi-linear up-sampling blocks and 2-D convolutional layers in the decoder to resize the depth map's resolution and improve the quality of the depth map. Instead of following this scheme in the decoder, Belagiannis et al. propose a guided up-sampling scheme inspired by guided image filtering [25, 26]. In this scheme, the guides are selected as downsampled input RGB images. This scheme is an image-based up-sampling scheme since it considers the downsampled of the images in the up-sampling of the features [18]. This approach does not only increase the quality of the depth maps generated from the monocular depth module but also does not introduce a significant complexity and memory burden to the depth model. Apart from these architectures employing the single-layer feature extractor, the encoders using the multi-layer feature extractions output the feature space with varying feature resolutions. Eigen et al. [27] pioneered this approach. Moreover, leveraging strong scene priors for surface normal estimation [28], translating the depth regression problem into depth classification [29], supervising via virtual planes [30], and incorporating an additional auxiliary loss function [31, 32] are built upon Eigen et al's framework. Apart from these modified models on Eigen et al.'s model, Dong et al. illustrate that edge information is crucial for the convolutional neural networks to estimate depth maps, and they propose a transformer-based feature aggregation module capturing the long-range dependencies between the context information and edge attention features through cross-attention mechanism [33]. Furthermore, Chiu et al. introduce teacher models in the training of their proposed convolutional-based depth model [34]. In their work, the teacher models generate depth maps and semantic segmentation for the training of their proposed model. To reduce the complexity of their depth model, the pixel shuffle operation [35] is used for scaling the depth features [34]. Wofk et al. use MobileNet [36] as their depth encoder, whereas they use separable convolution layers to reconstruct the depth map in the decoder. In their depth model, they employ the NetAdapt pruning scheme [37] to reduce the complexity of their depth model, and they utilize the TVM compiler stack [38] to compile their proposed depth model on NVIDIA Jetson TX2 to reduce the inference runtime [39]. The main obstacles in using supervised schemes are scenarios that lack ground truth depth maps and miss depth data in the ground truth. This missing depth data emerges due to highly reflective surfaces, light-absorbing surfaces or regions outside the sensor's range of measurement [40]. To tackle this issue, Saxena et al. propose a diffusion model to impute the missing depth data in the ground truth depth data [41], and this model is inspired by the success of diffusion models in image generation tasks [42, 43, 44]. Furthermore, Breckon et al. propose a monocular depth

estimation model using synthetic data with domain adaptation via image style transfer. In this work, Breckon et al. use an adversarial network to reduce the discrepancy between the real-world and synthetic data [45]. Moreover, modifying or customizing the depth dataset may still cause problems in the training and evaluation of the supervised depth models. This concern leads to the adoption of self-supervised methods over supervised approaches.

2.2 Self-Supervised Monocular Depth Estimation Models

In self-supervised deep learning approaches, the ground truths of the depth maps are either missing or insufficient for the training of the models. The disparity between the original and reconstructed input image from the depth model is used in the training loss. The reprojection principle uses the camera position, depth map and camera intrinsic matrix, composed of the focal length, aspect ratio, sensor skew, and the principal point (center of projection x and y) [46] to reconstruct the input image [47]. In general, the intrinsic parameter of the camera is known in monocular depth estimation, whereas the position of the camera is missing. To estimate this missing information, a pose estimation network estimates 6 degrees-of-freedom camera coordinates by considering consecutive temporal frames. This pose network is used along with the monocular depth estimation network to reconstruct the input image only in the training. To improve the robustness of the self-supervised depth model against the occlusions and moving objects in scenes, Godard et al. propose re-projection loss against the occlusion problem and auto masking loss to filter out the moving objects having the same velocity as the camera [48]. Apart from these loss terms, multi-task learning schemes related to optical estimation [49] and semantic segmentation [50, 51] are employed to model the dynamic scenes in depth estimation. Moreover, the camera intrinsic matrix can be treated as unknown in the monocular depth estimation. The focal length and principal point along with the translation and rotation can be estimated using a pair of consecutive images as in [52, 53]. Apart from using the reprojection principle. Shao et al. introduce pseudo-depth ground truths generated by a teacher and diffusion model [54]. The teacher model utilized in this work is Zhang et al.'s Lite Mono depth network, and this teacher model is used in the computation of the knowledge distillation loss [54]. Moreover, Shao et. al employ a multi-view check-filter [55] to reduce the negative impact of the pseudo-ground truths on the depth loss calculation [54].

2.3 Attention Modules in Monocular Depth Estimation Models

Considering the supervised and self-supervised approaches to monocular depth estimation, visual transformer-based models generate depth maps having higher quality than the CNN-based models lacking the transformer module [14, 17]. The main bottleneck of using visual transformer-based models is the expensive computations in the MHSA module. Zhang et al. use cross-covariance attention in MHSA to alleviate the complexity of MHSA [14], whereas Ning et al. developed a strategy to lower this complexity by introducing a depth-wise convolution mechanism to capture long-range information. In this strategy, traps are placed in the expanded space around each pixel, and the attention mechanism is built upon the feature retention ratio of the convolution window. With this strategy, the quadratic computational complexity associated with MHSA is transformed into a linear form [17]. Moreover, there are various attempts to simplify the attention architectures and reduce their complexities without performance drop as in [56, 57, 58]. These light attention architectures demonstrate promising performances in object detection tasks on the ImageNet dataset. These light attention modules exploit the interaction between the key and query matrices as in [56, 57], and a linear layer can be employed to model the key and value interaction to reduce the complexity as illustrated in [58]. Furthermore, the SE block [59] that generates coefficients for each channel dimension is employed to reduce the complexity in the attention module, as illustrated in [18]. Other than applying these tricks to the transformers to reduce the complexity, Tolstikhin et al introduce the MLP mixer architecture to tackle the complexity issue in the attention-based transformers. In this architecture, multi-layer perceptions (MLP) are applied iteratively across the spatial and channel dimensions to mimic the attention layers. Apart from this architecture, various architectures as in [60, 61] are built on this architecture to narrow down the performance gap between MLP-based and transformer-based approaches. Apart from these architecture strategies, initializing the weight with the ImageNet object detection dataset enhances the quality of the depth maps as in [14, 19, 48, 18]. Considering this, the lightweight attention and substitute mechanisms employed in object detection can be used in the encoder and decoder of the monocular depth estimation to reduce the complexity without a significant performance drop.

Chapter 3

Background

Chapter 3 introduces the theoretical background behind the deep learning modules while designing a monocular depth estimation network. Firstly, an overview of the commonly used building blocks used in monocular depth estimation networks is introduced. This introduction starts with a brief overview of the essential deep learning blocks which are MLP and Convolution blocks. After this brief introduction, the basic architecture of transformers is discussed along with the architectures of lightweight transformers and their possible substitutes, Mixer-based architectures, which are explained in detail. After this discussion on the architecture of these blocks, the reprojection principle for reconstructing the input frame using the depth map will be explained in detail.

3.1 Multi-Layer-Perception(MLP)

The MLP is a standard feed-forward mapping function $f : X \rightarrow Y$ between the input X and output Y. This structure can be represented by computational graphs with forward pass and backward pass [62]. The forward pass is the process of predicting the desired output by using a weight and bias. The σ in Eq.3.1 is a non-linear function introducing non-linearity into the model. This process is illustrated in Eq. 3.1:

$$\mathbf{y} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}) \tag{3.1}$$

In the backward pass, the gradients of the weight and bias are computed and these gradients are used in updating the weight and bias to minimize the loss function between the model prediction and ground truth.

$$\mathbf{w} = \mathbf{w} + lr \cdot \frac{\partial L}{\partial \mathbf{y}} \cdot \frac{\partial \mathbf{y}}{\partial \mathbf{w}} \tag{3.2}$$

Considering Eq. 3.2, the gradient of the weight is multiplied by the learning rate, denoted by lr, to adjust the magnitude of the weight's gradient during the training

process. This scheme is also known as a fully connected layer illustrated in Fig. 3.1 and is used in the classification layer in Convolutional Neural Network (CNN).

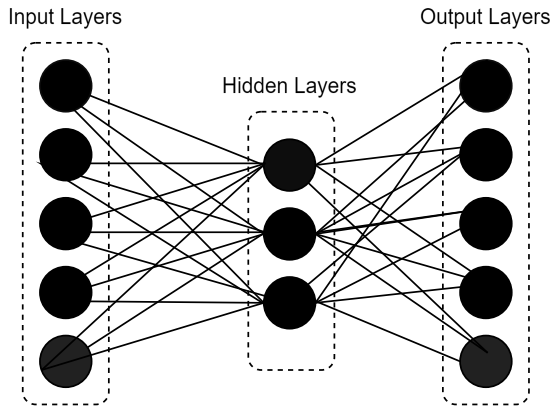


Figure 3.1: The architecture of MLP

3.2 Convolutional Neural Networks

The architecture of convolution neural networks (CNNs) resembles MLPs' architecture. The main distinction between these two architectures is the convolutional layers in CNN. Unlike an MLP, the layers of a CNN have neurons arranged in 3 dimensions: width, height, and channels demonstrated in Fig. 3.2. Instead of multiplying the weight matrix with the image in MLP, various kernels with fixed height and width are slid across the input. While sliding the kernels, the overlapping entries of the kernel and image are multiplied, and then stored in the output matrix. This approach is designed based on convolution operation, demonstrated in Eq. 3.3.

$$(y * w)[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot w[n-k] \tag{3.3}$$

The dimensions of the kernel, stride and padding are used in controlling the dimension of the output volume, The channel dimension of the kernel determines the output channel dimension, whereas the other parameters affect the height and width of the output matrix. The stride is the number of jumps of the sliding kernel. The padding is resizing the input while padding zeros or constants to the outer edges of the input. The output dimension after applying the convolutional layer is demonstrated in Eq. 3.4, 3.5, and 3.6

$$W_{\text{out}} = \left\lfloor \frac{W_{\text{in}} + (2 \cdot P) - (D \cdot (F-1)) - 1}{S} + 1 \right\rfloor \tag{3.4}$$

$$H_{\text{out}} = \left\lfloor \frac{H_{\text{in}} + (2 \cdot P) - (D \cdot (F-1)) - 1}{S} + 1 \right\rfloor \tag{3.5}$$

$$C_{\text{out}} = K \tag{3.6}$$

Where: W_{in} is input width, H_{in} is input height, C_{in} is input channels, F is filter size, P is padding size, S is stride, K is the number of filters, D is the dilation rate, W_{out} is output width, H_{out} is output height, and C_{out} is output channels.

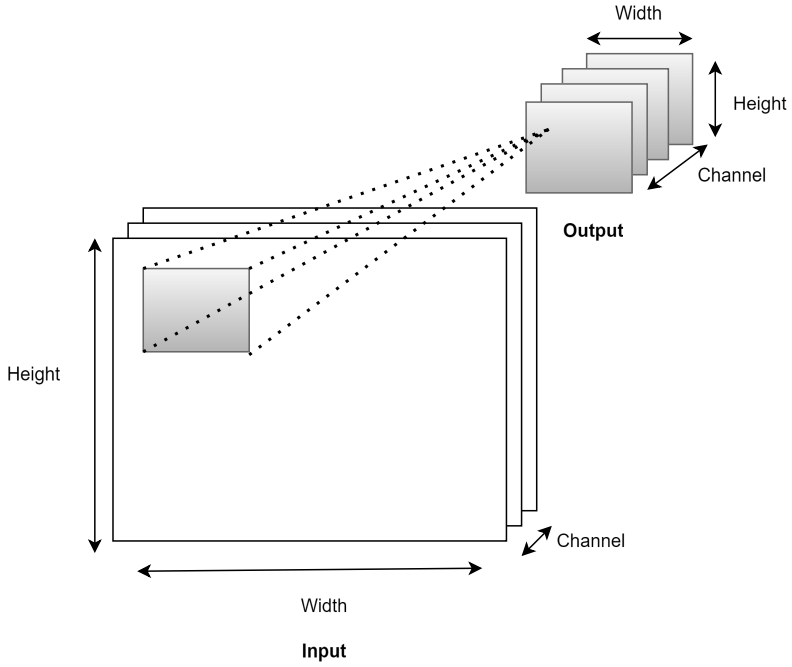


Figure 3.2: The architecture of CNN

In general, after applying the convolutional layer in a classification task, a flattening layer is applied to flatten the dimension of the output. After this flattening layer, a fully connected layer is applied to adjust the output dimension to the number of classification labels. This convolutional approach has several benefits compared with using MLP. This architecture has a lower tendency to lead to overfitting models and computation burden when it is compared with MLP-based architectures.

$$y[i,j] = \sum_{m=0}^{F-1} \sum_{n=0}^{F-1} x[i + m \cdot D, j + n \cdot D] \cdot w[m,n] \tag{3.7}$$

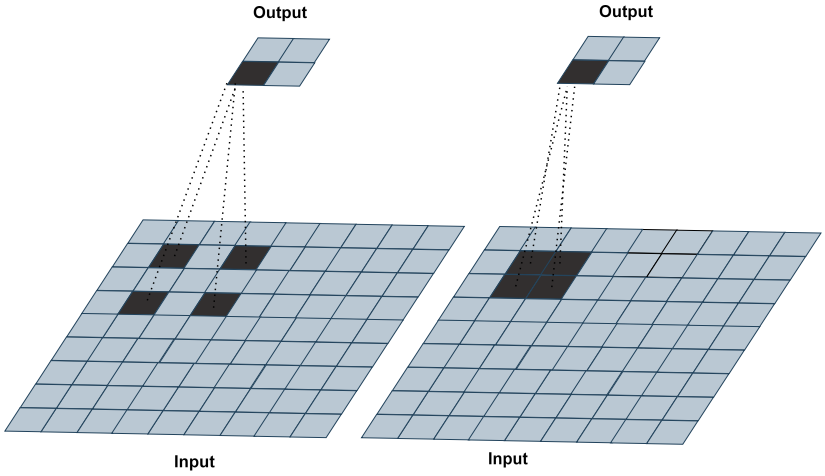


Figure 3.3: Visualization of a dilated convolution block on the left and a standard convolution block on the right.

In Eq. 3.4 and 3.5, the dilation rate expands the kernel by pixel skipping to cover a larger area of the input. As shown in Fig. 3.3, the dilation convolution blocks have larger receptive fields than the standard convolution blocks. The equation of the dilated convolution is demonstrated in Eq. 3.7, where D denotes the dilation rate. The standard convolution blocks can also be considered the dilated convolutional blocks whose dilated rate is 1. Dilated convolution blocks with different dilation rates can be used in extracting the multi-scale local features. Zhang et al. stack these dilation blocks to extract the multi-scale local features in the encoder of their lightweight depth estimation network [14]. After applying Consecutive Dilated Convolution (CDC) layers with varying dilation rates, fully connected layers with the Gaussian Error Linear Unit (GELU) activation function are utilized. Zhang et al.'s consecutive dilated convolution approach is demonstrated in Eq. 3.8 and 3.9. Considering an input feature X with dimension H $\times$ W $\times$ C Zhang et al.'s consecutive dilated convolution block module outputs $\hat{X}$ as in Eq. 3.9 [14]:

$$\mathrm{GELU}(x) = 0.5x\left(1 + \tanh\left(\sqrt{\frac{2}{\pi}}\left(x + 0.044715x^3\right)\right)\right) \tag{3.8}$$

$$\hat{X} = X + \mathrm{Linear}_G\left(\mathrm{Linear}\left(\mathrm{BN}\left(\mathrm{DDWConv}_r\left(X\right)\right)\right)\right) \tag{3.9}$$

In Eq. 3.9, $Linear_G$ denotes a fully connected layer with GELU activation function, whereas BN and $DDWConv_r(\cdot)$ denote batch normalization and 3 $\times$ 3 depthwise dilated convolution with dilation rate respectively.

3.3 Attention Module

In this section, the basics of attention modules and foundations of recent lightweight attention are introduced. The section begins with a brief overview of the self-attention. After this overview, the schemes of transpose self-attention separable self-attention and efficient additive self-attention are discussed.

3.3.1 Self Attention

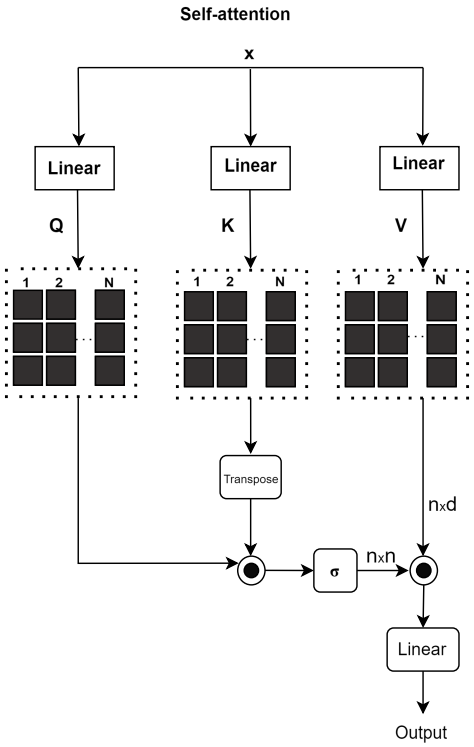


Figure 3.4: The architecture of self-attention

In the context of monocular depth estimation, self-attention-based approaches are used in modeling the interaction between the extracted visual tokens as in [14, 17]. The input to self-attention is composed of n tokens with d-dimensional embedding vectors. This input is projected to query (Q), key (K), and value (V) with the aid of three matrices, W_Q, W_K, and W_V [58]. To reduce the complexity and enable modeling of the interactions from different views, each self-attention layer contains h heads [58]. To model this framework, Eq. 3.10 is utilized.

$$\text{Self-Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q} \cdot \mathbf{K}^T}{\sqrt{d_k}}\right) \cdot \mathbf{V} \tag{3.10}$$

After the computations of Q, K, and V, the scheme in Fig. 3.4 is followed. First, the dot product of Q and K.T is computed and then normalized by the square root of the number of dimensions in the tokens. After this normalization, the softmax activation function is applied in this interaction to convert it into a probability distribution. After this, the final score modeling the interaction among the input tokens is obtained by performing a dot product operation with V. One drawback of utilizing this scheme for modeling the interaction between the tokens is its complexity and memory usage. The main bottleneck in this scheme is the computation of the dot-product of Q and K, which leads to quadratic complexity. The complexity of this self-attention scheme is $O(n^2 \cdot d)$, where n and d denote the number of tokens and hidden dimension in tokens. This quadratic complexity leads to slow inference speed and high memory usage, which makes this scheme unsuitable for real-time applications.

3.3.2 Transpose Self-Attention

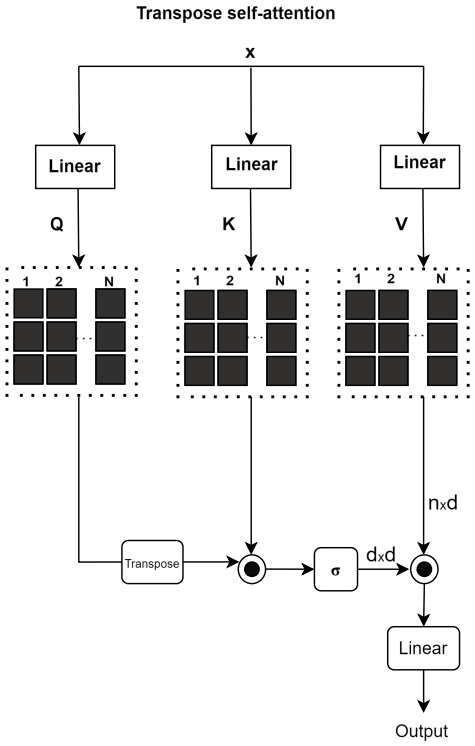


Figure 3.5: The architecture of transpose self-attention

The transpose self-attention, which is also known as cross-covariance attention, is an attention scheme that alleviates the computation and memory burden. This architecture is shown in Fig. 3.5. Instead of using the spatial dimension, using only the

channel dimensions allows us to compute the cross-covariance across the channels to generate attention feature maps that have implicit knowledge about the global representations [56]. The main complexity reduction appears in the dot product between Q and K since it considers only the channel dimensions during this operation. The complexity of this attention model is $O(n \cdot d^2)$, where n is the number of patches, and d is the feature/channel dimension [56]. This attention mechanism is also utilized in Zhang et al.'s monocular depth estimation to model the interaction between the features after consecutive dilated convolutional layers in their Local Global Feature Interaction (LGFI) module [14]. In this monocular depth estimation framework, before applying this attention module, Zhang et al. use a positional encoder, similar to the positional encoder in [63], to encode the positional information of tokens on the semantic information. The formula of the positional encoder in [63] is demonstrated in Eq. 3.11 and 3.12.

$$\mathrm{PE}(pos, 2i) = \sin\left(\frac{\mathrm{pos}}{10000^{2i/d}}\right) \tag{3.11}$$

$$\mathrm{PE}(pos, 2i+1) = \cos\left(\frac{\mathrm{pos}}{10000^{2i/d}}\right) \tag{3.12}$$

As shown in Eq. 3.11 and 3.12, the positional information is encoded using the sinusoidal waves whose frequencies are determined by the channel index (i) and the total number of channels (d). The "pos" term denotes the position of the token in the sequence. The reason behind using such a function is to break the correlation between the channels and positions in the channels. As illustrated in Eq. 3.11 and 3.12, the consecutive channels, $PE(pos, 2i+1)$ and $PE(pos, 2i)$, are not correlated since cos and sin are orthogonal to each other. Moreover, considering the non-consecutive channels, these are also not correlated due to using different frequencies. Considering the positions on the same channel, the position indices having different values break the correlation between the order of the tokens. Similar to this pose encoder, Zhang et al. modify this pose encoder architecture such that it applies this pose encoder to the x and y dimensions of the features extracted from consecutive dilated convolutional blocks. Following this operation, Zhang et al. use a 1x1 convolutional layer to result in the final position information [14].

3.3.3 Separable Self-Attention

Separable self-attention is an attention architecture to alleviate the quadratic complexity in the interaction between Q and K by introducing context vectors. Instead of directly using the Q matrix for Q-K interaction, Q, $n \times d$ matrix, is projected on q, $n \times 1$ vector [57]. After this multiplication, the output is passed through a softmax function to normalize the output. This output is called context scores which are the weights for capturing the importance of the queries. These scores are multiplied element-wise by the K matrix and pooled by summation. After this operation, the context vector modeling of the Q-K interaction is formed. The bottleneck introduced in the self-attention can be alleviated since the dot product between the Q and K matrices is replaced by the element-wise multiplication and pooling. This results in the reduction of quadratic complexity to linear complexity. In Fig. 3.6, the calculations of Q-K interaction in separable self-attention and self-attention are illustrated.

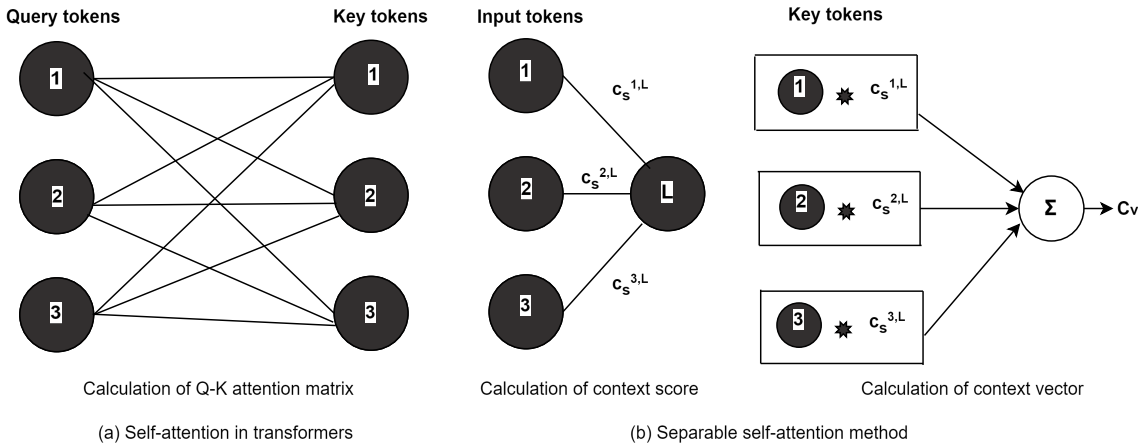


Figure 3.6: The calculation of Q-K interaction in separable self-attention and self-attention. c_s and c_v denote context score and context vector respectively.

After having this context vector, this context vector is multiplied element-wise by V followed by a linear layer to result in the final output. This procedure and final output are shown in Fig 3.7 and Eq. 3.13.

$$\hat{x} = \mathbf{V} * \left(\sum \mathbf{K} * \text{Softmax}(q) \right) \tag{3.13}$$

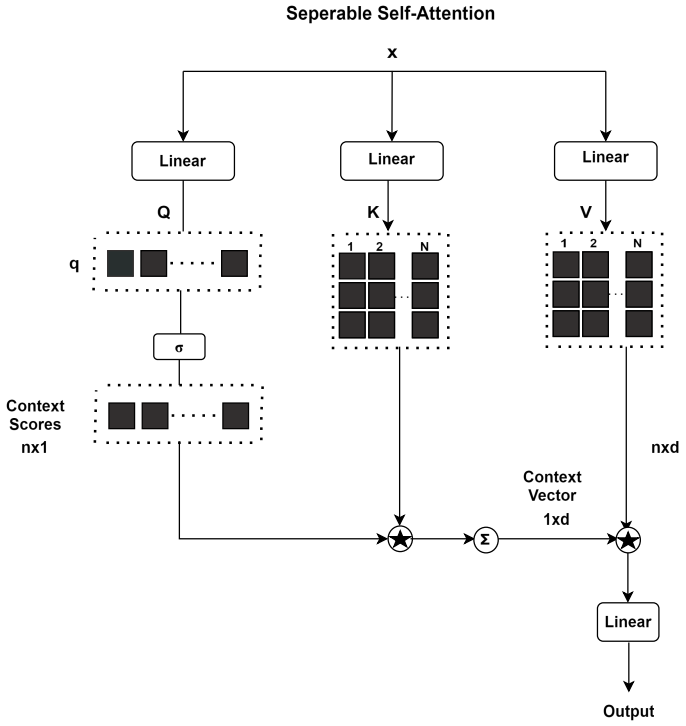


Figure 3.7: The architecture of separable self-attention

3.3.4 Efficient Additive Attention

To alleviate the quadratic complexity in self-attention, Shaker et al. propose an attention scheme having linear complexity and lacking K-V interaction. Shaker et al. demonstrate that replacing the Q-K interaction with a linear layer can be utilized for learning the relationship between the tokens [58]. In this attention framework, $Q \in \mathbb{R}^{n \times d}$ and $K \in \mathbb{R}^{n \times d}$ are obtained from the input using two matrices $W_Q \in \mathbb{R}^{d \times d}$ and $W_K \in \mathbb{R}^{d \times d}$ where n and d denote the token length and channel dimension of the input. After computing Q and K, Q is multiplied by $w_a \in \mathbb{R}^d$ to learn the attention weights of Q. This results in the global attention query vector $a \in \mathbb{R}^n$ which is calculated in Eq. 3.14. After calculating the global attention query vector, the global query vector $q \in \mathbb{R}^d$ is computed as shown in Eq. 3.15.

$$\alpha = \frac{Q \cdot w_a}{\sqrt{d}} \tag{3.14}$$

$$q = \sum_{i=1}^{n} \alpha_i \cdot Q_i \tag{3.15}$$

After calculating the global query vector, this vector is multiplied element-wise with

K to form the global context $\in \mathbb{R}^{n \times d}$. This matrix not only captures the information from every token but also learns the correlation in the input sequence [58]. After obtaining this matrix, a linear transform is applied to the global context matrix to learn the hidden representations of the tokens instead of using the V matrix. After applying this linear transform, the Q matrix is normalized along the channel dimension considering the L2 norm and added to the output of the linear transform. Moreover, this attention model has linear complexity with the token length. The scheme of this attention model is demonstrated in Fig. 3.8.

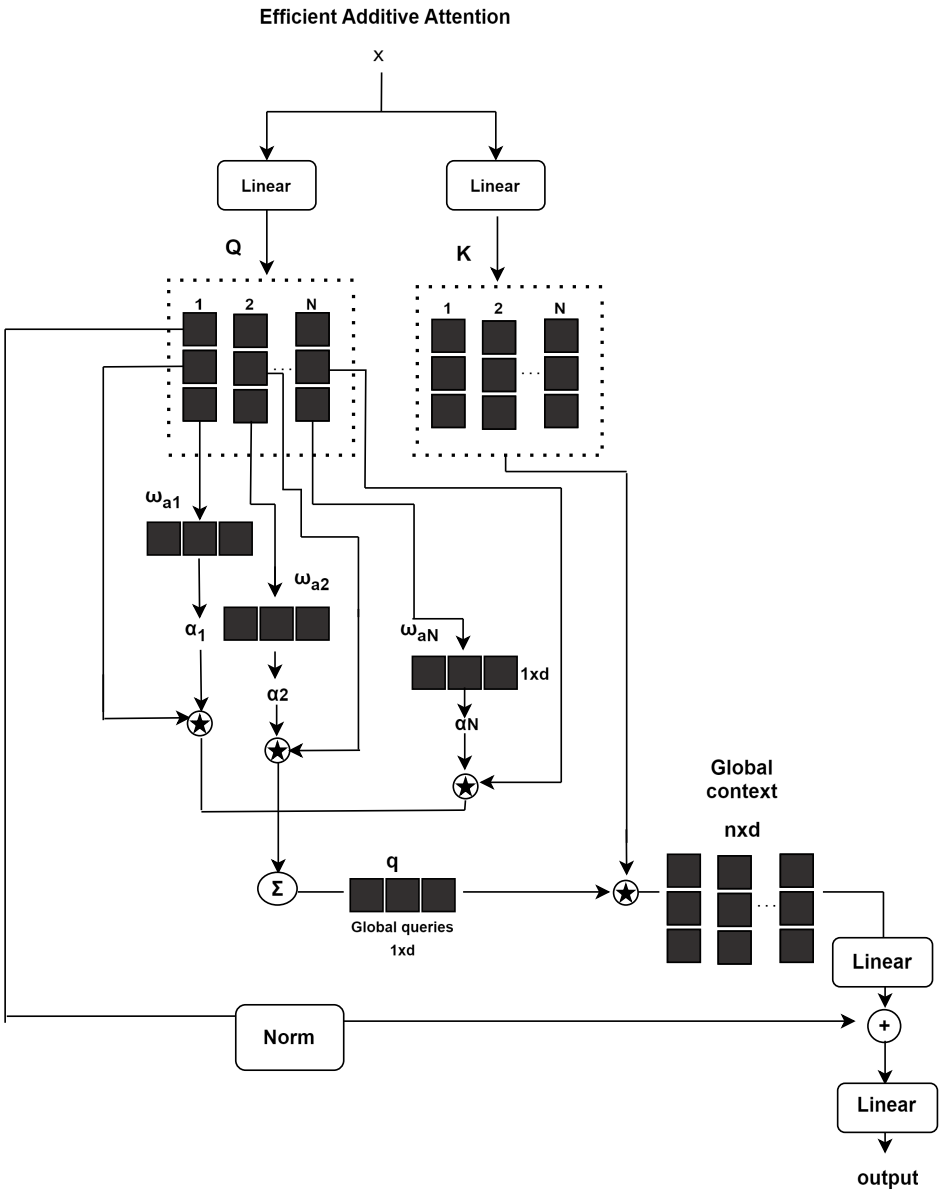


Figure 3.8: The architecture of efficient additive self-attention

3.4 MLP Mixer

Tolstikhin et al. propose an alternative, architecture for the vision transformers which does not use convolutions or self-attention modules. Instead of using these modules, Tolskin et al. use MLPs to create the interaction between the tokens and channels. These MLPs are applied across the spatial and feature dimensions. This MLP-based architecture is called the mixer layer which is shown in Fig. 3.9.

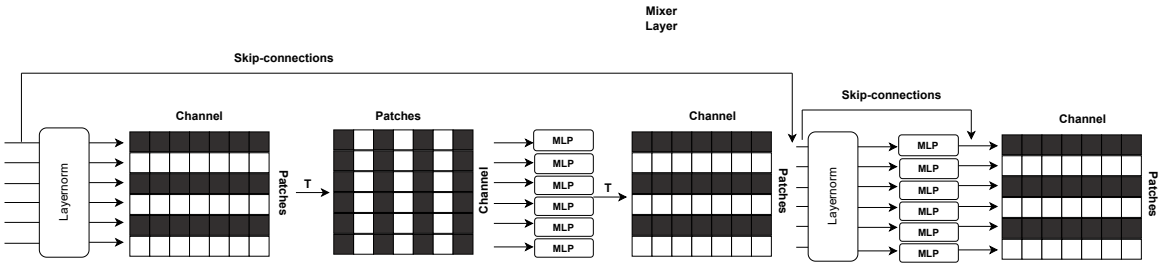


Figure 3.9: The architecture of MLP mixer layer

As shown in Fig. 3.9, the input to the mixer layer is composed of image patches and channels. Before applying this mixer layer, the input with $H \times W$ resolution is passed through a 2D convolutional layer to generate the image patches having $P \times P$ resolution. After splitting the image into patches, the total number of these patches is $S = HW/P^2$ and these patches are fed to the mixer layer. The image patches $X \in \mathbb{R}^{S \times C}$ are passed through normalization and then to the first MLP layer as demonstrated in Eq. 3.16. Before applying this MLP layer, the transpose of the image patches is taken. This MLP layer, known as token mixing MLP, establishes the interaction between the tokens. After applying this token-mixing MLP on the image patches, the second MLP layer, known as channel mixing MLP, is applied to this output to establish the channel interaction, illustrated in Eq. 3.17. Similar to token-mixing MLP, the output of the token-mixing MLP is transposed before passing through the channel-mixing MLP.

$$U_{*,i} = X_{*,i} + W_2\sigma\left(W_1\text{LayerNorm}(X_{*,i})\right) \quad \text{for } i = 1, \ldots, C \tag{3.16}$$

$$Y_{j,*} = U_{j,*} + W_4\sigma\left(W_3\text{LayerNorm}(U_{j,*})\right) \quad \text{for } j = 1, \ldots, S \tag{3.17}$$

As shown in Eq. 3.16 and 3.17, C, S and σ denote the number of channels, the number of patches, and the GELU layer respectively whereas X, U and Y correspond to the input, the output of the token-mixing MLP and the output of the channel-

mixing MLP. The W terms denote the weights used in these MLP mixers. The layer norm is a normalization utilizing the mean and variance. The output of the layer norm is shown in Eq. 3.18, 3.19 and 3.20 when a vector x with length N is fed to the layer norm.

$$\text{mean}(x) = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{3.18}$$

$$\text{var}(x) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \text{mean}(x))^2 \tag{3.19}$$

$$\text{LayerNorm}(x) = \frac{1}{\sqrt{\text{var}(x)}} \cdot (x - \text{mean}(x)) \tag{3.20}$$

3.4.1 Spatial-Shift MLP Mixer

Spatial-Shift MLP Mixer is an MLP-mixer-based architecture using MLP layers instead of self-attention and convolution layers. For the communication between the channels, this architecture employs a channel-mixing layer. Different from the token-mixing MLP layer in MLP-Mixer, this architecture uses spatial-shift operations to achieve communication between the patches. This spatial-shift operation groups channels into multiple groups and then applies different shifts to them, as illustrated in Fig 3.10.

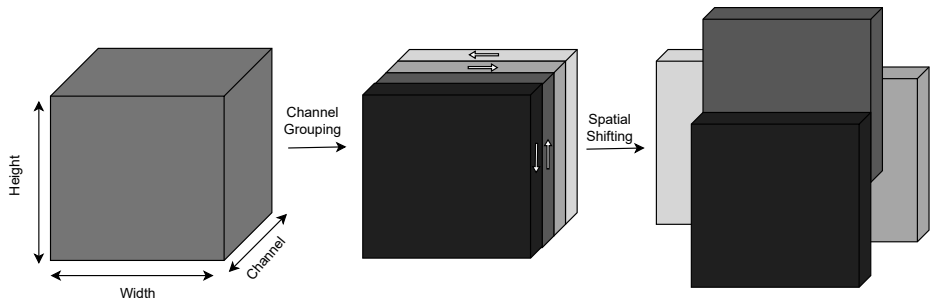


Figure 3.10: Spatial shift operation

Similar to the architecture of the MLP mixer, before applying these spatial shifts and channel-mixing MLP layers, the $W \times H$ input is split into $w \times h$ non-overlapping patches $P = \{P_i\}_{i=1}^{\text{wh}}$ where $P_i \in \mathbb{R}^{p \times p}$, $w = W/p$ and $h = H/p$ [61]. After obtaining these patches from the input, each patch is fed to a patch-wise fully connected layer following a layer norm, illustrated in Eq. 3.21.

$$\mathbf{e}_i = \text{LN}(\mathbf{W}_0\mathbf{p}_i + \mathbf{b_0}) \tag{3.21}$$

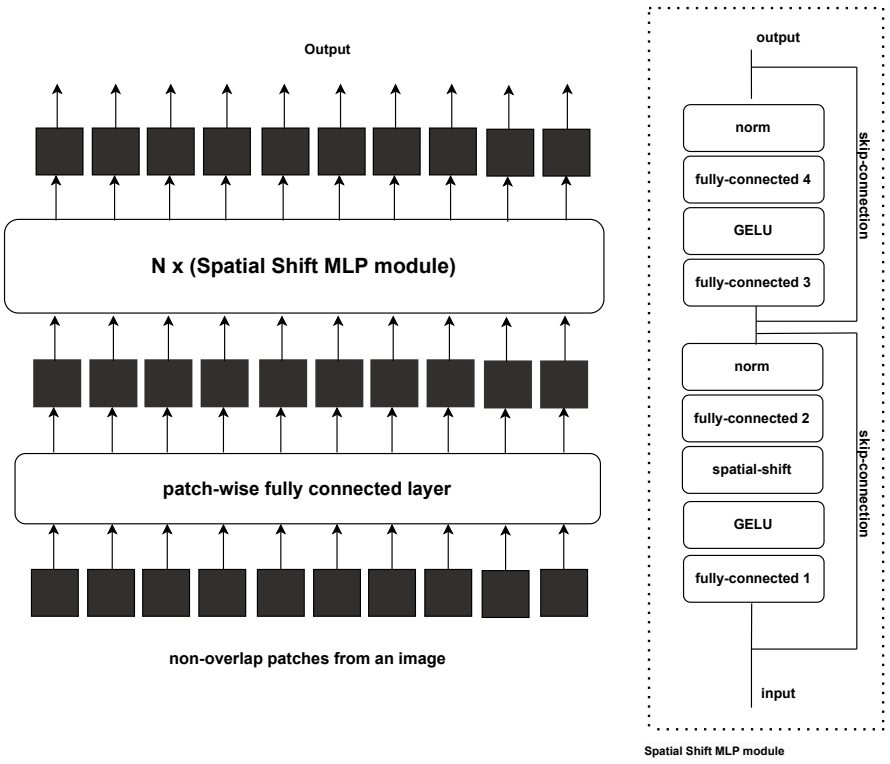


Figure 3.11: The architecture of spatial shift MLP mixer

After this patch-wise fully connected layer, the output is fed to the spatial shift MLP module composed of four fully connected layers, two layer-normalization layers, two GELU layers, two skip connections, and a spatial shift operation as demonstrated in Fig. 3.11. This spatial shift operation first groups the channels and applies different shifts on each channel group $\mathcal{T}_i$ as illustrated in Eq. 3.22. Using this shifting scheme absorbs the visual content from its adjoining patches instead of using token-mixing MLP [61].

$$\begin{aligned}
\boldsymbol{\mathcal{T}}1[1:w,:,:] &\leftarrow \boldsymbol{\mathcal{T}}1[0:w-1,:,:],\\
\boldsymbol{\mathcal{T}}2[0:w-1,:,:] &\leftarrow \boldsymbol{\mathcal{T}}2[1:w,:,:],\\
\boldsymbol{\mathcal{T}}3[:,1:h,:] &\leftarrow \boldsymbol{\mathcal{T}}3[:,0:h-1,:],\\
\boldsymbol{\mathcal{T}}4[:,0:h-1,:] &\leftarrow \boldsymbol{\mathcal{T}}4[:,1:h,:].
\end{aligned} \tag{3.22}$$

3.4.2 Phase-Aware Vision MLP

MLP architectures are composed of two main MLP layers which are channel and token MLP. The channel MLP establishes the interaction between the channels, whereas the

token MLP creates the relation between the tokens. The weights used in the token MLP are constant over channels. This results in an obstacle related to creating channel-based token interaction. In attention-based architectures, the inner product creates the channel-based token interaction. Tang et al. propose a modified MLP architecture in which the channel-based token interaction is integrated. This architecture adopts the wave structure in token modeling. The amplitude encodes the content of each token, whereas the phase encodes the relation between the tokens and fixed weights in tokens. This Wave MLP is composed of fully connected layers and Phase-Aware Token Mixing (PATM) used for establishing the channel-based token mixing, as demonstrated in Fig. 3.12.

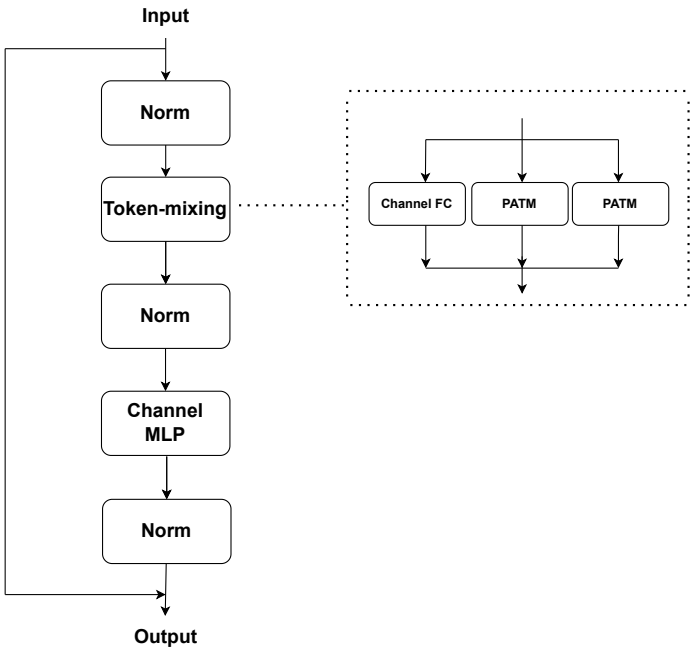


Figure 3.12: The scheme of MLP Wave

The input to the PATM is modeled as a wave having the amplitude and phase term, as illustrated in Eq.3.23. To model the wave-based MLP, fully connected layers are used to estimate the amplitude and the phase of the output of the PATM as shown in Eq. 3.24.

$$\tilde{\mathbf{z}}_j = |\mathbf{z}_j| \odot e^{i\boldsymbol{\theta}_j} = |\mathbf{z}_j| \odot \cos(\boldsymbol{\theta}_j) + i|\mathbf{z}_j| \odot \sin(\boldsymbol{\theta}_j), \quad j = 1, 2, \ldots, n. \tag{3.23}$$

$$\begin{aligned} \omega_j =& \sum_k W^t_{jk}\mathbf{z}_k \odot \cos\theta_k + W^i_{jk}\mathbf{z}_k \odot \sin\theta_k, \\ \theta_j =& \Theta(x_j, W_\theta), \\ & j = 1, 2, \ldots, n. \end{aligned} \tag{3.24}$$

As shown in Eq. 3.24, the output of the PATM is represented as ω_j characterized by the real and imaginary components. W^i_{jk} and W^t_{jk} are the weights for the magnitudes of the tokens. W^t_{jk} corresponds to the weights of the real part of the tokens, whereas W^i_{jk} denotes the weights of the imaginary part of the tokens. This complex sum is implemented by concatenating the real and imaginary parts and storing them in a matrix. The weighted phase contributions, θ_j, are also modeled as a fully connected layer as depicted in Eq. 3.24. The input tokens are fed to this fully connected layer and the fully connected layer outputs the phase contribution by weighting the inputs with W_θ. After this computation, the sinusoidal terms are formed considering Euler's law shown in Eq. 3.23. After forming the complex and real parts, the real and complex parts are weighted by the amplitude weights and input token, as demonstrated in Eq. 3.24. The token-mixing module consists of the channel-based fully connected and PATM layers. To preserve the original features, the original features are also added to the final output of the wave MLP.

3.5 Reprojection Principle

In the self-supervised monocular depth estimation, the pose estimation and depth estimation are utilized to reconstruct the original image, and the disparity between the reconstructed image and the original image determines the training loss. The depth network inputs a single image I_t, and it outputs a pixel-wise depth map $\hat{D}$. In the pose estimation, the pose network not only takes the input image to the depth network but also inputs the adjacent images (I_{t+1} and I_{t-1}) to the input image with respect to time. This pose network outputs the relative camera positions which are $\hat{T}_{t\to t-1}$ and $\hat{T}_{t\to t+1}$ [47]. Zhang et al. adopt the encoder-decoder architecture for the pose estimation. In their pose estimation framework, they use PoseNet as the encoder, whereas they use a pose decoder with four convolutional layers to estimate the corresponding 6-Degrees of Freedom (DoF) relative pose between adjacent images[14]. The camera intrinsic matrix K is a 3 × 3 matrix having focal length, aspect ratio, sensor skew, and the center of projection x and y as the parameters [46], and this matrix has been already provided in the dataset.

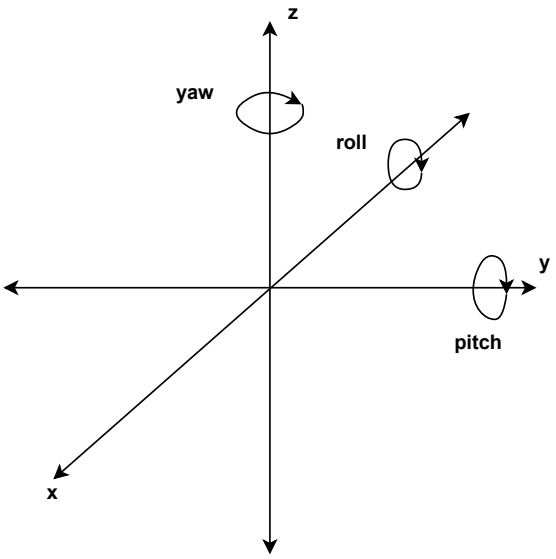


Figure 3.13: The visualization of 6 DoF

After the estimation of the 6-DoF relative camera poses and the pixel-wise depth maps, the adjacent frame I_{t+1} is reconstructed using the input frame I_t, pixel-wise depth map $\hat{D}$, camera intrinsic matrix K and $\hat{T}_{t\to t+1}$ as demonstrated in Eq. 3.25 [47]. Similar to Eq.3.25, the equation for the reconstruction of the adjacent frame I_{t+1} is formed by replacing the term $t+1$ with $t-1$. This reconstruction of the adjacent frame can be defined as the projection of the pixels on the input frame I_t to the adjacent frame I_{t+1}. Moreover, this projection considers a continuous pixel space in the I_{t+1} frame, and the projected pixels in this frame can be between the pixel coordinates as illustrated in Fig. 3.14. Bi-linear interpolation is used to distribute these projected pixel values among adjacent pixels in the I_{t+1} frame, as demonstrated in Fig. 3.14. The effect of this pixel on the closest adjacent pixel is higher than the effects on the other adjacent pixels, as shown in Eq. 3.26. The term where w_{ij} is linearly proportional to the spatial proximity between p_{t+1} and p_{t+1}^{ij} ,

$$p_{\mathrm{t}+1} = K \cdot \hat{T}_{t\to t+1} \cdot \hat{D}_t(p_t) \cdot K^{-1} \cdot p_t \tag{3.25}$$

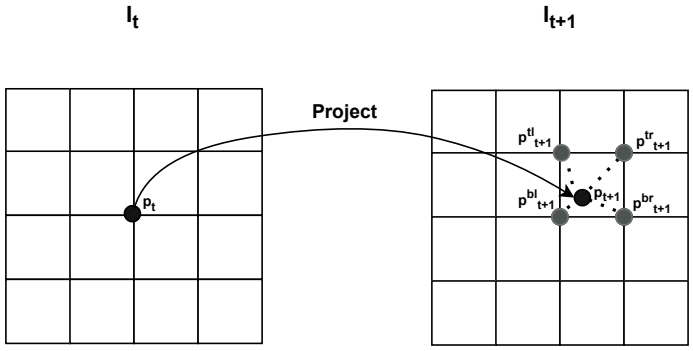


Figure 3.14: The bi-linear interpolation of the projected pixel in I_{t+1}

$$I_{t+1}(p_{t+1}) = \sum_{i\in\{t,b\},j\in\{l,r\}} \omega^{ij} \cdot I_{t+1}(p^{ij}_{t+1}) \quad \text{where} \quad \sum_{ij} \omega^{ij} = 1 \tag{3.26}$$

To establish the disparity between the reconstructed, the photometric reprojection is utilized. This loss consists of L1 and Structural Similarity Index Measure (SSIM), as illustrated in Eq. 3.27. The terms $\hat{I}_{\tilde{t}}$ and $I_{\tilde{t}}$ denote the reconstructed adjacent frame and original adjacent frame respectively. SSIM is a metric to compare the disparity between the two images considering the luminance, contrast and structure features of the images [64]. This metric uses the mean and standard deviations of the images, as illustrated in Eq. 3.28, 3.29,3.30, and 3.31.

$$L_p(\hat{I}_{\tilde{t}}, I_{\tilde{t}}) = \alpha \cdot \frac{\left(1 - \text{SSIM}(\hat{I}_{\tilde{t}}, I_{\tilde{t}})\right)}{2} + (1-\alpha)\|\hat{I}_{\tilde{t}} - I_{\tilde{t}}\| \tag{3.27}$$

$$\text{SSIM}(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{3.28}$$

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{3.29}$$

$$\sigma_x = \left(\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)^2\right)^{\frac{1}{2}} \tag{3.30}$$

$$\sigma_{xy} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y) \tag{3.31}$$

As shown in Eq. 3.28, the C terms are the constants. The term α scales the contributions of SSIM and L1 loss to the total loss, and α is set to 0.85. To alleviate

the effect of the occluded objects in the I_t frame, the adjacent frame more resembling the I_t frame is selected for the loss calculation as shown in Eq. 3.32. To remove the moving pixels and alleviate the impacts of the occlusions, a binary map is applied, which considers the disparity between frames I_t and I_{t+1}, as well as the disparity between the reconstructed and original frame, illustrated in Eq. 3.33. After applying this binary map, edge-aware smoothness loss is added to the loss term to smooth the inverse depth maps, as shown in Eq. 3.35. Moreover, this smoothing term uses the mean-normalized inverse depth maps labeled as $d_t^* = d_t/\hat{d}_t$ in the smoothing process. After this smoothing, the loss in different scales is utilized to construct the final loss term, demonstrated in Eq. 3.36. This final loss term is used in the training to tune the monocular depth encoder and decoder networks.

$$L_{\tilde{p}}(\hat{I}_{\tilde{t}}, I_{\tilde{t}}) = \min_{I_{\tilde{t}} \in (t-1,t+1)} (L_p(\hat{I}_{\tilde{t}}, I_{\tilde{t}})) \tag{3.32}$$

$$\mu = \min_{I_{\tilde{t}} \in (t-1,t+1)} (L_p(I_{\tilde{t}}, I_t) > \min_{I_{\tilde{t}} \in (t-1,t+1)} (L_p(\hat{I}_{\tilde{t}}, I_{\tilde{t}})) \tag{3.33}$$

$$L_r(\hat{I}_t, I_t) = \mu \cdot L_{\tilde{p}}(\hat{I}_{\tilde{t}}, I_{\tilde{t}}) \tag{3.34}$$

$$L_{\text{smooth}} = |\partial_x d_t^*| \, e^{-|\partial_x I_t|} + |\partial_y d_t^*| \, e^{-|\partial_y I_t|} \tag{3.35}$$

$$L = \frac{1}{3} \sum_{s \in (1,2,3)} \left(L_r + \lambda L_{\text{smooth}} \right), \quad \lambda = 10^{-3} \tag{3.36}$$

Chapter 4

Methodology

In this part of the thesis, the baseline architecture of the monocular depth estimation is presented alongside the modified up-sampling blocks and different attention-based models integrated into the baseline. In the early part of this chapter, a brief overview of the monocular depth estimation baseline is presented. After this overview, the Guided Up-sampling Block, an image-based up-sampling scheme, is introduced and how this up-sampling block is integrated into the baseline is explained in detail. After introducing this up-sampling scheme, the principle of pixel-shuffling and how it can be integrated into the decoder to alleviate the computation and memory burden are discussed in this chapter. After this discussion, the structure of the lightweight channel attention is presented alongside how this channel attention is incorporated into the up-sampling blocks in the baseline decoder. After this, various MLP-based architectures are replaced with the LGFI module in the baseline model, and how these are integrated into the baseline model is explained in detail.

4.1 The Baseline Architecture of Monocular Depth Estimation

In this thesis, Zhang et al.'s monocular depth estimation model is chosen as the baseline model. This baseline model is composed of an encoder and decoder, as illustrated in Fig. 4.1. The depth encoder mainly consists of the CDC and LGFI modules, whereas the depth decoder has upsampling blocks composed of 2-D convolutional blocks and bi-linear interpolation [14]. The CDC block is composed of dilated convolutional layers, whereas the LGFI module consists of cross-covariance attention. The frameworks of these modules are demonstrated in Fig. 4.3. The baseline depth model outputs 3 depth maps with varying resolutions. These depth maps are utilized in the reconstruction of the RGB images considering the reprojection principle, and this baseline uses a pose estimation network to aid the training of the depth network. In the training of this baseline model, the self-supervised scheme for the monocular depth estimation is followed.

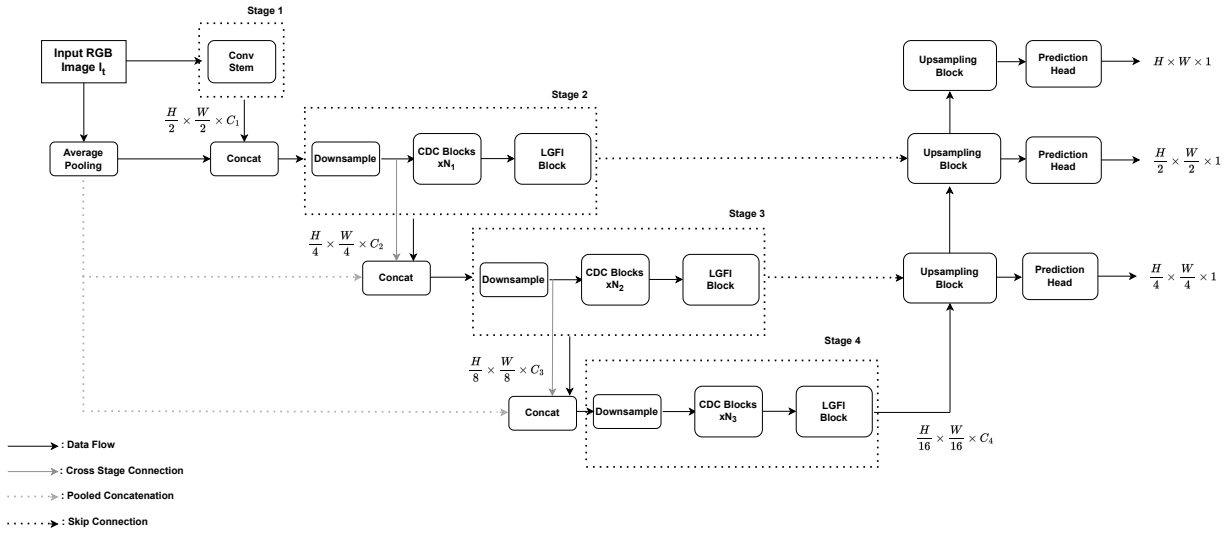


Figure 4.1: The architecture of baseline depth estimation model

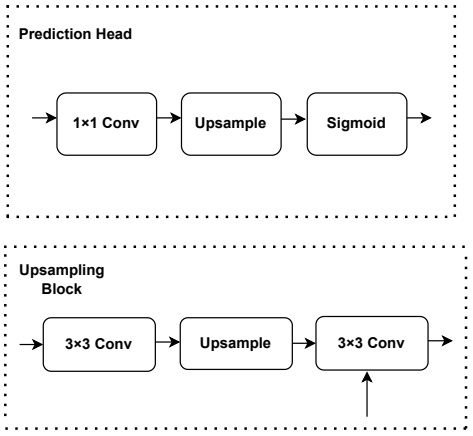


Figure 4.2: The architecture of upsampling block and prediction head in the baseline depth estimation model

As shown in Fig. 4.1, the input RGB image I_t with $H \times W$ resolution is given as the input to the depth estimation network. The input image is downsampled using pooling and these downsampled images are passed to the CDC blocks followed by the LGFI blocks. To establish the interaction between the features with different resolutions, concatenation operations are used, as seen in Fig. 4.1. After extracting the feature with the help of CDC and LGFI layers, the features are passed to the 2-d convolutional-based up-sampling block, as illustrated in Fig. 4.2. After this upsampling block, the prediction head shown in Fig. 4.2 is employed to predict the final depth maps with different resolutions. This baseline monocular depth estimation uses Zhang et al's Lite-Mono model parameters [14], as demonstrated in Table 4.1.

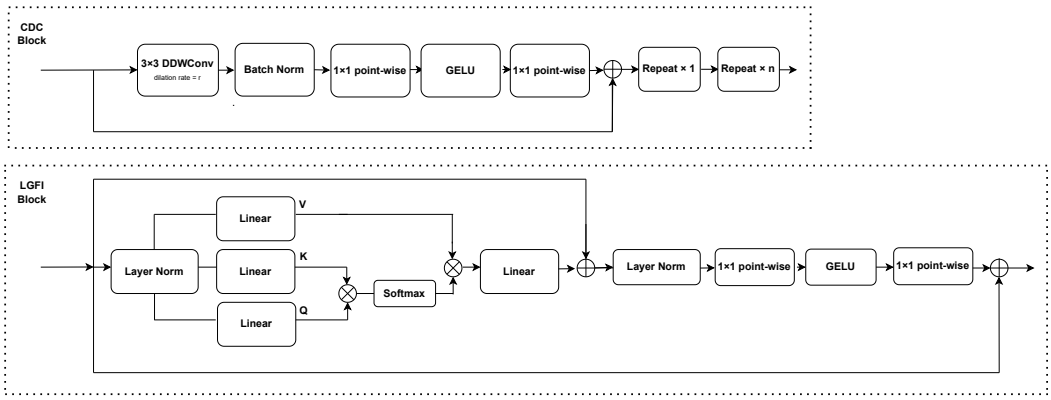


Figure 4.3: The architecture of CDC and LGFI in the baseline depth estimation model

Output Size	Layers	Lite-Mono
640×192	Input	
320×96	Conv Stem	$3 \times 3, 48$, stride $= 2$
		$[3 \times 3, 48] \times 2$
160×48	Downsampling	$3 \times 3, 48$, stride $= 2$
Stage 1	CDC blocks	$[3 \times 3, 48] \times 3$
	LGFI block	dilation $= 1, 2, 3$
80×24	Downsampling	$3 \times 3, 80$, stride $= 2$
Stage 2	CDC blocks	$[3 \times 3, 80] \times 3$
	LGFI block	dilation $= 1, 2, 3$
40×12	Downsampling	$3 \times 3, 128$, stride $= 2$
Stage 3	CDC blocks	$[3 \times 3, 128] \times 9$
	LGFI block	dilation $= [1, 2, 3] \times 2, 2, 4, 6$
#Total Params. (M)		3.1

Table 4.1: The parameters of the Lite-Mono framework and its total number of parameters. In $[3 \times 3, C] \times N$, N and C denote the number of repetitions of the dilated convolutional block and the channel dimension respectively.

4.1.1 Dataset

In this thesis, the KITTI [65] dataset is used to train the monocular depth models. This dataset is composed of 61 stereo road scenes collected by multiple sensors, which are camera, 3-D LiDAR and GPU/IMU. While training the models, a partition of the dataset called Eigen-split is used in the training and evaluation of the models instead of using the complete dataset [66]. The numbers of monocular triplets used in training, evaluation, and testing are 39,180, 4,424, and 697, respectively. The camera intrinsic matrix K is formed by averaging the focal lengths of the images across the dataset, and this K matrix is considered a constant, which is used in the reprojection principle. Apart from this dataset, the Make3D dataset [67], composed of 134 test images of

outdoor scenes, is utilized in the performance evaluation of the depth models. Apart from these datasets, the ImageNet 2012 [68] dataset is used in the depth encoder's pre-training to enhance the depth models' performance. This dataset is an object detection dataset with 1000 classes. To evaluate the performance of this pre-training, Top-1 and Top-5 accuracies are used. Top-1 accuracy denotes the single-best predictions, whereas Top-5 accuracy considers the top-5 predictions.

4.2 The Integration of GUB into the Baseline

Belagiannis et al. propose an image-based upsampling block in the decoder, which uses the downsampled images as guides for the upsampling [18]. They use a pre-trained DDRNet-23-slim network, and their proposed method is illustrated in Fig. 4.4.

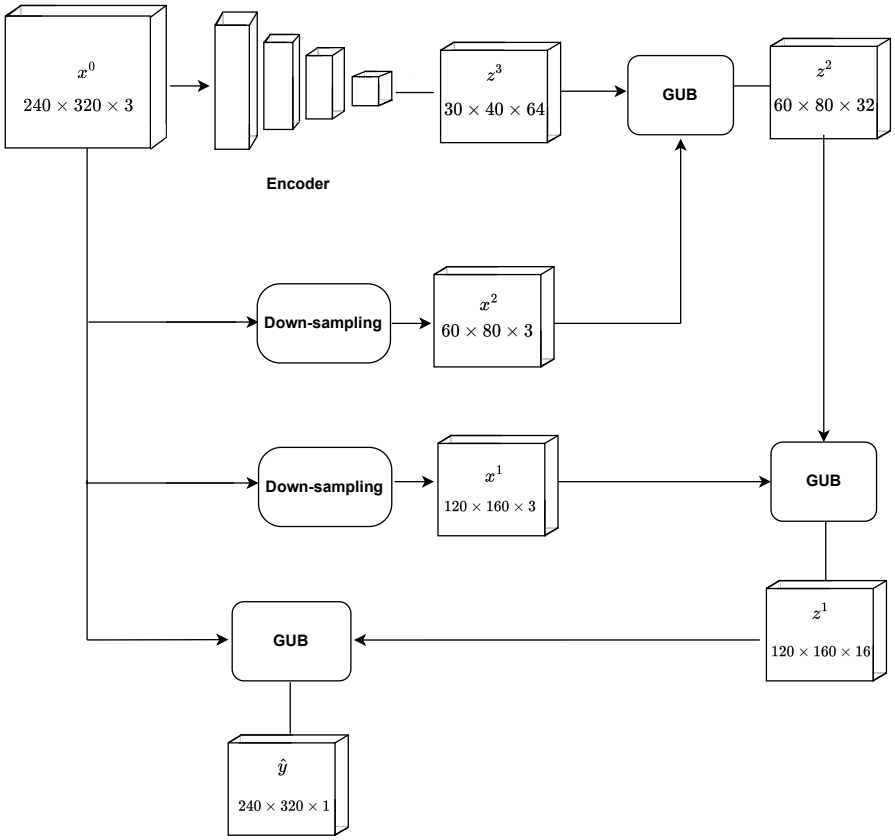


Figure 4.4: The proposed architecture of Belagiannis et al.'s guided upsampling-based depth model

The GUB's operation is demonstrated in Fig. 4.5. In this scheme, the feature is the output of the depth encoder, while the guide is the downsampled input RGB image. The features and outputs are passed through convolutional layers and then

concatenated, illustrated in Fig. 4.5. After these layers, the concatenated output is passed through a Squeeze and Excitation (SE) layer. This layer weights the channel dimension by averaging the concatenated spatial output dimensions followed by the fully connected layers. After this SE layer, the weighted output is passed to the convolutional layers followed by a skip connection of the upsampled feature. While integrating GUBs into the baseline, the GUBs are used for the upsampling of the features in the decoder. Instead of solely concatenating the features with different resolutions, feature-based upsampling schemes are adopted. In the first feature-based upsampling scheme, the higher and lower-resolution features are treated as the features and guides of the GUB respectively, whereas, in the second feature-based upsampling scheme, the higher and lower-resolution features are treated as the guides and features of the GUB respectively. The scheme of the GUB-based monocular depth estimation is demonstrated in Fig. 4.6.

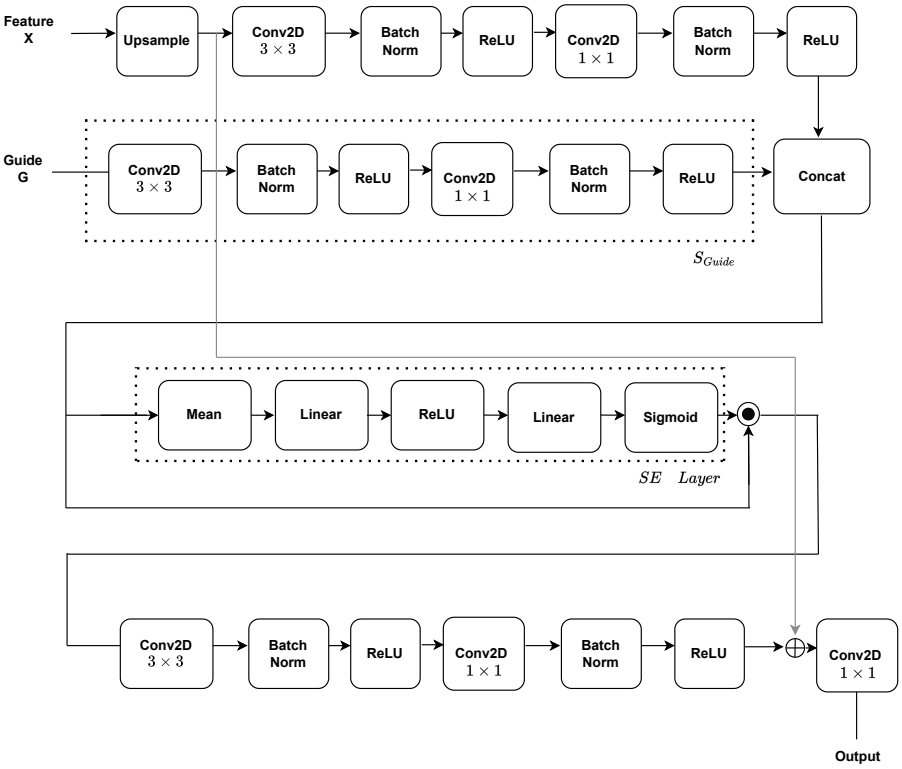


Figure 4.5: The architecture of GUB

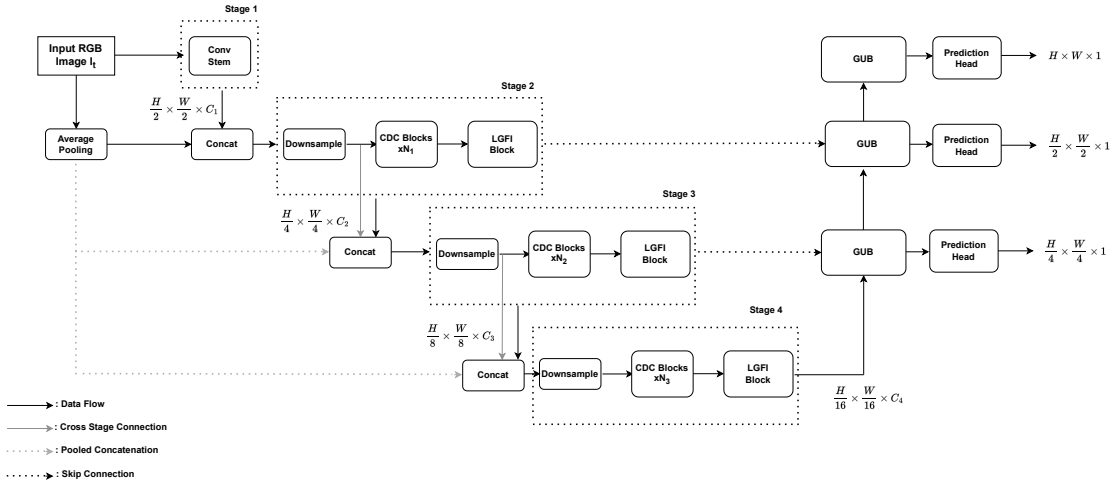


Figure 4.6: The architecture of GUB-based monocular depth estimation model

4.3 Pixel Shuffle

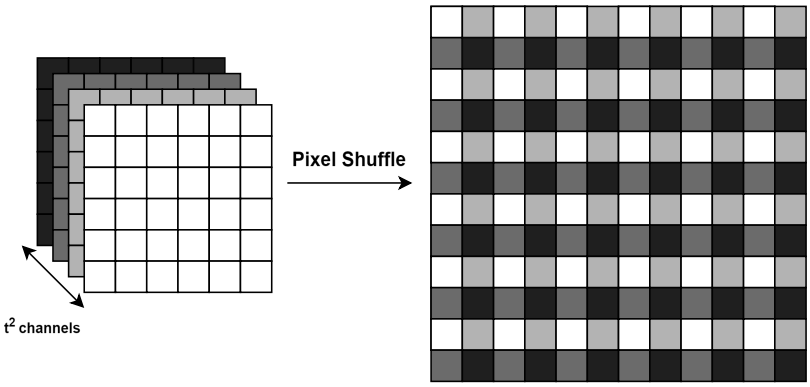


Figure 4.7: Pixel-shuffle operation

In the decoder of the monocular depth estimation baseline, bi-linear up-sampling-based 2-D convolutional layers are adopted, illustrated in Fig, 4.1 and 4.2. This bi-linear up-sampling is an up-sampling scheme using the linear interpolation in its width and height dimensions. This bi-linear up-sampling does not provide any additional information about the features extracted from the encoder and increases the height and width of the features. Shi et al. introduce a scheme for up-sampling the images and videos by rearranging the features in real-time [35]. Their proposed up-sampling scheme is called pixel-shuffle, and this shuffling procedure is depicted in Fig. 4.7. This shuffling operation re-arranges the features resulting in a reduction of the channel dimension by the square root. This rearrangement of the features increases the height and width of the features, shown in Fig. 4.7. In this thesis, the bilinear up-sampling is replaced

with this operation to reduce the complexity and memory of the depth model.

4.4 The Integration of Channel Attention Block into the Baseline

The monocular depth estimation baseline's encoder is composed of LGFI and CDC. LGFI has a cross-covariance attention block followed by fully connected layers, demonstrated in Fig 4.3. Different from the scheme illustrated in Fig. 4.3, the last layers composed of pointwise convolutional layers are removed and this is considered as the channel attention block. This channel attention block and the pixel shuffling operations are incorporated into the baseline model's decoder. These are used in the up-sampling of the features in the decoder, similar to GUB. The proposed method is up-sampling the feature space with a lower resolution with pixel shuffling. The pixel shuffling reduces the number of channels due to its re-arrangement scheme, distinguishing it from the baseline's bilinear interpolation. After this pixel shuffling-based up-sampling, the up-sampled features are concatenated with the feature space with a higher resolution. Then this concatenated output is passed through a cross-covariance attention following a 2-D convolutional layer. After applying this channel attention, the output and feature spaces with the highest resolution are treated as the feature spaces with a lower and higher resolution respectively for the next iteration, and the same procedure is repeated.

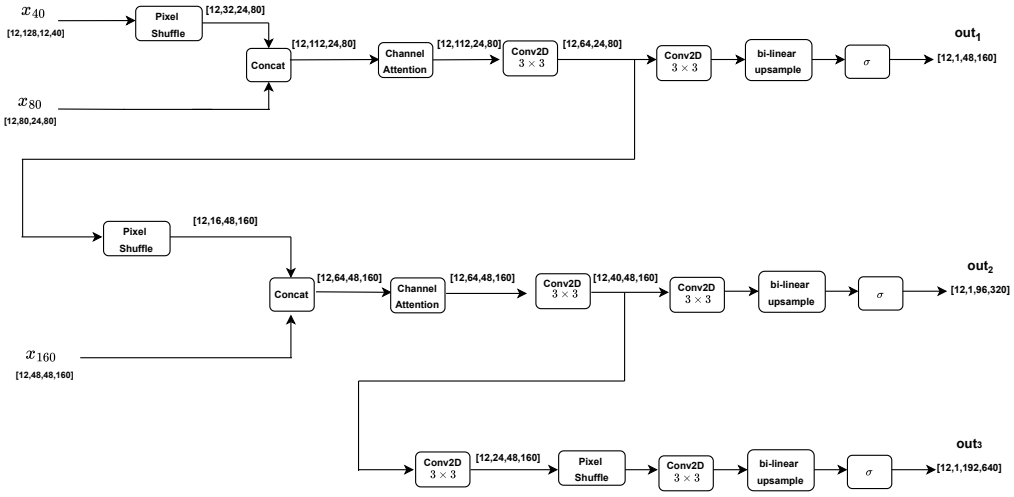


Figure 4.8: The first channel attention-based monocular depth estimation model's decoder

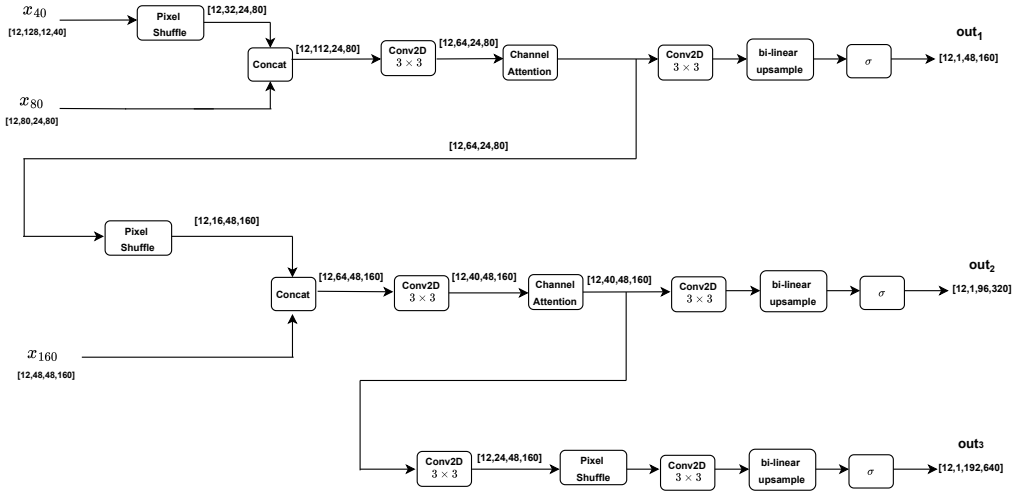


Figure 4.9: The second channel attention-based monocular depth estimation model's decoder

While implementing this channel attention-based monocular depth estimation, two different approaches of this model are considered: one involving channel attention followed by a 2-D convolutional layer, and the other featuring a 2-D convolutional layer followed by channel attention. The main difference between these approaches is the order of the operations and the impact of this order on the complexity and performance of the model is investigated. The scheme of the monocular depth estimation utilizing this channel attention-based upsampling decoder is illustrated in Fig. 4.10.

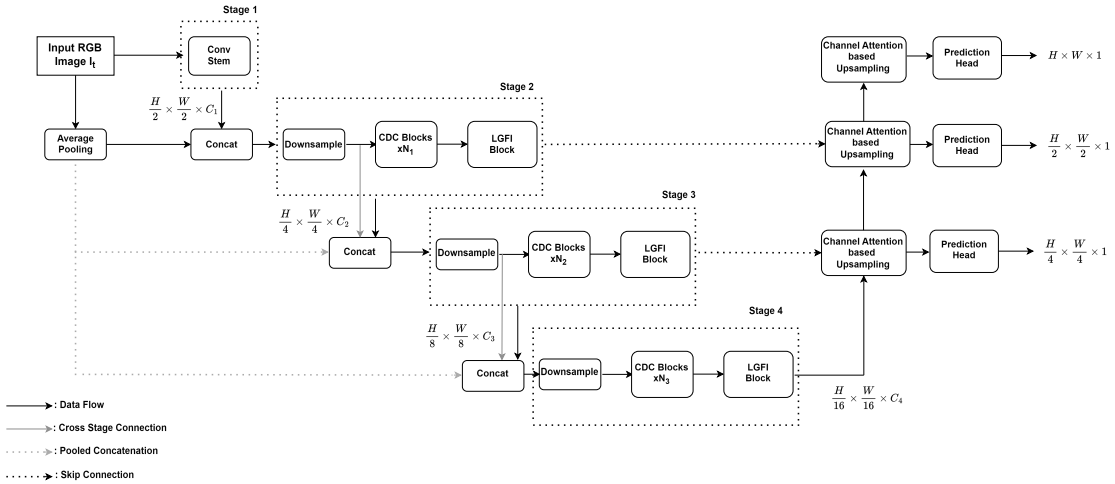


Figure 4.10: The channel attention-based monocular depth scheme

4.5 The Integration of MLP Mixer into the Baseline

In this part of the thesis, the LGFI blocks in the encoder are replaced with the MLP mixer blocks, shown in Fig. 4.12. These MLP mixer blocks are composed of fully connected layers different from the attention-based architectures as illustrated in Sec. 3.4. The MLP-Mixer architecture proposed by Tolkstikhin et al. [69] is utilized in the object detection and classification on the Image-net dataset. This architecture is modified to be compatible with the monocular depth estimation model in this thesis. Firstly, this architecture's fully connected layer and averaging layers are removed, and the dimensions of the patch-wise features extracted from MLP are adjusted. The initial dimension of the MLP mixer's output consists of the number of channels and patchwise feature blocks extracted from the MLP mixer. These patch-wise feature blocks are split into two dimensions which correspond to the height and width of the feature block, demonstrated in Fig. 4.11. After this dimension adjustment, the bi-linear upsampling layers and 2-D convolution layers are employed to upsample the output features of the MLP mixer, and after this procedure, the output of the MLP block is passed to the monocular depth estimation baseline's decoder. The decoder of this modified baseline uses the channel attention-based upsampling scheme while constructing the depth maps, which is explained in detail in Sec. 4.4.

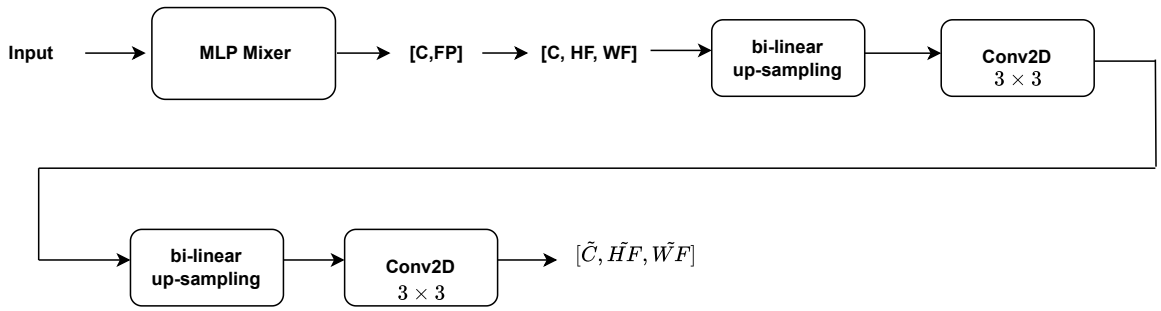


Figure 4.11: The modified MLP mixer block scheme.

As shown in Fig. 4.11, the terms C and FP correspond to the output channel and patch-wise feature dimensions respectively, whereas HF and WF denote the height and width feature dimensions respectively. The output is shown as $[\tilde{C},\tilde{WF},\tilde{HF}]$, where $\tilde{C},\tilde{HF},\tilde{WF}$ denote the final channel, height feature and width feature dimensions respectively. The MLP mixer-based monocular depth estimation framework is shown in Fig. 4.12.

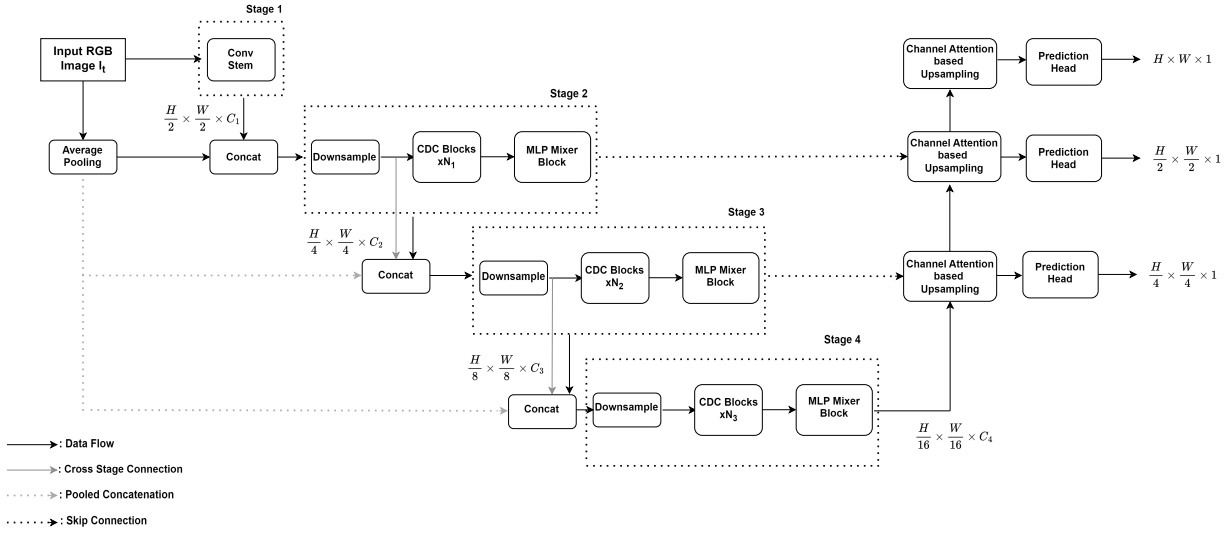


Figure 4.12: The modified MLP mixer-based monocular depth estimation framework.

While constructing MLP mixer layers in the MLP mixers, 2 fully connected layers and GELU functions are utilized. The first fully connected layer expands the dimension of the input, whereas the second fully connected layer returns the output of the first MLP layer to its original dimension, which is illustrated in Fig. 4.13.

Figure 4.13: The fully connected layer architecture in the MLP mixer.

4.6 The Integration of Spatial-Shift MLP Mixer into the Baseline

Spatial Shift MLP mixer uses only the channel mixing layer, which is different from Tolkstikhin et al.'s MLP mixer architecture. Instead of using the token mixing layer, the spatial shift operations are applied to obtain the spatial interaction between the tokens, as demonstrated in Sec. 3.4.1. Similar to Tolkstikhin et al.'s MLP mixer architecture, the Spatial Shift MLP mixer is used in object detection on the ImageNet dataset. The averaging layers and the fully connected layers, related to the classification task, are replaced with the bilinear up-sampling and 2-D convolutional layer. Before applying these layers, the patch-wise feature dimension of the output of the Spatial MLP mixer is split into height and width feature dimensions. The fully connected layers used in the spatial MLP mixer follow the scheme of the MLP mixer's

fully connected layer demonstrated in Fig. 4.13. The modified part of this encoder is illustrated in 4.14. The decoder of this architecture employs channel attention-based up-sampling, explained in Sec. 4.4. The scheme of the modified baseline considering the Spatial Shift MLP mixer is illustrated in Fig. 4.15.

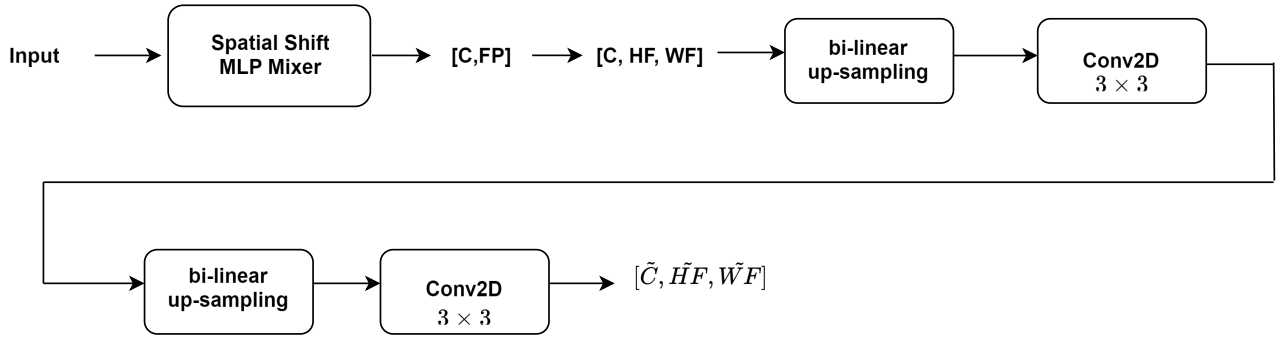


Figure 4.14: The modified scheme of the Spatial Shif MLP mixer.

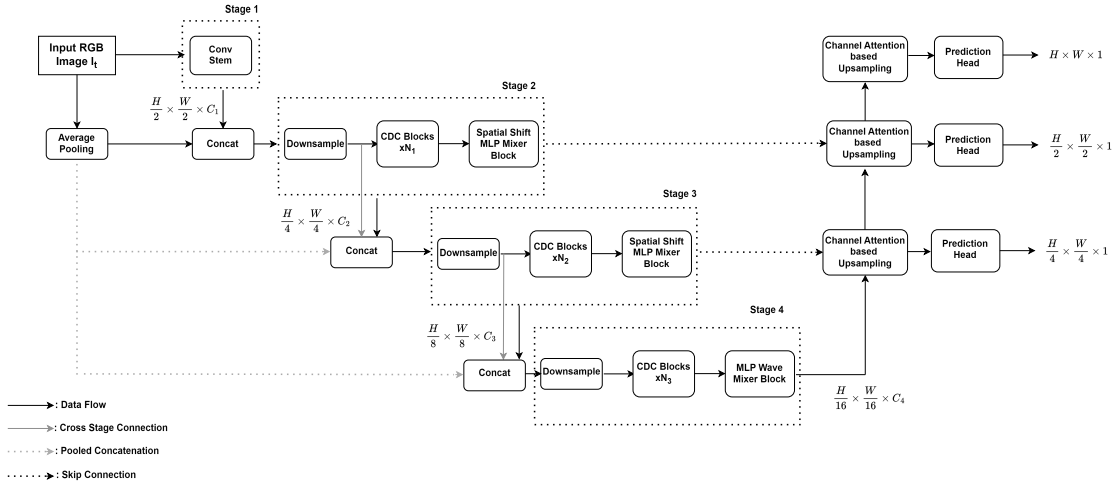


Figure 4.15: The Spatial Shift MLP Mixer-based monocular depth estimation framework.

4.7 The Integration of Wave MLP Mixer into the Baseline

Unlike the previous MLP architectures, the MLP wave architecture utilizes fully connected layers to create channel-based interaction between the tokens. While creating this interaction, each token is modeled as a wave having an amplitude and a phase. The phase of the token mainly establishes this interaction, whereas the amplitudes of each token encode the content of each token. Tang et al. use this wave MLP mixer

in object detection on the ImageNet dataset. To adopt this architecture into monocular depth estimation, the classification layers are replaced with 2D convolutional and pixel shuffling, illustrated in Fig. 4.16. The main reason behind using a pixel shuffling layer in this architecture is to reduce the impact of PATM layers on the complexity and memory of the model. Similar to previous MLP mixer-based monocular depth estimation models, the LGFI module of the baseline is replaced with this architecture whereas the decoder with channel-based attention upsampling scheme is employed to generate the depth maps, as shown in Fig. 4.17.

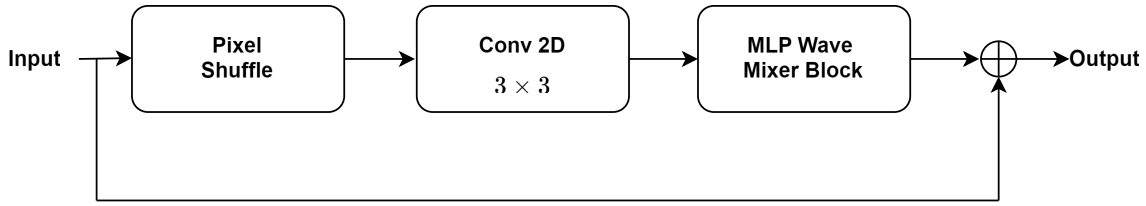


Figure 4.16: The scheme of the MLP wave block.

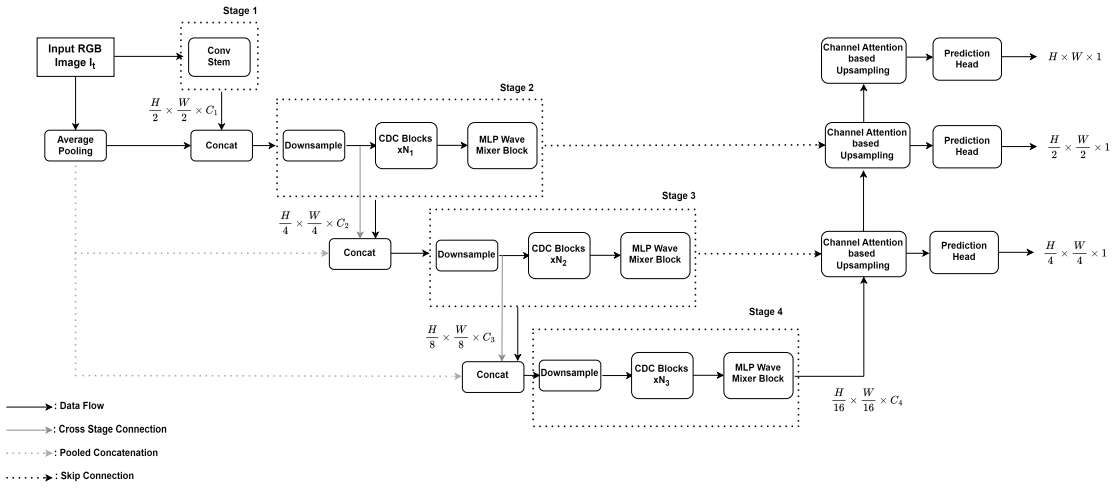


Figure 4.17: The MLP wave Mixer-based monocular depth estimation framework.

4.8 The Integration of Swift Former into the Baseline

The Swift Former architecture is composed of efficient additive attention and dilated convolution blocks, demonstrated in Fig. 4.18. Different from the self-attention models, the efficient additive attention architecture utilizes a linear layer instead of a V matrix used in the interaction between key and value. This scheme is explained in detail in Sec. 3.3.4. This additive efficient attention architecture encodes the global

context, whereas the CDC and convolutional layer are employed to extract the multi-scale features. On the decoder side, the channel attention-based upsampling scheme is employed. The scheme of this Swift Former is demonstrated in Fig. 4.18 and Eq. 4.1.-4.3.

Figure 4.18: The Swift Former architecture.

$$\hat{X}_i = \text{Conv}_1(\text{DWConvBN}(\hat{X}_i)) \tag{4.1}$$

$$\hat{X}_i = QK(\hat{X}_i) + \hat{X}_i \tag{4.2}$$

$$\hat{X}_{i+1} = \text{Conv}_1(\text{Conv}_{\text{BN,1,G}}(\hat{X}_i)) + \hat{X}_i \tag{4.3}$$

In Eq. 4.1-4.3, $conv_1$, $DWConv$ and QK denote the 2-D convolutional, 2-D dilated convolutional and additive efficient attention layers respectively, whereas BN and G correspond to the batch norm and GELU layers respectively. The decoder of this framework uses the channel attention upsampling scheme. The overall scheme of this modified framework is demonstrated in Fig. 4.19.

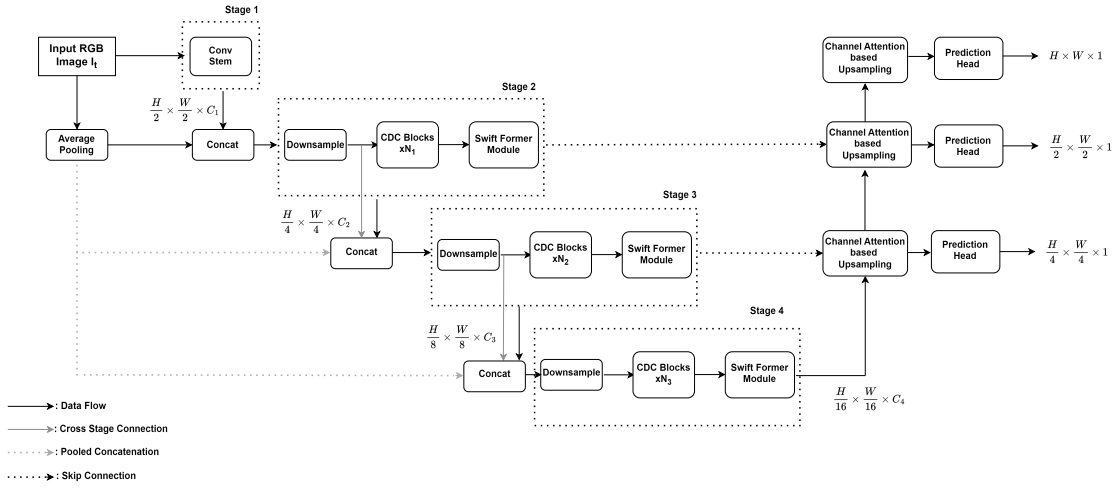


Figure 4.19: The Swift Former based monocular depth estimation framework

4.9 Training of the Depth Models

In the training of the monocular depth estimation models, the self-supervised learning technique is utilized to generate multi-scale depth maps. In this technique, the reprojection loss is used as the training loss, which is explained in Sec. 3.5. During the training process. pose estimation network is trained along with the depth model, and the reprojection loss is calculated considering the outputs of this joint network. While training the depth and pose estimation networks, the cosine annealing learning rate schedule is used to adjust the convergence speed of the model, and this schedule models the learning rate as a cosine function, as illustrated in Eq. 4.4. In Eq. 4.4, $\eta_{\min}$, $\eta_{\max}$ and T denote the minimum learning rate, maximum learning rate and the maximum number of iterations respectively.

$$\eta_t = \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min}) \left(1 + \cos\left(\frac{t}{T}\pi\right)\right) \tag{4.4}$$

To get ahead of the possible overfitting of the models, the drop path is utilized in CDC, LGFI and modified LGFI blocks. The drop path resembles drop-out. The main difference between them is that the drop-out prevents co-adaptation of activations, whereas the drop-path prevents co-adaptation of parallel paths by randomly dropping operands of the join layers [70]. AdamW, a modified Adam optimizer utilizing weight decay, is used as an optimizer for the pose and depth estimation networks [71], and the weight decay is set to 0.01. The algorithm of AdamW is demonstrated in Algorithm 1. Before the training on the KITTI depth dataset, the encoder of the Swift-Former and baseline depth estimation models are trained on the ImageNet dataset [68], and this pre-training makes the convergence of the depth networks faster [14]. To change the architecture of the depth estimation encoder to an object detection classifier, fully connected and averaging layers are appended to the end of the encoder architectures, as illustrated in Fig. 4.20 and 4.21. While pre-training the encoder, the Cross-entropy loss, demonstrated in Eq. 4.5, is used as the criterion, whereas AdamW is used as the optimizer for the training on the ImageNet dataset.

$$\text{Cross-Entropy Loss} = -\sum_{i}^{N} y_i \log(p_i) \tag{4.5}$$

where:

N is the number of classes,

y_i is the true probability distribution (one-hot encoded vector for class i),

p_i is the predicted probability distribution for class i.

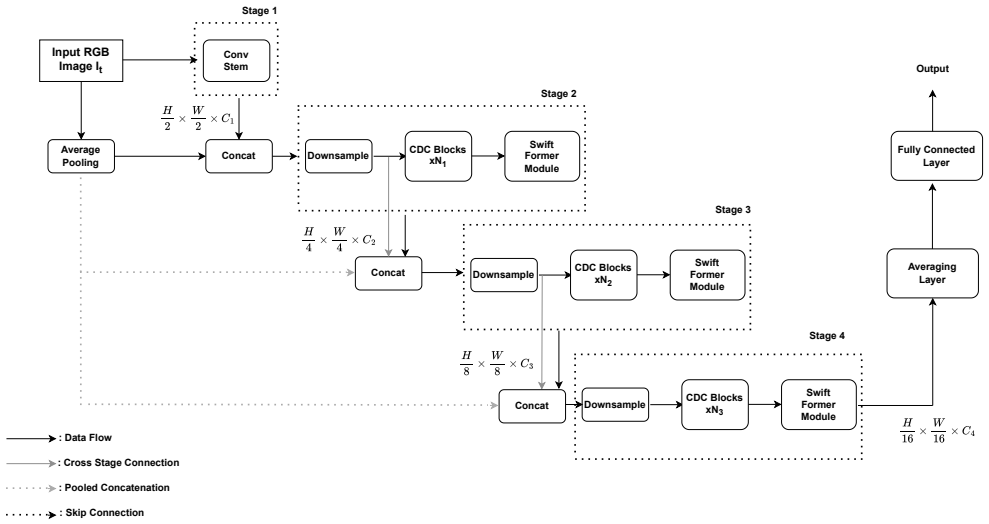


Figure 4.20: The object detector of the Swift-Former based depth estimator .

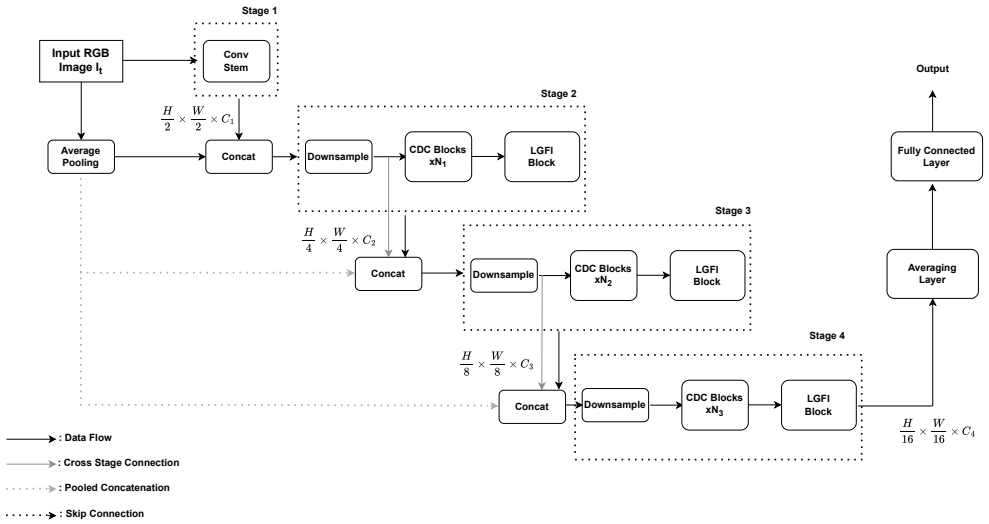


Figure 4.21: The object detector of the baseline depth estimator .

Algorithm 1: AdamW

Input: $\eta = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, $\lambda \in \mathbb{R}$
Output: Optimized parameters $\boldsymbol{\theta}_t$
Initialize: time step $t \leftarrow 0$, parameter vector $\boldsymbol{\theta}_{t=0} \in \mathbb{R}^n$, first moment vector $\mathbf{m}_{t=0} \leftarrow \mathbf{0}$, second moment vector $\mathbf{v}_{t=0} \leftarrow \mathbf{0}$, schedule multiplier $\eta_t = 0 \in \mathbb{R}$
Repeat until stopping criterion is met:
1: $t \leftarrow t + 1$
2: $\nabla f_t(\boldsymbol{\theta}_{t-1}) \leftarrow \text{SelectBatch}(\boldsymbol{\theta}_{t-1})$ ▷ Select batch and return the corresponding gradient
3: $\mathbf{g}_t \leftarrow \nabla f_t(\boldsymbol{\theta}_{t-1})$
4: $\mathbf{m}_t \leftarrow \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1)\mathbf{g}_t$ ▷ Update first moment
5: $\mathbf{v}_t \leftarrow \beta_2 \mathbf{v}_{t-1} + (1 - \beta_2)\mathbf{g}_t^2$ ▷ Update second moment
6: $\hat{\mathbf{m}}_t \leftarrow \frac{\mathbf{m}_t}{1-\beta_1^t}$ ▷ Bias correction for first moment
7: $\hat{\mathbf{v}}_t \leftarrow \frac{\mathbf{v}_t}{1-\beta_2^t}$ ▷ Bias correction for second moment
8: $\eta_t \leftarrow \text{SetScheduleMultiplier}(t)$ ▷ Schedule multiplier
9: $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \eta_t(\frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t}+\epsilon}) + \lambda\boldsymbol{\theta}_{t-1}$ ▷ Update parameters using weight decay
return Optimized parameters $\boldsymbol{\theta}_t$

Chapter 5

Evaluation

In the first part of this chapter, the Experimental Setup part is presented, and this part mainly focuses on the implementation details and dataset. The next part is the Performance Evaluation of the Modified Depth Models, and this part provides the complexity and performance comparisons between the modified and baseline depth models. After this part, the depth maps of the depth models are visualized and compared with each other in the Depth Map Visualization of the Proposed Depth Estimation Models part.

5.1 Experimental Set-up

This part is composed of 3 sections, which are Implementation & Training, Dataset, and Evaluation Metrics. In the Implementation & Training section, the hyper-parameters for the depth and pose estimation are given along with the GPU devices used in training. Additionally, the hyper-parameters used in the pre-training are also provided. The Dataset section provides brief information related to the dataset used in training and evaluation. Finally, the Evaluation Metrics section explains the performance and complexity metrics in detail.

5.1.1 Implementation and Training

The proposed and baseline depth networks are implemented on Pytorch [72]. AdamW, explained in Sec. 4.9, is selected as the optimizer for the depth and pose networks in the training. The weight decay in this optimizer is set to 1×10^{-2}. The pose network is composed of a ResNet18 encoder and a pose decoder with four convolutional layers. This pose estimation network is the same pose estimation network in [48, 14]. Zhang et al.'s Lite-Mono model [14] is chosen as the baseline depth model whose architecture is explained in Sec. 4.1. In the training of the depth model, the learning rate is initialized with 1×10^{-4}, and this learning rate is adjusted considering a cosine learning

rate schedule explained in Sec. 4.9. The values for the cosine schedules used in these networks are illustrated in Table 5.1.

	Depth Model	Pose Estimation Model
Initial Learning Rate	1×10^{-4}	1×10^{-4}
Maximum Learning Rate	1×10^{-4}	1×10^{-4}
Minimum Learning Rate	5×10^{-6}	1×10^{-5}
Learning Rate Cycle T	31	31

Table 5.1: The cosine scheduler values for depth and pose estimation networks

The depth and pose estimation models are trained by using the V100 and RTX2080Ti GPUs. The stopping criterion for the network is the saturation of the validation loss curves. These loss curves reflect the depth map quality metrics for the generated depth maps, explained in Sec. 5.1.3. To increase the robustness of the depth models, the augmentations proposed by Zhang et al. are followed in the pre-processing. These augmentations are performed with a %50 chance: horizontal flips, brightness adjustment ($\pm$0.2), saturation adjustment ($\pm$0.2), contrast adjustment ($\pm$0.2), and hue jitter ($\pm$0.1), which are applied in a random order [14]. During the training of the depth models, the batch size is selected as 12.

For the pre-training on ImageNet, the depth models are trained on two V100 GPUs. Similar to the training of the depth models, AdamW and cosine scheduler are used in the pre-training. AdamW uses weight decay with 0.05, and the learning rate is initialized with 0.001. The minimum learning rate and learning rate cycle are set to 1×10^{-6} and the total number of training steps over an epoch respectively. The batch size is set to 128. The stopping criterion of this pre-training is based on the model performance on the validation set.

While implementing the modified depth models, the GUB upsampling is integrated into the baseline, explained in Sec. 4.2. While integrating this module into the baseline, Belagiannis et al.'s GUB architecture [18] is utilized. Instead of using the RGB down-sampled images, the features extracted from the baseline are used as guides in the up-sampling, explained in detail in Sec. 4.2. While integrating the channel attention, Zhang et al.'s channel attention [14] block is applied to the concatenated features in the up-sampling on the decoder, explained in detail in Sec. 4.4. In the training of the MLP-mixer-based depth model, the LGFI model in the baseline model is replaced with a single MLP-mixer block, and the architecture of this modified model is explained in Sec. 4.5. While implementing the Spatial-Shift MLP-mixer-based depth model, the LGFI

module in the baseline's encoder is replaced with a single spatial-shift MLP-mixer, and this architecture is explained in detail in Sec. 4.6. While implementing the MLP-wave-based depth model, the LGFI module in the baseline's encoder is replaced with consecutive 2 MLP-wave mixers, and the architecture of this depth model is explained in Sec. 4.7. While implementing the Swift-Former-based depth model, the architecture of the Swift-Former is adapted from Shaker et al.'s Swift-Former architecture [58], and the LGFI module in the baseline's encoder is replaced with this architecture. This architecture is explained in detail in 4.8.

5.1.2 Dataset

In the evaluation, Make3D and KITTI test datasets are used as the test datasets while the KITTI training dataset is used to train the models as explained in Sec 4.1.1. The resolution of these two datasets is different from each other. KITTI dataset is composed of the 640×192 frames, whereas the Make3D dataset consists of 305×407 frames. The main reason behind using the Make3D dataset as a test dataset is to demonstrate the depth generalization capabilities of the depth models.

5.1.3 Evaluation Metrics

While assessing the quality of the depth maps generated from the monocular depth estimation networks, the ground truths of the depth maps are utilized. The metrics used in this assessment compare the ground truths of the depth map and generated depth maps considering image quality metrics. These metrics are root mean square, relative absolute error, relative square error, scale-invariant error and the threshold accuracies δ_j, as illustrated in Eq. 5.1 -5.5.

$$rmse = \sqrt{\frac{1}{P}\sum_{i=1}^{P}(y_i - \hat{y}_i)^2} \tag{5.1}$$

$$\text{rel} = \frac{1}{P}\sum_{i=1}^{P}\frac{|y_i - \hat{y}_i|}{y_i} \tag{5.2}$$

$$\log_{10} = \frac{1}{P}\sum_{i=1}^{P}|\log_{10}(y_i) - \log_{10}(\hat{y}_i)| \tag{5.3}$$

$$\delta_j = \max\left(\frac{y_i}{\hat{y}_i}, \frac{\hat{y}_i}{y_i}\right) < 1.25 \quad \text{for } j \in \{1, 2, 3\} \tag{5.4}$$

$$\text{sq\_rel} = \frac{1}{P}\sum_{i=1}^{P}\left(\frac{(y_i - \hat{y}_i)^2}{y_i^2}\right) \tag{5.5}$$

Depth maps with higher values of δ_i have higher quality compared to those with lower δ_i. Conversely, lower values of rmse, log10, sq_{rel}, and rel indicate higher quality for the depth maps. While assessing the complexity of the models, parameter size and FLOPS are considered as the complexity metrics. While evaluating these metrics, the thops [73] library is used.

5.2 Performance Evaluation of the Modified Depth Models

In this part of the thesis, the proposed monocular depth estimation frameworks are evaluated considering their complexity and performance on the depth datasets. Mean square, relative absolute error, relative square error, scale-invariant error and threshold accuracies are used as the metrics for the performance evaluation, whereas FLOPS and parameter size are considered as the criteria for the complexity of the models. While training and evaluating networks in terms of their performance and complexity, two training strategies are utilized: training with randomly initialized weights and training with weights initialized using the ImageNet dataset. The first strategy is employed to find the best-performing networks among the modified monocular depth estimation, After finding the best-performing networks whose weights are initialized randomly, the ImageNet dataset is used to pre-train these models to enhance their performance. The modified depth models are not only compared with each other but also with the baseline monocular depth estimation network, which is Zhang et al.'s Lite Mono architecture.

5.2.1 GUB-based Monocular Depth Estimation Model

Belagiannis et al. propose an image-based upsampling depth decoder. In this architecture. the features are extracted from pre-train DDRNet-23, and the downsampled RGB images are treated as the guide for the upsampling [18]. Similar to this approach, the baseline model's outputs are treated as guides and features during the upsampling operation. Based on the feature-guide interaction, two GUB-based monocular depth estimation decoder architectures are proposed. In the first approach, lower-resolution encoder outputs are treated as the guide, whereas the higher-resolution encoder outputs denote the features. In the second approach, the higher-resolution encoder outputs are treated as the guides, whereas the lower-resolution encoder outputs are con-

sidered as the features. To observe the impact of using pixel shuffling, pixel shuffle is applied to the baseline and GUB-based depth models. Tables 5.3 and 5.2 demonstrate the performance and complexity of the models respectively. As shown in Tables 5.2 and 5.3, the GUB model is the depth model proposed by Belagiannis et al., whereas Model A is the GUB-based model utilizing the lower-resolution encoder outputs as the guides. Model B is the GUB-based monocular depth estimation model employing the higher-resolution features as the guides. Moreover, Model C uses the same guides as Model B but this model lacks the 2-D convolutional layer before the GUB block. The last Model D is similar to Model B but it uses pixel shuffling while up-sampling the depth maps different from Model B,

	Params[M]	FLOPS[G]
Lite-Mono Baseline	3.069	5.032
GUB Model	5.824	4.237
Model A	3.578	6.612
Model B	3.574	6.598
Model C	3.499	6.764
Model D	3.068	5.076

Table 5.2: Parameter size and FLOPS of the GUB-based models. M and G denote 10^6 and 10^9

As shown in Table 5.2, the comparison between Model B and D demonstrates that the pixel shuffle operation leads to parameter size and FLOPS reduction. With the pixel shuffle operation, introducing the GUB module to the baseline does not lead to a significant increase in parameter size and FLOPS of the model. As shown in Table 5.3, adding the GUB to the baseline model does not lead to a performance increase, but the GUB-based baseline outperforms the GUB model proposed by Belagiannis et al., and the convolution following concatenation is a better strategy for up-sampling the depth maps in an upsampling scenario utilizing the multi-scale features. Pixel-shuffle operation slightly decreases the performance of the GUB-based baseline but it decreases the parameter size and FLOPS, as illustrated in Table 5.2. In Table 5.3, the performance values of Lite-Mono correspond to the values proposed by Zhang et al, whereas Lite-Mono (best chkp) demonstrates the performance of the Lite-Mono that I have trained.

	abs_{rel}	sq_{rel}	rmse	log10	δ_1	δ_2	δ_3
Lite-Mono	0.121	0.876	4.918	-	0.859	0.953	0.980
Lite-Mono (best chkp)	0.123	0.905	4.963	0.054	0.856	0.952	0.979
Model A	0.123	0.902	4.939	0.054	0.856	0.952	0.978
Model B	0.124	0.949	4.992	0.054	0.855	0.951	0.978
Model C	0.121	0.882	4.907	0.053	0.859	0.953	0.979
Model D	0.126	0.882	4.907	0.053	0.859	0.953	0.979
GUB model	0.142	-	5.480	0.063	0.784	0.936	0.981

Table 5.3: The performance evaluation of GUB-based models on the KITTI dataset.

5.2.2 Channel Attention-based Monocular Depth Estimation Model

The channel-attention scheme utilized in the baseline encoder is added to the up-sampling of the extracted features in the decoder. While adding this scheme to the decoder, two channel-attention schemes are proposed. The first scheme involves channel attention followed by a 2-D convolutional layer, whereas the other features a 2-D convolutional layer followed by channel attention, which is explained in detail in Sec. 4.4. Besides adding the channel attention-based up-sampling scheme to the decoder. pixel shuffling is also used to reduce the complexity of the model. Table 5.4 illustrates the complexity of the models having channel attention-based upsampling in the decoder.

	Params[M]	FLOPS[G]
Lite-Mono Baseline	3.069	5.032
GUB Model	5.824	4.237
Channel Attention Model 1	3.018	4.954
Channel Attention Model 2	2.969	4.794

Table 5.4: Parameter size and FLOPS of the Channel Attention-based models. M and G denote 10^6 and 10^9

Channel Attention-based Models 1 and 2 are explained in Sec. 4.4. The second channel attention model's parameter size and FLOPS are smaller than the parameter size and FLOPS of the baseline. Moreover, the second channel attention model's complexity is smaller than the complexity of the first channel attention model's complexity because the channel attention is applied after the 2-D convolution layer reducing the channel dimension of the features. The second channel attention-based upsampling scheme does not introduce significant complexity to the model, and using pixel shuf-

fling results in a model having a smaller complexity than the baseline model even with the introduction of the channel attention-based upsampling module in the decoder, as illustrated in Table 5.4.

	abs_{rel}	sq_{rel}	rmse	log10	δ_1	δ_2	δ_3
Lite-Mono	0.121	0.876	4.918	-	0.859	0.953	0.980
Lite-Mono (best chkp)	0.123	0.905	4.963	0.054	0.856	0.952	0.979
Channel Attention Model 1	0.124	0.925	4.978	0.054	0.852	0.952	0.979
Channel Attention Model 2	0.120	0.895	4.889	0.054	0.854	0.952	0.981
GUB model	0.142	-	5.480	0.063	0.784	0.936	0.981

Table 5.5: The performance evaluation of Channel Attention-based models on the KITTI dataset.

As shown in Table 5.5, the performance of the depth model utilizing the second channel attention-based upsampling has a slightly better performance in terms of abs_{rel} and $rmse$. Moreover, adding the pixel shuffle to the decoder reduces the complexity introduced by the channel attention. The depth model with this pixel shuffle operation and channel attention module is a contender depth model against the baseline depth model considering its complexity and performance on the KITTI test dataset.

5.2.3 MLP Mixer-based Monocular Depth Estimation Models

In this part of the thesis, the performance and complexity of monocular depth estimation models based on the MLP-mixer architecture, the MLP-mixer, spatial MLP-mixer, and MLP wave-mixer depth models, are examined. While incorporating these MLP mixers into the baseline depth model, the LGFI module in the baseline is replaced with the mixer models. These mixer models are explained in Sec. 3.4. The complexities of the MLP mixer-based depth estimation models are demonstrated in Table 5.6.

	Params[M]	FLOPS[G]
Lite-Mono Baseline	3.069	5.032
GUB Model	5.824	4.237
MLP Mixer Model	4.073	4.898
Spatial MLP Mixer Model	3.669	4.643
MLP Wave Mixer Model	2.914	4.486

Table 5.6: Parameter size and FLOPS of the MLP mixer-based models. M and G denote 10^6 and 10^9

As shown in Table 5.6, the depth model having the lowest complexity is the MLP Wave Model, and the other MLP Mixer-based depth models have higher parameter size and FLOPS than the baseline model. The performances of the mixer-based depth models are illustrated in Table 5.7. The spatial and wave MLP mixers have better performance on the KITTI dataset compared to the MLP mixer-based depth and GUB model. The performance of the Wave MLP mixer is slightly worse than the baseline depth model, but it has slightly less complexity than the baseline depth model considering the parameter size and FLOPS.

	abs_{rel}	sq_{rel}	rmse	log10	δ_1	δ_2	δ_3
Lite-Mono	0.121	0.876	4.918	-	0.859	0.953	0.980
Lite-Mono (best chkp)	0.123	0.905	4.963	0.054	0.856	0.952	0.979
MLP Mixer Model	0.141	1.165	5.471	0.061	0.826	0.938	0.972
Spatial MLP Mixer Model	0.121	0.922	4.967	0.053	0.862	0.953	0.979
MLP Wave Mixer Model	0.122	0.901	4.928	0.054	0.860	0.953	0.979
GUB model	0.142	-	5.480	0.063	0.784	0.936	0.981

Table 5.7: The performance of MLP Mixer-based models on the KITTI dataset.

5.2.4 Swift Former-based Monocular Depth Estimation Model

The Swift-Former is composed of Efficient-Additive attention and CDC block. This attention mechanism is a lightweight attention mechanism explained in Sec. 4.8 and 3.3.4. The performance and complexity comparisons of the Swift-Former-based depth model with the previous depth models are illustrated in Table 5.8 and 5.9 respectively.

	abs_{rel}	sq_{rel}	rmse	log10	δ_1	δ_2	δ_3
Lite-Mono	0.121	0.876	4.918	-	0.859	0.953	0.980
Lite-Mono (best chkp)	0.123	0.905	4.963	0.054	0.856	0.952	0.979
MLP Mixer Model	0.141	1.165	5.471	0.061	0.826	0.938	0.972
Spatial MLP Mixer Model	0.121	0.922	4.967	0.053	0.862	0.953	0.979
MLP Wave Mixer Model	0.122	0.901	4.928	0.054	0.860	0.953	0.979
GUB model	0.142	-	5.480	0.063	0.784	0.936	0.981
Channel Attention Model 1	0.124	0.925	4.978	0.054	0.852	0.952	0.979
Channel Attention Model 2	0.120	0.895	4.889	0.054	0.854	0.952	0.981
Swift Former Model	0.119	0.880	4.854	0.052	0.864	0.954	0.980

Table 5.8: The performance evaluation of Swift-Former-based model on the KITTI dataset.

	Params[M]	FLOPS[G]
Lite-Mono Baseline	3.069	5.032
GUB Model	5.824	4.237
MLP Mixer Model	4.073	4.898
Spatial MLP Mixer Model	3.669	4.643
MLP Wave Mixer Model	2.914	4.486
Channel Attention Model 1	3.018	4.954
Channel Attention Model 2	2.969	4.794
Swift Former Model	2.920	4.705

Table 5.9: Parameter size and FLOPS of the Swift-Former-based models. M and G denote 10^6 and 10^9

As shown in Table 5.8, the Swift-Former-based depth model has slightly better performance than the baseline depth model and previous modified depth models. Furthermore, the Swift-Former used in the encoder does not lead to a significant increase in the parameter size and FLOPS, but it has a lower complexity in terms of the parameter size and FLOPS than the baseline depth model with the help of the pixel shuffling operations in the decoder, as shown in Table 5.9.

5.2.5 Monocular Depth Estimation Models with Pre-training on ImageNet

While evaluating the performance of the pre-trained depth models, Zhang et al.'s depth encoder checkpoint is used in the depth models having channel attention-based decoder, whereas the Swift-Former based encoders are trained on the ImageNet dataset by appending averaging and fully connected layers to their encoders. While training the Swift-Former on the ImageNet, top-1 and top-5 classifications are considered as the metrics for the model performance, demonstrated in Fig. 5.1. Moreover, while training the Swift-Former model, cross-entropy is selected as the loss function, and the training and testing losses are demonstrated in Fig. 5.1. The 98^{th} Swift-Former checkpoint having 71.612 top-1 test accuracy and 90.124 top-5 test accuracy is selected as the pre-train checkpoint for the depth estimation network.

Apart from this pre-training, the LGFI module composed of channel attention and point-wise convolution blocks is used in the decoder, and its performance and complexity are compared with the performance and complexity of the depth modules utilizing the channel attention blocks. The modules utilizing the LGFI modules instead of the channel attention blocks are termed modified LGFI depth modules.

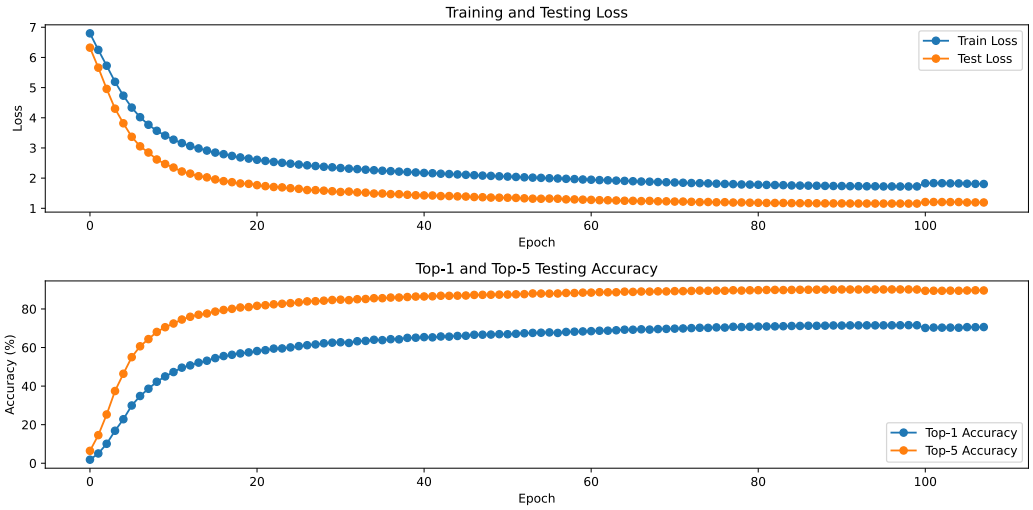


Figure 5.1: Training and testing losses in Swift-Former based depth model. The performance of Swift-Former on the Imagenet test dataset.

	Params[M]	FLOPS[G]
Channel Attention Model 2	2.969	4.794
Channel Attention Model 2 Modified LGFI	3.038	4.314
Swift Former Model	2.920	4.705
Swift Former Model Modified LGFI	2.989	4.947

Table 5.10: Parameter size and FLOPS of the modified LFGI depth models. M and G denote 10^6 and 10^9

Table 5.11 illustrates the performances of the proposed depth models on the KITTI dataset. Lite-Mono (Given chkp) is Zhang et al.'s pre-trained depth model, whereas Lite-Mono (Best chkp) is the Lite-Mono depth model that is trained within this thesis. The Swift-Former depth model utilizing the channel attention-based up-sampling block in the decoder has a slightly better performance than Zhang et al's proposed model. Apart from this, this model shows slightly better performance than the other proposed depth models in this thesis. Furthermore, the complexity introduced by attention-based upsampling block can be reduced by pixel shuffle operations, as shown in Table 5.10. The SwiftFormer and channel attention-based models have slightly better performance and smaller complexity than the baseline model, as demonstrated in Tables 5.10 and 5.11.

	abs_{rel}	sq_{rel}	rmse	log10	δ_1	δ_2	δ_3
Lite-Mono (Given chkp)	0.107	0.765	4.561	0.047	0.886	0.963	0.983
Lite-Mono (Best chkp)	0.107	0.818	4.627	0.047	0.887	0.962	0.983
GUB	0.142	-	5.480	0.063	0.784	0.936	0.981
2^{nd} Channel Attention	0.107	0.774	4.595	0.047	0.883	0.962	0.983
2^{nd} Channel Attention Modified	0.106	0.752	4.591	0.047	0.883	0.961	0.983
Swift Former Channel Attention	0.106	0.736	4.537	0.047	0.885	0.963	0.983
Swift Former Channel Attention Modified	0.104	0.754	4.529	0.046	0.890	0.963	0.983

Table 5.11: The performance evaluation of proposed pre-trained depth modules on the KITTI dataset. chkp is the abbreviation of the checkpoint.

	abs_{rel}	sq_{rel}	rmse	log10
Lite-Mono (Given chkp)	0.305	3.060	6.981	0.158
Lite-Mono (Best chkp)	0.318	3.451	7.254	0.163
Lite-Mono (Paper)	0.305	3.060	6.981	0.158
2^{nd} Channel Attention	0.308	2.871	6.929	0.160
2^{nd} Channel Attention Modified	0.308	3.016	7.013	0.160
Swift Former Channel Attention	0.306	3.247	7.096	0.156
Swift Former Channel Attention Modified	0.337	3.841	7.452	0.168

Table 5.12: Performance evaluation of proposed pre-trained depth modules on the Make3D dataset. "chkp" is the abbreviation of the checkpoint, whereas the Paper term refers to the performance values proposed by Zhang et al. in their paper.

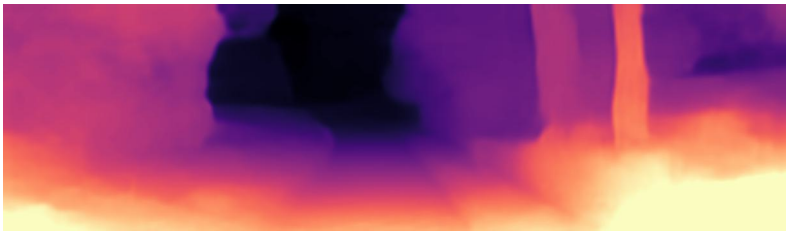
After evaluating the performance of the pre-trained proposed models on the KITTI test dataset, the Make3D dataset is utilized to illustrate the generalization capabilities of the proposed depth models. Therefore, this dataset is only used in testing, and this dataset is composed of images with different resolutions. The performance comparison on the Make3D dataset is illustrated in Table 5.12. As shown in 5.12, the model having the best performance among the models is Zhang et al.'s Lite-Mono model but this model has higher complexity than the models proposed in the thesis. The second-best performing model is the Swift Former-based depth model utilizing the channel attention in the decoder. This model is lighter than the Lite-Mono model and the performance of this model does not drop significantly as illustrated in Table 5.12. The Swift Former-based depth model utilizing the LGFI block in the decoder has poor performance scores compared to the other proposed depth models. The reason behind this might be that the model might be over-fitted on the KITTI dataset. To

understand how these depth models generate their corresponding depth maps, the depth map outputs are visualized in the next section.

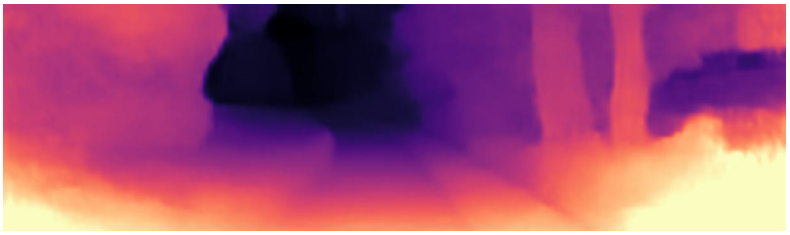
5.3 Depth-map Visualization of the Proposed Depth Estimation Models

In this section, sample images from the KITTI and Make3D datasets are employed to assess the visual quality of the depth maps. For the visualizations, test images from the KITTI dataset and Make3D are used. The depth maps generated from the baseline model, Swift-former-based model utilizing the channel-attention upsampling scheme and Swift Former-based model using LGFI upsampling scheme are visualized as demonstrated in Fig. 5.2-5.11. As shown in Fig. 5.2-5.7, the Swift-Former-based model utilizing the channel-attention-based upsampling scheme has a better performance in creating depth maps than the other models. The depth maps generated by the Lite-Mono model are more noisy than depth maps created by the Swift-Former-based depth model utilizing the channel attention-based upsampling scheme. Also, this Swift Former-based model captures the structures of the objects better than the other models when there are no transparent objects or shadows. The illumination in the images affects the quality of the depth maps created by the depth models, as visualized in Fig. 5.7, 5.10 and 5.11. As shown in Fig 5.10 and 5.11, the shadows and transparent objects distort the Swift former-based depth models more significantly than the Lite-Mono model. Moreover, these affect the structure of the objects in the background as visualized in 5.11. The structure of the transparent objects is captured more precisely in the Lite-Mono depth model. In Fig 5.10, the edges and frames of the windows are more visible in the Lite-Mono model compared with the other generated depth maps. Moreover, Swift-Former-based depth models are not robust to the sharp transitions in color caused by shadows, and these sharp transitions also affect the structure of the objects in the background, as visualized in 5.11. The transparent objects and sharp color transitions caused by shadows are the main obstacles in creating high-quality depth maps by using the Swift-Former-based depth models.

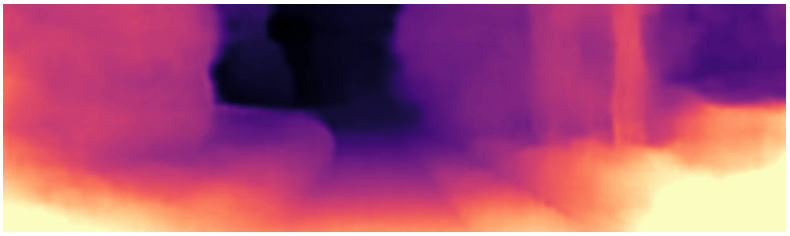
(a) Original RGB image from the KITTI dataset



(b) Depth map generated from Lite Mono model



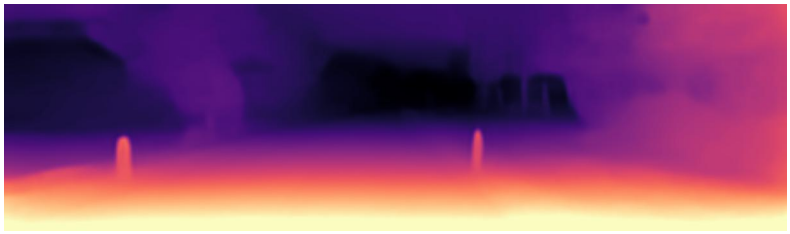
(c) Depth map generated from Swift-Former based depth model using LGFI based upsampling



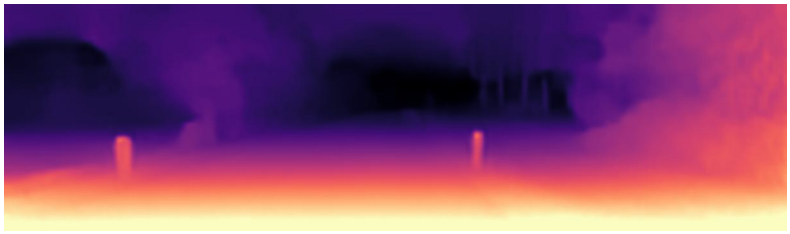
(d) Depth map generated from Swift-Former based depth model using channel-attention based upsampling

Figure 5.2: KITTI image and corresponding depth maps from the depth models

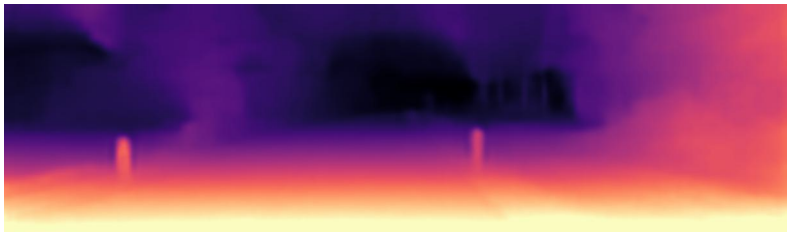
(a) Original RGB image from the KITTI dataset



(b) Depth map generated from Lite Mono model

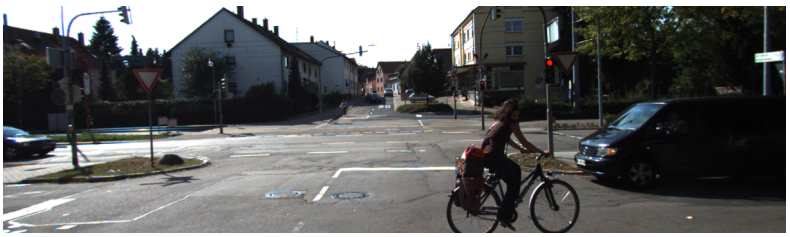


(c) Depth map generated from Swift-Former based depth model using LGFI based upsampling



(d) Depth map generated from Swift-Former based depth model using channel-attention based upsampling

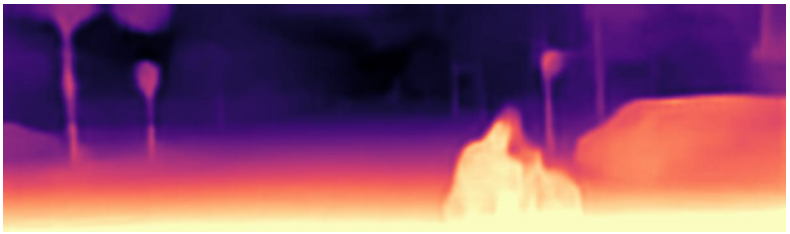
Figure 5.3: KITTI image and corresponding depth maps from the depth models



(a) Original RGB image from the KITTI dataset

(b) Depth map generated from Lite Mono model

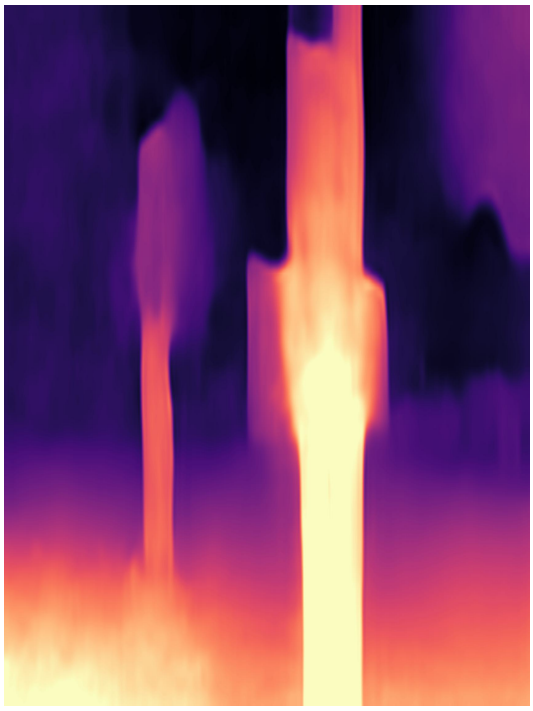
(c) Depth map generated from Swift-Former based depth model using LGFI based upsampling



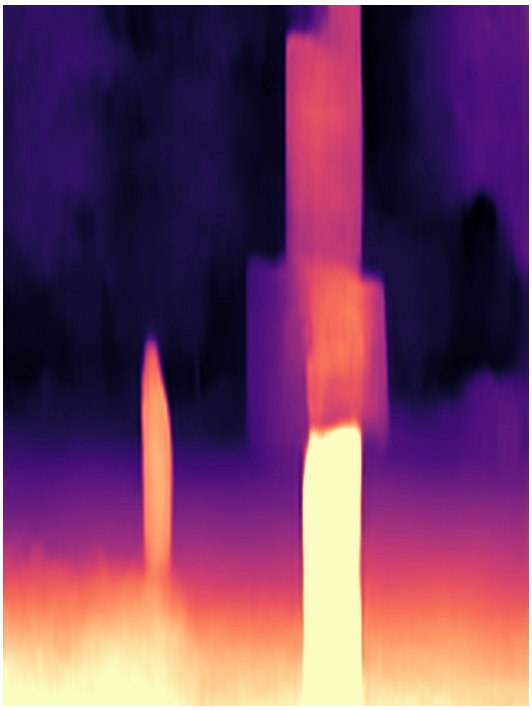
(d) Depth map generated from Swift-Former based depth model using channel-attention based upsampling

Figure 5.4: KITTI image and corresponding depth maps from the depth models

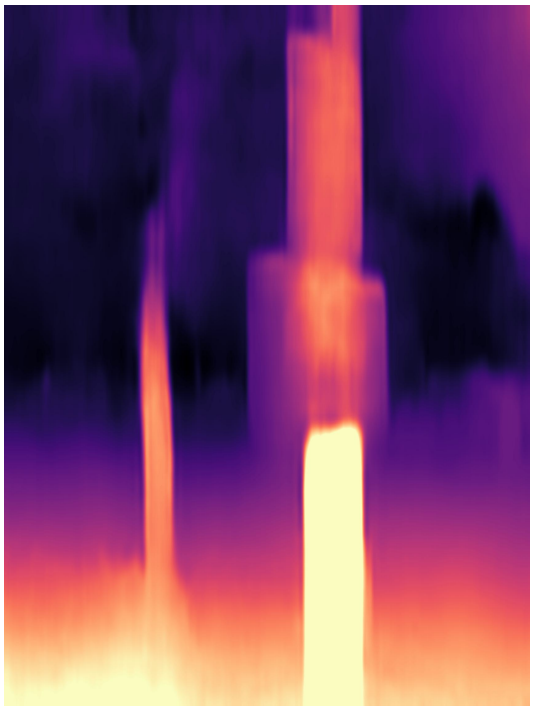
(a) Original RGB image from the Make3D dataset



(b) Depth map generated from Lite Mono model



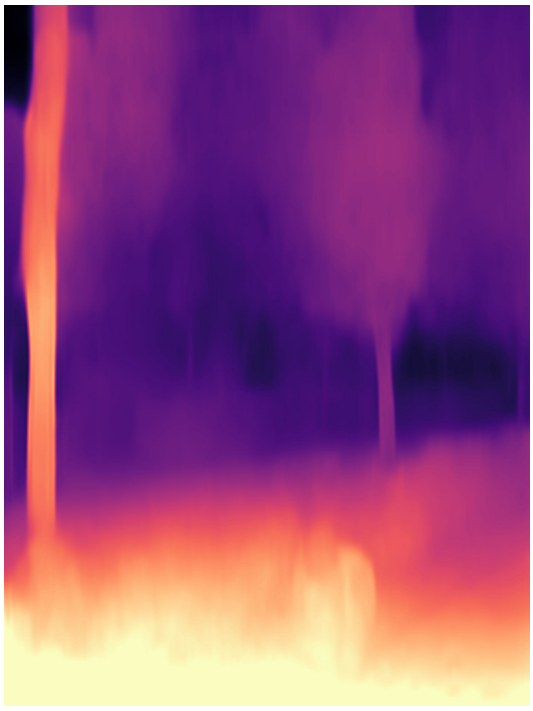
(c) Depth map generated from Swift-Former based depth model using LGFI-based upsampling



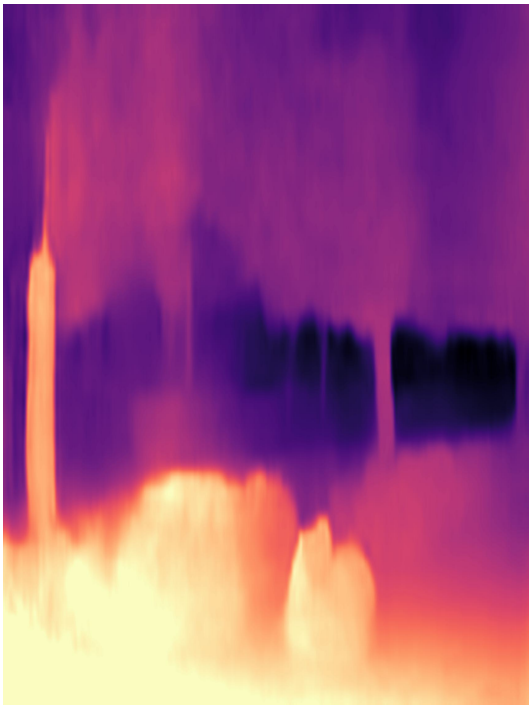
(d) Depth map generated from Swift-Former based depth model using channel-attention-based upsampling

Figure 5.5: Make3D image and corresponding depth maps from the depth models

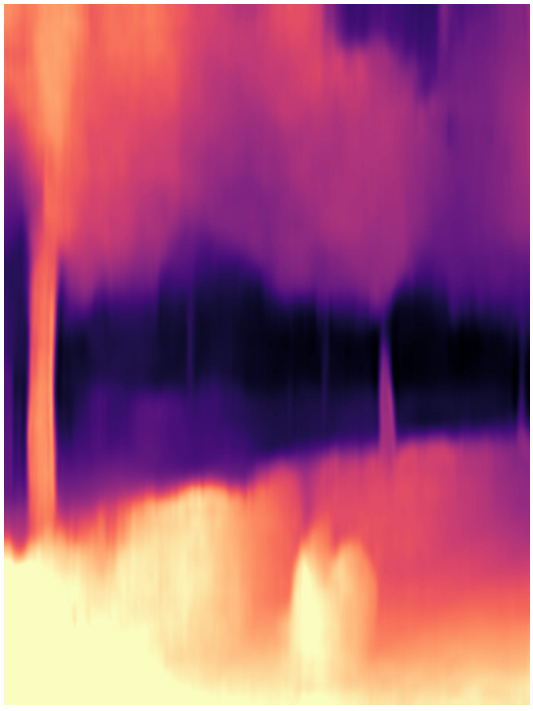
(a) Original RGB image from the Make3D dataset



(b) Depth map generated from Lite Mono model



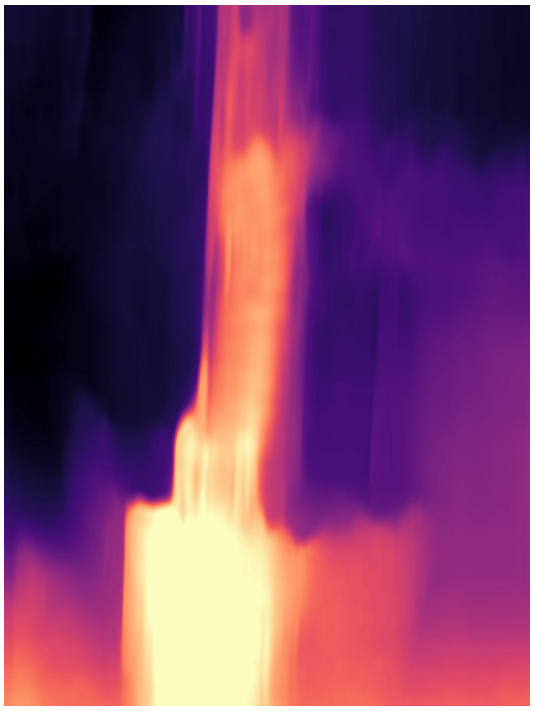
(c) Depth map generated from Swift-Former based depth model using LGFI-based upsampling



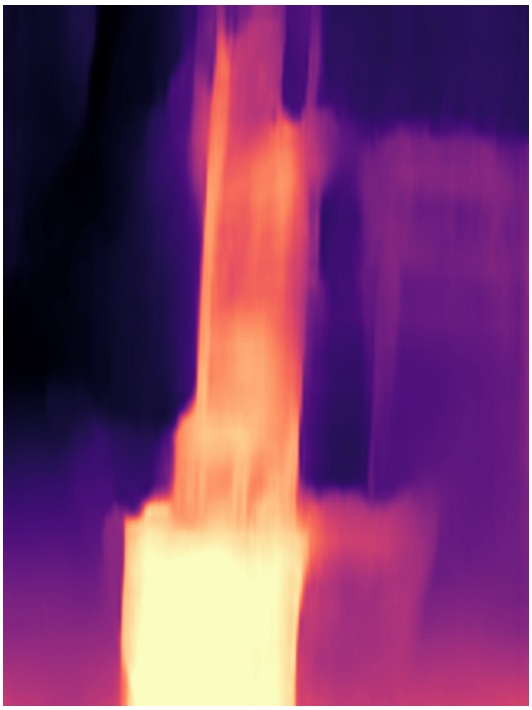
(d) Depth map generated from Swift-Former based depth model using channel-attention-based upsampling

Figure 5.6: Make3D image and corresponding depth maps from the depth models

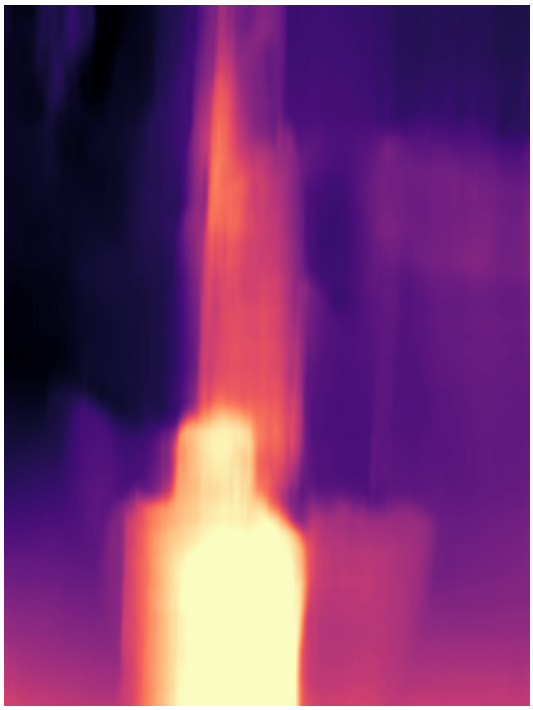
(a) Original RGB image from the Make3D dataset



(b) Depth map generated from Lite Mono model



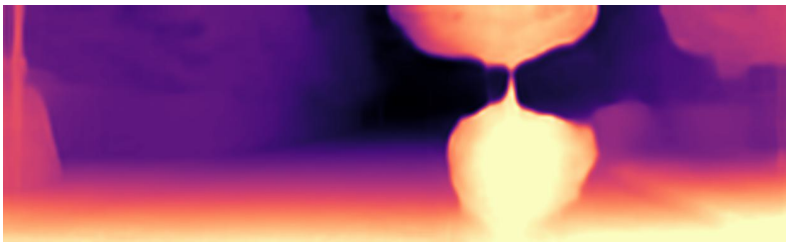
(c) Depth map generated from Swift-Former based depth model using LGFI-based upsampling



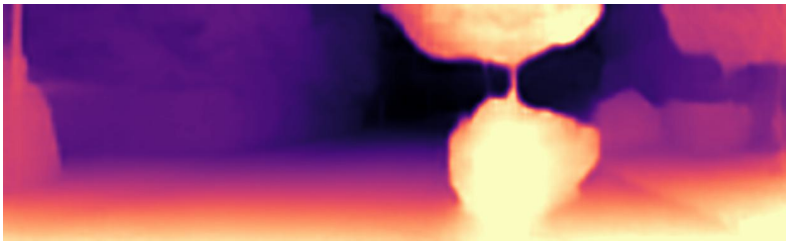
(d) Depth map generated from Swift-Former based depth model using channel-attention-based upsampling

Figure 5.7: Make3D image and corresponding depth maps from the depth models

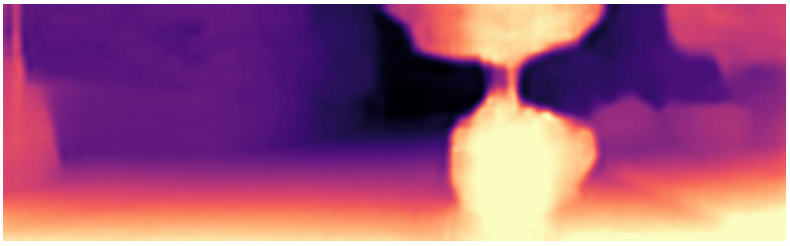
(a) Original RGB image from the KITTI dataset



(b) Depth map generated from Lite Mono model



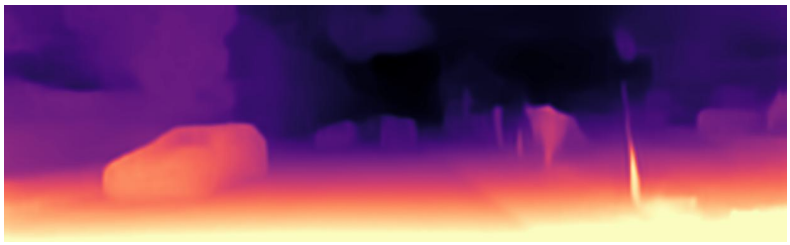
(c) Depth map generated from Swift-Former based depth model using LGFI based upsampling



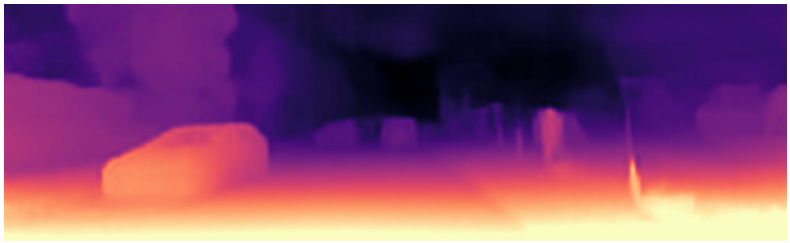
(d) Depth map generated from Swift-Former based depth model using channel-attention based upsampling

Figure 5.8: KITTI image and corresponding depth maps from the depth models

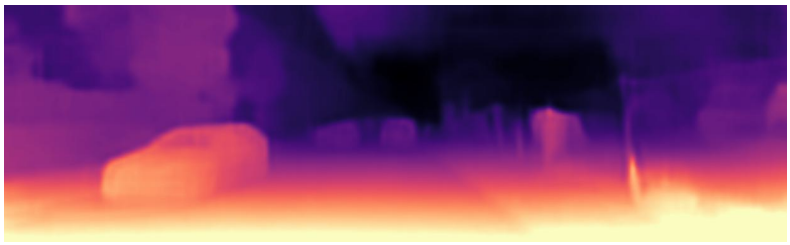
(a) Original RGB image from the KITTI dataset



(b) Depth map generated from Lite Mono model



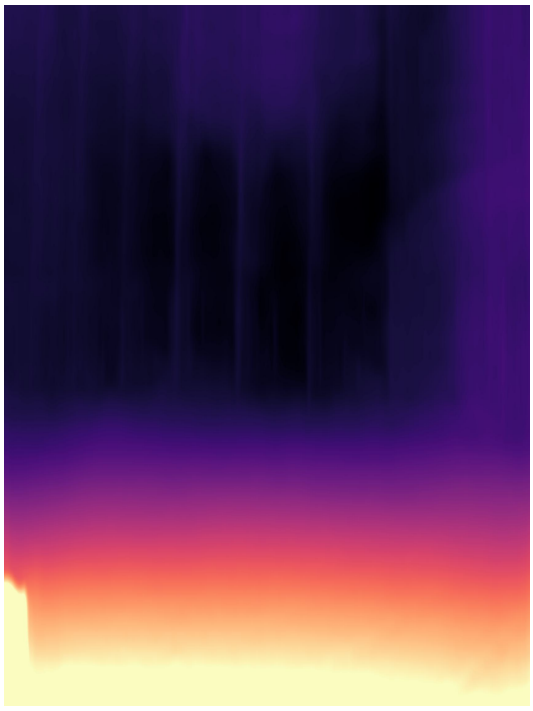
(c) Depth map generated from Swift-Former based depth model using LGFI based upsampling



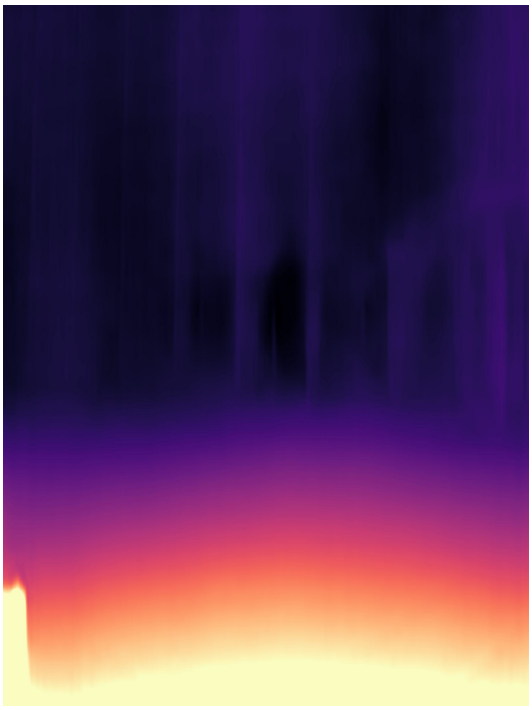
(d) Depth map generated from Swift-Former based depth model using channel-attention based upsampling

Figure 5.9: KITTI image and corresponding depth maps from the depth models

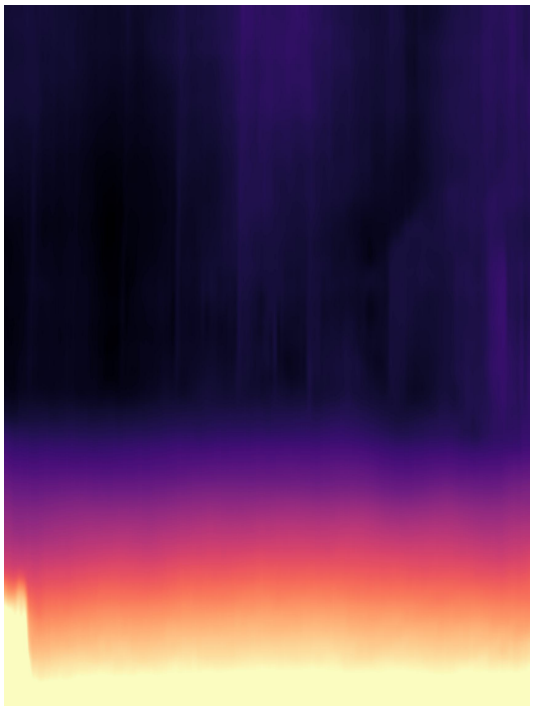
(a) Original RGB image from the Make3D dataset



(b) Depth map generated from Lite Mono model



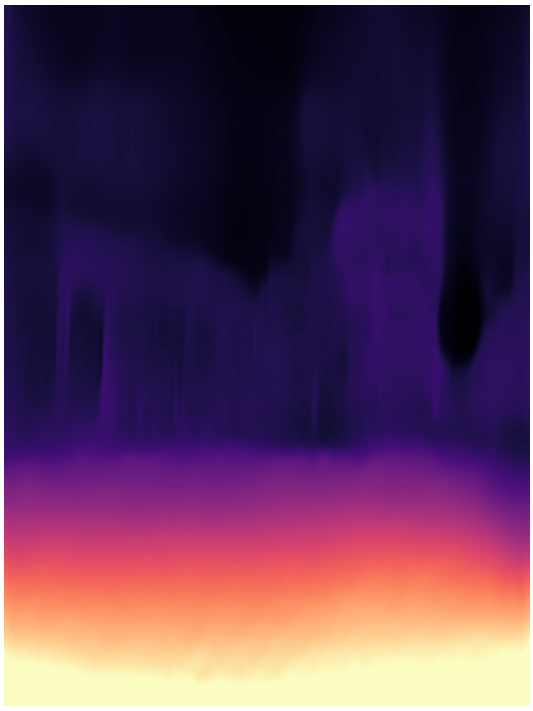
(c) Depth map generated from Swift-Former based depth model using LGFI-based upsampling



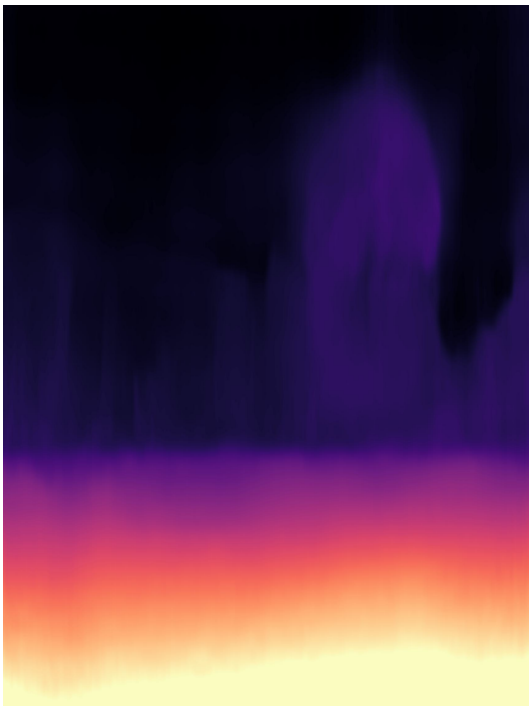
(d) Depth map generated from Swift-Former based depth model using channel-attention-based upsampling

Figure 5.10: Make3D image and corresponding depth maps from the depth models

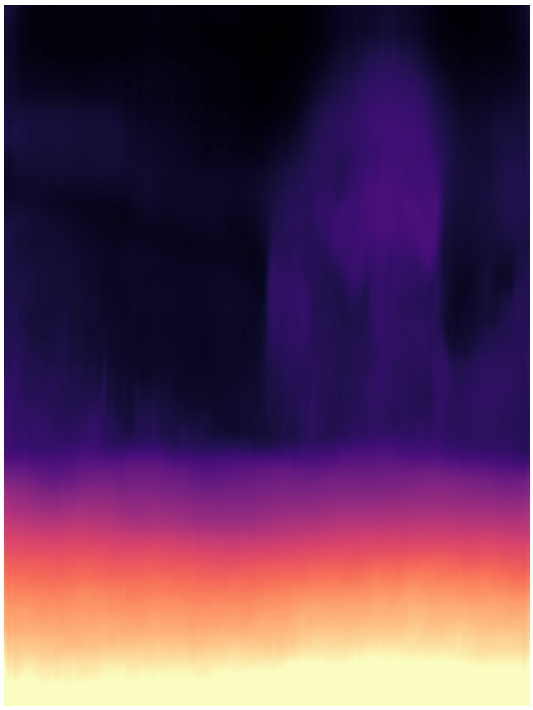
(a) Original RGB image from the Make3D dataset



(b) Depth map generated from Lite Mono model



(c) Depth map generated from Swift-Former based depth model using LGFI-based upsampling



(d) Depth map generated from Swift-Former based depth model using channel-attention-based upsampling

Figure 5.11: Make3D image and corresponding depth maps from the depth models

Chapter 6

Conclusion

In this thesis, the Lite-Mono model was selected as the monocular depth estimation baseline. The LGFI module in the baseline's encoder was replaced with lightweight attention-based and substitute architectures for the attention module. These substitute architectures were mainly MLP-mixer-based models which were MLP mixer, spatial MLP mixer and MLP wave mixer models. The performance and complexity change while integrating these modules was investigated. Apart from these attention-substitute models, the LGFI module in the baseline's encoder was replaced with a light attention model called Swift Former architecture that utilized the efficient additive attention module. Furthermore, a channel attention-based upsampling scheme in the depth decoder was proposed. Pixel shuffle, a re-arrangement of the feature pixels, was utilized to alleviate the computation burden introduced by this channel attention module. The channel attention-based upsampling scheme's impacts on the depth model's complexity and performance were investigated. While comparing the performance of these modified depth networks with the baseline, root mean square, relative absolute error, relative square error, scale-invariant error and the threshold accuracies δ_j were selected as quality metrics for the generated depth maps, whereas parameter size and FLOPS were chosen as the criteria for assessing the complexity of the models.

While integrating these modifications into the baseline, the KITTI training dataset was used in training, whereas the KITTI test and Make3D datasets were used in evaluating the trained models' performances. The channel attention-based upsampling slightly increased the quality of the generated depth map without introducing additional model complexity on the KITTI dataset. The pixel shuffle operation used in this upsampling scheme reduced the overall complexity of the modified models compared to the baseline. Using the mixer-based modules in the depth model did not lead to a performance increase or decrease in the model complexity, unlike the channel attention-based upsampling scheme. Apart from these modifications on the baseline, the Swift-Former architecture led to a decrease in the complexity of the depth model

since it did not use the V matrix to create the Q-K interaction. In addition to this reduction in the complexity of the depth model, the performance of this model was higher than the baseline model on the KITTI dataset. When the Make3d dataset was used to evaluate this model's performance, the Swift-Former-based model had a slightly worse performance compared with the baseline model. However, the performance gap between these models was not significant and this might have originated from the samples having different resolution from the training samples. While evaluating the performance of the proposed models against the baseline, both the provided Lite-Mono checkpoint and the checkpoint trained within this thesis were taken into account.

In conclusion, the Swift-Former depth model utilizing the channel attention-based upsampling had a slightly better performance than the baseline on the KITII dataset. The overall model complexity of this proposed model was also lighter than the baseline. However, when the Make3D dataset is used to evaluate the generalization capabilities in generating depth maps, this proposed model demonstrated slightly worse performance than the baseline. This might be related to the illumination of the scenes and the resolution difference between the test and training samples. .

6.1 Future Works

While evaluating the model's performance on the Make3D dataset, some of the generated depth maps tended to be affected by the light. To get ahead of this problem, the scenes with different illuminations can be added to the KITTI dataset, or after the training on the KITTI dataset, the model can be trained on a different dataset to enhance its capability in producing the general depth maps of the scenes. The resolution of the images also affects the quality of the generated depth maps. To reduce the dependency on the resolution of the images, augmentation schemes specialized in the resolution might be followed. Apart from these, synthetic scene data can be used to boost the performance of the depth models, and model quantization techniques on these models can be investigated to obtain the models having lower complexities. Apart from these works, the model compression techniques can be applied to the depth models to make depth models lighter.

List of Figures

List of Tables

Bibliography

[1] D. Hoiem, "Epipolar geometry and stereo vision."

[2] A. Kamenev, "Stereo dnn tensorrt inference library." [Online]. Available: https://github.com/NVIDIA-AI-IOT/redtail/tree/master/stereoDNN

[3] Y. Xu, X. Yang, Y. Yu, W. Jia, Z. Chu, and Y. Guo, "Depth estimation by combining binocular stereo and monocular structured-light," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 1736–1745.

[4] C. B. Hochberg and J. E. Hochberg, "Familiar size and the perception of depth," *The Journal of Psychology*, vol. 34, no. 1, pp. 107–114, 1952. [Online]. Available: https://doi.org/10.1080/00223980.1952.9916110

[5] F.-T. Hong, L. Zhang, L. Shen, and D. Xu, "Depth-aware generative adversarial network for talking head video generation," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 3387–3396.

[6] H.-N. Hu, Q.-Z. Cai, D. Wang, J. Lin, M. Sun, P. Krahenbuhl, T. Darrell, and F. Yu, "Joint monocular 3d vehicle detection and tracking," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[7] Y. Yang, S. Shao, T. Yang, P. Wang, Z. Yang, C. Wu, and H. Liu, "A geometry-aware deep network for depth estimation in monocular endoscopy," *Engineering Applications of Artificial Intelligence*, vol. 122, p. 105989, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197623001732

[8] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *Computer Vision – ECCV 2012*, A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 746–760.

[9] D. Hoiem, A. A. Efros, and M. Hebert, "Automatic photo pop-up," *ACM SIGGRAPH 2005 Papers*, 2005. [Online]. Available: https://api.semanticscholar.org/CorpusID:834882

[10] 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.

[11] R. Hadsell, P. Sermanet, J. Ben, A. Erkan, M. Scoffier, K. Kavukcuoglu, U. Muller, and Y. LeCun, "Learning long-range vision for autonomous off-road driving," *Journal of Field Robotics*, vol. 26, no. 2, pp. 120–144, 2009.

[12] J. Michels, A. Saxena, and A. Ng, "High speed obstacle avoidance using monocular vision and reinforcement learning," *Proceedings of the 22nd international conference on Machine learning*, 2005. [Online]. Available: https://api.semanticscholar.org/CorpusID:2035627

[13] A. M. Haubenwaller and K. Vandikas, "Computations on the edge in the internet of things," *Procedia Computer Science*, vol. 52, pp. 29–34, 2015, the 6th International Conference on Ambient Systems, Networks and Technologies (ANT-2015), the 5th International Conference on Sustainable Energy Information Technology (SEIT-2015). [Online]. Available: https://www.sciencedirect.com/science/article/pii/S187705091500811X

[14] 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)*, June 2023, pp. 18 537–18 546.

[15] S. Chen, T. Cheng, J. Fang, Q. Zhang, Y. Li, W. Liu, and X. Wang, "Tinydet: Accurate small object detection in lightweight generic detectors," 2023.

[16] G. Xu, J. Li, G. Gao, H. Lu, J. Yang, and D. Yue, "Lightweight real-time semantic segmentation network with efficient transformer and cnn," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 12, pp. 15 897–15 906, 2023.

[17] C. Ning and H. Gan, "Trap attention: Monocular depth estimation with manual traps," in *Proceedings of the IEEE/CVF International Conference on Computer Vision and Pattern Recognition*, 2023.

[18] M. Rudolph, Y. Dawoud, R. Güldenring, L. Nalpantidis, and V. Belagiannis, "Lightweight monocular depth estimation through guided decoding," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 2344–2350.

[19] C. Zhao, Y. Zhang, M. Poggi, F. Tosi, X. Guo, Z. Zhu, G. Huang, Y. Tang, and S. Mattoccia, "Monovit: Self-supervised monocular depth estimation with a vision transformer," in *2022 International Conference on 3D Vision (3DV)*. IEEE, Sep. 2022. [Online]. Available: http://dx.doi.org/10.1109/3DV57658.2022.00077

[20] J. Bae, S. Moon, and S. Im, "Monoformer: Towards generalization of self-supervised monocular depth estimation with transformers," 05 2022.

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

[22] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[23] H. Ni and S. Jiang, "Deep dual-resolution road scene segmentation networks based on decoupled dynamic filter and squeeze–excitation module," *Sensors*, vol. 23, no. 16, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/16/7140

[24] Y. Tadepalli, M. Kollati, S. Kuraparthi, and P. Kora, "Efficientnet-b0 based monocular dense-depth map estimation," *Traitement du Signal*, vol. 38, no. 5, p. 1485–1493, Oct 2021.

[25] K. He, J. Sun, and X. Tang, "Guided image filtering," in *Proceedings of the 11th European Conference on Computer Vision: Part I*, ser. ECCV'10. Berlin, Heidelberg: Springer-Verlag, 2010, p. 1–14.

[26] Y. Li, J. Huang, N. Ahuja, and M. Yang, "Deep joint image filtering," in *Computer Vision - 14th European Conference, ECCV 2016, Proceedings*, ser. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Germany: Springer, 2016, pp. 154–169, publisher Copyright: ©

Springer International Publishing AG 2016.; 14th European Conference on Computer Vision, ECCV 2016 ; Conference date: 11-10-2016 Through 14-10-2016.

[27] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *Neural Information Processing Systems*, 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:2255738

[28] B. Li, C. Shen, Y. Dai, A. van den Hengel, and M. He, "Depth and surface normal estimation from monocular images using regression on deep features and hierarchical crfs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[29] Y. Cao, Z. Wu, and C. Shen, "Estimating depth from monocular images as classification using deep fully convolutional residual networks," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 11, pp. 3174–3182, 2018.

[30] W. Yin, Y. Liu, C. Shen, and Y. Yan, "Enforcing geometric constraints of virtual normal for depth prediction," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 5683–5692.

[31] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks," in *3D Vision (3DV), 2016 Fourth International Conference on*. IEEE, 2016, pp. 239–248.

[32] J.-H. Lee and C.-S. Kim, "Multi-loss rebalancing algorithm for monocular depth estimation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[33] X. Dong, M. A. Garratt, S. G. Anavatti, H. A. Abbass, and J. Dong, "Lightweight monocular depth estimation with an edge guided network," *2022 17th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pp. 204–210, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:252596051

[34] M.-J. Chiu, W.-C. Chiu, H.-T. Chen, and J.-H. Chuang, "Real-time monocular depth estimation with extremely light-weight neural network," in *2020 25th International Conference on Pattern Recognition (ICPR)*, 2021, pp. 7050–7057.

[35] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient

sub-pixel convolutional neural network," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 1874–1883.

[36] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017.

[37] T.-J. Yang, A. G. Howard, B. Chen, X. Zhang, A. Go, V. Sze, and H. Adam, "Netadapt: Platform-aware neural network adaptation for mobile applications," in *European Conference on Computer Vision*, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:4746618

[38] T. Chen, T. Moreau, Z. Jiang, L. Zheng, E. Q. Yan, H. Shen, M. Cowan, L. Wang, Y. Hu, L. Ceze, C. Guestrin, and A. Krishnamurthy, "Tvm: An automated end-to-end optimizing compiler for deep learning," in *USENIX Symposium on Operating Systems Design and Implementation*, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:52939079

[39] D. Wofk, F. Ma, T.-J. Yang, S. Karaman, and V. Sze, "Fastdepth: Fast monocular depth estimation on embedded systems," *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6101–6108, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:72940839

[40] M. Stommel, M. Beetz, and W. Xu, "Inpainting of missing values in the kinect sensor's depth maps based on background estimates," *IEEE Sensors Journal*, vol. 14, pp. 1107–1116, 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:39915686

[41] S. Saxena, A. Kar, M. Norouzi, and D. J. Fleet, "Monocular depth estimation using diffusion models," 2023.

[42] J. Ho, C. Saharia, W. Chan, D. J. Fleet, M. Norouzi, and T. Salimans, "Cascaded diffusion models for high fidelity image generation," *Journal of Machine Learning Research*, vol. 23, no. 47, pp. 1–33, 2022. [Online]. Available: http://jmlr.org/papers/v23/21-0635.html

[43] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates,

Inc., 2021, pp. 8780–8794. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2021/file/49ad23d1ec9fa4bd8d77d02681df5cfa-Paper.pdf

[44] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," 2021.

[45] A. Atapour-Abarghouei and T. P. Breckon, "Real-time monocular depth estimation using synthetic data with domain adaptation via image style transfer," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2800–2810.

[46] G. Roth and A. Whitehead, "Estimating intrinsic camera parameters from the fundamental matrix using an evolutionary approach," *EURASIP Journal on Advances in Signal Processing*, vol. 2004, 07 2004.

[47] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6612–6619.

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

[49] Z. Yin and J. Shi, "Geonet: Unsupervised learning of dense depth, optical flow and camera pose," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1983–1992, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:3714620

[50] V. Casser, S. Pirk, R. Mahjourian, and A. Angelova, "Unsupervised monocular depth and ego-motion learning with structure and semantics," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 381–388, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:189762473

[51] H.-J. Jung, E. Park, and S. Yoo, "Fine-grained semantics-aware representation enhancement for self-supervised monocular depth estimation," *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 12 622–12 632, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:237213275

[52] A. Varma., H. Chawla., B. Zonooz., and E. Arani., "Transformers in self-supervised monocular depth estimation with unknown camera intrinsics," in *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2022) - Volume 4: VISAPP*, INSTICC. SciTePress, 2022, pp. 758–769.

[53] S. S. Chanduri, I. Vozniak, and Z. K. Suri, "Camlessmonodepth: Monocular depth estimation with unknown camera parameters," in *32nd British Machine Vision Conference 2021, BMVC 2021, Online, November 22-25, 2021*. BMVA Press, 2021, p. 92. [Online]. Available: https://www.bmvc2021-virtualconference.com/assets/papers/1381.pdf

[54] S. Shao, Z. Pei, W. Chen, D. Sun, P. C. Y. Chen, and Z. Li, "Monodiffusion: Self-supervised monocular depth estimation using diffusion model," 2023.

[55] Z. Liu, R. Li, S. Shao, X. Wu, and W. Chen, "Self-supervised monocular depth estimation with self-reference distillation and disparity offset refinement," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 33, no. 12, pp. 7565–7577, 2023.

[56] M. Maaz, A. Shaker, H. Cholakkal, S. Khan, S. W. Zamir, R. M. Anwer, and F. S. Khan, "Edgenext: Efficiently amalgamated cnn-transformer architecture for mobile vision applications," 2022.

[57] S. Mehta and M. Rastegari, "Separable self-attention for mobile vision transformers," 2022.

[58] A. Shaker, M. Maaz, H. Rasheed, S. Khan, M.-H. Yang, and F. S. Khan, "Swiftformer: Efficient additive attention for transformer-based real-time mobile vision applications," in *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 17 379–17 390.

[59] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7132–7141.

[60] Y. Tang, K. Han, J. Guo, C. Xu, Y. Li, C. Xu, and Y. Wang, "An image patch is a wave: Phase-aware vision mlp," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA,

USA: IEEE Computer Society, jun 2022, pp. 10 925–10 934. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR52688.2022.01066

[61] T. Yu, X. Li, Y. Cai, M. Sun, and P. Li, "S2-mlp: Spatial-shift mlp architecture for vision," in *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Los Alamitos, CA, USA: IEEE Computer Society, jan 2022, pp. 3615–3624. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/WACV51458.2022.00367

[62] V. Belagiannis, "Machine learning in signal processing: 6. neural networks," Nov 2022.

[63] A. Vaswani, N. M. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Neural Information Processing Systems*, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:13756489

[64] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *Image Processing, IEEE Transactions on*, vol. 13, pp. 600 – 612, 05 2004.

[65] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: the kitti dataset," *The International Journal of Robotics Research*, vol. 32, pp. 1231–1237, 09 2013.

[66] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[67] A. Saxena, M. Sun, and A. Ng, "Make3d: Learning 3d scene structure from a single still image," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, pp. 824–840, 2009. [Online]. Available: https://api.semanticscholar.org/CorpusID:253064894

[68] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. [Online]. Available: https://api.semanticscholar.org/CorpusID:57246310

[69] I. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit, M. Lucic, and A. Dosovitskiy, "Mlp-mixer: An all-mlp architecture for vision," 2021.

[70] G. Larsson, M. Maire, and G. Shakhnarovich, "Fractalnet: Ultra-deep neural networks without residuals," *ArXiv*, vol. abs/1605.07648, 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:3067546

[71] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=Bkg6RiCqY7

[72] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf

[73] L. Zhu, "Lyken17/pytorch-opcounter: Count the macs / flops of your pytorch model." [Online]. Available: https://github.com/Lyken17/pytorch-OpCounter/

Curriculum Vitae

Mehmet Ömer Eyi

+49-175-455-0-217 eyiomer@yahoo.com.tr linkedin.com/MehmetOmerEyi

Education

Friedrich Alexander University Oct 2021 – Present
Master of Science in Communications and Multimedia Engineering (GPA: 2.10 / 1.00) *Erlangen, Germany*
- **Relevant Coursework:** Information Theory and Coding, Statistical Signal Processing, Image and Video Compression, Pattern Recognition, Deep Learning, Computer Vision, Machine Learning in Signal Processing, Radar, RFID and Wireless Sensor Systems, Radar Signal Processing

Bilkent University Sept. 2017 – June 2021
Bachelor of Electrical and Electronics Engineering (GPA: 3.32 / 4.00) *Ankara, Turkey*
- **Relevant Coursework:** Algorithms and Programming I&II (Java), Microprocessors, Signals and Systems, Digital Signal Processing, Feedback Control Systems, Neural Networks, Medical Imaging, Telecommunications I&II, Robust Feedback Theory, Fundamental Structures of Computer Science I (C++)

Experience

Friedrich Alexander University Apr. 2023 – Feb 2024
Machine Learning in Signal Processing Lab Teaching Assistant (Python, PyTorch) *Erlangen, Germany*
- Teaching, interviewing and grading students in the laboratory course
- Prepared Auto-encoders lab notebooks.
- Prepared semi-supervised & supervised GYM car racing notebooks utilizing RNN, CNN and Mix-Match.

Fraunhofer IIS Apr 2023 – Aug 2023
Video Coding Intern (Python, PyTorch) *Erlangen, Germany*
- Worked on talking head generation in ultra-low bitrate video conferencing.
- Used GANs and monocular depth estimation networks to enhance the quality of the reconstructed video.

Friedrich Alexander University Dec. 2022 – Feb 2023
Statistical Signal Processing Lab Teaching Assistant (Python) *Erlangen, Germany*
- Teaching, interviewing and grading students in the laboratory sessions of statistical Signal Processing course

Aselsan Defence Inc Aug. 2020 – Sept. 2020
Digital Communications Intern (Python, Matlab) *Ankara, Turkey*
- Worked on the implementation of digital modulation simulations on Matlab and Python

Projects

Image and Video Processing on Embedded Platforms Lab Course | *Python* Oct. 2022 – Jan. 2023
- Implemented image and video processing tasks on Raspberry Pi (images & video coding and panorama).

Machine Learning in Signal Processing Lab Course | *Python, PyTorch* Apr. 2022 – Jul. 2021
- Implemented U-Net-based lane detection model using the dataset generated by CARLA Simulator.

Bachelor's Senior Design Project | *Python, PyTorch, TensorFlow* Sept. 2020 – Jun. 2021
- Worked on a lightweight encoder-decoder-based model for audio fault detection and integrating the Deepsort model into the framework for tracking people.
- Funded by The Scientific and Technological Research Council of Turkey.

Neural Network Term Project | *Python, PyTorch* Sept. 2020 – Jan. 2021
- Implemented an image captioning model using AlexNet /Inceptionv3 and LSTM/GRU in encoder and decoder respectively.

Technical Skills

Languages: Python, Matlab, Java, C++
Technologies: TensorFlow, PyTorch, Simulink
Interests: Radar Signal Processing, Machine Learning, Video Codec, Image & Video Processing, Computer Vision

Language Skills

Languages: Turkish (Native), English (B2), German (A2), French (A2)