PPEA-Depth: Progressive Parameter-Efficient Adaptation for Self-Supervised Monocular Depth Estimation

Yue-Jiang Dong[1], Yuan-Chen Guo[1], Ying-Tian Liu[1], Fang-Lue Zhang[2], Song-Hai Zhang[1]*

[1]BNRist, Department of Computer Science and Technology, Tsinghua University, China
[2]Victoria University of Wellington, New Zealand
[1]{dongyj21@mails., guoyc19@mails., liuyingt23@mails., shz@}tsinghua.edu.cn [2]fanglue.zhang@vuw.ac.nz

Abstract

Self-supervised monocular depth estimation is of significant importance with applications spanning across autonomous driving and robotics. However, the reliance on self-supervision introduces a strong static-scene assumption, thereby posing challenges in achieving optimal performance in dynamic scenes, which are prevalent in most real-world situations. To address these issues, we propose PPEA-Depth, a Progressive Parameter-Efficient Adaptation approach to transfer a pre-trained image model for self-supervised depth estimation. The training comprises two sequential stages: an initial phase trained on a dataset primarily composed of static scenes, succeeded by an expansion to more intricate datasets involving dynamic scenes. To facilitate this process, we design compact encoder and decoder adapters to enable parameter-efficient tuning, allowing the network to adapt effectively. They not only uphold generalized patterns from pre-trained image models but also retain knowledge gained from the preceding phase into the subsequent one. Extensive experiments demonstrate that PPEA-Depth achieves state-of-the-art performance on KITTI, CityScapes and DDAD datasets.

1 Introduction

In the realm of computer vision, accurate depth perception of a scene is a fundamental aspect that underpins a wide array of applications, from autonomous vehicles navigating complex environments to immersive virtual reality experiences. Depth estimation has witnessed remarkable advancements in recent years due to the proliferation of deep learning techniques. Especially, self-supervised methods (Zhou et al. 2017; Godard et al. 2019; Watson et al. 2021; Guizilini et al. 2022; Bangunharcana et al. 2023) that leverage the inherently contained rich and diverse sources of depth-related information in monocular videos open the door to scalability and real-world adaptability for depth estimation methods.

Existing self-supervised monocular depth estimation methods are commonly based on fine-tuning pre-trained image models learned from large image datasets, such as ImageNet (Deng et al. 2009), on depth datasets, such as KITTI (Geiger, Lenz, and Urtasun 2012), improving depth estimation accuracy compared to training from scratch (Godard et al. 2019). Some recent approaches introduce cost volume construction within the network to incorporate multi-frame inputs during inference (Watson et al. 2021; Guizilini et al. 2022; Bangunharcana et al. 2023). However, the above methods are all rooted in the assumption of a static scene, where solely the camera moves. This strong assumption subsequently creates challenges for these methods to reach optimality in dynamic scenes, limiting the efficient utilization of actual, unlabeled data for self-supervised training. Some other methods propose sophisticated algorithms that incorporate supplementary components like semantic segmentation or motion prediction networks to model object motion (Klingner et al. 2020; Lee et al. 2021a; Feng et al. 2022; Hui

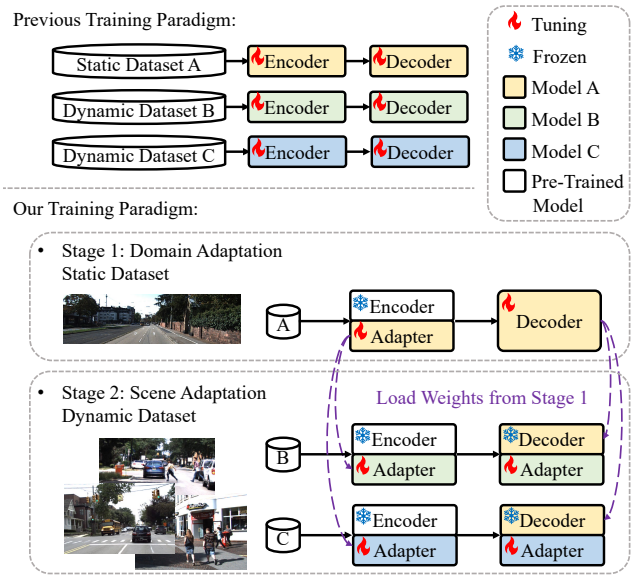


Figure 1: **Previous Paradigm v.s. Our Paradigm**. The conventional training approach employs a consistent process for both static and dynamic datasets: it includes using a pre-trained image model as an encoder and fine-tuning all U-Net parameters for each dataset. In contrast, our novel two-stage training paradigm integrates adapters to progressively tailor the pre-trained image models for depth perception initially on simple datasets (static scenes primarily) and then extends to intricate datasets (with dynamic scenes).

*Corresponding author.

2022). However, these methods often yield less satisfactory outcomes in static scenes compared to dedicated static methods (Guizilini et al. 2022; Bangunharcana et al. 2023).

In this paper, we aim to provide a new learning paradigm for self-supervised depth estimation to improve the performance and generalizability of the model, specifically by equipping it with improved capabilities to handle dynamic scenes. During our preliminary experiments of directly fine-tuning pre-trained models using such videos, we observed that excessive training with a relatively small dataset can disrupt the generalized patterns learned during pre-training, potentially resulting in catastrophic forgetting. Inspired by the recent success of parameter-efficient fine-tuning (PEFT) in natural language processing (NLP) community (Houlsby et al. 2019; Pfeiffer et al. 2020; Hu et al. 2021) and image and video classification (Jia et al. 2022; Chen et al. 2022b; Lin et al. 2022; Chen et al. 2022a; Yang et al. 2023), we extend it to self-supervised depth estimation, a loose-contrained regression task that remains unexplored in this field. We not only investigate adapters on encoders (Houlsby et al. 2019) to improve the adaptation from pre-trained models to the depth perception task, but also propose a novel decoder adapter to boost network robustness towards dynamic scenes.

We propose an innovative two-stage self-supervised depth estimation approach, PPEA-Depth, based on parameter-efficient adaptation. We devise lightweight encoder adapters and decoder adapters within our framework. Our method is guided by insights drawn from human learning mechanisms, which typically progress from simple to complex tasks. Our approach first trains on datasets primarily featuring static scenes, which adhere to the static scene assumptions. The pre-trained encoder is frozen to retain general patterns gained from large image datasets and encoder adapters are tuned to adapt the network to learn depth priors. Subsequently, we train the network on more intricate and dynamic scenes. To retain the knowledge learned from static scenes in the preceding stage, we load weights from the previous stage, freeze both the encoder and decoder, and just train extra encoder and decoder adapters to summarize network updates for adaptation to new scenes.

The second scene adaptation stage extends beyond a single dataset. The domain-adapted model can be flexibly adapted to various new scenes solely by tuning the adapters. We only need to tune and store a small number of scene-specific parameters to generalize across different datasets. With these innovations, PPEA-Depth achieves state-of-the-art performance on KITTI, CityScapes(Cordts et al. 2016) and DDAD(Guizilini et al. 2020a) datasets.

Our main contributions can be summarized as:

- We propose a new paradigm to transfer upstream pre-trained models to self-supervised monocular depth estimation in a progressive manner from static scenes to more challenging dynamic scenes.
- We design encoder adapters to take advantage of pre-trained image models. Reducing tunable parameters by up to 90%, tuning encoder adapters demonstrates less depth estimation errors than full fine-tuning.
- We design a decoder adapter to enhance the adaptability of the decoder to more challenging datasets. Remarkably, merely tuning encoder adapters and the decoder adapter yields a 6% improvement in absolute relative errors compared to fine-tuning all U-Net parameters on the full training set when utilizing only 3% of the training data.

2 Related Work

2.1 Self-Supervised Monocular Depth Estimation

Self-supervised monocular depth estimation predicts depth and camera ego-motion from an outdoor monocular video, and is supervised by image reprojection loss (Zhou et al. 2017). On the basis of such methodology, previous works make progress in designing loss functions for better convergence to optimum (Godard et al. 2019; Shu et al. 2020), designing more complicated encoder structure with cost volume (Watson et al. 2021; Bangunharcana et al. 2023) and attention scheme (Guizilini et al. 2022), and leveraging cross-domain information of optical flow (Yin and Shi 2018; Chen, Schmid, and Sminchisescu 2019; Ranjan et al. 2019) or scene semantics (Casser et al. 2019; Klingner et al. 2020; Jung, Park, and Yoo 2021; Lee et al. 2021a,b; Feng et al. 2022) to handle dynamic objects, etc. Training self-supervised depth estimation from a pre-trained model yields superior performance compared to training from scratch (Godard et al. 2019), implying the generalized patterns learned in pre-training benefit this task.

2.2 Parameter-Efficient Fine-Tuning

A variety of parameter-efficient fine-tuning (PEFT) methods have been proposed in recent NLP works (Houlsby et al. 2019; Pfeiffer et al. 2020; Karimi Mahabadi et al. 2021; Zaken, Ravfogel, and Goldberg 2021; Zhu et al. 2021; Li and Liang 2021; Hu et al. 2021; He et al. 2021). Different from the traditional training patterns which fine-tune large pre-trained models on different downstream tasks, PEFT freezes parameters in pre-trained models and only fine-tunes a small number of extra parameters to obtain strong performance with less tuned parameters.

In the computer vision field, PEFT has been mainly studied and applied in classification tasks including image classification (Jia et al. 2022; Bahng et al. 2022; Chen et al. 2022a; Jie and Deng 2023) and video action recognition (Lin et al. 2022; Chen et al. 2022a; Yang et al. 2023). A recent work (Chen et al. 2022b) investigates PEFT for the application of Vision Transformer (Dosovitskiy et al. 2020) in dense prediction tasks including semantic segmentation and object detection. Different from previous work, we study PEFT in self-supervised monocular depth estimation, a challenging dense regression task. The PEFT algorithm is not only designed and applied for parameters of the encoder backbone but also for the dense prediction decoder in our method. To the best of our knowledge, we are the first to study PEFT in the depth estimation area.

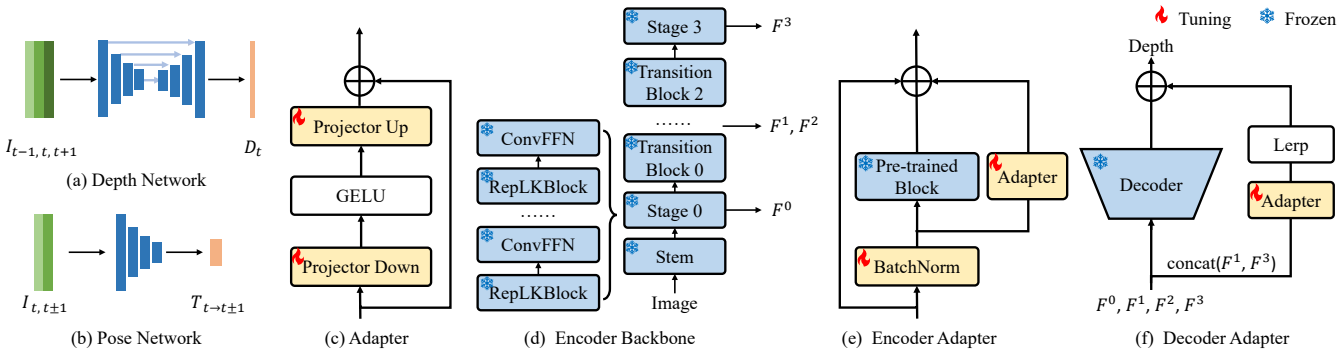


Figure 2: (a) Depth network is a U-Net structure predicting depth taking three consecutive frames. (b) Pose network regresses the camera relative pose given two images. (c) Adapter is a bottleneck structure with a skip connection. (d) Structure of RepLKNet (Ding et al. 2022) backbone. (e) Our encoder adapter design. We attach encoder adapters to pre-trained RepLKBlock and ConvFFN. (f) Our decoder adapter design. Lerp represents linear interpolation.

3 Method

3.1 Overview

Our network comprises a depth network (Fig. 2(a)) and a pose network (Fig. 2(b)). The depth network employs a U-Net structure, encompassing an encoder to extract image features and a decoder to predict dense depth maps. Meanwhile, the pose network predicts the camera transformation between two frames. It has a feature extractor followed by a prediction head, which outputs a six-dimensional vector – three for rotation angles and the other three for translation.

Our network takes three consecutive frames from a monocular video as input. The middle frame is reconstructed with its adjacent frames, and the difference between the reconstructed and original images serves as the supervision signal. This reconstruction relies on cross-frame pixel correspondences in the structure-from-motion theory (Zhou et al. 2017). Given camera intrinsics K and camera relative pose T between two frames a, b from a video, the pixel correspondence between them can be computed by:

$$p_a \sim KT_{b\to a}D_b(p_b)K^{-1}p_b, \tag{1}$$

where p_a and p_b are corresponding pixels in the two frames, and D_b is the depth of p_b. Assuming a static scene, these correspondences reconstruct frame b from pixels in frame a.

We adopt the identical architecture and training strategy for the pose network as ManyDepth (Watson et al. 2021), with our primary focus centered on the depth network. In the conventional training approach, encoders are initialized with transferred pre-trained weights, while decoders and prediction heads are trained from scratch. In our method, we introduce the PEFT scheme by adding adapters (Houlsby et al. 2019) to the depth network's encoder and decoder to encapsulate network adaptations across distinct domains.

3.2 Adapter

Adapters are compact architectures designed to tailor a pre-trained module for a specific downstream task (Houlsby et al. 2019). While parameters of the pre-trained model remain static, only adapter parameters are modified during training. Illustrated in Fig. 2(c), adapters follow a bottleneck structure, encompassing two linear projection layers, an activation layer, and a skip connection. The initial projection layer reduces the input feature dimension, and the subsequent one restores it to the original input dimension after the activation layer. Based on such architecture, we respectively design adapters for the depth network encoder and decoder.

3.3 Encoder Adapter

Backbone We opt for RepLKNet (Ding et al. 2022), a CNN architecture featuring a notable kernel size of 31×31, as the encoder backbone. This selection is attributed to its adaptability concerning input image resolution, comparable accuracy to Swin Transformer (Liu et al. 2021), and enhanced inference speed when applied to downstream tasks.

As illustrated in Fig. 2(d), RepLKNet generates feature maps of four different scales: F^1, F^2, F^3, F^4 at four stages. Within each stage, a RepLKBlock and a ConvFFN are interleaved in their arrangement. Please refer to the supplementary materials for more details. To leverage the pre-established generalized patterns of the RepLKNet and fine-tune them for the depth regression task, we integrate adapters to the RepLKBlocks and ConvFFNs.

Architecture As depicted in Fig. 2(e), the input feature is initially processed through a batch normalization layer and then fed into the pre-trained block and adapter using two parallel streams. Within each adapter, we adhere to the conventional bottleneck structure, with a notable distinction being the replacement of the first linear projection module with a convolutional layer for the adapters of RepLKBlock. This convolutional layer employs a kernel size of 3, a stride of 1, and padding of 1. These settings maintain the spatial dimensions of the input feature maps unchanged after convolution. Substituting the linear projection with a 3×3 convolution offers adapters a larger receptive field, proving advantageous for per-pixel regression tasks such as depth estimation. Meanwhile, the ConvFFN's adapter continues to adopt linear projection in order to minimize the parameter count. Given the input x, the output x' of the module after

incorporating the adapted block $\mathcal{A}$ can be written as:

$$x' = x + \mathcal{M}(\mathcal{N}(x)) + \mathcal{A}(\mathcal{N}(x)), \tag{2}$$

where $\mathcal{M}$ is the pre-trained block, $\mathcal{A}$ is its adapter, and $\mathcal{N}$ represents batch normalization.

3.4 Decoder Adapter

The architecture of the decoder adapter is illustrated in Fig. 2(f). To strike a balance between the additional parameter count and the sufficiency of adapter input, we perform an interpolation of F^3 to match the spatial dimensions of F^0, and subsequently use their concatenation as the input to the adapter. The inner structure of the decoder adapter adopts a *Projector-GELU-Projector* configuration, where both projectors employ linear projection. Given that F^0 has dimensions of $(H/4, W/4)$ spatially, the output of the decoder adapter requires restoration to the original size of the input images, i.e., (H, W). This is achieved through linear interpolation. The computation for incorporating the decoder adapter can be written as:

$$x' = \mathcal{D}(F^0, F^1, F^2, F^3) + \mathcal{A}(F^0, F^3), \tag{3}$$

where $\mathcal{D}$ is the decoder and $\mathcal{A}$ is its adapter.

3.5 Progressive Adaptation

As illustrated in Fig. 1, our progressive adaptation involves two stages. Stage 1 is trained on a dataset that primarily follows the static-scene assumption. In Stage 1, a pre-trained model, capable of efficiently extracting color features, is tailored from an image classification task to depth regression. We freeze all encoder parameters and only train encoder adapters and U-Net decoder. The frozen encoder retains the generalized patterns acquired during the model pre-training.

Stage 2 is conducted on datasets that predominantly feature dynamic scenes, which are more challenging for training because dynamic objects violate Eqn. (1) and mislead the self-supervision signal. Our method capitalizes on the depth priors obtained from static scenes in Stage 1 and applies them to dynamic scenes. In Stage 2, we load the weights of the U-Net encoder, the encoder adapters, and the U-Net decoder from Stage 1, and freeze both the encoder and decoder, with only adapter parameters being updated. This paradigm preserves the depth perception ability obtained from Stage 1, as most network parameters are frozen and are unaffected by the erroneous loss caused by object motion. Meanwhile, the lightweight adapters make minor adjustments based on this robust depth prior, fitting data distribution in new scenes.

4 Experiments

In this section, we (1) demonstrate the effectiveness of the two stages separately, (2) showcase that PPEA-Depth yields state-of-the-art results on standard benchmarks, and (3) assess the generalizability of the proposed methods. Supplementary materials contain additional results and ablation studies to validate our approach.

4.1 Datasets

Our method comprises two stages. The domain adaptation stage is trained and evaluated on the KITTI dataset, as it contains a substantial number of scenes that adhere to the static-scene assumption. Subsequently, the scene adaptation stage is built upon the parameters acquired from the domain adaptation stage on KITTI. This stage is then evaluated on more challenging datasets, including CityScapes and DDAD.

KITTI The KITTI dataset (Geiger, Lenz, and Urtasun 2012) serves as the standard benchmark for evaluating self-supervised monocular depth estimation methods. We adhere to the established training protocols (Eigen, Puhrsch, and Fergus 2014) and utilize the data pre-processing approach introduced by (Zhou et al. 2017), yielding 39,810 monocular triplets for training, 4,424 for validation, and 697 for testing.

CityScapes The CityScapes dataset (Cordts et al. 2016) is notably more challenging due to its inclusion of numerous dynamic scenes with multiple moving objects (Casser et al. 2019). While fewer results are reported compared to KITTI, CityScapes serves as a prominent benchmark for those studies that focus on developing algorithms to handle dynamic objects (Lee et al. 2021a; Li et al. 2021; Feng et al. 2022; Hui 2022). Following the setup of previous work (Feng et al. 2022), we train on 58,355 and evaluate 1,525 images.

DDAD DDAD is a more recent autonomous driving dataset (Guizilini et al. 2020a). It is challenging owing to its extended depth range of up to 200 meters and inclusion of moving objects (Guizilini et al. 2022). We follow the official DGP codebase of DDAD dataset to load images, and use 12,350 monocular triplets for train and 3,850 for evaluation.

4.2 Evaluation Details

As self-supervised learning predicts relative depth, we adhere to the established practice of scaling it before conducting evaluations (Godard et al. 2019). We use standard depth assessment metrics (Eigen and Fergus 2015), encompassing absolute and squared relative errors (AbsRel and SqRel), root mean squared error (RMSE), root mean squared log error (RMSElog), and accuracy within a threshold (δ).

PPEA-Depth adopts the well-established multi-frame inference and teacher-student distillation training scheme (Watson et al. 2021; Feng et al. 2022; Guizilini et al. 2022; Bangunharcana et al. 2023). The main network contains a cost volume construction process, using both the current frame I_t and its preceding frame I_{t-1} to predict depth D_t. The teacher network does not involve cost volume generation and only takes the current frame during inference. Albeit for the difference in inner structure, the teacher and student share the same adapter design and training paradigm. Please refer to the supplementary for more details.

We carry out experiments using two variations of RepLKNet, each with different scales of parameter counts: RepLKNet-B and RepLKNet-L (Ding et al. 2022). In line with the approach taken by Houlsby et al. (2019); He et al. (2021); Yang et al. (2023), all adapter weights are initialized to zero to ensure stable training.

Pre-Trained Backbone	Tuning Strategy	Tuning Params (M)		*Errors*↓				*Accuracy*↑		
		Encoder	Decoder	AbsRel	SqRel	RMSE	$\text{RMSE}_{\log}$	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
RepLKNet-B	Frozen	0	12.5	0.128	0.938	4.908	0.200	0.850	0.953	0.981
	Full Fine-Tuned	78.8	12.5	0.092	0.774	4.355	0.175	0.911	0.966	0.982
	Adapter (0.0625)	8.15	12.5	**0.092**	**0.686**	**4.207**	**0.170**	0.910	**0.968**	**0.984**
	Adapter (0.25)	21.2	12.5	**0.090**	**0.666**	**4.175**	**0.168**	**0.912**	**0.969**	**0.984**
RepLKNet-L	Frozen	0	28.2	0.129	0.938	4.937	0.201	0.846	0.952	0.980
	Full Fine-Tuned	171	28.2	0.089	0.734	4.306	0.169	0.917	0.968	0.983
	Adapter (0.0625)	18.1	28.2	0.090	**0.666**	**4.146**	**0.168**	0.915	**0.969**	**0.985**
	Adapter (0.25)	47.6	28.2	**0.088**	**0.649**	**4.105**	**0.167**	**0.917**	**0.968**	**0.984**

Table 1: **Encoder Adapters are Effective for Domain Adaptation.** Our method achieves better depth estimation accuracy with fewer tuned parameters. Numbers in brackets after *Adapter* indicate the bottleneck ratio.

4.3 Stage 1: Domain Adaptation

Here we demonstrate the effectiveness of our adapter-based tuning strategy in the domain adaptation stage. We compare our method with two baselines on the KITTI dataset, all using the same pre-trained RepLKNet as the depth encoder. The first baseline (Frozen) freezes the depth encoder and only trains the depth decoder. The second baseline (Full Fine-Tuned) tunes all the parameters of the depth encoder and decoder. The goal of our domain adaptation stage is to add a few tunable parameters to the first baseline and close the gap between it and the full fine-tuning method.

Encoder adapters project input features to a lower dimensional space and then project them back. Bottleneck ratio is the ratio between the input and the intermediate feature channels and directly influences the number of adapter parameters. We control the number of adapter parameters by setting different bottleneck ratios. As shown in Table 1, for RepLKNet-B, the frozen encoder with our adapters achieves comparable performance with fine-tuning the entire backbone, while using 90% fewer parameters than it. Tuning 21.2M encoder adapter parameters surpasses the performance of full fine-tuned RepLKNet-B. Experimental results for RepLKNet-L are similar. Our method reduces the number of tunable parameters by up to 90% to achieve comparable performance with the full fine-tuned RepLKNet-L.

Our domain adaptation stage can preserve and utilize the generalized patterns in the ImageNet-pre-trained model by freezing the encoder backbone. It's worth noting that adapter tuning leads to significantly lower SqRel and RMSE values. This observation suggests that the generalized patterns retained through our adapter tuning strategy are beneficial for reducing extreme depth estimation errors.

4.4 Stage 2: Scene Adaptation

We verify the efficacy of our second training stage: the scene adaptation stage and its decoder adapter. We consistently select RepLKNet-B as the depth encoder and set the adapter bottleneck ratio to 0.25. We present a comparison on CityScapes under varying training strategies in Table 2.

Our proposed paradigm trains based on the weights learned in the first training stage on KITTI. Quantitative results in Table 2 and qualitative results in Fig. 3(b-d) show that tuning on KITTI before training on CityScapes significantly outperforms the method of training from scratch. There are a large number of dynamic objects in CityScapes, making training from scratch on it challenging because moving objects disrupt the pixel correspondences computed by Eqn. (1). This may mislead the loss, resulting in updates to the network's parameters in incorrect directions.

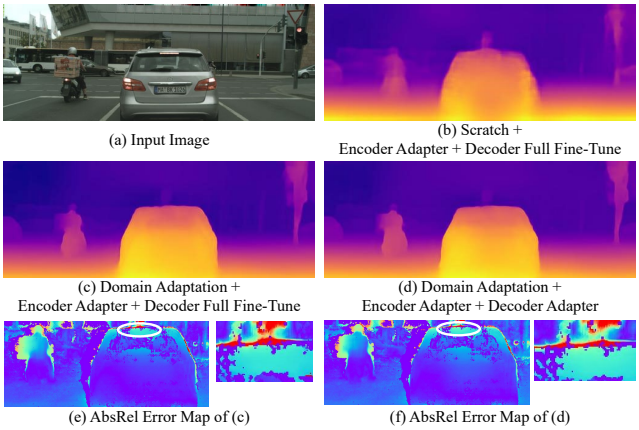


Figure 3: **Comparisons of Different Training Strategies on CityScapes**. Training from domain adaptation yields better depth estimates on vehicles and cyclists compared to training from scratch. Tuning the decoder adapter demonstrates improved depth estimates in the upper portion of the car compared to the full fine-tuned decoder.

We directly load the model trained from Stage 1 on KITTI and evaluate it on CityScapes (the third row of Table 2). Its outcomes are unsatisfactory, signifying that there exist notable distinctions between the two datasets. We also assess another baseline: starting from weights learned on KITTI and subsequently full fine-tuning both the encoder and decoder on CityScapes (the fourth row of Table 2). It outperforms methods that train from scratch but evidently falls short in comparison to the adapter-tuning approach that also starts from the weights learned in the domain adaptation stage. This observation suggests that leveraging knowledge from earlier training phases benefits dynamic object handling, but directly fine-tuning all parameters disrupts previously acquired patterns.

Train From	Tuning Strategy		*Errors*↓				*Accuracy*↑		
	Encoder	Decoder	AbsRel	SqRel	RMSE	$\text{RMSE}_{\log}$	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Scratch	Full Fine-Tuned	Full Fine-Tuned	0.130	1.717	6.448	0.184	0.857	0.958	0.984
	Adapter	Full FT.	0.130	1.473	6.735	0.179	0.859	0.964	0.988
Domain Adaptation	Original Weights	Original Weights	0.138	1.319	7.211	0.198	0.819	0.957	0.987
	Full FT.	Full FT.	0.116	1.120	6.193	0.168	0.873	0.969	0.991
	Adapter	Full FT.	**0.103**	**0.962**	**5.716**	**0.155**	**0.897**	**0.976**	**0.992**
	Adapter	Adapter	**0.100**	**0.976**	**5.673**	**0.152**	**0.904**	**0.977**	**0.992**

Table 2: **Effectiveness of Our Scene Adaptation Strategy and Decoder Adapter**. We compare different training strategies on CityScapes (Cordts et al. 2016). Our strategy tunes based on the domain adaptation stage, yielding better estimated depth than training from scratch. The last two rows indicate tuning decoder adapter performs better than tuning the whole U-Net decoder.

Percentage of Training Data	Tuning Strategy		*Errors*↓				*Accuracy*↑		
	Encoder	Decoder	AbsRel	SqRel	RMSE	$\text{RMSE}_{\log}$	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
3%	Adapter	Adapter	0.109	1.177	6.180	0.162	0.889	0.973	0.990
5%	Adapter	Adapter	0.107	1.137	6.081	0.160	0.892	0.973	0.991
10%	Adapter	Adapter	0.105	1.134	5.997	0.158	0.896	0.974	0.991
25%	Adapter	Adapter	0.102	1.104	5.844	0.154	0.902	0.975	0.991

Table 3: **Our Scene Adaptation and Adapters Enhance Data Efficiency**. Our scene adaptation strategy outperforms training from scratch and full fine-tuning from domain adaptation by a large margin even when utilizing only 3% of the training data from CityScapes (Cordts et al. 2016).

As evident from the last two rows in Table 2, fine-tuning the decoder adapter with only 0.185M parameters yields superior outcomes compared to fine-tuning the entire U-Net decoder. As illustrated in Fig. 3 (e-f), tuning decoder adapter produces more precise depth estimations compared to the full fine-tuned decoder. Freezing both the U-Net encoder and decoder better preserves the depth priors acquired in previous stages and enhances the robustness towards errors in training losses caused by object motion. Our method, which merely tunes encoder adapters and the decoder adapter, offers a solution that strikes a balance between adapting the network to new datasets and conserving valuable depth perception patterns gained from preceding training phases.

Our method also enhances data efficiency. We conducted Stage 2 training using randomly sampled subsets of 2.5%, 10%, and 25% of the data from the CityScapes training set. The evaluation results for each subset are presented in Table 3. Notably, tuning encoder adapters and decoder adapters with just around 3% of the training data significantly surpasses the strategies of training from scratch and full fine-tuning the entire U-Net in Stage 2. Training with approximately 25% of the data yields results comparable to tuning adapters on the entire training set, particularly for the accuracy metric $\delta < 1.25$. This suggests that the adapter-based learning scheme can swiftly enhance the accuracy of depth estimation even with a limited amount of data.

4.5 Depth Evaluation

Table 4 and Table 5 provide comprehensive comparisons between our method and state-of-the-art models on the two widely recognized benchmarks for self-supervised depth estimation: KITTI (Geiger, Lenz, and Urtasun 2012) and CityScapes (Cordts et al. 2016). As specified in section 4.2, PPEA-Depth incorporates both a teacher and a student network, following ManyDepth (Watson et al. 2021). The teacher network utilizes a single frame during inference, whereas the student network employs two frames (preceding and current). In Table 4 and 5, we present the results of the teacher and student networks by indicating the number of frames in the column of *Test Frames* as 1 or 2.

Our model outperforms most previous models on both benchmarks. Specifically, our method demonstrates a notable enhancement in the accuracy metric $\delta < 1.25$, signifying a high degree of accurate inliers. In contrast to prior state-of-the-art models that mainly focus on improving the self-supervised monocular depth estimation methodology itself – such as designing intricate sub-modules for iterative refinement of estimated depth and pose (Bangunharcana et al. 2023), enhancing cost volume generation with transformers (Guizilini et al. 2022), or incorporating a motion field prediction network (Hui 2022) – our approach centers on introducing a more effective strategy for the training process. We aim to contribute by designing structures that leverage the generalized patterns in robust pre-trained models and creating more sensible learning paradigms to address challenging scenarios in self-supervised depth estimation.

4.6 Generalization Ability

To substantiate the generalization capability of our method, we extend our evaluation beyond the two standard benchmarks, and assess the performance of PPEA-Depth on a more recent dataset, DDAD (Guizilini et al. 2020a). As detailed in Section 4.2, the student network involves a cost volume construction process, which explicitly incorporates the

Method	Test Frames	S	W×H	*Errors*↓				*Accuracy*↑		
				AbsRel	SqRel	RMSE	$\text{RMSE}_{\log}$	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Casser et al. (2019)	1	●	416×128	0.141	1.026	5.291	0.215	0.816	0.945	0.979
Bian et al. (2019)	1		416×128	0.137	1.089	5.439	0.217	0.830	0.942	0.975
Gordon et al. (2019)	1	●	416×128	0.128	0.959	5.230	0.212	0.845	0.947	0.976
Godard et al. (2019)	1		640×192	0.115	0.903	4.863	0.193	0.877	0.959	0.981
Lee et al. (2021a)	1	●	832×256	0.112	0.777	4.772	0.191	0.872	0.959	0.982
Guizilini et al. (2020a)	1		640×192	0.111	0.785	4.601	0.189	0.878	0.960	0.982
Hui (2022)	1		640×192	0.108	0.710	4.513	0.183	0.884	0.964	0.983
Wang et al. (2021)	1		640×192	0.109	0.779	4.641	0.186	0.883	0.962	0.982
Johnston et al. (2020)	1		640×192	0.106	0.861	4.699	0.185	0.889	0.962	0.982
Guizilini et al. (2020b)	1	●	640×192	0.102	0.698	4.381	0.178	0.896	0.964	0.984
Wang et al. (2020)	2(-1,0)		640×192	0.106	0.799	4.662	0.187	0.889	0.961	0.982
Watson et al. (2021)	2(-1,0)		640×192	0.098	0.770	4.459	0.176	0.900	0.965	0.983
Feng et al. (2022)	2(-1,0)	●	640×192	0.096	0.720	4.458	0.175	0.897	0.964	0.984
Guizilini et al. (2022)	2(-1,0)		640×192	0.090	0.661	4.149	0.175	0.905	0.967	0.984
Bangunharcana et al. (2023)	2(-1,0)		640×192	**0.087**	0.698	4.234	0.170	0.914	0.967	0.983
PPEA-Depth (RepLKNet-B)	2(-1,0)		640×192	0.090	0.666	4.175	**0.168**	0.912	**0.969**	**0.984**
PPEA-Depth (RepLKNet-L)	2(-1,0)		640×192	0.088	**0.649**	**4.105**	**0.167**	**0.917**	**0.968**	**0.984**

Table 4: **Depth Estimation Results on KITTI** Eigen Split (Eigen and Fergus 2015). S denotes a need for semantic information.

Method	Test Frames	S	W×H	*Errors*↓				*Accuracy*↑		
				AbsRel	SqRel	RMSE	$\text{RMSE}_{\log}$	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Casser et al. (2019)	1	●	416×128	0.145	1.737	7.280	0.205	0.813	0.942	0.976
Godard et al. (2019)	1		416×128	0.129	1.569	6.876	0.187	0.849	0.957	0.983
Gordon et al. (2019)	1	●	416×128	0.127	1.330	6.960	0.195	0.830	0.947	0.981
Li et al. (2021)	1		416×128	0.119	1.290	6.980	0.190	0.846	0.952	0.982
Lee et al. (2021a)	1	●	832×256	0.111	1.158	6.437	0.182	0.868	0.961	0.983
Watson et al. (2021)	2(-1, 0)		416×128	0.114	1.193	6.223	0.170	0.875	0.967	0.989
Feng et al. (2022)	2(-1, 0)	●	416×128	0.103	1.000	5.867	0.157	0.895	0.974	0.991
Hui (2022)	1		416×128	0.100	**0.839**	5.774	0.154	0.895	0.976	**0.993**
PPEA-Depth (RepLKNet-B)	1		416×128	**0.099**	1.115	5.995	0.155	**0.905**	**0.976**	0.991
PPEA-Depth (RepLKNet-B)	2(-1, 0)		416×128	**0.100**	0.976	**5.673**	**0.152**	**0.904**	**0.977**	0.992

Table 5: **Depth Estimation Results on CityScapes** (Cordts et al. 2016). S denotes a need for semantic information.

depth range of the dataset. In the domain adaptation stage on KITTI, this range is set to 0-100m, which is not compatible with the DDAD dataset with a depth range of 0-200m. Therefore, we evaluate the teacher network on DDAD.

Table 6 compares our method with state-of-the-art models on DDAD. Using only a single frame during testing, our method outperforms previous models, showing the potent generalization capability of our adapters and the potential of transfer learning across varied datasets via adapter tuning with a core model. In this manner, we only need to maintain and incorporate a modest count of dataset-specific parameters, in addition to the core model, for each featured scene.

Method	TF	*Errors*↓		*Accuracy*↑
		AbsRel	SqRel	$\delta < 1.25$
Guizilini et al. (2020a)	1	0.162	3.917	0.823
Guizilini et al. (2022)	2	0.135	2.953	0.836
PPEA-Depth (B)	1	**0.134**	**2.809**	**0.836**
PPEA-Depth (L)	1	**0.130**	**2.695**	**0.846**

Table 6: **Depth Estimation Results on DDAD** (Guizilini et al. 2020a) (WxH = 640x384) . TF represents Test Frames.

5 Conclusion

In this paper, we introduce PPEA-Depth, a novel framework designed to enable the progressive transfer of pre-trained image models into the realm of self-supervised depth estimation. This transfer is orchestrated through the utilization of encoder and decoder adapters. Initially, the pre-trained model is tailored to accommodate depth perception using datasets primarily aligned with the static-scene assumption of self-supervised depth estimation methodology. Subsequently, it is further adapted to more challenging datasets involving a large number of moving objects.

PPEA-Depth achieves state-of-the-art results on the KITTI and CityScapes, while also demonstrating its robust transferability on the DDAD dataset. Our method is promising to transfer to other tasks with loose-constrained loss, boosting network robustness towards loss errors.

Acknowledgements

This work was supported by the National Key Research and Development Program of China (No. 2023YFF0905104), the Natural Science Foundation of China (No. 62132012), Beijing Municipal Science and Technology Project (No. Z221100007722001) and Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology. Fang-Lue Zhang was supported by the Marsden Fund Council managed by the Royal Society of New Zealand (No. MFP-20-VUW-180).

References

Bahng, H.; Jahanian, A.; Sankaranarayanan, S.; and Isola, P. 2022. Visual prompting: Modifying pixel space to adapt pre-trained models. *arXiv preprint arXiv:2203.17274*.

Bangunharcana, A.; Magd, A.; Kim, K.-S.; et al. 2023. DualRefine: Self-Supervised Depth and Pose Estimation Through Iterative Epipolar Sampling and Refinement Toward Equilibrium. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 726–738.

Bian, J.; Li, Z.; Wang, N.; Zhan, H.; Shen, C.; Cheng, M.-M.; and Reid, I. 2019. Unsupervised scale-consistent depth and ego-motion learning from monocular video. *Advances in neural information processing systems*, 32.

Casser, V.; Pirk, S.; Mahjourian, R.; and Angelova, A. 2019. Unsupervised monocular depth and ego-motion learning with structure and semantics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 0–0.

Chen, S.; Ge, C.; Tong, Z.; Wang, J.; Song, Y.; Wang, J.; and Luo, P. 2022a. Adaptformer: Adapting vision transformers for scalable visual recognition. *arXiv preprint arXiv:2205.13535*.

Chen, Y.; Schmid, C.; and Sminchisescu, C. 2019. Self-supervised learning with geometric constraints in monocular video: Connecting flow, depth, and camera. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 7063–7072.

Chen, Z.; Duan, Y.; Wang, W.; He, J.; Lu, T.; Dai, J.; and Qiao, Y. 2022b. Vision transformer adapter for dense predictions. *arXiv preprint arXiv:2205.08534*.

Cordts, M.; Omran, M.; Ramos, S.; Rehfeld, T.; Enzweiler, M.; Benenson, R.; Franke, U.; Roth, S.; and Schiele, B. 2016. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3213–3223.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255. Ieee.

Ding, X.; Zhang, X.; Zhou, Y.; Han, J.; Ding, G.; and Sun, J. 2022. Scaling Up Your Kernels to 31x31: Revisiting Large Kernel Design in CNNs. *arXiv preprint arXiv:2203.06717*.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Eigen, D.; and Fergus, R. 2015. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE international conference on computer vision*, 2650–2658.

Eigen, D.; Puhrsch, C.; and Fergus, R. 2014. Depth map prediction from a single image using a multi-scale deep network. *Advances in neural information processing systems*, 27.

Feng, Z.; Yang, L.; Jing, L.; Wang, H.; Tian, Y.; and Li, B. 2022. Disentangling Object Motion and Occlusion for Unsupervised Multi-frame Monocular Depth. *arXiv preprint arXiv:2203.15174*.

Geiger, A.; Lenz, P.; and Urtasun, R. 2012. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, 3354–3361. IEEE.

Godard, C.; Mac Aodha, O.; Firman, M.; and Brostow, G. J. 2019. Digging Into Self-Supervised Monocular Depth Estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Gordon, A.; Li, H.; Jonschkowski, R.; and Angelova, A. 2019. Depth from videos in the wild: Unsupervised monocular depth learning from unknown cameras. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 8977–8986.

Guizilini, V.; Ambrus, R.; Pillai, S.; Raventos, A.; and Gaidon, A. 2020a. 3d packing for self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2485–2494.

Guizilini, V.; Ambruș, R.; Chen, D.; Zakharov, S.; and Gaidon, A. 2022. Multi-Frame Self-Supervised Depth with Transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 160–170.

Guizilini, V.; Hou, R.; Li, J.; Ambrus, R.; and Gaidon, A. 2020b. Semantically-Guided Representation Learning for Self-Supervised Monocular Depth. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

He, J.; Zhou, C.; Ma, X.; Berg-Kirkpatrick, T.; and Neubig, G. 2021. Towards a unified view of parameter-efficient transfer learning. *arXiv preprint arXiv:2110.04366*.

Houlsby, N.; Giurgiu, A.; Jastrzebski, S.; Morrone, B.; De Laroussilhe, Q.; Gesmundo, A.; Attariyan, M.; and Gelly, S. 2019. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*, 2790–2799. PMLR.

Hu, E.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, L.; and Chen, W. 2021. LoRA: Low-Rank Adaptation of Large Language Models. arXiv:2106.09685.

Hui, T.-W. 2022. Rm-depth: Unsupervised learning of recurrent monocular depth in dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1675–1684.

Jia, M.; Tang, L.; Chen, B.-C.; Cardie, C.; Belongie, S.; Hariharan, B.; and Lim, S.-N. 2022. Visual prompt tuning. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIII*, 709–727. Springer.

Jie, S.; and Deng, Z.-H. 2023. FacT: Factor-Tuning For Lightweight Adaptation on Vision Transformer. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*.

Johnston, A.; Carneiro; Gustavo; et al. 2020. Self-supervised monocular trained depth estimation using self-attention and discrete disparity volume. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, 4756–4765.

Jung, H.; Park, E.; and Yoo, S. 2021. Fine-grained semantics-aware representation enhancement for self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 12642–12652.

Karimi Mahabadi, R.; Henderson, J.; Ruder, S.; et al. 2021. Compacter: Efficient low-rank hypercomplex adapter layers. *Advances in Neural Information Processing Systems*, 34: 1022–1035.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Klingner, M.; Termöhlen, J.-A.; Mikolajczyk, J.; and Fingscheidt, T. 2020. Self-supervised monocular depth estimation: Solving the dynamic object problem by semantic guidance. In *European Conference on Computer Vision*, 582–600. Springer.

Lee, S.; Im, S.; Lin, S.; and Kweon, I. S. 2021a. Learning Monocular Depth in Dynamic Scenes via Instance-Aware Projection Consistency. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Lee, S.; Rameau, F.; Pan, F.; and Kweon, I. S. 2021b. Attentive and contrastive learning for joint depth and motion field estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 4862–4871.

Li, H.; Gordon, A.; Zhao, H.; Casser, V.; and Angelova, A. 2021. Unsupervised monocular depth learning in dynamic scenes. In *Conference on Robot Learning*, 1908–1917. PMLR.

Li, X. L.; and Liang, P. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Lin, Z.; Geng, S.; Zhang, R.; Gao, P.; de Melo, G.; Wang, X.; Dai, J.; Qiao, Y.; and Li, H. 2022. Frozen clip models are efficient video learners. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXV*, 388–404. Springer.

Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; and Guo, B. 2021. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in pytorch.

Pfeiffer, J.; Kamath, A.; Rücklé, A.; Cho, K.; and Gurevych, I. 2020. AdapterFusion: Non-destructive task composition for transfer learning. *arXiv preprint arXiv:2005.00247*.

Ranjan, A.; Jampani, V.; Balles, L.; Kim, K.; Sun, D.; Wulff, J.; and Black, M. J. 2019. Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 12240–12249.

Shu, C.; Yu, K.; Duan, Z.; and Yang, K. 2020. Feature-metric loss for self-supervised learning of depth and egomotion. In *European Conference on Computer Vision*, 572–588. Springer.

Wang, J.; Zhang, G.; Wu, Z.; Li, X.; and Liu, L. 2020. Self-supervised joint learning framework of depth estimation via implicit cues. *arXiv preprint arXiv:2006.09876*.

Wang, L.; Wang, Y.; Wang, L.; Zhan, Y.; Wang, Y.; and Lu, H. 2021. Can scale-consistent monocular depth be learned in a self-supervised scale-invariant manner? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 12727–12736.

Watson, J.; Aodha, O. M.; Prisacariu, V.; Brostow, G.; and Firman, M. 2021. The Temporal Opportunist: Self-Supervised Multi-Frame Monocular Depth. In *Computer Vision and Pattern Recognition (CVPR)*.

Yang, T.; Zhu, Y.; Xie, Y.; Zhang, A.; Chen, C.; and Li, M. 2023. AIM: Adapting Image Models for Efficient Video Action Recognition. *arXiv preprint arXiv:2302.03024*.

Yin, Z.; and Shi, J. 2018. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1983–1992.

Zaken, E. B.; Ravfogel, S.; and Goldberg, Y. 2021. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*.

Zhou, T.; Brown, M.; Snavely, N.; and Lowe, D. G. 2017. Unsupervised Learning of Depth and Ego-Motion from Video. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6612–6619. Honolulu, HI: IEEE. ISBN 978-1-5386-0457-1.

Zhu, Y.; Feng, J.; Zhao, C.; Wang, M.; and Li, L. 2021. Counter-interference adapter for multilingual machine translation. *arXiv preprint arXiv:2104.08154*.

Appendix

A Supplementary Experiments

A.1 Robustness to Dynamic Scenes

In our method, most network parameters are frozen and unaffected by the erroneous loss caused by object motion. Evidence is presented in Table 2, where tuning adapters (the last row) outperforms tuning the entire network (the third to last row) on CityScapes, a dataset with prevalent moving objects.

We conduct further experiments to validate the efficacy of our method by excluding areas of movable objects, such as cars and pedestrians, from the loss computation during training. Only the result of tuning the entire network is improved but our method cannot be enhanced (see Tab. 7). This shows the robustness of our approach to dynamic objects. Furthermore, even when excluding dynamic objects, tuning the entire network is still worse than our method, emphasizing the benefits of the adapter-tuning strategy.

Tuning Strategy	Dynamic Objects	AbsRel	SqRel	$\delta < 1.25$
Full Fine-Tune	Included	0.116	1.120	0.873
	Excluded	0.106	1.000	0.890
Adapters (Ours)	Included	0.100	0.976	0.904
	Excluded	0.102	1.113	0.904

Table 7: PPEA-Depth's Robustness to Dynamic Objects.

A.2 Improvement with encoder adapters arises from a more complex model?

Supplement to Table 1, we did another experiment on Stage 1. We employ RepLKNet-B as the depth encoder and tune all the parameters of both the base model and encoder adapters. There is no significant performance gain (see Table 8). This proves that our PEFT scheme is the main reason for the improvement shown in Table 1 and our approach can better exploit the pre-trained encoder than tuning all parameters.

Encoder	Decoder	Encoder Adapters	AbsRel	SqRel	$\delta < 1.25$
Tuning	Tuning	Tuning	0.093	0.703	0.911

Table 8: Tuning both encoder adapter and entire network is worse than freezing encoder and tuning encoder adapters.

A.3 Comparison to Linear Probing

PPEA-Depth can tune each layer in the network by adding adapters. It is more flexible than linear probing, where only the last linear layer can be tuned. We've made a comparison with a strategy similar to classifier adjustment, where we freeze the encoder and only tune the decoder *in Table 1 (Row 1)* for Stage 1. Results of this strategy for Stage 2 are also worse than our approach (see Table 9), showing the necessity of encoder adapters for domain shift in Stage 2.

B Ablations for Adapter Design

We conduct experiments to assess various adapter designs using RepLKNet-B. All the reported results are from the student network.

B.1 Encoder Adapter

We compare different encoder adapter designs for the domain adaptation stage on the KITTI dataset.

Type of Blocks to Attach Adapters Our method involves attaching encoder adapters to two types of blocks within the RepLKNet (Ding et al. 2022) backbone: RepLKBlock and ConvFFN. We conducted an ablation study by adding adapters to only one type of these blocks. The results are presented in Table 10. Evidently, the frozen encoder benefits more from adding adapters to RepLKBlock rather than ConvFFN. RepLKBlock, which leverages a large kernel size of up to 31x31, stands as the core innovation of RepLKNet. This outcome aligns with the intuition that adapters on more intricate blocks carry greater significance.

Receptive Field of Projectors Adapters project the input feature down to a lower dimension and then project it up. As depth estimation is a regression task, will encoder adapters benefit from a larger receptive field? We conducted experiments using different receptive fields for the adapters of RepLKBlock, which, as indicated by the results in Table 10, holds greater importance than ConvFFN. We compared two types of projectors: linear projection and convolution with a kernel size of 3, while maintaining a bottleneck ratio of 0.25. The results are presented in Table 11. From the results, it can be inferred that all metrics benefit from a larger receptive field in the down projector, while the enhancement resulting from a wider receptive field in the up projector is not as apparent.

BatchNorm and Skip Connections We experiment with different encoder adapter designs by varying the position of the BatchNorm module and skip connections (Fig. 4), while keeping the inner adapter design the same (bottleneck ratio is 0.25, down projector is a 3x3 convolution and up projector is linear). The evaluation results of different designs are in Table 12.

The BatchNorm module before the pre-trained block is part of the original RepLKBlock and ConvFFN design (more details are shown in Fig. 6). We design to let the adapter share the BatchNorm module with the original pre-trained block, and unfreeze the parameters of the BatchNorm during training. From the results in Table 12, it can be concluded that:

- The final depth estimation accuracy benefit from the BatchNorm block before the encoder adapter. Removing the prefix BatchNorm before encoder adapters results in an obvious performance loss (Design (b)).
- The BatchNorm module before the pre-trained block is indispensable (Design (c)). As the parameters in the pre-trained blocks are frozen, the frozen parameters are matched with the output of the prefix BatchNorm, removing the prefix BatchNorm for the pre-trained block leads to degraded results.

Encoder	Encoder Adapters	Decoder	Decoder Adapter	*Errors*↓		*Accuracy*↑
				AbsRel	SqRel	$\delta < 1.25$
Frozen	Frozen	Tuning	Not Used	0.114	1.215	0.878
Frozen	Frozen	Frozen	Tuning	0.107	1.004	0.889

Table 9: Results of *Classifier Adjustment* for Stage 2.

Adapter Position	Params (M)	*Errors*↓				*Accuracy*↑		
		AbsRel	SqRel	RMSE	$\text{RMSE}_{\log}$	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
RepLKBlock	17.7	0.093	0.718	4.285	0.171	0.909	0.968	0.984
ConvFFN	3.78	0.100	0.769	4.442	0.175	0.900	0.966	0.984
All	21.9	0.090	0.666	4.175	0.168	0.912	0.969	0.984

Table 10: **Comparison of Attaching Adapters to Different Types of Blocks.** Attaching adapters to more sophisticated blocks in backbone leads to better results.

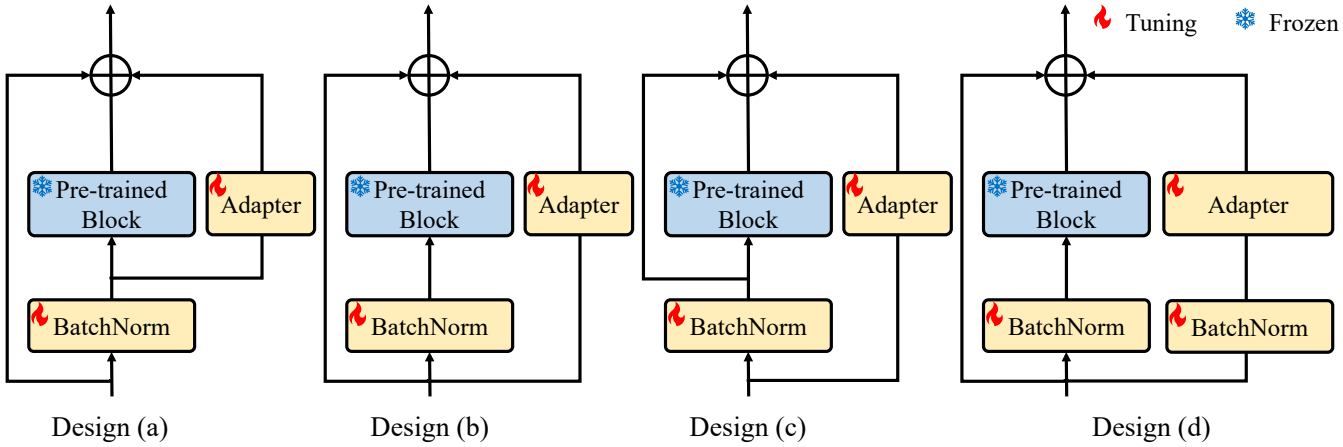


Figure 4: **Different Encoder Adapter Designs**. Design (a) is our final choice.

- Sharing the same prefix BatchNorm module between the pre-trained block and the encoder adapter (Design (d)) and training an extra BatchNorm for the encoder adapter (Design (a)) show no obvious difference in the final depth estimation errors and accuracy. The sharing strategy leads to a lower trainable parameter cost, so design (a) in Fig. 4 is our choice.

Bottleneck Ratio. The performance improves with more tunable parameters up to a bottleneck ratio of 0.25 in adapters, beyond which the benefits plateau (Table 14).

B.2 Decoder Adapter

We compare different decoder adapter designs for the scene adaptation stage on the CityScapes dataset. We keep the encoder adapter design consistent (bottleneck ratio is 0.25, down projector is a 3x3 convolution, and up projector is linear) and use the same weights trained from the domain adaptation stage as initial model weights in the following experiments.

Input Scales The depth encoder generates feature maps at four levels (F^0, F^1, F^2, F^3), corresponding to $1/4$, $1/8$, $1/16$, and $1/32$ of the original image spatial shape. Upsampling feature maps at different levels to the $1/4$ scale, we concatenate and feed them to the decoder adapters. We compare different input feature scales for the decoder adapter, and the results are in Table 13, suggesting that using the concatenation of the shallowest and deepest features as input is the optimal choice, considering both parameter count and performance.

C Inference Latency Introduced by Adapter

As adapters are additional modules added to the network, we investigate the inference latencies introduced by the encoder and decoder adapters. We measure the total time required for the model to predict depth maps for 1,525 images in the CityScapes test dataset and calculate the average inference time per image. Our encoder backbone is RepLKNet-B, and we use a batch size of 12. The results are presented in Table 15. Thanks to the parallel data stream design, the inference time remains largely consistent whether adapters are used or not.

Down Projector	Up Projector	Params (M)	*Errors*↓				*Accuracy*↑		
			AbsRel	SqRel	RMSE	$\text{RMSE}_{\log}$	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Linear	Linear	7.5	0.095	0.776	4.372	0.174	0.909	0.967	0.983
Conv	Linear	21.9	**0.090**	**0.666**	**4.175**	**0.168**	**0.912**	**0.969**	**0.984**
Conv	Conv	35.4	**0.090**	0.671	4.203	0.169	**0.912**	0.968	**0.984**

Table 11: **Influence of Encoder Adapter Receptive Field.** Domain adaptation stage benefits from employing encoder adapters with down projector of larger receptive fields, while the improvement brought by up projector of larger receptive fields is not obvious.

Design	*Errors*↓				*Accuracy*↑		
	AbsRel	SqRel	RMSE	$\text{RMSE}_{\log}$	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
a	**0.090**	**0.666**	**4.175**	**0.168**	**0.912**	**0.969**	**0.984**
b	0.095	0.769	4.407	0.175	0.905	0.965	0.983
c	0.103	0.761	4.499	0.181	0.890	0.962	0.983
d	0.091	0.680	4.219	0.170	0.911	0.968	0.984

Table 12: **Comparison of Different BatchNorm and Skip Connection Designs** in Encoder Adapters.

Input Scales	Params (M)	*Errors*↓				*Accuracy*↑		
		AbsRel	SqRel	RMSE	$\text{RMSE}_{\log}$	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
3	0.149	0.103	0.977	5.693	0.157	0.895	0.974	0.991
0, 3	0.185	**0.100**	**0.976**	5.673	**0.152**	**0.904**	**0.977**	**0.992**
0, 1, 2, 3	0.486	0.101	0.982	**5.666**	**0.152**	**0.904**	**0.977**	**0.992**

Table 13: **Comparison of Different Input Feature Levels** to Decoder Adapter.

Ratio	Tuning Params (M)	AbsRel	SqRel	$\delta < 1.25$
0.0375	6.4	0.098	0.779	0.902
0.0625	8.2	0.092	0.686	0.910
0.1	10.7	0.093	0.704	0.911
0.25	21.9	0.090	0.666	0.912
0.5	38.7	0.090	0.674	0.912
1	73.6	0.090	0.659	0.911

Table 14: **Effect of Adapter Bottleneck Ratio.**

Encoder Adapters	Decoder Adapters	Time (s)
		0.037
●		0.036
●	●	0.038

Table 15: **Average Inference Time Per Image** on CityScapes.

D Supplementary Details

D.1 Overview of the Whole Framework

We follow the self-supervised depth estimation framework proposed by Watson et al. (2021), whose depth network consists of a teacher network and a student network to improve estimated depths and better handle dynamic objects, and this design is also adopted in Guizilini et al. (2022); Feng et al. (2022). The teacher depth network is trained jointly with the student, sharing the same pose predictions, and is discarded during evaluation. Both the teacher and student networks are based on the U-Net architecture. They share identical designs for the depth network decoder but differ in their encoders. The teacher's input is a single image frame (the frame at timestamp t, denoted as I_t). This input, I_t, undergoes processing through the depth encoder and subsequently the depth decoder, leading to pixel-wise depth estimation.

As shown in Figure 5, the input of the student network includes two frames, I_t and its previous frame I_{t-1}. The two frames are first separately extracted by the first stage of the encoder to generate first-level features F_t^0, F_{t-1}^0, which are both in the shape of $(B, C, H/4, W/4)$, where B is the batch size, C is the number of channels, H and W represent the spatial shape of I_t. Then a cost volume is built based on F_t^0, F_{t-1}^0, relative camera pose T (which is currently predicted by the pose network), and a set of depth values D_c.

Iterating over D_c, we project F_{t-1}^0 using T and the iterated depth value d_i to generate $F_{t-1\to t}^0$ according to the pixel correspondences mentioned in Equation (1) in the main paper. The L1 difference between $F_{t-1\to t}^0$ and F_t^0 is adopted to build the cost volume. The minimum and maximum depths (d_{min}, d_{max}) are initialized as 0.1m and 100m respectively, and are dynamically tuned in the learning pro-

cess according to the strategy proposed in Watson et al. (2021). The depth set D_c is generated by uniformly sampling depth values in $[d_{min}, d_{max}]$ in logarithm space.

The generated cost volume C_t is in the shape of $(B, |D_c|, H/4, W/4)$, where $|D_c|$ represents the number of depths in set D_c. Then C_t is concatenated with F_t^0 at the second dimension, and the concatenated feature is compressed to the shape of $(B, C, H/4, W/4)$ by a 3x3 convolution called *reduce conv*. Then the output of the reduce conv is fed to the rest stages of the depth encoder and then to the depth decoder, and finally predicts a depth map D_t.

The cost volume design in the student network introduces I_{t-1} and an iteration over all possible depths in the encoder, and exploits the relationship between two consequent frames to improve depth estimation. However, such a design exaggerates the depth estimation error in the dynamic object areas and tends to predict depths of such areas as infinity (Watson et al. 2021).

D.2 Teacher-Student Distillation Scheme

To overcome the infinity-depth issue as mentioned above, a teacher-student training scheme is employed in our network as in previous works (Watson et al. 2021; Feng et al. 2022; Guizilini et al. 2022).

For the dynamic object areas, the predictions from the student network are unreliable. Such unreliable area M is computed by comparing the predicted depths from the teacher and student in a pixel-wise manner:

$$M = \max(\frac{D_s - D_t}{D_t}, \frac{D_t - D_s}{D_s}) > 1$$

During the training process, the teacher network is supervised by the image reprojection loss (as mentioned in Section 3.1 in the main paper). For the student network, the reliable area ($\neg M$) is also supervised by the image reprojection loss, while the unreliable area (M) is instead supervised by a depth consistency loss to enforce a knowledge distillation from the teacher. The depth consistency loss is the L1 difference of the predicted depths between the teacher and the student. In the depth consistency loss, gradients to depths predicted by the teacher are blocked, ensuring the distillation is unidirectional, i.e. only from teacher to student.

Different from the previous works (Watson et al. 2021; Feng et al. 2022; Guizilini et al. 2022), we do not freeze the teacher depth network and pose network to fine-tune in the last five epochs, since we do not observe a significant performance improvement with such technique on PPEA-Depth.

D.3 Amount of Encoder Adapter Parameters

All parameter counts mentioned in the main paper are based on the teacher network. Here, we provide supplementary details regarding the tunable parameter count for encoder and decoder adapters in the student network. Throughout all experiments, the settings for student encoder adapters (down projector, up projector, and bottleneck ratio) remain the same as those of the teacher network. The student network has more tunable parameters than the teacher because apart from the encoder adapters and the BatchNorm module, the reduce conv module (as detailed in Section D.1) also receives updates. Refer to Table 16 for specific parameter details.

Pre-trained Backbone	Adapter Type	Ratio	Tuning Params(M)	
			Teacher	Student
RepLKNet-B	Encoder	0.0625	8.15	8.41
	Encoder	0.25	21.9	22.2
	Decoder	0.25	0.185	0.185
RepLKNet-L	Encoder	0.0625	18.1	18.6
	Encoder	0.25	47.6	48.1
	Decoder	0.25	4.15	4.15

Table 16: **Details of Tunable Parameters** in Teacher and Student Depth Network Adapters.

D.4 RepLKNet

In Section 3.3 of the main paper, we briefly introduce the main pipeline of RepLKNet-31B, which we adopt as the encoder backbone of both the teacher and student depth network. Here we introduce the detailed structure of the two key modules to which we attach encoder adapters, RepLKBlock and ConvFFN. For more details please refer to Ding et al. (2022).

The detailed structures of RepLKBlock and ConvFFN are shown in Figure 6. Our proposed encoder adapters share the same prefix BatchNorm module with RepLKBlock and ConvFFN (as in Figure 4). Since we adopt a different batch size during training compared with the RepLKNet pretraining process, all parameters of BatchNorm modules are not frozen.

D.5 Training Details

PPEA-Depth is implemented with PyTorch (Paszke et al. 2017) and is trained on GeForce RTX 3090 GPU. We use the Adam optimizer (Kingma and Ba 2014), with $\beta_1 = 0.9, \beta_2 = 0.999$. At the domain adaptation stage (Stage 1), the batch size is set to 12. This stage trains for 25 epochs, and the initial learning rate is 1e-4. The learning rate decays to 1e-5 after 15 epochs and decays to 1e-6 after 20 epochs. During the scene adaptation stage (Stage 2), we utilize a batch size of 24 for training. This stage is conducted over 10 epochs, employing a learning rate of 1e-5 for CityScapes and 5e-5 for DDAD.

D.6 Evaluation Metrics Details

We evaluate our depth estimation results using the standard depth assessment metrics (Eigen and Fergus 2015), including Absolute Relative Error (*AbsRel*), Squared Relative Error (*SqRel*), Root Mean Squared Error (RMSE), Root Mean Squared Log Error ($\mathrm{RMSE}_{\log}$), δ_1, δ_2, and δ_3. The specific

formulas to calculate these metrics are as follows:

$$
\begin{aligned}
AbsRel &= \frac{1}{n}\sum_{i}\frac{p_i - g_i}{g_i} \\
SqRel &= \frac{1}{n}\sum_{i}\frac{(p_i - g_i)^2}{g_i} \\
\mathrm{RMSE} &= \sqrt{\frac{1}{n}\sum_{i}(p_i - g_i)^2} \\
\mathrm{RMSE}_{\log} &= \sqrt{\frac{1}{n}\sum_{i}(\log p_i - \log g_i)^2}
\end{aligned}
\tag{4}
$$

and the values δ_1, δ_2, and δ_3 represent the percentages of pixels where the condition $\max\,(p/q,\ q/p) < 1.25, 1.25^2, 1.25^3$ is satisfied. Here, g denotes the ground truth depth, p stands for the predicted depth, and n represents the number of pixels.

E Supplementary Qualitative Results

We provide more qualitative comparisons on CityScapes in the last two pages. The images are organized from left to right, showcasing the original image, the estimated depth obtained by full fine-tuning a U-Net from scratch, and the estimated depth produced by our PPEA-Depth approach. We also provide a qualitative video demo (*demo.mp4* in the .zip file). Please check it out for a more dynamic representation of our approach's performance.

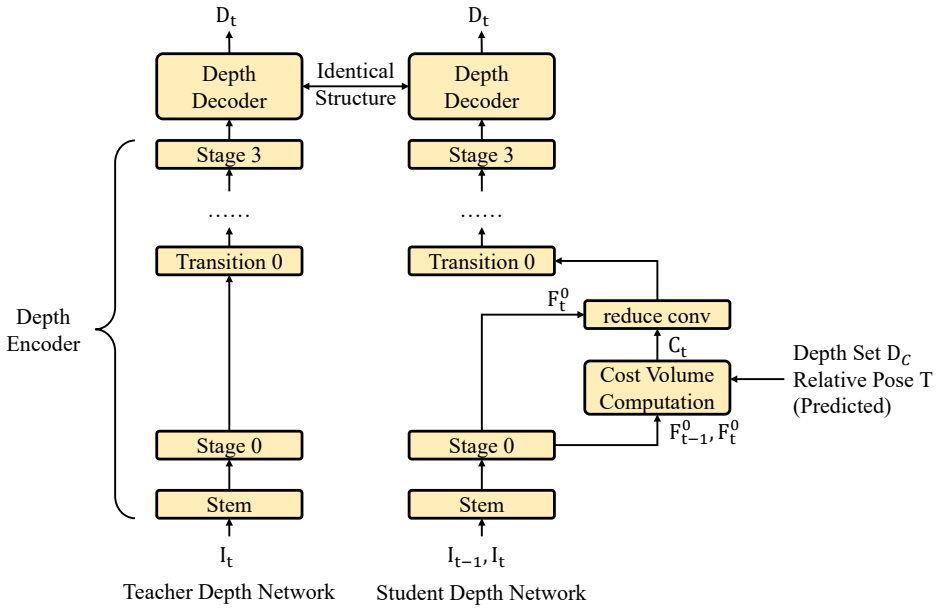


Figure 5: **Detailed Structure of Teacher and Student Depth Network.**

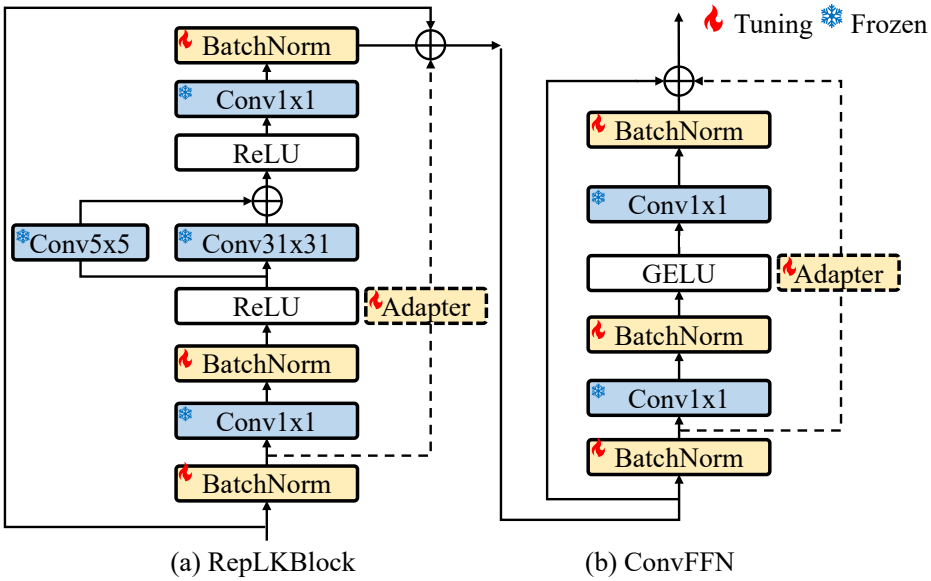
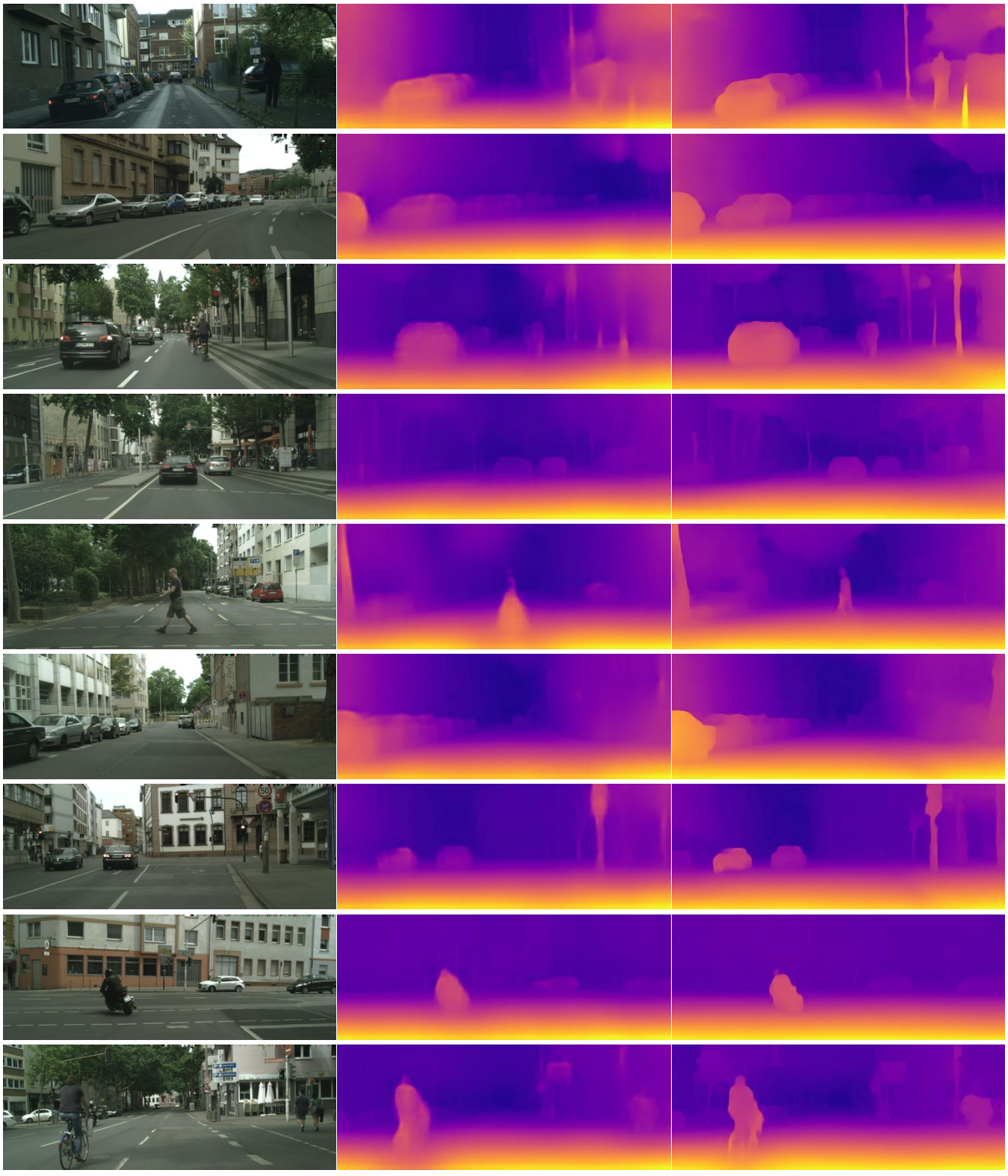


Figure 6: **Detailed Structure of RepLKBlock and ConvFFN**. The dashed line depicts the position of encoder adapters in PPEA-Depth.



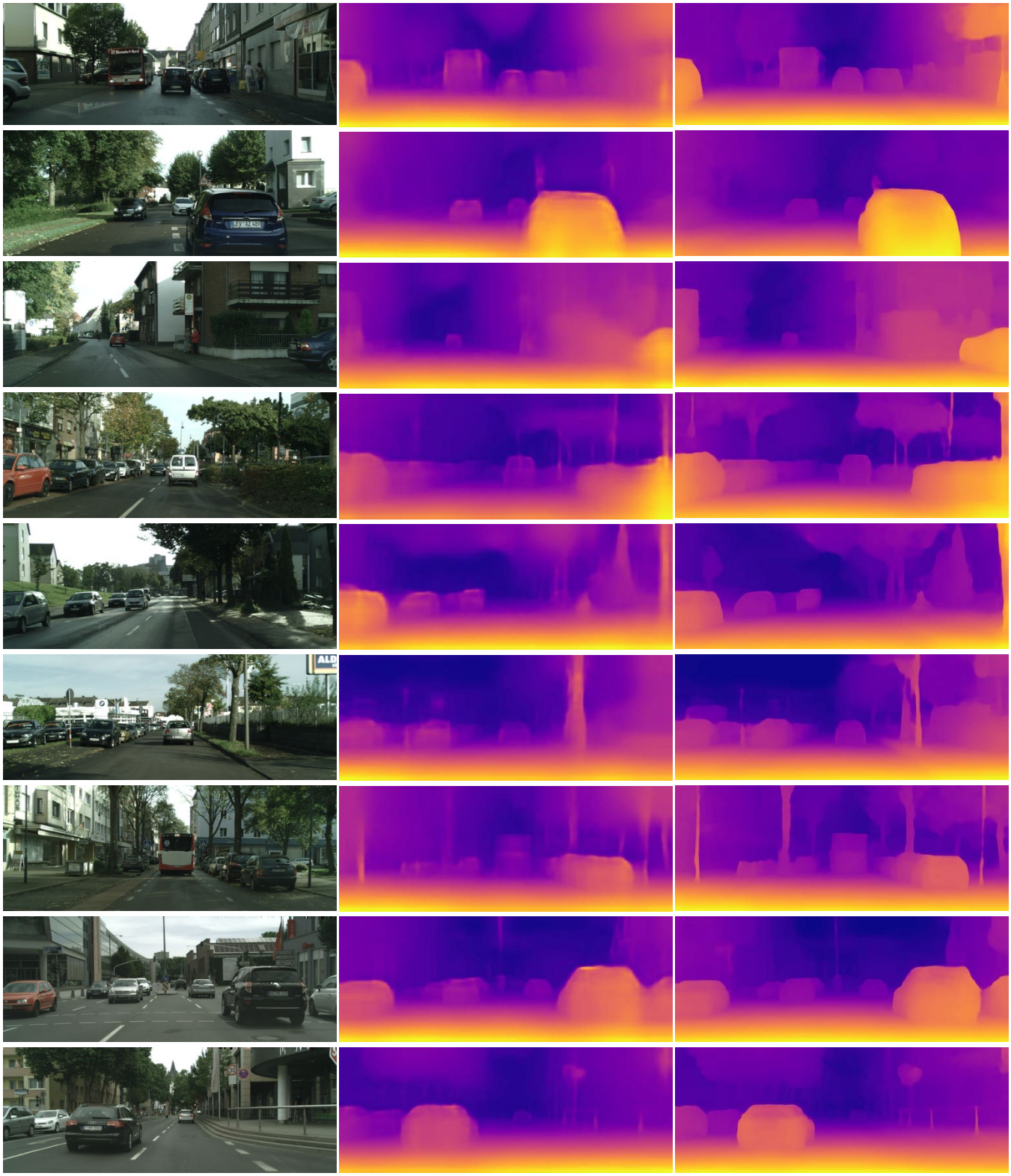


Figure 7: **Qualitative Comparisons on CityScapes test dataset**. *Left*: original input; *Middle*: estimated depth by full fine-tuning a U-Net from scratch; *Right*: estimated depth by our PPEA-Depth.