Learn to Adapt for Monocular Depth Estimation

Qiyu Sun, Gary G. Yen, Fellow, IEEE, Yang Tang, Senior Member, IEEE, and Chaoqiang Zhao

Abstract—Monocular depth estimation is one of the fundamental tasks in environmental perception and has achieved tremendous progress in virtue of deep learning. However, the performance of trained models tends to degrade or deteriorate when employed on other new datasets due to the gap between different datasets. Though some methods utilize domain adaptation technologies to jointly train different domains and narrow the gap between them, the trained models cannot generalize to new domains that are not involved in training. To boost the transferability of depth estimation models, we propose an adversarial depth estimation task and train the model in the pipeline of meta-learning. Our proposed adversarial task mitigates the issue of meta-overfitting, since the network is trained in an adversarial manner and aims to extract domain invariant representations. In addition, we propose a constraint to impose upon cross-task depth consistency to compel the depth estimation to be identical in different adversarial tasks, which improves the performance of our method and smoothens the training process. Experiments demonstrate that our method adapts well to new datasets after few training steps during the test procedure.

Index Terms—meta-learning, transferability, depth estimation

I. INTRODUCTION

Monocular depth estimation is a classical task in computer vision, which aims to estimate the distances between the objects in environment and the agent itself [1], [2], and thus it is an essential task in environmental perception [3], [4]. Recently, deep learning-based depth estimation methods, including supervised methods [5] and unsupervised methods [1], [2], [6], are proposed and achieve significant progress. The supervised methods are trained through images with ground truth and their performances are often reliable. On the other hand, the unsupervised methods are trained through unlabeled images and hence the training data are more available. In general, though the deep learning-based depth estimation models perform well on the training dataset, they suffer from distinct performance degradation when applied to new datasets [7]. Such degradation occurs due to the differences of data distributions exist between the training and testing datasets, and such phenomenon is known as domain shift [8], [9]. In order to ameliorate the issue of domain shift, some domain adaptation technologies are proposed to fill the gap existing between different datasets [8], [10]. For example, some approaches attempt to reduce the domain gap between synthetic datasets and real datasets in depth estimation [11]–[14]. Even though, synthetic (source domain) and real datasets (target domain) are indispensable during training, and the model cannot generalize to new domains that not participating in training. In practical scenarios, the model that can adapt to various unseen domains is urgently needed.

Recently, meta-learning [17], [18], also known as learning to learn, shows the possibility of learning an adaptive model which can generalize well to new domains with limited training steps. In the training phase, the model is trained on different tasks to learn how to adapt to new tasks or new domains with few samples. In the test phase, the model is trained with new datasets on the basis of the model obtained in the training phase. Since the model trained with meta-learning includes prior mutual knowledge about training tasks, it can be applied to a new task with a small amount of follow-up training steps during test. It is worth noting that large quantities of datasets are an essential precondition for the excellent transferability of meta-learning, therefore meta-learning is well applied to image classification tasks thanks to the existing numerous classification datasets [19], [20]. If the tasks for training are limited, meta-learning suffers from distribution shift issue as well, which is also called the meta-overfitting [20]–[22]. This is because that the limited tasks cannot ensure the distribution of testing tasks resembles that of training tasks. Unfortunately, the datasets for depth estimation are quite limited, which raises challenges when integrating meta-learning into depth estimation.

As the problem of poor generalization between datasets imposes great restrictions on real-world applications of depth estimation, this work explores the possibility of learning a more adaptive depth estimation model which can generalize...
well to new domains. We refer to model-agnostic meta-learning (MAML) [16], a well-known algorithm of meta learning, to endow depth estimation models the transferability across different domains. To maintain the generalization of meta-learning in depth estimation tasks in case of limited training datasets, we design an adversarial depth estimation task. Our proposed adversarial task estimates the depth maps of two images sampled from different datasets simultaneously and such combination of datasets increases the amount of tasks. Then, the estimated depth maps are compared in an adversarial manner to extract the domain invariant representations, thus avoiding the overfitting in training. Since the supervised methods rely on plenty of training data with depth ground truth, which is laborious and time consuming to collect for real-world scenes, we adopt the unsupervised pipeline using monocular image sequences for training to obtain more training datasets [1], [15]. Both real images collected by cameras and synthetic images generated by graphics engines [23], [24] are utilized for training in this work. In view of the fact that the unsupervised pipeline is vulnerable to the dynamics in environment [2], we propose a constraint to compel the depth maps estimated in different adversarial tasks to be identical. We name it as the cross-task depth consistency constraint and it is applied in meta-update. In addition, the proposed constraint helps to improve the performance of training and smoothen the convergence curve because it establishes the connection between different training tasks and then the model is optimized in a collaborative manner.

To sum up, we propose an adversarial domain-adaptive algorithm for monocular depth estimation, which is able to generalize to unseen datasets or scenes quickly in test phase and achieves a satisfactory performance. The adversarial depth estimation task we introduce alleviates the meta-overfitting even trained on a handful of datasets. Experiments demonstrate that our meta-learning based monocular depth learning method can rapidly adapt to new, unseen datasets during test. As shown in Figure 1, our method (Figure 1(f)) achieves better performance than the state-of-the-art work [15] (Figure 1(c)) after few steps of training. For further comparison, we train the model using the basic [15] (Figure 1(d)) and in the pipeline of MAML (Figure 1(e)) for 0.5 epoch, respectively. The results show that our method adapts well to a new dataset rapidly. Our contributions are summarized as follows:

- We propose an adversarial training pipeline for monocular depth estimation based on meta-learning, which learns appropriate initial network parameters for adapting efficiently to unseen domains with few steps of adaptation. Meanwhile, our method alleviates the issue of meta-overfitting even when trained on few datasets.
- We propose a cross-task depth consistency constraint handling for meta-update. It compels the depth estimated from the same image in different tasks of meta-learning to be identical, which stabilizes the training process and improves the performance of our method.
- The model trained by our method adapts well to datasets which do not appear in the training phase. Our method trained through few updates obtains comparable results with some state-of-the-art works trained with much more time.

The organization of this paper is arranged as follows. Section II presents a brief review of related works. The method of this paper is introduced in the Section III. The Section IV shows the experimental results about the transferability of the proposed depth estimation model. Finally, the Section V draws a conclusion on this study and some future prospects are provided.

II. RELATED WORKS

A. Depth Estimation

Depth estimation is of great importance in environmental perception, and many existing methods attempt to solve it in an end-to-end manner thanks to deep learning [25]. The methods estimate depth in either a supervised manner or an unsupervised manner or an unsupervised manner according to training data. The supervised methods [5] are trained with images and their corresponding ground truth, thus the estimation results are more reliable. However, the acquisition of depth labels in real environments is time consuming and costly. For unsupervised methods, they can be further divided into two groups according to training data, trained on stereo image pairs [26], [27] or monocular image sequences [1], [15], [28], [29]. Among all the training modes, the unsupervised methods based on monocular images are the most attractive because plenty of training data are available. However, unsupervised methods are not as reliable as supervised ones and are more vulnerable to the dynamic variations in environments [2]. Though all these deep learning based depth estimation methods dedicate to improve the performance of the model on a particular dataset, they neglect the transferability of the model, which imposes significant restrictions on real-world applications.

B. Adversarial Domain Adaptation

Recently, deep learning flourishes in the performance of numerous computer vision tasks [30]–[32]. However, the model trained on a specific dataset is frequently incapable of generalizing well on another dataset due to dataset bias [33], [34]. Domain adaptation [8], [10] is proposed to narrow the gap between different datasets and proved to be efficient in different computer vision tasks, including classification [30], semantic segmentation [31], and depth estimation [32], etc.

Among various domain adaptation works, adversarial-based domain adaptation methods are the dominant ones [8], [10]. Some methods narrow the distribution shift between different datasets in latent feature space [31], [35], [36], which are categorized as feature-based domain adaptation methods. Though these feature-based methods can perform well in classification tasks, they tend to fail in more complicated tasks like depth estimation and semantic segmentation [37], [38]. The development of Generative Adversarial Networks (GAN) [39] promotes the emergency of input-based [32], [40], [41] and output-based domain adaptation methods [13], [14], [42], [43]. These methods greatly enhance the transferability of deep models among different datasets. The input-based
domain adaptation methods almost take advantage of style transfer networks [44], [45] to make the training images indiscernible across domains. The output-based domain adaptation methods [13], [14], [42], [43] take advantage of the network predictions. For pixel wise estimation tasks such as depth estimation, the output predictions contain structured spatial information, which are not sensitive to domains.

Specially, numerous domain adaptation works focus on depth estimation. Some methods use an adversarial objective to distinguish which domain the extracted feature comes from with feature-based methods [11], [12], while other works dedicate to improving the transferability of the model with input-based methods [32], [41], [46], [47]. Notice that depth estimation is a spatially-structured prediction task, Kundu et al. [13], [14] improve its across-domain generalization capability by adding a consistency constraint on the output space. Though domain adaptation methods can relax the domain shift issue between two or several datasets, the model trained using these methods can only perform well on these datasets and tends to fail in other datasets which are not involved in training.

C. Meta-Learning

Meta-learning [17], [18] is designed to train a model through various tasks to equip the model with the ability of solving new tasks with a small quantity of training samples. It has gained tremendous attention in few-shot learning [48] and reinforcement learning [49]. Recently, meta-learning is applied to some computer vision tasks as well [50]–[52]. MAML [16] is a well-known algorithm for meta learning. It prompts the network to learn common initial parameters across different training tasks, and the initial parameters can be reloaded in the testing phase to ensure an advanced generalization performance on unseen tasks quickly.

However, the superior performance of meta-learning in generalization is obtained by training on various tasks and an appropriate task distribution [20], [22]. Some reinforcement learning works have noticed such problem and attempted to address it [20], [22]. Mehta et al. [22] utilize domain randomization methods to optimize the distribution of tasks and obtain more stable initial parameters for variable testing tasks. Lin et al. [20] propose AdMRL to optimize the worst performance over all the tasks and improve the generalization ability of the model. The same problem exists in depth estimation as well. Dozens of tasks are needed for training in meta-learning generally [53], while the datasets for depth estimation are quite limited. Thus, we train the depth estimation model in an adversarial manner using only four datasets to extract domains invariant representations, and attempt to alleviate the issue of meta-overfitting in depth estimation tasks due to the limited training tasks.

III. PROPOSED METHOD

In this section, we introduce the proposed method in detail. The adversarial depth estimation task for meta-learning is introduced first in Subsection III-A, the parameter update of the proposed task is introduced in Subsections III-B, and the proposed cross-task depth consistency constraint is presented in Subsections III-C.

A. Adversarial Task for Depth Estimation

As shown in Figure 2(a), the adversarial task designed includes a generator $G$ for depth estimation and a discriminator $D$ for domain invariant feature extraction. Images from different domains are trained jointly in an adversarial manner. Monocular depth is estimated through sequence of snippets consisting of two consecutive frames through (b) a DepthNet for pixel-level depth estimation and (c) a PoseNet for relative pose estimation between two images.

![Fig. 2. The illustration of the adversarial task designed for meta-learning. (a) The task includes a generator $G$ for depth estimation and a discriminator $D$ for domain invariant feature extraction. Images from different domains are trained jointly in an adversarial manner. Monocular depth is estimated through sequence of snippets consisting of two consecutive frames through (b) a DepthNet for pixel-level depth estimation and (c) a PoseNet for relative pose estimation between two images.](image-url)
representations [35] to ease the meta-overfitting when trained on few tasks.

In order to demonstrate the effectiveness of our meta-learning algorithm when trained on only few datasets, we merely choose four representative datasets for training in this work. Other datasets for depth estimation can be added for training as well, which may facilitate meta-learning but increase training time. To be specific, we choose SYNTHIA [24], Cityscapes [54], Oxford RobotCar [55] and vKITTI [23] as training datasets. Two of them are synthetic datasets and the other two are real datasets collected by camera. We combine them in pairs (SYNTHIA and Cityscapes, for example) to obtain six different combinations and constitute six tasks used for training, thus this kind of combination increases the number of training tasks. Therefore, the adversarial task for depth estimation can not only extract domain invariant representations but also increase the number of training tasks, which help to overcome the difficulty in meta-learning brought by the lack of training datasets. 

The depth estimation model is trained on both real monocular image sequences and synthetic image-depth pairs. Real images without the corresponding ground truth are trained in unsupervised manner, and the mainly supervised signal comes from view reconstruction [15]. The loss function is denoted as $L^u$ and it is the same as the loss function used for monocular depth estimation in [15]. For synthetic images with ground truth, they are trained in both of supervised and unsupervised manners. The supervised loss $L^s$ is calculated by the L1-norm of the predicted depth and depth ground truth, and the loss function is $L^s + L^u$. Then, the loss for depth estimation (generator), $L^G$, is

$$L^G = \begin{cases} L^u, & \text{for real data} \\ L^s + L^u, & \text{for synthetic data} \end{cases}. \quad (1)$$

After the training snippets from different datasets being fed into the generator for depth estimation, the depth estimation results are sent into the discriminator. If the discriminator cannot distinguish whether the depth estimation results come from dataset $A$ or dataset $B$, it demonstrates that the generator can extract the domain invariant features. We adopt the GAN loss [39] as the adversarial objective, $L^D$,

$$L^D = \mathbb{E}_{x_A \sim p_{data}(x_A)} [\log D(G(x_A))] + \mathbb{E}_{x_B \sim p_{data}(x_B)} [\log D(1 - D(G(x_B)))] \quad (2)$$

where $x_A$, $x_B$ are the training data from dataset $A$ and $B$, $G(x_A)$ and $G(x_B)$ aim to estimate the depth from images from dataset $A$ and $B$, and $D$ tries to distinguish whether the input depth map is from dataset $A$ or $B$.

**B. Learn to Adapt with Meta-learning**

Numerous tasks are needed for meta-learning, and we denote the task set as $\mathcal{T}$. During training, several tasks are sampled from $\mathcal{T}$ in order to train an adaptive model which can perform well on new tasks after test. Different samples are fetched in the selected task, and are split into two sets, the support set and the query set. To be specific, when a task $\mathcal{T}_i$ is sampled from the task set $\mathcal{T}$, it is trained through the corresponding loss function $L_{\mathcal{T}_i}$ to update parameters. $K_s$ samples (the support set $\mathcal{D}^s_{\mathcal{T}_i}$) are fetched from $\mathcal{T}_i$ for training using $L_{\mathcal{T}_i}$, and then tested on $K_q$ new samples (the query set $\mathcal{D}^q_{\mathcal{T}_i}$) sampled from $\mathcal{T}_i$ each time. The model is optimized according to the errors of all the samples in query set calculated by $L_{\mathcal{T}_i}$, that is to say, the test error of the query set is the meta-training error of training. During test, new tasks will be sampled, trained on its corresponding support sets, and then tested on query sets to demonstrate the meta-performance.

MAML [16] is designed to train the initial parameters of a model, so that the model can adapt to a new task through one or several gradient updates given a handful of data from the new task. As shown in Figure 3(a), considering a model $f_\theta$ with parameters $\theta$, the model optimizes through training on different tasks (six tasks in this work). For each task $\mathcal{T}_i$, the model updates its parameters $\theta$ to $\theta_i'$. The model is updated through one or several gradient descents. The update using one gradient descent is as follow:

$$\theta_i' = \theta - \alpha \nabla_\theta L_{\mathcal{T}_i}(f_\theta)\quad (3)$$

where $\alpha$ is the learning rate of the fast adaptation. Then, the model is trained by minimizing all the errors of $D^q_{\mathcal{T}_i}$ tested on the updated model $f_{\theta_i'}$. The meta-optimization is performed through stochastic gradient descent (SGD) according to the losses of query sets and the initial parameters $\theta$ are updated as follows:

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim \mathcal{P}(\mathcal{T})} L_{\mathcal{T}_i}(f_{\theta_i'}) \quad (4)$$

where $\beta$ is the learning rate of the meta-learning. In this work, the single task for meta-learning is our designed adversarial depth estimation task introduced in sub-
section III-A, as shown in Figure 2. In the training phase, a task-wise support set \( \mathcal{D}_{T_i}^s \) and a task-wise query set \( \mathcal{D}_{T_i}^q \) are sampled from the sampled task \( \mathcal{T}_i \), as shown in Figure 3(a). The depth estimation model is trained through the support set \( \mathcal{D}_{T_i}^s \), and for each training sample in \( \mathcal{D}_{T_i}^s \), the parameters of the model are fast updated in the inner loop. The initial parameters of the model are updated with the outer loop. We denote the parameters and discriminator in our model as \( \theta_G \) and \( \theta_D \), parameterized by \( \theta_G \) and \( \theta_D \), respectively. As illustrated in Figure 3(a), the inner loop of our meta-learning based method updates the model parameters in each separate task, and the outer loop updates initial model parameters of the inner loop. The loss function for inner loop training is the combination of depth estimate loss \( \mathcal{L}_{T_i}^\theta(G_{\theta_G}, D_{\theta_D}) \) and the adversarial loss \( \mathcal{L}_{T_i}^\alpha(D_{\theta_D}, \mathcal{D}_{T_i}^q) \). Then, the inner loop update loss \( \mathcal{L}_{T_i}^\theta(G_{\theta_G}, D_{\theta_D}) \) for task \( \mathcal{T}_i \) is:

\[
\begin{align*}
\mathcal{L}_{T_i}^\theta(G_{\theta_G}) &= \mathcal{L}_{T_i}^\theta(G_{\theta_G}(\mathcal{D}_{T_i}^s)) \\
\mathcal{L}_{T_i}^\theta(D_{\theta_D}) &= \mathcal{L}_{T_i}^\theta(D_{\theta_D}(G_{\theta_G}(\mathcal{D}_{T_i}^s))).
\end{align*}
\]

(5)

The parameters of \( G_{\theta_G} \) and \( D_{\theta_D} \) are updated alternately. For each task \( \mathcal{T}_i \), \( \mathcal{L}_{T_i}^\theta(G_{\theta_G}) \) and \( \mathcal{L}_{T_i}^\theta(D_{\theta_D}) \) are calculated to deduce their gradient, respectively. Then, the parameters \( \theta_G \) and \( \theta_D \) are updated to \( \theta_G' \) and \( \theta_D' \) with gradient descent. Note that the parameters of the generator and discriminator are updated in an alternative manner, as shown in Figure 3(b). The outer loop update is conducted when all the inner loops are terminated, and the parameters are optimized by minimizing the loss generated by \( G_{\theta_G} \) and \( D_{\theta_D} \), with query set \( \mathcal{D}_{T_i}^q \). The optimization objective of the outer loop meta-update (meta-objective) is as follows:

\[
\begin{align*}
\mathcal{L}_{T_i}^{\text{out}}(G_{\theta_G}) &= \mathcal{L}_{T_i}^{\text{out}}(G_{\theta_G}(\mathcal{D}_{T_i}^q)) \\
\mathcal{L}_{T_i}^{\text{out}}(D_{\theta_D}) &= \mathcal{L}_{T_i}^{\text{out}}(D_{\theta_D}(G_{\theta_G}(\mathcal{D}_{T_i}^q))).
\end{align*}
\]

(6)

Then, the corresponding gradients can be calculated by Equation 6 to update the model parameters. The parameters updating process of our method is illustrated in Figure 3 and is summarized in Algorithm 1. The framework of our adversarial meta-training monocular depth estimation is shown in Figure 4. During test, only the inner loop updates the parameters for a specific dataset and the outer loop is switched off. That is, we use initial parameters trained during training and continue to train a model when given a new dataset during test phase.

C. Cross-task depth consistency

In the proposed framework, an image is sent to different tasks for depth estimation during training and an example consisting of three different tasks is shown in Figure 5. As an image (Image \( I_A \) for example) is sent to different tasks (Task 1 and Task 2) for estimation, the predicted depth maps (\( d_A \) and

---

**Algorithm 1: Adversarial Domain-Adaptive Depth Estimation**

**Input:** \( T \): task set

**Input:** \( \alpha, \beta \): step size hyper-parameters

1. Initialize parameters \( \theta \) while not done
2. Sample batch of tasks \( \mathcal{T}_i \sim \mathcal{T} \) for each \( \mathcal{T}_i \) do
3. Sample \( K_s \) and \( K_q \) datapoints from \( \mathcal{T}_i \) as support set \( \mathcal{D}_{T_i}^s \) and query set \( \mathcal{D}_{T_i}^q \)
4. Evaluate \( \nabla_\theta \mathcal{L}_{T_i}^\theta(G_{\theta_G}) \) using \( \mathcal{D}_{T_i}^s \) and \( \mathcal{L}_{T_i}^G \) in Equation 5
5. Evaluate \( \nabla_\theta \mathcal{L}_{T_i}^\alpha(D_{\theta_D}) \) using \( \mathcal{D}_{T_i}^s \) and \( \mathcal{L}_{T_i}^D \) in Equation 5
6. Compute adapted parameters with gradient descent:
   \( \theta_G' = \theta_G - \alpha \nabla_\theta \mathcal{L}_{T_i}^\theta(G_{\theta_G}) \),
   \( \theta_D' = \theta_D - \beta \nabla_\theta \mathcal{L}_{T_i}^\alpha(D_{\theta_D}) \)
7. Update \( \theta_G \leftarrow \theta_G' - \beta \nabla_\theta \mathcal{L}_{T_i}^{\text{out}}(G_{\theta_G}) \),
   \( \theta_D \leftarrow \theta_D' - \alpha \nabla_\theta \mathcal{L}_{T_i}^{\text{out}}(D_{\theta_D}) \)
8. end
9. end

---

Fig. 4. The framework of our adversarial meta-learning monocular depth estimation method. Four datasets are used to construct six datasets pairs for meta-learning.
The depth maps estimated from the same scene (the depth maps in the dotted boxes with the same color) in different tasks are forced to be identical (connected with dotted lines).

\( d'_A \) can be different due to the differences in network parameters. Nevertheless, the depth map of a specific scene ought to be the same and the cross-task depth consistency constraint is proposed to compel the depth estimated in different tasks to be identical. The differences in network parameters of different tasks are caused by the individual parameter updates in inner loops with different training samples. The cross-task depth consistency is added in the outer loop update and denoted as \( \mathcal{L}_c \), which is calculated by:

\[
\mathcal{L}_c = |d_{i,1} - d'_{i,1}|, \tag{7}
\]

where \( d_{i,1} \) and \( d'_{i,1} \) are the estimated depth maps in different tasks. Since our \( \mathcal{L}_c \) compels the depth estimated in different tasks to be identical, it can smoothen the convergence curve and improve the performance of our method. Related experiments can be found in Section IV. Then, the optimization objectives of the generator and discriminator for outer loop meta-update can be updated as:

\[
\begin{align*}
\mathcal{L}_{T_i}^{G_{\theta_{G}}} = & \mathcal{L}_{T_i}^{u}(G_{\theta_{G}}(D_{G_{\theta_{D}}}(d_{i,1}))) + \mathcal{L}_c \\
\mathcal{L}_{T_i}^{D_{\theta_{D}}} = & \mathcal{L}_{D}^{u}(D_{\theta_{D}}(G_{\theta_{G}}(d_{i,1}))). \tag{8}
\end{align*}
\]

Equation 6 in Algorithm 1 should also be updated as Equation 8.

IV. EXPERIMENTS

A. Datasets

Four datasets, SYNTHIA [24], Cityscapes [54], Oxford RobotCar [55] and vKITTI [23], are used for training. To prove the effectiveness of our method, three different datasets: KITTI [56], nuScenes [57] and DrivingStereo [58], are used for test to prove that our method can generalize well to different new scenes.

CityScapes [54]: CityScapes is a well-known benchmark comprised of images collected in 50 cities in Germany, including a collection of 22,973 stereo pairs with the resolution of 1024 \( \times \) 2048.

Oxford RobotCar [55]: Oxford RobotCar is composed of image sequences collected in different weather conditions with the resolution of 960 \( \times \) 1280. In our experiments, we use the sequences from 2014-12-09-13-21-02 for training.

vKITTI [23]: Virtual KITTI (vKITTI) contains 21,260 image-depth pairs with the resolution of 375 \( \times \) 1242, which is generated by computer graphic engines. vKITTI tries to imitate the style of KITTI and generates images under different environmental conditions.

SYNTHIA [24]: SYNTHIA is a synthetic dataset containing images generated in the style of different seasons, scenarios, weathers, and illuminations.

KITTI [56]: KITTI dataset is a common benchmark in computer vision and contains 42,382 rectified stereo pairs in the resolution of 375 \( \times \) 1242. Though KITTI contains the ground truth of depth which is collected by Radar, we only use the images.

nuScenes [57]: NuSenses dataset comprises 1000 image sequences collected in street scenes with the resolution of 900 \( \times \) 1600. Different sensors are equipped, including 6 cameras deployed.

DrivingStereo [58]: DrivingStereo is a large-scale stereo dataset which contains more than 180,000 images collected from diverse driving scenarios. The resolution of the stereo pairs is 800 \( \times \) 1762 and all the images have their corresponding ground truth depth collected by LiDAR.

B. Implementation Details

Our framework is implemented in Pytorch [59] and optimized by Adam [60], which includes a generator and a discriminator. The generator adopts the depth estimation architecture in [15] as the basic model. The discriminator is the same as the discriminator in CycleGAN [44]. We use MAML [16] as the pipeline of meta-learning. During training, we select \( K_s = 4, K_q = 4, \alpha = 10^{-4}, \beta = 10^{-4} \) in Algorithm 1 for the training of the fast adapters in inner loop and the meta-adapter in outer loop. The batch size is set to be six, which means that all the tasks in task set are chosen for training in our domain-adaptive depth estimation algorithm. During test, we reload the network parameters obtained in training as the initial parameters.

Since the training images in different datasets have different resolutions and some of them contains some superfluous detail (e.g., car hood), we crop the images and resize them to the resolution of 640 \( \times \) 192 in training. During test, we resize the images in KITTI into the resolution of 640 \( \times \) 192, and resize the images in nuScenes and DrivingStereo into the resolution of 640 \( \times \) 320. As the proposed method aims to obtain appropriate initial parameters that can adapt rapidly to unseen domains, the quantity of network parameters remain the same.

C. Ablation Experiments

Ablation experiments are conducted on the KITTI dataset to analyze if the strategies used in our method are effective. The quantitative results are shown in Table I and the learning curves of different models are plotted in Figure 6. The Basic in Table I refers to the depth estimation model in [15]. We employ the MAML pipeline, our designed adversarial depth estimation task, the proposed cross-task depth consistency...
TABLE I
THE EVALUATION OF EACH COMPONENT USED IN THE PROPOSED ALGORITHM ON EIGEN SPLIT [5] AND THE PROPOSED STRATEGIES ARE PROVED TO BE EFFECTIVE.

| Method       | Abs Rel | Sq Rel | RMSE  | RMSE log | δ < 1.25 | δ < 1.25² | δ < 1.25³ |
|--------------|---------|--------|-------|----------|----------|-----------|-----------|
| Training (5 epochs) |         |        |       |          |          |           |           |
| Basic        | 0.183   | 2.908  | 6.366 | 0.252    | 0.781    | 0.928     | 0.968     |
| MAML         | 0.172   | 1.794  | 6.415 | 0.258    | 0.766    | 0.920     | 0.965     |
| MAML + AT    | 0.163   | 1.542  | 5.824 | 0.238    | 0.788    | 0.933     | 0.972     |
| Ours (MAML + AT + \(L_c\)) | 0.149   | 1.280  | 5.624 | 0.224    | 0.807    | 0.942     | 0.976     |
| Test (0.5 epoch) |         |        |       |          |          |           |           |
| Basic        | 0.121   | 0.930  | 4.917 | 0.196    | 0.864    | 0.958     | 0.981     |
| MAML         | 0.121   | 0.861  | 4.820 | 0.196    | 0.860    | 0.958     | 0.982     |
| MAML + AT    | 0.118   | 0.829  | 4.791 | 0.195    | 0.866    | 0.959     | 0.982     |
| Ours (MAML + AT + \(L_c\)) | 0.115   | 0.784  | 4.612 | 0.191    | 0.872    | 0.961     | 0.983     |

\(L_c\) is designed for the outer loop update in meta-learning, it can only be employed along with MAML. Our method use both AT and \(L_c\) in training.

During training, four datasets are used to train initial parameters for the test phase. We test the trained model directly on Eigen test-split [5] without test to evaluate if the training progress achieves convergence. The performances of different variants of our model are listed in Table I (Training (5 epochs)) and the corresponding training curves are plotted in Figure 6(a). From Figure 6(a), we can find that nearly all the modified models perform better than the basic one on the average. However, the performance of the model trained with MAML directly is instable and does not perform significantly better than the basic model, which may result from the meta-overfitting. Fortunately, the adversarial task designed helps to smoothen the training curves, which proves that our modified meta-learning algorithm helps to overcome the meta-overfitting. Further, our proposed \(L_c\) accelerates convergence and improves the performance as well.

Then, the test process is conducted on KITTI. We continue to train the model obtained in training on Eigen train-split [5] and then test the model on Eigen test-split. The results are shown in Table I (Test (0.5 epoch)) and Figure 6(b). As shown in Figure 6(b), the performances of different models in test are similar to those in training. The MAML training pipeline fails to improve the transferability of the basic model notably and our designed adversarial task helps to reduce the negative effect of training. Since our proposed cross-task depth consistency constraint \(L_c\) compels the predicted depth maps of the same scene in different tasks to be identical, the structure information can be learnt. Thus, \(L_c\) helps to smoothen the training curve and improve the performance significantly. According to Figure 6(b), both our designed adversarial task and the proposed \(L_c\) improve the performance and convergence rate. All the results demonstrate that our method can generalize well to KITTI, which is new to the training process.

D. Transferability of Our Depth Estimation Model

To demonstrate the generalizability of the proposed method, we conduct experiments on different auto-driving datasets: KITTI [56], nuScenes [57] and DrivingStereo [58]. As the backbone of our depth estimation network is [15], we compare...
our work with [15] and the quantitative comparison results are shown in Table II. The results prove that our method obtains satisfactory results on several new datasets with limited training iterations when compared to [15] trained with much more iterations. Since KITTI is a widely used dataset in depth estimation, we compare our methods with some other state-of-the-art methods [1], [5], [11], [15], [27], [28], [41], [61]–[64], as shown in Table III. As for nuScenes and DrivingStereo, few related works conduct depth estimation experiments on them, thus we only compare our work with [15].

For KITTI, we adopt the Eigen split [5] used in the compared works in test phase to evaluate the transferability of our method. We compare different methods with ours and the quantitative results are shown in Table III. It should be noticed that all the competing methods have access to KITTI dataset during the whole training. On the contrary, our model is trained on other datasets in training first and then trained on KITTI for limited steps, and obvious domain gaps exist between these datasets and KITTI. As shown in Table III, the compared methods are trained on KITTI for dozens of epochs, while our method is trained on KITTI for about 0.5 epoch during test. Actually, as shown in Figure 6(b), our method achieves a satisfactory performance after 500 iterations (1/6 epoch). The experimental results show that our method performs comparable to state-of-the-art works through a few updates with a relative low resolution, which

---

TABLE II

| Dataset     | Method                   | Resolution | Training time | Abs Rel | Sq Rel | RMSE | RMSE log | δ < 1.25 | δ < 1.25² | δ < 1.25³ |
|-------------|--------------------------|------------|---------------|---------|--------|------|----------|------------|------------|------------|
| KITTI       | Monodepth2 [15]          | 640×192    | 60K iterations | 0.115   | 0.903  | 4.863| 0.193    | 0.877     | 0.959     | 0.981     |
|             | Ours                     | 640×192    | 1.5K iterations | 0.115   | 0.784  | 4.612| 0.191    | 0.872     | 0.961     | 0.983     |
| nuScenes    | Monodepth2 [15]          | 640×320    | 40K iterations | 0.122   | 1.380  | 6.368| 0.227    | 0.852     | 0.940     | 0.968     |
|             | Ours                     | 640×320    | 3K iterations  | 0.118   | 1.306  | 6.620| 0.221    | 0.853     | 0.936     | 0.969     |
| DrivingStereo| Monodepth2 [15]         | 640×320    | 40K iterations | 0.1077  | 0.825  | 5.138| 0.114    | 0.910     | 0.980     | 0.993     |
|             | Ours                     | 640×320    | 3K iterations  | 0.093   | 0.801  | 5.058| 0.139    | 0.912     | 0.980     | 0.993     |

TABLE III

| Method                      | Supervision | Training time | Resolution | Abs Rel | Sq Rel | RMSE | RMSE log | δ < 1.25 | δ < 1.25² | δ < 1.25³ |
|-----------------------------|-------------|---------------|------------|---------|--------|------|----------|------------|------------|------------|
| Eigen et al. [5]            | Pose        | 50 epochs     | 612×184    | 0.190   | 1.515  | 7.156| 0.270    | 0.692     | 0.899     | 0.967     |
| Godard et al. [27]          |             |               | 512×256    | 0.148   | 1.344  | 5.927| 0.247    | 0.803     | 0.922     | 0.964     |
| SM-Learner [1]              | No          | 150K iterations | 128×416    | 0.183   | 1.595  | 6.709| 0.270    | 0.734     | 0.902     | 0.959     |
| CC [61]                     | No          | 100K iterations | 832×256    | 0.140   | 1.070  | 5.326| 0.217    | 0.826     | 0.941     | 0.975     |
| Alex et al. [62]            | No          | 50 epochs     | 512×256    | 0.133   | 1.126  | 5.515| 0.231    | 0.826     | 0.934     | 0.969     |
| Jian et al. [63]            | No          | 200K iterations | 832×256    | 0.137   | 1.089  | 5.439| 0.217    | 0.830     | 0.942     | 0.975     |
| GASDAS [41]                 | No          | 40 epochs     | 640×192    | 0.149   | 1.003  | 4.995| 0.227    | 0.824     | 0.941     | 0.973     |
| SynDeMo [11]                | No          | 280K iterations | 608×160    | 0.112   | 0.740  | 4.619| 0.187    | 0.863     | 0.958     | 0.983     |
| Monodepth2 [15]            | No          | 20 epochs     | 1024×320   | 0.115   | 0.882  | 4.701| 0.190    | 0.879     | 0.961     | 0.982     |
| Zhao et al. [64]            | No          | 50 epochs     | 832×256    | 0.113   | 0.704  | 4.581| 0.184    | 0.871     | 0.961     | 0.984     |
| Johnston et al. [28] (ResNet18) | No        | 20 epochs     | 640×192    | 0.111   | 0.941  | 4.817| 0.189    | 0.885     | 0.961     | 0.981     |
| Ours (ResNet18)             | No          | 0.5 epoch (1/6 epoch) | 640×192    | 0.115   | 0.764  | 4.612| 0.191    | 0.872     | 0.961     | 0.983     |

Fig. 7. Comparisons of the depth estimation results of different methods, including the original [15], [15] trained in the pipeline of basic MAML, and the proposed method.
proves the outstanding transferability of our methods. Figure 7 visualizes the depth maps estimated by different models, and we compare the model trained by Monodepth2 [15] in the basic MAML with our method to prove the effectiveness of our proposed strategies further. Our method specializes in capturing and reconstructing the detail of objects and provides sharper predictions, for example, the outline of trees. The segmentation of the background is also more distinct and we highlight some regions in Figure 7. As our algorithm is not sensitive to model structure, other methods, such as [28], [64] can be used as the baseline of our algorithm, which can improve the performance of depth estimation further.

For nuScenes, we use about 200,000 images for training and validation, and randomly choose 660 images for test. For DrivingStereo, 350,000 images are utilized for training and validation, and 547 images are selected stochastically to test. We compare our method with the state-of-the-art depth estimation method Monodepth2 [15] to prove the generalizability of our method on these two datasets. Similar to the experiments on KITTI dataset, [15] have access to the whole training set during training process. On the contrary, our method is trained for limited iterations, which means that our method only sees some part images in the training set for just one time. The quantitative and qualitative results are shown in Table II and Figure 8. The experimental results show that our method performs well on KITTI, nuScenes and DrivingStereo, which demonstrates that our method adapts well to new datasets rapidly.

V. CONCLUSIONS AND FUTURE WORKS

To deal with the issue of poor generalization, we propose an adversarial domain-adaptive algorithm for depth estimation, which can transfer well to new, unseen datasets in the presented work. We design an adversarial task for monocular depth estimation and the task is trained in the manner of meta-learning. The proposed adversarial task alleviates the issue of overfitting in meta-learning when the training tasks are limited. In addition, a cross-task depth consistency constraint is imposed for meta-update. Experiments show that our method generalizes well to new, unseen datasets, which demonstrates that our method learns to adapt for depth estimation gracefully. In the presented work, our method is trained and tested on autonomous driving datasets. In the future, we will dedicate to strengthen the transferability of our method further, for example, to enable our method to adapt well to not only outdoor scenes, but also indoor scenes.

REFERENCES

[1] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, “Unsupervised learning of depth and ego-motion from video,” in Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 1851–1858.
[2] Q. Sun, Y. Tang, C. Zhang, C. Zhao, F. Qian, and J. Kurths, “Unsupervised estimation of monocular depth and VO in dynamic environments via hybrid masks,” IEEE Transactions on Neural Networks and Learning Systems, 2021.
[3] Y. Yuan, L. Mou, and X. Lu, “Scene recognition by manifold regularized deep learning architecture,” IEEE Transactions on Neural Networks and Learning Systems, vol. 26, no. 10, pp. 2222–2233, 2015.
[4] Y. Tang, C. Zhao, J. Wang, C. Zhang, Q. Sun, W. Zheng, W. Du, F. Qian, and J. Kurths, “An overview of perception and decision-making in autonomous systems in the era of learning,” arXiv preprint arXiv:2001.02319, 2020.
[5] D. Eigen, C. Puhrsch, and B. Fergus, “Depth map prediction from a single image using a multi-scale deep network,” in Proceedings of the 2014 Advances in Neural Information Processing Systems (NIPS), 2014, pp. 2366–2374.
[6] C. Zhao, G. G. Yen, Q. Sun, C. Zhang, and Y. Tang, “Masked gan for unsupervised depth and pose prediction with scale consistency,” IEEE Transactions on Neural Networks and Learning Systems, 2020.
[7] C. Zhang, J. Wang, G. G. Yen, C. Zhao, Q. Sun, Y. Tang, F. Qian, and J. Kurths, “When autonomous systems meet accuracy and transferability through ai: A survey,” Patterns, vol. 1, no. 4, p. 100050, 2020.
[8] G. Csurka, “Domain adaptation for visual applications: A comprehensive survey,” arXiv preprint arXiv:1702.05374, 2017.
[9] L. Shao, F. Zhu, and X. Li, “Transfer learning for visual categorization: A survey,” IEEE Transactions on Neural Networks and Learning Systems, vol. 26, no. 5, pp. 1019–1034, 2014.
[10] G. Wilson and D. J. Cook, “A survey of unsupervised deep domain adaptation,” ACM Transactions on Intelligent Systems and Technology, vol. 11, no. 5, pp. 1–46, 2020.
[11] B. Bozorgtabar, M. S. Rad, D. Mahapatra, and J.-P. Thiran, “Syndemo: Synergistic deep feature alignment for joint learning of depth and ego-motion,” in Proceedings of the 2019 IEEE International Conference on Computer Vision (ICCV), 2019, pp. 4210–4219.
[12] M. B. Vankadari, S. Garg, A. Majumdar, S. Kumar, and A. Behera, “Unsupervised monocular depth estimation for night-time images using adversarial domain feature adaptation,” in Proceedings of the 2020 European Conference on Computer Vision (ECCV), 2020, pp. 443–459.
[13] J. Nath Kundu, P. Krishna Uppala, A. Pahuja, and R. Venkatesh Babu, “Adadepth: Unsupervised content congruent adaptation for depth estimation,” in Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 2656–2665.
[55] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, “1 year, 1000 km: The oxford robotcar dataset,” The International Journal of Robotics Research, vol. 36, no. 1, pp. 3–15, 2017.

[56] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the kitti vision benchmark suite,” in Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012, pp. 3354–3361.

[57] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, “nuscenes: A multimodal dataset for autonomous driving,” in Proceedings of the 2020 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 11 621–11 631.

[58] G. Yang, X. Song, C. Huang, Z. Deng, J. Shi, and B. Zhou, “Driving-stereo: A large-scale dataset for stereo matching in autonomous driving scenarios,” in Proceedings of the 2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 899–908.

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

[60] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” arXiv preprint arXiv:1412.6980, 2014.

[61] A. Ranjan, V. Jampani, L. Balles, K. Kim, D. Sun, J. Wulff, and M. J. Black, “Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation,” in Proceedings of the 2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 12 240–12 249.

[62] A. Wong and S. Soatto, “Bilateral cyclic constraint and adaptive regularization for unsupervised monocular depth prediction,” in Proceedings of the 2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 5644–5653.

[63] J.-W. Bian, Z. Li, N. Wang, H. Zhan, C. Shen, M.-M. Cheng, and I. Reid, “Unsupervised scale-consistent depth and ego-motion learning from monocular video,” arXiv preprint arXiv:1908.10553, 2019.

[64] W. Zhao, S. Liu, Y. Shu, and Y.-J. Liu, “Towards better generalization: Joint depth-pose learning without posenet,” in Proceedings of the 2020 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 9151–9161.