2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Distill-then-prune: An Efficient Compression Framework for Real-time Stereo Matching Network on Edge Devices

Baiyu pan[1,3], Jichao jiao[1,2], Jianxing pang[1], Jun cheng[3]

***Abstract*— In recent years, numerous real-time stereo matching methods have been introduced, but they often lack accuracy. These methods attempt to improve accuracy by introducing new modules or integrating traditional methods. However, the improvements are only modest. In this paper, we propose a novel strategy by incorporating knowledge distillation and model pruning to overcome the inherent trade-off between speed and accuracy. As a result, we obtained a model that maintains real-time performance while delivering high accuracy on edge devices. Our proposed method involves three key steps. Firstly, we review state-of-the-art methods and design our lightweight model by removing redundant modules from those efficient models through a comparison of their contributions. Next, we leverage the efficient model as the teacher to distill knowledge into the lightweight model. Finally, we systematically prune the lightweight model to obtain the final model. Through extensive experiments conducted on two widely-used benchmarks, Sceneflow and KITTI, we perform ablation studies to analyze the effectiveness of each module and present our state-of-the-art results.**

I. Introduction

Stereo matching estimates the depth information of a scene using a pair of rectified binocular images captured by cameras. The depth is determined by the disparity between the corresponding pixels in the two images. Stereo matching is important in many fields including robot navigation [1], drone control [2], [3] and autonomous driving [4], [5]. However, these applications share similar requirements, including implementation on edge devices, real-time processing, and high accuracy. In recent years, deep neural networks have achieved significant success in the field of stereo matching. In pursuit of accuracy and robustness, researchers have proposed architectures that are deeper and wider, even though this comes with a greater storage and computation time. For instance, the PSMnet [6], which is a most commonly adopted backbone, runs 1 fps on AGX. This high computation requirements pose an obstacle on the feasible application of stereo matching.

With the goal of achieving real-time processing on edge devices (GPU/NPU), many lightweight methods have been proposed [7]–[12]. Those methods can be roughly divided into two categories: multi-stage method [8], [10], [13] and model compression method [7], [9], [11]. The computational complexity of the network depends on two factors: the size of the feature map and the number of convolution kernels. Thus, multi-stage methods simultaneously decrease these two factors. For example, Wang *et al.* [10] employs four coarse-to-fine stages that dynamically up-sample the feature maps as needed. However, this merely achieves a balance in the trade-off between accuracy and latency, without gaining any extra efficiency. On the other hand, the main goal of model compression is to remove the redundant part of the model to achieve better efficiency. The Duggal *et al.* [7] proposed to progressively reduce the search space of disparity to accelerate the speed.

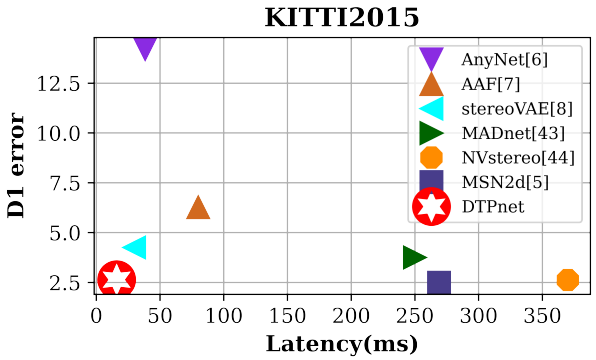

Fig. 1. Latency vs. D1 error on the KITTI 2015 [5] validation set. The unit of Latency is millisecond/frame. Both metrics are lower the better. As shown, our **DTPnet** achieves a good balance between accuracy and speed.

The typical backbone of end-to-end neural networks for stereo matching involves three main steps: feature extraction [14], [15], cost volume construction [5], [15], [16], and disparity regression module [17]. The cost volume is a 5-dimensional tensor with *batch, channel, depth, height, width*. In particular, the *depth* dimension is created by shifting the feature maps pixel-wise along the *width*. The extra *depth* not only increase the computation complexity and also requires 3D convolution for processing. Moreover, this introduces another challenge: the inference SDK provided by edge devices, like TensorRT [18], only provide limited support for 3D convolution operation. However, replacing the 3D convolution is challenging. Shamsafar *et al.* [9] has achieved promising result by compressing the *channel*. More importantly, only 2D convolution is used in their network.

We conducted extensive experiments and identified three key obstacles in achieving real-time stereo matching. First, the support for operations by the device's inference SDK, which directly determines the network's feasibility on the device but is often overlooked in previous real-time-oriented studies. Second, the heavy computation, as the computational

[1]The authors are with the UBTech Robotics Corp, Shenzhen, China {baiyu.pan,jichao.jiao,walton}@ubtrobot.com
[2]The author is with the Beijing University of Posts and Telecommunications, Beijing, China jiaojichao@bupt.edu.cn
[3]The authors are with the Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China jun.cheng@siat.ac.cn

complexity of the network directly affects the processing speed. Third, low accuracy, which stands on the opposite side of the scale with computation.

In this paper, we comprehensively analyze the commonly used modules and based on our analysis, we have designed an implementation-friendly lightweight network. Moreover, we propose the distill-then-prune (**DTP**) framework to further compress and improve the network. In DTP, we leverage the knowledge distillation and model pruning, both techniques that have proven to be efficient in model compression, to enhance performance. Overall, our main contributions are:

1) A implementation-friendly lightweight network is proposed.
2) We conduct a thorough investigation of knowledge distillation and propose a feasible scheme for stereo matching.
3) We combine model pruning with knowledge distillation to develop the DTP framework, which is compatible with any existing stereo matching methods.

Two benchmarks *Flyingthings3D* [16] and *KITTI* [5] are leveraged for evaluation. Extensive ablation studies are conducted to evaluate the components of our proposed networks. As shown in Fig. 1, we compare against current state-of-the-art methods and backbones to demonstrate our efficiency.

II. Related works

Stereo matching, also known as disparity estimation, was regarded as an optimization problem of locating the matching pixels between binocular images. The traditional methods [19]–[22] can be summarized as "search, compare, optimize", and this behavior can still be found in the CNN-based methods [6], [17], [23]–[25]. Based on [26], the commonly used backbone PSMnet [6] has established the training workflow and fundamental components of the end-to-end stereo matching network. Many methods have proposed effective modules or deeper network to improve the performance of the backbone, like atrous convolution [27], multi-scale regression [25], [28], volume fusion [29], [30].

Besides the aforementioned real-time-oriented methods, there are other multi-stage methods. HITnet [13] has adopted a structure similar to that of [10], but with the addition of geometric warping to obtain multi-resolution results. Xing *et al.* [8] has proposed adjust multi-branch module which combines depth-wise convolution to reduce the number of channels. The mobilestereonet's [9] main contribution is that they have proposed to reform the cost volume with convolution and use the 2D convolution only for the disparity regression. However, the final FLOPs and latency are still too larger and far from real-time.

Knowledge distillation [31]–[36] is an effective tool for improving the performance of the compressed neural network. Hinton *et al.* [31] proposed the vanilla logits-based knowledge distillation framework. In this framework, the student network learns from the logits of the teacher network, using a temperature hyperparameter to control the difficulty of learning. While the logits-based method is suitable for a wide range of tasks and models, its limitation lies in the fact that the student cannot learn the internal representations of the teacher. Zhang et al. [36] proposed distilling the network by minimizing the distance of extracted features between the teacher and student networks. The main advantage of feature-based knowledge distillation is that the student can learn more informative and robust representations from the teacher. However, feature-based knowledge is difficult to train [34] and requires a delicate training protocol to acquire meaningful knowledge. In this paper, we have compared both schemes, and the results show that only the logits-based approach [31] is feasible in the task of stereo matching.

Model pruning is an efficient tool for compress the neural network. The methods of model pruning have two main categories: non-structure pruning [37]–[39] and structure pruning [34], [40]–[42]. The non-structure methods are implementation-unfriendly for edge devices due to their sparse structure. The structure method prunes certain filters of a layer based on their importance. Luo *et al.* [41] rank and prune the filters based on their activation. In this paper, we follow the method [42] to prune the network and obtain a compact network.

III. Proposed lightweight network

We designed our network based on two goals, *lightweight* and *implementation-friendly*. To achieve *lightweight*, we compare the commonly used modules [6], [10], [14], [43] and remove the redundant ones. And to make it *implementation-friendly*, we have made three main changes: 1. We have replaced all the 3D convolution kernels with 2D convolution kernels. 2. The iterative cost volume construction is replaced with a channel-to-disparity module. 3. The commonly used trilinear interpolation after softmax layer is replaced by a bilinear interpolation in the inference stage.

A. Architecture

In our network, three modules are included, *Feature extraction network*, *Cost volume construction* and *Disparity regression*. The feature extraction network is a siamese network that shared the same weight, and process the input binocular image pairs parallel. After obtaining the features f_l and f_r from the left and right images, the cost volume construction is responsible for integrating them into a cost volume. Finally, the disparity regression module is responsible for predicting the disparity based on the cost volume.

TABLE I
THE PARAMETERS AND FLOPS OF EACH MODULE.

Moudle	Params$_{(M)}$	FLOPs$_{(G)}$
$Feature\ extraction$	0.10	0.72
$Cost\ volume\ construction$	0.03	0.11
$Disparity\ regression$	0.51	5.42
Total	0.64	6.25

In Table I, we present the details of the proposed lightweight network. We conducted extensive ablation studies to compare the commonly used architectures and number

of layers. By comparing the gains and losses in terms of parameters and accuracy, we removed the redundant modules and arrived at our final architecture. More details can be found in Section V.

Feature Extraction Network: Our network leverages the feature pyramid with two residual blocks to extract and concatenate multi-scale features. The outputs $\{f_l, f_r \in \mathbb{R} : |f| = (B \times C \times \frac{H}{4} \times \frac{W}{4})\}$ are the features of left and right image respectively.

Cost Volume Construction: Shamsafar *et al.* [9] proposed to iteratively construct the cost volume while simultaneously applying channel compression through convolutional layers. However, the total number of floating-point operations will increase cumulatively with each iteration. Therefore, we proposed a channel-to-disparity module where we directly compress the channel and guide the convolutions in learning the mapping from channel to disparity.

The module is composed by 3 layers of conventional layer:

$$y = \mathcal{F}(concat(f_l, f_r), \{W_i\}). \tag{1}$$

The f_l, f_r are concatenated first, and take as input of $\mathcal{F}$. The function $\mathcal{F} = W_i\sigma(W_{i-1}\sigma(...W_0(\cdot)...))$ in which σ denotes the activation layer and batch normalization layer, and $i = 3$.

Disparity Regression Module includes three parts, the stacked hourglass module [6], [43], upsampling layer and softmax layer.

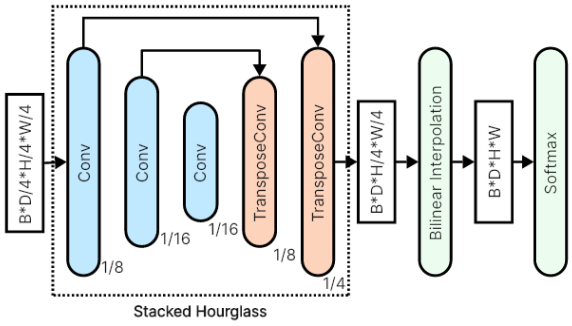

Fig. 2. Disparity regression module. The stacked hourglass module is composed by convolutions and transpose convolutions. The arrow line above denotes the skip connection.

As shown in Fig. 2, the encoder-decoder structure of the stacked hourglass provides a larger receptive field for feature maps, enabling it to capture contextual and semantic information. Moreover, as shown in Table I, the stacked hourglass module has the most parameters and FLOPs compared to others, but it also results in the most significant improvement in the outcomes. Further details can be found in Table II.

The upsampling layer is used to upsample the feature to $\{p \in \mathbb{R}^3 : |p| = (B \times d_{max} \times H \times W)\}$ where the hyperparameter d_{max} is set to 192.

After softmax, a soft-argmax (Equ. 2) is adopted for transform the probabilities to the predicted disparity result $\mathcal{D}$,

$$\mathcal{D} = \sum_{i=0}^{d_{max}} d_i * \frac{exp(p_i)}{exp(\sum_j^{d_{max}} p_j)}. \tag{2}$$

In the typical training workflow of disparity estimation network, the $smooth_{l1}$ loss is employed [6]. However, in our approach, we train the network using our proposed DTP framework, and the training details are provided in Section IV.

B. Adaptation for implementation

During implementation, we discovered that several operations, including 3D convolutions, slicing, iteratively applied convolutions, and trilinear interpolation, are not supported by the hardware. As a result, we implemented the aforementioned changes to address these issues while still achieving competitive results.

The 3D convolution is employed because of the 5D cost volume. In MobileStereoNet [9], they propose an iterative convolution approach to compress the channel of the cost volume and replace the 3D convolution with 2D convolution.

$$\begin{aligned} C(c,d,x,y) &= g_c(f_l(.,x,y), f_y(.,x-d,y)) \\ &\Downarrow \\ C(1,d,x,y) &= g_{c=1}(f_l(.,x,y), f_y(.,x-d,y)), \end{aligned} \tag{3}$$

where c represents the channel, and g_c denotes the convolution layers with an output channel c. During each iteration, MobileStereoNet [9] compressed the channel to 1 and then squeezed it afterward to obtain the 4D cost volume. Inspired by their work, we have designed our channel-to-disparity module, which completely eliminates the need for iterations and slice operations.

The remaining challenge is the trilinear interpolation. It is employed to upsample the D, H, W of the feature simultaneously. We have introduced an equivalent two-step operation, where a convolutional layer is utilized to map the channel from $D/4$ to D, followed by bilinear interpolation to upsample the feature map's H and W dimensions.

IV. Distill-Then-Prune framework

As the name of our proposed Distill-Then-Prune (DTP) framework indicates, it includes two main parts: *knowledge distillation* and *model pruning*. In Algorithm 1, we provide details of our framework.

Algorithm 1 Distill-Then-Prune(DTP) training framework

Input: Binocular images I_l, I_r; Max training step M; Learning rate γ; Parameter of teacher model θ_T; Pruning rate r and step E;

Output: Parameter of student model θ_S;

Initialize parameters of model: θ_S;
while $m < M$ **do** ▷ Knowledge Distillation
  $\theta_s \leftarrow \theta_s - \gamma \frac{\partial \mathcal{L}oss(\theta_s, \theta_t)}{\partial \theta_s}$
end while
while $e < E$ **do** ▷ Pruning
  $\theta_S^e \leftarrow prune(\theta_S^{e-1}, r)$
  while $m < M$ **do** ▷ Finetune by Distillation
    $\theta_S^e \leftarrow \theta_S^e - \gamma \frac{\partial \mathcal{L}oss_(\theta_S^e, \theta_T)}{\partial \theta_S^e}$
  end while
end while

It's worth emphasizing that our DTP framework can be applied to any stereo matching network, without being limited to training lightweight networks. After completing the DTP, $r * E$ percent of the model's parameters will be pruned.

A. Knowledge distillation

Our knowledge distillation has a teacher model and a student model. We train the student model to learn from the teacher's logits directly. Our loss function is defined as,

$$\mathcal{L}oss(p,q) = \sum_{i=0}^{d_{max}} |\frac{exp(p_i/t)}{exp(\sum_j^{d_{max}}(p_j/t))} - \frac{exp(q_i/t)}{exp(\sum_j^{d_{max}}(q_j/t))}|_1. \tag{4}$$

The temperature t is used to control the learning difficulties. The t grows from 0.5 to 1.0 as the epoch goes.

Additionally, we have conducted comparisons with other settings, such as combined training with ground truth, a comparison between KL loss and L1 loss. Further details are available in Section V.

B. Model pruning

We have adopted Depgraph [42] to conduct structural pruning. First, a dependency graph of the trained student network is constructed. The dependency graph groups the coupled convolutions between paired layers.

A simple L2-norm-based scheme is adopted to calculate the importance of convolution kernels, $w = \{w_1, w_2, ..., w_j\}$. The importance of kernels is defined as $I(w) = \{\|w_i\|_2 : w_i \in w\}$. Based on the ranking of $I(w)$, $r\%$ of the kernels is marked. Subsequently, adjacent layers with the same pruning scheme are grouped together. Then, the aggregated importance of a group $I(g)$ is calculated by,

$$I(g) = \sum_{w_i \in g} \|w_i\|_2. \tag{5}$$

Finally, we prune the entire group based on the ranking of their aggregated importance, $I(g)$.

V. Experimental results

In this section, we will refer our network as **DTPnet**. We conducted experiments on two datasets: **SceneFlow** [16] is a large scale of synthetic stereo dataset which contains more than 35k training pairs and 4.3k testing pairs with resolution 960x540. **KITTI** [4], [5] includes several driving-scene related subsets and challenges. We use *KITTI2015* [5] for train and test. It contains 200 pairs for training and 200 pairs for testing; with resolution 1242×375.

Metrics: End-point error (EPE) [16] is commonly used to optical flow evaluation. D1 error [5] are used for KITTI, D1 calculates the percentage of error pixels to the whole image. Pixel with EPE larger than 3 will be considered as error.

A. Implementation details

Our method was implemented using the PyTorch. Our hand-trained PSMnet [6] with EPE 0.69 is adopted as the teacher network. The training of the teacher is following the standard training protocol.

In DTP, we train the student network with AdamW [44], with $\beta_1 = 0.9, \beta_2 = 0.999$, learning rate $= 1e^{-3}$ and weight decay rate $= 1e^{-2}$. The training protocol is to train on SceneFlow for 20 epochs. Then, we finetune on KITTI for 300 epochs with $1e^{-3}$, then decay the weight to $1e^{-4}$ for another 300 epochs. We set prune rate $r = 0.1$ and step $E = 5$. At the end of the prune, 50% parameters of the model are pruned. At each iteration, we finetuning the pruned network for 5 epochs on Sceneflow, and 100 epochs on KITTI.

B. Ablation study

We have conducted comprehensive ablation studies to compare the architecture of our model and DTP framework. Firstly, we compare the commonly used modules of the backbone and illustrate the efficiency of our network. Secondly, we demonstrate the necessity of knowledge distillation and compared different settings.

1) ***Comparison of modules****:* As aforementioned, three modules are commonly used in disparity estimation [6], [9]. And these modules have various different settings internally, such as the repeated times of the layers. Therefore, we conducted the experiments from two perspectives: the first being the *inter-comparison*, in which we compared the influence on results of the internal layer settings of each module. The second is the *intra-comparison*, where we compared the influence on results between the different modules.

a. ***Inter-comparison***: In Table II, we compared three settings $\{1, 2, 3\}$. The setting 1 is proposed based on [9]. The setting 3 is our adopted network structure for **DTPnet**.

TABLE II
INTER-COMPARISON: THE PARAMETERS AND FLOPS OF MODULES

	Setting1		Setting2		Setting3 (DTP)	
Module	Params (M)	FLOPs (G)	Params (M)	FLOPs (G)	Params (M)	FLOPs (G)
$Feature_1$	0.01	0.72	0.01	0.72	0.01	0.72
$Feature_2$	0.16	3.17	0.09	0.50	0.09	0.50
$Feature_3$	0.10	2.21	0.10	2.21	/	/
$Feature_4$	0.11	2.13	0.11	2.13	/	/
$CostVolume$	0.13	7.16	0.13	7.16	0.03	0.11
$Hourglass_1$	0.51	2.67	0.51	2.67	0.51	2.67
$Hourglass_2$	0.51	2.67	/	/	/	/
$Hourglass_3$	0.51	2.67	/	/	/	/
Total	2.04	23.4	0.95	15.4	0.64	4.00
EPE	1.27		$1.48_{(+0.21)}$		$1.73_{(+0.46)}$	
Latency*	104 ms		$44_{(-60)}$ ms		**$16_{(-88)}$ ms**	

"/" denotes the layer is not adopted.
$*$ Latency is single frame inference time on Nvidia Jetson AGX.

The $Feature_*$ denotes the feature extraction modules, each includes 4 residual blocks [6]. The disparity regression module includes 3 identical stacked hourglass blocks,

TABLE III

THE QUANTITATIVE COMPARISON WITH OTHER LIGHTWEIGHT METHODS ON KITTI2015 [5]

	ALL-D1$_{bg}$	ALL-D1$_{fg}$	ALL-D1$_{all}$	NOC-D1$_{bg}$	NOC-D1$_{fg}$	NOC-D1$_{all}$	FLOPs(G)	Latency(ms)
AnyNet [10]	14.2 %	12.1 %	8.51 %	/	/	/	0.04	38.4(AGX)
AAF [11]	6.27 %	13.9 %	7.54 %	5.96 %	13.01 %	7.12 %	0.02	80(Tx2)
StereoNet [45]	4.30 %	7.45 %	4.83 %	/	/	/	0.36	1000+(Tx2)
StereoVAE [12]	4.25 %	10.1 %	5.23 %	3.88%	8.94 %	4.71 %	/	29.8(AGX)
MADnet [46]	3.75 %	9.20 %	4.66 %	3.45 %	8.41 %	4.27 %	3.82	250(Tx2)
MABnet [8]	3.04 %	8.07 %	3.88 %	2.80 %	7.28 %	3.54 %	0.04	/
DWARF [35]	3.20 %	3.94 %	3.33 %	2.95 %	3.66 %	3.07 %	19.6	1000+(Tx2)
DTPnet(ours)	2.64 %	6.47 %	3.28 %	2.46 %	5.61 %	2.98 %	**0.63**	**16.3**(AGX)
NVStereo [47]	2.62 %	5.69 %	3.13 %	**2.03** %	4.41 %	**2.42** %	3.10	370(Tx2)
AdaStereo [48]	2.59 %	5.55 %	3.08 %	2.39 %	5.06 %	2.83 %	9.37	/
MSN2d [9]	2.49 %	4.53 %	2.83 %	2.29 %	3.81 %	2.54 %	2.23	269(AGX)
DeepPruner [49]	2.32 %	**3.91** %	**2.59** %	2.13 %	**3.43** %	**2.35** %	103.6	/

* The latency is a single frame inference on Nvidia Tx2 or AGX

$Hourglass_*$. We reduced the repetition of each block and compared the results. And leading to the Setting 3 as the one that best suited our objectives.

b. ***Intra-comparison***: The SPP [50] module was adopted in [6], [9] to aggregate the cross spatial features from the feature extraction pyramid. In our network, we have reduced the pyramid to two layers. Therefore, the SPP module is redundant and does not improve the result.

TABLE IV

INTRA-COMPARISON: THE PARAMETERS AND FLOPS OF MODULES

Module	Params (M)	FLOPs (G)	EPE
$Feature$	0.10	1.22	1.76
$+SPP$ [50]	0.11(+0.01)	1.62(+0.40)	1.75(-0.01)
$CostVolume$ [9]	0.13	7.16	1.57
$CostVolume_{our}$	0.03(-0.1)	0.11(-7.05)	1.73(+0.16)
$Setting3$	0.63	6.30	1.73
$-hourglass$	0.13(-0.5)	3.63(-2.67)	5.04(+3.31)

The $CostVolume_{our}$ denotes our channel-to-disparity module. The iterative channel compression module [9] achieves a result that is 0.16 higher than ours. However, it also incurs an additional 7G FLOPs compared to ours. Therefore, we are satisfied with this trade-off. Additionally, we want to highlight that, based on our observations, the stacked hourglass module contributes the most to the overall results. Removing it results in a reduction of nearly 3G FLOPs, as shown in Table IV, but it also leads to a significant drop in accuracy.

2) ***Comparison of knowledge distillation****:* We compare the training protocol of knowledge distillation from two aspects, the supervision signal, and training loss.

a. ***Supervision signal***: In the vanilla knowledge distillation [31], the student network is learned from the teacher and ground truth simultaneously. This training protocol was designed for the object detection task, where the ground truth is sparse. However, in the task of disparity estimation, the ground truth is a dense disparity map, more similar to the softmax output. Therefore, we argue that supervision by the teacher alone is sufficient. Thus, we designed three pairs of supervision signals: 1. Ground truth only, 2. Ground truth and knowledge from the teacher, 3. Knowledge from the teacher only. As demonstrated in Table. V, the results confirm our hypothesis.

TABLE V

ABLATION STUDY ON SUPERVISION SIGNAL

Signal	EPE
Ground truth only	1.73
Knowledge + Ground truth	1.61
Knowledge only	**1.56**

b. ***Comparison of training loss***: *Kullback-Leibler* divergence is commonly used in knowledge distillation method [31], [33], [36]. However, based on our assumption that the supervision signal from the teacher's knowledge is similar to the ground truth, and considering that the L1-norm is a standard choice in training disparity estimation networks, we assert that the L1-norm is a viable alternative. As demonstrated in Table VI, models trained with the L1-norm loss have achieved superior results.

TABLE VI

COMPARISON OF TRAINING LOSS

	KL divergence	L1-norm
EPE	1.86	**1.78**

C. Quantitative comparison results

SceneFlow: For a fair comparison with other state-of-the-art complex methods, the Latency in Table. VII is tested on Titan XP. Despite our network being a lightweight method, we achieve competitive results compared to complex models on this dataset. Table. VII is arranged in descending order of end-point error (EPE), and notably, our **DPTnet** achieves the lowest EPE among all the methods.

KITTI2015: Table III provides a detailed comparison with other lightweight methods on the KITTI2015. Notably, our proposed **DTPnet** not only outperforms the other lightweight methods in terms of accuracy. We also achieve the lowest

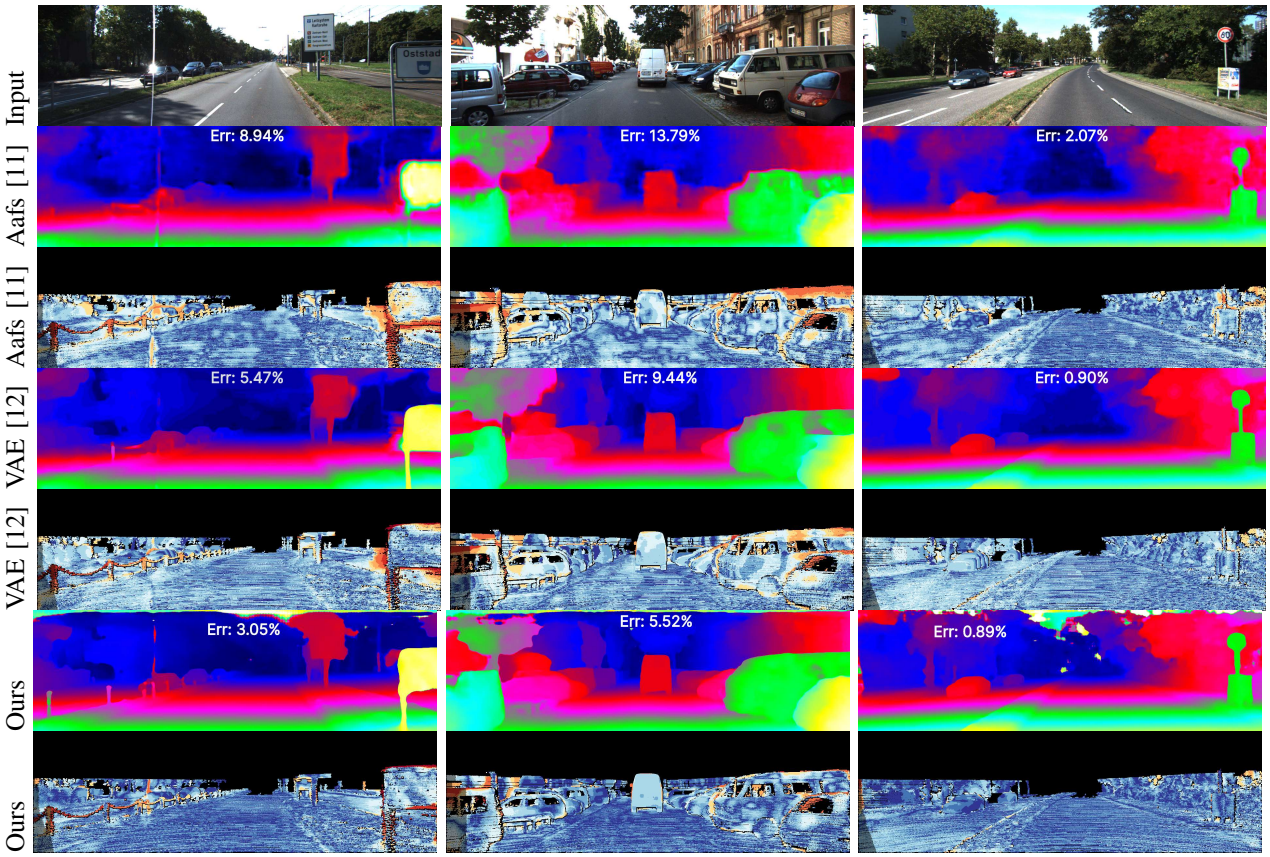

Fig. 3. Qualitative comparison (disparity results and error maps) of aafs [11], StereoVAE [12], and our **DTPnet**. Warmer colors in error maps indicate larger error.

TABLE VII
QUANTITATIVE COMPARISON ON SCENEFLOW

Model	EPE	Params	FLOPs	Latency*
PSMnet [6]	1.12	9.37	1083	450
MSN3d [9]	0.80	5.22	578.9	226
DeepPruner [49]	0.97	7.47	103.6	182
DWARF [35]	1.78	19.6	/	114
MSN2d [9]	1.12	2.28	128.8	107
MADnet [46]	1.66	**0.47**	15.6	65
MABnet [8]	1.63	0.04	380.9	50
AAFnet [11]	3.90	0.02	0.54	11
DTPnet(ours)	1.56	0.26	3.677	8

*Latency is the single frame inference time(ms) on *Titan XP*

latency. (ps. the same method's inference time on Tx2 and AGX is roughly 4x times) outperform the other methods in latency. The results highlight the effectiveness and competitiveness of our approach for disparity estimation on this dataset.

D. Qualitative results

In Figure 3, we present our **DTPnet** along with the results of [11], [12]. A common issue observed in many methods applied to the KITTI dataset is the presence of numerous mismatches caused by low-texture areas, such as the car window. Our **DTPnet** learns semantic information from the context and predicts the correct disparity of the whole objects.

VI. CONCLUSION

In this paper, we proposed a real-time lightweight stereo matching network and training framework. The proposed method includes: 1. An efficient and lightweight neural network for stereo matching, 2. The Distill-then-Prune (DTP) framework for model compression. By conducting comprehensive comparisons of accuracy and FLOPs with existing methods, we demonstrate that our **DTPnet** successfully achieves our objective of providing a real-time, high-accuracy network suitable for computational resource-limited platforms. In the future, we plan to further compress the network through quantization to suit more hardware platforms while maintaining a high accuracy.

VII. ACKNOWLEDGEMENT

This work was supported in part by the National Natural Science Foundation of China under Grant U2013601, and the Program of Guangdong Provincial Key Laboratory of Robot Localization and Navigation Technology, under Grant 2020B121202011 and Key-Area Research and Development Program of Guangdong Province, China, under Grant 2019B010154003.

REFERENCES

[1] Y. Bi, C. Li, and G. Wang, "An application of stereo matching algorithm based on transfer learning on robots in multiple scenarios," *Scientific Reports*, 2023.

[2] C. Cigla, R. Thakker, and L. Matthies, "Onboard stereo vision for drone pursuit or sense and avoid," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018, pp. 738–7388.

[3] X. Zhang, X. Cao, A. Yu, W. Yu, Z. Li, and Y. Quan, "Uavstereo: A multiple resolution dataset for stereo matching in uav scenarios," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 16, pp. 2942–2953, 2023.

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

[5] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *The Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[6] J.-R. Chang and Y.-S. Chen, "Pyramid stereo matching network," *arXiv:1803.08669 [cs]*, pp. 5410–5418, Mar. 2018, arXiv: 1803.08669.

[7] S. Duggal, S. Wang, W.-C. Ma, R. Hu, and R. Urtasun, "Deeppruner: Learning efficient stereo matching via differentiable patchmatch," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[8] J. Xing, Z. Qi, J. Dong, J. Cai, and H. Liu, "Mabnet: a lightweight stereo network based on multibranch adjustable bottleneck module," in *European Conference on Computer Vision*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds., vol. 12373, Springer. Cham: Springer International Publishing, 2020, pp. 340–356, series Title: Lecture Notes in Computer Science.

[9] F. Shamsafar, S. Woerz, R. Rahim, and A. Zell, "Mobilestereonet: Towards lightweight deep networks for stereo matching," *arXiv:2108.09770 [cs]*, Aug. 2021, arXiv: 2108.09770.

[10] Y. Wang, Z. Lai, G. Huang, B. H. Wang, L. van der Maaten, M. Campbell, and K. Q. Weinberger, "Anytime stereo image depth estimation on mobile devices," *arXiv preprint arXiv:1810.11408*, Mar. 2018, arXiv:1810.11408 [cs].

[11] J.-R. Chang, P.-C. Chang, and Y.-S. Chen, "Attention-aware feature aggregation for real-time stereo matching on edge devices," in *Proceedings of the Asian Conference on Computer Vision (ACCV)*, H. Ishikawa, C.-L. Liu, T. Pajdla, and J. Shi, Eds., vol. 12622. Cham: Springer International Publishing, November 2020, pp. 365–380, series Title: Lecture Notes in Computer Science.

[12] Q. Chang, X. Li, X. Xu, X. Liu, Y. Li, and J. Miyazaki, "Stereovae: A lightweight stereo-matching system using embedded gpus," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 1982–1988.

[13] V. Tankovich, C. Häne, Y. Zhang, A. Kowdle, S. Fanello, and S. Bouaziz, "Hitnet: Hierarchical iterative tile refinement network for real-time stereo matching," *arXiv:2007.12140 [cs]*, Apr. 2021, arXiv: 2007.12140.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[15] J. Zbontar and Y. LeCun, "Stereo matching by training a convolutional neural network to compare image patches," *CoRR*, vol. abs/1510.05970, 2015.

[16] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4040–4048, Jun. 2016, arXiv: 1512.02134.

[17] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, and A. Bry, "End-to-end learning of geometry and context for deep stereo regression," *arXiv:1703.04309 [cs]*, Mar. 2017, arXiv: 1703.04309.

[18] NVIDIA, "Tensorrt," https://github.com/NVIDIA/TensorRT, 2023.

[19] B. Pan, L. Zhang, Y. Zhu, and G. Luo, "An improved depth map estimation method based on motion detection," in *TENCON 2015-2015 IEEE Region 10 Conference*. Ieee, 2015, pp. 1–5.

[20] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "Patchmatch: A randomized correspondence algorithm for structural image editing," *ACM Transactions on Graphics-TOG*, vol. 28, no. 3, p. 24, 2009.

[21] H. Hirschmuller, "Stereo processing by semiglobal matching and mutual information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 328–341, Feb 2008.

[22] R. Song, H. Ko, and C.-C. J. Kuo, "Mcl-3d: A database for stereoscopic image quality assessment using 2d-image-plus-depth source," *Journal of Information Science and Engineering*, vol. 31, pp. 1593–1611, 2014.

[23] H. Park and K. M. Lee, "Look wider to match image patches with convolutional neural networks," *IEEE Signal Processing Letters*, 2016.

[24] G. Xu, J. Cheng, P. Guo, and X. Yang, "Attention concatenation volume for accurate and efficient stereo matching," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. arXiv, Jun. 2022, pp. 12 981–12 990, arXiv:2203.02146 [cs].

[25] B. Pan, L. Zhang, and H. Wang, "Multi-stage feature pyramid stereo network based disparity estimation approach for two to three-dimensional video conversion," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–1, 2020.

[26] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, and A. Bry, "End-to-end learning of geometry and context for deep stereo regression," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[27] X. Du, M. El-Khamy, and J. Lee, "Amnet: Deep atrous multiscale stereo disparity estimation networks," *arXiv preprint arXiv:1904.09099*, 2019.

[28] H. Xu and J. Zhang, "Aanet: Adaptive aggregation network for efficient stereo matching," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Seattle, WA, USA: Ieee, Jun. 2020, pp. 1956–1965.

[29] X. Guo, K. Yang, W. Yang, X. Wang, and H. Li, "Group-wise correlation stereo network," *arXiv:1903.04025 [cs]*, Mar. 2019, arXiv: 1903.04025.

[30] S. Chen, B. Li, W. Wang, H. Zhang, H. Li, and Z. Wang, "Cost affinity learning network for stereo matching," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Ieee, 2021, pp. 2120–2124.

[31] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015.

[32] I. Turc, M.-W. Chang, K. Lee, and K. Toutanova, "Well-read students learn better: On the importance of pre-training compact models," 2020.

[33] X. Liu, P. He, W. Chen, and J. Gao, "Improving multi-task deep neural networks via knowledge distillation for natural language understanding," *CoRR*, vol. abs/1904.09482, 2019.

[34] Y. Liu, J. Cao, B. Li, W. Hu, and S. Maybank, "Learning to explore distillability and sparsability: a joint framework for model compression," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 3, pp. 3378–3395, Mar. 2022, conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[35] F. Aleotti, M. Poggi, F. Tosi, and S. Mattoccia, "Learning end-to-end scene flow by distilling single tasks knowledge," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 10 435–10 442.

[36] L. Zhang and K. Ma, "Improve object detection with feature-based knowledge distillation: Towards accurate and efficient detectors," in *International Conference on Learning Representations*, Jan. 2021.

[37] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *International Conference on Learning Representations (ICLR)*, 2016.

[38] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 1135–1143.

[39] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," Mar. 2019, arXiv:1803.03635 [cs] version: 5.

[40] M. Lin, R. Ji, Y. Zhang, B. Zhang, Y. Wu, and Y. Tian, "Channel pruning via automatic structure search," *CoRR*, vol. abs/2001.08565, 2020. [Online]. Available: https://arxiv.org/abs/2001.08565

[41] J.-H. Luo, J. Wu, and W. Lin, "Thinet: A filter level pruning method for deep neural network compression," Jul. 2017, arXiv:1707.06342 [cs].

[42] G. Fang, X. Ma, M. Song, M. B. Mi, and X. Wang, "Depgraph: Towards any structural pruning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 16 091–16 101.

[43] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," *CoRR*, vol. abs/1603.06937, 2016.
[44] I. Loshchilov and F. Hutter, "Fixing weight decay regularization in adam," *CoRR*, vol. abs/1711.05101, 2017.
[45] S. Khamis, S. Fanello, C. Rhemann, A. Kowdle, J. Valentin, and S. Izadi, "Stereonet: Guided hierarchical refinement for real-time edge-aware depth prediction," *arXiv:1807.08865 [cs]*, Jul. 2018, arXiv: 1807.08865.
[46] R. Lan, L. Sun, Z. Liu, H. Lu, C. Pang, and X. Luo, "Madnet: A fast and lightweight network for single-image super resolution," *IEEE Transactions on Cybernetics*, vol. 51, no. 3, pp. 1443–1453, 2021.
[47] N. Smolyanskiy, A. Kamenev, and S. Birchfield, "On the importance of stereo for accurate depth estimation: An efficient semi-supervised deep neural network approach," *CoRR*, vol. abs/1803.09719, 2018.
[48] X. Song, G. Yang, X. Zhu, H. Zhou, Z. Wang, and J. Shi, "Adastereo: A simple and efficient approach for adaptive stereo matching," *CoRR*, vol. abs/2004.04627, 2020.
[49] S. Duggal, S. Wang, W.-C. Ma, R. Hu, and R. Urtasun, "Deeppruner: Learning efficient stereo matching via differentiable patchmatch," in *Iccv*, 2019.
[50] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 346–361.