2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Lightweight Monocular Depth Estimation via Token-Sharing Transformer

Dong-Jae Lee[1*], Jae Young Lee[1*], Hyunguk Shon[1], Eojindl Yi[1],
Yeong-Hun Park[2], Sung-Sik Cho[2], and Junmo Kim[1]

***Abstract*— Depth estimation is an important task in various robotics systems and applications. In mobile robotics systems, monocular depth estimation is desirable since a single RGB camera can be deployable at a low cost and compact size. Due to its significant and growing needs, many lightweight monocular depth estimation networks have been proposed for mobile robotics systems. While most lightweight monocular depth estimation methods have been developed using convolution neural networks, the Transformer has been gradually utilized in monocular depth estimation recently. However, massive parameters and large computational costs in the Transformer disturb the deployment to embedded devices. In this paper, we present a Token-Sharing Transformer (TST), an architecture using the Transformer for monocular depth estimation, optimized especially in embedded devices. The proposed TST utilizes global token sharing, which enables the model to obtain an accurate depth prediction with high throughput in embedded devices. Experimental results show that TST outperforms the existing lightweight monocular depth estimation methods. On the NYU Depth v2 dataset, TST can deliver depth maps up to 63.4 FPS in NVIDIA Jetson nano and 142.6 FPS in NVIDIA Jetson TX2, with lower errors than the existing methods. Furthermore, TST achieves real-time depth estimation of high-resolution images on Jetson TX2 with competitive results.**

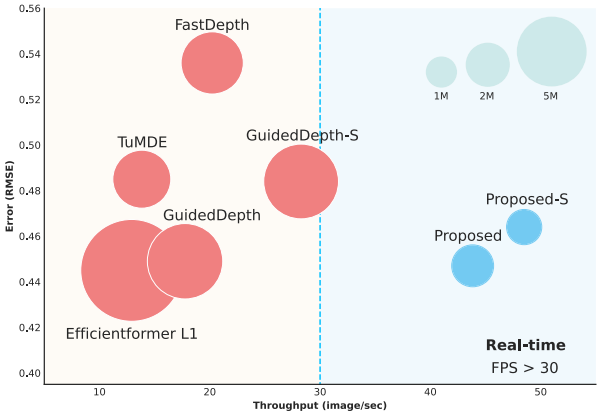


Fig. 1: Frame per second (FPS), root mean squared error (RMSE) versus model size on the NYU Depth V2 test set. The existing methods are marked as red and the proposed models (TST) are marked as blue. The throughput is measured with full-resolution 480×640 input on Jetson TX2.

I. INTRODUCTION

Depth information plays a fundamental and crucial role in various robotics systems and applications such as visual odometry, autonomous driving, robot localization, and visual perception. Several dedicated sensors such as light detection and ranging, time-of-flight, and structured light are widely used for capturing a depth map. However, such sensors are costly and bulky, thus making them unsuitable for mobile robotics platforms and edge devices. In contrast, monocular depth estimation, which estimates the depth map using a single RGB image, can be easily deployed at a low cost and compact size. Thus, lightweight deep networks for monocular depth estimation have been widely studied.

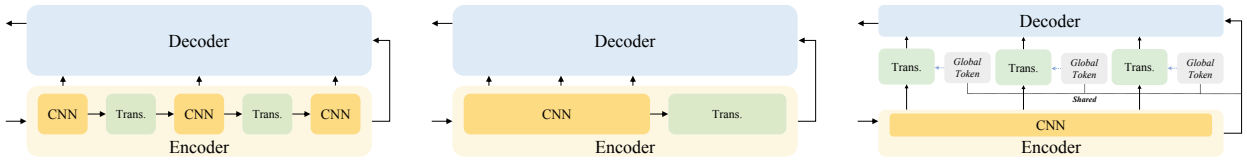
Recently, state-of-the-art monocular depth estimation methods [1], [2], [3] focus on utilizing the Vision Transformer (ViT) [4], which enables the model to learn the global information. However, the Transformers are usually slower than convolutional neural networks (CNN) due to their massive parameters and quadratic complexity of attention computation. This characteristic of the Transformer makes it not applicable directly in real-time tasks. Therefore, lightweight deep networks for monocular depth estimation are usually based on CNN [5], [6], [7], especially for mobile robotics and embedded devices.

To increase the throughput of the Transformer, various methods have been proposed, including redesigning the self-attention [8], [9], [10] or adopting a CNN-Transformer hybrid architecture [11], [12], [13], [14], [15].x The CNN-Transformer hybrid architecture can be categorized into two groups: hierarchy-focused and bottleneck-focused architectures. These architectures focused on the resolution of tokens[1] to reduce the complexity of self-attention. The hierarchy-focused architecture gradually decreased the resolution of tokens using the CNN between Transformer, while the bottleneck-focused architecture used only a low-resolution token by placing the Transformer at the end of CNN blocks. In view of performance, hierarchy-focused architecture shows better accuracy since they can learn the multi-level features containing the global information. However, bottleneck-focused architecture shows better throughput since they apply self-attention only in low-resolution tokens.

In this paper, we combine the design concepts of hierarchy-focused and bottleneck-focused architectures. The main idea in the proposed architecture is *global token shar-*

[1]The authors are with the Department of Electrical Engineering, Korea Advanced Institute of Science and Technology, South Korea. email: {jhtwosun, mcneato, hyounguk.shon, djwld93, junmo.kim}@kaist.ac.kr
[2]The authors are with MOBIS, South Korea. email: {yhpark0119, sscho}@mobis.co.kr
*The authors are equally contributed.

[1]For the readers who are not familiar to the Transformer, *tokens* referred to as features, especially for the feature used as the Transformer input.



(a) Hierarchy-focused architecture (b) Bottleneck-focused architecture (c) The proposed architecture

Fig. 2: Conceptual description of the existing architectures using the transformer and the proposed architecture.

ing. The hierarchy-focused architecture can learn the multi-level features containing global information. However, repetitive self-attention brings significant computational overhead. To overcome this limitation, the proposed architecture injects the global information into the multi-level features via the shared global token. Specifically, the global tokens and the multi-level features are extracted from a single lightweight CNN and fed to a cross-attention Transformer. Furthermore, like bottleneck-focused architecture, attention is applied to the low-resolution tokens by downsampling the multi-level features. Thus, via global token sharing, the model can learn rich global information like the hierarchy-focused architecture, with a high throughput like bottleneck-focused architecture.

To demonstrate the effectiveness of our approach, we experiment with NYU Depth V2 [16] and KITTI datasets [17], and further examine embedded systems, i.e., NVIDIA Jetson Nano and Jetson TX2, as well as Titan XP. As shown in Fig. 1, the proposed model achieves better results than the existing methods with high throughput.

II. RELATED WORK

A. Vision Transformer

Based on the success of the Vision Transformer (ViT) [4] on the image classification task, diverse architectures using the Transformer have been extensively studied for various applications. DeiT [18] used distillation learning to reduce the needs of a large dataset. Swin Transformer [8] introduced local-window-based self-attention, which leads to efficient computation of self-attention. Compact-T [19] and Segformer [12] combined CNN and Transformer to boost their performance and reduce computational costs. However, it is still incompatible with edge devices because of its high computation complexity.

Recently, several researchers have utilized the Transformer in resource-constrained edge devices. The existing methods using the Transformer for edge devices can be divided into two categories according to their main focus: 1) reducing the computational cost of the attention mechanism itself [20], [21] and 2) designing lightweight architecture [11], [12], [14], [13], [15]. To alleviate the computational costs, while the methods in the first category mathematically approximated the attention mechanism, those in the second category proposed hybrid architectures using both CNN and Transformer. Our method is grouped into the second category. We conceptually summarize the existing lightweight architectures using the Transformer grouped in the second category.

Fig. 2 shows the conceptual description of the existing lightweight architectures and the proposed model. The existing methods can be subdivided into two groups: 1) hierarchy-focused and 2) bottleneck-focused architecture. LeViT [11], Segformer [12], and MobileViT [14] are categorized into the first sub-group (See Fig. 2a). LeViT simply used the Transformers between convolution layers to reduce the number of tokens in each transformer block. MobileViT reduced computational costs based on MobileNetV2 [22] backbone with repeated CNN-Transformer blocks. Segformer expanded the usage of CNN layers to overlapped patch merging, which preserves the inductive bias of CNN while fully exploiting the power of the attention mechanism in the Transformer blocks.

Efficientformer [13] and Topformer [15] are categorized into the second sub-group (See Fig. 2). Both methods used MobileNetV2 as a backbone for their CNN blocks and utilized an output of CNN blocks as an input of the transformer blocks. EfficientFormer analyzed various methods using the transformer and proposed several strategies. They observed that the operations of repeated reshaping and permutations in tensors, which are used in the first sub-group, are one of the reasons making the model slow. Also, they observed that layer normalization is inferior to batch normalization to optimize the models. Based on the observations, Efficientformer proposed a model with CNN and transformer blocks that run at MobileNet speed. Topformer introduced token pyramid pooling, which used the concatenation of feature maps as an input of the transformer blocks, reinforcing the model's representation ability.

To further improve the aforementioned architectures, we propose Token-Sharing Transformer, which is described in Fig. 2c. The proposed model achieves the comparable accuracy of hierarchy-focused architecture and throughput of bottleneck-focused architecture. The details of the proposed model and design concept are described in Section III.

B. Monocular Depth Estimation

Monocular depth estimation aims to predict a depth map from a given RGB image. Eigen *et al*. [23] pioneered monocular depth estimation using CNN. Laina *et al*. [24] introduced a fully-convolutional network for monocular depth estimation. Based on Laina *et al*., various methods have been studied with pre-trained encoders in the classification task

as their backbone. These methods [3], [2], [1], [5], [6], [7] usually focused on designing a decoder network, whereas using the existing networks in the encoder.

1) Transformer-Based Monocular Depth Estimation: There have been several efforts to use Vision Transformer in Monocular depth estimation. DPT [3] utilized the global information and large receptive field from ViT encoder. For their decoder, they use CNN to make a dense prediction. DepthFormer [2] used two encoders, which consist of transformer and CNN, respectively. They introduced the hierarchical aggregation and heterogeneous interaction module to combine the features from the transformer and CNN encoders. GLPDepth [1] utilized Segformer [12] as their encoder and proposed a selective feature fusion module in their decoder to fuse local and global features. Since the decoder of GLPDepth is light-weighted and shows reasonable performance, our method utilizes the GLPDepth decoder.

2) Lightweight Monocular Depth Estimation: Since the above networks have heavy complexity for edge devices, several methods have been proposed in lightweight networks for depth estimation. FastDepth [5] utilized MobileNet [25] as their encoder and proposed a lightweight decoder for depth estimation. FastDepth further used network pruning [26] to reduce inference time for edge devices. Tu *et al.*'s method (TuMDE) used MobileNetV2 as their encoder and a more simplified decoder than FastDepth decoder. TuMDE also used a pruning algorithm with reinforcement learning. Recently, GuidedDepth [7] introduced a guided upsampling block for their lightweight decoder. These works utilized compile-time hardware-level optimization such as Tensor Virtual Machine [27], TensorRT [28], and quantization on half-precision (FP16) to reduce the computational complexity and inference time. Following these works, we also use the same hardware-level optimization techniques.

III. METHOD

A. Problem Formulation

Monocular depth estimation aims to learn images to a depth mapping function $f_\theta : \mathbb{R}^{H\times W\times 3} \rightarrow \mathbb{R}^{H\times W}$, where θ denotes model weights in f. Recently, various existing methods have utilized deep neural networks f as a mapping function composed of the encoder and decoder. To obtain a resulting depth map $\widehat{\mathbf{y}}$ from an input RGB image $\mathbf{x}$, it can be expressed as

$$\widehat{\mathbf{y}} = f_\theta(\mathbf{x}) = \psi(\xi(\phi(\mathbf{x}))), \tag{1}$$

where $\phi(\cdot)$, $\xi(\cdot)$, and $\psi(\cdot)$ signify the encoder, connection module, and decoder, respectively. The encoder $\phi(\cdot)$ extracts the feature maps from the input image and the decoder $\psi(\cdot)$ reconstructs the feature maps into a depth map. In general, for $\xi(\cdot)$, skip-connections with simple additions or concatenations have been widely used. In contrast, our Token-Sharing Transformer (TST) is placed in $\xi(\cdot)$, for efficient global information learning. The details of the proposed architecture are presented in the following sections.

B. Design Concept and Proposed Model

To design an architecture for monocular depth estimation on edge devices, we start by revisiting the existing lightweight architectures design concept: hierarchy-focused (Fig.2a) and bottleneck-focused architectures (Fig.2b).

In general, the Transformer based model learns the global information by self-attention. Specifically, query, key, and value are extracted from the token. Then, self-attention calculates the similarity between query and key, and recalculates each pixel feature with the weighted sum of the key according to the similarity. However, this brings significant computational overhead. To reduce the computational complexity, the hierarchy-focused architecture gradually reduces the resolutions of tokens, so that the model can learn the multi-level features containing global information. However, the effect of reducing the complexity is not so significant. On the other hand, the bottleneck-focused architecture reduces the resolution through CNN and applies self-attention only in low-resolution tokens to remove the computational complexity significantly. Thus, the hierarchy-focused architecture can achieve high performance, while the bottleneck-focused architecture can achieve high throughput. To achieve better performance and throughput simultaneously, our TST combines the two design concepts. Specifically, TST focuses on learning the multi-level features containing the global information, with proposed global token sharing.

In the proposed model, we extract the multi-level features from lightweight CNN. The multi-level features are used as the local tokens, and the high-level feature is used as a shared global token. Each Transformer block is composed of Multi-Head Attention (MHA) and Feed-Foward Network (FFN). In MHA, TST computes the cross-attention between the shared global token and each local token. Since the proposed model utilizes a lightweight CNN for low computational complexity, the low-level and mid-level features have insufficient global information. However, the cross-attention injects the global information into each local token, thus the model can learn the multi-level features containing the global information. Specifically, the query is extracted from the local tokens while the key and value are extracted from the shared global token. The output features are calculated by the weighted sum of keys according to the similarity between queries and keys. To further reduce the computational complexity, the local tokens are downsampled to the size of the global token before cross-attention. Then, the output local tokens are fed to FFN with residual mapping for the feature refinement. After the feature refinement, the outputs of FFN are upsampled to the original token size. With residual mapping, the multi-level feature maps containing global information are fed to the decoder to reconstruct the feature maps into the depth map.

C. Overall Architecture

Fig. 3 shows the overall architecture of the proposed model in detail. With the input image $I \in \mathbb{R}^{H\times W\times C}$, where H, W, and C indicate the height, width, and RGB channels of the image, we extract a set of multi-resolution feature maps

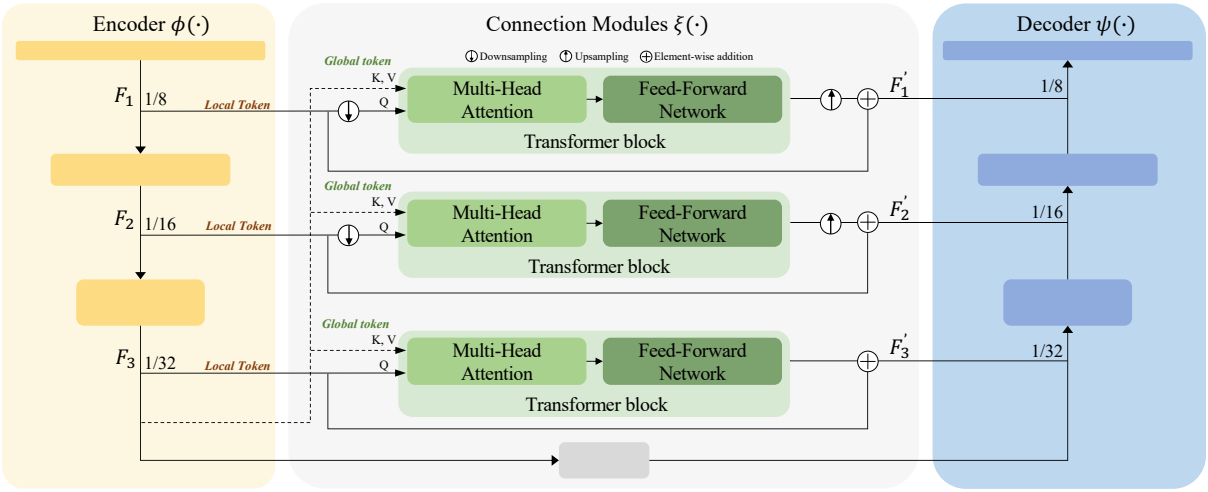


Fig. 3: Overall architecture of the proposed model (TST).

$\mathbf{F} = \{F_1, F_2, \cdots, F_N\}$ through the encoder $\phi(\cdot)$ where $F_n \in \mathbb{R}^{\frac{H}{2^{n+2}} \times \frac{W}{2^{n+2}} \times C_n}$. A set of feature maps $\mathbf{F}$ is fed to the connection layers. In the connection layers, $\mathbf{F}$ are firstly passed to average pooling layers, so that a set of features maps $\mathbf{F}^{\downarrow} = \{F_1^{\downarrow} F_2^{\downarrow}, \cdots F_N^{\downarrow}\}$, where $F_n^{\downarrow} \in \mathbb{R}^{\frac{H}{2^{N+2}} \times \frac{W}{2^{N+2}} \times C_n}$, are obtained. Note that each element in $\mathbf{F}^{\downarrow}$ has the same resolution scaled by 2^{N+2}. Each Transformer block in the connection module consists of the multi-head cross-attention (MHA) and feed-forward network (FFN). In MHA, we utilize each multi-resolution feature map $F_n^{\downarrow}$ as queries Q and feature map from the last layer of the encoder F_{N+1} as key K and value V. Then, passing through FFN, the output of the FFN is upsampled to the same resolution with each F_n and added to F_n. The outputs of the connection module $F_n^{'}$ are finally fed to the decoder. The feature map of the last layer in the encoder F_N is further fed to the first layer of the decoder passing through a single convolution block.

To customize the proposed model for various devices, we design the base model, coined Token-Sharing Transformer (denoted TST and its small version TST-S). For TST and TST-S, the dimension of the multi-level features $\{F_1, F_2, F_3\}$ are set to $\{64, 128, 160\}$ and $\{48, 96, 128\}$, respectively. For the loss function, we use scale-invariant log loss [23].

1) Token-Sharing Transformer: In each transformer block, following the observation of Efficientformer [13], we replace the Layer Normalization and GELU activation function [29] to Batch Normalization layer and ReLU6 [25] activation function, respectively. For MHA, we follow the design as the same as LeViT [11], Segformer [12], and Topformer [15]. The dimension of queries and keys are set to 16 and that of values is set to 16 and 32, respectively. The number of the heads N_{head} in each Transformer block is set 2, 4, and 5, which are proportional to the dimension of the input feature map C_n to further reduce the computational cost. The FFN consists of a single depth-wise convolution with kernel size 3 between two point-wise convolution layers. To reduce the computational cost, the dimension of the output feature map from FFN is set to the same as that of the input feature map (i.e., $F_n^{\downarrow}$).

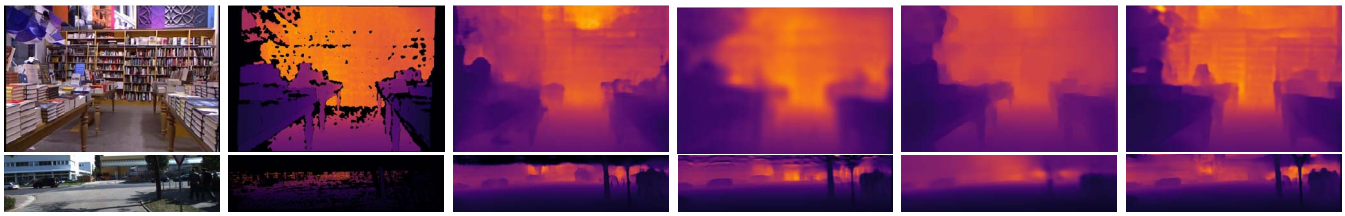
2) Encoder and Decoder: In the encoder, following MobileNetV2 [22] and TopFormer [15], we use ImageNet [30] pre-trained CNN consisting of Inverted Residual Block. For details, we follow the setting of TopFormer [15]. We do not aim to extract rich information or expect a large-receptive field through the encoder. Instead, we utilize the encoder as a simple feature extractor for tokens. Thus, the encoder is constructed with shallow layers, which is preferable for the embedded device. For the decoder, we utilize the decoder of GLPDepth [1]. Based on the decoder of GLPDepth, we adjust the resolution of feature maps properly for the proposed model.

IV. EXPERIMENTS

A. Experimental Setup

1) Datasets: **NYU Depth V2** [16] consists of 640×480 indoor images with corresponding depth map captured with Microsft Kinect camera. We train the proposed model with approximately 24K, 586×448 random cropped images and test on 654 images, using the split proposed by Eigen *et al.* [23]. **KITTI** [17] consists of approximately 24K images from outdoor driving scenes with spare depth maps captured by the LIDAR sensor. The input images have a resolution of 384×1224. For a fair comparison with GuidedDepth [7], we resize the RGB image into 384×1280. We use approximately 23K images and 697 images for the test, using the split proposed by Eigen *et al.* [23].

2) Hardware Platforms: Following the existing methods [5], [6], [7], we evaluate the model performance on embedded devices: NVIDIA Jetson Nano and NVIDIA Jetson TX2. Jetson Nano has a 128-core Maxwell architecture GPU with a Quad-core ARM A57 CPU and 4GB of RAM. Jetson TX2 has a 256-core NVIDIA Pascal architecture GPU with Dual-Core NVIDIA Denver CPU, Quad-Core ARM A57 MPCore and 8GB of RAM. All evaluation results are reported on 10W and 15W power mode for Jetson Nano and Jetson TX2.



(a) RGB (b) Ground Truth (c) FastDepth [5] (d) TuMDE [6] (e) GuidedDepth [7] (f) TST

Fig. 4: Qualitive comparison to the existing methods on NYU Depth V2 (1^{st} row) and KITTI (2^{nd} row) datasets.

TABLE I: Quantitative evaluation on NYU Depth V2 [16] and KITTI [17] datasets. ↑ and ↓ indicate the higher and the lower are better, respectively.

		Methods	#Param. (M)	MACs (G)	FPS			Metrics					
					Nano	TX2	Titan X	δ_1 ↑	δ_2 ↑	δ_3 ↑	AbsRel ↓	SqRel ↓	RMSE ↓
NYU	240 × 320	TuMDE [6]	3.44	1.03	18.93	47.64	104.66	0.802	0.953	0.989	0.148	0.106	0.510
		FastDepth [5]	3.96	1.20	29.89	72.43	134.37	0.778	0.943	0.987	0.163	0.121	0.576
		GuidedDepth [7]	5.80	2.63	24.41	55.60	70.31	0.823	0.962	0.992	0.138	0.095	0.501
		GuidedDepth-S [7]	5.70	1.52	38.02	91.60	71.17	0.787	0.958	0.992	0.146	0.099	0.503
		TST	1.80	0.67	57.15	127.85	96.33	0.815	0.962	0.990	0.143	0.102	0.487
		TST-S	1.27	0.56	63.43	142.58	96.91	0.802	0.957	0.990	0.148	0.104	0.499
	480 × 640	TuMDE [6]	3.44	3,96	5.05	13.84	58.40	0.806	0.964	0.993	0.143	0.096	0.485
		FastDepth [5]	3.96	4.69	8.13	20.22	87.90	0.781	0.944	0.987	0.157	0.117	0.536
		GuidedDepth [7]	5.82	10.60	6.51	17.76	56.67	0.840	0.968	0.994	0.129	0.088	0.449
		GuidedDepth-S [7]	5.72	6.10	10.65	28.28	62.61	0.817	0.961	0.991	0.140	0.095	0.484
		TST	1.80	2.65	17.03	43.83	93.21	0.841	0.968	0.992	0.132	0.088	0.447
		TST-S	1.27	2.21	18.98	48.50	94.59	0.828	0.965	0.992	0.136	0.091	0.464
KITTI	192 × 640	TuMDE [6]	3.44	1.57	13.12	33.95	94.08	0.813	0.954	0.987	0.148	0.890	5.282
		FastDepth [5]	3.96	1.82	19.39	47.72	116.20	0.808	0.945	0.981	0.150	0.890	5.321
		GuidedDepth [7]	5.80	4.24	15.97	40.82	67.94	0.857	0.965	0.990	0.119	0.771	4.456
		TST	1.80	1.06	41.25	96.12	94.27	0.866	0.974	0.995	0.114	0.649	4.406
		TST-S	1.27	0.89	46.48	102.30	95.08	0.852	0.972	0.994	0.135	0.740	4.621
	384 × 1280	TuMDE [6]	3.44	6.29	2.81	7.30	42.55	0.893	0.977	0.995	0.094	0.447	3.705
		FastDepth [5]	3.96	7.51	5.03	12.66	55.80	0.889	0.974	0.993	0.094	0.499	3.983
		GuidedDepth [7]	5.80	16.9	4.12	11.27	46.65	0.868	0.970	0.991	0.115	0.736	4.227
		TST	1.80	4.25	10.77	26.77	92.34	0.905	0.983	0.997	0.087	0.437	3.798
		TST-S	1.27	3.57	11.89	29.62	92.57	0.900	0.980	0.996	0.089	0.468	3.904

3) Implementation Details: We implement the proposed model on the PyTorch[31] framework. For training, we use ADAM optimizer [32] with customized Cosine annealing warm restarts learning rate scheduler[33]. Specifically, for optimizer, we use $\beta_1 = 0.9$, $\beta_2 = 0.99$, *learning rate* $= 0.0003$. For scheduler, we use $T_0 = 10, T_{mult} = 2, \gamma = 0.5$. We train for 100 epochs and apply the learning rate scheduler for additional 100 epochs. For the data augmentation, random horizontal flips, random brightness, contrast, gamma, hue, saturation, and value are used with 0.5 probabilities. Vertical CutDepth [1] is also utilized with 0.25 probability.

4) Evaluation Metrics and Protocol: We use evaluation metrics and protocol, following the existing methods [34], [2], [1], [5], [6], [3]. For quantitative evaluation, the metrics of δ_1, δ_2, δ_3, absolute relative difference (AbsRel), squared relative difference (SqRel), and root mean squared error (RMSE) are used. In the case of resolutions, following resolutions are used for evaluation: full-resolution 480×640, 384×1280 and half-resolution 240×480, 192×640 for NYU Depth V2 and KITTI, respectively. Eigen *et al.*'s cropping method [23] is used to evaluate for KITTI dataset. Note that for evaluation, the predicted depth maps are evaluated after up-sampled to the ground-truth depth map resolution, following GuidedDepth [7]. Since GuidedDepth uses customized inpainted dense ground-truth depth maps, we reproduce their results with the evaluation protocol of the other existing methods. For measurement of throughput in embedded devices, we convert the models to TensorRT engine [28] with FP16. For the measurement in Titan XP, PyTorch with CUDA utility is used without data type conversion. For throughput, FPS is measured with 200 samples.

B. Experimental Results

Table I shows the performance comparisons of the publicly available existing methods and the proposed model (Token-Sharing Transformer, shortly TST) on NYU Depth V2 and KITTI datasets. Note that TuMDE [6], FastDepth[5], GuidedDepth[7] are CNN-based methods. Since previous light-weight depth estimation methods are based on CNN architecture, comparsion with Transformer based architecture are presented in ablation study. "-S" indicates the small version of each model. Fig. 4 shows the qualitative evaluation of the existing method and TST.

1) NYU Depth V2: For the evaluation of the full resolution, TST outperforms the state-of-the-art models for monocular depth estimation in embedded devices in terms of parameter, computation, and inference speed. TST achieves better or comparable performance compared to GuidedDepth [7] base model with $3\times$ less parameter (1.9M vs. 5.82M),

$4\times$ less computation (2.21G vs. 10.60G), and $3\times$ faster FPS on both embedded devices (17.03 FPS vs. 6.51 FPS, 43.83 FPS vs. 17.76 FPS for Jetson Nano and TX2, respectively) in full-resolution prediction. There are slight performance drops when evaluating the half resolution because we aim to optimize TST to predict the high-quality depth map concerning the input RGB resolution. However, TST achieved comparable performance with much fewer parameters and computation, and faster inference times. Furthermore, our small model can achieve 63.43 FPS on Jetson Nano with better accuracy compared to GuidedDepth small in half-resolution prediction.

2) KITTI: For the evaluation of KITTI, we reproduce the existing methods since they use different resolutions [5], [6] or inpainted ground truth depth map [7]. Note that our evaluation is done without post-processing of the ground-truth depth map. Since the model weight of GuidedDepth-S [7] for KITTI is not publicly available, we can not reproduce their results. In this experiment, TST outperforms the existing methods in terms of performance, with far more small parameters, computation, and latency. On Jetson Nano, both TST and TST-S achieve over 40 FPS and over 10 FPS for the half and full resolutions, respectively. It is worth noting that TST achieves the highest FPS with the smallest parameter and computation without decreasing the overall accuracy.

C. Ablation Study

In this subsection, we validate our observation and the effectiveness of TST. We perform various ablation studies on the NYU Depth V2 dataset. All experiments are done in the image resolution 480×640 and with evaluation protocol described in Section IV-A.4. For a fair comparison, we use the same GLPDepth decoder [1] in TST for all experiments.

1) Experiments of the role of Transformer: TST utilizes the Transformer with effective global information injection to local features via cross-attention between multi-level local tokens and global sharing tokens. To investigate the effect of TST, we conduct an ablation experiment with TST without the Transformer and with a self-attention Transformer. The results are shown in Table II. It is obvious that the role of the Transformer is critical to the accuracy of TST. Furthermore, compared to using self-attention, TST with cross-attention can achieve better performance. The results firmly improve the effectiveness of the Transformer. Remaining problem is, how to implement the Transformer for embedded devices. We present the ablation studies on the design concepts of the architectures using the Transformer for the embedded devices in the following section.

2) Experiments on different architectures: Table III shows the experimental result using different encoders. The most light-weighted and second-light-weighted models of each encoder are used for the experiments. Segformer [12] is categorized into the hierarchy-focused architecture while Efficientformer [13] and Topformer [15] are categorized into the bottleneck-focused architecture. In addition, although both Efficientformer and Topformer are categorized as bottleneck-focused architecture, there is a subtle difference between them. Efficientformer L1 and L3 focused on increasing overall performance, thus resulting in the model with larger parameters and low throughput compared to Topformer. On the other hand, Topformer focused on improving the throughput, thus resulting in sacrificing the accuracy. In view of the accuracy of the depth prediction, the hierarchy-focused architecture (Segformer B0 and B1) shows better performance than the bottleneck-focused architecture (Topformer). Although Efficientformer L3 achieve higher accuracy compared to Segformer B1, they use $2\times$ larger parameters. In contrast, in view of the throughput, the bottleneck-focused architecture shows better performance than the hierarchy-focused architecture. The experimental results show that TST successfully combines the two different architectures, i.e., comparable accuracy with $5\times$ and $4\times$ throughput in Jetson Nano, compared to Segformer B0 and Effifientformer L1, respectively.

TABLE II: Ablation studies on the role of the Transformer.

Methods	Metrics			
	$\delta_1 \uparrow$	abs rel $\downarrow$	sq rel $\downarrow$	RMSE $\downarrow$
TST w/o Trans.	0.804	0.143	0.095	0.481
TST w/ self-attn.	0.828	0.134	0.090	0.454
TST	0.841	0.132	0.088	0.447

TABLE III: Experimental results on using the encoders of the existing methods.

Methods	#Param. (M)	MACs (G)	FPS		Metrics
			Nano	TX2	$\delta_1 \uparrow$
Seg. B0 [12]	3.41	6.24	3.47	8.26	0.845
Seg. B1 [12]	13.53	24.06	1.67	4.03	0.863
Eff. L1 [13]	10.61	15.88	4.65	12.92	0.842
Eff. L3 [13]	30.77	37.65	2.52	6.82	0.872
Top.-S [15]	3.33	2.96	16.53	41.82	0.819
Top. [15]	5.34	3.92	14.51	36.38	0.836
TST-S	1.27	2.21	18.98	48.50	0.828
TST	1.80	2.65	17.03	43.83	0.841

V. CONCLUSIONS

In this paper, we propose a lightweight monocular depth estimation method using a Token-Sharing Transformer. Token-Sharing Transformer uses multi-level features as local tokens like the hierarchy-focused architecture and uses a single shared token, which has low resolution like the bottleneck-focused architecture, as a global token. By Token-Sharing Transformer, the proposed model can achieve high throughput without performance drop, which is desirable on embedded devices. The experimental results show that the proposed model outperforms the existing monocular depth estimation methods in accuracy and throughput. The ablation studies show the effectiveness of the Token-Sharing Transformer. Also, compared to the experimental results using the existing methods as the encoder, the design of the proposed architecture is effective and suitable for the embedded devices.

REFERENCES

[1] D. Kim, W. Ga, P. Ahn, D. Joo, S. Chun, and J. Kim, "Global-local path networks for monocular depth estimation with vertical cutdepth," 2022. [Online]. Available: https://arxiv.org/abs/2201.07436

[2] Z. Li, Z. Chen, X. Liu, and J. Jiang, "Depthformer: Exploiting long-range correlation and local information for accurate monocular depth estimation," arXiv preprint arXiv:2203.14211, 2022.

[3] R. Ranftl, A. Bochkovskiy, and V. Koltun, "Vision transformers for dense prediction," in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 12 179–12 188.

[4] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in International Conference on Learning Representations, 2021. [Online]. Available: https://openreview.net/forum?id=YicbFdNTTy

[5] D. Wofk, F. Ma, T.-J. Yang, S. Karaman, and V. Sze, "Fastdepth: Fast monocular depth estimation on embedded systems," in 2019 International Conference on Robotics and Automation (ICRA), 2019, pp. 6101–6108.

[6] X. Tu, C. Xu, S. Liu, R. Li, G. Xie, J. Huang, and L. T. Yang, "Efficient monocular depth estimation for edge devices in internet of things," IEEE Transactions on Industrial Informatics, vol. 17, no. 4, pp. 2821–2832, 2021.

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

[8] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 10 012–10 022.

[9] S. Jaszczur, A. Chowdhery, A. Mohiuddin, L. Kaiser, W. Gajewski, H. Michalewski, and J. Kanerva, "Sparse is enough in scaling transformers," Advances in Neural Information Processing Systems, vol. 34, pp. 9895–9907, 2021.

[10] B. Chen, P. Li, B. Li, C. Li, L. Bai, C. Lin, M. Sun, J. Yan, and W. Ouyang, "Psvit: Better vision transformer via token pooling and attention sharing," arXiv preprint arXiv:2108.03428, 2021.

[11] B. Graham, A. El-Nouby, H. Touvron, P. Stock, A. Joulin, H. Jégou, and M. Douze, "Levit: a vision transformer in convnet's clothing for faster inference," in Proceedings of the IEEE/CVF international conference on computer vision, 2021, pp. 12 259–12 269.

[12] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, "Segformer: Simple and efficient design for semantic segmentation with transformers," Advances in Neural Information Processing Systems, vol. 34, pp. 12 077–12 090, 2021.

[13] Y. Li, G. Yuan, Y. Wen, E. Hu, G. Evangelidis, S. Tulyakov, Y. Wang, and J. Ren, "Efficientformer: Vision transformers at mobilenet speed," arXiv preprint arXiv:2206.01191, 2022.

[14] S. Mehta and M. Rastegari, "Mobilevit: Light-weight, general-purpose, and mobile-friendly vision transformer," in International Conference on Learning Representations, 2021.

[15] W. Zhang, Z. Huang, G. Luo, T. Chen, X. Wang, W. Liu, G. Yu, and C. Shen, "Topformer: Token pyramid transformer for mobile semantic segmentation," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 12 083–12 093.

[16] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in European conference on computer vision. Springer, 2012, pp. 746–760.

[17] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," The International Journal of Robotics Research, vol. 32, no. 11, pp. 1231–1237, 2013.

[18] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jegou, "Training data-efficient image transformers distillation through attention," in Proceedings of the 38th International Conference on Machine Learning, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 10 347–10 357. [Online]. Available: https://proceedings.mlr.press/v139/touvron21a.html

[19] A. Hassani, S. Walton, N. Shah, A. Abuduweili, J. Li, and H. Shi, "Escaping the big data paradigm with compact transformers," arXiv preprint arXiv:2104.05704, 2021.

[20] Y. Xiong, Z. Zeng, R. Chakraborty, M. Tan, G. Fung, Y. Li, and V. Singh, "Nyströmformer: A nyström-based algorithm for approximating self-attention," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, no. 16, 2021, pp. 14 138–14 148.

[21] S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma, "Linformer: Self-attention with linear complexity," 2020.

[22] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 4510–4520.

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

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

[25] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," arXiv preprint arXiv:1704.04861, 2017.

[26] T.-J. Yang, A. Howard, B. Chen, X. Zhang, A. Go, M. Sandler, V. Sze, and H. Adam, "Netadapt: Platform-aware neural network adaptation for mobile applications," in Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 285–300.

[27] T. Chen, T. Moreau, Z. Jiang, L. Zheng, E. Yan, H. Shen, M. Cowan, L. Wang, Y. Hu, L. Ceze, C. Guestrin, and A. Krishnamurthy, "TVM: An automated End-to-End optimizing compiler for deep learning," in 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18). Carlsbad, CA: USENIX Association, Oct. 2018, pp. 578–594. [Online]. Available: https://www.usenix.org/conference/osdi18/presentation/chen

[28] NVIDIA, "TensorRT," 2022. [Online]. Available: https://developer.nvidia.com/tensorrt

[29] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," arXiv preprint arXiv:1606.08415, 2016.

[30] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 248–255.

[31] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in Advances in Neural Information Processing Systems 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., 2019, pp. 8024–8035.

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

[33] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," arXiv preprint arXiv:1608.03983, 2016.

[34] Z. Li, X. Wang, X. Liu, and J. Jiang, "Binsformer: Revisiting adaptive bins for monocular depth estimation," arXiv preprint arXiv:2204.00987, 2022.