MTMamba: Enhancing Multi-task Dense Scene Understanding by Mamba-Based Decoders

Baijiong Lin[1,4], Weisen Jiang[2,3], Pengguang Chen[5], Yu Zhang[3], Shu Liu[5](✉), and Ying-Cong Chen[1,2,4](✉)

[1] The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China
bj.lin.email@gmail.com, yingcongchen@ust.hk
[2] The Hong Kong University of Science and Technology, Hong Kong, China
[3] Southern University of Science and Technology, Shenzhen, China
[4] HKUST(GZ) - SmartMore Joint Lab, Guangzhou, China
[5] SmartMore, Hong Kong, China
liushuhust@gmail.com

Abstract. Multi-task dense scene understanding, which learns a model for multiple dense prediction tasks, has a wide range of application scenarios. Modeling long-range dependency and enhancing cross-task interactions are crucial to multi-task dense prediction. In this paper, we propose MTMamba, a novel Mamba-based architecture for multi-task scene understanding. It contains two types of core blocks: self-task Mamba (STM) block and cross-task Mamba (CTM) block. STM handles long-range dependency by leveraging Mamba, while CTM explicitly models task interactions to facilitate information exchange across tasks. Experiments on NYUDv2 and PASCAL-Context datasets demonstrate the superior performance of MTMamba over Transformer-based and CNN-based methods. Notably, on the PASCAL-Context dataset, MTMamba achieves improvements of +2.08, +5.01, and +4.90 over the previous best methods in the tasks of semantic segmentation, human parsing, and object boundary detection, respectively. The code is available at https://github.com/EnVision-Research/MTMamba.

Keywords: multi-task learning · scene understanding · Mamba

1 Introduction

Multi-task dense scene understanding is an essential problem in computer vision [36] and has a variety of practical applications, such as autonomous driving [20,23], healthcare [19], and robotics [49]. It aims to train a model for simultaneously handling multiple dense prediction tasks, such as semantic segmentation, monocular depth estimation, and surface normal estimation.

The prevalent multi-task architecture follows an encoder-decoder framework, consisting of a task-shared encoder for feature extraction and task-specific

A. Leonardis et al. (Eds.): ECCV 2024, LNCS 15128, pp. 314–330, 2025.
https://doi.org/10.1007/978-3-031-72897-6_18

decoders for predictions [36]. This framework is very general and many variants have been proposed [37,42,43,47] to improve its performance in multi-task scene understanding. One promising approach is the decoder-focused method [36] with the aim of enhancing cross-task interaction in task-specific decoders through well-designed fusion modules. For example, derived from the convolutional neural network (CNN), PAD-Net [42] and MTI-Net [37] incorporate a multi-modal distillation module to promote information fusion between different tasks in the decoder and achieve better performance than the encoder-decoder framework. Since the convolution operation mainly focuses on local features [2], recent methods [43,47] propose Transformer-based decoders with attention-based fusion modules. These methods leverage the attention mechanism to capture global context information, resulting in better performance than CNN-based methods. Previous works demonstrate that *enhancing cross-task correlation* and *modeling long-range spatial relationships* are critical for multi-task dense prediction.

Very recently, Mamba [13], a new architecture derived from state space models (SSMs) [14,15], has shown better long-range dependencies modeling capacity and superior performance than Transformer models in various domains, including language modeling [12,13,39], graph reasoning [1,38], medical images analysis [30,41], and point cloud analysis [22,50]. However, all of these works focus on single-task learning, while how to adopt Mamba for multi-task training is still under investigation. Moreover, achieving cross-task correlation in Mamba remains unexplored, which is critical for multi-task scene understanding.

To fill these gaps, in this paper, we propose **MTMamba**, a novel multi-task architecture featuring a Mamba-based decoder and superior performance in multi-task scene understanding. The overall framework is shown in Fig. 1. MTMamba is a decoder-focused method with two types of core blocks: the self-task Mamba (STM) block and the cross-task Mamba (CTM) block, illustrated in Fig. 2. Specifically, STM, inspired by Mamba, can effectively capture global context information. CTM is designed to enhance each task's features by facilitating knowledge exchange across different tasks. Therefore, through the collaboration of STM and CTM blocks in the decoder, MTMamba not only enhances cross-task interaction but also effectively handles long-range dependency.

We evaluate MTMamba on two standard multi-task dense prediction benchmark datasets, namely NYUDv2 [35] and PASCAL-Context [6]. Quantitative results demonstrate that MTMamba largely outperforms both CNN-based and Transformer-based methods. Notably, on the PASCAL-Context dataset, MTMamba outperforms the previous best by +2.08, +5.01, and +4.90 in semantic segmentation, human parsing, and object boundary detection tasks, respectively. Qualitative studies show that MTMamba generates better visual results with more accurate details than state-of-the-art Transformer-based methods.

Our main contributions are summarized as follows:

- We propose MTMamba, a novel multi-task architecture for multi-task scene understanding. It contains a novel Mamba-based decoder, which effectively models long-range spatial relationships and achieves cross-task correlation;
- We design a novel CTM block to enhance cross-task interaction in multi-task dense prediction;

- Experiments on two benchmark datasets demonstrate the superiority of MTMamba on multi-task dense prediction over previous CNN-based and Transformer-based methods;
- Qualitative evaluations show that MTMamba captures discriminative features and generates precise predictions.

2 Related Works

2.1 Multi-task Learning

Multi-task learning (MTL) is a learning paradigm that aims to simultaneously learn multiple tasks in a single model [51]. Recent MTL research mainly focuses on multi-objective optimization [24–26,34,44–46,48] and network architecture design [37,42,43,47]. In multi-task dense scene understanding, most existing works focus on designing architecture [36], especially designing specific modules in the decoder to achieve better cross-task interaction. For example, based on CNN, Xu et al. [42] introduce PAD-Net, incorporating an effective multi-modal distillation module to promote information fusion between different tasks in the decoder. MTI-Net [37] is a complex multi-scale and multi-task CNN architecture with an information distillation across various feature scales. As the convolution operation mainly captures local features [2], recent approaches [43,47] utilize the attention mechanism to grasp global context and develop Transformer-based decoders for multi-task scene understanding. For instance, Ye & Xu [47] introduce InvPT, a Transformer-based multi-task architecture, employing an effective UP-Transformer block for multi-task feature interaction at different feature scales. MQTransformer [43] designs a cross-task query attention module to enable effective task association and information exchange in the decoder.

Previous works demonstrate long-range dependency modeling and enhancing cross-task correlation are critical for multi-task dense prediction. Unlike existing methods, we propose a novel multi-task architecture derived from Mamba to capture global information better and promote cross-task interaction.

2.2 State Space Models

State space models (SSMs) are a mathematical representation of dynamic systems, which models the input-output relationship through a hidden state. SSMs are general and have achieved great success in a wide variety of applications such as reinforcement learning [16], computational neuroscience [10], and linear dynamical systems [18]. Recently, SSMs are introduced as an alternative network architecture to model long-range dependency. Compared with CNN-based networks [17,21], which are designed for capturing local dependence, SSMs are more powerful for long sequences; Compared with Transformer-based networks [8,40], which require the quadratic complexity of the sequence length, SSMs are more computation- and memory-efficient.

Many different structures have been proposed recently to improve the expressivity and efficiency of SSMs. Gu et al. [14] propose structured state space

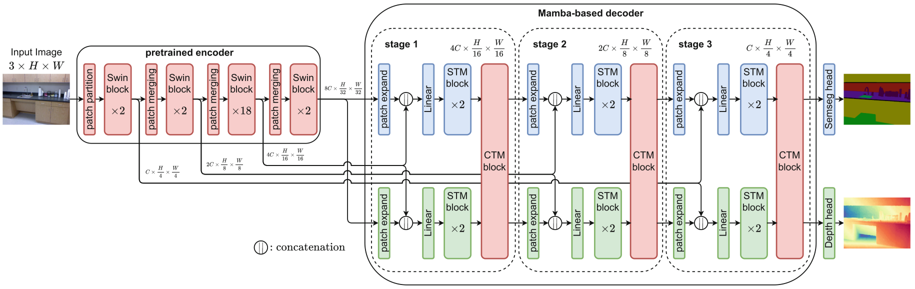
Input Image
$3 \times H \times W$
pretrained encoder
patch partition
Swin block ×2
patch merging
Swin block ×2
patch merging
Swin block ×18
patch merging
Swin block ×2
$8C \times \frac{H}{32} \times \frac{W}{32}$
$4C \times \frac{H}{16} \times \frac{W}{16}$
$2C \times \frac{H}{8} \times \frac{W}{8}$
$C \times \frac{H}{4} \times \frac{W}{4}$
: concatenation
Mamba-based decoder
stage 1
$4C \times \frac{H}{16} \times \frac{W}{16}$
stage 2
$2C \times \frac{H}{8} \times \frac{W}{8}$
stage 3
$C \times \frac{H}{4} \times \frac{W}{4}$
patch expand
Linear
STM block ×2
CTM block
Semseg head
Depth head

Fig. 1. Overview of the proposed MTMamba for multi-task dense scene understanding, illustrating with semantic segmentation (abbreviated as "Semseg") and depth estimation (abbreviated as "Depth") tasks. The red blocks are shared across all tasks, while the blue and green ones are task-specific. The pretrained encoder (Swin-Large Transformer is used) extracts multi-scale generic visual representations from the input RGB image. In the decoder, all task representations from task-specific STM blocks are fused and refined in the CTM block. Each task has its own head to generate the final predictions. Note that the structures of STM and CTM blocks (details in Fig. 2) in the decoder are Mamba-based (i.e., non-attention).

models (S4) to improve computational efficiency, where the state matrix is a sum of low-rank and normal matrices. Many follow-up works attempt to enhance the effectiveness of S4. For example, Fu et al. [11] design a new SSM layer H3 to fill the performance gap between SSMs and Transformers in language modeling. Mehta et al. [32] introduce a gated state space layer using gated units for improving expressivity. Recently, Gu & Dao [13] further propose Mamba with the core operation S6, an input-dependent selection mechanism of S4, which achieves linear scaling in sequence length and demonstrates superior performance over Transformers on various benchmarks. Mamba has been successfully applied in image classification [27,54], image segmentation [41], and graph prediction [38]. Different from them, which use Mamba in the single-task setting, we consider a more challenging multi-task setting and propose novel self-task and cross-task Mamba modules to capture intra-task and inter-task dependence.

3 Methodology

In this section, we first introduce the background knowledge of state space models and Mamba in Sect. 3.1. Then, we introduce the overall architecture of the proposed MTMamba in Sect. 3.2. Subsequently, we delve into a detailed exploration of each part in MTMamba, including the encoder in Sect. 3.3, the Mamba-based decoder in Sect. 3.4, and the output head in Sect. 3.5.

3.1 Preliminaries

SSMs [13–15], originated from the linear systems theory [5,18], map input sequence $x(t) \in \mathbb{R}$ to output sequence $y(t) \in \mathbb{R}$ though a hidden state $\mathbf{h} \in \mathbb{R}^N$

by a linear ordinary differential equation:

$$\mathbf{h}'(t) = \mathbf{A}\mathbf{h}(t) + \mathbf{B}x(t), \tag{1}$$

$$y(t) = \mathbf{C}^\top \mathbf{h}(t) + Dx(t), \tag{2}$$

where $\mathbf{A} \in \mathbb{R}^{N \times N}$ is the state matrix, $\mathbf{B} \in \mathbb{R}^N$ is the input matrix, $\mathbf{C} \in \mathbb{R}^N$ is the output matrix, and $D \in \mathbb{R}$ is the skip connection. Equation (1) defines the evolution of the hidden state $\mathbf{h}(t)$, while Eq. (2) determines the output is composed of a linear transformation of the hidden state $\mathbf{h}(t)$ and a skip connection from $x(t)$. For the remainder of this paper, D is omitted for explanation (i.e., $D = 0$).

Since the continuous-time system is not suitable for digital computers and real-world data, which are usually discrete, a discretization procedure is introduced to approximate it by a discrete-time one. Let $\Delta \in \mathbb{R}$ be a discrete-time step. Equations (1) and (2) are discretized as

$$\mathbf{h}_t = \bar{\mathbf{A}}\mathbf{h}_{t-1} + \bar{\mathbf{B}}x_t, \tag{3}$$

$$y_t = \bar{\mathbf{C}}^\top \mathbf{h}_t, \tag{4}$$

where $x_t = x(\Delta t)$, and

$$\bar{\mathbf{A}} = \exp(\Delta\mathbf{A}), \quad \bar{\mathbf{B}} = (\Delta\mathbf{A})^{-1}(\exp(\Delta\mathbf{A}) - \mathbf{I}) \cdot \Delta\mathbf{B} \approx \Delta\mathbf{B}, \quad \bar{\mathbf{C}} = \mathbf{C}. \tag{5}$$

In S4 [14], $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \Delta)$ are trainable parameters learned by gradient descent and do not explicitly depend on the input sequence, resulting in weak contextual information extraction. To overcome this, Mamba [13] proposes **S6**, which introduces an input-dependent selection mechanism to allow the system to select relevant information based on the input sequence. This is achieved by making $\mathbf{B}$, $\mathbf{C}$, and $\boldsymbol{\Delta}$ as functions of the input x_t. More formally, given an input sequence $\mathbf{x} \in \mathbb{R}^{B \times L \times C}$ where B is the batch size, L is the sequence length, and C is the feature dimension, the input-dependent parameters $(\mathbf{B}, \mathbf{C}, \Delta)$ are computed as

$$\mathbf{B} = \texttt{Linear}(\mathbf{x}) \in \mathbb{R}^{B \times L \times N}, \tag{6}$$

$$\mathbf{C} = \texttt{Linear}(\mathbf{x}) \in \mathbb{R}^{B \times L \times N}, \tag{7}$$

$$\Delta = \texttt{SoftPlus}(\tilde{\Delta} + \texttt{Linear}(\mathbf{x})) \in \mathbb{R}^{B \times L \times C}, \tag{8}$$

where $\tilde{\Delta} \in \mathbb{R}^{B \times L \times C}$ is a learnable parameter, $\texttt{SoftPlus}(\cdot)$ is the SoftPlus function, and $\texttt{Linear}(\cdot)$ is the linear layer. $\mathbf{A} \in \mathbb{R}^{L \times C}$ is a trainable parameter as in S4. After computing $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \Delta)$, $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ are discretized via Equation (5), then the output sequence $\mathbf{y} \in \mathbb{R}^{B \times L \times C}$ is computed by Equations (3) and (4).

3.2 Overall Architecture

An overview of MTMamba is illustrated in Fig. 1. It contains three components: an off-the-shelf encoder, a Mamba-based decoder, and task-specific heads. Specifically, the encoder is shared across all tasks and responsible for extracting multi-scale generic visual representations from the input image. The decoder consists of three stages. Each stage contains task-specific STM blocks to capture the long-range spatial relationship for each task and a shared CTM block to enhance each task's feature by exchanging knowledge across tasks. In the end, an output head is used to generate the final prediction for each task. We introduce the details of each part as follows.

3.3 Encoder

We take the Swin Transformer [28] as an example. Consider an input RGB image $\mathbf{x} \in \mathbb{R}^{3\times H\times W}$, where H and W are the height and width of the image, respectively. The encoder employs a patch-partition module to segment the input image into non-overlapping patches. Each patch is regarded as a token, and its feature representation is a concatenation of the raw RGB pixel values. In our experiment, we use a standard patch size of 4×4. Therefore, the feature dimension of each patch is $4 \times 4 \times 3 = 48$. After patch splitting, a linear layer is applied to project the raw token into a C-dimensional feature embedding. The patch tokens, after being transformed, sequentially traverse multiple Swin Transformer blocks and patch merging layers, which collaboratively produce hierarchical feature representations. Specifically, the patch merging layer [28] is used to 2× downsample the spatial dimensions (i.e., H and W) and 2× expand the feature dimension (i.e., C), while the Swin Transformer block focuses on learning and refining the feature representations. Formally, after forward passing the encoder, we obtain the output from four stages:

$$\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3, \mathbf{f}_4 = \texttt{encoder}(\mathbf{x}), \tag{9}$$

where $\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3$, and $\mathbf{f}_4$ have a size of $C \times \frac{H}{4} \times \frac{W}{4}, 2C \times \frac{H}{8} \times \frac{W}{8}, 4C \times \frac{H}{16} \times \frac{W}{16}$, and $8C \times \frac{H}{32} \times \frac{W}{32}$, respectively.

3.4 Mamba-Based Decoder

Extend SSMs to 2D Images. Different from 1D language sequences, 2D spatial information is crucial in vision tasks. Therefore, SSMs introduced in Sect. 3.1 cannot be directly applied in 2D images. Inspired by [27], we incorporate the 2D-selective-scan (SS2D) operation to address this problem. This method involves expanding image patches along four directions, generating four unique feature sequences. Then, each feature sequence is fed to an SSM (such as S6). Finally, the processed features are combined to construct the comprehensive 2D feature map. Formally, given the input feature $\mathbf{z}$, the output feature $\bar{\mathbf{z}}$ of SS2D is computed as

$$\mathbf{z}_v = \texttt{expand}(\mathbf{z}, v), \quad \text{for } v \in \{1,2,3,4\}, \tag{10}$$

$$\bar{\mathbf{z}}_v = \texttt{S6}(\mathbf{z}_v), \quad \text{for } v \in \{1,2,3,4\}, \tag{11}$$

$$\bar{\mathbf{z}} = \texttt{sum}(\bar{\mathbf{z}}_1, \bar{\mathbf{z}}_2, \bar{\mathbf{z}}_3, \bar{\mathbf{z}}_4), \tag{12}$$

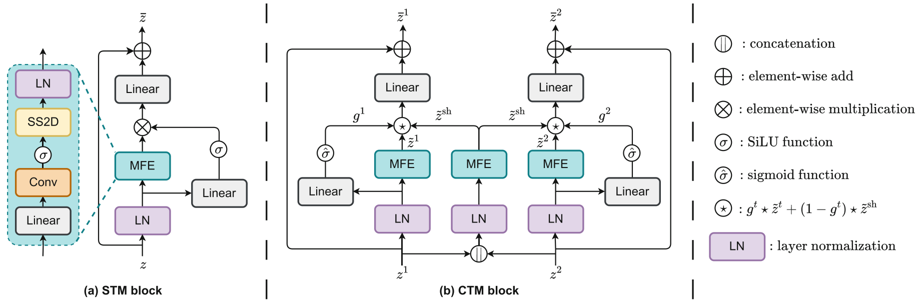

Fig. 2. **(a)** Illustration of the self-task Mamba (STM) block. Its core module is the Mamba-based feature extractor (MFE), where 1D S6 operation (introduced in Sect. 3.1) is extended on 2D images, namely SS2D. MFE is responsible for learning discriminant features and an input-dependent gate $\sigma(\texttt{Linear}(\texttt{LN}(\mathbf{z})))$ further refines the learned features. **(b)** Overview of the cross-task Mamba (CTM) block, illustrating with two tasks. Suppose T is the number of tasks ($T = 2$ in this illustration). The CTM block inputs T features, outputs T features, and contains $T+1$ MFE modules. One is used to generate a global feature $\tilde{\mathbf{z}}^{\text{sh}}$ and the other is to obtain the task-specific feature $\tilde{\mathbf{z}}^t$. Each output feature is the aggregation of task-specific feature $\tilde{\mathbf{z}}^t$ and global feature $\tilde{\mathbf{z}}^{\text{sh}}$ weighted by a task-specific gate $\mathbf{g}^t$. More details about these two blocks are provided in Sect. 3.4.

where $v \in \{1, 2, 3, 4\}$ is the four different scanning directions, $\texttt{expand}(\mathbf{z}, v)$ is to expand 2D feature map $\mathbf{z}$ along direction v, $\texttt{S6}(\cdot)$ is the S6 operation introduced in Sect. 3.1, and $\texttt{sum}(\cdot)$ is the element-wise add operation.

Mamba-Based Feature Extractor (MFE). We introduce a Mamba-based feature extractor to learn the representation of 2D images. It is a critical module in the proposed Mamba-based decoder. As shown in Fig. 2(a), motivated by [13], MFE consists of a linear layer used to expand the feature dimension by a controllable expansion factor α, a convolution layer with an activation function for extracting local features, an SS2D operation for modeling long-range dependency, and a layer normalization to normalize the learned features. More formally, given the input feature $\mathbf{z}$, the output $\bar{\mathbf{z}}$ of MFE is calculated as

$$\bar{\mathbf{z}} = (\texttt{LN} \circ \texttt{SS2D} \circ \sigma \circ \texttt{Conv} \circ \texttt{Linear})(\mathbf{z}), \tag{13}$$

where $\texttt{LN}(\cdot)$ is the layer normalization, $\sigma(\cdot)$ is the activation function and the SiLU function is used in our experiment, $\texttt{Conv}(\cdot)$ is the convolution operation.

Self-Task Mamba (STM) Block. We introduce a self-task Mamba block for learning task-specific features based on MFE, which is illustrated in Fig. 2(a). Inspired by [13], we use an input-dependent gate to adaptively select useful representations learned from MFE. After that, a linear layer is used to reduce the feature dimension expanded in MFE. Specifically, given the input feature $\mathbf{z}$, the computation in the STM block is as

$$\mathbf{z}_{\mathrm{LN}} = \mathtt{LN}(\mathbf{z}), \tag{14}$$

$$\tilde{\mathbf{z}} = \mathtt{MFE}(\mathbf{z}_{\mathrm{LN}}), \tag{15}$$

$$\mathbf{g} = \sigma(\mathtt{Linear}(\mathbf{z}_{\mathrm{LN}})), \tag{16}$$

$$\bar{\mathbf{z}} = \tilde{\mathbf{z}} \star \mathbf{g}, \tag{17}$$

$$\bar{\mathbf{z}} = \mathbf{z} + \mathtt{Linear}(\bar{\mathbf{z}}), \tag{18}$$

where $\star$ is the element-wise multiplication.

Cross-Task Mamba (CTM) Block. Although the STM block can effectively learn representations for each individual task, it lacks inter-task connections to share information which is crucial to the performance of MTL. To tackle this problem, we design a novel cross-task Mamba block (as shown in Fig. 2(b)) by modifying the STM block to achieve knowledge exchange across different tasks. Specifically, given all tasks' features $\{\mathbf{z}^t\}_{t=1}^T$ where T is the number of tasks, we first concatenate all task features and then pass it through an MFE to learn a global representation $\tilde{\mathbf{z}}^{\mathrm{sh}}$. Each task also learns its corresponding feature $\tilde{\mathbf{z}}^t$ via its own MFE. Then, we use an input-dependent gate to aggregate the task-specific representation $\tilde{\mathbf{z}}^t$ and global representation $\tilde{\mathbf{z}}^{\mathrm{sh}}$. Thus, each task adaptively fuses the global representation and its features. Formally, the forward process in the CTM block is as

$$\mathbf{z}_{\mathrm{LN}}^t = \mathtt{LN}(\mathbf{z}^t), \quad \text{for } t \in \{1, 2, \cdots, T\}, \tag{19}$$

$$\mathbf{z}_{\mathrm{LN}}^{\mathrm{sh}} = \mathtt{LN}(\mathtt{concat}(\mathbf{z}^1, \mathbf{z}^2, \cdots, \mathbf{z}^T)), \tag{20}$$

$$\tilde{\mathbf{z}}^t = \mathtt{MFE}(\mathbf{z}_{\mathrm{LN}}^t), \quad \text{for } t \in \{1, 2, \cdots, T\}, \tag{21}$$

$$\tilde{\mathbf{z}}^{\mathrm{sh}} = \mathtt{MFE}(\mathbf{z}_{\mathrm{LN}}^{\mathrm{sh}}), \tag{22}$$

$$\mathbf{g}^t = \hat{\sigma}(\mathtt{Linear}(\mathbf{z}_{\mathrm{LN}}^t)), \quad \text{for } t \in \{1, 2, \cdots, T\}, \tag{23}$$

$$\bar{\mathbf{z}}^t = \mathbf{g}^t \star \tilde{\mathbf{z}}^t + (\mathbf{1} - \mathbf{g}^t) \star \tilde{\mathbf{z}}^{\mathrm{sh}}, \quad \text{for } t \in \{1, 2, \cdots, T\}, \tag{24}$$

$$\bar{\mathbf{z}}^t = \mathbf{z}^t + \mathtt{Linear}(\bar{\mathbf{z}}^t), \quad \text{for } t \in \{1, 2, \cdots, T\}, \tag{25}$$

where $\mathtt{concat}(\cdot)$ is the concatenation operation, $\hat{\sigma}(\cdot)$ is the activation function and instead of SiLU used in STM block, we use the sigmoid function which is more suitable for generating the gating factors $\mathbf{g}^t$ used in Equation (24).

Stage Design. As shown in Fig. 1, the Mamba-based decoder contains three stages. Each stage has a similar design and comprises patch expand layers, STM blocks, and a CTM block. The patch expand layer is used to 2× upsample the feature resolution and 2× reduce the feature dimension. For each task, its feature will be expanded by a patch expand layer and then fused with multi-scale features from the encoder via skip connections to complement the loss of spatial information caused by down-sampling. Then, a linear layer is used to reduce the feature dimension and two STM blocks are responsible for learning task-specific representation. Finally, a CTM block is applied to enhance each task's feature by knowledge exchange across tasks. Except for the CTM block, other modules are task-specific. More formally, the forward process of i-stage ($i = 1, 2, 3$) is formulated as

$$\mathbf{r}_i^t = \texttt{PatchExpand}(\mathbf{z}_{i-1}^t) \quad \text{for } t \in \{1, 2, \cdots, T\}, \tag{26}$$

$$\mathbf{r}_i^t = \texttt{Linear}(\texttt{concat}(\mathbf{r}_i^t, \mathbf{f}_{4-i})), \quad \text{for } t \in \{1, 2, \cdots, T\}, \tag{27}$$

$$\mathbf{r}_i^t = \texttt{STM}(\texttt{STM}(\mathbf{r}_i^t)), \quad \text{for } t \in \{1, 2, \cdots, T\}, \tag{28}$$

$$\{\mathbf{z}_i^t\}_{t=1}^T = \texttt{CTM}(\{\mathbf{r}_i^t\}_{t=1}^T), \tag{29}$$

where $\mathbf{z}_0^t = \mathbf{f}_4$, $\texttt{PatchExpand}(\cdot)$ is the patch expand layer, $\texttt{STM}(\cdot)$ and $\texttt{CTM}(\cdot)$ are STM and CTM blocks, respectively.

3.5 Output Head

After obtaining each task's feature from the decoder, each task has its own output head to generate its final prediction. Inspired by [4], each output head contains a patch expand layer and a linear layer, which is lightweight. Specifically, given the t-th task feature $\mathbf{z}^t$ with the size of $C \times \frac{H}{4} \times \frac{W}{4}$ from the decoder, the patch expand layer performs 4× up-sampling to restore the resolution of the feature maps to the input resolution $H \times W$, and then the linear layer is used to output the final pixel-wise prediction.

4 Experiments

In this section, we conduct extensive experiments to demonstrate the effectiveness of the proposed MTMamba in multi-task dense scene understanding.

4.1 Experimental Setups

Datasets. Following [43,47], experiments are conducted on two benchmark datasets with multi-task labels: NYUDv2 [35] and PASCAL-Context [6]. The NYUDv2 dataset comprises a variety of indoor scenes, containing 795 and 654 RGB images for training and testing, respectively. It consists of four tasks: 40-class semantic segmentation (Semseg), monocular depth estimation (Depth), surface normal estimation (Normal), and object boundary detection (Boundary). The PASCAL-Context dataset, derived from the PASCAL dataset [9], includes both indoor and outdoor scenes and provides pixel-wise labels for tasks like semantic segmentation, human parsing (Parsing), and object boundary detection, with additional labels for surface normal estimation and saliency detection tasks generated by [31]. It contains 4,998 training images and 5,105 testing images.

Implementation Details. We use Swin-Large Transformer [28] pretrained on the ImageNet-22K dataset [7] as the encoder. All models are trained with a batch size of 8 for 50,000 iterations. The AdamW optimizer [29] is adopted with a

Table 1. Comparison with state-of-the-art methods on NYUDv2 (**left**) and PASCAL-Context (**right**) datasets. ↑ (↓) indicates that a higher (lower) result corresponds to better performance. The best and second best results are highlighted in **bold** and underline, respectively.

Method	Semseg	Depth	Normal	Boundary
	mIoU↑	RMSE↓	mErr↓	odsF↑
CNN-based decoder				
Cross-Stitch [33]	36.34	0.6290	20.88	76.38
PAP [52]	36.72	0.6178	20.82	76.42
PSD [53]	36.69	0.6246	20.87	76.42
PAD-Net [42]	36.61	0.6270	20.85	76.38
MTI-Net [37]	45.97	0.5365	20.27	77.86
ATRC [3]	46.33	0.5363	20.18	77.94
Transformer-based decoder				
InvPT [47]	53.56	0.5183	19.04	78.10
MQTransformer [43]	54.84	0.5325	19.67	78.20
Mamba-based decoder				
MTMamba (**ours**)	**55.82**	**0.5066**	**18.63**	**78.70**

Method	Semseg	Parsing	Saliency	Normal	Boundary
	mIoU↑	mIoU↑	maxF↑	mErr↓	odsF↑
CNN-based decoder					
ASTMT [31]	68.00	61.10	65.70	14.70	72.40
PAD-Net [42]	53.60	59.60	65.80	15.30	72.50
MTI-Net [37]	61.70	60.18	84.78	14.23	70.80
ATRC [3]	62.69	59.42	84.70	14.20	70.96
ATRC-ASPP [3]	63.60	60.23	83.91	14.30	70.86
ATRC-BMTAS [3]	67.67	62.93	82.29	14.24	72.42
Transformer-based decoder					
InvPT [47]	79.03	67.61	**84.81**	14.15	73.00
MQTransformer [43]	78.93	67.41	83.58	14.21	73.90
Mamba-based decoder					
MTMamba (**ours**)	**81.11**	**72.62**	84.14	**14.14**	**78.80**

learning rate of 10^{-4} and a weight decay of 10^{-5}. The polynomial learning rate scheduler is used in the training process. The expansion factor α in MFE is set to 2. Following [47], we resize the input images of NYUDv2 and PASCAL-Context as 448×576 and 512×512, respectively, and use the same data augmentation including random color jittering, random cropping, random scaling, and random horizontal flipping. We use ℓ_1 loss for depth estimation and surface normal estimation tasks and the cross-entropy loss for other tasks.

Evaluation Metrics. Following [47], we use mean intersection over union (mIoU) for semantic segmentation and human parsing tasks, root mean square error (RMSE) for monocular depth estimation task, mean error (mErr) for surface normal estimation task, maximal F-measure (maxF) for saliency detection task, and optimal-dataset-scale F-measure (odsF) for object boundary detection task. Besides, we use the average relative MTL performance Δ_m (defined in [36]) as the overall performance metric.

4.2 Comparison with State-of-the-Art Methods

We compare the proposed MTMamba method with two types of MTL methods: CNN-based methods, including Cross-Stitch [33], PAP [52], PSD [53], PAD-Net [42], MTI-Net [37], ATRC [3], and ASTMT [31], and Transformer-based methods, i.e., InvPT [47] and MQTransformer [43].

Table 1 shows the results on NYUDv2 and PASCAL-Context datasets. As can be seen, MTMamba shows superior performance in all four tasks on NYUDv2. For example, the performance of the semantic segmentation task has notably improved from the Transformer-based methods (i.e., InvPT and MQTransformer), increasing by +2.26 and +0.98, respectively, which demonstrates the effectiveness of MTMamba. The results on PASCAL-Context show the clear superiority of MTMamba. Especially, MTMamba significantly improves the previous best by +2.08, +5.01, and +4.90 in semantic segmentation, human parsing,

Table 2. Effectiveness of the STM and CTM blocks on NYUDv2. Swin-Large encoder is used in this experiment. “Multi-task” denotes an MTL model where each task only uses two standard Swin Transformer blocks in each decoder stage. “Single-task” is the single-task counterpart of “Multi-task”. ♦, ♠, ■, and ★ are different variants of MTMamba. ★ is the default configuration of MTMamba. ↑ (↓) indicates that a higher (lower) result corresponds to better performance.

Method	Each Decoder	Semseg	Depth	Normal	Boundary	Δ_m[%]	#Param	FLOPs
	Stage	mIoU↑	RMSE↓	mErr↓	odsF↑	↑	MB↓	GB↓
Single-task	2*Swin	54.32	0.5166	19.21	77.30	0.00	888.77	1074.79
Multi-task	2*Swin	53.72	0.5239	19.97	76.50	–1.87	303.18	466.35
MTMamba	♦1*STM	54.61	0.5059	19.00	77.40	+0.95	252.51	354.13
	♠2*STM	54.66	**0.4984**	18.81	78.20	+1.84	276.48	435.47
	■3*STM	54.75	0.5054	18.81	78.20	+1.55	300.45	516.82
	★2*STM+1*CTM	**55.82**	0.5066	**18.63**	**78.70**	**+2.38**	307.99	540.81

Table 3. Effectiveness of MFE module in MTMamba on NYUDv2. Swin-Large encoder is used. “W-MSA” is the window-based multi-head self-attention module in Swin Transformer [28]. “MFE” denotes all MFE modules in both STM and CTM blocks.

	Semseg	Depth	Normal	Boundary	Δ_m[%]	#Param	FLOPs
	mIoU↑	RMSE↓	mErr↓	odsF↑	↑	MB↓	GB↓
MFE→W-MSA	54.57	0.5109	19.95	76.60	–0.79	451.81	884.61
MTMamba	**55.82**	**0.5066**	**18.63**	**78.70**	**+2.38**	307.99	540.81

Table 4. Effectiveness of linear gate in MTMamba on NYUDv2. Swin-Large encoder is used. “W-MSA” is the window-based multi-head self-attention module in Swin Transformer [28]. “Linear” denotes all linear gates in both STM and CTM blocks.

	Semseg	Depth	Normal	Boundary	Δ_m[%]	#Param	FLOPs
	mIoU↑	RMSE↓	mErr↓	odsF↑	↑	MB↓	GB↓
Linear→W-MSA	55.01	**0.4990**	18.73	78.20	+2.08	345.33	659.29
MTMamba	**55.82**	0.5066	**18.63**	**78.70**	**+2.38**	307.99	540.81

and object boundary detection tasks, respectively, showing the effectiveness of MTMamba again. The qualitative comparison with InvPT on NYUDv2 and PASCAL-Context is shown in Figs. 4 and 5, showing that MTMmaba provides more precise predictions and details.

4.3 Model Analysis

Effectiveness of STM and CTM Blocks. The decoder of MTMamba contains two types of core blocks: STM and CTM blocks. We experiment on NYUDv2 to study the effectiveness of each type when the encoder is fixed as a Swin-Large

Table 5. Effectiveness of cross-task interaction in CTM block, i.e., Equation (24), on the NYUDv2 dataset. Swin-Large encoder is used in this experiment. "adaptive $\mathbf{g}^t$" means that $\mathbf{g}^t$ is computed by Equation (23). ↑ (↓) indicates that a higher (lower) result corresponds to better performance.

	Semseg	Depth	Normal	Boundary	Δ_m[%]
	mIoU↑	RMSE↓	mErr↓	odsF↑	↑
$\mathbf{g}^t = 0$	55.37	0.5087	18.76	78.30	+1.77
$\mathbf{g}^t = 1$	54.50	**0.4981**	18.83	78.20	+1.76
adaptive $\mathbf{g}^t$	**55.82**	0.5066	**18.63**	**78.70**	**+2.38**

Table 6. Performance of MTMamba with different scales of Swin Transformer encoder on the NYUDv2 dataset. ↑ (↓) indicates that a higher (lower) result corresponds to better performance.

Encoder	Semseg	Depth	Normal	Boundary
	mIoU↑	RMSE↓	mErr↓	odsF↑
Swin-Tiny	49.25	0.5299	19.74	76.90
Swin-Small	51.93	0.5246	19.45	77.80
Swin-Base	53.62	0.5126	19.28	77.70
Swin-Large	**55.82**	**0.5066**	**18.63**	**78.70**

Transformer. The results are shown in Table 2, where "Multi-task" represents an MTL model using two standard Swin Transformer blocks in each decoder stage for each task, and "Single-task" is the single-task counterpart of "Multi-task" (i.e., each task has a task-specific encoder-decoder). According to Table 2, the STM block achieves better performance and is more efficient than the Swin Transformer block (♠ vs. "Multi-task"), demonstrating that Mamba is more beneficial to multi-task dense prediction. Simply increasing the number of STM blocks from two to three fails to boost the performance. However, when the CTM is used, MTMamba has a significantly better performance in terms of Δ_m (★ vs. ♠/■). Moreover, the default configuration of MTMamba (i.e., ★) significantly outperforms "Single-task" on all tasks, showing that MTMamba is more powerful.

Effectiveness of MFE Module. As shown in Fig. 2, the MFE module is SSM-based and is the core of both STM and CTM blocks. We conduct an experiment by replacing all MFE modules in MTMamba with the attention module. As shown in Table 3, MFE is more effective and efficient than attention.

Effectiveness of Linear Gate. As shown in Fig. 2, in both STM and CTM blocks, we use an input-dependent gate to select useful representations adaptively from MFE modules. The linear layer is a simple but effective option for the gate function. We conduct an experiment by replacing all linear gates in MTMamba with the attention-based gate on the NYUDv2 dataset. As shown in Table 4, the linear gate (i.e., MTMamba) performs comparably to the attention gate in terms of Δ_m, while the linear gate is more efficient.

Effectiveness of Cross-task Interaction in CTM Block. The core of the CTM block is the cross-task interaction, i.e., Eq. (24), where we fuse task-specific representation $\tilde{\mathbf{z}}^t$ and shared representation $\tilde{\mathbf{z}}^{\text{sh}}$ via a task-specific gate $\mathbf{g}^t$. In this experiment, we study its effectiveness by comparing it with the cases of $\mathbf{g}^t = 0$ and $\mathbf{g}^t = 1$. The experiments are conducted with a Swin-Large Transformer encoder on NYUDv2. The results are shown in Table 5. As can be seen, using

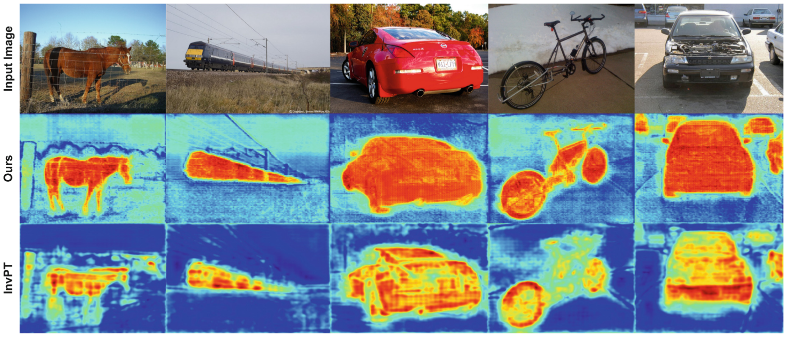

Fig. 3. Visualization of the final decoder feature of semantic segmentation. Compared with InvPT [47], our method generates more discriminative features.

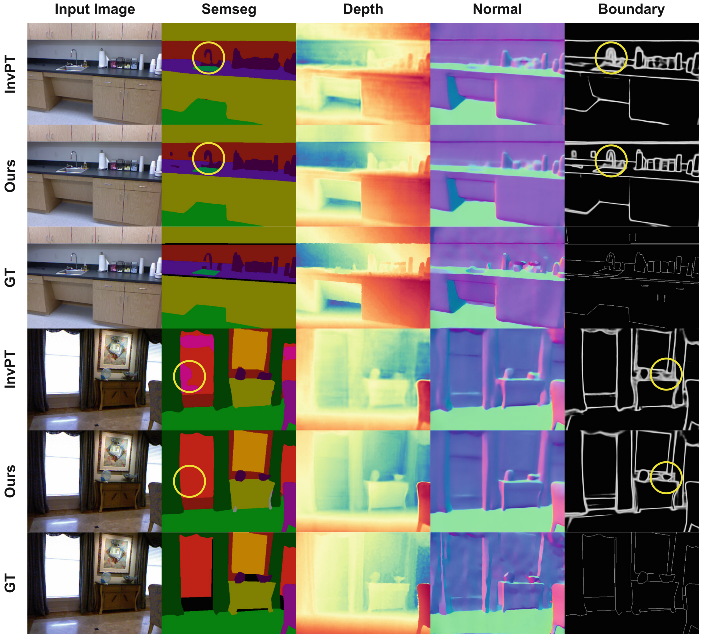

Fig. 4. Qualitative comparison with state-of-the-art method (i.e., InvPT [47]) on the NYUDv2 dataset. The proposed method generates better predictions with more accurate details as marked in yellow circles. Zoom in for more details.

a specific $\tilde{\mathbf{z}}^t$ (i.e., the case of $\mathbf{g}^t = 0$) or shared $\tilde{\mathbf{z}}^{\text{sh}}$ (i.e., the case of $\mathbf{g}^t = 1$) degrades the performance, demonstrating that the adaptive fusion is better.

Performance on Different Encoders. In this experiment, we investigate the performance of the proposed MTMamba with different scales of Swin Transformer encoder on the NYUDv2 dataset. The results are shown in Table 6. As can be seen, as the model capacity increases, all the tasks perform better accordingly.

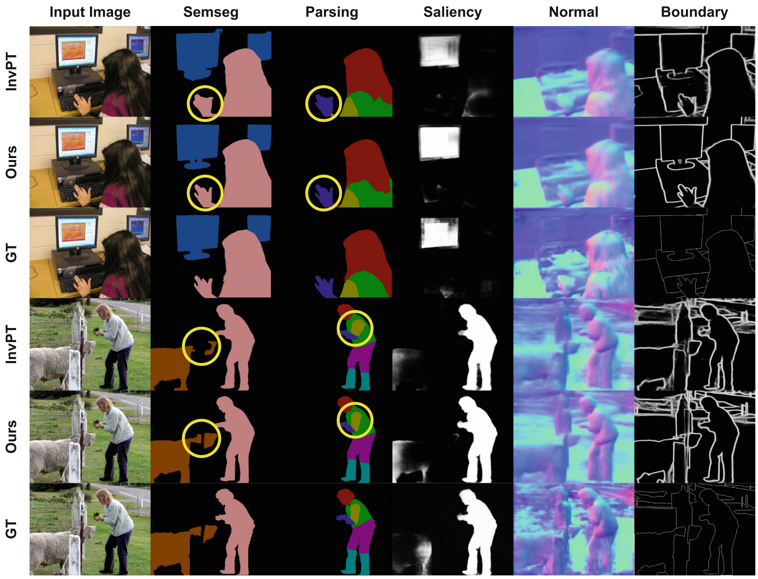

Fig. 5. Qualitative comparison with state-of-the-art method (i.e., InvPT [47]) on the PASCAL-Context dataset. The proposed method generates better predictions with more accurate details as marked in yellow circles. Zoom in for more details.

4.4 Qualitative Evaluations

Visualization of Learned Features. Fig. 3 shows the comparison of the final decoder feature between MTMamba and Transformer-based method InvPT [47] in the semantic segmentation task. As can be seen, our method highly activates the regions with contextual and semantic information, which means it captures more discriminative features, resulting in better segmentation performance.

Visualization of Predictions. We conduct qualitative studies by comparing the output predictions from our proposed MTMamba against the state-of-the-art Transformer-based method, InvPT [47]. Figures 4 and 5 show the qualitative results on the NYUDv2 and PASCAL-Context datasets, respectively. As can be seen, our method has better visual results than InvPT in all tasks. For example, as highlighted with yellow circles in Fig. 4, MTMamba generates more accurate results with better alignments for the semantic segmentation task and clearer object boundaries for the object boundary detection task. Figure 5 demonstrates that MTMamba produces better predictions with more accurate details (like the fingers as highlighted) for both semantic segmentation and human parsing tasks and more distinct boundaries for the object boundary detection task. Hence, both qualitative study (Figs. 4 and 5) and quantitative study (Table 1) show the superior performance of the proposed MTMamba method.

5 Conclusion

In this paper, we propose MTMamba, a novel multi-task architecture with a Mamba-based decoder for multi-task dense scene understanding. With two core blocks (STM and CTM blocks), MTMamba can effectively model long-range dependency and achieve cross-task interaction. Experiments on two benchmark datasets demonstrate that the proposed MTMamba achieves better performance than previous CNN-based and Transformer-based methods.

Acknowledgements. This work is supported by Guangzhou-HKUST(GZ) Joint Funding Scheme (No. 2024A03J0241).

References

1. Behrouz, A., Hashemi, F.: Graph Mamba: towards learning on graphs with state space models. arXiv preprint arXiv:2402.08678 (2024)
2. Bello, I., Zoph, B., Vaswani, A., Shlens, J., Le, Q.V.: Attention augmented convolutional networks. In: IEEE/CVF International Conference on Computer Vision (2019)
3. Brüggemann, D., Kanakis, M., Obukhov, A., Georgoulis, S., Van Gool, L.: Exploring relational context for multi-task dense prediction. In: IEEE/CVF International Conference on Computer Vision (2021)
4. Cao, H., et al.: Swin-unet: unet-like pure transformer for medical image segmentation. In: European Conference on Computer Vision, pp. 205–218. Springer, Heidelberg (2022). https://doi.org/10.1007/978-3-031-25066-8_9
5. Chen, C.T.: Linear System Theory and Design. Saunders college publishing, Philadelphia (1984)
6. Chen, X., Mottaghi, R., Liu, X., Fidler, S., Urtasun, R., Yuille, A.: Detect what you can: detecting and representing objects using holistic models and body parts. In: IEEE Conference on Computer Vision and Pattern Recognition (2014)
7. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: a large-scale hierarchical image database. In: IEEE Conference on Computer Vision and Pattern Recognition (2009)
8. Dosovitskiy, A., et al.: An image is worth 16×16 words: transformers for image recognition at scale. In: International Conference on Learning Representations (2021)
9. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. Int. J. Comput. Vision **88**, 303–338 (2010)
10. Friston, K.J., Harrison, L., Penny, W.: Dynamic causal modelling. Neuroimage **19**, 1273–1302 (2003)
11. Fu, D.Y., Dao, T., Saab, K.K., Thomas, A.W., Rudra, A., Re, C.: Hungry hungry hippos: towards language modeling with state space models. In: International Conference on Learning Representations (2023)
12. Grazzi, R., Siems, J., Schrodi, S., Brox, T., Hutter, F.: Is mamba capable of in-context learning? arXiv preprint arXiv:2402.03170 (2024)
13. Gu, A., Dao, T.: Mamba: linear-time sequence modeling with selective state spaces. arXiv preprint arXiv:2312.00752 (2023)
14. Gu, A., Goel, K., Re, C.: Efficiently modeling long sequences with structured state spaces. In: International Conference on Learning Representations (2022)

15. Gu, A., Johnson, I., Goel, K., Saab, K., Dao, T., Rudra, A., Ré, C.: Combining recurrent, convolutional, and continuous-time models with linear state space layers. In: Neural Information Processing Systems (2021)
16. Hafner, D., Lillicrap, T., Ba, J., Norouzi, M.: Dream to control: learning behaviors by latent imagination. In: International Conference on Learning Representations (2020)
17. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (2016)
18. Hespanha, J.P.: Linear Systems Theory. Princeton University Press, Princeton (2018)
19. Hur, K., et al.: Genhpf: general healthcare predictive framework for multi-task multi-source learning. IEEE J. Biomed. Health Inf. (2023)
20. Ishihara, K., Kanervisto, A., Miura, J., Hautamaki, V.: Multi-task learning with attention for end-to-end autonomous driving. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (2021)
21. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. Commun. ACM (2017)
22. Liang, D., et al.: PointMamba: a simple state space model for point cloud analysis. arXiv preprint arXiv:2402.10739 (2024)
23. Liang, X., Liang, X., Xu, H.: Multi-task perception for autonomous driving. In: Autonomous Driving Perception: Fundamentals and Applications, pp. 281–321. Springer, Heidelberg (2023). https://doi.org/10.1007/978-981-99-4287-9_9
24. Lin, B., et al.: Dual-balancing for multi-task learning. arXiv preprint arXiv:2308.12029 (2023)
25. Lin, B., Ye, F., Zhang, Y., Tsang, I.: Reasonable effectiveness of random weighting: a litmus test for multi-task learning. Trans. Mach. Learn. Res. (2022)
26. Liu, B., Liu, X., Jin, X., Stone, P., Liu, Q.: Conflict-averse gradient descent for multi-task learning. In: Neural Information Processing Systems (2021)
27. Liu, Y., et al.: Vmamba: visual state space model. arXiv preprint arXiv:2401.10166 (2024)
28. Liu, Z., et al.: Swin transformer: hierarchical vision transformer using shifted windows. In: IEEE/CVF International Conference on Computer Vision (2021)
29. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: International Conference on Learning Representations (2019)
30. Ma, J., Li, F., Wang, B.: U-mamba: enhancing long-range dependency for biomedical image segmentation. arXiv preprint arXiv:2401.04722 (2024)
31. Maninis, K.K., Radosavovic, I., Kokkinos, I.: Attentive single-tasking of multiple tasks. In: Computer Vision and Pattern Recognition (2019)
32. Mehta, H., Gupta, A., Cutkosky, A., Neyshabur, B.: Long range language modeling via gated state spaces. In: International Conference on Learning Representations (2023)
33. Misra, I., Shrivastava, A., Gupta, A., Hebert, M.: Cross-stitch networks for multi-task learning. In: IEEE Conference on Computer Vision and Pattern Recognition (2016)
34. Sener, O., Koltun, V.: Multi-task learning as multi-objective optimization. In: Neural Information Processing Systems (2018)
35. Silberman, N., Hoiem, D., Kohli, P., Fergus, R.: Indoor segmentation and support inference from RGBD images. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012. LNCS, vol. 7576, pp. 746–760. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33715-4_54

36. Vandenhende, S., Georgoulis, S., Van Gansbeke, W., Proesmans, M., Dai, D., Van Gool, L.: Multi-task learning for dense prediction tasks: a survey. IEEE Trans. Pattern Anal. Mach. Intell. **44**(7), 3614–3633 (2021)
37. Vandenhende, S., Georgoulis, S., Van Gool, L.: MTI-net: multi-scale task interaction networks for multi-task learning. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12349, pp. 527–543. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58548-8_31
38. Wang, C., Tsepa, O., Ma, J., Wang, B.: Graph-Mamba: towards long-range graph sequence modeling with selective state spaces. arXiv preprint arXiv:2402.00789 (2024)
39. Wang, J., Gangavarapu, T., Yan, J.N., Rush, A.M.: MambaByte: token-free selective state space model. arXiv preprint arXiv:2401.13660 (2024)
40. Wolf, T., et al.: Transformers: state-of-the-art natural language processing. In: Conference on Empirical Methods in Natural Language Processing (2020)
41. Xing, Z., Ye, T., Yang, Y., Liu, G., Zhu, L.: Segmamba: long-range sequential modeling mamba for 3d medical image segmentation. arXiv preprint arXiv:2401.13560 (2024)
42. Xu, D., Ouyang, W., Wang, X., Sebe, N.: Pad-net: multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing. In: IEEE Conference on Computer Vision and Pattern Recognition (2018)
43. Xu, Y., Li, X., Yuan, H., Yang, Y., Zhang, L.: Multi-task learning with multi-query transformer for dense prediction. IEEE Trans. Circuits Syst. Video Technol. **34**(2), 1228–1240 (2024)
44. Ye, F., Lin, B., Cao, X., Zhang, Y., Tsang, I.: A first-order multi-gradient algorithm for multi-objective bi-level optimization. arXiv preprint arXiv:2401.09257 (2024)
45. Ye, F., Lin, B., Yue, Z., Guo, P., Xiao, Q., Zhang, Y.: Multi-objective meta learning. In: Neural Information Processing Systems (2021)
46. Ye, F., Lyu, Y., Wang, X., Zhang, Y., Tsang, I.: Adaptive stochastic gradient algorithm for black-box multi-objective learning. In: International Conference on Learning Representations (2024)
47. Ye, H., Xu, D.: Inverted pyramid multi-task transformer for dense scene understanding. In: European Conference on Computer Vision (2022)
48. Yu, T., Kumar, S., Gupta, A., Levine, S., Hausman, K., Finn, C.: Gradient surgery for multi-task learning. In: Neural Information Processing Systems (2020)
49. Ze, Y., et al.: Gnfactor: multi-task real robot learning with generalizable neural feature fields. In: Conference on Robot Learning (2023)
50. Zhang, T., Li, X., Yuan, H., Ji, S., Yan, S.: Point could mamba: point cloud learning via state space model. arXiv preprint arXiv:2403.00762 (2024)
51. Zhang, Y., Yang, Q.: A survey on multi-task learning. IEEE Trans. Knowl. Data Eng. **34**(12), 5586–5609 (2022)
52. Zhang, Z., Cui, Z., Xu, C., Yan, Y., Sebe, N., Yang, J.: Pattern-affinitive propagation across depth, surface normal and semantic segmentation. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (2019)
53. Zhou, L., et al.: Pattern-structure diffusion for multi-task learning. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (2020)
54. Zhu, L., Liao, B., Zhang, Q., Wang, X., Liu, W., Wang, X.: Vision mamba: efficient visual representation learning with bidirectional state space model. In: International Conference on Machine Learning (2024)