arXiv:2308.05721v5 [cs.CV] 2 Nov 2025

DeGMix: Efficient Multi-Task Dense Prediction with Deformable and Gating Mixer

Yangyang Xu, Yibo Yang, Bernard Ghanem, *Member, IEEE*, Lefei Zhang, *Senior Member, IEEE*, Bo Du, *Senior Member, IEEE*, and Jun Zhu, *Fellow, IEEE*

***Abstract*—Convolution neural networks (CNNs) and Transformers have their own advantages and both have been widely used for dense prediction in multi-task learning (MTL). CNNs excel at capturing local spatial patterns, whereas Transformers are effective at modeling long-range dependencies. However, existing approaches typically employ them independently, leaving the potential of their complementary strengths underexploited. In this work, we present DeGMix (Deformable and Gating Mixer), an efficient encoder–decoder framework that unifies convolution and attention mechanisms for MTL. DeGMix integrates a *deformable mixer encoder* and a *task-aware gating Transformer decoder* to achieve robust and parameter-efficient learning. First, the deformable mixer encoder contains two types of operators: the channel-aware mixing operator leveraged to allow communication among different channels, and the spatial-aware deformable operator with deformable convolution applied to efficiently sample more informative spatial locations. By simply stacking the operators, we obtain the deformable mixer encoder, which effectively captures significant deformable features. Second, the task-aware gating transformer decoder is used to perform task-specific predictions, in which task interaction block integrated with self-attention is applied to capture task interaction features, and the task query block integrated with gating attention is leveraged to dynamically select the corresponding task-specific features. Furthermore, the results of the experiment demonstrate that the proposed DeGMix uses fewer GFLOPs and significantly outperforms current Transformer-based and CNN-based competitive models on a variety of metrics on three dense prediction datasets (*i.e.,* NYUD-v2, PASCAL-Context, and Cityscapes). For example, using Swin-L as the backbone, our method achieves 57.55 mIoU in segmentation on NYUD-v2, outperforming the best existing method by +5.2 mIoU. Our fine-tuning experiments on the PASCAL-Context show that DeGMix achieves higher accuracy on downstream tasks compared to the MTL-fully fine-tuning model while reducing the number of trainable parameters by 9.39×. Our code and models are publicly available at https://github.com/yangyangxu0/DeGMix.**

***Index Terms*—Scene Understanding, Multi-Task Learning, Dense Prediction, CNNs, Transformers.**

I. Introduction

The human visual system possesses remarkable capabilities, enabling it to perform a multitude of tasks within a single visual scene, including classification, segmentation, and recognition. Consequently, multi-task learning (MTL) has become a significant area of research in computer vision. Our objective is focused on developing an advanced vision model capable of simultaneously performing multiple tasks across diverse visual scenarios, with a strong emphasis on efficiency. The core challenge lies in designing an efficient architecture that can effectively capture both task-specific and shared representations across tasks like semantic segmentation, human parts segmentation, depth estimation, boundary detection, saliency estimation, and normal estimation, all while maintaining high performance and computational efficiency.

Recently, existing works [1]–[6] have achieved significant advancements through CNN and Transformer technologies for MTL of dense prediction. CNN-based MTL models excel at local representation learning and have been carefully engineered for cross-task feature sharing; $e.g.$, Cross-stitch [7] mixes shared and task-specific activations, while others [1], [3] leverage distillation and enlarged receptive fields to strengthen cross-task information flow. Transformer-based MTL models [4], [5], [8] utilize the efficient self-attention mechanism [9], which facilitates efficient modeling of global interactions and task interactions. These approaches demonstrate the growing potential of CNNs and Transformers in enhancing the performance of multi-task dense prediction.

We, in particular, identify three primary challenges in MTL. First, although CNN-based MTL models have been proven effective for MTL methods [1], [2], [10], which better capture the multiple task context in a local field, they suffer from a lack of global modeling and task interaction. These CNN-based MTL methods usually focus on modeling locality naturally, thus lacking the capability to model long-range dependencies. Recently, there have been a few attempts [3] to introduce the global and local context through distillation using scaled dot-product attention. However, these distillation pipelines often struggle to fully address global dependencies and tend to introduce unnecessary complexity.

Second, existing Transformer-based models [5], [6] excel at modeling the global dependencies across tasks and enhancing task interactions in MTL. However, these models rely on standard self-attention mechanisms, which overlook the task-specific local information and introduce huge computational costs. Specifically, it only focuses on capturing long-range dependencies without task-aware local information during self-attention processing. This results in significant computational demands, particularly when dealing with high-resolution inputs, as the calculations involving queries, keys, and values

Yangyang Xu and Jun Zhu are with the Department of Computer Science & Technology, Institute for AI, Tsinghua University, Beijing, 100084, China. E-mail: yangyangxu@mail.tsinghua.edu.cn; dcszj@tsinghua.edu.cn.

Lefei Zhang and Bo Du are with the School of Computer Science, Wuhan University, Wuhan 430072, China. E-mail: {zhanglefei,dubo}@whu.edu.cn.

Yibo Yang and Bernard Ghanem are with the King Abdullah University of Science and Technology, Thuwal, Mecca Province, 23955-6900, Saudi Arabia. E-mail:{yibo.yang,Bernard.Ghanem}@kaust.edu.sa.

Manuscript received April xx, 2025; revised xx xx, 202x.

Corresponding Author: Lefei Zhang.

become increasingly intensive.

Third, although several methods [11] combine CNNs and Transformers to capture both local and global features, they struggle to effectively select task-relevant features in the corresponding task. This limitation can undermine the overall performance of the model and its ability to adapt effectively across a range of diverse tasks, ultimately affecting its versatility and generalization capabilities.

Recent works [12], [13] demonstrate that locality is a reasonable compensation for global dependency. However, the potential benefits of integrating CNN and Transformer in MTL have yet to be fully explored. These challenges motivate us to seek a principled combination of CNN and Transformer architectures that can inherit both local sensitivity and global reasoning for multi-task dense prediction in MTL. To address the challenges, we propose the deformable and gating mixer (DeGMix) model: a simple and effective approach to multi-task dense prediction. DeGMix leverages the strengths of deformable CNNs and query-based Transformers, enhanced with spatial gating mechanisms, to deliver superior performance. Instead of a naive combination of CNN and Transformer, we introduce an innovative design comprising a deformable mixer encoder and a task-aware gating Transformer decoder, ensuring a seamless integration of their complementary capabilities.

Specifically, the deformable mixer encoder of our DeGMix consists of channel-aware mixing convolution and spatial-aware deformable convolution. Motivated by the success of deformable convolutional networks [14] in vision tasks, our deformable mixer encoder learns different deformed features for each task based on more efficient sampling spatial locations and channel location mixing ($i.e.$, deformed feature). It learns multiple deformed features that highlight more informative regions with respect to the different tasks. The task-aware gating transformer decoder of our DeGMix consists of task interaction block and task query gating blocks. In the task-aware gating transformer decoder, the multiple deformed features are fused and fed into our task interaction block. We use the fused feature to generate task-interacted features via a multi-head self-attention for model task interactions. Current Transformer-based studies for MTL use regular query tokens. In contrast, we use deformed features as query tokens via attention with gating, which increases the task awareness for the final representation output by the Task-aware Gating Transformer Decoder. Thus, we use deformed features directly as *query* tokens to focus on the task awareness of each task. We expect the set of candidate *key/value* to be from task-interacted features. Then, our task query gating block uses the deformed features and task-interacted features as input and generates the dynamically selected task awareness features.

In this way, our deformable mixer encoder adaptively selects informative spatial regions as deformed features to alleviate the lack of global modeling in CNN. The task-aware gating transformer decoder performs the task interactions by self-attention and enhances task awareness via a query-based Transformer with a spatial gating. By integrating the complementary advantages of both modules, the framework delivers deformable and comprehensive representations from local and global perspectives, while maintaining efficiency and emphasizing task-specific relevance.

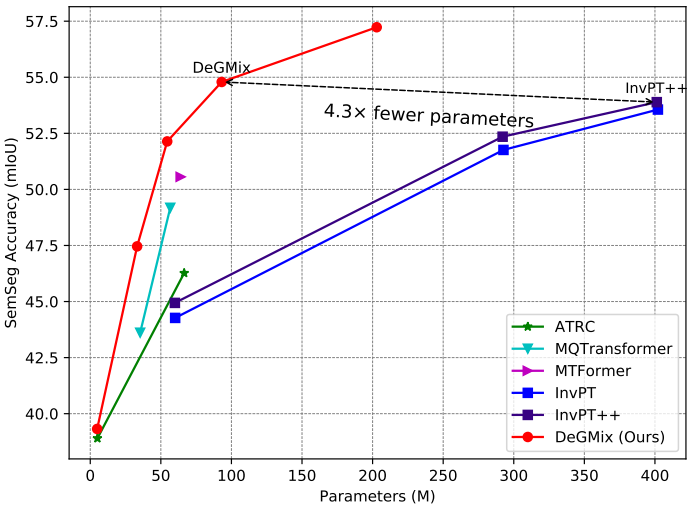


Fig. 1. Accuracy-to-Parameter trade-off on NYUD-v2 dataset. Our DeGMix model can outperform existing MTL methods by a large margin while maintaining an optimal balance between accuracy and model parameters.

We conduct extensive experiments on three dense prediction datasets to evaluate the DeGMix. Our DeGMix approach consistently outperforms the state-of-the-art methods on a variety of metrics (see Figure 1). We perform ablation studies to validate the effectiveness of DeGMix, highlighting its key strengths and unique properties. We also fine-tune [15] DeGMix for a parameter-efficient MTL approach. The main contributions of this paper are summarized as follows:

1) We propose a simple and effective DeGMix method for MTL of dense prediction via combining both merits of CNN and Transformer. Most importantly, our method addresses the limitations of CNN-based models by incorporating global modeling capabilities, while simultaneously overcoming the task-awareness shortcomings of Transformer-based models.

2) We introduce an efficient multi-task dense prediction with deformable and gating mixer (DeGMix) model, which consists of the deformable mixer encoder (Section III-C) and task-aware gating transformer decoder (Section III-D). DeGMix can adaptively control task-specific feature learning by capturing deformable features using deformable convolution with spatial gating in the deformable mixer encoder, as well as task-aware features through query-based attention with spatial gating in the task-aware gating transformer decoder.

3) A novel shared spatial gating layer is introduced across all tasks to perform dynamic feature selection. This gating mechanism effectively highlights informative features and ensures that only task-relevant information is propagated through both the encoder and the decoder, leading to high-quality, task-specific outputs, enhancing the model's ability to effectively learn and distinguish between different tasks.

4) Extensive experiments are conducted in NYUD-v2 [16], PASCAL-Context [17], and Cityscapes [18]. DeGMix demonstrates superior accuracy in downstream tasks compared to fully fine-tuning MTL model while training significantly fewer parameters. The visualization results show the efficacy of our model. DeGMix's strong performance on MTL can demon-

strate the benefits of combining the deformable CNN and query-based Transformer. As shown in Figure 1, it outperforms all state-of-the-art methods by a large margin on SemSeg.

II. Related Work

A. Multi-Task Learning (MTL)

Multi-task learning (MTL) has become an increasingly popular area of research in recent years, with many works proposing new methods and architectures to tackle the challenges of learning multiple tasks simultaneously. MTL has undergone significant evolution with the advancements in CNNs and Vision Transformers. MTL tasks are primarily distributed in two aspects: the development of innovative model architectures [3] and the optimization of task loss weighting [5]. In the vision domain, the core idea of MTL is to use a single model to predict semantic segmentation, human parts segmentation, depth, surface normal, boundary, $etc.$, which is an interesting topic. Recent works propose novel architectures that learn shared representation to capture both task-specific and shared information, enhancing overall performance. MuST [2] model uses the knowledge in independent specialized teacher models to train a general model for creating general visual representations. Several recent MTL frameworks follow different technologies: [10] and [1] are CNN-based MTL models, [3] is Neural Architecture Search-based model and [4]–[6], [8] are Transformer-based models.

[19] states that the MTL structures in vision tasks can be summarized into two categories: encoder-focused and decoder-focused architectures. Encoder-focused approaches [20], [21] employ a shared encoder to extract general visual representations, which are then processed by task-specific heads for individual predictions. Decoder-focused works [3], [10], [22], [23] use a shared backbone network to extract a shared feature for each task. Then, the designed task-specific modules perform the interactions to capture valuable information from the cross-task. Representative examples include MTI-Net [1], which models task interactions at multiple scales through feature propagation and aggregation, and ATRC [3], which adaptively selects the most informative attention context for task refinement.

With the success of Vision Transformers [24], recent advances in computational modeling have significantly improved MTL performance [4], [6], [8], [23]. MTFormer [8] and InvPT [6] enhance dense prediction through self-attention mechanisms built on strong Transformer backbones. In contrast, MQTransformer [4] introduces multi-task queries to explicitly model task interactions and transfer task-specific knowledge. However, most Transformer-based MTL methods still emphasize shared representations, limiting their ability to disentangle task-specific features.

B. Deformable CNNs and Vision Transformers.

Deformable CNNs. CNNs have achieved milestone contributions in many domains. Deformable ConvNets [14], [25] harness the enriched modeling capability of CNNs via deformable convolution and deformable spatial locations. A deformable convolution dynamically attends to flexible spatial locations conditioned on learned offsets, enabling more expressive feature modeling. Inspired by this idea, deformable attention [26] extends the concept to Transformers by attending to a sparse set of informative sampling points, efficiently capturing local-to-global dependencies. CNNs are widely used for MTL, and recent works have proposed various CNN-based architectures to learn shared features across tasks. UberNet [27] is proposed to train in an end-to-end manner a unified CNN architecture to solve multiple vision tasks. NDDR-CNN [10] introduces a CNN structure, neural discriminative dimensionality reduction, for multi-task learning, which enables automatic feature fusing at every layer and is formulated by combining existing CNN components in a novel way with clear mathematical interpretability. In addition, MTI-Net [1] proposes a MTL method, which models task interactions at multiple scales through a multi-scale multi-modal distillation unit, feature propagation module, and feature aggregation unit to produce refined task features and improve performance on dense prediction tasks. Previous CNN-based MTL methods utilize plain convolution to capture the task's local information from the input feature. However, conventional CNN-based MTL models mainly rely on local receptive fields and insufficiently exploit global context. In contrast, our approach replaces the self-attention variables (query, key, and value) in Transformers with deformable feature tensors, integrating the flexibility of deformable convolutions with the global reasoning ability of Transformers.

Vision Transformers. Transformers and attention mechanisms [9] have shown great success in natural language processing and have been rapidly extended to computer vision tasks such as detection, classification, and segmentation. Vision Transformers (ViT) [24] pioneered the adaptation of self-attention to image analysis by representing an image as a sequence of visual patches, achieving performance comparable to or surpassing modern CNNs. Subsequent works [12], [28] have been introduced by improving the attention mechanism for dense prediction tasks. Unlike the original designs in [9], [24], some vision Transformers usually employ a hierarchical architecture and replace the global self-attention among all patches to local self-attention to develop efficient attention mechanisms. These efficient attention mechanisms include windowed attention [12], [29], global tokens [30], gated attention [31] and deformable attention [26]. Recently, these works are also extended to the MTL domain to learn good representations for multiple tasks of dense predictions. MulT [5] uses a shared attention mechanism to model task dependencies and perform MTL for dense prediction. MTFormer [8] designs the shared transformer encoder and shared transformer decoder, and task-specific self-supervised cross-task branches are introduced to generate the final outputs, thereby enhancing the MTL performance. InvPT [6] and InvPT++ [23] employ an inverted pyramid multi-task transformer to learn long-range interaction via plain self-attention in both spatial and all-task contexts on the multi-task features with a gradually increased spatial resolution for dense prediction. In the DeMT [11] method, convolution and attention are combined to perform muli-task dense prediction. In contrast, our framework in-

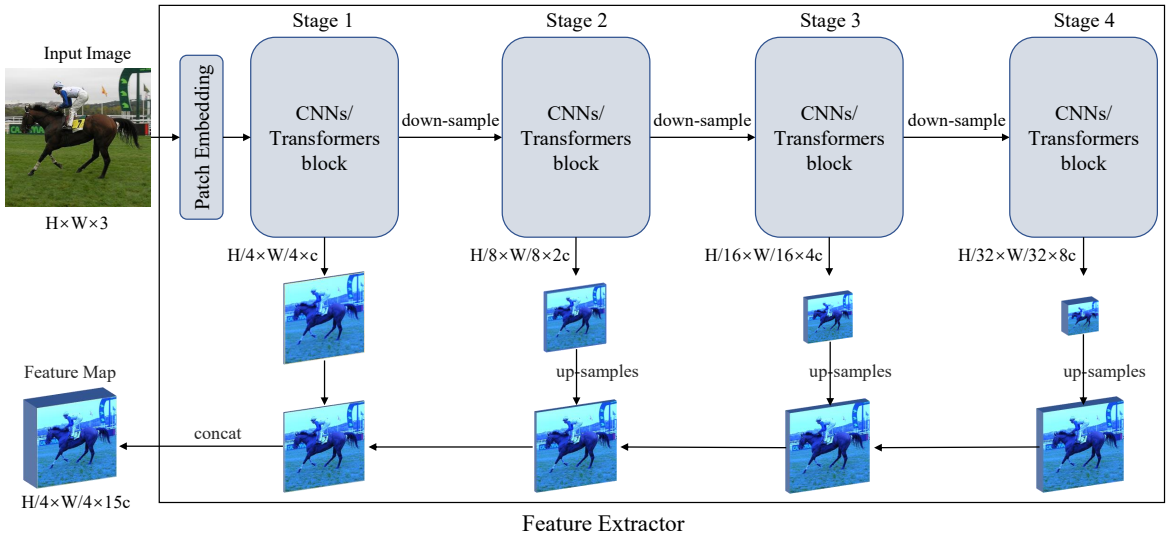


Fig. 2. Illustration of our feature extractor. Our feature extractor can be compatible with three kinds of backbone networks (*i.e.,* HRNet, Swin-Transformer (Swin) and Vision Transformer (ViT)). Note that when ViT is used as backbone, we select four block-specific features. We conduct the scale number ablation study in Table V. In this figure, we set the feature map channel $15c = C$.

troduces two key innovations: **First**, we introduce a novel simplified gating mechanism in the deformable mixer encoder and task-aware gating transformer decoder to selectively gather contexts for each query token based on its content. **Second**, we design a shared spatial gating layer that applies spatial gating to each task-specific feature within the task query gating block the most informative regions. In addition, we use the query-based transformer with task shared gating block approach for modeling and leverage deformed features as queries in transformer calculations to enhance task-relevant features. These queries can naturally disentangle the task-specific feature from the fused feature. By integrating deformable convolutional flexibility with Transformer-based global reasoning, our method achieves state-of-the-art results on NYUD-v2 [16], PASCAL-Context [17], and Cityscapes [18].

III. Approach

A. Overall Architecture

DeGMix adopts a non-shared encoder–decoder architecture. Specifically, a deformable mixer encoder extracts task-specific spatial representations, while the task interaction and task query gating blocks decode task-aware features through self-attention with gating mechanisms. The following section presents the multi-task learning objectives.

B. Feature Extractor

The feature extractor is employed to aggregate multi-scale features and generate a shared feature map for each task. The utilization of a shared feature map enables the model to leverage information from different scales, thereby enhancing its capacity to perform multiple tasks effectively. As shown in Figure 2, the initial image data $X_{in} \in \mathbb{R}^{H\times W\times 3}$ (3 is image channel) is input to the backbone (CNN-based or Transformer-based), which then generates four stages of image features. In addition, we collect the features of four blocks when ViT is used as backbone. The four-stage image features are up-sampled to the same resolution, and then they are concatenated along the channel dimension to obtain an image feature $X \in \mathbb{R}^{\frac{H}{4}\times\frac{W}{4}\times C}$, where H, W, and C are the height, width, and channel of the image feature, respectively. An ablation on the four-stage features is provided in Table V.

C. Deformable Mixer Encoder

Motivation. Inspired by Deformable ConvNets [14] and Deformable DETR methods [26], we propose the deformable mixer encoder that adaptively provides more efficient receptive fields and sampling spatial locations for each task. To achieve this objective, we first introduce a deformable mixer encoder that consists of four operation layers, the linear, a channel-aware mixing convolution, the GELU&BN, a spatial-aware deformable convolution, and a spatial gating. The deformable mixer encoder separates the mixing of channel-aware location features and spatial-aware deformable features. As shown in Figure 4, the channel-aware mixing convolution and spatial-aware deformable convolution operators are interleaved to enable interaction of both input feature dimensions (*i.e.,* $(HW)\&C$). The shared spatial gating layer allows the model to learn task-agnostic features that are beneficial across all tasks, enabling better synergy between related tasks.

Specifically, the channel-aware mixing convolution learns task-specific channel dependencies, enabling adaptive information sharing across tasks, while the spatial-aware deformable convolution aggregates contextual information from a small set of learnable sampling points. Stacking these modules sequentially yields a single deformable mixer block, whose depth effect is analyzed in Table V.

The deformable mixer encoder structure is shown in Figure 4. First, a linear layer reduces the channel dimension of the image feature $X \in \mathbb{R}^{\frac{H}{4}\times\frac{W}{4}\times C}$ from C to a smaller dimension C'. The linear layer can be formally expressed as:

$$X = W \cdot \mathrm{Norm}(X), \tag{1}$$

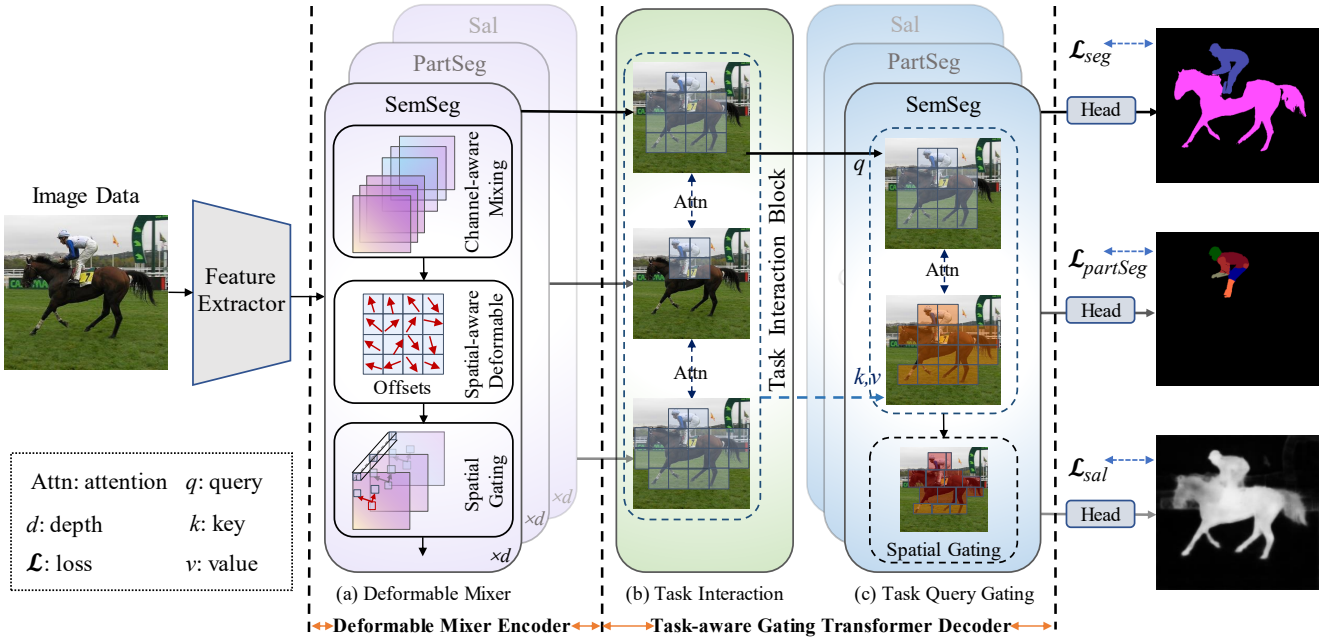


Fig. 3. An overview of our model jointly handles multiple tasks with a unified encoder-decoder architecture. The DeGMix model consists of the deformable mixer encoder and task-aware gating transformer decoder. The core blocks of our DeGMix are (a) the deformable mixer (DM) that generates task-relevant deformable features, (b) the task interaction block that performs the interactions among tasks, and (c) the task query gating block that can dynamically select the task-aware features. The depth d refers to the number of repetitions of the Deformable Mixer (see ablation on d in Table V).

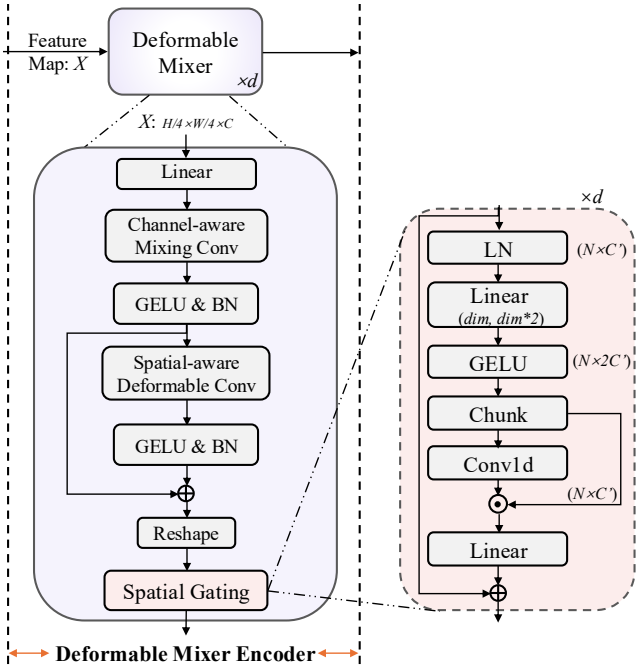


Fig. 4. Illustration of our Deformable Mixer Encoder. For simplicity, we assume there is only one task ($T = 1$) in this figure.

where Norm means LayerNorm function, after the linear layer, we obtain a smaller dimension image feature map $X \in \mathbb{R}^{\frac{H}{4} \times \frac{W}{4} \times C'}$ as the input for the downstream layers.

Channel-aware mixing convolution. Given the input image feature $X_{i,j} \in \mathbb{R}^{\frac{H}{4} \times \frac{W}{4} \times C'}$ obtained from the output of Eq. (1) output, the point (i, j) represents the spatial location in the single channel feature map. The channel-aware mixing convolution facilitates communication between different channels by applying the standard point-wise convolution (where the convolution kernel is 1×1) to mix channel locations. It can be formulated as follows:

$$X_{i,j} = W_p \cdot X_{i,j} + b, \tag{2}$$

where the W_p is the point-wise convolution weights. b is a learnable bias. Subsequently, we add GELU activation and BatchNorm as well. This operation is calculated as follows:

$$X_{i,j} = \mathrm{BN}(\sigma(X_{i,j})), \tag{3}$$

where $\sigma(\cdot)$ denotes the activation function.

Spatial-aware deformable convolution. To generate the relative offsets with respect to the reference point, the image feature $X_{i,j} \in \mathbb{R}^{\frac{H}{4} \times \frac{W}{4} \times C'}$ obtained from the output of Eq. (3) output, the point (i, j) represents the spatial location. $X_{i,j}$ is fed to the convolution operator to learn the corresponding offsets $\Delta_{(i,j)}$ for all reference points. For each location point (i, j) on the image feature $X_{i,j}$, the spatial deformable can be mathematically represented as follows:

$$D_S(X_{i,j}) = \sum_{C'=0}^{C'-1} W_{def} \cdot X((i, j) + \Delta_{(i,j)}), \tag{4}$$

where the W_{def} is a deformable convolution weights. The $\Delta_{(i,j)}$ is the learnable offset. The spatial-aware deformable convolution is followed by a GELU activation, BatchNorm, and residual connection:

$$X_{def'} = X_{i,j} + \mathrm{BN}(\sigma(D_S(X_{i,j}))), \tag{5}$$

where $\sigma(\cdot)$ represents the non-linearity function (*i.e.,* GELU), and BN denotes the BatchNorm operation.

We use *Reshape* to denote the rearrangement function. Our feature map $X_{def'}$ can written as:

$$X_{d'} = \text{Reshape}(X_{def'}), \tag{6}$$

where the *Reshape* operation flattens the feature map $X_{d'} \in \mathbb{R}^{\frac{H}{4}\times\frac{W}{4}\times C'}$ into a sequence $\mathbb{R}^{N\times C'}$ ($N = \frac{H}{4}\times\frac{W}{4}$). For T tasks, the deformable mixer encoder generates a feature set $(X^1_{d'}, X^2_{d'}, \cdots X^T_{d'})$, where T denotes the number of tasks, as shown in Figure 4. The task-specific features are obtained through channel-aware mixing and spatially deformable convolutions, forming the *deformed features*, which are subsequently fused into the input of the shared spatial gating layer.

Shared spatial gating layer. As illustrated in Figure 4, a shared spatial gating (SSG) layer is employed to enhance information flow across tasks through a gating mechanism [32]. Gating mechanisms control the path through which information flows in the network and have proven to be useful for CNN-based and Transformer-based networks [31], [32]. In MTL, the shared gating facilitates the exchange of information across tasks, allowing each task to leverage the information learned by the others. Formally, given input features x and a, a gating layer computes an output collection of feature y by $y = x \cdot \sigma(a)$, where $\sigma(\cdot)$ is some activation function. The proposed shared spatial gating is applied to refine each task independently feature through task-relevant dynamic selection.

We first apply a linear transformation to the incoming task deformed features (*i.e.,* $X_{d'}$ from Eq. 6):

$$\tilde{X}_{d'} = \sigma(\text{Linear}(\text{LN}(X_{d'}))), \tag{7}$$

where LN is the layer normalization. $\sigma(\cdot)$ is the non-linearity function (*i.e.,* GELU). After linear layer with $X_{d'} \in \mathbb{R}^{N\times C'}$, we obtain the feature $\tilde{X}_{d'} \in \mathbb{R}^{N\times 2C'}$. The $\tilde{X}_{d'}$ is split into $\tilde{X}_{d^1} \in \mathbb{R}^{N\times C'}$ and $\tilde{X}_{d^2} \in \mathbb{R}^{N\times C'}$ along the channel dimension. Given two features $\tilde{X}_{d^1}$ and $\tilde{X}_{d^2}$, the spatial gating is formulated as follows:

$$\tilde{X}'_{d''} = \tilde{X}_{d^1} \odot \text{Conv1d}(\text{LN}(\tilde{X}_{d^2})), \tag{8}$$

$$X_d = \text{Linear}(\tilde{X}'_{d''}) + X_{d'}, \tag{9}$$

where LN is the layer normalization. $\odot$ is the element-wise multiplication between matrices. *Conv1d* denotes 1D convolution operation. The residual feature $X_{d'}$ comes from Eq. 6. The final spatial gating feature $X_d \in \mathbb{R}^{\frac{H}{4}\times\frac{W}{4}\times C'}$ is reshaped from $\mathbb{R}^{N\times C'}$ ($N = \frac{H}{4}\times\frac{W}{4}$) via *Reshape* operation. After the shared spatial gating layer, we collect the T tasks' deformed feature set $(X^1_d, X^2_d, \cdots X^T_d)$.

The SSG improves the exchange of information across tasks, enabling each to benefit from others' representations. Its task-specific outputs are refined by a deformable mixer that generates *deformed features*, which are then integrated into the downstream task-interaction and task-query blocks.

D. Task-aware Gating Transformer Decoder

In the task-aware gating transformer decoder, we design the task interaction block and task query gating block (see Figure 5). To effectively model inter-task dependencies, the task interaction block employs a self-attention mechanism composed of a multi-head self-attention (MHSA) module and a lightweight multi-layer perceptron (sMLP). Similarly, the downstream task query block also integrates MHSA and sMLP modules but differs in the nature of its query features, which are tailored to task-specific representations. The feature is projected into the queries (Q), keys (K), and values (V) of dimension d_k, and self-attention is being computed by the Q, K and V. The self-attention operator is calculated as follows:

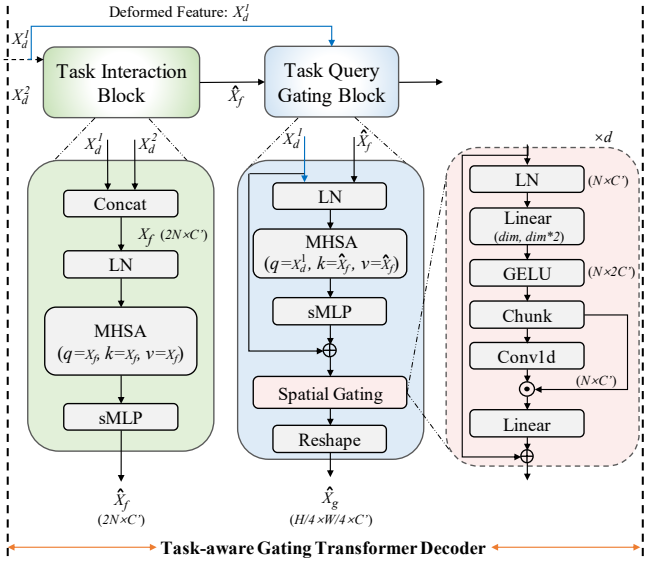


Fig. 5. Illustration of our Task-aware Gating Transformer Decoder. The *dim* denotes the input dimension. *Conv1d* denotes 1D convolution operation. For simplicity, we assume there is two tasks in the task interaction block and one task in the task query gating block.

$$\text{MHSA}(Q, K, V) = \text{softmax}(\frac{QK^{\mathcal{T}}}{\sqrt{d_k}})V, \tag{10}$$

where $Q \in \mathbb{R}^{N\times C'}$, $K \in \mathbb{R}^{N\times C'}$ and $V \in \mathbb{R}^{N\times C'}$ are the query, key and value matrices; $\text{MHSA}(Q, K, V) \in \mathbb{R}^{N\times C'}$.

Task interaction block. As shown in Figure 4 (green box), we first concatenate the deformed features from all output of the deformable mixer encoder.

$$X_f = \text{Concat}(X^1_d, X^2_d, \cdots X^T_d), \tag{11}$$

where $X_f \in \mathbb{R}^{TN\times C'}$ is the fused feature. The T means task number in $X^T_d \in \mathbb{R}^{N\times C'}$. Then, for efficient task interaction, we construct a self-attention strategy via the fused feature X_f:

$$X'_f = \text{MHSA}(Q, K, V = \text{LN}(X_f), \text{LN}(X_f), \text{LN}(X_f)), \tag{12}$$

$$\hat{X}_f = \text{sMLP}(X'_f), \tag{13}$$

where $\hat{X}_f \in \mathbb{R}^{TN\times C'}$ is the task-interacted feature. LN means LayerNorm. sMLP consists of a linear layer and a LayerNorm.

Task query gating block. As illustrated in Figure 5 (left), we take the deformed feature X_d as task query and the task-interacted feature $\hat{X}_f$ as key & value to MHSA. The deformed feature is applied as a query in MHSA to decode the task awareness feature from the task-interacted feature for each task prediction. We first apply the LayerNorm in parallel to generate queries Q, keys K, and values V:

$$\hat{Q} = \mathrm{LN}(X_d), \quad \hat{K} = \mathrm{LN}(\hat{X}_f), \quad \hat{V} = \mathrm{LN}(\hat{X}_f), \tag{14}$$

where LN is the layer normalization. X_d and $\hat{X}_f$ are the output of the deformable mixer encoder and the task interaction block, respectively. Then, the task query block operation using a MHSA is calculated as follows:

$$\hat{X}_d = \mathrm{MHSA}(\hat{Q}, \hat{K}, \hat{V}), \tag{15}$$

$$\hat{X} = X_d + \mathrm{sMLP}(\hat{X}_d), \tag{16}$$

where the residual feature X_d comes from Eq. (9). The task awareness feature $\hat{X} \in \mathbb{R}^{N \times C'}$.

Shared spatial gating layer. As illustrated in Figure 5, we use a shared spatial gating (SSG) layer with a gating mechanism [32] in decoder to allow the tasks to interact and share important features from each other. The SSG process can be formulated as follows:

$$\tilde{X}_g = \sigma(\mathrm{Linear}(\mathrm{LN}(\hat{X}))), \tag{17}$$

where LN is the layer normalization. $\sigma(\cdot)$ is the non-linearity function (*i.e.,* GELU). We first apply a linear transformation to the incoming task awareness feature (*i.e.,* $\hat{X}$). After Linear with $\hat{X} \in \mathbb{R}^{N \times C'}$, we obtain the feature $\tilde{X}_g \in \mathbb{R}^{N \times 2C'}$. The $\tilde{X}_g$ is split into $X_{g^1} \in \mathbb{R}^{N \times C'}$ and $X_{g^2} \in \mathbb{R}^{N \times C'}$ along the channel dimension. Given two features X_{g^1} and X_{g^2}, the spatial gating is formulated as:

$$X'_g = X_{g^1} \odot \mathrm{Conv1d}(\mathrm{LN}(X_{g^2})), \tag{18}$$

$$\hat{X}_g = \mathrm{Reshape}(\mathrm{Linear}(X'_g) + \hat{X}), \tag{19}$$

where $\odot$ is the element-wise multiplication between matrices. *Conv1d* denotes 1D convolution operation. The residual feature $\hat{X}$ comes from Eq. (16). The final spatial gating feature $\hat{X}_g \in \mathbb{R}^{\frac{H}{4} \times \frac{W}{4} \times C'}$ is reshaped from $\mathbb{R}^{N \times C'}$ ($N = \frac{H}{4} \times \frac{W}{4}$) via *Reshape* operation. Overall, the SSG controls the task information flow by gating through the task awareness features, thereby allowing each task to focus on the fine details complementary to the other tasks. By leveraging this gating mechanism, the SSG can compute nonlinear interactions between two input features, which significantly enriches the expressiveness of the dynamically selected features. To further demonstrate the advantages of the shared spatial gating layer, we conduct an ablation study, as shown in Table V.

DeGMix integrates a deformable mixer encoder and a task-aware gating Transformer decoder. The encoder adaptively focuses on task-relevant spatial regions, enhancing localization-sensitive tasks such as boundary detection. The decoder promotes cross-task interaction while preserving task specificity, aided by a shared spatial gating layer that maintains coherence and efficiency across tasks.

E. MTL Loss Function

For balancing the loss contribution for each task, we set the weight α_t to decide the loss contribution for the task t. A weighted sum $\mathcal{L}_{total}$ of task-specific losses:

$$\mathcal{L}_{total} = \sum_{t=1}^{T} \alpha_t \mathcal{L}_t, \tag{20}$$

where the $\mathcal{L}_t$ is a loss function for task t. For fair comparison, both α_t and $\mathcal{L}_t$ follow the settings in ATRC [3] and MQTransformer [4]. Let $\mathcal{L}_{total}$ be the set of total loss of the MTL framework ($\mathcal{L}_{total} = \{\mathcal{L}_{seg}, \mathcal{L}_{partseg}, \mathcal{L}_{sal}, \mathcal{L}_{depth}, \mathcal{L}_{normal}, \mathcal{L}_{bound}\}$). Cross entropy loss is used for SemSeg ($\mathcal{L}_{seg}$), Human Parts Seg ($\mathcal{L}_{partseg}$) and saliency ($\mathcal{L}_{sal}$) tasks, which computes the cross entropy loss between input and target. L1 loss is adopted for Depth ($\mathcal{L}_{depth}$) and Surface Normal ($\mathcal{L}_{normal}$) tasks. Binary cross entropy loss is employed for Boundary ($\mathcal{L}_{bound}$), which measures binary cross entropy between target and input logits.

IV. EXPERIMENTS

A. Setup

Implementation. All Swin-Transformer backbones generate four feature scales (1/4, 1/8, 1/16, 1/32) that are aggregated through a simple multi-scale feature extractor (see Section III-B). For ViT-B/L, we extract features from every three and six transformer blocks, respectively (*i.e.,* the 3rd–6th–9th–12th layers for ViT-B and the 6th–12th–18th–24th for ViT-L). DeGMix is trained using SGD with a learning rate of 10^{-3} and weight decay of 5×10^{-4}. Unless specified otherwise, all models are trained for 400 epochs. The task weights in Eq. (20) are set as α_{seg} = 1.0, α_{depth} = 1.0, $\alpha_{partseg}$ = 2.0, α_{sal} = 5.0, α_{normal} = 10.0, and α_{bound} = 50.0 to balance multi-task objectives. All experiments are conducted on 8 A100 GPUs.

Datasets. We conduct experiments on three publicly accessible datasets, NYUD-v2 [16], PASCAL-Context [17], and Cityscapes [18]. NYUD-v2 contains paired RGB–Depth images with 795 training and 654 testing samples, and is widely used for semantic segmentation (SemSeg), depth estimation (Depth), surface normal estimation (Normal), and boundary detection (Bound). PASCAL-Context contains 10103 training/validation and 9637 testing images, offering dense annotations for the entire scene across tasks such as SemSeg, human-part segmentation (PartSeg), saliency estimation (Sal), Normal, and Bound. Cityscapes focuses on urban street understanding and is commonly adopted for SemSeg and depth estimation.

Metrics. We adopt six standard metrics to evaluate our model against prior multi-task approaches: mean Intersection over Union (mIoU), root mean square error (rmse), mean Error (mErr), absolute error (aErr), optimal dataset scale F-measure (odsF), and maximum F-measure (maxF). These metrics provide comprehensive assessments of our model's performance across various tasks and datasets. Since, for multi-task learning, it's difficult to measure the performance of a model with a single metric; we introduce the average per-task performance drop (Δ_m) metric. Δ_m is used to quantify multi-task performance. $\Delta_m = \frac{1}{T}\sum_{i=1}^{T}(F_{m,i} - F_{s,i})/F_{s,i} \times 100\%$, where m, s and T mean multi-task model, single task baseline and task numbers. Δ_m: the higher, the better.

Backbones. We evaluate our method using several CNN and Vision Transformer backbones: HRNetV2-W18-small (HRNet18), HRNetV2-W48 (HRNet48) [36], Swin-Tiny (Swin-T), Swin-Small (Swin-S), Swin-Base (Swin-B), Swin-Large (Swin-L) [12], ViT-Base (ViT-B), and ViT-Large (ViT-L) [24].

Baselines. We categorize baselines into two groups: multi-task learning (MTL) and single-task learning (STL). These

TABLE I
COMPARISON OF THE MTL MODELS WITH STATE-OF-THE-ART ON NYUD-V2 DATASET. THE NOTATION '↓': LOWER IS BETTER. THE NOTATION '↑': HIGHER IS BETTER. Δ_m DENOTES THE AVERAGE PER-TASK PERFORMANCE DROP. "PARAMS" DENOTES PARAMETERS. SWIN-◇ INDICATES THAT THE SPECIFIC SWIN MODEL IS UNCERTAIN. † DENOTES RESULTS NOT REPORTED IN INVPT [6] BUT FROM OUR EVALUATION USING THE OFFICIAL CODE.

Model	Backbone	Params (M)	FLOPs (G)	SemSeg (mIoU)↑	Depth (rmse)↓	Normal (mErr)↓	Bound (odsF)↑	Δ_m[%]↑
single task baseline	HRNet18	16.09	40.93	38.02	0.6104	20.94	76.22	0.00
multi-task baseline	HRNet18	4.52	17.59	36.35	0.6284	21.02	76.36	-1.89
Cross-Stitch [7]	HRNet18	4.52	17.59	36.34	0.6290	20.88	76.38	-1.75
Pad-Net [33]	HRNet18	5.02	25.18	36.70	0.6264	20.85	76.50	-1.33
PAP [22]	HRNet18	4.54	53.04	36.72	0.6178	20.82	76.42	-0.95
PSD [34]	HRNet18	4.71	21.10	36.69	0.6246	20.87	76.42	-1.30
NDDR-CNN [10]	HRNet18	4.59	18.68	36.72	0.6288	20.89	76.32	-1.51
MTI-Net [1]	HRNet18	12.56	19.14	36.61	0.6270	20.85	76.38	-1.44
ATRC [3]	HRNet18	5.06	25.76	38.90	0.6010	20.48	76.34	1.56
DeMT	HRNet18	4.76	22.07	39.18	0.5922	20.21	76.40	2.37
DeGMix (Ours)	HRNet18	4.92	28.83	39.32	0.5927	20.17	76.40	2.50
single task baseline	Swin-T	115.08	161.25	42.92	0.6104	20.94	76.22	0.00
multi-task baseline	Swin-T	32.50	96.29	38.78	0.6312	21.05	75.60	-3.74
MQTransformer [4]	Swin-T	35.35	106.02	43.61	0.5979	20.05	76.20	0.31
InvPT [6]	Swin-T	60.14†	162.52†	44.27	0.5589	20.46	76.10	3.35†
InvPT++ [23]	Swin-T	59.95†	172.29†	44.94	0.5554	20.43	76.20	3.75†
DeMT	Swin-T	32.07	100.70	46.36	0.5871	20.65	76.90	3.36
DeGMix (Ours)	Swin-T	33.50	125.49	47.30	0.5630	20.14	77.20	5.10
single task baseline	Swin-S	200.33	242.63	48.92	0.5804	20.94	77.20	0.00
multi-task baseline	Swin-S	53.82	116.63	47.90	0.6053	21.17	76.90	-1.96
MQTransformer [4]	Swin-S	56.67	126.37	49.18	0.5785	20.81	77.00	1.59
MTFormer [8]	Swin-◇	64.03	117.73	50.56	0.4830	-	-	4.12
DeMT	Swin-S	53.03	121.05	51.50	0.5474	20.02	78.10	4.12
DeGMix (2-task) (Ours)	Swin-S	52.16	96.73	53.59	0.5521	-	-	6.5
DeGMix (Ours)	Swin-S	54.82	145.84	52.36	0.5430	19.99	78.50	4.83
single task baseline	Swin-L	789.96	819.93	56.46	0.508	19.38	78.80	0.00
multi-task baseline	Swin-L	204.96	316.87	54.53	0.532	19.51	78.30	-2.36
InvPT [6]	Swin-L	292.7†	417.27†	51.76	0.5020	19.39	77.60	-2.22†
InvPT++ [23]	Swin-L	292.1†	451.3†	52.35	0.4921	18.99	77.90	-0.82†
DeGMix (Ours)	Swin-L	202.92	321.22	57.55	0.5037	19.21	79.00	0.98
InvPT [6]	ViT-B	160.82†	327.61†	50.30	0.5367	19.00	77.60	-
InvPT++ [23]	ViT-B	159.94†	359.93†	49.83	0.5303	18.83	77.20	-
DeGMix (Ours)	ViT-B	114.33	321.04	52.09	0.5467	20.34	77.90	-
InvPT [6]	ViT-L	402.1†	555.57†	53.56	0.5183	19.04	78.10	-
InvPT++ [23]	ViT-L	401.3†	589.49†	53.89	0.5091	18.65	78.10	-
TaskPrompter [35]	ViT-L	513.1†	789.90†	55.30	0.5152	18.47	78.20	-
DeGMix (Ours)	ViT-L	334.1	507.72	56.29	0.5067	19.34	78.90	-

serve as reference points to evaluate the proposed DeGMix framework. DeGMix employs both CNN-based (*e.g.*, HRNet) and Transformer-based (*e.g.*, Swin-Transformer, ViT) backbones, including HRNet18, Swin-T/S/B/L, and ViT-B/L, for experiments on NYUD-v2 and PASCAL-Context. We also compare with representative Transformer-based MTL approaches, such as InvPT [6], MTFormer [8], and InvPT++ [23]. For fine-tuning settings, "single-task full fine-tuning" denotes training a separate pre-trained model per task, while "MTL-tuning decoders only" fixes the encoder and updates only decoder parameters. DeGMix follows the latter strategy. We further include parameter-efficient tuning baselines: Adapter [37], which adds task-specific modules to each Transformer layer; BitFit [38], which tunes only bias and patch layers; and LoRA [39], which applies low-rank adaptation (rank=4, scale=4) to attention layers.

B. Comparison with the State-of-the-art

NYUD-v2. We evaluate DeGMix on the NYUD-v2 [16] dataset, a widely used benchmark for multi-task dense prediction. Following the standard evaluation protocol [4], [16], we compare with state-of-the-art methods across multiple backbones, including HRNet18, Swin-T/S/B/L, and ViT-B/L. As shown in Table I, DeGMix consistently surpasses previous CNN- and Transformer-based models such as MQTransformer [4], MTFormer [8], InvPT [6], and InvPT++ [23]. In addition, we also observe that using a Transformer as a backbone model is more promising compared to CNN as the backbone. Because Transformer-based and CNN-based models use similar Parameters, the former shows higher accuracy in all metrics. Specifically, our DeGMix using Swin-T obtains 47.30 mIoU (SemSeg) accuracy, which is +3.69 (47.30 $vs.$ 43.61) higher than that of MQTransformer with the same Swin-T backbone and similar FLOPs, while being more parameter efficient. We also find that DeGMix can substantially outperform

TABLE II
COMPARISON OF THE MTL MODELS WITH STATE-OF-THE-ART ON PASCAL-CONTEXT DATASET. THE NOTATION '↓': LOWER IS BETTER. THE NOTATION '↑': HIGHER IS BETTER. Δ_m DENOTES THE AVERAGE PER-TASK PERFORMANCE DROP (THE HIGHER, THE BETTER). SWIN-◇ INDICATES THAT THE SPECIFIC SWIN MODEL IS UNCERTAIN. † MEANS WE TEST.

Model	Backbone	SemSeg (mIoU)↑	PartSeg (mIoU)↑	Sal (maxF)↑	Normal (mErr)↓	Bound (odsF)↑	Δ_m[%]↑
single task baseline	HRNet18	62.23	61.66	85.08	13.69	73.06	0.00
multi-task baseline	HRNet18	51.48	57.23	83.43	14.10	69.76	-6.77
PAD-Net [33]	HRNet18	53.60	59.60	65.80	15.30	72.50	-4.41
ATRC [3]	HRNet18	57.89	57.33	83.77	13.99	69.74	-4.45
MQTransformer [4]	HRNet18	58.91	57.43	83.78	14.17	69.80	-4.20
DeMT	HRNet18	59.23	57.93	83.93	14.02	69.80	-3.79
DeGMix (Ours)	HRNet18	59.56	57.71	84.03	14.00	69.70	-3.40
single task baseline	Swin-T	67.81	56.32	82.18	14.81	70.90	0.00
multi-task baseline	Swin-T	64.74	53.25	76.88	15.86	69.00	-3.23
MQTransformer [4]	Swin-T	68.24	57.05	83.40	14.56	71.10	1.07
DeMT	Swin-T	69.71	57.18	82.63	14.56	71.20	1.75
DeGMix (Ours)	Swin-T	69.44	58.02	83.31	14.31	71.20	1.94
single task baseline	Swin-S	70.83	59.71	82.64	15.13	71.20	0.00
multi-task baseline	Swin-S	68.10	56.20	80.64	16.09	70.20	-3.97
MQTransformer [4]	Swin-S	71.25	60.11	84.05	14.74	71.80	1.27
DeMT	Swin-S	72.01	58.96	83.20	14.57	72.10	1.36
DeGMix (Ours)	Swin-S	71.54	61.49	83.70	14.90	72.20	1.62
single task baseline	Swin-B	74.91	62.13	82.35	14.83	73.30	0.00
multi-task baseline	Swin-B	73.83	60.59	80.75	16.35	71.10	-3.81
MTFormer [8]	Swin-◇	74.15	64.89	67.71	-	-	2.41
DeMT	Swin-B	75.33	63.11	83.42	14.54	73.20	1.04
DeGMix (3-task) (Ours)	Swin-B	76.77	66.05	83.53	-	-	3.0
DeGMix (Ours)	Swin-B	75.37	64.82	83.75	14.22	73.0	1.84
single task baseline	Swin-L	79.26	68.92	83.84	14.28	74.50	0.00
multi-task baseline	Swin-L	77.35	63.86	82.87	14.84	73.10	-3.33
InvPT [6]	Swin-L	78.53	68.58	83.71	14.56	73.60	-0.947†
DeGMix (Ours)	Swin-L	78.54	67.42	83.74	14.17	74.90	-0.38

InvPT [6] by 3.03 mIoU SemSeg (47.30 $vs.$ 44.27), and we note DeGMix still beats it by +1.75 on Δ_m metric. MuIT [5] shows an improvement with a 13.3% increase in relative performance for semantic segmentation and an 8.54% increase for depth tasks. While our origin DeMT achieves a 14.74% and 9.43% increase compared to MuIT [5]. As shown in Figure 1, our method can outperform the state-of-the-art InvPT, which requires 4.3× more parameters compared to DeGMix. As expected, our DeGMix using Swin-L performs the best, with an almost +5.79 (57.55 $vs.$ 51.76) mIoU improvement in the SemSeg task compared to InvPT [6]. In addition, we also report the performance of InvPT [6] and InvPT++ [23] with ViT-B and ViT-L [24] backbones, respectively. These results demonstrate that DeGMix achieves consistent gains across diverse multi-task benchmarks, surpassing prior methods with fewer parameters. The model exhibits strong flexibility and generalization, setting new state-of-the-art results on NYUD-v2 [16] and outperforming both CNN- and Transformer-based counterparts across all metrics.

PASCAL-Context. We evaluate DeGMix on the PASCAL-Context [17] dataset, a widely used benchmark for dense prediction. Table II reports results across five backbones (HRNet18, Swin-T/S/B/L). Our model obtains significantly better results when compared with the baseline and other models. For example, DeGMix improves MQTransformer [4] with the same Swin-T backbone by +1.2 mIoU in SemSeg. Our DeGMix using Swin-T achieves the best performance among

TABLE III
COMPARISON WITH MTL MODELS ON CITYSCAPES. WE REPORT THE RESULTS FOR SEMANTIC SEGMENTATION (SEMSEG) AND DEPTH ESTIMATION (DEPTH) TASKS. Δ_m DENOTES THE AVERAGE PER-TASK PERFORMANCE DROP (HIGHER IS BETTER). '↓': LOWER IS BETTER. NOTATION '↑': HIGHER IS BETTER. THE BEST NUMBERS ARE IN **BOLD**.

Model	SemSeg (mIoU)↑	Depth (aErr)↓	Δ_m[%] (%)↑
Sing-task Learning (STL)	75.92	0.0119	0.00
Uncertainty [40]	74.95	0.0121	-1.73
GradNorm [41]	74.88	0.0123	-2.70
DWA [42]	75.39	0.0121	-1.42
PCGrad [43]	75.62	0.0122	-1.73
GradDrop [44]	75.69	0.0123	-2.15
IMTL-H [45]	75.33	0.0120	-1.20
CAGrad [46]	75.45	0.0124	-2.69
DeGMix (Ours)	**77.10**	**0.0118**	**1.18**

models on several metrics and can reach an average per-task performance drop (Δ_m) performance of 1.94 %. Compared to the MTFormer [8], our DeGMix (3-task) significantly outperforms it on the same three tasks (*i.e.,* Semseg, PartSeg, and Sal). In addition, when using HRNet18 and Swin-L as the backbone, our DeGMix also achieves the best performance on the Δ_m metric. We visualize qualitative results on PASCAL-Context in Figure 7. DeGMix consistently outperforms InvPT across datasets, confirming its effectiveness and generalization under both CNN- and Transformer-based backbones.

TABLE IV
WE REPORT THE EVALUATION RESULTS FOR OUR DEGMIX ON NYUD-V2 DATASET. THREE SETS ARE '$S-D$' (SEGMENTATION + DEPTH), '$S-D-N$' (+ NORMALS), AND '$S-D-N-B$' (+ BOUNDARY). NOTATION '↓': LOWER IS BETTER. NOTATION '↑': HIGHER IS BETTER.

Exp.	Model	Trained	Tested (+ (Task) is on new task)	SemSeg (mIoU)↑	Depth (rmse)↓	Normal (mErr)↓	Bound (odsF)↑
1	Multi-task learning is trained 400 epochs						
1.1	DeGMix	'$S-D$'	'$S-D$'	47.98	0.5707	-	-
1.2	DeGMix	'$S-D-N$'	'$S-D-N$'	47.07	0.5640	20.00	-
1.3	DeGMix	'$S-D-B$'	'$S-D-B$'	47.60	0.5737	-	77.6
1.4	DeGMix	'$S-N-B$'	'$S-N-B$'	47.00	-	20.18	77.2
1.5	DeGMix	'$D-N-B$'	'$D-N-B$'	-	0.5732	20.04	77.0
1.6	DeGMix	'$S-D-N-B$'	'$S-D-N-B$'	47.30	0.5630	20.14	77.2
2	Multi-task learning, adding of task fine-tuning with 100 epochs						
2.1	DeGMix	'$S-D$'	'$S-D$ $(+N)$'	47.59	0.5772	21.06	-
2.2	DeGMix	'$S-D$'	'$S-D$ $(+N,B)$'	47.47	0.5738	21.01	76.7
2.3	DeGMix	'$S-D-N$'	'$S-D-N$ $(+B)$'	46.63	0.5653	19.99	76.6
2.4	DeGMix	'$S-D-N-B$'	'$S-D-N-B$'	47.30	0.5630	20.14	77.2
3	Multi-task learning, leave one task out format, addition of the left out task						
3.1	DeGMix	'$S-D-N$'	'$S-D-N$ $(+B)$'	46.72	0.5671	20.12	29.3
3.2	DeGMix	'$S-D-B$'	'$S-D-B$ $(+N)$'	47.03	0.5717	100.66	22.5
3.3	DeGMix	'$S-N-B$'	'$S-N-B$ $(+D)$'	46.72	3.0126	47.42	21.2
3.4	DeGMix	'$D-N-B$'	'$D-N-B$ $(+S)$'	0.475	2.2626	50.26	14.0

Results on Cityscapes. To further demonstrate the robustness of our method, we also evaluate DeGMix on the Cityscapes dataset. We present the results in Table III. Specifically, for the Cityscapes 2-task scenario, we report the performance for semantic segmentation (SemSeg), depth estimation (Depth), and the average per-task performance drop ((Δ_m)). Our experiments consistently show that DeGMix outperforms existing approaches across all metrics. Notably, our method achieves a significant improvement of +1.18 (Δ_m) over the STL baseline, a trend that mirrors the performance gains observed on the NYUD-v2 and PASCAL-Context datasets. These results highlight the effectiveness and generalizability of DeGMix across diverse dense prediction tasks.

C. MTL Performs Comparable to Unseen Tasks

To evaluate the generalization of our MTL framework, we test it on unseen tasks from the NYUD-v2 dataset (Table IV). We design two experimental settings: augmenting tasks (Exp. 1–2) and leave-one-out (Exp. 3). All Exp. 2 models are fine-tuned for 100 epochs, while Exp. 3 uses a leave-one-task-out model to assess performance across all four tasks.

The effects of unseen task fine-tuning. Table IV presents the results of fine-tuning DeGMix on unseen tasks, where "+Task" denotes training on existing tasks followed by adaptation to a new one. Across Exp. 1 and 2, DeGMix demonstrates strong adaptability when integrating new tasks. For instance, in Exp. 2.3, after only 100 epochs of fine-tuning on the Boundary task, the model achieves an odsF of 76.6% on NYUD-v2, confirming its rapid adaptation ability. Moreover, joint multi-task training (Exp. 2.4) still achieves superior overall performance compared to individual fine-tuning (Exp. 2.1–2.3), indicating that integrated optimization yields better generalization. These findings suggest that DeGMix enables efficient task expansion through fine-tuning, while full multi-task training remains optimal for overall consistency and generalization.

Evaluate new tasks using the trained model. As shown in Exp. 3 of Table IV, we adopt a leave-one-task-out strategy, where the model is trained on all tasks except one and then evaluated on the omitted task. The results (Exp. 3.1–3.4) reveal that excluding a task consistently weakens performance, as indicated by the orange numbers in Exp. 3. In particular, the *S–N–B* model (Exp. 3.3) suffers notable degradation in depth, normal, and boundary estimation, while maintaining moderate segmentation performance. Conversely, the *D–N–B* model (Exp. 3.4) struggles to capture semantic information, leading to performance drops across all tasks. These findings highlight the pivotal role of semantic segmentation and emphasize how task interactions shape multi-task learning effectiveness.

D. Ablation Studies

We carry out ablations using Swin-T on the NYUD-v2 dataset. We ablate DeGMix to understand the individual contribution of each component and settings. This analysis allows us to gain insights into the importance and effectiveness of each part in the overall performance of our method.

Ablation on modules. We analyze the influence of every key component and design choice in our full method. All models are trained for 400 epochs on NYUD-v2. The DeGMix model consists of three components: deformable mixer, task interaction, and task query gating blocks. As shown in Table V, we demonstrate the advantages of the deformable mixer, task interaction, and task query gating blocks. After adding the proposed blocks to the multi-task baseline (MTB) network, the results improved persistently and notably. Specifically, we observe that the task interaction block has more effect on the performance, and it is essential to interact with all the tasks for task interaction information. This indicates that task interaction and task query gating blocks are essential to the task-aware gating transformer decoder. From Figure 7 and Table V it can be observed that different components play a beneficial role.

Effectiveness of shared spatial gating (SSG). We further examine the role of the shared spatial gating (SSG) in the task query gating block. As shown in Table V, integrating SSG into both the deformable mixer encoder and the task-aware gating Transformer decoder yields notable improvements—+8.52

TABLE V
ABLATION STUDIES AND ANALYSIS ON NYUD-V2 DATASET USING A SWIN-T BACKBONE. DEFORMABLE MIXER (DM), TASK INTERACTION (TI) BLOCK, AND TASK QUERY GATING (TQG) BLOCK ARE THE PARTS OF OUR MODEL. THE SHARED SPATIAL GATING (SSG) LAYER INSERTS INTO THE TQG BLOCK. HR48 DENOTES HRNET48. MTB DENOTES THE MULTI-TASK BASELINE. THE NOTATION '↓': LOWER IS BETTER. THE NOTATION '↑': HIGHER IS BETTER. THE W/ AND W/O INDICATE "WITH" AND "WITHOUT", RESPECTIVELY.

Model	SemSeg (mIoU)↑	Depth (rmse)↓	Normal (mErr)↓	Bound (odsF)↑
MTB	38.78	0.6312	21.05	75.6
w/ DM	42.40	0.6069	20.83	76.2
w/ DM+SSG+TI	45.24	0.5862	20.55	76.9
DM+TI+{TQG*w/o*SSG}	46.51	0.5719	20.42	77.2
DM+TI+TQG+SSG	47.30	0.5630	20.14	77.2
DeGMix Gain *vs.* MTB	+8.42	+0.0652	+0.9	+1.6

(a) Ablation on components

d	SemSeg (mIoU)↑	Depth (rmse)↓	Normal (mErr)↓	Bound (odsF)↑
1	46.96	0.5717	20.22	77.1
2	47.30	0.5630	20.14	77.2
4	48.12	0.5601	20.01	77.3
8	48.01	0.5521	19.81	77.2

(b) Ablation on the depths (d) of DM encoder

Scale	SemSeg (mIoU)↑	Depth (rmse)↓	Normal (mErr)↓	Bound (odsF)↑
1/4	6.81	1.0851	31.65	66.5
1/4, 1/8	16.91	0.8133	27.06	71.2
1/4, 1/8, 1/16	41.32	0.6254	21.18	76.8
1/4,1/8,1/16,1/32	47.30	0.5630	20.14	77.2

(c) Ablation on scales

Backbone	SemSeg (mIoU)↑	Depth (rmse)↓	Normal (mErr)↓	Bound (odsF)↑
HR48 baseline	41.96	0.5543	20.36	77.6
HR48 **w/** ours	46.87	0.5409	20.10	77.8
Swin-B baseline	51.44	0.5813	20.44	78.0
Swin-B+InvPT++ [23]	51.08	0.5008	19.28	77.1
Swin-B **w/** ours	54.45	0.5228	19.33	78.6

(d) Ablation on backbones

mIoU (47.30 vs. 38.78) on SemSeg and a 0.0682 RMSE reduction (0.5630 vs. 0.6312). This confirms that SSG effectively enhances task-aware feature selection. Qualitative results in Figure 7 also show that DeGMix surpasses the baseline MTB with DM (42.40 to 47.30 mIoU; 0.6069 to 0.5630 RMSE), validating the efficiency of shared gating across tasks.

Ablation on the depths d. As shown in Figure 4, the depth d is the number of repetitions of the deformable mixer. We add the d to analyze the effect of the depth of the deformable mixer on the DeMT model. In Table V, We vary the number of used deformable mixer depths ($e.g.$, 1, 2, 4, 8) and compare their performances. Comparing the first to last row in Table V, we observe the best performance when the depth is set to 4. However, as increasing the depth, the parameters and FLOPs also become more extensive. Practically, we choose a depth d = 1 for all models in this paper.

Ablation on scales. We investigate the effect of multi-scale features on model performance. As illustrated in Figure 2, the backbone generates features at four scales: 1/4, 1/8, 1/16, and 1/32. Table V reports the performance under different scale combinations. Model performance improves steadily with more feature scales. Using only the 1/4-scale features leads to a significant drop in accuracy, highlighting the necessity of multi-scale integration for richer semantic representation. By leveraging features from all four scales, our method effectively captures complementary semantic cues across tasks. Hence, we adopt four-scale features for all models to balance representation capacity and computational efficiency.

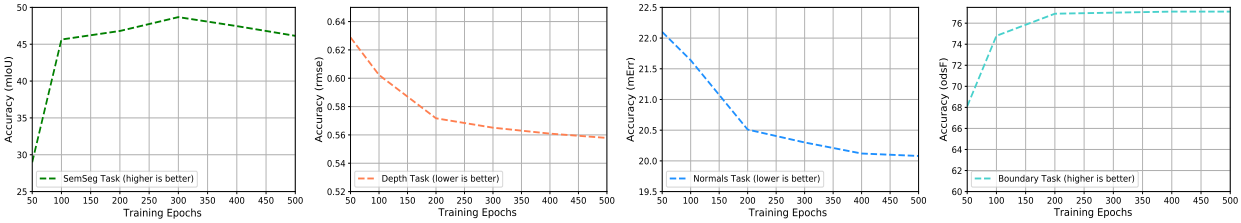
Ablation on backbones. Table V compares DeGMix with different backbones (*i.e.,*, HRNet48 and Swin-B). As shown in Table V and Figure 1, our DeGMix model with the Swin-B backbone achieves a notable improvement of +3.37 mIoU on the SemSeg task compared to the competitive InvPT++ [23]. In addition, we also observe the inspiring fact that using a larger transformer backbone can easily reach top-tier performance. The different backbones are compared to demonstrate the generalization of our method.

TABLE VI
ABLATION ON SHARED SPATIAL GATING DEPTH IN TASK QUERY GATING.

d	SemSeg (mIoU)↑	Depth (rmse)↓	Normals (mErr)↓	Bound (odsF)↑
1	47.30	0.5630	20.14	77.2
2	47.33	0.5611	20.12	77.1
4	**47.55**	0.5646	20.12	77.3
6	47.29	0.5621	20.05	77.5
8	47.45	**0.5587**	**20.05**	**77.6**

Ablation on the depths (d) of shared spatial gating in task query gating block. We analyze the impact of the shared spatial gating depth within the task query gating block for $d \in \{1, 2, 4, 6, 8\}$ (Table VI). The results indicate that increasing the depth of the shared spatial gating provides marginal performance gains. However, these improvements come at the cost of a substantial increase in model parameters, which may limit the model's efficiency and scalability. Using several depths shows a slight performance gain, and at a cost of significantly more parameters. Given this trade-off between accuracy and efficiency, we adopt $d = 1$ as the default setting to maintain balanced performance and scalability.

Ablation on different epochs. To further evaluate the performance of DeGMix, we conduct an ablation study on the impact of training for different numbers of epochs, as shown in Figure 6. We pick out models at the 50^{th}, 100^{th}, 200^{th}, 300^{th}, 400^{th} and 500^{th} epochs. Two key findings emerge: (1) performance improves with longer training until overfitting occurs beyond a certain point, and (2) the optimal epoch count varies across tasks. For example, semantic segmentation



(a) Semantic Segmentation (b) Depth Estimation (c) Surface Normal Estimation (d) Boundary Detection

Fig. 6. Influence of the number of epochs in multi-task training using Swin-T backbone on NYUD-v2. To evaluate the DeGMix performance of different epochs, we pick out models at the 50^{th}, 100^{th}, 200^{th}, 300^{th}, 400^{th} and 500^{th} epoch.

TABLE VII
COMPARISON OF DEGMIX-TUNING WITH EXISTING FINE-TUNING METHODS ON PASCAL-CONTEXT DATASET. RESULTS ARE REPORTED USING SWIN-T/L BACKBONES. THE "SINGLE INFERENCE FOR ALL TASKS" COLUMN INDICATES WHETHER A MODEL SUPPORTS JOINT TASK EXECUTION. '↓' AND '↑' DENOTE THAT LOWER OR HIGHER VALUES ARE BETTER. Δ_m DENOTES THE AVERAGE PER-TASK PERFORMANCE DROP (HIGHER IS BETTER).

Model	SemSeg (mIoU)↑	PartSeg (mIoU)↑	Sal (mIoU)↑	Normal (mErr)↓	Δ_m[%]↑	Trainable Parameters (M)	Single Inference For All Tasks
single task baseline	67.21	61.93	62.35	17.97	0	112.62	✗
MTL-Tuning Decoder Only	65.09	53.48	57.46	20.69	-9.95	1.94	✓
MTL - Full Fine Tuning	67.56	60.24	65.21	16.64	+2.23	30.06	✓
Adapte [37]	69.21	57.38	61.28	18.83	-2.71	11.24	✗
Bitfit [38]	68.57	55.99	60.64	19.42	-4.60	2.85	✗
VPT-shallow [15]	62.96	52.27	58.31	20.90	-11.18	2.57	✗
VPT-deep [15]	64.35	52.54	58.15	21.07	-10.85	3.43	✗
LoRA [39]	70.12	57.73	61.90	18.96	-2.17	2.87	✗
Polyhistor [47]	70.87	59.54	65.47	17.47	+2.34	8.96	✗
MTLoRA [48]	68.19	58.99	64.48	17.03	+1.35	4.95	✓
DeGMix-Tuning (Ours)	67.37	56.87	64.12	15.67	+1.96	2.2	✓
DITASK+Swin-L	76.23	67.53	64.07	16.90	+7.79	7.13	✓
DeGMix-Tuning+Swin-L (Ours)	78.14	64.44	64.20	16.30	+8.01	3.2	✓

peaks near 300 epochs, whereas other tasks continue to benefit from extended training. These results emphasize the need for careful tuning of training epochs to balance convergence and generalization across tasks.

E. Parameter Efficient Training

Parameter Efficient Training on PASCAL-Context. For a fair comparison, we evaluate DeGMix-Tuning with Swin-T and Swin-L backbones (Table VII). With only 2.2M trainable parameters, DeGMix-Tuning achieves a +1.96 Δ_m improvement, demonstrating strong efficiency–performance trade-offs. Although HyperFormer (+2.64), Polyhistor (+2.34), and MTL-Full Fine-Tuning (+2.23) yield slightly higher Δ_m, they require 33×, 4×, and 13.7× more parameters, respectively. When scaled to Swin-L, our method attains the best overall gain (+8.01 Δ_m) using just 3.2M parameters, surpassing DITASK [49] (+7.79 Δ_m, 7.13M params). Overall, DeGMix-Tuning achieveshigh parameter efficiency by effectively transferring multi-task knowledge with minimal additional cost.

Qualitative Comparison. Figure 7 illustrates the visual impact of each component. While the Baseline + DM (deformable mixer) struggles to recover certain objects, DeGMix produces clearer and more complete predictions across multiple tasks. Compared with ATRC [3] and InvPT [6], DeGMix yields markedly sharper boundaries and more accurate semantic and part segmentation, especially on PASCAL-Context. Consistent with quantitative results, these visualizations confirm the superior task consistency and robustness of DeGMix.

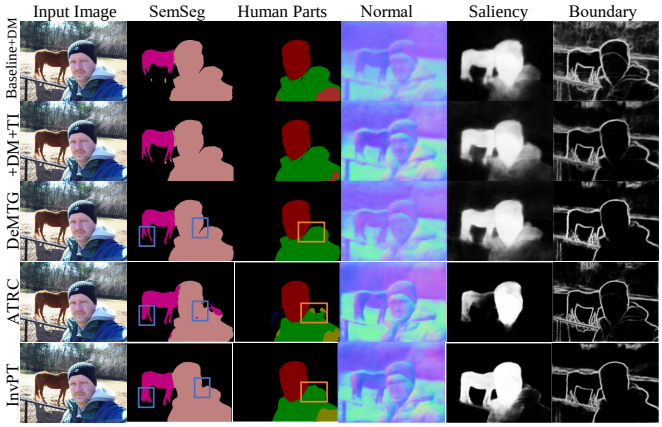


Fig. 7. Visual results on 5-task PASCAL-Context dataset. Qualitative analysis of the components according to Table V. We also select the visualization results of ATRC [3] and InvPT [6] as a comparison. Note that they are marked in blue and orange square boxes. The qualitative comparison clearly proves the efficacy of our model in capturing the objects' subtle contextual details.

Generalization Performance. Figure 8 presents qualitative results on the DAVIS dataset, highlighting the strong generalization ability of DeGMix across multiple tasks. Compared with ATRC [3] and InvPT++ [23], DeGMix produces more precise and coherent predictions, with notably sharper and more detailed segmentation boundaries in complex scenes. The results show that DeGMix generalizes effectively to diverse scenarios and consistently outperforms competing methods.

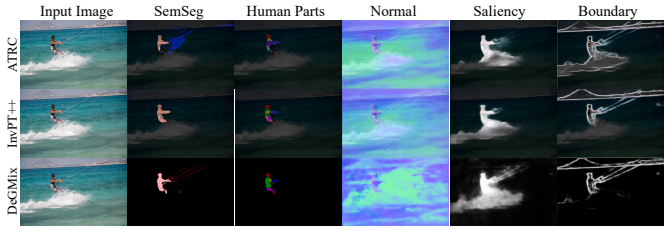


Fig. 8. Visual results of the generated 5-task predictions on DAVIS dataset. We also select the visualization results of ATRC [3] and InvPT++ [23] as a comparison. The model is trained with PASCAL-Context dataset. The qualitative comparison clearly proves the efficacy of our model in capturing detailed segmentation of subtle objects (*e.g.,* the flying lines and control bar).

V. Conclusion

In this work, we introduce DeGMix, a simple and effective approach that unifies deformable CNNs and query-based Transformers with spatial gating for multi-task dense prediction. A key innovation of our method is the use of deformed features, generated by the deformable mixer encoder, as task queries within the task-aware gating transformer decoder. The model employs query-based attention and a gating mechanism to generate dynamically selected task awareness features, enabling efficient learning of multiple related tasks. Extensive experiments on NYUD-v2, PASCAL-Context, and Cityscapes demonstrate that DeGMix achieves superior accuracy with substantially fewer trainable parameters, validating its efficiency and generality. DeGMix surpasses the accuracy of a fully fine-tuned MTL model while requiring substantially fewer trainable parameters.

Limitations and future work. While DeGMix achieves strong multi-task performance, its parameter and computational costs scale with task number and network depth. Future work will explore more efficient designs, such as fine-tuning task-specific subsets to preserve accuracy with lower complexity.

References

[1] S. Vandenhende, S. Georgoulis, L. Van Gool, and L. Van Gool, "Mti-net: Multi-scale task interaction networks for multi-task learning," in *ECCV*, 2020, pp. 527–543.

[2] G. Ghiasi, B. Zoph, E. D. Cubuk, Q. V. Le, and T.-Y. Lin, "Multi-task self-training for learning general representations," in *ICCV*, 2021, pp. 8856–8865.

[3] D. Bruggemann, M. Kanakis, A. Obukhov, S. Georgoulis, and L. V. Gool., "Exploring relational context for multi-task dense prediction." in *ICCV*, 2021, pp. 15 869–15 878.

[4] Y. Xu, X. Li, H. Yuan, Y. Yang, and L. Zhang, "Multi-task learning with multi-query transformer for dense prediction," *IEEE TCSVT*, vol. 34, no. 2, pp. 1228–1240, 2024.

[5] D. Bhattacharjee, T. Zhang, S. Süsstrunk, and M. Salzmann, "Mult: An end-to-end multitask learning transformer," in *CVPR*, 2022, pp. 12 031–12 041.

[6] H. Ye and D. Xu, "Inverted pyramid multi-task transformer for dense scene understanding," in *ECCV*, 2022.

[7] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert, "Cross-stitch networks for multi-task learning," in *CVPR*, 2016, pp. 3994–4003.

[8] X. Xu, H. Zhao, V. Vineet, S.-N. Lim, and A. Torralba, "MTFormer: Multi-task learning via transformer and cross-task reasoning," in *ECCV*, 2022, pp. 304–321.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, vol. 30, 2017.

[10] Y. Gao, J. Ma, M. Zhao, W. Liu, and A. L. Yuille, "NDDR-CNN: Layerwise feature fusing in multi-task cnns by neural discriminative dimensionality reduction," in *CVPR*, 2019, pp. 3205–3214.

[11] Y. Xu, Y. Yang, and L. Zhang, "DeMT: Deformable mixer transformer for multi-task learning of dense prediction," in *AAAI*, 2023, pp. 3072–3080.

[12] 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 *ICCV*, 2021, pp. 10 012–10 022.

[13] Z. Chen, Y. Duan, W. Wang, J. He, T. Lu, J. Dai, and Y. Qiao, "Vision transformer adapter for dense predictions," in *ICLR*, 2023.

[14] X. Zhu, H. Hu, S. Lin, and J. Dai, "Deformable convnets v2: More deformable, better results," in *CVPR*, 2019, pp. 9308–9316.

[15] M. Jia, L. Tang, B.-C. Chen, C. Cardie, S. Belongie, B. Hariharan, and S.-N. Lim, "Visual prompt tuning," in *ECCV*, 2022, pp. 709–727.

[16] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *ECCV*, 2012, pp. 746–760.

[17] X. Chen, R. Mottaghi, X. Liu, S. Fidler, R. Urtasun, and A. Yuille, "Detect what you can: Detecting and representing objects using holistic models and body parts," in *CVPR*, 2014, pp. 1971–1978.

[18] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *CVPR*, 2016, pp. 3213–3223.

[19] S. Vandenhende, S. Georgoulis, W. V. Gansbeke, M. Proesmans, D. Dai, and L. V. Gool, "Multi-task learning for dense prediction tasks: A survey," *IEEE TPAMI*, vol. 44, no. 7, pp. 3614–3633, 2022.

[20] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *CVPR*, 2018, pp. 7482–7491.

[21] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich, "Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks," in *ICML*, 2018, pp. 794–803.

[22] Z. Zhang, Z. Cui, C. Xu, Y. Yan, N. Sebe, and J. Yang, "Pattern-affinitive propagation across depth, surface normal and semantic segmentation," in *CVPR*, 2019, pp. 4106–4115.

[23] H. Ye and D. Xu, "Invpt++: Inverted pyramid multi-task transformer for visual scene understanding," *IEEE TPAMI*, vol. 46, no. 12, pp. 7493–7508, 2024.

[24] 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 *ICLR*, 2021.

[25] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *ICCV*, 2017, pp. 764–773.

[26] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable DETR: deformable transformers for end-to-end object detection," in *ICLR*, 2021.

[27] I. Kokkinos, "Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory," in *CVPR*, 2017, pp. 6129–6138.

[28] R. Ranftl, A. Bochkovskiy, and V. Koltun, "Vision transformers for dense prediction," in *CVPR*, 2021, pp. 12 179–12 188.

[29] X. Dong, J. Bao, D. Chen, W. Zhang, N. Yu, L. Yuan, D. Chen, and B. Guo, "Cswin transformer: A general vision transformer backbone with cross-shaped windows," in *CVPR*, 2022, pp. 12 124–12 134.

[30] A. Jaegle, F. Gimeno, A. Brock, O. Vinyals, A. Zisserman, and J. Carreira, "Perceiver: General perception with iterative attention," in *ICML*, 2021, pp. 4651–4664.

[31] J. Yang, C. Li, X. Dai, and J. Gao, "Focal modulation networks," in *NeurIPS*, vol. 35, 2022, pp. 4203–4217.

[32] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in *ICML*, 2017, pp. 933–941.

[33] D. Xu, W. Ouyang, X. Wang, and N. Sebe, "PAD-Net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing," in *CVPR*, 2018, pp. 675–684.

[34] Z. Ling, C. Zhen, X. Chunyan, Z. Zhenyu, W. Chaoqun, Z. Tong, and Y. Jian, "Pattern-structure diffusion for multi-task learning," in *CVPR*, 2020, pp. 4514–4523.

[35] H. Ye and D. Xu, "Taskprompter: Spatial-channel multi-task prompting for dense scene understanding," in *ICLR*, 2023.

[36] K. Sun, B. Xiao, D. Liu, and J. Wang, "Deep high-resolution representation learning for human pose estimation," in *CVPR*, 2019, pp. 5693–5703.

[37] J. He, C. Zhou, X. Ma, T. Berg-Kirkpatrick, and G. Neubig, "Towards a unified view of parameter-efficient transfer learning," in *ICLR*, 2022.

[38] E. B. Zaken, Y. Goldberg, and S. Ravfogel, "Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models," in *ACL*, 2022, pp. 1–9.

[39] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," in *ICLR*, 2022.

[40] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *CVPR*, 2018, pp. 7482–7491.
[41] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich, "Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks," in *ICML*, 2018, pp. 794–803.
[42] S. Liu, E. Johns, and A. J. Davison, "End-to-end multi-task learning with attention," in *CVPR*, 2019, pp. 1871–1880.
[43] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn, "Gradient surgery for multi-task learning," in *NeurIPS*, vol. 33, 2020, pp. 5824–5836.
[44] Z. Chen, J. Ngiam, Y. Huang, T. Luong, H. Kretzschmar, Y. Chai, and D. Anguelov, "Just pick a sign: Optimizing deep multitask models with gradient sign dropout," in *NeurIPS*, vol. 33, 2020, pp. 2039–2050.
[45] L. Liu, Y. Li, Z. Kuang, J. Xue, Y. Chen, W. Yang, Q. Liao, and W. Zhang, "Towards impartial multi-task learning," in *ICLR*, 2021.
[46] B. Liu, X. Liu, X. Jin, P. Stone, and Q. Liu, "Conflict-averse gradient descent for multi-task learning," in *NeurIPS*, vol. 34, 2021, pp. 18 878–18 890.
[47] Y. Liu, C. Ma, J. Tian, Z. He, and Z. Kira, "Polyhistor: Parameter-efficient multi-task adaptation for dense vision tasks," in *NeurIPS*, 2022.
[48] A. Agiza, M. Neseem, and S. Reda, "Mtlora: A low-rank adaptation approach for efficient multi-task learning," in *CVPR*, 2024, pp. 16 196–16 205.
[49] K. S. I. Mantri, C. Schönlieb, B. Ribeiro, C. Baskin, and M. Eliasof, "Ditask: Multi-task fine-tuning with diffeomorphic transformations," in *CVPR*, 2025, pp. 25 218–25 229.