MTFormer: Multi-task Learning via Transformer and Cross-Task Reasoning

Xiaogang Xu[1], Hengshuang Zhao[2,3](✉), Vibhav Vineet[4], Ser-Nam Lim[5], and Antonio Torralba[2]

[1] CUHK, Hong Kong, China
[2] MIT, Cambridge, USA
hszhao@csail.mit.edu
[3] HKU, Hong Kong, China
[4] Microsoft Research, Redmond, USA
[5] Meta AI, New York, USA

Abstract. In this paper, we explore the advantages of utilizing transformer structures for addressing multi-task learning (MTL). Specifically, we demonstrate that models with transformer structures are more appropriate for MTL than convolutional neural networks (CNNs), and we propose a novel transformer-based architecture named MTFormer for MTL. In the framework, multiple tasks share the same transformer encoder and transformer decoder, and lightweight branches are introduced to harvest task-specific outputs, which increases the MTL performance and reduces the time-space complexity. Furthermore, information from different task domains can benefit each other, and we conduct cross-task reasoning. We propose a cross-task attention mechanism for further boosting the MTL results. The cross-task attention mechanism brings little parameters and computations while introducing extra performance improvements. Besides, we design a self-supervised cross-task contrastive learning algorithm for further boosting the MTL performance. Extensive experiments are conducted on two multi-task learning datasets, on which MTFormer achieves state-of-the-art results with limited network parameters and computations. It also demonstrates significant superiorities for few-shot learning and zero-shot learning.

Keywords: Multi-task learning · Transformer · Cross-task reasoning

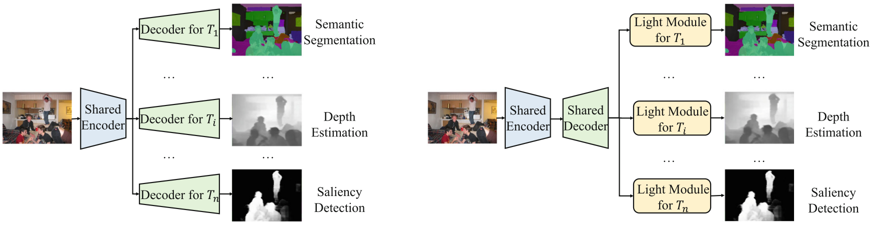
1 Introduction

Multi-task learning (MTL) aims to improve the learning efficiency and accuracy by learning multiple objectives from shared representations [25,31,49]. It is of great importance for practical applications, e.g., autonomous driving [37], healthcare [51], agriculture [52], manufacturing [27], which cannot be addressed by merely seeking perfection on solving individual tasks.

X. Xu and H. Zhao—Indicates equal contribution.

S. Avidan et al. (Eds.): ECCV 2022, LNCS 13687, pp. 304–321, 2022.
https://doi.org/10.1007/978-3-031-19812-0_18

To tackle MTL for visual scene understanding in computer vision, various solutions have been proposed. They simultaneously handle multiple tasks by utilizing classic convolutional neural networks (CNNs) [4,25,31,36,42,49]. These approaches always meet performance drop compared to single-task learning (STL), or they need to add extra intricate loss functions and complex network structures with large network parameters and computations to overcome the performance decrease shortcoming. We own this phenomenon to the limited capacity of convolution operations and seek powerful structures with huge capacity like transformers for handling the MTL problem.



(a) CNN-based MTL Framework (b) Transformer-based MTL Framework

Fig. 1. The differences between current CNN-based MTL framework (a) and our transformer-based MTL framework (b).

In contrast to the classical CNN-based MTL framework [4,25,31,36,42,49], in this work, we find there are significant advantages of utilizing transformers for MTL. We illustrate the differences in Fig. 1(a) and (b). For the traditional framework, as in (a), multiple task-specific decoders are needed to generate task-related predictions, resulting in considerable network parameters and computations. While for our transformer-based MTL framework named *MTFormer* as in (b), both the encoder and decoder parts are constructed based on transformers, and they are shared among different tasks. Only lightweight output branches are utilized for harvesting task-specific outputs. Such a design can vastly reduce the parameter number and inference time. Besides, transformer operations are of higher-order with complex capacity, and our experiments demonstrate they perform even better than STL methods with better performance.

Moreover, information from different tasks can benefit each other, and we further conduct cross-task reasoning to enhance the MTL performance. We propose a cross-task attention mechanism where information inside one task could be utilized to help the predictions of others and vice versa. For classical self-attention-based transformers, both query, key, and value are from the same task representation. Our design introduces similarity maps from other tasks (e.g., the query and key are from another task) for better information aggregation. Such cross-task attention is shown to be general for an arbitrary number of tasks and is proved to be more effective than self-attention.

Furthermore, to fully utilize the cross-task knowledge, we design a self-supervised cross-task contrastive learning algorithm for further enhancing the

MTL performance. As stated in [45], a powerful representation is the one that models view-invariant factors. In the MTL framework, the feature representations of different tasks are views of the same scene. They can be treated as positive pairs, and feature representations from different scenes are negative samples. Therefore, we innovatively propose the cross-task contrastive learning approach to maximize the mutual information between different views of the same scene, resulting in more compact and better feature representations. Simultaneously conducting multi-task learning and contrastive learning could further boost the performance of the MTL system.

Last but not least, we also explore the ability of MTFormer for knowledge transfer under few-shot learning and zero-shot learning settings. With shared feature extractors like shared encoder and decoder, the identical feature representations are expressive enough, and they can be easily transferred for few-shot learning where annotations are limited for specific tasks, and for zero-shot learning where no annotations are available for a dataset and knowledge from other datasets could be transferred. We conduct extensive experiments on two public datasets with various task numbers and task categories, including NYUD-v2 [40] and PASCAL VOC [15], on which our MTFormer harvest the state-of-the-art results on both MTL and knowledge transfer. We give all implementation details and will make our code and trained models publicly available. Our main contribution is three-fold:

– We investigate the advantages of transformers for MTL. We conduct an in-depth analysis and propose a novel MTL architecture named MTFormer, which performs better with smaller parameters and computations.
– We explore cross-task reasoning where both a cross-task attention mechanism and a cross-task contrastive learning algorithm are proposed, which further enhance the performance of MTL.
– We conduct extensive experiments on two competitive datasets. The state-of-the-art performance on MTL and transfer learning demonstrates the effectiveness and generality of the proposed MTFormer.

2 Related Work

Multi-task Learning. Multi-task learning is concerned with learning multiple tasks simultaneously, while exerting shared influence on model parameters [3,4,12,16,17,25,28,29,42,43,56,62]. The potential benefits are manifold and include speed-up training or inference, higher accuracy, and lower parameters.

Many MTL methods perform multiple tasks by a single forward pass, using shared trunk [2,13,26,31,33,47], cross talk [36], or prediction distillation [54,59,60]. A recent work of MTI-Net [49] proposes to utilize the task interactions between multi-scale features. Another stream of MTL is based on task-conditional networks [24,34], which perform a separate forward pass and activate some task-specific modules for each task. Although the transformer-based MTL frameworks have been studied in the language-related domain [21,35,48], existing MTL frameworks for vision tasks mainly adopt CNNs while have not explored the effectiveness of vision transformers.

Vision Transformers. CNNs have dominated the computer vision field for many years and achieved tremendous successes [10,19,20,22,39,44]. Recently, the pioneering work ViT [14] demonstrates that transformer-based architectures can also achieve competitive results. Built upon the success of ViT, many efforts have been devoted to designing better transformer-based architectures for various vision tasks, including low-level image processing [6], image classification [11,18,23,46,50,53], object detection [5,63], semantic segmentation [41,61], depth estimation [38,55], saliency detection [30,57], etc. Rather than concentrating on one special task, some recent works [32,50,58] try to design a general vision transformer backbone for general-purpose vision tasks.

3 Method

In this section, we will first describe the details of our MTFormer in Sect. 3.1. Next, we will detail the cross-task attention in Sect. 3.2. Furthermore, the self-supervised cross-task contrastive learning algorithm and the final loss function of the framework will be depicted in Sects. 3.3 and 3.4.

3.1 MTFormer

Our MTL framework consists of only transformer blocks, and its visual illustration is shown in Fig. 2. It consists of two parts, i.e., the shared feature extractor and the lightweight task-specific branches. Their details are summarized below.

Shared Feature Extractor. The shared feature extractor consists of an encoder and a decoder. As illustrated in Fig. 2, the shared encoder is built based on a pre-trained transformer with a stack of down-sampling operations, e.g., the Swin-Transformer [32]. And the shared decoder consists of a stack of shared transformer blocks (with self-task attention only) with a flexible module design.

Lightweight Task-Specific Branches. The task-specific branch consists of two parts, i.e., the transformer-based feature transformation branch and the output head with non-linear projection. For different tasks, its related feature transformation follows the shared decoder, consisting of only a few transformer blocks thus is lightweight. The first part in each transformation branch includes only self-task attention modules to obtain unique representations. And the second part is a stack of transformer blocks with cross-task attention that will be detailed in Sect. 3.2. At the end of each branch, an output head with a non-linear projection is utilized to harvest the final prediction for the related task.

Self-task Attention for MTL. Suppose $T_1, ..., T_h$ are h input feature tokens, we flatten $T_1, ..., T_h$ into 1D features, and the transformer block with self-task attention is processed as

$$
\begin{aligned}
&q = k = v = LN([T_1, ..., T_h]),\ [\widehat{T}_1, ..., \widehat{T}_h] = MSA(q, k, v) + [T_1, ..., T_h], \\
&[\mathcal{T}_1, ..., \mathcal{T}_m] = FFN(LN([\widehat{T}_1, ..., \widehat{T}_h])) + [\widehat{T}_1, ..., \widehat{T}_h],
\end{aligned} \tag{1}
$$

where LN denotes layer normalization, MSA denotes the multi-head self-attention module, q, k, and v denote the query, key, and value vector to complete the computation of MSA, FFN represents the forward module in the transformer block.

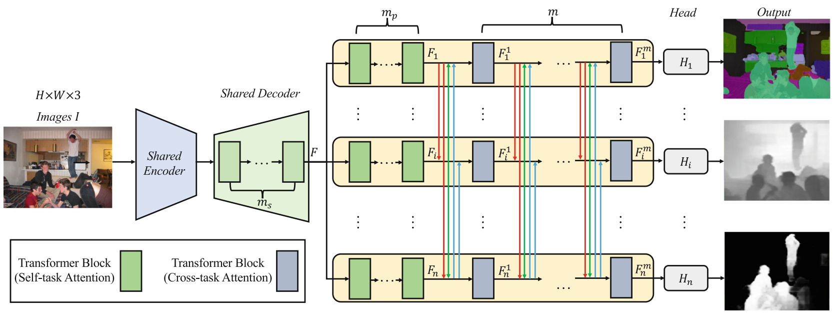


Fig. 2. The illustration of the proposed MTFormer framework.

Now we describe the self-task attention computation process, e.g., the $MSA()$. Suppose the head number is B for MSA, the b-th head self-attention calculation $Attention_b$ in one transformer block is formulated as

$$\mathbf{Q}_b = qW_b^q, \mathbf{K}_b = kW_b^k, \mathbf{V}_b = vW_b^v, \;\; Attention_b(\mathbf{Q}_b, \mathbf{K}_b, \mathbf{V}_b) = SoftMax(\frac{\mathbf{Q}_b \mathbf{K}_b^T}{\sqrt{d_b}})\mathbf{V}_b, \tag{2}$$

where q, k, v are features in Eq. 1; W_b^q, W_b^k, W_b^v represent the projection matrices for the b-th head; $\mathbf{Q}_b$, $\mathbf{K}_b$, $\mathbf{V}_b$ are the projected query, key, and value features, respectively. The cross-task attention for MTL will be introduced in Sect. 3.3.

Superiority Analysis. We summarize the structure of current MTL structures with CNNs and our transformer-only framework (MTFormer) as displayed in Fig. 1. The main difference is that the MTL framework with transformer allows the deep shared network module, including the shared encoder as well as the shared decoder. And as suggested by recent works [11,18,23,41,46,50,53,61], the transformer has strength in the computation of long-range attention and the general representation ability for various tasks. Such advantage is essential since the individual branches start from the same feature in MTL and need various long-range attention and a joint representation strategy.

Especially, as proved in Sect. 4.2, these advantages in the structures lead to significant superiority. 1) The superiority in the performance: combined with the simple MTL loss function (e.g., the uncertainty loss in [25]), the CNN-based MTL has worse performance than STL. On the other hand, the performance of the transformer-only MTL is better than STL on all tasks. 2) The superiority in the model parameters: we find that transformer-only MTL has a smaller parameter number ratio between MTL and STL, meaning that the superiority of the model parameter is more prominent for transformer-only MTL. This is because different tasks in MTFormer have a deeply shared backbone.

3.2 Cross-Task Attention for MTL

To achieve feature propagation across different tasks, previous methods mainly build the inter-task connections among different task-specific branches [36,49]. However, such a propagation mechanism has two significant drawbacks. First, different branches need to learn the task-common feature besides the task-specific part, which will impede the learning of each branch in MTL compared with STL since STL just needs to learn the task-specific feature. Second, the connections between different branches will increase the parameter number of the MTL system. To achieve the feature propagation among different tasks efficiently, we propose the cross-task attention, as shown in Fig. 3, novelly modifying the attention computation mechanism in the transformer block by merging attention maps from different tasks.

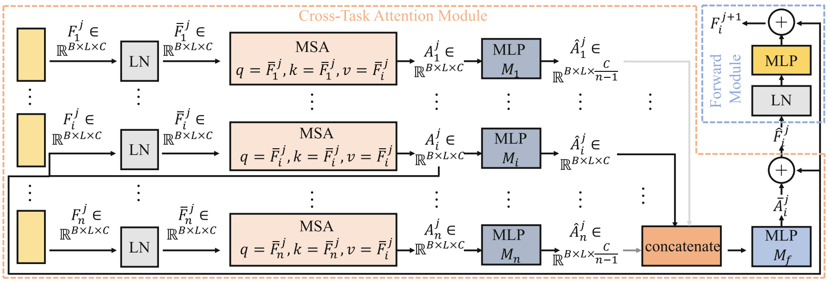


Fig. 3. Detailed structure of the proposed cross-task attention for MTL.

As shown in Fig. 3, suppose there are n tasks and thus corresponding n task features. The input features of the j-th transformer block with cross-task attention are denoted as $F_1^j, ..., F_n^j$. Without loss of generality, suppose we aim to compute the cross-task attention for F_i^j (the computations of cross-task attention for $F_1^j, ..., F_{i-1}^j, F_{i+1}^j, ..., F_n^j$ are similar). For $F_1^j, ..., F_n^j$, we can obtain n attention maps as $A_1^j, ..., A_n^j$, and how to fuse different attention maps, i.e., self-task attention map and the other attention maps, is the vital problem in the cross-task attention module. The combination of different attention maps can be achieved with n mapping functions (M_1,...,M_n) to adjust the dimension and one MLP (M_f) to fuse the adjusted outputs (as shown in Fig. 3), as

$$
\begin{aligned}
\bar{F}_1^j &= \mathrm{LN}(F_1^j), ..., \bar{F}_n^j = \mathrm{LN}(F_n^j), \\
A_1^j &= MSA(\bar{F}_1^j, \bar{F}_1^j, \bar{F}_i^j), ..., A_n^j = MSA(\bar{F}_n^j, \bar{F}_n^j, \bar{F}_i^j), \\
\widehat{A}_1^j &= M_1(A_1^j), ..., \widehat{A}_n^j = M_n(A_n^j), \\
\bar{A}_i^j &= M_f([\widehat{A}_1^j, ..., \widehat{A}_n^j]).
\end{aligned}
\tag{3}
$$

Especially, we find that the self-task attention output should take the primary role ($\widehat{A}_i^j \in \mathbb{R}^{B \times L \times C}$) while the cross-task attention outputs should take the auxiliary role (with smaller feature channel number, as $B \times L \times \frac{C}{n-1}$). This is verified by the ablation experiments in Sect. 4.

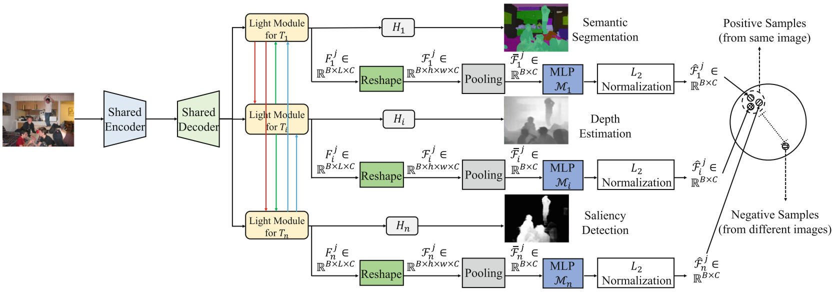


Fig. 4. Framework of self-supervised cross-task contrastive learning for MTL.

3.3 Cross-Task Contrastive Learning

As stated in [45], a powerful representation is one that models view-invariant factors. In MTFormer, the outputs are indeed multiple views of an input image, e.g., the depth value and semantic segmentation of the image on NYUD-v2 [40]. Therefore, we design a contrastive learning strategy by adopting the feature representations of different tasks for the same scene as positive pairs, and the representations from different scenes as negative pairs.

Suppose there are n tasks, and we take every two of them for contrastive learning. We set the features obtained from the intermediate representation in the task-specific branch as the inputs to compute the contrastive loss. The details of the contrastive loss can be viewed in Fig. 4. As we can see, for the intermediate features $F_1^j, ..., F_n^j$, we apply the global average pooling operation, and then a set of mapping functions for the processing, which is combined with an L_2 normalization operation. Note that the global average pooling operation, mapping functions, and the L_2 normalization operation are no longer needed during the inference of MTL. Suppose there are D negative samples, the loss for the task y and task z (the contrastive loss is computed for every two tasks) is

$$\mathcal{L}^{y}_{contrast} = -\mathbb{E}[\log \frac{g(\widehat{\mathcal{F}}_y, \widehat{\mathcal{F}}_z)}{\sum_{d=1}^{D} g(\widehat{\mathcal{F}}_y, \widehat{\mathcal{F}}_{z,d})}], \; \mathcal{L}^{z}_{contrast} = -\mathbb{E}[\log \frac{g(\widehat{\mathcal{F}}_z, \widehat{\mathcal{F}}_y)}{\sum_{d=1}^{D} g(\widehat{\mathcal{F}}_z, \widehat{\mathcal{F}}_{y,d})}], \tag{4}$$

$$\mathcal{L}^{y,z}_{contrast} = \mathcal{L}^{y}_{contrast} + \mathcal{L}^{z}_{contrast}, \; g(\widehat{\mathcal{F}}_y, \widehat{\mathcal{F}}_z) = exp(\frac{\widehat{\mathcal{F}}_y \cdot \widehat{\mathcal{F}}_z}{||\widehat{\mathcal{F}}_y|| \times ||\widehat{\mathcal{F}}_z||}), \tag{5}$$

where $g()$ is the function to measure similarity, $\mathbb{E}$ is the average operation, $\widehat{\mathcal{F}}_{z,d}$ and $\widehat{\mathcal{F}}_{y,d}$ are the d-th negative sample for $\widehat{\mathcal{F}}_z$ and $\widehat{\mathcal{F}}_y$, respectively. And the overall contrastive loss can be written as

$$\mathcal{L}_{contrast} = \sum_{1 \leq y \leq n, 1 \leq z \leq n} \mathcal{L}^{y,z}_{contrast}. \tag{6}$$

3.4 Loss Function

Different from existing MTL methods, our framework can achieve SOTA performance on different datasets without complex loss functions. We utilized the classical MTL loss function, which weighs multiple loss functions by considering the homoscedastic uncertainty of each task [25]. To implement such loss, we add the trainable values $\sigma_1, ..., \sigma_n$ to estimate the uncertainty of each task. And the final loss function can be written as

$$\mathcal{L}(\sigma_1, ..., \sigma_n) = \frac{1}{r_1\sigma_1^2}\mathcal{L}_1 + ... + \frac{1}{r_2\sigma_n^2}\mathcal{L}_n + \log\sigma_1 + ... + \log\sigma_n + \mathcal{L}_{contrast}, \tag{7}$$

where $\mathcal{L}_1$ to $\mathcal{L}_n$ are n loss functions for n different tasks, r_1 to r_n belongs to $\{1, 2\}$ and their values are decided whether the corresponding outputs are modeled with Gaussian likelihood or softmax likelihood [25].

4 Experiments

4.1 Experimental Setting

Dataset. We follow the experimental setting of the recent MTL method [49], and perform the experimental evaluations on two competitive datasets, i.e. NYUD-v2 [40] and PASCAL VOC [15]. The NYUD-v2 dataset contains both semantic segmentation and depth estimation tasks, and it has 1449 images in total, with 795 images for training and the remaining 654 images for validation. For the PASCAL dataset, we use the split from PASCAL-Context [9], which has annotations for semantic segmentation, human part segmentation, and composited saliency labels from [34] that are distilled from pre-trained state-of-the-art models [1,8]. The dataset contains 10103 images, with 4998 and 5105 images for training and validation, respectively.

Evaluation Metric. The semantic segmentation, saliency estimation, and human part segmentation tasks are evaluated with the mean intersection over union (mIoU). The depth estimation task is evaluated using the root mean square error (rmse). Besides the metric for each task, we measure the multi-task learning performance Δ_m as in [34], where the MTL performance is defined as $\Delta_m = \frac{1}{n}\sum_{i=1}^{n}(-1)^{l_i}(M_{m,i} - M_{s,i})/M_{s,i}$, where $M_{m,i}$ and $M_{s,i}$ are the MTL and STL performance on the i-th task, $l_i = 1$ if a lower value means better performance for task i, and 0 otherwise.

Implementation Detail. Theoretically speaking, the individual branch for each task can have an arbitrary number of cross-task attention modules. More cross-task attention modules mean more feature propagation, leading to better MTL performance. However, due to the limitation of computation resources, we only set $m = m_s = m_p = 2$ in the experiments. We adopt the Swin-Transformer [32] as the shared encoder for transformer-based MTL, ResNet50 [19] for CNN-based MTL. And for the transformer block in the decoder, we employ the strategy of W-MSA and SW-MSA in the Swin-Transformer [32] as the MSA module. For the individual branch in the CNN-based transformer, we use the ASPP in [7]. For the baseline STL with transformer, it consists of the shared encoder, decoder, one lightweight branch, and one head in Fig. 2.

Table 1. Results on NYUD-v2 of STL and MTL with CNN (C) and transformer (T). Cross-task attention and contrastive learning are not utilized.

Method	Seg↑	Dep↓	$\delta_m\%$ ↑
STL (C)	**43.11**	**0.507**	+0.00
MTL (C)	42.05	0.521	–2.61
STL (T)	47.96	0.497	+0.00
MTL (T)	**50.04**	**0.490**	+2.87

Table 2. Results on PASCAL of STL and MTL with CNN (C) and transformer (T). Cross-task attention and contrastive learning are not utilized.

Method	Seg↑	Part↑	Sal↑	$\delta_m\%$ ↑
STL (C)	**69.06**	**62.12**	**66.42**	+0.00
MTL (C)	61.87	60.97	64.68	–4.96
STL (T)	71.17	63.90	66.71	+0.00
MTL (T)	**73.52**	**64.26**	**67.24**	+1.55

Table 3. Resource analysis for Table 1. R-Params and R-FLOPS mean the ratio with the Params and FLOPs of STL.

Method	Params (M)	FLOPS (G)	R-Params	R-FLOPS
STL (C)	79.27	384.35	1.0	1.0
MTL (C)	**55.77**	**267.10**	0.704	0.695
STL (T)	114.26	157.52	1.0	1.0
MTL (T)	**62.45**	**94.62**	0.547	0.639

Table 4. Resource analysis for Table 2. R-Params and R-FLOPS mean the ratio with the Params and FLOPs of STL.

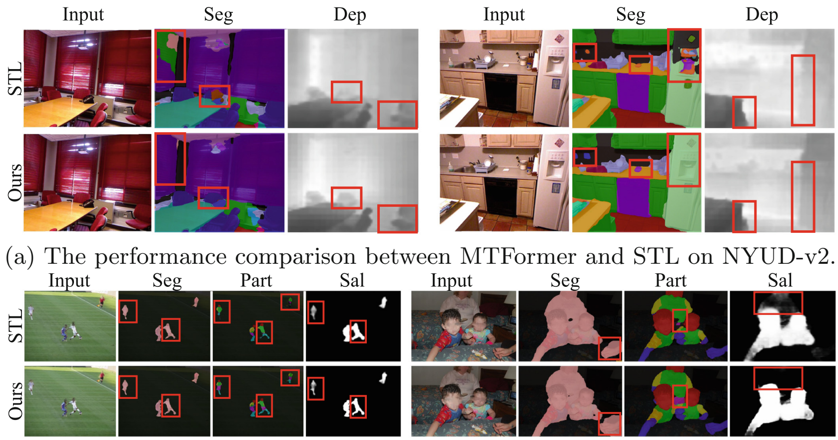
Method	Params (M)	FLOPS (G)	R-Params	R-FLOPS
STL (C)	118.91	491.93	1.0	1.0
MTL (C)	**71.89**	**291.83**	0.605	0.593
STL (T)	171.43	201.76	1.0	1.0
MTL (T)	**67.80**	**94.42**	0.395	0.519

4.2 MTFormer Superiority

The results of CNN-based and transformer-based STL/MTL frameworks on NYUD-v2 and PASCAL are shown in Tables 1 and 2, respectively. As we can see, combined with the simple MTL loss, the CNN-based MTL has worse performance than STL, while transformer-based MTL algorithm MTL (T) (a.k.a, vanilla MTFormer without cross-task reasoning) is better than STL on all tasks. Specifically, on NYUD-v2 and PASCAL, we observe a significant decrease of 2.61 and 4.96% points from STL to MTL for the CNN-based framework, while an improvement of 2.87 and 1.55% points from STL to MTL for the transformer-based framework MTFormer. We provide the qualitative analysis in Fig. 5 by providing the visual comparison for the transformer-based STL and the proposed MTFormer framework. As we can see, our MTFormer can bring noticeable improvement on all tasks compared with STL. We also conduct the resource analysis by computing the number of parameters, and computation FLOPS for different models, as displayed in Tables 3 and 4. Significantly, for the ratio between MTL's parameter-number/FLOPS and STL's parameter-number/FLOPS, the transformer-based MTL has a smaller value. Therefore, the transformer-based MTL has a more prominent advantage in reducing model parameters and computations w.r.t. STL. In conclusion, transformers have a larger capacity and are more suitable for MTL, and our proposed MTFormer achieves the best results with reduced parameters and computations.

4.3 Cross-Task Reasoning

The information inside different task domains can benefit the understanding of each other, and we conduct cross-task reasoning, which includes cross-task



(a) The performance comparison between MTFormer and STL on NYUD-v2.

(b) The performance comparison between MTFormer and STL on PASCAL.

Fig. 5. Visual comparisons between the predictions of STL and our MTFormer. Regions highlighted by red rectangle show more clear differences. (Color figure online)

attention and cross-task contrastive learning. To demonstrate the effectiveness of our designed algorithms, we compare the results of the proposed MTFormer framework with self-task attention or cross-task attention, and with or without cross-task contrastive learning. The results on NYUD-v2 and PASCAL are shown in Tables 5 and 6 respectively.

Cross-Task Attention. Our cross-task attention mechanism is designed that attention maps and similarity relationships from other tasks are introduced to help the prediction of the current task. Comparing the two methods named "Ours wo CA&CL" and "Ours wo CL" in Tables 5 and 6, we can see that method with cross-task attention can get extra 0.5% points improvement for δ_m, which proves the effectiveness of the cross-task attention. We compute the parameter number/FLOPS for MTL frameworks with cross-task attention or with self-task attention only, and the results are shown in Tables 7 and 8. Compared with the framework with only self-task attention ("Ours w/o CA&CL"), the framework with cross-task attention ("Ours w/o CL") only increases a small number of model parameters (2.5% for NYUD-v2, 9.3% for PASCAL).

Ablation Study. To fuse the attention maps from different tasks, we first utilize feature mapping functions ($M_1, ...M_n$) for attention outputs and then use an MLP (M_f) for feature fusion. Alternatively, we can use only an MLP instead of the mapping functions, and this setting is called "Ours w/o FM&CL"; if we remove the MLP for fusion, we can just add the outputs from mapping functions for fusion, and this setting is denoted as "Ours w/o FF&CL". The results are shown in Tables 5 and 6, and they are all weaker than our original strategy. As

Table 5. Cross-task reasoning on NYUD-v2. 'CA' and 'CL' stand for cross-task attention and contrastive learning, 'FM', 'FF', and 'FB' denote feature mapping, fusion, and balance.

Method	Seg↑	Dep↓	δ_m% ↑
STL (T)	47.96	0.497	+0.00
Ours w/o CA&CL	50.04	0.490	+2.87
Ours w/o FM&CL	50.34	0.487	+3.49
Ours w/o FF&CL	50.33	0.488	+3.38
Ours w/o FB&CL	–	–	–
Ours w/o CL	50.31	0.486	+3.56
Ours (MTFormer)	**50.56**	**0.483**	+4.12

Table 6. Cross-task reasoning on PASCAL. 'CA' and 'CL' stand for cross-task attention and contrastive learning, 'FM', 'FF', and 'FB' denote feature mapping, fusion, and balance.

Method	Seg↑	Part↑	Sal↑	δ_m% ↑
STL (T)	71.17	63.90	66.71	+0.00
Ours w/o CA&CL	73.52	64.26	67.24	+1.55
Ours w/o FM&CL	73.74	64.37	66.97	+1.58
Ours w/o FF&CL	73.84	64.42	67.14	+1.74
Ours w/o FB&CL	73.98	64.41	66.95	+1.70
Ours w/o CL	73.77	64.47	67.49	+1.91
Ours (MTFormer)	**74.15**	**64.89**	**67.71**	+2.41

Table 7. Resource analysis for Table 5.

Method	Params (M)	FLOPS (G)
Ours w/o CA&CL	62.45	94.62
Ours w/o CL	64.03	117.73

Table 8. Resource analysis for Table 6.

Method	Params (M)	FLOPS (G)
Ours w/o CA&CL	67.80	94.42
Ours w/o CL	74.12	128.77

stated in Sect. 3.2, the self-task attention output takes the primary role while the cross-task attention outputs take the auxiliary position. And we have feature dimension adaptation to balance their contributions. We conduct experiments to prove this claim by setting the feature channel of all attention outputs as $B \times L \times C$, and then use MLP for fusion. This setting is represented as "Ours w/o FB&CL", and the results are shown in Table 6, demonstrating the correctness of our claim since a decrease of 0.2% points for δ_m is caused by adopting "Ours w/o FB&CL" compared with "Ours w/o CL".

Cross-Task Contrastive Learning. We conduct cross-task contrastive learning to further enhance the MTL performance. We combine the contrastive loss with the supervised loss for MTL and optimize these two loss terms simultaneously. The results are in Tables 5 and 6 and our framework with contrastive learning is called "Ours (MTFormer)". We can see that extra cross-task contrastive learning introduces 0.56 and 0.50% points improvements for δ_m on NYUD-v2 and PASCAL, compared to those without cross-task contrastive learning ("Ours wo CL"). These results reveal the effectiveness of the proposed cross-task contrastive learning algorithm.

4.4 Comparison with Others

We conduct method comparisons with existing state-of-the-art MTL frameworks, which adopt various complex network structures and loss terms. The compared

Table 9. Results on NYUD-v2 for comparison with SOTA MTL methods.

Method	Seg↑	Dep↓	$\delta_m\%\uparrow$
STL (T)	47.96	0.497	+0.00
AST [34]	42.16	0.570	–13.39
Auto [3]	41.10	0.541	–11.58
Cross-stitch [36]	41.01	0.538	–11.37
NDDR-CNN [17]	40.88	0.536	–11.30
MTL-A [31]	42.03	0.519	–8.40
Repara [24]	43.22	0.521	–7.36
PAD-Net [54]	50.20	0.582	–6.22
ERC [4]	46.33	0.536	–5.62
Switching [42]	45.90	0.527	–5.17
MTI-Net [49]	49.00	0.529	–2.14
MTFormer	**50.56**	**0.483**	+4.12

Table 10. Results on PASCAL for comparison with SOTA MTL methods.

Method	Seg↑	Part↑	Sal↑	$\delta_m\%\uparrow$
STL (T)	71.17	63.90	66.71	+0.00
Repara [24]	56.63	55.85	59.32	–14.70
Switching [42]	64.20	55.03	63.31	–9.59
NDDR-CNN [17]	63.22	56.12	65.16	–8.56
MTL-A [31]	61.55	58.89	64.96	–7.99
Auto [3]	64.07	58.60	64.92	–6.99
PAD-Net [54]	60.12	60.70	67.20	–6.60
Cross-stitch [36]	63.28	60.21	65.13	–6.41
ERC [4]	62.69	59.42	67.94	–5.70
AST [34]	68.00	61.12	66.10	–3.24
MTI-Net [49]	64.98	62.90	**67.84**	–2.86
MTFormer	**74.15**	**64.89**	**67.71**	+2.41

Table 11. Results on PASCAL of few-shot learning with different tasks.

Method	Few-shot data	Seg↑	Part↑	Sal↑	$\delta_m\%\uparrow$
Single task	Seg	3.34	63.90	66.71	+0.00
Ours	Seg	**35.26**	**64.26**	**67.26**	+ 319.03
Single task	Part	71.17	11.27	66.71	+0.00
Ours	Part	**73.36**	**51.74**	**67.64**	+121.19
Single task	Sal	71.17	63.90	44.39	+0.00
Ours	Sal	**76.00**	**66.89**	**55.55**	+12.20

approaches on NYUD-v2 and PASCAL are MTL-A [31], Cross-stitch [36], MTI-Net [49], Switching [42], ERC [4], NDDR-CNN [17], PAD-Net [54], Repara [24], AST [34], and Auto [3]. The comparison results between our MTFormer framework and all the others on NYUD-v2 are shown in Table 9. It can be seen that our MTFormer's results are superior to the baselines in terms of both semantic segmentation and depth estimation results. The comparisons on PASCAL, as displayed in Table 10, also demonstrate the effectiveness of our MTFormer.

4.5 MTFormer for Few-Shot Learning

In Natural Language Processing (NLP) applications [35], it is observed that MTL's improvements (compared to STL) are usually focused on tasks that have fewer training samples. Here we explore the ability of MTFormer for transfer learning. And we demonstrate that our MTFormer can boost the few-shot learning performance on vision tasks without complex few-shot learning loss, which is due to the beneficial feature propagation among different tasks.

We take PASCAL dataset as an example. Annotating images with the accurate human segmentation ground truth label is more accessible than the human part segmentation, since human part segmentation needs more details. Therefore, we can set the human part segmentation's annotation as the few-shot samples. Specifically, for the PASCAL dataset, we take all the annotations for semantic segmentation and saliency detection, while randomly sampling only about 1% (40 out of 4998) human part segmentation annotations. We set the baseline as the STL with such few-shot samples. As shown in Tables 11, our MTFormer can significantly improve the performance on the few-shot learning task compared with STL (i.e., the accuracy is improved more than 40% points for human part segmentation), while keeping the performance on other tasks almost unchanged (compared to the results in Tables 10). And the few-shot learning settings for other tasks are also included in Tables 11. This shows MTFormer's strong ability to handle few-shot learning problems.

Table 12. Simultaneously training multiple datasets for zero-shot learning ('ZSL') does not affect the performance much.

Method	Seg↑ (NYUD-v2)	Dep↓ (NYUD-v2)	Seg↑ (PASCAL)	Part↑ (PASCAL)	Sal↑ (PASCAL)
Ours w/o ZSL	50.03	0.488	73.70	64.36	67.82
Ours w ZSL	48.26	0.480	70.18	61.47	67.59

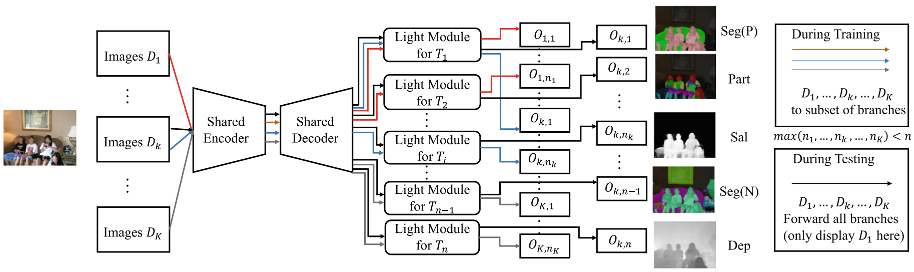


Fig. 6. Our MTFormer framework can be utilized for achieving zero-shot learning.

4.6 MTFormer for Zero-Shot Learning

Further, our MTFormer can also perform well on zero-shot learning. As exhibited in Fig. 6, MTFormer can be utilized to transfer the knowledge of one dataset to another dataset. Take the dataset of NYUD-v2 and PASCAL as an example. The NYUD-v2 dataset has the annotation of semantic segmentation and depth, and the PASCAL dataset has the annotation of semantic segmentation, human part segmentation, and saliency. We can simultaneously use the data of NYUD-v2 and PASCAL to train the proposed MTFormer, whose output includes semantic segmentation, depth estimation, human part segmentation, and saliency detection.

In the framework, we use the annotation of each dataset to train the corresponding output branch.

We are surprised to find that the trained framework can have comparable performances on the tasks which have annotations as shown in Table 12, i.e., the semantic segmentation and depth prediction on NYUD, the semantic segmentation, human part segmentation, and saliency detection on PASCAL. Meanwhile, the trained network can predict the outputs with no annotations, e.g., the saliency detection results on NYUD and the depth prediction on PASCAL VOC, as displayed in Fig. 7.

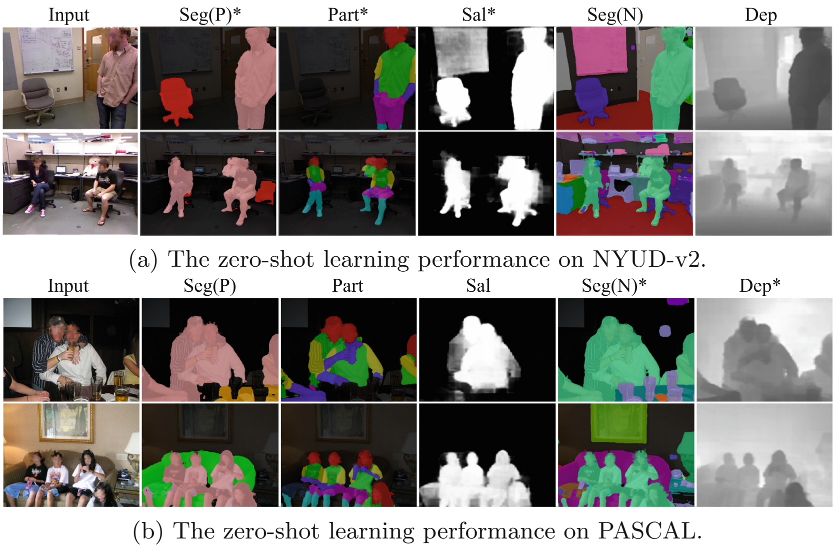


Fig. 7. Visual illustration of MTFormer for zero-shot learning on PASCAL and NYUD-v2. 'Seg(P)' and 'Seg(N)' denote the segmentation with PASCAL class and NYUD-v2 class, respectively. '*' means there is no training ground truth.

Such great property can be achieved because different tasks in our framework have a deeply shared backbone, and the branches for individual tasks are lightweight. Thus, different tasks can have a shared representation even with samples from various datasets. Besides, combined with our cross-task attention, the feature propagation can be implemented across different tasks, which also contributes to the zero-shot learning for the MTL framework.

5 Conclusion

In this paper, we first explore the superiority of using transformer structures for MTL and propose a transformer-based MTL framework named MTFormer. It is proved that MTL with deeply shared network parameters for different tasks can better reduce the time-space complexity and increase the performance compared with STL. Moreover, we also conduct cross-task reasoning and propose the cross-task attention mechanism to improve the MTL results, which can achieve effective feature propagation among different tasks. Besides, a contrastive learning algorithm is proposed to further enhance the MTL results. Extensive experiments on NYUD-v2 and PASCAL show that the proposed MTFormer can achieve state-of-the-art performance with fewer parameters and computations. And MTFormer also shows great superiorities for transfer learning tasks.

References

1. Bansal, A., Chen, X., Russell, B., Gupta, A., Ramanan, D.: Pixelnet: representation of the pixels, by the pixels, and for the pixels. arXiv:1702.06506 (2017)
2. Bragman, F.J., Tanno, R., Ourselin, S., Alexander, D.C., Cardoso, J.: Stochastic filter groups for multi-task cnns: learning specialist and generalist convolution kernels. In: ICCV (2019)
3. Bruggemann, D., Kanakis, M., Georgoulis, S., Van Gool, L.: Automated search for resource-efficient branched multi-task networks. In: BMVC (2020)
4. Bruggemann, D., Kanakis, M., Obukhov, A., Georgoulis, S., Van Gool, L.: Exploring relational context for multi-task dense prediction. In: ICCV (2021)
5. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-End object detection with transformers. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12346, pp. 213–229. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58452-8_13
6. Chen, H., et al.: Pre-trained image processing transformer. In: CVPR (2021)
7. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. IEEE TPAMI **40**, 834–848 (2017)
8. Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11211, pp. 833–851. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01234-2_49
9. 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: CVPR (2014)
10. Chen, Y., Li, J., Xiao, H., Jin, X., Yan, S., Feng, J.: Dual path networks. arXiv:1707.01629 (2017)
11. Chu, X., et al.: Twins: revisiting spatial attention design in vision transformers. arXiv:2104.13840 (2021)
12. Crawshaw, M.: Multi-task learning with deep neural networks: a survey. arXiv:2009.09796 (2020)
13. Doersch, C., Zisserman, A.: Multi-task self-supervised visual learning. In: ICCV (2017)
14. Dosovitskiy, A., et al.: An image is worth 16×16 words: transformers for image recognition at scale. arXiv:2010.11929 (2020)
15. 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)
16. Gao, Y., Bai, H., Jie, Z., Ma, J., Jia, K., Liu, W.: Mtl-nas: task-agnostic neural architecture search towards general-purpose multi-task learning. In: CVPR (2020)

17. Gao, Y., Ma, J., Zhao, M., Liu, W., Yuille, A.L.: Nddr-cnn: layerwise feature fusing in multi-task cnns by neural discriminative dimensionality reduction. In: CVPR (2019)
18. Han, K., Xiao, A., Wu, E., Guo, J., Xu, C., Wang, Y.: Transformer in transformer. arXiv:2103.00112 (2021)
19. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
20. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: CVPR (2018)
21. Hu, R., Singh, A.: Unit: multimodal multitask learning with a unified transformer. In: ICCV (2021)
22. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: CVPR (2017)
23. Jiang, Z., et al.: Token labeling: training a 85.5% top-1 accuracy vision transformer with 56m parameters on imagenet. arXiv:2104.10858 (2021)
24. Kanakis, M., Bruggemann, D., Saha, S., Georgoulis, S., Obukhov, A., Van Gool, L.: Reparameterizing convolutions for incremental multi-task learning without task interference. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12365, pp. 689–707. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58565-5_41
25. Kendall, A., Gal, Y., Cipolla, R.: Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In: CVPR (2018)
26. Kokkinos, I.: Ubernet: training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In: CVPR (2017)
27. Li, Y., Yan, H., Jin, R.: Multi-task learning with latent variation decomposition for multivariate responses in a manufacturing network. IEEE Trans. Autom. Sci. Eng. (2022)
28. Liu, B., Liu, X., Jin, X., Stone, P., Liu, Q.: Conflict-averse gradient descent for multi-task learning. In: NIPS (2021)
29. Liu, L., et al.: Towards impartial multi-task learning. In: ICLR (2020)
30. Liu, N., Zhang, N., Wan, K., Shao, L., Han, J.: Visual saliency transformer. In: ICCV (2021)
31. Liu, S., Johns, E., Davison, A.J.: End-to-end multi-task learning with attention. In: CVPR (2019)
32. Liu, Z., et al.: Swin transformer: hierarchical vision transformer using shifted windows. In: ICCV (2021)
33. Lu, Y., Kumar, A., Zhai, S., Cheng, Y., Javidi, T., Feris, R.: Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. In: CVPR (2017)
34. Maninis, K.K., Radosavovic, I., Kokkinos, I.: Attentive single-tasking of multiple tasks. In: CVPR (2019)
35. McCann, B., Keskar, N.S., Xiong, C., Socher, R.: The natural language decathlon: multitask learning as question answering. arXiv:1806.08730 (2018)
36. Misra, I., Shrivastava, A., Gupta, A., Hebert, M.: Cross-stitch networks for multi-task learning. In: CVPR (2016)
37. Muhammad, K., Ullah, A., Lloret, J., Del Ser, J., de Albuquerque, V.H.C.: Deep learning for safe autonomous driving: current challenges and future directions. IEEE Trans. Intell. Transp. Syst. **22**, 4316–4336 (2020)
38. Ranftl, R., Bochkovskiy, A., Koltun, V.: Vision transformers for dense prediction. In: CVPR (2021)

39. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv 2: inverted residuals and linear bottlenecks. In: CVPR (2018)
40. 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
41. Strudel, R., Garcia, R., Laptev, I., Schmid, C.: Segmenter: transformer for semantic segmentation. arXiv:2105.05633 (2021)
42. Sun, G., et al.: Task switching network for multi-task learning. In: ICCV (2021)
43. Sun, X., Panda, R., Feris, R., Saenko, K.: Adashare: learning what to share for efficient deep multi-task learning. In: NIPS (2020)
44. Tan, M., Le, Q.: Efficientnet: rethinking model scaling for convolutional neural networks. In: ICML, pp. 6105–6114 (2019)
45. Tian, Y., Krishnan, D., Isola, P.: Contrastive multiview coding. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12356, pp. 776–794. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58621-8_45
46. Touvron, H., Cord, M., Sablayrolles, A., Synnaeve, G., Jégou, H.: Going deeper with image transformers. arXiv:2103.17239 (2021)
47. Vandenhende, S., Georgoulis, S., De Brabandere, B., Van Gool, L.: Branched multi-task networks: deciding what layers to share. In: BMVC (2019)
48. Vandenhende, S., Georgoulis, S., Van Gansbeke, W., Proesmans, M., Dai, D., Van Gool, L.: Multi-task learning for dense prediction tasks: a survey. IEEE TPAMI **44**, 3614–3633 (2021)
49. 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
50. Wang, W., et al.: Pyramid vision transformer: a versatile backbone for dense prediction without convolutions. arXiv:2102.12122 (2021)
51. Wang, W., et al.: Graph-driven generative models for heterogeneous multi-task learning. In: AAAI (2020)
52. Wen, C., et al.: Multi-scene citrus detection based on multi-task deep learning network. In: 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC) (2020)
53. Wu, H., et al.: Cvt: introducing convolutions to vision transformers. arXiv:2103.15808 (2021)
54. 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: CVPR (2018)
55. Yang, G., Tang, H., Ding, M., Sebe, N., Ricci, E.: Transformer-based attention networks for continuous pixel-wise prediction. In: ICCV (2021)
56. Yu, T., Kumar, S., Gupta, A., Levine, S., Hausman, K., Finn, C.: Gradient surgery for multi-task learning. In: NIPS (2020)
57. Zhang, J., Xie, J., Barnes, N., Li, P.: Learning generative vision transformer with energy-based latent space for saliency prediction. In: NIPS (2021)
58. Zhang, P., et al.: Multi-scale vision longformer: a new vision transformer for high-resolution image encoding. arXiv:2103.15358 (2021)
59. Zhang, Z., Cui, Z., Xu, C., Jie, Z., Li, X., Yang, J.: Joint task-recursive learning for semantic segmentation and depth estimation. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11214, pp. 238–255. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01249-6_15

60. Zhang, Z., Cui, Z., Xu, C., Yan, Y., Sebe, N., Yang, J.: Pattern-affinitive propagation across depth, surface normal and semantic segmentation. In: CVPR (2019)
61. Zheng, S., et al.: Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In: CVPR (2021)
62. Zhou, L., et al.: Pattern-structure diffusion for multi-task learning. In: CVPR (2020)
63. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable detr: deformable transformers for end-to-end object detection. arXiv:2010.04159 (2020)