Dynamic Feature Integration for Simultaneous Detection of Salient Object, Edge and Skeleton

Jiang-Jiang Liu, Qibin Hou, and Ming-Ming Cheng, *Senior Member, IEEE*

***Abstract*—Salient object segmentation, edge detection, and skeleton extraction are three contrasting low-level pixel-wise vision problems, where existing works mostly focused on designing tailored methods for each individual task. However, it is inconvenient and inefficient to store a pre-trained model for each task and perform multiple different tasks in sequence. There are methods that solve specific related tasks jointly but require datasets with different types of annotations supported at the same time. In this paper, we first show some similarities shared by these tasks and then demonstrate how they can be leveraged for developing a unified framework that can be trained end-to-end. In particular, we introduce a selective integration module that allows each task to dynamically choose features at different levels from the shared backbone based on its own characteristics. Furthermore, we design a task-adaptive attention module, aiming at intelligently allocating information for different tasks according to the image content priors. To evaluate the performance of our proposed network on these tasks, we conduct exhaustive experiments on multiple representative datasets. We will show that though these tasks are naturally quite different, our network can work well on all of them and even perform better than current single-purpose state-of-the-art methods. In addition, we also conduct adequate ablation analyses that provide a full understanding of the design principles of the proposed framework.**

***Index Terms*—Salient object segmentation, edge detection, skeleton extraction, joint learning**

I. Introduction

With the rapid popularization of mobile devices, more and more deep learning based computer vision applications have been ported from computer platforms to mobile platforms. Many low-level computer vision tasks, benefitting from their category-agnostic characters, act as fundamental components in mobile devices. For example, when using a smartphone to photograph, many supporting tasks are running in the background to assist users with better pictures and provide real-time effect previews. Single-camera smartphones usually apply the salient object segmentation task to simulate the bokeh effect that requires depth information [1], [2]. To help users taking pictures with more visual pleasing compositions, the edge detection task is adopted to obtain structure information [3], [4]. And the skeleton extraction task plays an important role in supporting taking photos by gesturing and instructing users with more interesting poses [5]. However, due to the limited storage and computing resources of mobile devices, it is inconvenient and inefficient to store the pre-trained models for every different applications and perform multiple different tasks sequentially.

J.J. Liu, and M.M. Cheng are with College of Computer Science, Nankai University. M.M. Cheng is the corresponding author (cmm@nankai.edu.cn).

Q. Hou is with National University of Singapore.

The source code of this paper is publicly available on our project page: http://mmcheng.net/dfi/.

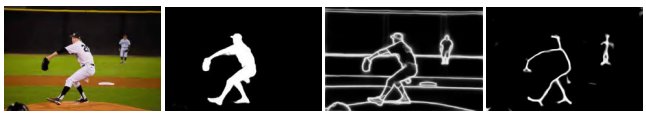


Image Saliency Edge Skeleton

Fig. 1. An example case where information may conflict when learning saliency, edge, and skeleton simultaneously. The man in the behind is not salient yet has skeletons, And for edge detection, it needs to detect all possible edge areas, whether being salient or belonging to skeletons. All the predictions are performed by our approach.

One feasible solution is to perform the aforementioned tasks within a single model but there exist two main challenges. One is how to learn different tasks simultaneously while the other is how to settle the divergence of feature domains and optimization targets of different tasks. Most previous work [6]–[9] solved the first challenge by observing the characteristics owning by different tasks and designing specialized network structures for each task manually. They assumed that all the tasks learned jointly are complementary and some tasks are auxiliary (*e.g.* utilizing extra edge information to help the salient object detection task with more accurate segmentations in edge areas). Usually the performances of the auxiliary tasks are sacrificed and ignored. But when facing the second challenge that the tasks being solved are contrasting, as demonstrated in Fig. 1, directly applying these methods often fails. As shown in the 3rd row of Tab. III, when trained jointly with the other two tasks, the performance of skeleton extraction is badly damaged.

The design criterion of previous work is usually task-oriented and specific, greatly restricting their applicability to other tasks [6]. From the standpoint of network architecture, in spite of three different tasks, all of them require multi-level features, though in varying degrees. Salient object segmentation requires the ability to extract homogeneous regions and hence relies more on high-level features [1]. Edge detection aims at detecting accurate boundaries and hence needs more low-level features to sharpen the coarse edge maps produced by deeper layers [10], [11]. Skeleton extraction [12], [13] prefers a proper combination of low-, mid- and high-level information to detect scale-variant (either thick or thin) skeletons. Thus, a natural question is whether it is possible to design an architecture that can coalesce these three contrasting low-level vision tasks into a unified but end-to-end trainable network with no loss on the performance of each task.

Taking into account the different characteristics of each task, we present a novel, unified framework to settle the above challenges. Specifically, our network comprises a shared backbone and three task branches of identical design, as shown in Fig. 2. To facilitate each task branch to select appropriate features at different levels of the backbone automatically, we

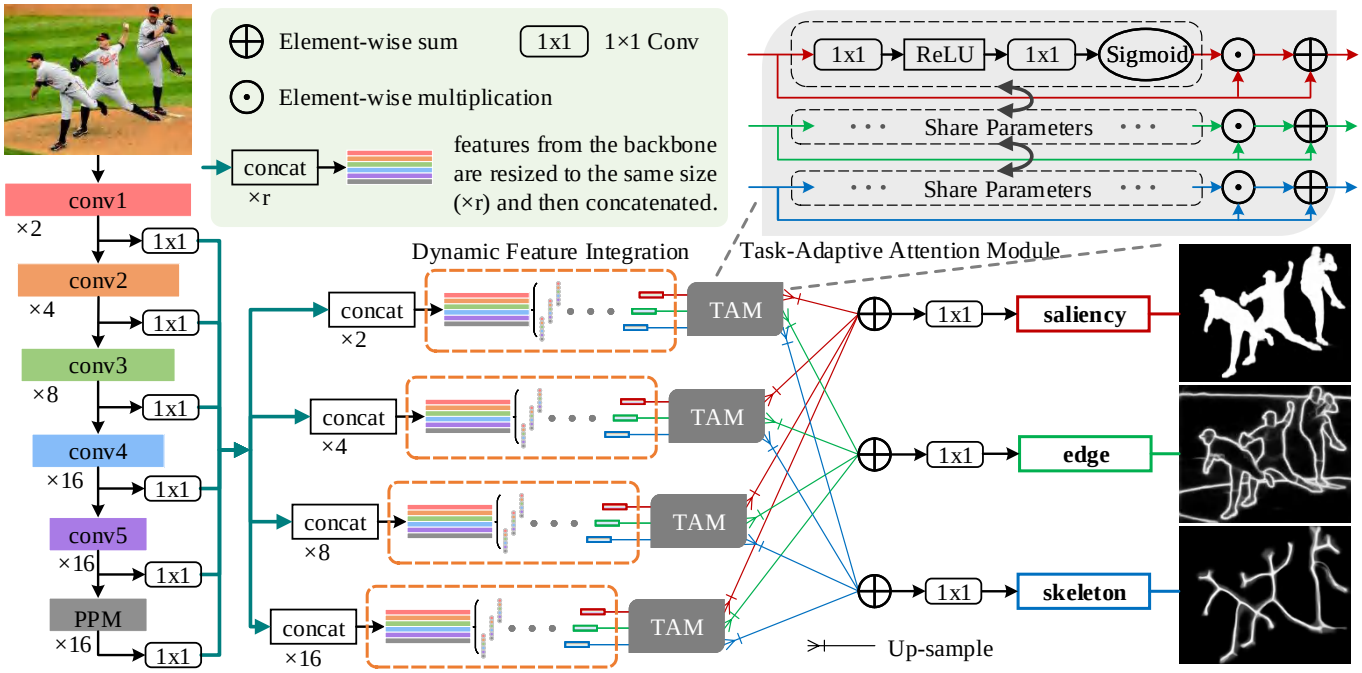


Fig. 2. The overall pipeline of our proposed approach (best viewed in color).

introduce a dynamic feature integration strategy that is able to choose favored features dynamically in an end-to-end learning manner. This dynamic strategy can largely ease the process of architecture building and promote the backbone to adjust its parameters for solving multiple problems adaptively. Then a task-adaptive attention module is adopted to enforce the interchange of information among different task branches in a separate-gather way. By coupling previously independent branches, we can avoid the network to optimize asymmetrically. Our approach is easy to follow and can be trained end-to-end on a single GPU. Without sacrifice on performance, it reaches a speed of 40 FPS performing the three tasks simultaneously when processing a 300×400 image.

To evaluate the performance of the proposed architecture, we compare it with the state-of-the-art methods of the three tasks. Experimental results show that our approach outperforms existing single-purpose methods on multiple widely used benchmarks. Specifically, for salient object segmentation, compared to previous state-of-the-art works, our method has a performance gain of 1.2% in terms of F-measure on average, over six popular datasets. For skeleton extraction, we also improve the state-of-the-art results by 1.9% in terms of F-measure on the SK-LARGE dataset [14]. Furthermore, to let readers better understand the proposed approach, we conduct extensive ablation experiments on different components of the proposed architecture.

To sum up, the contributions of this paper can be summarized as follows: (i) We design a dynamic feature integration strategy to explore the feature combinations automatically according to each input and task, and solve three contrasting tasks simultaneously in an end-to-end unified framework, running at 40 FPS. (ii) We compare our multi-task approach with the single-purpose state-of-the-arts of each task and obtain better performances.

II. Related Work

A. Relevant Binary Tasks

For salient object segmentation, traditional methods are mostly based on hand-crafted features [15]–[21]. With the popularity of CNNs, many methods [22]–[27] started to use CNNs to extract features. Some of them [28]–[33] incorporated the idea of iterative and recurrent learning to refine the predictions. There are also works solving the problem from the aspect of fusing richer features [34]–[43], introducing attention mechanism [44]–[46], using multiple stages to learn the prediction in a stage-wise manner [30], [47], [48], or adding more supervisions to get predictions with sharper edges [49]–[58]. For edge detection, early works [59]–[61] mostly relied on various gradient operators. Later works [62]–[64] further employed manually-designed features. Recently, CNN-based methods solved this problem by using fully-convolutional networks in a patch-wise [65]–[68] or pixel-wise prediction manner [69]–[73]. For skeleton extraction, earlier methods [74]–[76] mainly relied on gradient intensity maps of natural images to extract skeletons. Later, learning-based methods [77]–[80] viewed skeleton extraction as a per-pixel classification problem or a super-pixel clustering problem. Recent methods [5], [12], [13], [81] designed powerful network structures considering this problem hierarchically. Different from all the above approaches, our approach simultaneously solves the three tasks within a unified framework instead of learning each task with an individual network.

B. Multi-Task Learning

Multi-task learning (MTL) has a long history in the area of machine learning [82]–[85]. Recently, many CNN-based MTL methods had been proposed, most of which focused on the design of network architecture [6], [86]–[88], or loss functions to balance the importance of different tasks [89], [90], or both of them [91]. Different work also solved different task combinations, including: image classification in multiple domains

[92]; object recognition, localization, and detection [93]–[95]; pose estimation and action recognition [96]–[98]; semantic classes, surface normals, and depth prediction [86], [89], [91], [99]–[101]. However, the majority of these methods focused on specific related tasks requiring datasets with different types of annotations supported simultaneously. Different from the above methods, we aim to incorporate the idea of dynamic feature integration into architecture design. This allows our approach to learn multiple tasks together based on training data from multiple individual datasets. Moreover, unlike the previous methods [6], [7] which fix the strategies of how features integrate into network structures, our approach can adjust the network connections to select features dynamically to facilitate multi-task training.

C. Gating Mechanism

Gating mechanism was firstly introduced in the area of natural language and speech processing. Recent works adopt it into various computer vision tasks and prove its effectiveness. For the semantic segmentation task, Qi [102] proposed to use memory gates between layers to learn feature representations of customized scales for each individual pixels. Takikawa *et al.* [103] used the higher-level activations in the classical stream to gate the lower-level activations in the shape stream, which effectively removed the noise information. Ding *et al.* [104] proposed a scheme of gated sum to selectively aggregate multi-scale features for each spatial position. Cheng *et al.* [105] utilized RGB-D information and designed a gated fusion layer to combine the RGB and depth features. Zhu *et al.* [106] and Li *et al.* [107] solved the problem of object detection and used the gating techniques to select anchor feature. In the task of image classification, Chen *et al.* [108] proposed a gater network to select filters from the backbone network, while [109], [110] design a soft gating mechanism that allowed each neuron to adaptively adjust its receptive field size. Hua *et al.* [111] adopted the gating mechanism in network pruning by cutting off those less important channels. Different from the above methods, we utilize the gating mechanism to solve three contrasting tasks simultaneously. Also, different from selecting feature pixel-, channel-, or layer-wisely, we select feature from the backbone in a stage-wise way.

III. Method

In this section, instead of attempting to manually design an architecture that might work for all the three tasks, we propose to encourage the network to dynamically select features at different levels according to the favors of each task and the content of each input as described in Sec. I.

A. Overall Pipeline

We include three different tasks on multiple individual datasets (*i.e.* , DUTS [112] for saliency, BSDS 500 [64] and VOC Context [113] for edge, SK-LARGE [12] or SYM-PASCAL [13] for skeleton) within a unified network which can be trained end-to-end. All the datasets are directly used following the existing single-purpose methods proposed for each task with no extra processing.

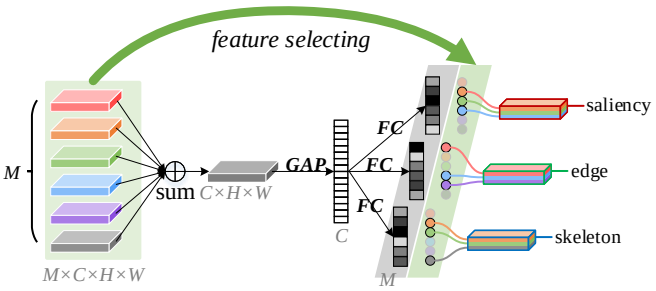


Fig. 3. Detailed illustration of DFIM. It takes the set of features extracted from the backbone as input, which is resized to the same size first. Then different stages of these features are dynamically selected for each task.

The overall pipeline of the proposed framework is illustrated in Fig. 2. We employ the ResNet-50 [114] network as the feature extractor. We take the feature maps outputted by `conv_1` as S_1, and the outputs by `conv2_3`, `conv3_4`, `conv4_6`, and `conv5_3` as S_2 to S_5, respectively. We set the dilation rates of the 3×3 convolutional layers in `conv5` to 2 as done in pixel-wise prediction tasks. Moreover, we add a pyramid pooling module (PPM) [115] on the top of ResNet-50 to capture more global information as done in [8], [48]. The output is denoted as S_6.

Rather than manually fixing the feature integration strategy in the network structure as done in most of the previous single-purpose methods, a serious of dynamic feature integration modules (DFIMs) of various output down-sampling rates (orange dashed rounded rectangles in Fig. 2) are arranged to integrate the features extracted from the backbone (*i.e.* $\{S_i\}$, where $1 \leq i \leq M$ and $M = 6$) dynamically and separately for the three tasks.

A task-adaptive attention module (TAM) is then followed after each DFIM to intelligently allocate information across tasks, preventing the network from tendentious optimization directions. Finally, the corresponding feature maps outputted by the TAMs for each task are up-sampled and summated and then followed by a 1×1 convolutional layer for final prediction, respectively.

B. Dynamic Feature Integration

It has been mentioned in many previous multi-task methods [6], [86], [89], [91] that the features required by different tasks vary greatly. And most of them require multiple kinds of annotations within a single dataset, which is difficult to obtain. Differently, we utilize training data of different tasks from multiple individual datasets, and it is more likely to meet circumstances where features required by different tasks conflict, as demonstrated in Fig. 1. To solve this problem, we propose DFIM, which adjusts the feature integration strategy dynamically according to each task and input during both training and testing. Compared to existing methods that integrate specific levels of features from the backbone based on manual observations of different tasks' characteristics, DFIM learns the feature integration strategy.

To be specific, we take the set of features $\{S_i\}$ extracted from the backbone as input for each DFIM. And the demanded output down-sampling rate $\times r$ of each DFIM is determined during the network definition period. As illustrated in Fig. 3,

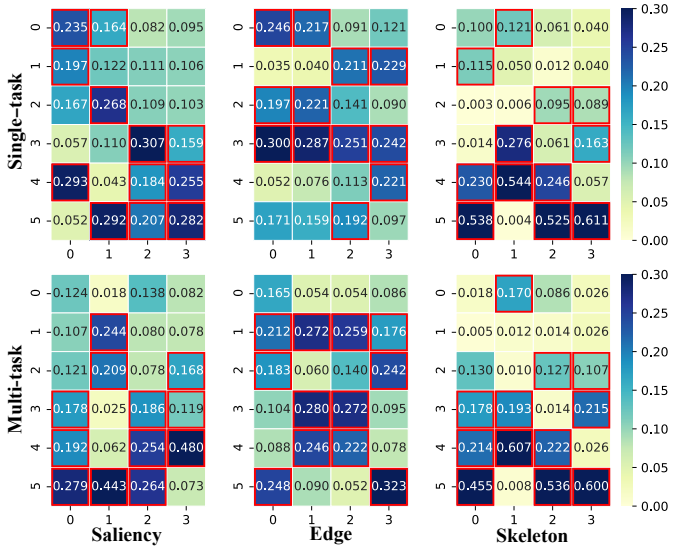


Fig. 4. The weights of each stage of features been selected by each DFIM. In each subplot, row indicates indexes of DFIMs, and column indicates stages of features. Only the top half stages (red rectangles) in each DFIM are kept.

we first transfer the features $\{S_i\}$ to have the same number of channels (*i.e.* , $C\times$) and down-sampling rate (*i.e.* , $\times r$), denoted as $\{S_i^r\}$, through a 1×1 convolutional layer and bilinear interpolation, respectively. To give DFIM a view of all the features to be selected, we summate $\{S_i^r\}$ and follow it with a global average pooling (GAP) layer to create a compact and global feature ($C\times$) as done in the SENet [116]. For each task $t \in \{saliency, edge, skeleton\}$, we use an independent fully connected (FC) layer to map the $C\times$ feature to $M\times$ channels, and then apply a softmax operator to transform the $M\times$ feature into the form of probability $\{p_i^{r,t}\}$ $(1 \le i \le M)$ that could be used as an indicator to select features. As not every stage of features from the backbone is always helpful, different from existing low-level vision methods [3], [5], [10] who keep dense connections of $\{S_i^r\}$, we only keep half the connections as $\{S_i^{r,t}\}$ $(1 \le i \le M)$:

$$S_i^{r,t} = \begin{cases} p_i^{r,t} * S_i^r, & \text{if } p_i^{r,t} \ge median(\{p_i^{r,t}\}) \\ 0, & \text{else,} \end{cases} \tag{1}$$

where $median(\cdot)$ means taking the median and $1 \le i \le M$. Thus the output of DFIM with down-sampling rate $\times r$ for task t can be obtained with

$$D^{r,t} = \sum_i S_i^{r,t}. \tag{2}$$

Detailed analysis of this design choice is described in Sec. V-B.

By arranging a series of DFIMs of various down-sampling rates, we can obtain the dynamically integrated feature maps $\{D^{r,t}\}(r \in \{2,4,8,16\}, t \in \{saliency, edge, skeleton\})$, as shown in Fig. 2. Since the feature integration strategies depend only on the input and task type, the network is able to learn integration strategies for each input and task within a broader and more flexible feature combination space in an end-to-end manner.

C. Task-Adaptive Attention

As we utilize training data from multiple individual datasets, the domain shifting [82], [117] problem can not be ignored. How to coalesce the information from diverse datasets effectively and efficiently is indispensable to the maintaining of the overall performance across all tasks. As illustrated in the first row (Single-task) of Fig. 4, the levels of features that different tasks favor vary greatly. If we use the task-specific feature maps $\{D^{r,t}\}(r \in \{2,4,8,16\})$ generated by DFIMs for the prediction of each task directly, the gradients of some task to the shared parts of the network may bias distinctly from the other tasks, hence deflecting the optimization direction to local minimums and causing underfitting.

To this end, we propose to let the network have the ability of intelligently allocating information for different tasks after the shared features from the backbone are dynamically integrated and tailored for each task. As shown in the top right corner of Fig. 2, the output feature maps $\{D^{r,t}\}(t \in \{saliency, edge, skeleton\})$ from the DFIM with a down-sampling rate of $\times r$ are further forwarded to a TAM.

In each TAM module, we first feed the input feature map $D^{r,t} \in \mathbb{R}^{C\times H\times W}$ into a 1×1 convolutional layer ($f_1^{1\times1}$) to reduce the aliasing effect of feature map addition after up-sampling (Eqn. (2)), which is followed by a ReLU activation function to introduce non-linearity. Then another 1×1 convolutional layer ($f_2^{1\times1}$) is adopted to map the cross-channel information. After that, we apply a sigmoid layer (σ) to calculate the spatial attention map $A^{r,t} \in \mathbb{R}^{C\times H\times W}$:

$$A^{r,t} = \sigma(f_2^{1\times1}(ReLU(f_1^{1\times1}(D^{r,t})))), \tag{3}$$

where the parameters in $f_1^{1\times1}$ and $f_2^{1\times1}$ are shared across tasks. With the input feature map and its attention map, the final output feature map of TAM can be obtained via:

$$T^{r,t} = D^{r,t} \odot (1 + A^{r,t}), \tag{4}$$

where $\odot$ denotes element-wise multiplication. $D^{r,t} \odot A^{r,t}$ roles as residuals to the input feature map.

We share the learnable parameters in TAM across tasks for the exchange of information. Compared to using the outputs for each task directly, the additional modeling of all tasks' relations gives DFIM the ability to adaptively adjust each task's influences on the shared backbone by considering the content of input and all task's characteristics simultaneously. TAM forces the interchange of information across tasks even after the feature maps are separated and tailored for each task. This is quite different from previous methods [6], [7], [91], which kept different tasks' branches independent of each other until the end.

To have a better perception, we visualize the intermediate feature maps around TAM in Fig. 5. As can be seen, in the 1st row, for salient object segmentation, before TAM (a,e), it is hard to distinguish the dog (child) from the background. The attention maps learned in TAM (b,f) erase the activation of background effectively. And after TAM (c,g), the dog (child) is highlighted clearly. In the 2nd row, for edge detection, the feature maps after TAM (c,g) have obvious thinner and sharper activations in the areas where edges may locate compared to the thick and blur activations before TAM (a,e). A similar phenomenon can also be observed for the skeleton extraction task. As shown in the last row, the skeletons of the dog (child) become stronger and clearer after TAM. All the aforementioned discussions verify the significant effect of TAM on better allocating the information for different tasks.

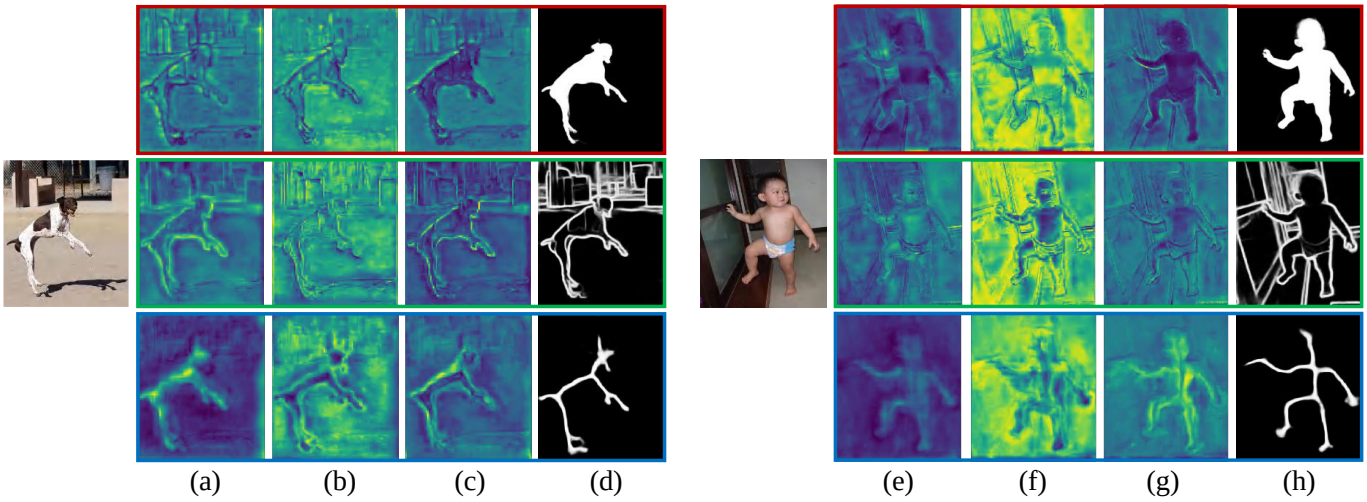


Fig. 5. Visualizing feature maps around TAM. (a,e) Before TAM; (b,f) Attention maps in TAM; (c,g) After TAM; (d,h) Predictions. As can be seen, TAM can tailor the feature maps for each task adaptively and accordingly. From top to bottom: more distinct saliency parts, sharper edges, and stronger skeletons.

TABLE I
THE DATASETS WE USE FOR TRAINING AND TESTING.

Task	Training	#Images	Testing	#Images
Saliency	DUTS-TR [112]	10553	ECSSD [121], PASCAL-S [122], DUT-OMRON [123], SOD [124], HKU-IS [22], DUTS-TE [112]	1000, 850, 5166, 300, 1447, 5019
Edge	BSDS500 [64] & VOC Context [113]	300 + 10103	BSDS500 [64]	200
Skeleton	SK-LARGE [14]	746	SK-LARGE [14]	745
	SYM-PASCAL [13]	648	SYM-PASCAL [13]	788

IV. EXPERIMENT SETUP

In this section, we describe the experiment setups, including the implementation details of the proposed network, the used datasets, the training procedure, and the evaluation metrics for the three tasks.

Implementation Details. We implement the proposed method based on PyTorch[1]. All experiments are carried out on a workstation with an Intel Xeon 12-core CPU (3.6GHz), 64GB RAM, and a single NVIDIA RTX-2080Ti GPU. We use the Adam [118] optimizer with an initial learning rate of 5e-5 and a weight decay of 5e-4. Our network is trained for 12 epochs in total, and the leaning rate is divided by 10 after 9 epochs. The parameters of the backbone (*i.e.* , ResNet-50 [114]) of our network are initialized with ImageNet [119] pre-trained model, while all other parameters are randomly initialized. Group normalization [120] is applied after each convolutional layer except for the backbone. The optimization configurations for all parameters in our network are identical, except for the parameters of the batch normalization layers of the backbone are frozen during both training and testing.

Datasets. We use individual datasets for different tasks, and each dataset only has one kind of annotation. The detailed configurations are listed in Tab. I. And all the datasets are directly used following the existing single-purpose methods proposed for each task [3], [5], [37] with no extra pre-processing.

[1]https://pytorch.org

Training Procedure. To jointly solve three different tasks from three individual datasets in an end-to-end way, for each iteration, we randomly sample an image-groundtruth pair for each of the three tasks, respectively. Then sequentially, each of the three image-groundtruth pairs is forwarded to the network, and the corresponding loss is calculated. At last, we simply summate the three calculated losses, backward through the network, and then take an optimization step. All the other training procedures are identical to typical single-purpose methods.

Loss Functions. We define the loss functions of the three tasks the same as most of the previous single-purpose methods. We use standard binary cross-entropy loss for salient object segmentation [1], [37] and balanced binary cross-entropy loss [3], [5], [10] for edge detection and skeleton extraction. The detailed formulas of the loss functions we used are as follows. Given an image's prediction map $\hat{Y}$ and its corresponding groundtruth map Y, for all pixels (i, j), we compute the standard binary cross-entropy loss as:

$$\begin{aligned}\mathcal{L}_s(\hat{Y}, Y) = -\sum_{i,j}[Y(i,j)\cdot\log\hat{Y}(i,j)+ \\ (1-Y(i,j))\cdot\log(1-\hat{Y}(i,j))],\end{aligned} \tag{5}$$

and the balanced binary cross-entropy loss as:

$$\begin{aligned}\mathcal{L}_b(\hat{Y}, Y) = -\sum_{i,j}[\beta\cdot Y(i,j)\cdot\log\hat{Y}(i,j)+ \\ (1-\beta)\cdot(1-Y(i,j))\cdot\log(1-\hat{Y}(i,j))],\end{aligned} \tag{6}$$

where $\beta = |Y^-|/|Y^+ + Y^-|$ while Y^+ and Y^- refer to the foreground and background pixels, respectively.

- Salient Object Segmentation:

$$\mathcal{L}_{sal}(\hat{Y}_{sal}, Y_{sal}) = \mathcal{L}_s(\hat{Y}_{sal}, Y_{sal}). \tag{7}$$

- Edge Detection:

$$\mathcal{L}_{edg}(\hat{Y}_{edg}, Y_{edg}) = \mathcal{L}_b(\hat{Y}_{edg}, Y_{edg}), \tag{8}$$

in which the Y^+ in the β of $\mathcal{L}_b$ refers to the edge pixels and Y^- refers to the non-edge pixels.

TABLE II
THE COMPOSITION OF THE PROPOSED NETWORK'S PARAMETERS. AS CAN BE SEEN, THE FEATURE EXTRACTOR (RESNET-50 & PPM) AND THE SHARED PARTS TAKE UP THE MAJORITY.

Total: 29.57M						
Shared: 27.01M (91.34%)				Specific: 2.56M (8.66%)		
ResNet-50	PPM	DFIMs	TAMs	Saliency	Edge	Skeleton
23.46M	1.31M	1.42M	0.83M	0.85M	0.85M	0.85M
79.34%	4.43%	4.80%	2.81%	2.87%	2.87%	2.87%

- Skeleton Extraction:

$$\mathcal{L}_{sk}(\hat{Y}_{sk}, Y_{sk}) = \mathcal{L}_b(\hat{Y}_{sk}, Y_{sk}), \tag{9}$$

  in which the Y^+ in the β of $\mathcal{L}_b$ refers to the skeleton pixels and Y^- refers to the non-skeleton pixels.

The overall loss is calculated as a simple summation of the losses of the three tasks, which weigh equally:

$$\mathcal{L} = \mathcal{L}_{sal}(\hat{Y}_{sal}, Y_{sal}) + \mathcal{L}_{edg}(\hat{Y}_{edg}, Y_{edg}) + \mathcal{L}_{sk}(\hat{Y}_{sk}, Y_{sk}). \tag{10}$$

Evaluation Criteria. For *salient object segmentation*, we use precision-recall (PR) curves, F-measure score (F_β), mean absolute error (MAE), and S-measure (S_m) [125] for evaluation. The F-measure score is computed by the weighted harmonic mean of the precision and recall and is an overall measurement of performance:

$$F_\beta = \frac{(1+\beta^2) \times Precision \times Recall}{\beta^2 \times Precision + Recall}, \tag{11}$$

where β^2 is set to 0.3 as done in previous work to weight precision more than recall. The MAE score measures the average pixel-wise absolute difference between the binary ground truth G and the predicted saliency map P.

$$MAE = \frac{1}{W \times H} \sum_{x=1}^{W} \sum_{y=1}^{H} |P(x,y) - G(x,y)|, \tag{12}$$

where W and H denote the width and height of P, respectively. The S-measure score evaluates the structural similarity by considering the object-aware (S_o) and region-aware (S_r) structure similarities simultaneously:

$$S_m = \alpha \times S_o + (1-\alpha) \times S_r, \tag{13}$$

where α is empirically set to 0.5.

For *edge detection*, before evaluation, we apply the standard non-maximal suppression (NMS) algorithm [126] to get thinned edges. To produce the binary edge map, there are two choices to set the threshold. One is to use a fixed threshold for all images in the dataset, providing optimal overall performance over the set. We refer to this as optimal dataset scale (ODS). The other is to select an optimal threshold for each image, which is called optimal image scale (OIS). For both ODS and OIS, we report the F-measure score:

$$F_m = \frac{2 \times Precision \times Recall}{Precision + Recall}. \tag{14}$$

For *skeleton extraction*, we follow the evaluation protocol in [77]. We report the precision-recall (PR) curves and maximum F-measure score (F_m). Before evaluation, the predicted skeleton maps are NMS-thinned as commonly done. To obtain the PR curves, given a NMS-thinned skeleton map, we firstly threshold it into a binary map and then match it with the corresponding groundtruth map. During the matching, minor localization errors are allowed between the predicted positive pixels and the groundtruth skeleton pixels. By applying different thresholds to the predicted skeleton map, a series of precision and recall pairs are obtained to draw the PR curve. The maximum F-measure is obtained under the optimal threshold over the whole dataset using Eqn. (14).

V. ABLATION STUDIES

In this subsection, we first analyse the composition of parameters of the proposed model. Then we investigate the effectiveness of the proposed DFIM by conducting experiments on both single- and multi-task settings. Finally we show the effect of TAM with a better overall convergence and performance.

A. Composition of Parameters

We list the composition of the parameters of our network in Tab. II. As can be seen, 91.34% of the parameters are shared across tasks where the feature extractor (ResNet-50 & PPM) takes up 91.71%. And the shared parts of DFIMs and TAMs only bring in 2.25M (8.33%) parameters. Each task owns 0.85M (2.87%) task-specific parameters, respectively. The polarized composition of parameters proves the efficiency and effectiveness of the proposed approach. By taking advantage of the shared features extracted from the backbone and adaptively recombining them, more parameters and space can be saved. At the meanwhile, by handing the design of feature integration strategies to the network itself, less human interaction is required.

B. Dynamic Feature Integration

Effectiveness of Dynamic Feature Integration. As shown in the 1st row of Tab. III, directly applying our method whilst only performing a single task can obtain comparable results with the state-of-the-art methods on the salient object detection and edge detection tasks. And a bigger promotion can be observed on the skeleton extraction task (1.7%). This indicates that the proposed DFIM is able to adjust it's feature selecting strategies according to the characteristics of the target task being solved. Rather than engineering specific network structure for different tasks manually as usually done in the previous methods, DFIM requires less human interactions.

When the three tasks are learned jointly (the 5th row of Tab. III), the performance of salient object segmentation task is promoted by a clear margin on three datasets in nearly all terms. This is consistent with previous researches that edge information can help the salient object segmentation task with more accurate segmentation in edge areas. The performance of edge detection task also increases, indicating that the edges of salient objects may provide useful guidance signals as well. The skeleton extraction task only drops slightly.

To have a numerical estimation of the difficulty in jointly training the three tasks, we set up a baseline by removing the dynamic feature selecting process in DFIM (the 3rd row in Tab. III, marked as 'identity'). This means that all the

TABLE III
QUANTITATIVE SALIENT OBJECT SEGMENTATION, EDGE DETECTION AND SKELETON EXTRACTION RESULTS FIVE WIDELY USED DATASETS. 'SINGLE-TASK' MEANS DIRECTLY APPLY OUR METHOD WHILST ONLY PERFORMING A SINGLE TASK. THE BEST RESULTS IN EACH COLUMN ARE HIGHLIGHTED IN **BOLD**.

No.	DFIM	TAM	Saliency									Edge		Skeleton
			PASCAL-S [122]			DUT-OMRON [123]			DUTS-TE [112]			BSDS 500 [64]		SK-LARGE [14]
			$F_\beta \uparrow$	MAE $\downarrow$	$S_m \uparrow$	$F_\beta \uparrow$	MAE $\downarrow$	$S_m \uparrow$	$F_\beta \uparrow$	MAE $\downarrow$	$S_m \uparrow$	ODS $\uparrow$	OIS $\uparrow$	$F_m \uparrow$
Our Method (Single-Task)														
1	sparse	w/o	0.860	0.075	0.849	0.811	0.059	0.835	0.875	0.042	0.878	0.815	0.831	0.749
2	sparse	independent	0.859	0.081	0.849	0.817	0.060	0.835	0.880	0.045	0.878	0.812	0.826	0.746
Our Method (Multi-Task)														
3	identity	w/o	0.877	**0.062**	**0.865**	0.818	0.056	0.836	0.885	**0.038**	0.886	0.811	0.828	0.708
4	dense	w/o	0.872	0.064	0.859	0.813	0.056	0.833	0.877	0.039	0.881	0.810	0.825	0.740
5	sparse	w/o	0.874	0.064	0.862	0.817	0.056	**0.842**	0.884	**0.038**	**0.887**	0.818	0.834	0.744
6	sparse	independent	0.873	0.065	0.861	0.815	0.057	0.836	0.879	0.039	0.883	0.815	0.832	**0.753**
7	sparse	share	**0.880**	0.065	**0.865**	**0.829**	**0.055**	0.839	**0.888**	**0.038**	**0.887**	**0.819**	**0.836**	0.751
Other Methods (Multi-Task)														
8	UberNet$_{17}$ [6]		0.823	-	-	-	-	-	-	-	-	0.785	0.805	-
9	MLMS$_{19}$ [7]		0.853	0.074	0.844	0.793	0.063	0.809	0.854	0.048	0.862	0.769	0.780	-

operations after the features extracted from the backbone being summated are removed, as shown in Fig. 3. This also equals to replacing Eqn. (2) with

$$D^{r,t} = \sum_i S_i^r. \tag{15}$$

By comparing the 'identity' version with the proposed 'sparse' feature selecting version (the 5th row), we can observe clear drops on the tasks of edge detection and skeleton extraction, 0.7% and 3.6%, respectively. These phenomenons demonstrate that simply fusing all levels of features damages the detection of edge and skeleton. It's difficult to design network structures manually when the involved tasks have distinct optimization targets and take training samples from different datasets. Similar circumstance occurs in previous works [6], [7], where the performance of partial tasks decrease dramatically when solving different tasks jointly, as shown in the last two rows of Tab. III, But with DFIM, by letting the network itself to integrate features dynamically and accordingly, all of the three tasks perform comparably to training each task separately.

Dynamically Learned Integration Strategies. To have a better understanding of what feature integration strategies have been learned by our proposed method, we randomly select 100 images from each of the testing set of DUTS (saliency), BSDS500 (edge) and SK-LARGE (skeleton) to form up a 300-image set. By forwarding these images, we average the $\{p_i^{r,t}\}$ values of all images, which are the indicators for features selecting. We plot the probabilities of each stage of features from the backbone been selected by each DFIM for different tasks in Fig. 4. If we compare the subplots column-wise, the stages of features preferred by different tasks vary greatly. This may explain why a good performing architecture for one task does not work on the other tasks [1], [3], [81]. If we compare the subplots row-wise, the stages of features been selected when each of the three tasks is separately trained in a single-task manner also differ greatly from those when they are jointly trained in a multi-task manner. This may be the reason why each of the three tasks has been well investigated, but little literature has tried to solve them jointly. It is hard to manually design architecture as the features from the shared backbone now will be simultaneously affected by all the tasks.

TABLE IV
ABLATION ANALYSIS OF OUR APPROACH ON DIFFERENT COMBINATIONS OF DOWN-SAMPLING RATES OF DFIMS. THE BEST RESULTS IN EACH COLUMN ARE HIGHLIGHTED IN **BOLD**.

Down-sampling Rates	Saliency						Edge		Skeleton
	DUT-OMRON			DUTS-TE			BSDS 500		SK-LAR
	$F_\beta \uparrow$	MAE $\downarrow$	$S_m \uparrow$	$F_\beta \uparrow$	MAE $\downarrow$	$S_m \uparrow$	ODS $\uparrow$	OIS $\uparrow$	$F_m \uparrow$
2,4,8	0.814	0.057	0.839	0.879	**0.038**	0.885	0.815	0.829	0.742
4,8,16	0.809	0.057	0.837	0.880	**0.038**	0.884	0.814	0.829	**0.745**
2,4,8,16	**0.817**	**0.056**	**0.842**	**0.884**	0.038	**0.887**	**0.818**	**0.834**	0.744

Sparse or Dense Connections. In Tab. III, we compare our sparse-connected network with a dense-connected version, which means all the feature maps in $\{S_i\}$ $(1 \leq i \leq M, M = 6)$ are kept rather than only half as formulated in Eqn. (1). As shown in the 4th and 5th rows, the dense version has worse performances on nearly all three tasks. This may indicate that not every stage of features from the backbone is always helpful [127]. For example, for edge detection, more lower-level feature maps are necessary for precise localization of edge pixels [3], [10], while for skeleton extraction, more higher-level information is essential to determine whether a pixel being skeleton or not [5], [81].

Down-Sampling Rates of DFIMs. As listed in Tab. IV, we conduct ablation experiments on the combinations of down-sampling rates of DFIMs. A wider range of down-sampling rates shows a better equilibrium of overall performances, especially on salient object segmentation and edge detection, which agrees with common sense that richer multi-scale information usually helps.

C. Task-Adaptive Attention

Effectiveness of TAM. With DFIM, we can jointly train the three tasks under a unified architecture. However, as shown in the 5th row of Tab. III, compared to separately training (the 1st row), the performance of skeleton extraction decreases. As the annotations of salient object segmentation and edge

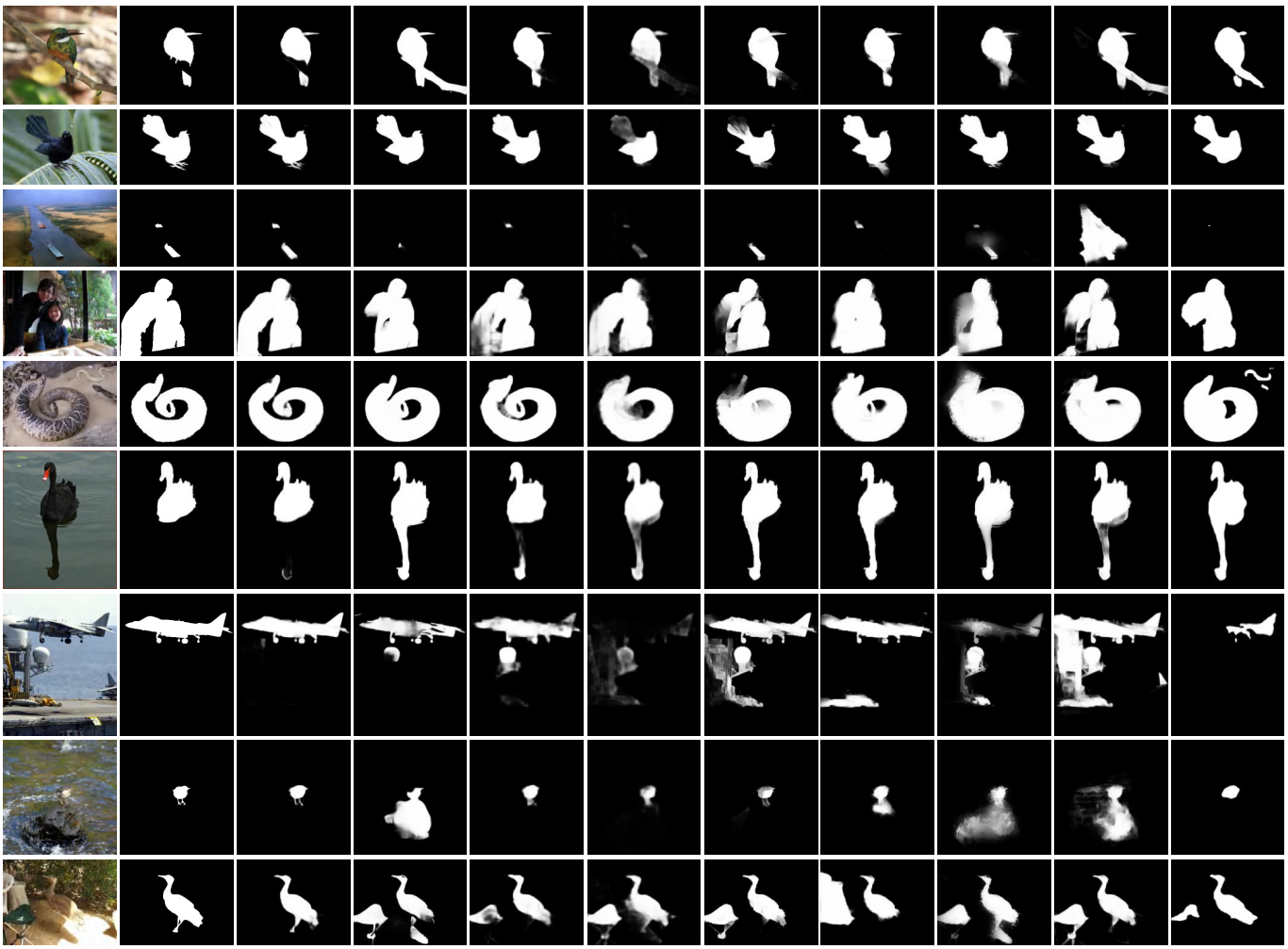


Fig. 6. Visual comparisons of different salient object segmentation approaches.

detection tasks pay more attention to pixels where edges exist, which disagree with the goal of skeleton extraction task, the optimization of skeleton extraction task could be influenced and misguided towards adverse directions. With TAM, the network is able to allocate the information of all tasks from a global view by adjusting the gradients of each task towards the shared backbone adaptively. As can be seen from the 7th row compared to the 5th row in Tab. III, better overall performances are reached. The performances of salient object segmentation and edge detection are slightly better while the skeleton extraction task outperforms with 0.7%.

Necessity of Information Exchange. To investigate whether the promotion brought by TAM is due to the introduction of additional parameters, we also conduct experiment leaving the parameters of different branches in TAM unshared (the 6th row) so that different task branches are independent of each other after selecting features from the shared backbone. By not sharing the parameters in TAM, extra 1.66M parameters are further lead in. However, as can be seen from the 6th row in Tab. III, even with more parameters introduced, the overall performance of the unshared version is obviously worse than the shared version of TAM (the 7th row). Though the skeleton task performs slightly better, the other two tasks decline greatly. These phenomenons show that enforcing the interchange of information across tasks after the separation of branches of each task is helpful to the overall convergence of all tasks, while simple attention mechanism works not well. This can also be observed from the first two rows of Tab. III, that appending TAM when each task is trained separately brings no help even downgrade to most of the three tasks.

VI. Comparisons to the State-of-the-Arts

In this section, we compare the proposed method (denoted as DFI for convenience) with state-of-the-art methods on salient object segmentation, edge detection, and skeleton extraction. As very little literature has solved the three tasks jointly before, *e.g.* UberNet [6] (CVPR'17) and MLMS [7] (CVPR'19), which solved salient object segmentation jointly with edge detection, we mainly compare with the state-of-the-art single-purpose methods of the three tasks for better illustration. For fair comparisons, for each task, the predicted maps (*e.g.* saliency maps, edge maps, skeleton maps) of other methods are generated by the original code released by the authors or directly provided by them. All the results are obtained directly from single-model test without relying on any other pre- or post-processing tools except for the NMS process before the evaluation of edge and skeleton maps [3], [5], [10]. And for each task, all the predicted maps are evaluated with the same evaluation code.

TABLE V
QUANTITATIVE SALIENT OBJECT SEGMENTATION RESULTS SIX WIDELY USED DATASETS. THE BEST RESULT IN EACH COLUMN ARE HIGHLIGHTED IN **BOLD**. AS CAN BE SEEN, OUR APPROACH ACHIEVES THE BEST RESULTS ON NEARLY ALL DATASETS IN TERMS OF F-MEASURE, MAE AND S-MEASURE.

	ECSSD [121]			PASCAL-S [122]			DUT-OMRON [123]			HKU-IS [22]			SOD [124]			DUTS-TE [112]		
Model	$F_\beta \uparrow$	MAE $\downarrow$	$S_m \uparrow$	$F_\beta \uparrow$	MAE $\downarrow$	$S_m \uparrow$	$F_\beta \uparrow$	MAE $\downarrow$	$S_m \uparrow$	$F_\beta \uparrow$	MAE $\downarrow$	$S_m \uparrow$	$F_\beta \uparrow$	MAE $\downarrow$	$S_m \uparrow$	$F_\beta \uparrow$	MAE $\downarrow$	$S_m \uparrow$
DCL_{16} [34]	0.896	0.080	0.869	0.805	0.115	0.800	0.733	0.094	0.762	0.893	0.063	0.871	0.831	0.131	0.763	0.786	0.081	0.803
$RFCN_{16}$ [28]	0.898	0.097	0.856	0.827	0.118	0.808	0.747	0.094	0.774	0.895	0.079	0.860	0.805	0.161	0.722	0.786	0.090	0.793
MSR_{17} [35]	0.903	0.059	0.887	0.839	0.083	0.835	0.790	0.073	0.805	0.907	0.043	0.896	0.841	0.111	0.782	0.824	0.062	0.834
DSS_{17} [1]	0.906	0.064	0.880	0.821	0.101	0.804	0.760	0.074	0.789	0.900	0.050	0.881	0.834	0.125	0.764	0.813	0.065	0.826
$NLDF_{17}$ [36]	0.903	0.065	0.870	0.822	0.098	0.805	0.753	0.079	0.770	0.902	0.048	0.878	0.837	0.123	0.759	0.816	0.065	0.816
$Amulet_{17}$ [37]	0.911	0.062	0.876	0.826	0.092	0.816	0.737	0.083	0.784	0.889	0.052	0.866	0.799	0.146	0.729	0.773	0.075	0.800
$PAGR_{18}$ [31]	0.924	0.064	0.883	0.847	0.089	0.822	0.771	0.071	0.775	0.919	0.047	0.889	-	-	-	0.854	0.055	0.839
$DGRL_{18}$ [30]	0.921	0.043	0.899	0.844	0.072	0.836	0.774	0.062	0.806	0.910	0.036	0.895	0.843	0.103	0.774	0.828	0.049	0.842
$MLMS_{19}$ [7]	0.924	0.048	0.905	0.853	0.074	0.844	0.793	0.063	0.809	0.922	0.039	0.907	0.857	0.106	0.790	0.854	0.048	0.862
$JDFPR_{19}$ [33]	0.925	0.052	0.902	0.854	0.082	0.841	0.802	0.057	0.821	-	-	-	0.836	0.121	0.767	0.833	0.058	0.836
$PAGE_{19}$ [50]	0.928	0.046	0.906	0.848	0.076	0.842	0.791	0.062	0.825	0.920	0.036	0.904	0.837	0.110	0.775	0.838	0.051	0.855
$CapSal_{19}$ [53]	-	-	-	0.862	0.073	0.837	-	-	-	0.889	0.058	0.851	-	-	-	0.844	0.060	0.818
CPD_{19} [40]	0.936	0.040	0.913	0.859	0.071	0.848	0.796	0.056	0.825	0.925	0.034	0.907	0.857	0.110	0.771	0.865	0.043	0.869
$PiCANet_{18}$ [45]	0.932	0.048	0.912	0.864	0.075	0.854	0.820	0.064	0.830	0.920	0.044	0.904	0.861	0.103	0.792	0.863	0.050	0.868
$AFNet_{19}$ [51]	0.932	0.045	0.907	0.861	0.070	0.849	0.820	0.057	0.825	0.926	0.036	0.906	-	-	-	0.867	0.045	0.867
$BASNet_{19}$ [54]	0.939	0.040	0.911	0.857	0.076	0.838	0.811	0.057	0.836	0.930	0.033	0.908	0.849	0.112	0.772	0.860	0.047	0.866
DFI (Ours)	**0.945**	**0.038**	**0.921**	**0.880**	**0.065**	**0.865**	**0.829**	**0.055**	**0.839**	**0.934**	**0.031**	**0.919**	**0.878**	**0.100**	**0.802**	**0.888**	**0.038**	**0.887**

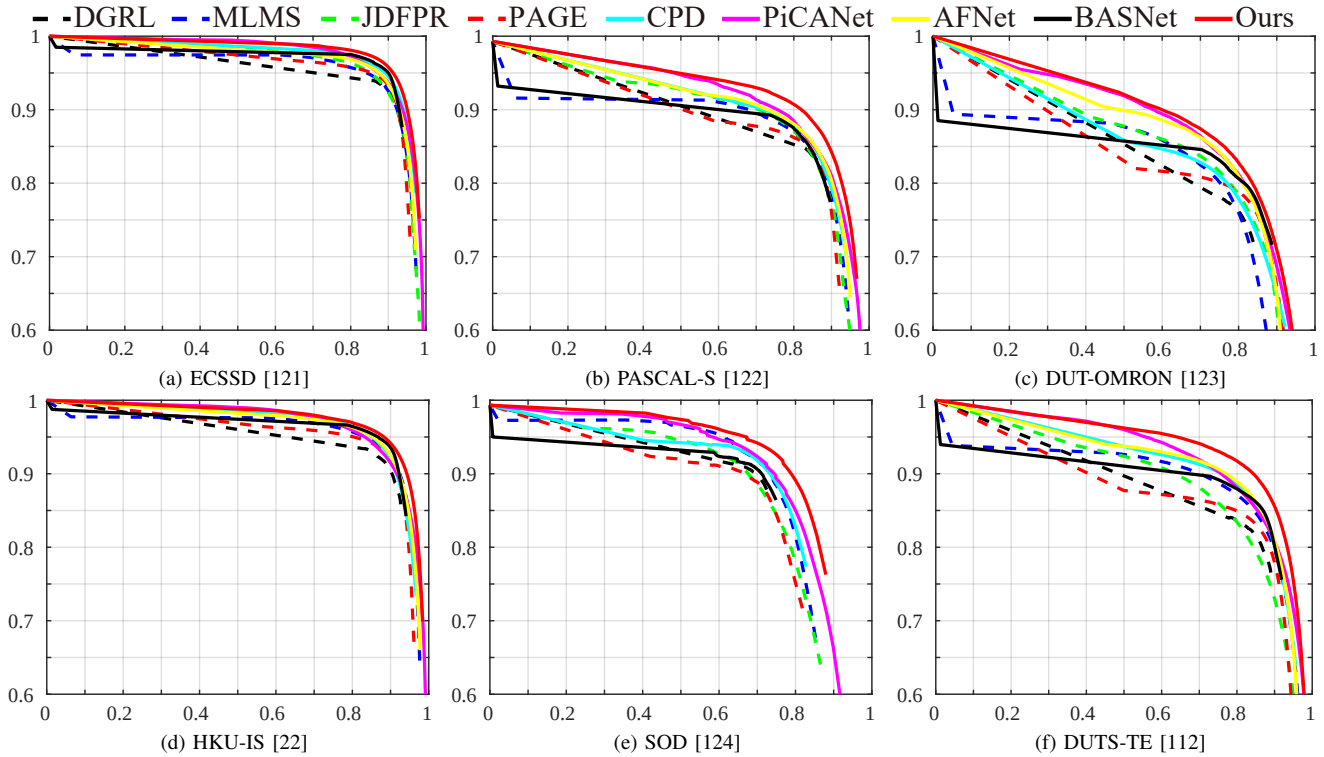


Fig. 7. Precision (vertical axis) recall (horizontal axis) curves on six popular salient object segmentation datasets.

A. Salient Object Segmentation

We exhaustively compare DFI with 16 existing state-of-the-art salient object segmentation methods including DCL [34], RFCN [28], MSR [35], DSS [1], NLDF [36], Amulet [37], PAGR [31], DGRL [30], MLMS [7], JDFPR [33], PAGE [50], CapSal [53], CPD [40], PiCANet [45], AFNet [51], and BASNet [54].

F-measure, MAE and S-measure Scores. Here, we compare DFI with the aforementioned approaches in terms of F-measure, MAE, and S-measure (See Tab. V). As can be seen, compared to the second-best methods on each dataset, DFI outperforms all of them over six datasets with average promotions of 1.2% and 1.0% in terms of F-measure and S-measure, respectively. Especially on the challenging DUTS-TE dataset, promotions of 2.1% and 1.8% in terms of F-measure and S-measure can be observed. Similar patterns can also be observed using the MAE score. Also, when compared to MLMS [7], which learns salient object segmentation and edge detection jointly, DFI has even larger improvements on both tasks, as shown in the 7th and 9th rows of Tab. III. Without TAM, DFI still outperforms MLMS [7] by a large margin (the 3rd and 9th rows). This phenomenon demonstrated the effectiveness of the proposed DFIM and TAM,

PR Curves. Other than numerical results, we also show the

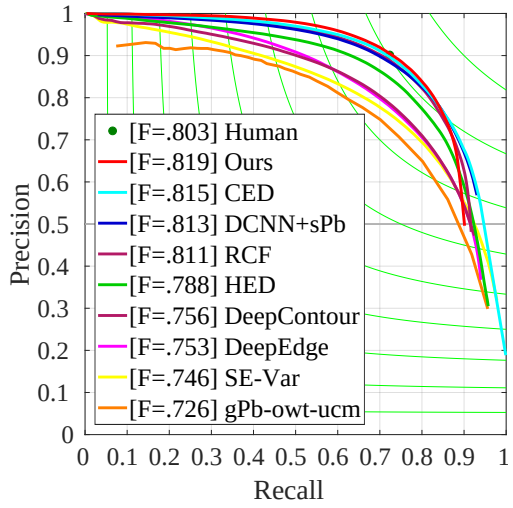


Fig. 8. The precision-recall curves on BSDS 500 dataset [64].

TABLE VI
QUANTITATIVE COMPARISONS OF DFI WITH EXISTING EDGE DETECTION METHODS. THE BEST RESULTS IN EACH COLUMN ARE HIGHLIGHTED IN **BOLD**.

Method	BSDS 500 [64]	
	ODS ↑	OIS ↑
gPb-owt-ucm$_{11}$ [64]	0.726	0.757
SE-Var$_{15}$ [126]	0.746	0.767
MCG$_{17}$ [128]	0.747	0.779
DeepEdge$_{15}$ [67]	0.753	0.772
DeepContour$_{15}$ [66]	0.756	0.773
HED$_{15}$ [10]	0.788	0.808
CEDN$_{16}$ [70]	0.788	0.804
RDS$_{16}$ [71]	0.792	0.810
COB$_{17}$ [11]	0.793	0.820
RCF$_{17}$ [3]	0.811	0.830
DCNN+sPb$_{15}$ [69]	0.813	0.831
CED$_{17}$ [72]	0.815	0.833
LPCB$_{18}$ [129]	0.815	0.834
DFI (Ours)	**0.819**	**0.836**

PR curves on the six datasets as shown in Fig. 7. As can be seen, the PR curves of DFI (red solid ones) are comparable to other previous approaches and even better on some datasets. Especially on the PASCAL-S and DUTS-TE datasets, DFI outstands compared to all other previous approaches. As the recall score approaches 1, our precision score is much higher than other methods, which reveals that the false positives in our saliency map are low.

Visual Comparisons. In Fig. 6, we show the visual comparisons with several previous state-of-the-art approaches. In the top row, the salient object is partially occluded. And DFI is able to segment the entire object without mixing in the unrelated areas. As shown in the 2nd row, DFI is also able to segment out the salient object with more precise boundaries and details. A similar phenomenon happens when processing images where salient objects are tiny and irregular or the contrast between foreground and background is low. For example, the bottom two rows of Fig. 6. These results demonstrate that DFI benefits from better distinguishing the edge pixels and segmenting out the whole objects, which might be the advantage of joint training with the edge detection and skeleton extraction tasks.

TABLE VII
QUANTITATIVE COMPARISONS OF DFI WITH EXISTING SKELETON EXTRACTION METHODS. THE BEST RESULTS IN EACH COLUMN ARE HIGHLIGHTED IN **BOLD**.

Method	SK-LARGE [14]	SYM-PASCAL [13]
	F_m ↑	F_m ↑
MIL$_{12}$ [77]	0.353	0.174
HED$_{15}$ [10]	0.497	0.369
RCF$_{17}$ [3]	0.626	0.392
FSDS$_{16}$ [12]	0.633	0.418
LMSDS$_{17}$ [14]	0.649	-
SRN$_{17}$ [13]	0.678	0.443
LSN$_{18}$ [130]	0.668	0.425
Hi-Fi$_{18}$ [5]	0.724	0.454
DeepFlux$_{19}$ [81]	0.732	0.502
DFI (Ours)	**0.751**	**0.511**

B. Edge Detection

We compare DFI with results from 13 existing state-of-the-art edge detection methods, including gPb-owt-ucm [64], SE-Var [126], MCG [128], DeepEdge [67], DeepContour [66], HED [10], CEDN [70], RDS [71], COB [11], RCF [3], DCNN+sPb [69], CED [72] and LPCB [129], most of which are CNN-based methods.

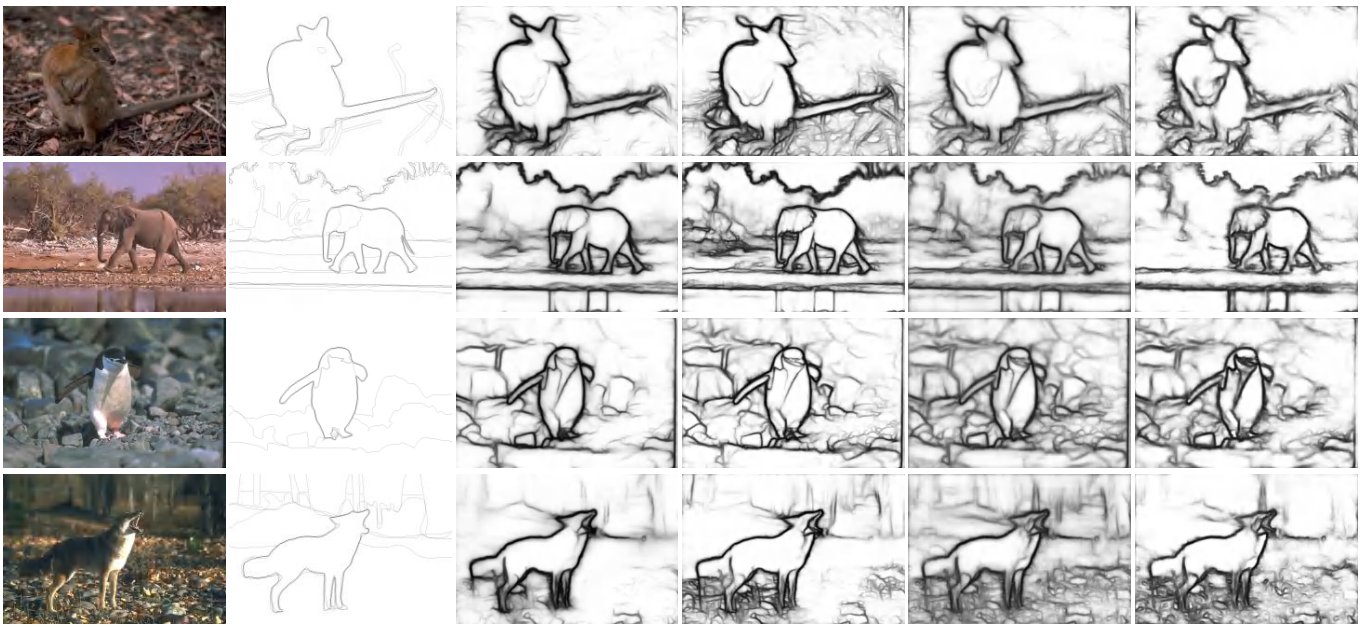
Quantitative Analysis. In Tab. VI, we show the quantitative results. DFI achieves ODS of 0.819 and OIS of 0.836, which are even better than the previous works that are well-designed for edge detection. Thanks to DFIM and TAM, the information from the other tasks not only does not influence but helps the performance of edge detection, as shown in the 1st and 7th rows of the 'Edge' column of Tab. III.

PR Curves. The precision-recall curves of our method and some selected methods on the BSDS 500 dataset [64] can be found in Fig. 8. One can observe that the PR curve produced by our approach is already better than human in some certain cases and is comparable to previous methods especially in precision.

Visual Analysis. In Fig. 9, we show some visual comparisons between DFI and some leading representative methods [10], [71], [72]. As can be observed, DFI performs better in detecting the boundaries compared to the others. In the last row of Fig. 9, it is apparent that the real boundaries of the wolf are well highlighted. Besides, thanks to the dynamic fusion mechanism, the features learned by DFI are much more powerful compared to [10], [71]. This is because the areas with no edges are rendered much cleaner. To sum up, in spite of the improvements in ODS and OIS, the quality of our results is much higher visually.

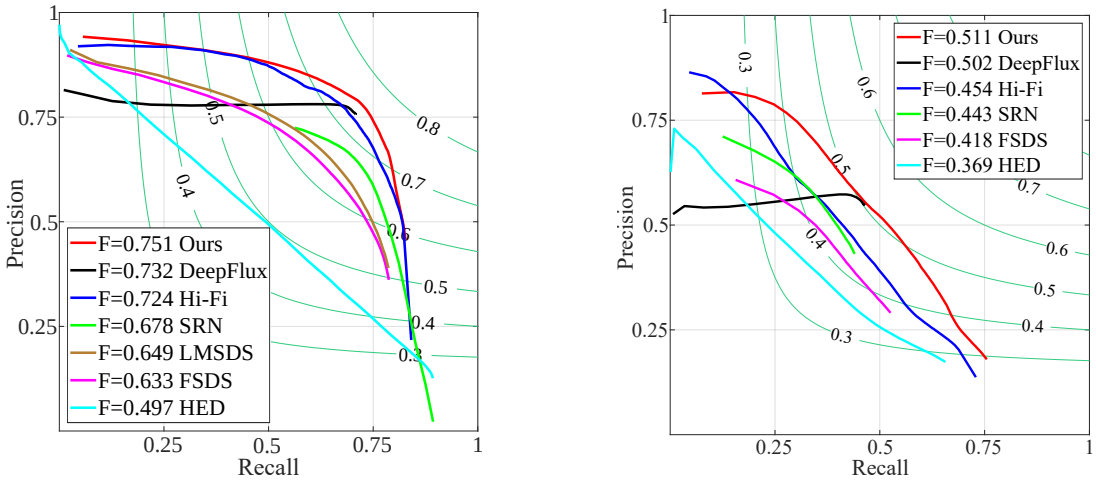
C. Skeleton Extraction

We compare DFI with 9 recent CNN-based methods including MIL [77], HED [10], RCF [3], FSDS [12], LMSDS [14], SRN [13], LSN [130], Hi-Fi [5], and DeepFlux [81] on 2 popular and challenging datasets including SK-LARGE [14] and SYM-PASCAL [13]. For fair comparisons, we train two different models using these two datasets separately, as done in the above methods.



(a) Image (b) GT (c) Ours (d) CED [72] (e) RCF [3] (f) HED [10]

Fig. 9. Visual comparisons with several recent state-of-the-art edge detectors. As can be seen, DFI is able to not only generate cleaner background but also more confident on the object boundaries compared to the other methods.



(a) SK-LARGE [14] (b) SYM-PASCAL [13]

Fig. 10. The precision-recall curves of some selected skeleton extraction methods on SK-LARGE dataset [14] and SYM-PASCAL dataset [13].

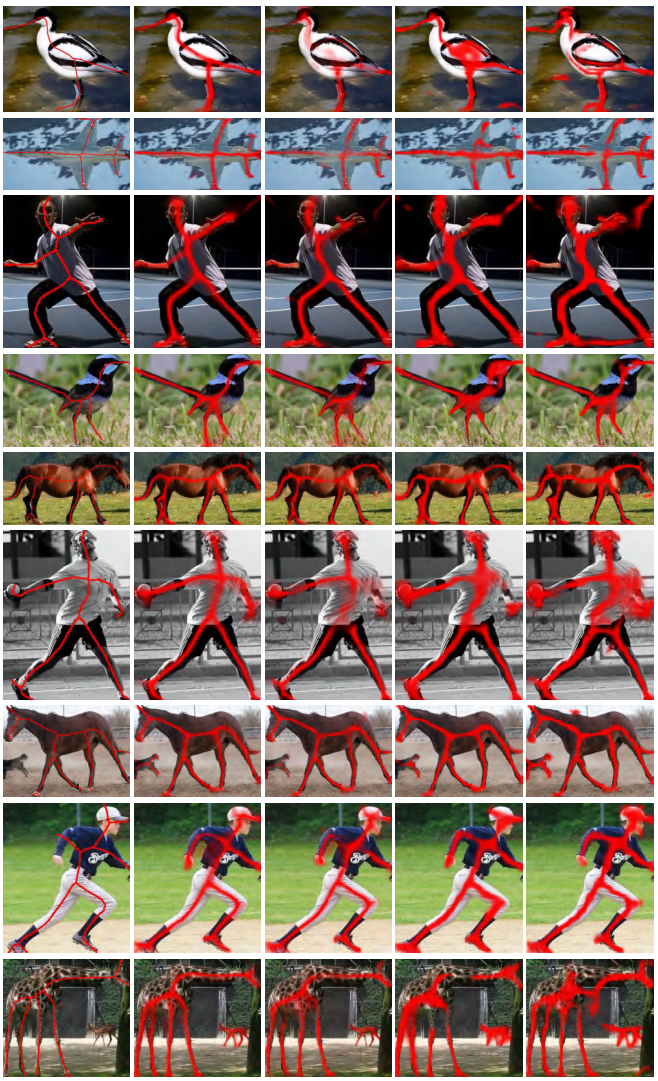
TABLE VIII
AVERAGE SPEED (FPS) COMPARISONS BETWEEN DFI AND THE PREVIOUS STATE-OF-THE-ART METHODS.

	DFI(Multi)	**DFI(Single)**	BASNet [54]	AFNet [51]	PiCANet [45]
Size	400 × 300	400 × 300	256 × 256	224 × 224	224 × 224
Speed	40	57	25	26	7
	PAGE [50]	CPD [40]	DGRL [30]	Amulet [37]	DSS [1]
Size	224 × 224	352 × 352	384 × 384	256 × 256	400 × 300
Speed	25	61	8	16	12
	RCF [3]	CED [72]	LPCB [129]	Hi-Fi [5]	DeepFlux [81]
Size	480 × 320	480 × 320	480 × 320	300 × 200	300 × 200
Speed	36	35	35	32	55

Quantitative Analysis. In Tab. VII, we show quantitative comparisons with existing methods. As can be seen, DFI wins dramatically by a large margin (1.9 points) on the SK-LARGE dataset [12]. There is also an improvement of 0.9 points on the SYM-PASCAL dataset [13].

PR Curves. In Fig. 10, we also show the precision-recall curves of our approach with some selected skeleton extraction methods. As can be seen, quantitatively, our approach on both datasets substantially outperforms other existing methods with a clear margin.

Visual Analysis. In Fig. 11, we show some visual comparisons. Owing to the advanced features integration strategy that is performed dynamically, DFI is able to locate the exact positions of the skeletons more accurately. This point can also be substantiated by the fact that our prediction maps are much thinner and stronger than other works. Both quantitative and visual results unveil that DFI provides a better way to combine different-level features for skeleton extraction, even in a multi-task manner.



(a) GT (b) Ours (c) Hi-Fi [5] (d) SRN [13] (e) FSDS [12]

Fig. 11. Visual comparisons with three recently representative skeleton extraction approaches. It can be easily found that our results are much thinner and stronger than the other three methods. Also, the skeletons produced by our results are continuous, which is essential for their applications.

D. Comparisons of Running Time

As shown in Tab. VIII, we compare the speed of DFI against other open source methods evaluated in our paper including all three tasks. We report average speed (fps) of different methods as well as the corresponding input size below (tested in the same environment). DFI can run at 57 FPS in single-task mode which is comparable to other methods while producing better detection results. Also, DFI runs at 40 FPS even in multi-task mode which means predicting three different tasks at the same time.

E. Ablation Study of Training Time

There is no existing method that performs all the three tasks simultaneously. To highlight the impact brought by the proposed DFIM and TAM, we compare the proposed method with its baseline version (the 7th row *vs.* the 3rd row in Tab. III) for demonstration. It takes about 30 hours for the proposed method to be trained while 25 hours for the baseline method. With about 20% more training time introduced by additional parameters in DFIM and TAM, the proposed method has much better and balanced overall performances across all tasks.

TABLE IX
ABLATION ANALYSIS OF THE IMPACT OF IMAGENET PRE-TRAINING.

Schedule	ImageNet Pre-training	Saliency			Edge		Skeleton
		DUTS-TE			BSDS 500		SK-LAR
		$F_\beta \uparrow$	MAE $\downarrow$	$S_m \uparrow$	ODS $\uparrow$	OIS $\uparrow$	$F_m \uparrow$
$1\times$	w/o	0.819	0.064	0.831	0.786	0.809	0.663
$4\times$	w/o	0.841	0.053	0.848	0.799	0.820	0.738
$1\times$	w/	**0.888**	**0.038**	**0.887**	**0.819**	**0.836**	**0.751**

F. Impact of ImageNet Pre-training

In the above sections, we conduct experiments using ImageNet pre-training as previous methods of the three tasks for fair comparisons. Here we investigate the impact of using ImageNet pre-training on the overall performance of the proposed method. When training from scratch, all parameters in the network are randomly initialized. And all other training settings are the same except for special declaration. By comparing the 1st and 3rd rows of Tab. IX, we can see that when training for $1\times$ schedule ($\sim$ 12 epochs), the overall performance of the version w/o ImageNet pre-training are much worse than the one w/ pre-training. Even after training for $4\times$ schedule ($\sim$ 48 epochs, the learning rate is divided by 10 after 36 epochs), there is still a clear margin between the overall performances. During the training process, we observe that when using ImageNet pre-training, the loss decreases and converges rapidly in the early stages while the randomly initialized version takes much longer iterations to converge. The ImageNet dataset has $\sim$ 1.28M (1,281,167) images, which is $\sim 59\times$ larger than the number of images (21,702, described in Tab. I) used by the three tasks. When training from scratch, it is insufficient to optimize a network well with only $\sim$ 22K images. We argue that the three tasks all take natural image as input, where the ImageNet pre-training helps in providing powerful feature extraction capabilities at the beginning of training. When training from scratch the model has to learn how to extract features effectively, and more iterations are required to converge. By extending the training schedule, the randomly initialized model may converge at last, but the gap caused by lacking of sufficient feature extraction capabilities can not be easily narrowed.

VII. CONCLUSION

In this paper, we solve three different low-level pixel-wise prediction tasks simultaneously, including salient object segmentation, edge detection, and skeleton extraction. We propose a dynamic feature integration module (DFIM) to learn the feature integration strategy for each task dynamically, and a task-adaptive attention module (TAM) to allocate information across tasks for better overall convergence. Experiments on a wide range of datasets show that DFI can perform comparably sometimes even better than the state-of-the-art methods of the solved tasks. DFI is fast as well, which can perform these three pixel-wise prediction tasks simultaneously with a speed of 40 FPS.

ACKNOWLEDGMENTS

This research was supported by Major Project for New Generation of AI under Grant No. 2018AAA0100400, NSFC (61922046), the national youth talent support program, and Tianjin Natural Science Foundation (18ZXZNGX00110).

REFERENCES

[1] Q. Hou, M.-M. Cheng, X. Hu, A. Borji, Z. Tu, and P. Torr, "Deeply supervised salient object detection with short connections," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 4, pp. 815–828, 2019.

[2] M.-M. Cheng, Q.-B. Hou, S.-H. Zhang, and P. L. Rosin, "Intelligent visual media processing: When graphics meets vision," *Journal of Computer Science and Technology*, vol. 32, no. 1, pp. 110–121, 2017.

[3] Y. Liu, M.-M. Cheng, X. Hu, J.-W. Bian, L. Zhang, X. Bai, and J. Tang, "Richer convolutional features for edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 8, pp. 1939 – 1946, 2019.

[4] M.-M. Cheng, X.-C. Liu, J. Wang, S.-P. Lu, Y.-K. Lai, and P. L. Rosin, "Structure-preserving neural style transfer," *IEEE Trans. Image Process.*, vol. 29, pp. 909–920, 2020.

[5] K. Zhao, W. Shen, S. Gao, D. Li, and M.-M. Cheng, "Hi-Fi: Hierarchical feature integration for skeleton detection," in *Int. Joint Conf. Artif. Intell.*, 2018, pp. 1191–1197.

[6] I. Kokkinos, "Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 6129–6138.

[7] R. Wu, M. Feng, W. Guan, D. Wang, H. Lu, and E. Ding, "A mutual learning method for salient object detection with intertwined multi-supervision," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[8] J.-J. Liu, Q. Hou, M.-M. Cheng, J. Feng, and J. Jiang, "A simple pooling-based design for real-time salient object detection," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[9] Z. Wu, L. Su, and Q. Huang, "Stacked cross refinement network for edge-aware salient object detection," in *Int. Conf. Comput. Vis.*, 2019.

[10] S. Xie and Z. Tu, "Holistically-nested edge detection," in *Int. Conf. Comput. Vis.*, 2015, pp. 1395–1403.

[11] K.-K. Maninis, J. Pont-Tuset, P. Arbelaez, and L. Van Gool, "Convolutional oriented boundaries: From image segmentation to high-level tasks," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2017.

[12] W. Shen, K. Zhao, Y. Jiang, Y. Wang, Z. Zhang, and X. Bai, "Object skeleton extraction in natural images by fusing scale-associated deep side outputs," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016, pp. 222–230.

[13] W. Ke, J. Chen, J. Jiao, G. Zhao, and Q. Ye, "Srn: Side-output residual network for object symmetry detection in the wild," *arXiv preprint arXiv:1703.02243*, 2017.

[14] W. Shen, K. Zhao, Y. Jiang, Y. Wang, X. Bai, and A. Yuille, "Deepskeleton: Learning multi-task scale-associated deep side outputs for object skeleton extraction in natural images," *IEEE Trans. Image Process.*, vol. 26, no. 11, pp. 5298–5311, 2017.

[15] M. Cheng, N. J. Mitra, X. Huang, P. H. Torr, and S. Hu, "Global contrast based salient region detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2015.

[16] X. Huang and Y.-J. Zhang, "300-fps salient object detection via minimum directional contrast," *IEEE Trans. Image Process.*, vol. 26, no. 9, pp. 4243–4254, 2017.

[17] T. Liu, Z. Yuan, J. Sun, J. Wang, N. Zheng, X. Tang, and H.-Y. Shum, "Learning to detect a salient object," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 2, pp. 353–367, 2011.

[18] J. Wang, H. Jiang, Z. Yuan, M.-M. Cheng, X. Hu, and N. Zheng, "Salient object detection: A discriminative regional feature integration approach," *International Journal of Computer Vision*, vol. 123, no. 2, pp. 251–268, 2017.

[19] F. Perazzi, P. Krähenbühl, Y. Pritch, and A. Hornung, "Saliency filters: Contrast based filtering for salient region detection," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2012, pp. 733–740.

[20] A. Borji, M.-M. Cheng, Q. Hou, H. Jiang, and J. Li, "Salient object detection: A survey," *Computational Visual Media*, vol. 5, no. 2, pp. 117–150, 2019.

[21] M.-M. Cheng, N. Mitra, X. Huang, and S.-M. Hu, "Salientshape: group saliency in image collections," *The Visual Computer*, vol. 30, no. 4, pp. 443–453, 2014.

[22] G. Li and Y. Yu, "Visual saliency based on multiscale deep features," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 5455–5463.

[23] L. Wang, H. Lu, X. Ruan, and M.-H. Yang, "Deep networks for saliency detection via local estimation and global search," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 3183–3192.

[24] R. Zhao, W. Ouyang, H. Li, and X. Wang, "Saliency detection by multi-context deep learning," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 1265–1274.

[25] R. Fan, M.-M. Cheng, Q. Hou, T.-J. Mu, J. Wang, and S.-M. Hu, "S4net: Single stage salient-instance segmentation," *Computational Visual Media*, vol. 6, no. 2, pp. 191–204, June 2020.

[26] L. Gayoung, T. Yu-Wing, and K. Junmo, "Deep saliency with encoded low level distance map and high level features," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016.

[27] S. He, R. Lau, W. Liu, Z. Huang, and Q. Yang, "Supercnn: A superpixelwise convolutional neural network for salient object detection," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 330–344, 2015.

[28] L. Wang, L. Wang, H. Lu, P. Zhang, and X. Ruan, "Saliency detection with recurrent fully convolutional networks," in *Eur. Conf. Comput. Vis.*, 2016.

[29] N. Liu and J. Han, "Dhsnet: Deep hierarchical saliency network for salient object detection," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016.

[30] T. Wang, L. Zhang, S. Wang, H. Lu, G. Yang, X. Ruan, and A. Borji, "Detect globally, refine locally: A novel approach to saliency detection," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 3127–3135.

[31] X. Zhang, T. Wang, J. Qi, H. Lu, and G. Wang, "Progressive attention guided recurrent network for salient object detection," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 714–722.

[32] W. Wang, J. Shen, M.-M. Cheng, and L. Shao, "An iterative and cooperative top-down and bottom-up inference network for salient object detection," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[33] Y. Xu, D. Xu, X. Hong, W. Ouyang, R. Ji, M. Xu, and G. Zhao, "Structured modeling of joint deep feature and prediction refinement for salient object detection," in *ICCV*, 2019.

[34] G. Li and Y. Yu, "Deep contrast learning for salient object detection," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016.

[35] G. Li, Y. Xie, L. Lin, and Y. Yu, "Instance-level salient object segmentation," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017.

[36] Z. Luo, A. K. Mishra, A. Achkar, J. A. Eichel, S. Li, and P.-M. Jodoin, "Non-local deep features for salient object detection." in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017.

[37] P. Zhang, D. Wang, H. Lu, H. Wang, and X. Ruan, "Amulet: Aggregating multi-level convolutional features for salient object detection," in *Int. Conf. Comput. Vis.*, 2017.

[38] P. Zhang, D. Wang, H. Lu, H. Wang, and B. Yin, "Learning uncertain convolutional features for accurate saliency detection," in *Int. Conf. Comput. Vis.*, 2017.

[39] L. Zhang, J. Dai, H. Lu, Y. He, and G. Wang, "A bi-directional message passing model for salient object detection," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 1741–1750.

[40] Z. Wu, L. Su, and Q. Huang, "Cascaded partial decoder for fast and accurate salient object detection," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[41] Y. Xu, D. Xu, X. Hong, W. Ouyang, R. Ji, M. Xu, and G. Zhao, "Structured modeling of joint deep feature and prediction refinement for salient object detection," in *Int. Conf. Comput. Vis.*, 2019.

[42] Y. Zeng, P. Zhang, J. Zhang, Z. Lin, and H. Lu, "Towards high-resolution salient object detection," in *Int. Conf. Comput. Vis.*, 2019.

[43] S.-H. Gao, M.-M. Cheng, K. Zhao, X.-Y. Zhang, M.-H. Yang, and P. Torr, "Res2net: A new multi-scale backbone architecture," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2020.

[44] S. Chen, X. Tan, B. Wang, and X. Hu, "Reverse attention for salient object detection," in *Eur. Conf. Comput. Vis.*, 2018, pp. 234–250.

[45] N. Liu, J. Han, and M.-H. Yang, "Picanet: Learning pixel-wise contextual attention for saliency detection," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 3089–3098.

[46] T. Zhao and X. Wu, "Pyramid feature attention network for saliency detection," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[47] H. Xiao, J. Feng, Y. Wei, M. Zhang, and S. Yan, "Deep salient object detection with dense connections and distraction diagnosis," *IEEE Transactions on Multimedia*, 2018.

[48] T. Wang, A. Borji, L. Zhang, P. Zhang, and H. Lu, "A stagewise refinement model for detecting salient objects in images," in *Int. Conf. Comput. Vis.*, 2017, pp. 4019–4028.

[49] X. Li, F. Yang, H. Cheng, W. Liu, and D. Shen, "Contour knowledge transfer for salient object detection," in *Eur. Conf. Comput. Vis.*, 2018, pp. 355–370.

[50] W. Wang, S. Zhao, J. Shen, S. C. H. Hoi, and A. Borji, "Salient object detection with pyramid attention and salient edges," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[51] M. Feng, H. Lu, and E. Ding, "Attentive feedback network for boundary-aware salient object detection," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[52] D.-P. Fan, M.-M. Cheng, J.-J. Liu, S.-H. Gao, Q. Hou, and A. Borji, "Salient objects in clutter: Bringing salient object detection to the foreground," in *Eur. Conf. Comput. Vis.*, 2018.

[53] L. Zhang, J. Zhang, Z. Lin, H. Lu, and Y. He, "Capsal: Leveraging captioning to boost semantics for salient object detection," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[54] X. Qin, Z. Zhang, C. Huang, C. Gao, M. Dehghan, and M. Jagersand, "Basnet: Boundary-aware salient object detection," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[55] J. Su, J. Li, Y. Zhang, C. Xia, and Y. Tian, "Selectivity or invariance: Boundary-aware salient object detection," in *Int. Conf. Comput. Vis.*, 2019.

[56] J.-X. Zhao, J.-J. Liu, D.-P. Fan, Y. Cao, J. Yang, and M.-M. Cheng, "Egnet: Edge guidance network for salient object detection," in *Int. Conf. Comput. Vis.*, 2019.

[57] K. Zhao, S. Gao, W. Wang, and M.-M. Cheng, "Optimizing the f-measure for threshold-free salient object detection," in *Int. Conf. Comput. Vis.*, 2019.

[58] Y. Liu, Q. Zhang, D. Zhang, and J. Han, "Employing deep part-object relationships for salient object detection," in *Int. Conf. Comput. Vis.*, 2019.

[59] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, no. 6, pp. 679–698, 1986.

[60] D. Marr and E. Hildreth, "Theory of edge detection," *Proceedings of the Royal Society of London B: Biological Sciences*, vol. 207, no. 1167, pp. 187–217, 1980.

[61] V. Torre and T. A. Poggio, "On edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, no. 2, pp. 147–163, 1986.

[62] S. Konishi, A. L. Yuille, J. M. Coughlan, and S. C. Zhu, "Statistical edge detection: Learning and evaluating edge cues," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 1, pp. 57–74, 2003.

[63] D. R. Martin, C. C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 5, pp. 530–549, 2004.

[64] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 898–916, 2011.

[65] Y. Ganin and V. Lempitsky, "Nˆ 4-fields: Neural network nearest neighbor fields for image transforms," in *Asian Conf. Comput. Vis.* Springer, 2014, pp. 536–551.

[66] W. Shen, X. Wang, Y. Wang, X. Bai, and Z. Zhang, "Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 3982–3991.

[67] G. Bertasius, J. Shi, and L. Torresani, "Deepedge: A multi-scale bifurcated deep network for top-down contour detection," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 4380–4389.

[68] J.-J. Hwang and T.-L. Liu, "Pixel-wise deep learning for contour detection," in *ICLR*, 2015.

[69] I. Kokkinos, "Pushing the boundaries of boundary detection using deep learning," *arXiv preprint arXiv:1511.07386*, 2015.

[70] J. Yang, B. Price, S. Cohen, H. Lee, and M.-H. Yang, "Object contour detection with a fully convolutional encoder-decoder network," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016, pp. 193–202.

[71] Y. Liu and M. S. Lew, "Learning relaxed deep supervision for better edge detection," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016, pp. 231–240.

[72] Y. Wang, X. Zhao, and K. Huang, "Deep crisp boundaries," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 3892–3900.

[73] J. He, S. Zhang, M. Yang, Y. Shan, and T. Huang, "Bi-directional cascade network for perceptual edge detection," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[74] Z. Yu and C. Bajaj, "A segmentation-free approach for skeletonization of gray-scale images via anisotropic vector diffusion," in *IEEE Conf. Comput. Vis. Pattern Recog.* IEEE, 2004.

[75] J.-H. Jang and K.-S. Hong, "A pseudo-distance map for the segmentation-free skeletonization of gray-scale images," in *Int. Conf. Comput. Vis.* IEEE, 2001.

[76] P. Majer, "On the influence of scale selection on feature detection for the case of linelike structures," *Int. J. Comput. Vis.*, vol. 60, no. 3, pp. 191–202, 2004.

[77] S. Tsogkas and I. Kokkinos, "Learning-based symmetry detection in natural images," in *Eur. Conf. Comput. Vis.* Springer, 2012, pp. 41–54.

[78] A. Sironi, V. Lepetit, and P. Fua, "Multiscale centerline detection by learning a scale-space distance transform," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2014.

[79] A. Levinshtein, C. Sminchisescu, and S. Dickinson, "Multiscale symmetric part detection and grouping," *Int. J. Comput. Vis.*, vol. 104, no. 2, pp. 117–134, 2013.

[80] N. Widynski, A. Moevus, and M. Mignotte, "Local symmetry detection in natural images using a particle filtering approach," *IEEE Trans. Image Process.*, vol. 23, no. 12, pp. 5309–5322, 2014.

[81] Y. Wang, Y. Xu, S. Tsogkas, X. Bai, S. Dickinson, and K. Siddiqi, "Deepflux for skeletons in the wild," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019, pp. 5287–5296.

[82] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.

[83] C. Doersch and A. Zisserman, "Multi-task self-supervised visual learning," in *Int. Conf. Comput. Vis.*, 2017, pp. 2051–2060.

[84] T. Evgeniou and M. Pontil, "Regularized multi–task learning," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2004, pp. 109–117.

[85] A. Kumar and H. Daume III, "Learning task grouping and overlap in multi-task learning," *arXiv preprint arXiv:1206.6417*, 2012.

[86] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert, "Cross-stitch networks for multi-task learning," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016, pp. 3994–4003.

[87] C. Ahn, E. Kim, and S. Oh, "Deep elastic networks with model selection for multi-task learning," in *Int. Conf. Comput. Vis.*, 2019.

[88] G. Strezoski, N. v. Noord, and M. Worring, "Many task learning with task routing," in *Int. Conf. Comput. Vis.*, 2019.

[89] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 7482–7491.

[90] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich, "Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks," in *Int. Conf. Mach. Learn.*, 2018, pp. 793–802.

[91] S. Liu, E. Johns, and A. J. Davison, "End-to-end multi-task learning with attention," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019, pp. 1871–1880.

[92] S.-A. Rebuffi, H. Bilen, and A. Vedaldi, "Learning multiple visual domains with residual adapters," in *Adv. Neural Inform. Process. Syst.*, 2017, pp. 506–516.

[93] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," in *Int. Conf. Learn. Represent.*, 2014.

[94] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Adv. Neural Inform. Process. Syst.*, 2015, pp. 91–99.

[95] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *ICCV*, 2017.

[96] G. Gkioxari, B. Hariharan, R. Girshick, and J. Malik, "R-cnns for pose estimation and action detection," *arXiv preprint arXiv:1406.5212*, 2014.

[97] A. Kendall, M. Grimes, and R. Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization," in *Int. Conf. Comput. Vis.*, 2015, pp. 2938–2946.

[98] K. Du, X. Lin, Y. Sun, and X. Ma, "Crossinfonet: Multi-task information sharing based hand pose estimation," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[99] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *Int. Conf. Comput. Vis.*, 2015, pp. 2650–2658.

[100] M. Teichmann, M. Weber, M. Zoellner, R. Cipolla, and R. Urtasun, "Multinet: Real-time joint semantic reasoning for autonomous driving," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1013–1020.

[101] 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 *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[102] G.-J. Qi, "Hierarchically gated deep networks for semantic segmentation," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016, pp. 2267–2275.

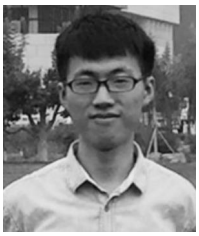
[103] T. Takikawa, D. Acuna, V. Jampani, and S. Fidler, "Gated-scnn: Gated shape cnns for semantic segmentation," in *Int. Conf. Comput. Vis.*, 2019, pp. 5229–5238.

[104] H. Ding, X. Jiang, B. Shuai, A. Qun Liu, and G. Wang, "Context contrasted feature and gated multi-scale aggregation for scene segmentation," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 2393–2402.

[105] Y. Cheng, R. Cai, Z. Li, X. Zhao, and K. Huang, "Locality-sensitive deconvolution networks with gated fusion for rgb-d indoor semantic segmentation," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 3029–3037.

[106] C. Zhu, Y. He, and M. Savvides, "Feature selective anchor-free module for single-shot object detection," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019, pp. 840–849.
[107] S. Li, L. Yang, J. Huang, X.-S. Hua, and L. Zhang, "Dynamic anchor feature selection for single-shot object detection," in *Int. Conf. Comput. Vis.*, 2019, pp. 6609–6618.
[108] Z. Chen, Y. Li, S. Bengio, and S. Si, "You look twice: Gaternet for dynamic filter selection in cnns," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019, pp. 9172–9180.
[109] X. Li, W. Wang, X. Hu, and J. Yang, "Selective kernel networks," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019, pp. 510–519.
[110] J.-J. Liu, Q. Hou, M.-M. Cheng, C. Wang, and J. Feng, "Improving convolutional networks with self-calibrated convolutions," in *IEEE Conf. Comput. Vis. Pattern Recog.*, June 2020.
[111] W. Hua, Y. Zhou, C. M. De Sa, Z. Zhang, and G. E. Suh, "Channel gating neural networks," in *Adv. Neural Inform. Process. Syst.*, 2019, pp. 1884–1894.
[112] L. Wang, H. Lu, Y. Wang, M. Feng, D. Wang, B. Yin, and X. Ruan, "Learning to detect salient objects with image-level supervision," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 136–145.
[113] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille, "The role of context for object detection and semantic segmentation in the wild," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2014, pp. 891–898.
[114] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016.
[115] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017.
[116] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 7132–7141.
[117] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.
[118] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Int. Conf. Learn. Represent.*, 2015.
[119] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Adv. Neural Inform. Process. Syst.*, 2012.
[120] Y. Wu and K. He, "Group normalization," in *Eur. Conf. Comput. Vis.*, 2018, pp. 3–19.
[121] Q. Yan, L. Xu, J. Shi, and J. Jia, "Hierarchical saliency detection," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2013, pp. 1155–1162.
[122] Y. Li, X. Hou, C. Koch, J. M. Rehg, and A. L. Yuille, "The secrets of salient object segmentation," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2014, pp. 280–287.
[123] C. Yang, L. Zhang, H. Lu, X. Ruan, and M.-H. Yang, "Saliency detection via graph-based manifold ranking," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2013, pp. 3166–3173.
[124] V. Movahedi and J. H. Elder, "Design and perceptual validation of performance measures for salient object segmentation," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2010, pp. 49–56.
[125] D.-P. Fan, M.-M. Cheng, Y. Liu, T. Li, and A. Borji, "Structure-measure: A New Way to Evaluate Foreground Maps," in *Int. Conf. Comput. Vis.*, 2017, pp. 4548–4557.
[126] P. Dollár and C. L. Zitnick, "Fast edge detection using structured forests," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 8, pp. 1558–1570, 2015.
[127] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.
[128] J. Pont-Tuset, P. Arbelaez, J. T. Barron, F. Marques, and J. Malik, "Multiscale combinatorial grouping for image segmentation and object proposal generation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 1, pp. 128–140, 2017.
[129] R. Deng, C. Shen, S. Liu, H. Wang, and X. Liu, "Learning to predict crisp boundaries," in *Eur. Conf. Comput. Vis.*, 2018, pp. 562–578.
[130] C. Liu, W. Ke, F. Qin, and Q. Ye, "Linear span network for object skeleton detection," in *Eur. Conf. Comput. Vis.*, 2018, pp. 133–148.

Jiang-Jiang Liu is currently a Ph.D. candidate with School of Computer Science, Nankai University, under the supervision of Prof. Ming-Ming Cheng. His research interests include deep learning, image processing, and computer vision.



Qibin Hou received his PhD degree from Nankai University in 2019. He is currently a research fellow at the Department of Electrical and Computer Engineering, National University of Singapore, working with Prof. Jiashi Feng. His research interests include deep learning and computer vision.

Ming-Ming Cheng received his PhD degree from Tsinghua University in 2012. Then he did 2 years research fellow, with Prof. Philip Torr in Oxford. He is now a professor at Nankai University, leading the Media Computing Lab. His research interests includes computer graphics, computer vision, and image processing. He has published 60+ refereed research papers, with $16,000+$ Google Scholar citations. He received research awards including ACM China Rising Star Award, IBM Global SUR Award, *etc.* He is a senior member of IEEE and on the editor board of IEEE TIP.