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—In this paper, we solve three low-level pixel-wise vision problems, including salient object segmentation, edge detection, and skeleton extraction, within a unified framework. 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 framework 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. To facilitate future research, source code will be released.

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 being ported from computer platforms to mobile platforms. Many low-level computer vision tasks, benefiting 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]. To help users taking pictures with more visual pleasing compositions, the edge detection task is adopted to obtain structure information [2]. And the skeleton extraction task plays an important role in supporting taking photos by gesturing and instructing users with more interesting poses [3]. 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.

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 [4]–[7] 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 [4]. 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 aims at detecting accurate boundaries 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 [8], [9]. Skeleton extraction [10], [11] 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 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

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.
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 [12]. 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 [13]–[17]. With the popularity of CNNs, many methods [18]–[22] started to use CNNs to extract features. Some of them [23]–[28] 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 [1], [29]–[37], introducing attention mechanism [26], [38]–[40], using multiple stages to learn the prediction in a stage-wise manner [25], [41], [42], or adding more supervisions to get predictions with sharper edges [5]–[7], [43]–[52]. For edge detection, early works [53]–[55] mostly relied on various gradient operators. Later works [56]–[58] further employed manually-designed features. Recently, CNN-based methods solved this problem by using fully-convolutional networks in a patch-wise [59]–[62] or pixel-wise prediction manner [2], [8], [9], [63]–[67]. For skeleton extraction, earlier methods [68]–[70] mainly relied on gradient intensity maps of natural images to extract skeletons. Later, learning-based methods [71]–[74] viewed skeleton extraction as a per-pixel classification problem or a super-pixel clustering problem. Recent methods [3], [10], [11], [75] 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 [76]–[79]. Recently, many CNN-based MTL methods had been proposed, most of which focused on the design of network architecture [4], [80]–[82], or loss functions to balance the importance of different tasks [83], [84], or both of them [85]. Different work also solved different task combinations, including: image classification in multiple domains [86]; object recognition, localization, and detection [87]–[89]; pose estimation and action recognition [90]–[92]; semantic classes, surface normals, and depth prediction [80], [83], [85].
### III. Method

#### A. Overall Pipeline

We include three different tasks on multiple individual datasets (i.e., DUTS [96] for saliency, BSDS 500 [58] and VOC Context [97] for edge, SK-LARGE [10] or SYMPASCAL [11] 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.

The overall pipeline of the proposed framework is illustrated in Fig. 2. We employ the ResNet-50 [98] network as the feature extractor. We take the feature maps outputted by conv1 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 $3 \times 3$ convolutional layers in conv5 to 2 as done in pixel-wise prediction tasks. Moreover, we add a pyramid pooling module (PPM) [99] on the top of ResNet-50 to capture more global information as done in [6], [42]. 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 levels according to the favors of each task and input during both training and testing. Compared to existing methods that 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.

#### B. Dynamic Feature Integration

It has been mentioned in many previous multi-task methods [4], [80], [83], [85] 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 $\{S_i\} (1 \leq i \leq M, M = 6)$ extracted from the backbone as input for each DFIM. And the demanded output down-sampling rate $r_x$ of each DFIM is determined during the network definition period. As illustrated in Fig. 3, we first transfer $\{S_i\} (1 \leq i \leq M, M = 6)$ to the same number of channels (i.e., $C \times r$) and down-sampling rate (i.e., $\times r$), denoted as $\{S'_i\} (1 \leq i \leq M, M = 6)$, through a $1 \times 1$ convolutional layer and bilinear interpolation, respectively. To give DFIM a view of all the features to be selected, we summarize $\{S'_i\}$, and follow it with a global average pooling (GAP) layer to create a compact and global feature ($C \times$) as done in [4], [5].

(i.e. $\{S'_i\}$ $(1 \leq i \leq M, M = 6)$) 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 summarized and then followed by a $1 \times 1$ convolutional layer for final prediction, respectively.
the task-specific feature maps of features that different tasks favor vary greatly. If we use maintaining of the overall performance across all tasks. As datasets effectively and efficiently is indispensable to the be ignored. How to coalesce the information from diverse datasets, the domain shifting [76], [101] problem can not C. Task-Adaptive Attention

For each task ∈ {saliency, edge, skeleton}, we use an independent fully connected (FC) layer to map the C× feature to M × channels, and then apply a softmax operator to transform the M × feature into the form of probability \{p^r_{i,task}\} (1 ≤ i ≤ M) that could be used as an indicator to select features. Different from those who keep dense connections of {S^i r}, we only keep half the connections as {S^i r,task} (1 ≤ i ≤ M):

\[
S^i r,task = \begin{cases} 
    p^r_{i,task} \ast S^i r, & \text{if } p^r_{i,task} \geq \text{median}(\{p^r_{i,task}\}) \\
    0, & \text{else}
\end{cases}
\]

in which median(·) means taking the median. Thus the output of DFIM with down-sampling rate r for the task can be obtained with

\[
D^r,task = \sum_i S^i r,task.
\]

By arranging a series of DFIMs of various down-sampling rates, we can obtain the dynamically integrated feature maps \{D^r,task\}(r ∈ \{2, 4, 8, 16\}, task ∈ {saliency, edge, skeleton}) from the DFIM with a down-sampling rate of r for each task. As shown in the top right corner of Fig. 2, the output feature maps \{D^r,task\}(task ∈ {saliency, edge, skeleton}) from the DFIM are further rescaled by being forwarded to a TAM. The parameters in TAM are shared 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 [4], [5], [85], 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 TAM (c,g) are more distinct saliency parts, sharper edges, and stronger skeletons.

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.

C. Task-Adaptive Attention

As we utilize training data from multiple individual datasets, the domain shifting [76], [101] 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,task\}(r ∈ \{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,task\}(task ∈ \{saliency, edge, skeleton\}) from the DFIM with a down-sampling rate of r are further rescaled by being forwarded to a TAM. The parameters in TAM are shared 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 [4], [5], [85], 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 TAM (c,g) become stronger and clearer after TAM. All the aforementioned discussions verify the significant effect of TAM on better allocating the information for different tasks.

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 implemented 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 [102] 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 learning rate is divided by 10 after 9 epochs. The parameters of the backbone (i.e., ResNet-50 [98]) of our network are initialized with ImageNet [103] pre-trained model, while all other parameters are randomly initialized. Group normalization [104] 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 followed using the existing single-purpose methods proposed for each task [2], [3], [32] with no extra pre-processing.

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 summarize the three calculated losses (integers), 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 as most of the previous single-purpose methods. We use standard binary cross-entropy loss for salient object segmentation [1], [32] and balanced binary cross-entropy loss [2], [3], [8] for edge detection and skeleton extraction. We simply summarize the losses of the three tasks as the overall loss. 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:

\[
\mathcal{L}_{\text{standard}}(\hat{Y}, Y) = - \sum_{i,j} [Y(i,j) \cdot \log \hat{Y}(i,j) + (1 - Y(i,j)) \cdot \log(1 - \hat{Y}(i,j))],
\]

and the balanced binary cross-entropy loss as:

\[
\mathcal{L}_{\text{balanced}}(\hat{Y}, Y) = - \sum_{i,j} [\beta \cdot Y(i,j) \cdot \log \hat{Y}(i,j) + (1 - \beta) \cdot (1 - Y(i,j)) \cdot \log(1 - \hat{Y}(i,j))],
\]

where \( \beta = |Y^-|/|Y^+ + Y^-| \).

- Salient Object Segmentation:

\[
\mathcal{L}_{\text{sal}}(\hat{Y}_{\text{sal}}, Y_{\text{sal}}) = \mathcal{L}_{\text{standard}}(\hat{Y}_{\text{sal}}, Y_{\text{sal}}).
\]

- Edge Detection:

\[
\mathcal{L}_{\text{edg}}(\hat{Y}_{\text{edg}}, Y_{\text{edg}}) = \mathcal{L}_{\text{balanced}}(\hat{Y}_{\text{edg}}, Y_{\text{edg}}),
\]

in which \( Y^+ \) refers to the edge pixels and \( Y^- \) refers to the non-edge pixels.

- Skeleton Extraction:

\[
\mathcal{L}_{\text{ske}}(\hat{Y}_{\text{ske}}, Y_{\text{ske}}) = \mathcal{L}_{\text{balanced}}(\hat{Y}_{\text{ske}}, Y_{\text{ske}}),
\]

in which \( Y^+ \) refers to the skeleton pixels and \( Y^- \) refers to the non-skeleton pixels.

Evaluation Criteria. For salient object segmentation, we use F-measure score \( F_\beta \), mean absolute error (MAE), precision-recall (PR) curves, and S-measure [109] for evaluation. And the hyper-parameter \( \beta^2 \) in \( F_\beta \) is set to 0.3 as done in previous work to weight precision more than recall. For edge detection, we use the fixed contour threshold (ODS) and per-image best threshold (OIS) as our measures. Before evaluation, we apply the standard non-maximal suppression (NMS) algorithm to get thinned edges. For skeleton extraction, the skeleton maps are NMS-thinned before evaluation. A series of precision/recall pairs are then obtained by applying different thresholds to the thinned skeleton maps to draw the PR-curve. And the F-measure score is obtained under the optimal threshold over the whole dataset.

V. ABALATION 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

---

**TABLE I: The datasets we use for training and testing.**

| Task       | Training #Images | Testing #Images |
|------------|-----------------|-----------------|
| Salient    | DUTS-TR [96] 10553 | ECSSD [105], PASCAL-S [106] 1000, 850, DUT-OMRON [107], SOD [108] 5166, 300, HKU-IS [18] 1447, 5019 |
| Edge       | BSDS500 [58] & VOC Context [97] 300 + | BSDS500 [58] 200 |
| Skeleton   | SK-LARGE [12] 746 | SK-LARGE [12] 745 |
|            | SYM-PASCAL [11] 648 | SYM-PASCAL [11] 788 |

**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.

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

\(^1\)https://pytorch.org
TABLE III: Quantitative salient object segmentation, edge detection and skeleton extraction results for 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 | Edge | Skeleton |
|-----|------|-----|------|----------|
|     |      |     | PASCAL-S [106] | DUTS-OMRON [107] | DUTS-TE [96] | BSDS 500 [58] | SK-LARGE [12] |
|     |      |     | $F_P \uparrow$ | $\text{MAE} \downarrow$ | $S_m \uparrow$ | $F_P \uparrow$ | $\text{MAE} \downarrow$ | $S_m \uparrow$ | $\text{OIS} \uparrow$ | $\text{OIS} \uparrow$ | $F_m \uparrow$ |
| 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 |
| 1   | sparse | w/o | 0.859 | 0.081 | 0.849 | 0.817 | 0.060 | 0.835 | 0.880 | 0.045 | 0.878 | 0.812 | 0.826 | 0.746 |
| 2   | independent |      | 0.877 | 0.062 | 0.865 | 0.818 | 0.056 | 0.836 | 0.885 | 0.038 | 0.886 | 0.811 | 0.824 | 0.708 |
| 2   | identity |      | 0.872 | 0.064 | 0.859 | 0.813 | 0.056 | 0.833 | 0.877 | 0.039 | 0.881 | 0.810 | 0.825 | 0.740 |
| 3   | dense | 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 |
| 3   | sparse | independent | 0.873 | 0.055 | 0.861 | 0.815 | 0.057 | 0.836 | 0.879 | 0.039 | 0.883 | 0.815 | 0.832 | 0.753 |
| 4   | 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)

| No. | Method | Dataset | Edge | Skeleton |
|-----|--------|---------|------|----------|
| 8   | UberNet [4] | BSDS500 | 0.823 | - | - | - | - | - | 0.785 | 0.805 | - |
| 9   | MLMS [5] | BSDS500 | 0.853 | 0.074 | 0.844 | 0.793 | 0.063 | 0.809 | 0.854 | 0.048 | 0.862 | 0.769 | 0.780 | - |

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 its 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. 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 phenomena 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 [4], [5], 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 $\{p_{r,\text{task}}^i\}$ 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], [2], [75]. 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.

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 $\{S_i\}$ ($1 \leq i \leq M, M =$...
As listed in Tab. IV, we conduct ablation experiments on the 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 \(_{m}\) | F \(_{\beta}\) \(\uparrow\) MAE \(\downarrow\) S \(_{m}\) | 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** |

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 [110]. For example, for edge detection, more lower-level feature maps are necessary for precise localization of edge pixels [2], [8], while for skeleton extraction, more higher-level information is essential to determine whether a pixel being skeleton or not [3], [75].

**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 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, compared to separately training (the 1st row), the performance of skeleton extraction decreases. 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...


| Model       | ECSSD [103] | PASCAL-S [106] | DUT-OMRON [107] | HKU-IS [18] | SOD [108] | DUTS-TE [96] |
|-------------|-------------|----------------|-----------------|-------------|-----------|--------------|
| DCL [29]    | 0.870       | 0.880          | 0.689           | 0.829       | 0.813     | -            |
| RFCN [23]   | 0.898       | 0.897          | 0.827           | 0.747       | 0.805     | -            |
| MSR [30]    | 0.903       | 0.905          | 0.838           | 0.790       | 0.805     | -            |
| DSS [11]    | 0.906       | 0.864          | 0.821           | 0.760       | 0.884     | -            |
| NLD [31]    | 0.903       | 0.867          | 0.822           | 0.753       | 0.879     | -            |
| Amulet [32] | 0.911       | 0.862          | 0.826           | 0.737       | 0.889     | -            |
| PAGR [26]   | 0.924       | 0.864          | 0.847           | 0.771       | 0.919     | -            |
| DGRL [25]   | 0.921       | 0.843          | 0.844           | 0.774       | 0.910     | -            |
| MLMS [5]    | 0.924       | 0.896          | 0.853           | 0.839       | 0.922     | -            |
| JDFPR [28]  | 0.925       | 0.502          | 0.854           | 0.828       | 0.930     | -            |
| PAG [44]    | 0.928       | 0.847          | 0.858           | 0.842       | 0.920     | -            |
| CapSal [47] | -           | -              | 0.862           | -           | 0.889     | -            |
| CDPS [35]   | 0.936       | 0.913          | 0.859           | 0.848       | 0.925     | -            |
| PicasNet [39]| 0.932      | 0.948          | 0.864           | 0.855       | 0.861     | -            |
| AFNet [45]  | 0.932       | 0.945          | 0.861           | 0.849       | 0.926     | -            |
| BSNNet [48] | 0.939       | 0.940          | 0.857           | 0.838       | 0.930     | -            |
| DFI (Ours)  | 0.945       | 0.038          | 0.800           | 0.865       | 0.839     | 0.934       |

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 [5], 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 [5] by a large margin (the 3rd and 9th rows). This phenomenon demonstrated the effectiveness of the proposed DFIM and TAM.

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 little literature has solved the three tasks jointly before, e.g. UberNet [4] (CVPR’17) and MLMS [5] (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 [2], [3], [8]. And for each task, all the predicted maps are evaluated with the same evaluation code.

A. Salient Object Segmentation

We exhaustively compare DFI with 16 existing state-of-the-art salient object segmentation methods including DCL [29], RFCN [23], MSR [30], DSS [11], NLDF [31], Amulet [32], PAGR [26], DGRL [25], MLMS [5], JDFPR [28], PAGE [44], CapSal [47], CDPS [35], PicasNet [39], AFNet [45], and BSNNet [48].
Fig. 7: Precision (vertical axis) recall (horizontal axis) curves on six popular salient object segmentation datasets.

Fig. 8: The precision-recall curves on BSDS 500 dataset [58].

TABLE VI: Quantitative comparisons of DFI with existing edge detection methods. The best results in each column are highlighted in **bold**.

| Method                  | BSDS 500 [58] |
|-------------------------|---------------|
|                         | ODS | OIS |
| gPb-owt-ucm [58]        | 0.726 | 0.757 |
| SE-Var [111]            | 0.746 | 0.767 |
| MCG [112]               | 0.747 | 0.779 |
| DeepEdge [61]           | 0.753 | 0.772 |
| DeepContour [60]        | 0.756 | 0.773 |
| HED [8]                 | 0.788 | 0.808 |
| CEDN [64]               | 0.788 | 0.804 |
| RDS [65]                | 0.792 | 0.810 |
| COB [9]                 | 0.793 | 0.820 |
| RCF [2]                 | 0.811 | 0.830 |
| DCNN+sPb [63]           | 0.813 | 0.831 |
| CED [66]                | 0.815 | 0.833 |
| LPCB [113]              | 0.815 | 0.834 |
| DFI (Ours)              | **0.819** | **0.836** |

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 [58] 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.

B. Edge Detection

We compare DFI with results from 13 existing state-of-the-art edge detection methods, including gPb-owt-ucm [58], SE-Var [111], MCG [112], DeepEdge [61], DeepContour [60], HED [8], CEDN [64], RDS [65], COB [9], RCF [2], DCNN+sPb [63], CED [66] and LPCB [113], most of which are CNN-based methods.
In Fig. 9, we show some visual comparisons between DFI and some leading representative methods [8], [65], [66]. 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 [8], [65]. 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 [71], HED [8], RCF [2], FSDS [10], LMSDS [12], SRN [11], LSN [114], Hi-Fi [3], and DeepFlux [75] on 2 popular and challenging datasets including SK-LARGE [12] and SYM-PASCAL [11]. For fair comparisons, we train two different models using these two datasets separately, as done in the above methods.

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 [10]. There is also an improvement of 0.9 points on the SYM-PASCAL dataset [11].

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.
TABLE VII: Quantitative comparisons of DFI with existing skeleton extraction methods. The best results in each column are highlighted in bold.

| Method    | SK-LARGE [12] | SYM-PASCAL [11] |
|-----------|----------------|-----------------|
| MIL [71]  | 0.353          | 0.174           |
| HED [8]   | 0.497          | 0.369           |
| RCF [2]   | 0.626          | 0.392           |
| FSDS [10] | 0.633          | 0.418           |
| LMSDS [12]| 0.649          |                 |
| SRN [11]  | 0.678          | 0.443           |
| LSN [114] | 0.668          | 0.425           |
| Hi-Fi [3] | 0.724          | 0.454           |
| DeepFlux [75] | 0.732 | 0.502 |
| DFI (Ours)| 0.751          | 0.511           |

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.

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.

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.

TABLE VIII: Average speed (FPS) comparisons between DFI and the previous state-of-the-art methods.

| Size      | DFI(Multi) | DFI(Single) | BASNet [48] | AFNet [45] | PtCANet [39] |
|-----------|------------|-------------|-------------|------------|--------------|
| Speed     | 40         | 57          | 25          | 26         | 7            |
| Size      | 400 × 300  | 400 × 300   | 356 × 256   | 224 × 224  | 224 × 224    |
| Speed     | 25         | 61          | 8           | 16         | 12           |
| Size      | 224 × 224  | 352 × 352   | 384 × 384   | 256 × 256  | 400 × 300    |
| Speed     | 36         | 35          | 35          | 32         | 55           |

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 TPAMI, vol. 41, no. 4, pp. 815–828, 2019.

[2] Y. Liu, M.-M. Cheng, X. Hu, J.-W. Bian, L. Zhang, X. Bai, and J. Tang, “Richer convolutional features for edge detection,” IEEE TPAMI, vol. 41, no. 8, pp. 1939–1946, 2019.

[3] K. Zhao, W. Shen, S. Gao, D. Li, and M.-M. Cheng, “Hi-Fi: Hierarchical feature integration for skeleton detection,” in IJCAI, 2018, pp. 1191–1197.
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 CVPR, 2016, pp. 222–230.

W. Ke, J. Chen, J. Jiao, G. Zhao, and Q. Ye, “Src: Side-output residual networks,” in CVPR, 2017.

W. Shen, K. Zhao, Y. Jiang, Y. Wang, X. Bai, and A. Yuille, “Deeptextor: Learning to detect a salient object,” IEEE TPAMI, vol. 33, no. 2, pp. 353–367, 2011.

H. Jiang, J. Wang, Z. Yuan, W. Yu, N. Zheng, and S. Li, “Salient object detection: A discriminative regional feature integration approach,” in CVPR, 2013, pp. 2083–2090.

F. Perazzi, P. Krähenbühl, Y. Pritch, and A. Homung, “Salience filters: Contrast based filtering for salient region detection,” in CVPR, 2012, pp. 733–740.

G. Li and Y. Yu, “Visual saliency based on multiscale deep features,” in CVPR, 2015, pp. 5455–5463.

L. Wang, H. Lu, X. Ruan, and M.-H. Yang, “Deep networks for saliency detection via minimum directional contrast,” IEEE TPAMI, vol. 36, no. 9, pp. 2423–2434, 2017.

T. Liu, Z. Yuan, J. Sun, J. Wang, N. Zheng, X. Tang, and H.-Y. Shum, “Learning to detect a salient object,” IEEE TPAMI, vol. 33, no. 2, pp. 353–367, 2011.

H. Wang, J. Wang, Z. Yuan, W. Yu, N. Zheng, and S. Li, “Salient object detection: A discriminative regional feature integration approach,” in CVPR, 2013, pp. 2083–2090.

F. Perazzi, P. Krähenbühl, Y. Pritch, and A. Homung, “Salience filters: Contrast based filtering for salient region detection,” in CVPR, 2012, pp. 733–740.

G. Li and Y. Yu, “Visual saliency based on multiscale deep features,” in CVPR, 2015, pp. 5455–5463.

L. Wang, H. Lu, X. Ruan, and M.-H. Yang, “Deep networks for saliency detection via local estimation and global search,” in CVPR, 2015, pp. 3183–3192.

R. Zhao, W. Ouyang, H. Li, and X. Wang, “Saliency detection by multi-context deep learning,” in CVPR, 2015, pp. 1265–1274.

L. Gayoung, T. Yu-Wing, and K. Junmo, “Deep saliency with encoded side outputs,” in CVPR, 2015, pp. 330–340, 2015.

L. Wang, L. Wang, H. Lu, P. Zhang, and X. Ruan, “Saliency detection with recurrent fully convolutional networks,” in ECCV, 2016.

N. Liu and J. Han, “Dhoset: Deep hierarchical saliency network for salient object detection,” in CVPR, 2016.

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 CVPR, 2018, pp. 3127–3135.

X. Zhang, T. Wang, J. Qi, H. Lu, and G. Wang, “Progressive attention guided recurrent network for salient object detection,” in CVPR, 2018, pp. 714–722.

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 CVPR, 2019.

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.

G. Li and Y. Yu, “Deep contrast learning for salient object detection,” in CVPR, 2016.

G. Li, Y. Xu, L. Lin, and Y. Yu, “Instance-level salient object segmentation,” in CVPR, 2017.

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 CVPR, 2017.

P. Zhang, D. Wang, H. Lu, H. Wang, and X. Ruan, “Amulet: Aggregating multi-level convolutional features for salient object detection,” in ICCV, 2017.

P. Zhang, D. Wang, H. Lu, H. Wang, and B. Yin, “Learning uncertain convolutional features for accurate salience detection,” in ICCV, 2017.
Y. Wang, X. Zhao, and K. Huang, “Deep crisp boundaries,” in CVPR, 2017, pp. 3892–3900.

J. He, S. Zhang, M. Yang, Y. Shan, and T. Huang, “Bi-directional cascade network for perceptual edge detection,” in CVPR, 2019.

Z. Yu and C. Bajaj, “A segmentation-free approach for skeletonization of gray-scale images via anisotropic vector diffusion,” in CVPR, IEEE, 2004.

Y. Jang and K.-S. Hong, “A pseudo-distance map for the segmentation-free skeletonization of gray-scale images,” in ICCV, IEEE, 2003.

M. Majer, “On the influence of scale selection on feature detection for T. Evgeniou and M. Pontil, “Regularized multi–task learning,” in R. Caruana, “Multitask learning,” in Machine learning, vol. 23, no. 12, pp. 5309–5322, 2014.

Z. Yu and C. Bajaj, “A segmentation-free approach for skeletonization of gray-scale images via anisotropic vector diffusion,” in CVPR, IEEE, 2004.

A. Sironi, V. Lepetit, and P. Fua, “Multiscale centerline detection by learning a scale-space distance transform,” in CVPR, 2014.

A. Levinstein, C. Sminchisescu, and S. Dickinson, “Multiscale symmetric part detection and grouping,” in IJCV, vol. 104, no. 2, pp. 117–134, 2013.

N. Widyanski, A. Moevus, and M. Mignotte, “Local symmetry detection in natural images using a particle filtering approach,” in IEEE TIP, vol. 23, no. 12, pp. 5309–5322, 2014.

Y. Wang, Y. Xu, S. Tsogkas, X. Bai, S. Dickinson, and K. Siddiqi, “Deepflux for skeletons in the wild,” in ICCV, Springer, 2012, pp. 41–54.

A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in NeurIPS, 2012.

Y. Wu and K. He, “Group normalization,” in ECCV, 2018, pp. 3–19.

Q. Yan, L. Xu, J. Shi, and J. Jia, “Hierarchical saliency detection,” in CVPR, 2013, pp. 1155–1162.

Y. Li, X. Hou, C. Koch, J. M. Rehg, and A. L. Yuille, “The secrets of salient object segmentation,” in CVPR, 2014, pp. 280–287.

C. Yang, L. Zhang, H. Lu, X. Ruan, and M.-H. Yang, “Saliency detection via graph-based manifold ranking,” in CVPR, 2013, pp. 3166–3173.

V. Movahedi and J. H. Elder, “Design and perceptual validation of performance measures for salient object segmentation,” in CVPR, 2010, pp. 49–56.

D.-P. Fan, M.-M. Cheng, Y. Liu, T. Li, and A. Borji, “Structure-measure: A New Way to Evaluate Foreground Maps,” in ICCV, 2017, pp. 4548–4557.

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.

F. Dollar and C. L. Zitnick, “Fast edge detection using structured forests,” IEEE TPAMI, vol. 37, no. 8, pp. 1558–1570, 2015.

J. Pont-Tuset, P. Arbelaez, J. T. Barron, F. Marques, and J. Malik, “Multiscale combinatorial grouping for image segmentation and object proposal generation,” IEEE TPAMI, vol. 39, no. 1, pp. 128–140, 2017.

R. Deng, J. Shi, S. Liu, H. Wang, and X. Liu, “Learning to predict crisp boundaries,” in ECCV, 2018, pp. 562–578.

C. Liu, W. Ke, F. Qin, and Q. Ye, “Linear span network for object skeleton detection,” in ECCV, 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 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 include computer graphics, computer vision, and image processing. He has published 60+ refereed research papers, with 15,000+ 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.

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 15,000+ 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.