A Highly Efficient Model to Study the Semantics of Salient Object Detection

Ming-Ming Cheng, Shang-Hua Gao, Ali Borji, Yong-Qiang Tan, Zheng Lin, Meng Wang

Abstract—CNN-based salient object detection (SOD) methods achieve impressive performance. However, the way semantic information is encoded in them and whether they are category-agnostic is less explored. One major obstacle in studying these questions is the fact that SOD models are built on top of the ImageNet pre-trained backbones which may cause information leakage and feature redundancy. To remedy this, here we first propose an extremely light-weight holistic model tied to the SOD task that can be freed from classification backbones and trained from scratch, and then employ it to study the semantics of SOD models. With the holistic network and representation redundancy reduction by a novel dynamic weight decay scheme, our model has only 100K parameters, $\sim 0.2\%$ of parameters of large models, and performs on par with SOTA on popular SOD benchmarks. Using CSNet, we find that a) SOD and classification methods use different mechanisms, b) SOD models are category insensitive, c) ImageNet pre-training is not necessary for SOD training, and d) SOD models require far fewer parameters than the classification models. The source code is publicly available at https://mmcheng.net/sod100k/.

Index Terms—salient object detection, efficient saliency prediction, semantics.

1 Introduction

Based on the observed human reaction time and singnal transmission time along biological pathway [11], [82], cognitive psychology studies suggest that human visual system (HVS) recruits pre-attentive, bottom-up attention mechanisms before recognizing the semantics of the scene [72]. Computer vision community has modeled these findings by using category-independent hand-crafted contrast features [2], [9] to build traditional salient object detection (SOD) models [3]. Many computer vision applications such as image retrieval [5], [8], [23], visual tracking [29], photographic composition [7], [21], image quality assessment [80], content-aware image processing [59], [92], and unsupervised semantic segmentation [18], utilize SOD models based on the hypothesis that salient objects are *generic and category-independent*.

While SOD methods based on convolutional neural networks (CNNs) have made significant progress, most of them focus on improving the state-of-the-art (SOTA) performance by learning fine local details and global features [75], [90], [95], [96], attention cues [4] as well as edge cues [14], [79], [98]. Existing CNN-based SOD models rely on ImageNet pre-trained backbone architectures [15], [25] with a considerable amount of parameters to extract features. However, ImageNet pre-training inevitably introduces category semantics into SOD models, causing a potential conflict with the conventional assumption that salient regions are category-independent [3], [37], [87]. This potential conflict raises new questions. How much of a role do category semantics play in the bottom-up SOD task? Is ImageNet pre-training inevitable and necessary for SOD training?

There are two principles in designing a SOD model that can be used to answer these questions. First and foremost, a SOD model should be possible to train without relying on ImageNet pre-training. Existing SOD models are built upon classification backbones that contain too many parameters, making them difficult to be trained from scratch. In oder to distinguish between thousand of different classification categories, even the light-weight classification backbone models ResNet-18 [25] and MobileNet v2 [34] contains 11M and 4.2M parameters (*vs.*100K parameters of our entire model). An important motivation behind this work is to verify if those category-oriented features and the corresponding parameters are indispensable for the SOD task. Second, the SOD task requires the model to have high feature resolution and strong multi-scale ability, that are non-essential for classification backbone [77], [95]. Existing works [14], [30], [78] relieve these problems by adding the SOD task related saliency heads to backbones, but inevitably introduce extra parameters.

To achieve the aforementioned requirements, we propose an extremely light-weight model that holistically consider the feature extractor and saliency head. We generalize the OctConv [6], namely gOctConv, with more flexibility and additional the self-adaptive property. A dynamic weight decay scheme is designed to enables self-adaptive learnable number of feature channels in gOctConv. This scheme not only helps analyze the semantic information in SOD models, but also allows $\sim 80\%$ parameter reduction with a negligible performance drop. Utilizing gOctConv, we propose a highly light-weight holistic **C**ross-**S**tage **C**ross-**S**cale network, namely CSNet. Benefiting from the holistically design and the dynamic weight decay, CSNet performs on par with SOTA but has only 100k parameters ($\sim 0.2\%$ parameters of SOTA models).

As a bonus to the extremely low number of parameters, our CSNet can be directly trained from scratch without ImageNet pre-training, providing a unique opportunity for answering questions

- **M.M. Cheng and S.H. Gao are joint first authors. M.M. Cheng is the corresponding author (cmm@nankai.edu.cn).*
- *M.M. Cheng, S.H. Gao, Y.Q Tan, Z Lin are with the TKLNDST, College of Computer Science, Nankai University, Tianjin 300350, China.*
- *A. Borji is with Primer Technologies Inc., San Francisco, USA (aliborji@gmail.com).*
- *M. Wang is with Hefei University of Technology.*
- *A preliminary version of this work has been presented in the ECCV 2020 [19].*

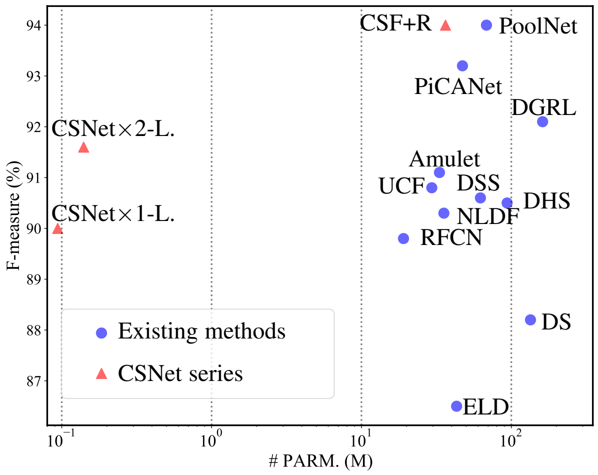

Fig. 1. Parameters vs. performance of SOD models.

regarding the semantics of CNN-based SOD models. We analyze the category sensitivity by transferring a SOD model to the classification task and test its performance on unseen categories. The experimental results reveal that SOD models are category insensitive and the detected salient objects are generic. Further, we observe that SOD models require far fewer parameters than classification models since the feature representation for distinguishing categories is not needed in SOD models. The category insensitive mechanism of our SOD model not only provides an opportunity to improve the efficiency of SOD models, but also mimic the category independent bottom-up human attention mechanism.

In summary, we make two major contributions in this paper:

- We thoroughly analyze the semantic information in CNN-based SOD models and experimentally verfiy that SOD models require negligible category information (*i.e.* category insensitive).
- By holistically redesigning feature extraction and SOD prediction, we abandon the wildly used pre-trained CNN backbones, which contains intensive parameters for unnecessary category information. Accompanying the sparsity introduced by dynamic weight decay, we significantly slim the model to $\sim 0.2\%$ parameters of SOTA with comparable performance.

Our focus in this work is on semantics of SOD whereas the conference version [19] was concerned mostly with building a light-wight SOD model. In Sec. 3, we introduce the dynamic weight decay scheme and the learnable channels for generalized OctConv. We then introduce the light-weight holistic CSNet designed for the analysis of SOD model in Sec. 4. With the CSNet, we analyze the semantics of the SOD model in Sec. 5. In Sec. 6, we do performance evaluation and ablations to show the efficiency of CSNet.

2 Related Works

2.1 Salient Object Detection

Traditional methods [9], [37], [73], [87], [102] mainly rely on hand-crafted features to detect salient objects. Early deep learning based methods [43], [50], [76] utilize CNNs to extract more informative features from image patches for improving the quality of saliency maps. Inspired by the fully convolutional networks (FCNs) [55], recent methods [13], [41], [51], [78], [93], [95] formulate salient object detection as a pixel-level prediction task and solve it in an end-to-end manner using FCN based models. These methods [30], [66], [77], [95], [96] capture both fine details and global features from different stages of the backbone network. Edge cues are introduced in [45], [57], [79], [98] to refine the boundary of saliency maps further. Other methods [83], [95], [99] also improve the saliency detection from the perspective of network optimization. Recently, Wei *et al.* [81] decomposes the original saliency map into a detail map to better learn edge features, and a body map to avoid the distraction from pixels near edges. Pan *et al.* [64] integrate the features from adjacent levels to solve the multi-scale issue in the SOD task. Multi-level gate units are used in [100] to balance the contribution from each encoder block and to suppress the activation of the features from non-salient regions. Despite the impressive performance, all of these CNN-based methods require powerful pre-trained ImageNet backbones as the feature extractor, which usually results in high computational cost. Moreover, none of them have studied the semantics behind the CNN-based SOTA SOD models and whether pre-training is indeed necessary.

2.2 Light-weight Models

Currently, most light-weight models that are initially designed for classification tasks utilize modules such as inverted block [33], [34], channel shuffling [58], [97], and SE attention module [33], [71] to improve network efficiency. Classification tasks [67] predict high-level semantic labels for an image, requiring more global information but fewer details. Thus, light-weight models [33], [34], [58], [97] designed for classification use aggressive downsampling strategies at earlier stages to save multiply-accumulate operations (MACC). These strategies makes them less useful for feature extraction since SOD task requires multi-scale information at both coarse and fine levels. Also, the SOD task focuses on determining the salient region while classification tasks predict category information. To improve saliency prediction performance under limited computing budget, the allocation of computational resources (*i.e.* resolution, channels) should be reconsidered.

2.3 Network Pruning

Many network pruning methods have been proposed to prune less important filters especially on the channel level [28], [44]. To prune filters, redundant filters can be estimated by norm criterion [26], [44], statistical information of the next layer [56], geometric median of weights [27], and reusing the scaling factor of the batch normalization layer [53]. Meta pruning *et al.* [54] utilizes generated weights to estimate the performance of the remaining filters. Most pruning approaches still rely on regularization tricks such as weight decay to introduce sparsity in filters. Our proposed dynamic weight decay stably introduces sparsity for assisting pruning algorithms in removing redundant filters, resulting in learnable channels for each scale in our proposed gOctConv.

3 gOctConv & Learnable Channels

To analyze the semantics of CNN-based SOD models without disturbed by the ImageNet pre-training, we need to build a light-weight SOD model that 1) can be trained from scratch without reliance on ImageNet pre-training; 2) is simple enough to avoid

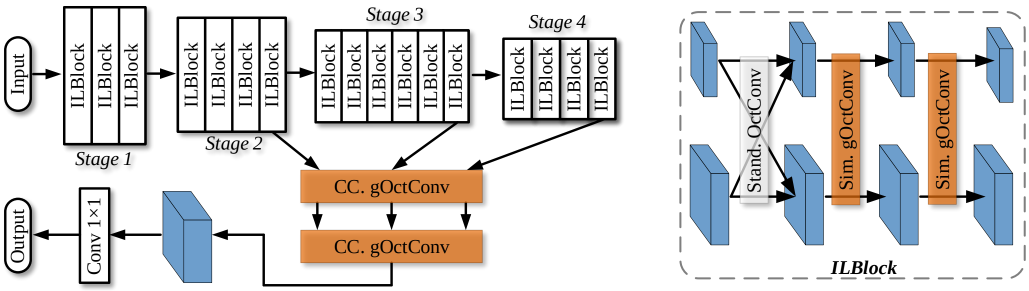

Fig. 2. Illustration of our salient object detection pipeline, which uses gOctConv to extract both within-stage and cross-stage multi-scale features in a highly efficient way. Sim. and CC. gOctConv denote the simplified and cross-stage instances of gOctConv, respectively.

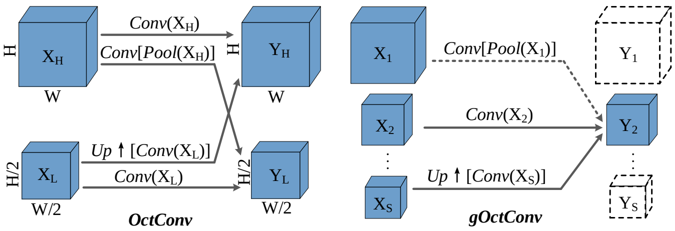

Fig. 3. While originally designed to be a replacement for the traditional convolution unit, the OctConv [6] takes two high/low-resolution inputs from the same stage with a fixed number of feature channels. Our gOctConv allows an arbitrary number of input resolutions from both within-stage and cross-stage conv features with a learnable number of channels.

potential confusion from complex modules; 3) have strong multi-scale representation ability for high SOD performance. Self-adaptive property, *i.e.* the self-adaptive computation allocation, is also required to better study the complexity and feature requirements of a SOD model. However, models constructed by native convolutional layers can hardly meet all of these requirements. Therefore, we need a new basic component to design the required SOD model.

We therefore propose a flexible self-adaptive convolutional layer, namely generalized Octave Convolution (gOctConv), to construct a light-weight SOD model with a simple yet effective structure. The proposed gOctConv is flexible enough to have multiple instances to build different parts of the SOD model with strong multi-scale representation abilities. Also, its self-adaptive property assists the analysis of SOD models.

3.1 Generalized OctConv

Originally designed to be a replacement for the traditional convolution unit, the vanilla OctConv [6] shown in Fig. 3 conducts the convolution operation across low and high scales within a layer. However, only two-scales within a stage are not enough to excavate multi-scale information required for the SOD task (see also Tab. 6). The number of channels for each scale in the vanilla OctConv is manually set, which requires a lot of effort to re-adjust for a saliency model. Therefore, we propose a generalized OctConv (gOctConv) that allows incorporating an arbitrary number of input scales from both within-stage and cross-stage conv features with self-adaptive learnable channels as shown in Fig. 3.

As a generalized version of the vanilla OctConv, gOctConv improves the vanilla OctConv for the SOD task in the following aspects. Firstly, instead of being a module with a fixed structure, the flexible gOctConv is a class that allows many instances under different SOD design requirements. For instance, the cross-scales feature interaction can be turned off to support large complexity flexibility. Arbitrary numbers of input and output scales are available to support a larger range of multi-scale representations. Except for within-stage features, the gOctConv can also process cross-stage features with arbitrary scales from the feature extractor. With the high flexible instances of gOctConv, we can design a highly efficient but very simple SOD model. Secondly, the gOctConv supports self-adaptive learnable channels for each scale. This property allows analyzing some properties of a SOD model such as model complexity and multi-scale feature requirements. The implementation details and complexity analysis of gOctConv is provided the in supplementary.

3.2 Learnable Channels

We propose to construct self-adaptive learnable channels for each scale in the gOctConv, by utilizing our proposed dynamic weight decay to assist channel pruning. Dynamic weight decay maintains a stable output feature distribution among channels while introducing sparsity, helping pruning algorithms to eliminate redundant channels at the expense of a negligible performance drop.

Introducing Sparsity with Dynamic Weight Decay: The commonly used regularization trick weight decay [39], [91] endows CNNs with better generalization performance. Mehta *et al.* [60] show that weight decay introduces sparsity into CNNs, which helps prune unimportant weights. Training with weight decay makes unimportant weights in CNN have values close to zero. Thus, weight decay has been widely used in pruning algorithms to introduce sparsity [27], [28], [44], [53], [54], [56]. The common implementation of weight decay is by adding the L2 regularization to the loss function, which can be written as follows:

$$\mathbf{L} = \mathbf{L_0} + \lambda \sum \frac{1}{2} \mathbf{w}_i^2, \tag{1}$$

where $\mathbf{L_0}$ is the loss for the specific task, $\mathbf{w}_i$ is the weight of the ith layer, and λ is the weight for weight decay. During back propagation, the weight $\mathbf{w}_i$ is updated as:

$$\mathbf{w}_i \leftarrow \mathbf{w}_i - \nabla f_i(\mathbf{w}_i) - \lambda \mathbf{w}_i, \tag{2}$$

where $\nabla f_i(\mathbf{w}_i)$ is the gradient to be updated, and $\lambda \mathbf{w}_i$ is the decay term, which is only associated with the weight itself. Applying a large decay term enhances sparsity, and meanwhile inevitably enlarges the diversity of weights among channels. Fig. 4 shows that diverse weights cause the unstable distribution of outputs among channels. Ruan *et al.* [12] reveal that channels with

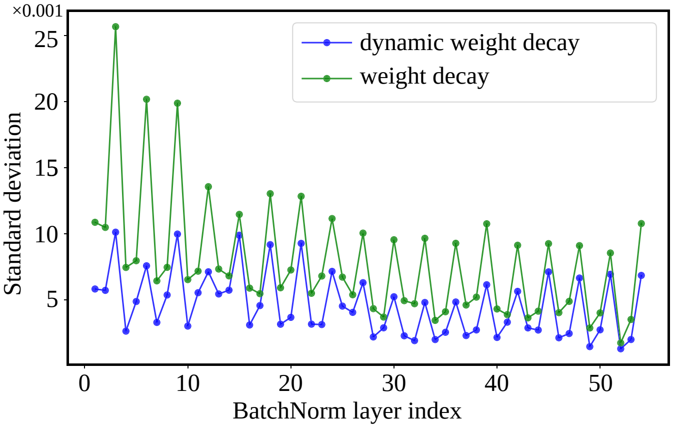

Fig. 4. The average standard deviation of outputs among channels after the BatchNorm and activation layer in models trained w/o dynamic weight decay.

diverse outputs are more likely to contain noise, leading to biased representation for subsequent filters. Attention mechanisms have been widely used to re-calibrate the diverse outputs with extra blocks and computational cost [12], [35]. We propose to relieve diverse outputs among channels with no extra cost during the inference. We argue that the diverse outputs are mainly caused by the indiscriminate suppression of decay terms to weights. Therefore, we propose to adjust the weight decay based on specific features of certain channels. Specifically, during back propagation, decay terms are dynamically changed according to features of certain channels. The weight update of the proposed dynamic weight decay is written as:

$$\mathbf{w}_i \leftarrow \mathbf{w}_i - \nabla f_i\left(\mathbf{w}_i\right) - \lambda_d \, \mathrm{M}\left(\mathbf{x}_i\right) \mathbf{w}_i, \tag{3}$$

where λ_d is the weight of dynamic weight decay, $\mathbf{x}_i$ denotes the features calculated by $\mathbf{w}_i$, and $\mathrm{M}\left(\mathbf{x}_i\right)$ is the feature metric, which can have multiple definitions depending on the task. In this paper, our goal is to stabilize the weight distribution among channels according to features. Thus, we simply use the global average pooling (GAP) [48] as the metric for a certain channel:

$$\mathrm{M}\left(\mathbf{x}_i\right) = \frac{1}{HW} \sum_{h=0}^{H} \sum_{w=0}^{W} \mathbf{x}_i(h, w), \tag{4}$$

where H and W are the height and width of the feature map $\mathbf{x}_i$. The dynamic weight decay with the GAP metric ensures that the weights producing large value features are suppressed, giving a compact and stable weight and output distribution as revealed in Fig. 4 and Fig. 5. The metric can also be defined as other forms to suit certain tasks, as we will study in our future work.

Self-adaptive Learnable channels: Now, we incorporate dynamic weight decay with pruning algorithms to remove redundant weights, to construct the self-adaptive learnable channels at each scale in gOctConvs. We follow [53] to use the weight of the BatchNorm layer as the indicator of the channel importance. The BatchNorm operation [36] is written as follows:

$$y = \frac{x - \mathrm{E}(x)}{\sqrt{\mathrm{Var}(x) + \epsilon}} \gamma + \beta, \tag{5}$$

where x and y are input and output features, $\mathrm{E}(x)$ and $\mathrm{Var}(x)$ are the mean and variance, respectively, and ϵ is a small factor in avoiding zero variance. γ and β are learned factors.We apply the

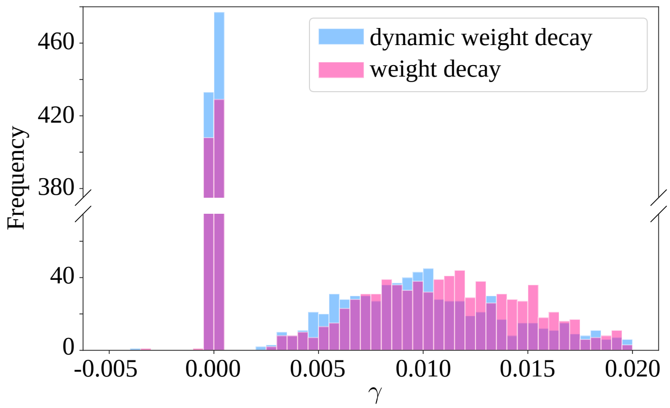

Fig. 5. Distribution of γ in Eqn. (5) for models trained w/o dynamic weight decay.

Algorithm 1 Learning Channels for gOctConv with Dynamic Weight Decay

Require: The initial CSNet in which channels for all scales in gOctConvs are set. Input images X and corresponding label Y.

for each iteration $i \in [1, MaxIteration]$ **do**
  Feed input X to the network to get the result $\hat{Y}$
  Compute $Loss = criterion(\hat{Y}, Y)$
  Compute metric for each channel using Eqn. (4)
  Backward with dynamic weight decay using Eqn. (3).
end for
Eliminate redundant channels to get the learnable channels for each scale in gOctConv.
Train for several iterations to finetune remaining weights.

dynamic weight decay to γ during training. The output features after the BatchNorm layer and the activation layer are used as the input to compute the metric in Eqn. (4). Fig. 5 reveals a clear gap between important and redundant weights, and unimportant weights are suppressed to nearly zero ($\mathbf{w}_i < 1e-20$). Thus, we can easily remove channels whose γ is less than a small threshold. The learnable channels of each resolution features in gOctConv are obtained. The algorithm of constructing learnable channels of gOctConvs is illustrated in Alg. 1.

4 Holistic Light-weight Model for Studying the Semantics of SOD model

4.1 Overview

While several CNN-based SOD models [30], [45], [57], [66], [77], [79], [83], [95], [95], [96], [98], [99] with impressive performance and efficiency have been proposed. However, the SOD community usually builds models on top of the ImageNet pre-trained backbones, which limits the design space and inherits a large number of parameters containing category-oriented representations. Even the light-weight classification backbone itself, *e.g.,* ResNet-18 and MobileNet v2, contain 11M and 4.2M parameters respectively. Thanks to the extremely low number of parameters (100K) and the holistic design, our CSNet can be directly trained from scratch without ImageNet pre-training. Such design frees the CSNet from unnecessary category-oriented information contained in ImageNet pre-trained models [62], [88].

To support the analysis of each component in the model, we use different instances of the proposed gOctConv to construct a simple yet effective SOD model. We holistically design the feature extractor and a cross-stage fusion part following the requirements of SOD task as shown in Fig. 2. It simultaneously processes features within multiple scales. The feature extractor is stacked with our proposed in-layer multi-scale block, namely ILBlocks. The cross-stage fusion part processes features from stages of the feature extractor to obtain a high-resolution output. The ILBlock and cross-stage fusion part, both composed of instances of gOctConv, enhance the within-stage and cross-stage multi-scale representation ability required by the SOD task. The straightforward structure of the CSNet avoids the potential influence of complex modules. Also, benefiting from the self-adaptive property of gOctConv, we can better study the complexity and feature requirements of the SOD model. Therefore, CSNet is a suitable tool for studying the semantics of SOD models.

4.2 In-layer Multi-scale Block

ILBlock enhances the multi-scale representation of features within a stage. gOctConvs are utilized to introduce multi-scale capacity within ILBlock. Borrowing the common definition from the classification models [33], [34], [68], [97], a highly light-weight SOD model should be at least $10\times$ smaller than existing SOD models. The vanilla OctConv requires about 60% MACC [6] to achieve similar performance as in the standard convolution, which is not enough to design a highly light-weight model. To save computational cost, integrating features with different scales in every layer is unnecessary. Therefore, we apply an instance of the gOctConv that eliminates the cross-scale operations while keeps within-scale operations, namely simplified gOctConv. In simplified gOctConv, each input channel corresponds to an output channel with the same resolution, and a depthwise operation within each scale is utilized to further save computational cost. The simplified instance of gOctConv only requires about $1/channel$ MACC compared to the vanilla OctConv. ILBlock is composed of a vanilla OctConv and two 3×3 simplified gOctConvs as shown in Fig. 2. The vanilla OctConv integrates features with two scales and simplified gOctConvs extract features within each scale. Multi-scale features within a block are separately processed and interacted alternately. Each gOctConv is followed by the BatchNorm [36] and PReLu [24].

4.3 Cross-stage Fusion

Common SOD methods retain a high output resolution by preserving high feature resolution at the high-level of the feature extractor, which inevitably increases the computational redundancy. Another solution is to construct complex multi-level aggregation modules to fuse high-level features with semantics and low-level features with details. The value of multi-level aggregation is widely recognized on many tasks, *e.g.,* edge detection [85], object detection [22], classification [70], and the SOD task [30]. However, these works utilize large backbone models. How to efficiently and concisely achieve cross-stage fusion for the SOD task remains challenging. In this works, we aims at designing a simple yet effective holistic model that strongly tied to the SOD task. We also need a simple yet effective multi-level aggregation strategy to analyze the semantics of SOD models. To this end, we simply use the cross-stage instance of gOctConvs to fuse multi-scale features from stages of the feature extractor and generate the high-resolution output. A cross-stage gOctConv 1×1 takes features with different scales from the last conv of each stage as input and conducts a cross-stage convolution to output features with different scales. To extract multi-scale features at a granular level, each scale of features is processed by a group of parallel convolutions with different dilation rates. Features are then sent to another cross-stage gOctConv 1×1 to generate features with the highest resolution. Another standard Conv 1×1 layer outputs the prediction result of the saliency map.

4.4 Implementation details of CSNet

CSNet consists of a feature extractor and a cross-stage fusion part. As shown in Tab. 2, the feature extractor is stacked with ILBlocks, and is split into 4 stages according to the resolution of feature maps, where each stage has 3, 4, 6, and 4 ILBlocks, respectively. Initially, we set the number of channels of the first stage to 20, and double the channels of ILBlocks as the resolution decreases, except for the last two stages that have the same number of channels. The channel for each gOctConv can be expanded to enlarge the model capability. Models with channels expanded k times are denoted by CSNet-$\times k$. Learnable channels of OctConvs are then obtained with the self-adaptive channel learning scheme. Given an input image with the shape (H, W), the first gOctConv in the first ILBlock takes in the image and outputs features with two resolutions (H, W) and $(H/2, W/2)$. The features with two scales are processed in parallel by the feature extractor. We denote the term "split-ratio" as the ratio of the number of channels among different feature scales in gOctConv. The split-ratio in ILBlocks can be adjusted to construct models with different MACC, denoted by C_H/C_L. Unless otherwise stated, the channels for different scales in ILBlocks are set evenly. For cross-stage fusion, only the output feature of each stage is used. The last ILBlock of each stage merges two streams of different scales to the high-resolution stream. The cross-stage fusion part processes features from stages of the feature extractor to obtain a high-resolution output. As a trade-off between efficiency and performance, features from the last three stages are used. Learnable channels of gOctConvs in this part are also obtained. The detailed configurations of the cross-stage fusion part are shown in Tab. 1.

5 Analysis of SOD Models

In this section, we are going to answer these questions about the semantics of the CNN-based SOD model with our proposed CSNet: 1) Are SOD models sensitive to category information? 2) Which part of the SOD model is most responsible for locating the salient regions? 3) Can the SOD model detect salient regions over unseen categories? 4) Do SOD models have the same complexity compared to classification models? 5) What are the feature requirements of SOD task from the feature extractor? 6) What role does ImageNet pre-training play in the SOD model training?

5.1 Category Sensitivity

Before the emergence of CNNs, SOD methods were regarded as being category agnostic [3], [37], [87]. Researchers have used these category agnostic saliency detection models to support downstream tasks [5], [21], [23], [31], [59], [80], [92], for tasks such as weakly supervised segmentation. With the widespread popularity of CNNs, the community has proposed several new SOD models utilizing powerful ImageNet pre-trained CNN backbones to extract features. Are these models still category insensitive and can they be used as generic features? How much

TABLE 1
Architecture for the cross-stage fusion part using four stages in CSNet×1.

name	output feature size	config
gOctConv	[224×224×20, 112×112×40, 56×56×80, 28×28×80]	gOctConv, kernel size 1×1, dilation 1
Parallel DilatedConvs	[224×224× (1+1+1+1+1), 112×112×(2+2+2+2+5), 56×56×(5+5+5+5+6),28×28×(5+5+5+5+6)]	[DilatedConvs, kernel size 3×3 dilations [1, 2, 4, 8, 16]]× scales
gOctConv	224×224×70	gOctConv, kernel size 1×1, dilation 1
StandardConv	224×224×1	StandardConv, kernel size 1×1, dilation 1

TABLE 2
Architecture for the feature extractor in CSNet×1.

stage	output feature size	config [op, kernel size, stride]
stage1	[224×224×10, 112×112×10]	[OctConv 3×3, 1; gOctConv 3×3, 1; gOctConv 3×3, 1] ×1
		[OctConv 1×1, 1; gOctConv 3×3, 1; gOctConv 3×3, 1] ×2
stage2	[112×112×20, 56×56×20]	[OctConv 3×3, 2; gOctConv 3×3, 1; gOctConv 3×3, 1] ×1
		[OctConv 1×1, 1; gOctConv 3×3, 1; gOctConv 3×3, 1] ×3
stage3	[56×56×40, 28×28×40]	[OctConv 3×3, 2; gOctConv 3×3, 1; gOctConv 3×3, 1] ×1
		[OctConv 1×1, 1; gOctConv 3×3, 1; gOctConv 3×3, 1] ×5
stage4	[28×28×40, 14×14×40]	[OctConv 3×3, 2; gOctConv 3×3, 1; gOctConv 3×3, 1] ×1
		[OctConv 1×1, 1; gOctConv 3×3, 1; gOctConv 3×3, 1] ×3

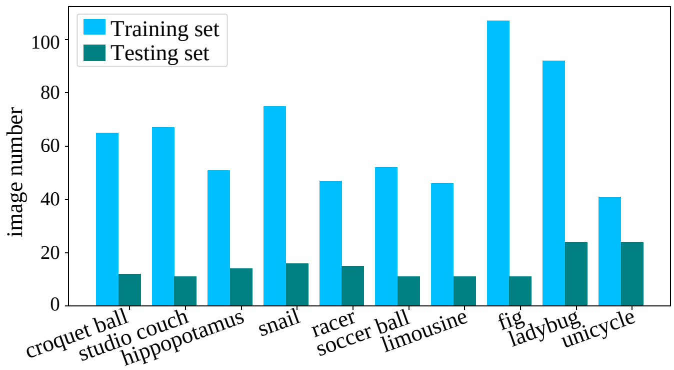

Fig. 6. Distribution of images in the classification dataset.

role category information plays in CNN based saliency models? Can we use saliency as general knowledge to reduce the domain-specific data annotation in tasks like weakly supervised semantic segmentation? These are very important questions but have been less explored. To study the category sensitivity of the SOD model, we analyze the CSNet trained with the same data but over the SOD and classification tasks.

5.1.1 Data preparation

Data distribution plays a vital role in representation learning. To remove the influence of different data distribution from different datasets, we train the SOD model and the classification model over the same set of images but use salient object mask labels and category labels as supervision for each task. Since the image-level category label is more accessible than pixel-level annotations for SOD, we annotate the existing SOD dataset with category labels. Specifically, we utilize the commonly used datasets DUTS-TR [74], DUTS-TE [74] and ECSSD [86] as the source dataset, and assign category labels of the ImageNet to images. These SOD datasets have imbalanced category distribution. Thus, they can not be directly used for the analysis of category sensitivity. Analyzing models under the classification metric and SOD metric requires different data distributions. We therefore choose two sub-datasets for the evaluation of classification and SOD tasks, respectively.

Data for classification: The DUTS-TR dataset is used as the training dataset. As the DUTS-TE dataset has a highly similar category distribution with the DUTS-TR dataset, we use the DUTS-TE dataset to evaluate the classification task. A classifier trained on imbalanced category data may overfit to some categories, making the classification metric meaningless. To harness this, we choose 10 categories that have a balanced number of images in both DUTS-TR and DUTS-TE datasets. The training set and testing set contain 644 and 150 images, respectively. The distribution of the classification dataset is illustrated in Fig. 6.

Data for SOD: Eliminating a certain small category among many categories will have almost no impact on the original SOD dataset. To reinforce the impact of the category on the SOD dataset, we organize images into major categories according to the WordTree of ImageNet [67], and select 12 merged categories with a considerable number of images. Most SOD experimental results presented in this paper are reported on the ECSSD dataset [86]. Thus, we use the ECSSD dataset for the evaluation of SOD tasks. The distribution of images from the selected categories in the SOD dataset is illustrated in Fig. 7. We show in Fig. 8 that even with a biased category distribution, SOD models can still achieve reasonable performance.

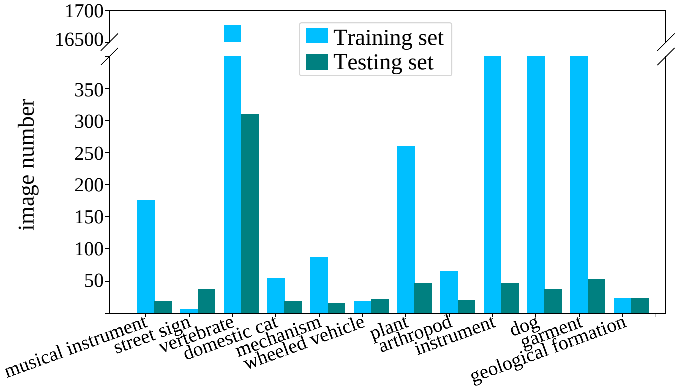

Fig. 7. Distribution of images of selected categories in the SOD dataset.

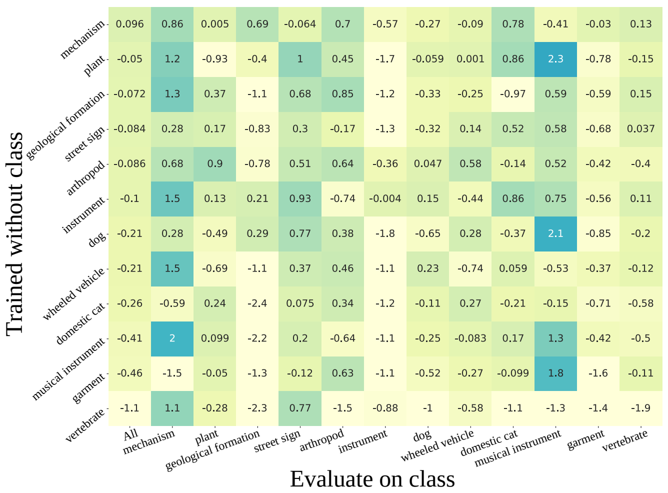

Fig. 8. The relative performance changes of F-measure after removing a certain category in training SOD models. Each row shows the performance change on images with all categories of a model trained without a certain category. Removing a category during training does not clearly influence the test performance of that category, which proves that the SOD model is not sensitive to the category information.

5.1.2 Transfer Learning from SOD to Classification

Settings: The basic idea of transfer learning is to pre-train the model on the source task, and finetune the pre-trained model on the target task to acquire performance gain or ease the convergence [25].

Transfer learning from ImageNet pretraining to many downstream tasks, *e.g.,* depth estimation, crowd counting, and bounding box regression, has been proved very effective. And transfer learning between the source and target tasks under similar semantic requirements, e.g., both tasks are category-related, is more effective than the transfer learning of two tasks under different semantics. Classification task is a category sensitive task, and the classification accuracy can be used to measure the category sensitivity of models. We have verified this hypothesis by conducting experiments of transferring the semantic segmentation model to the classification model. To study the category sensitivity of SOD models, we first pre-train the model on the SOD task and finetune it on the classification task. By fixing different parts of the SOD pre-trained model and finetuning the remaining parts, we can pinpoint which parts of the SOD model are more responsible for the classification and which parts are more specific to SOD.

CSNet is specifically designed for the SOD task, we modify the cross-stage fusion part to make it suitable for classification. The cross-stage fusion part takes in features from different stages and outputs features with the lowest resolution, as low-resolution features are more suitable for classification [25]. The last Conv 1×1 layer in CSNet is replaced with a global average pooling layer and a fully connected (FC) layer. By doing so, except for the last prediction layer, the remaining parts of CSNet trained for SOD and classification can share parameters. We use the top-1 classification accuracy as the metric for analyzing the category sensitivity of models. We conduct two groups of experiments as follows: (1) *Cls-scratch* is denoted as the model trained with only category labels from scratch for the classification task. (2) *Finetune-SOD* is denoted as the model that is pre-trained with SOD annotations on the SOD task and then finetuned on the classification task.

If *Finetune-SOD* model achieves significantly worse results than the *Cls-scratch* model, we can conclude that the SOD model is not sensitive to the category information. During finetuning, parts of the *Finetune-SOD* model are finetuned for the classification task and other parts are fixed with SOD pre-trained weights. By doing so, we can find out which parts of the SOD model are more related to the SOD task and which parts are more general to both classification and SOD tasks. For example, if finetuning stage 1 of the SOD model achieves very limited performance gain than only finetuning the FC layer, it indicates that features in stage 1 are more general instead of very closely related to the SOD task. Stages 1 to 4 and the cross-stage fusion parts are all studied as shown in Tab. 3.

With the transfer learning from SOD to classification, we can answer questions: 1) Whether SOD model is sensitive to category information. 2) The parts of SOD model that are more task-specific and parts that are more general to both classification and SOD tasks.

Experimental results: Tab. 3 shows the top-1 classification accuracy of models transferred from the SOD task to the classification task. The classification model trained from scratch achieves the top-1 acc. of 61.1%, while the SOD model with only the FC layer finetuned achieves the top-1 acc. of 18.1%. The result means that the model trained for SOD requires almost no category information to determine the salient region. To pinpoint which part of the SOD model is more task-specific or general to classification task, we finetune a part of the SOD model to check the relative classification performance gain. Finetuning the cross-stage fusion part achieves a considerable performance gain of 30.2%, indicating that the feature difference of this part between classification and SOD task is much larger than the other parts. Finetuning from stage1 to stage4 achieves increasingly higher performance gain, showing that high-level features are more task-specific while low-level features are general for both tasks.

To give a more direct evidence of the category sensitivity of SOD models, we give the class activation maps [101] (CAM) comparison between the SOD model finetuned on all stages (model with 58.4 acc. on Tab. 3) and the FC layer (model with 18.1 acc. on Tab. 3) for the classification task as shown in Tab. 9. CAM of the model finetuning on all stages focus on objects required by the classification, while CAM of the model finetuning on only the FC layer has no specific category-related focus. This comparison directly proves that the SOD model abandons the category-oriented features as its CAM cannot locate the category-related regions.

TABLE 3
The top-1 acc. of classification task using models transferred from the SOD task. s1 to s4 and fuse refer to stage1 to stage4 and the fusion part as shown in Fig. 2, respectively. ✓ indicates that parameters in a stage are finetuned.

Setups	s1	s2	s3	s4	fuse	top1 acc.	gain
Finetune-SOD						18.1	-
	✓					21.5	3.4
		✓				30.9	12.8
			✓			32.9	14.8
				✓		36.2	18.1
					✓	48.3	30.2
	✓	✓	✓	✓	✓	58.4	40.3
Cls-scratch	✓	✓	✓	✓	✓	61.1	43.0

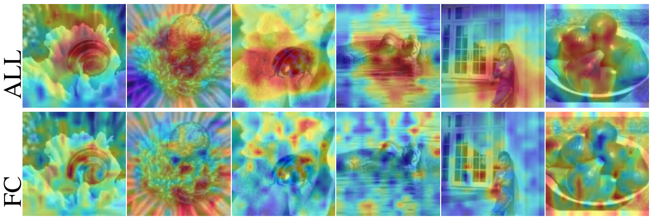

Fig. 9. Class activation maps comparison between the SOD model finetuned on all stages (ALL) and the FC layer (FC) for the classification task. CAM of the model finetuning on all stages focus on objects required by the classification, while CAM of the model finetuning on only the FC layer has no specific category-related focus.

5.1.3 SOD over Unseen Categories

Settings: We now study the category sensitivity of SOD models from the perspective of SOD metric. Generalizing to unseen objects or categories is one of the key features of SOD models, because this feature provides the basis of many down-stream vision tasks such as image retrieval [23], visual tracking [29], photographic composition [21], and image quality assessment [80]. We propose to test the performance of SOD models on images with unseen categories to verify the category sensitivity of SOD models. Given a SOD model trained without a certain category, if the model can still detect the salient object on the image with that category, it can be regarded as not sensitive to the category information. Specifically, based on the data for SOD, we have a series of models where each of them is trained by the data with images of one category removed. Each model is evaluated on images from all categories, respectively. We compare the relative performance changes between those models and the baseline model trained with images from all categories. Removing training images inevitably causes a performance drop. If the largest performance drop does not occur on the category that not exists on the training set, the SOD model can be regarded as category insensitivity.

Experimental results: We now test SOD models on images from unseen categories. Fig. 8 shows the relative performance changes of F-measure after removing a certain category for training. All results are the average of three runs. Removing a category during training does not significantly affect the test performance on that category, which indicates that SOD model is not sensitive to category information. For instance, removing the category 'vertebrate' during training causes the largest test performance drop of category 'geological formation', instead of that category itself. As shown in Fig. 7, the number of images from each category in the SOD dataset is not evenly distributed. Reducing the number of available training images has a significant impact on performance. For example, removing the category 'vertebrate' causes the largest performance drop in the ECSSD dataset, since this category has the largest number of training images. In contrast, removing categories such as 'street sign' and 'plant' have a very limited impact on the performance, since they have a relatively small number of training images. Therefore, we assume that instead of requiring category information, the SOD model needs to be trained with more images with various scenes to improve the performance.

TABLE 4
Complexity of different tasks based on the CSNet×1.5. SOD model is less complex compared to the classification model.

Setups	Full	SOD model	CLS model
Parms.	455K	167K	296K
MACC.	1.17G	0.70G	0.85G

5.2 Model Complexity

Settings: The complexity of models for different tasks is not necessarily the same. Current CNN-based SOD models are mostly built on top of the classification backbone models such as VGGNet [69] and ResNet [25]. Effective as these backbone models are on the classification, their large model complexity may not be necessary for the SOD task. Unlike the classification task that needs to learn category related features, the SOD task has almost no requirement for category information. We propose to analyze the complexity of models for the SOD task and the classification task from the perspective of model compression. As our proposed gOctConv is capable of eliminating redundant parameters with the help of the effective dynamic weight decay scheme, we can have a clear insight regarding the complexity of models for each task.

Experimental results: We use the dataset for classification metric in Sec. 5.1.2 as the training set. Since the training set contains only 644 images, a standard configuration of 300 epochs training is not enough for the convergence of a compact model. We therefore increase the number of the training epochs to 3000. Tab. 4 shows the model complexity for different tasks based on the CSNet×1.5. The SOD model requires 36% parameters of the original model, while the classification model requires 65% of parameters, indicating that SOD models require fewer parameters as they need almost no category information. With this observation, we believe a more compact SOD model can be designed specifically for the SOD task.

5.3 Feature Requirements from the Extractor

Using features from different stages of the extractor: As a pixel-level prediction task, SOD should generate a high resolution output map. Therefore, it is a common choice for existing SOD models [49], [51], [94] to utilize features from different stages of the feature extractor. Utilizing more features from earlier stages results in higher resolution prediction maps, while introducing more computational complexity. Here we study whether using more features from earlier stages can improve the SOD performance. As shown in Fig. 2, the cross-stage fusion part takes in

TABLE 5
Using features from different stages of the extractor in CSNet×2-L as the input of the cross-stage fusion part.

Stages	4	3 to 4	2 to 4	1 to 4
MACC	0.58G	0.66G	0.72G	1.29G
Parms.	134k	139k	141k	177k
F_β	90.0	91.0	91.6	91.8

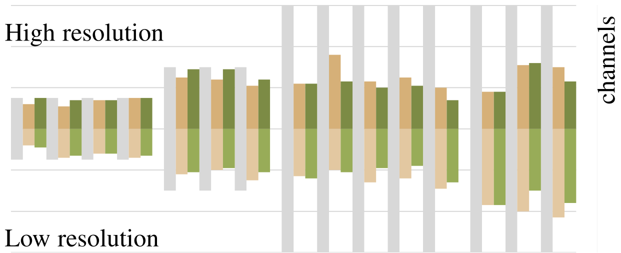

Fig. 10. Visualization of the number of channels of CSNet feature extractor. Gray is CSNet with the fixed channel, Yellow and Green are the CSNet-L trained with standard/dynamic weight decay, respectively. The horizontal axis indicates ILBlocks of the feature extractor starting from the early stage.

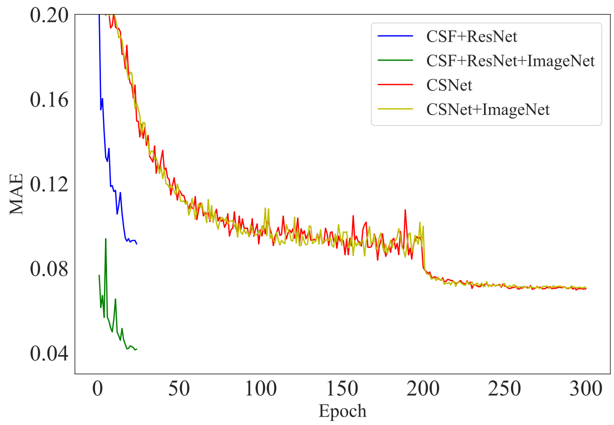

Fig. 11. The test MAE of models w/o ImageNet pre-training.

features from different stages of the extractor. We now perform the ablation study using features from different numbers of stages. To minimize the impact of the initial number of channels, we use the CSNet×2-L model with learned channels in gOctConvs. As shown in Tab. 5, using more stages from the feature extractor results in better performance and causes larger model complexity. CSNet×2-L using stages 1 to 4 achieves a 0.2% gain (91.8 vs 91.6) with 37k (177k vs 141k) more parameters than CSNet×2-L using stages 2 to 4. Therefore, using more features from earlier stages does benefit the quality of SOD. In our work, as a trade-off between efficiency and performance, we choose to use the last three stages as the input of the cross-stage fusion part if not otherwise stated.

Visualization of feature scale requirement: Since the feature extractor of our CSNet is composed of gOctConvs, we can study the feature scale requirement of the feature extractor with the self-adaptive channel learning property of gOctConvs. We visualize the learned number of channels of gOctConvs in Fig. 10. It can be seen that as the network goes deeper, the feature extractor shows a trend of utilizing more low-resolution features. Within the same stage, high-resolution features are urged in the middle of the stage. Also, the model trained with dynamic weight decay has a more stable number of channel variation among different layers. Deeper layers contain more redundant channels compared with shallower ones.

5.4 ImageNet Pre-training

ImageNet pretraining has been proved to be very effective for many down-stream tasks, *e.g.,* , depth estimation, crowd counting, and bounding box regression. Utilizing ImageNet pretrained feature extractor has been a default configuration in CNN-based SOD models due to its effectiveness. We compare the SOD model convergence speed with/without ImageNet pre-training on both light-weight and large models, shown in Fig. 11. For large model CSF+ResNet, the ImageNet pre-training helps the model converge in few epochs. As shown in Sec. 5.1.2, the early stage of the model contains more general low-level features. Large SOTA SOD models need ImageNet pre-training because the universal low-level features in the early stage of the pre-trained ImageNet models may help the convergence of large models that contain a lot of trainable parameters. For light-weight model CSNet, the convergence speed between the model with ImageNet pre-training and model trained from scratch shows almost no difference. Light-weight models are too small to benefit from the convergence acceleration of ImageNet pre-training. We pre-train the extractor of the CSNet on ImageNet to see if ImageNet pre-training can further benefit the performance of the SOD task. As shown in Tab. 7, the ImageNet pre-trained CSNet×1.5-L has similar performance compared with the model trained from scratch. Therefore, the effect of ImageNet pre-training is limited for the light-weight SOD models. ImageNet pre-training, however, still benefits large model convergence.

6 Performance Analysis and Ablation

6.1 Implementation

Training: Our method is implemented in PyTorch. We train the light-weight models using the Adam optimizer [38] with a batch-size of 24 for 300 epochs from scratch. Even with no ImageNet pre-training, the proposed CSNet still performs on par with large models based on pre-trained backbones [25], [69]. The learning rate is initially set to 1e-4, and is divided by 10 at the epochs of 200, and 250. We eliminate redundant weights and finetune the model for the last 20 epochs to compress models and get gOctConvs with the learnable channels of different resolutions. We only utilize random flip and crop for data augmentation. The weight decay of BatchNorms following gOctConvs is replaced with our proposed dynamic weight decay with the default weight of 3, while the weight decay for other weights is set to 5e-3 by default. For large models based on the pre-trained backbones, we train our models following the implementation of [49].

Datasets: While MSRA 10K [9], MSRA-B [52], DUT-O [87], and HKU-IS [42] datasets are used by earlier methods [20], [43], [46] for training salient object detectors, these datasets are either too small or lack diversity. We follow common settings of recent methods [49], [51], [77], [78], [96], [98] to train our models using the DUTS-TR [74] dataset, and evaluate the performance on several commonly used datasets, including ECSSD [86], PASCAL-S [47], DUT-O [87], HKU-IS [42], SOD [63], and DUTS-TE [74]. On ablation studies, the performance on the ECSSD dataset is reported

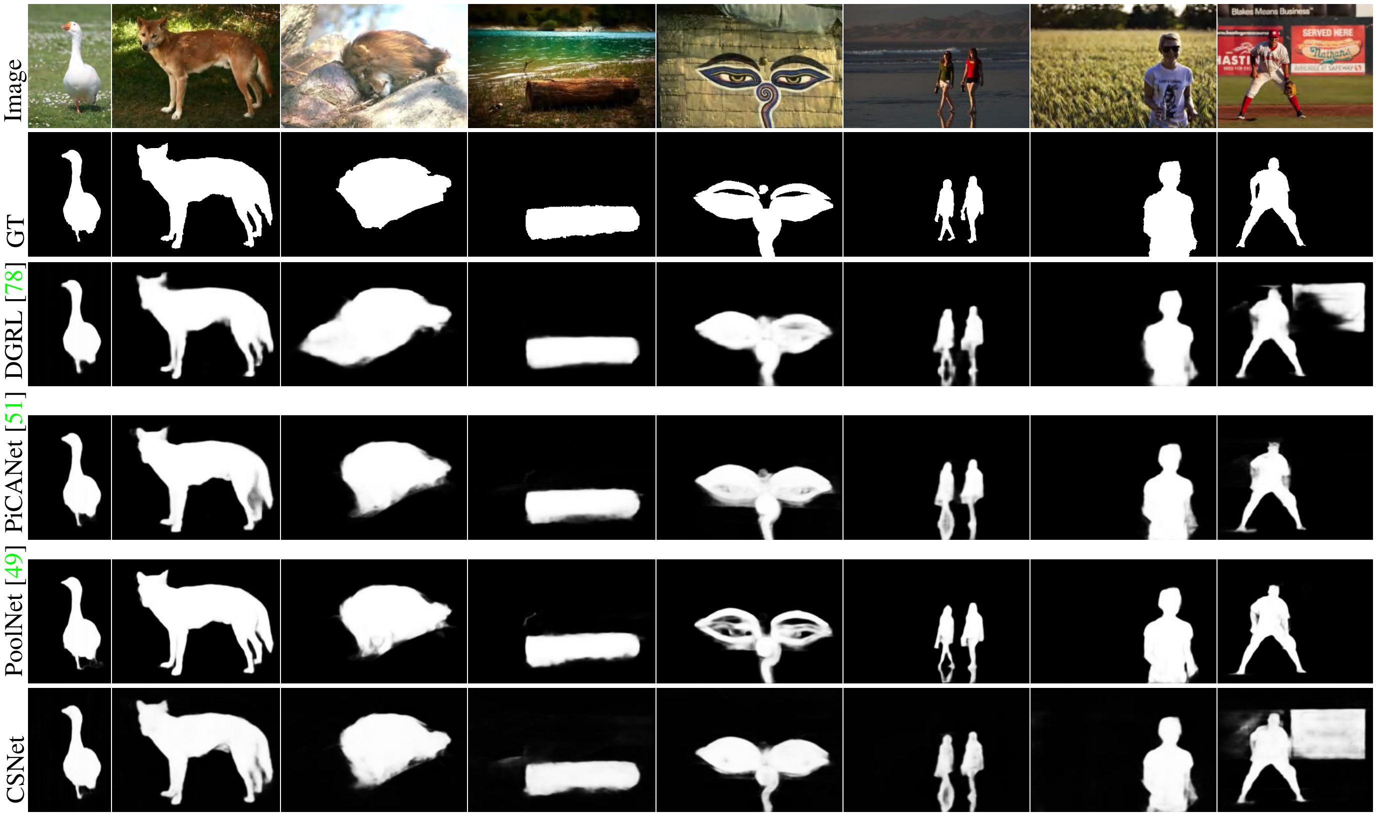

Fig. 12. Visual qualitative comparison between our proposed CSNet and existing SOTA models. The last column shows a failure case of CSNet, which we assume is caused by the limited representation ability of the extremely small model parameters.

if not mentioned otherwise. For the analysis of the semantics of SOD models, we use our two proposed datasets as described in Sec. 5.1.1.

Evaluation metrics: The commonly used evaluation metrics maximum F-measure (F_β) [1] and MAE (M) [10] are used for evaluation. MACC of light-weight models is computed with an image size of 224 $\times$ 224.

6.2 Performance Analysis

In this section, we first evaluate the performance of our proposed light-weight model CSNet with fixed channels. Then, the performance of CSNet with learnable channels using dynamic weight decay is measured. We show that ImageNet pre-training is not inevitable for CNN-based SOD models. Fig. 12 shows the qualitative results of salient object detection using our proposed light-weight CSNet. Also, we transfer the proposed cross-stage fusion part to commonly used large backbones [25] to verify the cross-stage feature extraction ability.

Performance of CSNet with fixed channels in gOctConv: The extractor model is only composed of ILBlocks. As shown in Tab. 6, when replacing the gOctConvs in ILBlocks with Vanilla OctConvs, the extractor has $\times$8 and $\times$7 of original size in terms of parameters and MACC, while the performance gain is very limited. The large model complexity gap shows the efficiency of the simplified instance of gOctConv in the ILBlock. Tab. 6 shows feature extraction models with different split-ratios of high/low-resolution features. Extractors achieve a low complexity thanks to the simplified instance of gOctConvs. Benefiting from the within-stage multi-scale representation and the low-resolution features in ILBlock, the extractor-3/1 achieves a performance gain of 0.4% in F-measure with 80% MACC over the extractor-1/0. The gOctConvs in the cross-stage fusion part enhance the cross-stage multi-scale ability of the network while maintaining the high output resolution by utilizing features from different stages. As shown in Tab. 6, the CSNet-5/5 surpasses the extractor-3/1 by 1.4% in F-measure with fewer MACC. Even in the extreme case, the CSNet-0/1 with only low-resolution features in extractor performs on par with the extractor-1/0 that has all high-resolution features, while only requires 44% MACC of the extractor-1/0. However, manually tuning the overall split-ratio of feature channels of different resolutions may achieve a sub-optimal balance between performance and computational cost. To further verify the effectiveness of the cross-stage fusion (CSF) part on large models, we add this part into the commonly used backbone network ResNet [25] and Res2Net [15]. Tab. 7 shows that the ResNet+CSF achieves similar performance to the ResNet+PoolNet with 53% parameters and 21% MACC. Unlike other models (*e.g.,* PoolNet) that eliminate downsampling operations to maintain a high feature resolution at high-levels of the backbone, the gOctConvs obtain both high and low resolution features across different stages of the backbone, achieving a high-resolution output while saving a large amount of computational cost.

Performance of CSNet with learnable channels in gOctConv: We further train the model with our proposed dynamic weight decay and obtain the learnable channels in gOctConv as described in Alg. 1. The obtained models are named CSNet-L. Tab. 11 shows that our proposed dynamic weight decay assisted pruning scheme can compress the model up to 18% of the original model size

TABLE 6
Performance of CSNet with the fixed split-ratio of channels in gOctConvs, and CSNet with learnable channels. CSNet: model with the fixed split-ratio in gOctConvs. Extractor: the network composed of only ILBlocks. Vanilla: the Extractor made of only vanilla OctConvs. CSNet-L: the model with learnable channels using Alg. 1.

Method		PARM.	MACC	$F_\beta \uparrow$	$M \downarrow$
Vanilla	5/5	1457K	3.31G	88.4	0.088
Extractor	1/0	180K	0.80G	88.2	0.088
	3/1	180K	0.64G	**88.6**	**0.085**
	5/5	180K	0.45G	88.1	0.086
	1/3	180K	0.30G	87.4	0.090
	0/1	180K	0.20G	86.4	0.095
CSNet	1/0	211K	0.91G	90.0	**0.076**
	3/1	211K	0.78G	89.9	0.077
	5/5	211K	0.61G	**90.1**	0.077
	1/3	211K	0.47G	89.2	0.082
	0/1	211K	0.35G	88.2	0.089
CSNet-L	×2	141K	0.72G	**91.6**	**0.066**
	×1	94K	0.43G	90.0	0.075

with a negligible performance drop. Compared with manually tuned split-ratio of feature resolution, the learnable channels of gOctConvs obtained by model compression achieves much better efficiency. As shown in Tab. 6, the compressed CSNet×2-L outperforms the CSNet-5/5 by 1.6% with fewer parameters and comparable MACC. The CSNet×1-L performs on par with CSNet-5/5 with about 45% parameters and about 70% MACC. Tab. 7 shows that CSNet-L series achieves comparable performance compared with some models with extensive parameters such as SRM [77], and Amulet [95] with $\sim 0.2\%$ parameters. Note that our light-weight models are trained from scratch while those large models are pre-trained with ImageNet. The performance gap between the proposed light-weight model and the SOTA models with extensive parameters and MACC is only $\sim 2\%$. Utilizing new techniques, *e.g.*, representative batch normalization [16] and receptive fields searching [17], will further close the gap with large models.

Comparison with light-weight models: To the best of our knowledge, we are the first to design an extremely light-weight model for the SOD task. For a more exhaustive analysis, we adopt several SOTA light-weight models, designed for other tasks such as classification and semantic segmentation, for salient object detection. All models share the same training configuration as in our training strategy. When transferring classification models to the SOD task, the FC layer is replaced with the Conv 1×1 layer to output saliency maps. For segmentation models, the number of channel of the output layer is changed from the number of classes to 1. Tab. 7 shows that our proposed models have considerable improvements compared with the light-weight models.

Run-Time: CSNet is designed to be light-weight and highly efficient on the SOD task. We compare the run-time of our proposed CSNet with existing models from Tab. 7 as shown in Tab. 8. The run-time is tested on a single core of i7-8700K CPU using 224 $\times$ 224 images. Our proposed CSNet is x10 faster compared with large-weight models. With similar speed, CSNet achieves up to 6% gain in F-measure compared with the models designed for other tasks. However, there is still a gap between MACC and run-time, as current deep learning frameworks are not optimized for vanilla and our proposed gOctConvs yet.

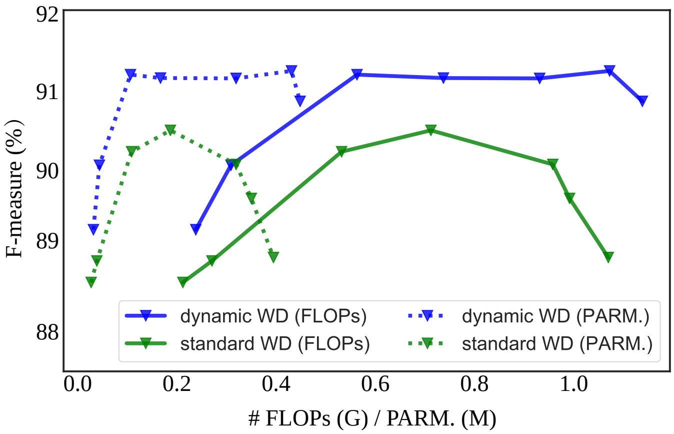

Fig. 13. Performance and complexity of our compressed model using dynamic/standard weight decay under different λ as shown in Eqn. (1). Applying different degrees of weight decay results in a trade-off between model performance and sparsity.

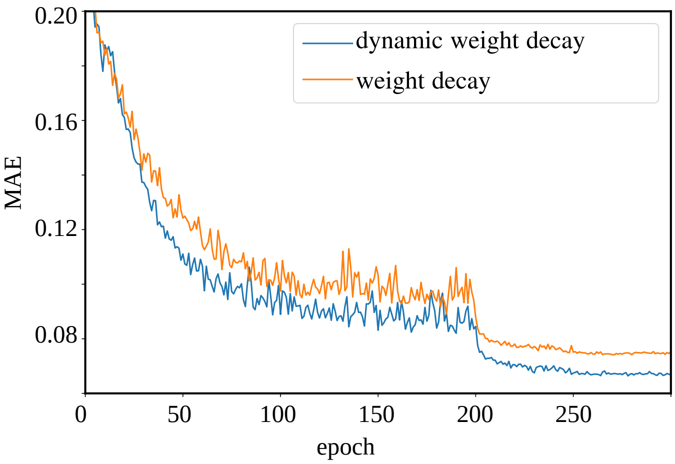

Fig. 14. The test MAE of models w/o dynamic weight decay.

6.3 Ablations

Dynamic weight decay: In this section, we assess the effectiveness of our proposed dynamic weight decay. We apply different degrees of weights to standard weight decay to balance model performance and sparsity, while keeping the weights for dynamic weight decay unchanged. We plug in our proposed dynamic weight decay into the weights of the BatchNorm layers while using the standard weight decay on remaining weights for a fair comparison. Fig. 13 shows the performance and complexity of the compressed model using dynamic/standard weight decay under different λ in Eqn. (1). The λ_d in Eqn. (3) for dynamic weight decay on BatchNorm is set to 3 by default. Models trained with dynamic weight decay have better performance under the same complexity. Also, the performance of dynamic weight decay based models is less sensitive to the model complexity. We eliminate redundant channels according to the absolute value of γ in Eqn. (5) as described in Sec. 3.2. Fig. 5 shows the distribution of γ for models trained with/without dynamic weight decay. By suppressing weights according to features, dynamic weight decay enforces the model with more sparsity. Fig. 4 reveals the average standard deviation of outputs among channels after the BatchNorm and activation layer of models trained with/without

TABLE 7
Performance and complexity comparison with state-of-the-art SOD methods. +R and +R2 denotes using the ImageNet pre-trained ResNet50 [25] and Res2Net50 [15] backbone. Unlike previous methods that require the ImageNet pre-training, our light-weight CSNet is trained from scratch.

Model	Complexity		ECSSD		PASCAL-S		DUT-O		HKU-IS		SOD		DUTS-TE	
	#PARM.	MACC	F_β	M	F_β	M	F_β	M	F_β	M	F_β	M	F_β	M
ELD [20]$_{\mathrm{CVPR'16}}$	43.15M	17.63G	.865	.981	.767	.121	.719	.091	.844	.071	.760	.154	-	-
DS [46]$_{\mathrm{TIP'16}}$	134.27M	211.28G	.882	.122	.765	.176	.745	.120	.865	.080	.784	.190	.777	.090
DCL [43]$_{\mathrm{CVPR'16}}$	-	-	.896	.080	.805	.115	.733	.094	.893	.063	.831	.131	.786	.081
RFCN [76]$_{\mathrm{ECCV'16}}$	19.08M	64.95G	.898	.097	.827	.118	.747	.094	.895	.079	.805	.161	.786	.090
DHS [50]$_{\mathrm{CVPR'16}}$	93.76M	25.82G	.905	.062	.825	.092	-	-	.892	.052	.823	.128	.815	.065
MSR [41]$_{\mathrm{CVPR'17}}$	-	-	.903	.059	.839	.083	.790	.073	.907	.043	.841	.111	.824	.062
DSS [30]$_{\mathrm{PAMI'19}}$	62.23M	276.37G	.906	.064	.821	.101	.760	.074	.900	.050	.834	.125	.813	.065
NLDF [57]$_{\mathrm{CVPR'17}}$	35.48M	57.73G	.903	.065	.822	.098	.753	.079	.902	.048	.837	.123	.816	.065
UCF [95]$_{\mathrm{CVPR'17}}$	29.47M	146.42G	.908	.080	.820	.127	.735	.131	.888	.073	.798	.164	.771	.116
Amulet [94]$_{\mathrm{ICCV'17}}$	33.15M	40.22G	.911	.062	.826	.092	.737	.083	.889	.052	.799	.146	.773	.075
GearNet [32]$_{\mathrm{CoRR'17}}$	-	-	.923	.055	-	-	.790	.068	.934	.034	.853	.117	-	-
PAGR [96]$_{\mathrm{CVPR'18}}$	-	-	.924	.064	.847	.089	.771	.071	.919	.047	-	-	.854	.055
SRM [77]$_{\mathrm{ICCV'17}}$	53.14M	36.82G	.916	.056	.838	.084	.769	.069	.906	.046	.840	.126	.826	.058
DGRL [78]$_{\mathrm{CVPR'18}}$	161.74M	191.28G	.921	.043	.844	.072	.774	.062	.910	.036	.843	.103	.828	.049
PiCANet [51]$_{\mathrm{CVPR'18}}$	47.22M	54.05G	.932	.048	.864	.075	.820	.064	.920	.044	.861	.103	.863	.050
PoolNet [49]$_{\mathrm{CVPR'19}}$	68.26M	88.89G	.940	.042	.863	.075	.830	.055	.934	.032	.867	.100	.886	.040
Light-weight models designed for other tasks:														
Eff.Net [71]$_{\mathrm{ICML'19}}$	8.64M	2.62G	.828	.129	.739	.158	.696	.129	.807	.116	.712	.199	.687	.135
Sf.Netv2 [58]$_{\mathrm{ECCV'18}}$	9.54M	4.35G	.870	.092	.781	.127	.720	.100	.853	.078	.779	.163	.743	.096
ENet [65]$_{\mathrm{CoRR'16}}$	0.36M	0.40G	.857	.107	.770	.138	.730	.109	.839	.094	.741	.183	.730	.111
CGNet [84]$_{\mathrm{CoRR'18}}$	0.49M	0.69G	.868	.099	.784	.130	.727	.108	.849	.088	.772	.168	.742	.106
DABNet [40]$_{\mathrm{BMVC'19}}$	0.75M	1.03G	.877	.091	.790	.123	.747	.094	.862	.078	.778	.157	.759	.093
ESPNetv2 [61]$_{\mathrm{CVPR'19}}$	0.79M	0.31G	.889	.081	.795	.119	.760	.088	.872	.069	.780	.157	.765	.089
BiseNet [89]$_{\mathrm{ECCV'18}}$	12.80M	2.50G	.894	.078	.817	.115	.762	.087	.872	.071	.796	.148	.778	.084
Ours:														
CSF+R	36.37M	18.40G	.940	.041	.866	.073	.821	.055	.930	.033	.866	.106	.881	.039
CSF+R2	36.53M	18.96G	.947	.036	.876	.068	.833	.055	.936	.030	.870	.098	.893	.037
CSNet×1-L	94K	0.43G	.900	.075	.819	.110	.777	.087	.889	.065	.809	.149	.799	.082
CSNet×1.5-L	118K	0.63G	.912	.070	.831	.105	.783	.082	.893	.062	.808	.139	.809	.076
CSNet×1.5-L$_{\mathrm{ImageNet}}$	124K	0.63G	.911	.070	.835	.103	.781	.084	.898	.060	.818	.141	.810	.077
CSNet×2-L	141K	0.72G	.916	.066	.835	.102	.792	.080	.899	.059	.825	.137	.819	.074

TABLE 8
Run-time of models using 224 × 224 input on a single core i7-8700K CPU.

Method	MACC (G)	Run-time (ms)
PiCANet [50]	54.06	2850.2
PoolNet [49]	88.89	997.3
ENet [65]	0.40	89.9
ESPNetv2 [61]	0.31	186.3
CSNet×1	0.61	135.9
CSNet×1-L	0.43	95.3

TABLE 9
Incorporating dynamic weight decay into pruning methods. Standard/Dynamic denotes standard/dynamic weight decay.

	PARM.	MACC	F_β	M
Pruning Filters [44]				
Standard	227K	0.69G	88.7	0.080
Dynamic	226K	0.69G	89.4	0.078
Geometric-Median [27]				
Standard	227K	0.70G	88.7	0.083
Dynamic	226K	0.68G	89.6	0.082

dynamic weight decay. Features of dynamic weight decay based models are more stabilized due to the weights that form stable output feature distribution. Fig. 14 shows the testing MAE of each epoch with/without dynamic weight decay. Training with dynamic weight decay leads to better performance in terms of MAE.

Fixed pruning ratio/threshold: We follow [53] to use the weight of the BatchNorm layer as the indicator of the channel importance. We modify the pruning method in [53] by using a fixed threshold to eliminate channels instead of using a fixed pruning ratio. Tab. 10 shows that pruning with a fixed threshold achieves better performance with fewer parameters than using a fixed pruning ratio. The reason behind this result is that different layers require a different number of channels. Therefore, pruning with a threshold can get a unique number of channel for each layer. As shown in Fig. 5, there is a clear gap between the large weights and the weights close to zero. Using arbitrary thresholds within this gap would almost have no difference to the final performance of models.

Integrating dynamic weight decay into pruning methods: By default, we use the pruning method in [53] to eliminate the redundant weights. Since our proposed dynamic weight decay focuses on introducing sparsity while maintaining a stable and compact distribution of weights among channels, it is orthogonal to commonly used pruning methods that focus on identifying unnecessary weights. Thus, we integrate the dynamic weight decay into several pruning methods as shown in Tab. 9. All configurations remain the same except for replacing the standard weight decay with our proposed dynamic weight decay. Pruning

TABLE 10
Pruning with fixed threshold/ratio in the CSNet×2-L.

	Fixed threshold										Fixed ratio	
threshold/ratio	1e-2	8e-3	6e-3	5e-2	3e-2	1e-3	1e-5	1e-10	1e-15	1e-20	38%	51%
Parms. (K)	55.7	79.4	105.0	118.5	135.9	136.2	139.9	140.5	140.8	140.8	300.0	400.0
F_β	37.8	39.3	57.1	82.5	91.2	91.2	91.5	91.5	91.5	91.6	87.7	91.1

TABLE 11
The compression ratio of the CSNet with different initial channel widths. The pruning rate is defined as the ratio of model complexity between pruned parts and the complete CSNet.

Width	Prune	×1	×1.2	×1.5	×1.8	×2.0
Parms	N	211K	298K	455K	645K	788K
	Y	94K	109K	118K	134K	141K
Ratio		55%	63%	74%	79%	82%
MACC	N	0.61G	0.82G	1.17G	1.58G	1.87G
	Y	0.43G	0.52G	0.63G	0.71G	0.72G
Ratio		30%	37%	46%	55%	61%
F_β	N	90.0	90.7	91.1	91.2	91.5
	Y	90.0	90.7	91.2	91.3	91.6

methods [27], [44] equipped with dynamic weight decay achieve better performance using similar parameters.

Pruning rate & Channel width: An initial model with a large channel width is required for learning more useful features. We linearly expand the number of channel of gOctConvs to enlarge the initial model capacity. A pruning rate is defined as the ratio of model complexity between pruned parts and the complete CSNet. Tab. 11 shows the pruning rate of CSNet with different initial channel widths. The split-ratio of gOctConvs for the initial model is set to 5/5. Larger initial width results in better performance as expected. As the initial width increases, the complexity of pruned models only has a small increment. The quality of the pruned model is dependant on the initial model size. Also, benefiting from the stable distribution introduced by dynamic weight decay, compressed models have similar or even better performance than the initial model.

7 Conclusion and Discussion

In this paper, we propose an extremely light-weight holistic model strongly tied to the SOD task, by abandoning the classification backbone and reducing the representation redundancy with a novel dynamic weight decay scheme. The dynamic weight decay scheme maintains a stable weights distribution among channels and stably boosts the sparsity of parameters during training, allowing 80% reduction in parameters with a negligible performance drop. Our proposed CSNet achieves comparable performance with $\sim 0.2\%$ parameters (100k) of large models on popular salient object detection benchmarks. Based on our proposed CSNet, we reveal several properties of the CNN-based SOD model including 1) SOD models are category insensitive and the detected salient objects are generic and category-independent, 2) ImageNet pre-training is not necessary for SOD training, and 3) SOD models require fewer parameters compared with classification models.

Our two major contributions: analyzing the semantics of SOD model and the extremely light-weight holistic SOD model are interdependent of each other. The intentional design of our holistic CSNet make it possible to analyze the semantics of the SOD model. CSNet is trained from scratch, and therefore be free from the potential influence of ImageNet pre-trained backbones, forming the basis of analyzing the category dependency of SOD models. Also, the self-adaptive property and the simple yet effective structure of CSNet benefit the analysis of SOD model complexity and feature requirements. From another perspective, the analysis of SOD model support the designing principle of CSNet. Our analysis proves that category information is not needed by the SOD model, therefore we can abandon the ImageNet pretrained backbone to reduce a great deal of redundancy for the SOD task. And we can design the holistic model specifically for the SOD task instead of adding extra modules on classification backbones to make up for the different between classification backbones and the SOD task.

Future research should focus on analyzing SOD models from other perspectives in particular with more diversity in SOD model structures, and building even more efficient models in terms of speed and accuracy. To facilitate follow-up works we share our code at https://mmcheng.net/sod100k/.

Acknowledgement. This research was supported by the Major Project for New Generation of AI under Grant No. 2018AAA0100400, NSFC (61620106008), S&T innovation project from Chinese Ministry of Education, the Fundamental Research Funds for the Central Universities (Nankai University, 63213090), and Tianjin Natural Science Foundation (18ZXZNGX00110).

References

[1] R. Achanta, S. Hemami, F. Estrada, and S. Süsstrunk. Frequency-tuned salient region detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1597–1604, 2009.
[2] A. Borji, M.-M. Cheng, Q. Hou, H. Jiang, and J. Li. Salient object detection: A survey. *Computational Visual Media*, 5(2):117–150, 2019.
[3] A. Borji, M.-M. Cheng, H. Jiang, and J. Li. Salient object detection: A benchmark. *IEEE transactions on image processing*, 24(12):5706–5722, 2015.
[4] S. Chen, X. Tan, B. Wang, and X. Hu. Reverse attention for salient object detection. In *European Conference on Computer Vision*, 2018.
[5] T. Chen, M.-M. Cheng, P. Tan, A. Shamir, and S.-M. Hu. Sketch2photo: Internet image montage. *ACM T. Graph.*, 28(5):124:1–10, 2009.
[6] Y. Chen, H. Fan, B. Xu, Z. Yan, Y. Kalantidis, M. Rohrbach, S. Yan, and J. Feng. Drop an octave: Reducing spatial redundancy in convolutional neural networks with octave convolution. In *Int. Conf. Comput. Vis.*, 2019.
[7] 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*, 32(1):110–121, 2017.
[8] M.-M. Cheng, N. Mitra, X. Huang, and S.-M. Hu. Salientshape: group saliency in image collections. *The Visual Computer*, 30(4):443–453, 2014.
[9] M.-M. Cheng, N. J. Mitra, X. Huang, P. H. Torr, and S.-M. Hu. Global contrast based salient region detection. *IEEE T. Pattern Anal. Mach. Intell.*, 37(3):569–582, 2015.
[10] M.-M. Cheng, J. Warrell, W.-Y. Lin, S. Zheng, V. Vineet, and N. Crook. Efficient salient region detection with soft image abstraction. In *Int. Conf. Comput. Vis.*, pages 1529–1536, 2013.

[11] R. Desimone and J. Duncan. Neural mechanisms of selective visual attention. *Annual review of neuroscience*, 18(1):193–222, 1995.

[12] R. Dongsheng, W. Jun, and Z. Nenggan. Linear context transform block. *arXiv preprint arXiv:1909.03834*, 2019.

[13] 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.*, September 2018.

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

[15] 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 T. Pattern Anal. Mach. Intell.*, 2020.

[16] S.-H. Gao, Q. Han, D. Li, P. Peng, M.-M. Cheng, and P. Peng. Representative batch normalization with feature calibration. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021.

[17] S.-H. Gao, Q. Han, Z.-Y. Li, P. Peng, L. Wang, and M.-M. Cheng. Global2local: Efficient structure search for video action segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021.

[18] S.-H. Gao, Z.-Y. Li, M.-H. Yang, M.-M. Cheng, J. Han, and P. Torr. Large-scale unsupervised semantic segmentation, 2021.

[19] S.-H. Gao, Y.-Q. Tan, M.-M. Cheng, C. Lu, Y. Chen, and S. Yan. Highly efficient salient object detection with 100k parameters. In *Eur. Conf. Comput. Vis.*, 2020.

[20] 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.

[21] Q. Han, K. Zhao, J. Xu, and M.-M. Cheng. Deep hough transform for semantic line detection. In *Eur. Conf. Comput. Vis.*, 2020.

[22] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 447–456, 2015.

[23] J. He, J. Feng, X. Liu, C. Tao, and S. F. Chang. Mobile product search with bag of hash bits and boundary reranking. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2012.

[24] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Int. Conf. Comput. Vis.*, pages 1026–1034, 2015.

[25] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 770–778, 2016.

[26] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang. Soft filter pruning for accelerating deep convolutional neural networks. In *Int. Jt. Conf. Artif. Intell.*, 2018.

[27] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4340–4349, 2019.

[28] Y. He, X. Zhang, and J. Sun. Channel pruning for accelerating very deep neural networks. In *Int. Conf. Comput. Vis.*, pages 1389–1397, 2017.

[29] S. Hong, T. You, S. Kwak, and B. Han. Online tracking by learning discriminative saliency map with convolutional neural network. In *International Conference on Machine Learning (ICML)*, 2015.

[30] Q. Hou, M.-M. Cheng, X. Hu, A. Borji, Z. Tu, and P. Torr. Deeply supervised salient object detection with short connections. *IEEE T. Pattern Anal. Mach. Intell.*, 41(4):815–828, 2019.

[31] Q. Hou, P.-T. Jiang, Y. Wei, and M.-M. Cheng. Self-erasing network for integral object attention. In *NeurIPS*, 2018.

[32] Q. Hou, J. Liu, M.-M. Cheng, A. Borji, and P. H. Torr. Three birds one stone: a unified framework for salient object segmentation, edge detection and skeleton extraction. *arXiv preprint arXiv:1803.09860*, 2018.

[33] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, et al. Searching for mobilenetv3. *arXiv preprint arXiv:1905.02244*, 2019.

[34] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[35] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018.

[36] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, 2015.

[37] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE T. Pattern Anal. Mach. Intell.*, 20(11):1254–1259, 1998.

[38] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Int. Conf. Learn. Represent.*, 2014.

[39] A. Krogh and J. A. Hertz. A simple weight decay can improve generalization. In *Adv. Neural Inform. Process. Syst.*, pages 950–957, 1992.

[40] G. Li and J. Kim. Dabnet: Depth-wise asymmetric bottleneck for real-time semantic segmentation. In *Brit. Mach. Vis. Conf.*, 2019.

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

[42] G. Li and Y. Yu. Visual saliency based on multiscale deep features. In *IEEE Conf. Comput. Vis. Pattern Recog.*, June 2015.

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

[44] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning filters for efficient convnets. In *Int. Conf. Learn. Represent.*, 2016.

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

[46] X. Li, L. Zhao, L. Wei, M.-H. Yang, F. Wu, Y. Zhuang, H. Ling, and J. Wang. Deepsaliency: Multi-task deep neural network model for salient object detection. *IEEE T. Image Process.*, 25(8):3919 – 3930, Aug 2016.

[47] 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.*, June 2014.

[48] M. Lin, Q. Chen, and S. Yan. Network in network. In *Int. Conf. Learn. Represent.*, 2013.

[49] 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.

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

[51] N. Liu, J. Han, and M.-H. Yang. Picanet: Learning pixel-wise contextual attention for saliency detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, June 2018.

[52] 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*, 33(2):353–367, 2011.

[53] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang. Learning efficient convolutional networks through network slimming. In *Int. Conf. Comput. Vis.*, pages 2736–2744, 2017.

[54] Z. Liu, H. Mu, X. Zhang, Z. Guo, X. Yang, T. K.-T. Cheng, and J. Sun. Metapruning: Meta learning for automatic neural network channel pruning. In *Int. Conf. Comput. Vis.*, 2019.

[55] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3431–3440, 2015.

[56] J.-H. Luo, J. Wu, and W. Lin. Thinet: A filter level pruning method for deep neural network compression. In *Int. Conf. Comput. Vis.*, pages 5058–5066, 2017.

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

[58] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Eur. Conf. Comput. Vis.*, pages 116–131, 2018.

[59] R. Margolin, L. Zelnik-Manor, and A. Tal. Saliency for image manipulation. *The Visual Computer*, 29(5):381–392, 2013.

[60] D. Mehta, K. I. Kim, and C. Theobalt. On implicit filter level sparsity in convolutional neural networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 520–528, 2019.

[61] S. Mehta, M. Rastegari, L. Shapiro, and H. Hajishirzi. Espnetv2: A light-weight, power efficient, and general purpose convolutional neural network. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 9190–9200, 2019.

[62] A. Mordvintsev, C. Olah, and M. Tyka. Inceptionism: Going deeper into neural networks, 2015.

[63] V. Movahedi and J. H. Elder. Design and perceptual validation of performance measures for salient object segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog. Worksh.*, pages 49–56, June 2010.

[64] Y. Pang, X. Zhao, L. Zhang, and H. Lu. Multi-scale interactive network for salient object detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 9413–9422, 2020.

[65] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*, 2016.

[66] Y. Piao, W. Ji, J. Li, M. Zhang, and H. Lu. Depth-induced multi-scale recurrent attention network for saliency detection. In *Int. Conf. Comput. Vis.*, October 2019.

[67] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vis.*, 115(3):211–252, 2015.

[68] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4510–4520, 2018.

[69] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *Int. Conf. Learn. Represent.*, 2014.

[70] K. Sun, B. Xiao, D. Liu, and J. Wang. Deep high-resolution representation learning for human pose estimation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 5693–5703, 2019.

[71] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning (ICML)*, 2019.

[72] A. M. Treisman and G. Gelade. A feature-integration theory of attention. *Cognitive psychology*, 12(1):97–136, 1980.

[73] J. Wang, H. Jiang, Z. Yuan, M.-M. Cheng, X. Hu, and N. Zheng. Salient object detection: A discriminative regional feature integration approach. *Int. J. Comput. Vis.*, 123(2):251–268, 2017.

[74] 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.

[75] L. Wang, H. Lu, R. Xiang, and M. H. Yang. Deep networks for saliency detection via local estimation and global search. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[76] L. Wang, L. Wang, H. Lu, P. Zhang, and X. Ruan. Saliency detection with recurrent fully convolutional networks. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *Eur. Conf. Comput. Vis.*, pages 825–841, 2016.

[77] 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.*, Oct 2017.

[78] 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.*, June 2018.

[79] W. Wang, S. Zhao, J. Shen, S. C. H. Hoi, and A. Borji. Salient object detection with pyramid attention and salient edges. In *The IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[80] X. Wang, X. Liang, B. Yang, and F. W. Li. No-reference synthetic image quality assessment with convolutional neural network and local image saliency. *Computational Visual Media*, 5(2):193–208, 2019.

[81] J. Wei, S. Wang, Z. Wu, C. Su, Q. Huang, and Q. Tian. Label decoupling framework for salient object detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 13025–13034, 2020.

[82] J. M. Wolfe and T. S. Horowitz. What attributes guide the deployment of visual attention and how do they do it? *Nature reviews neuroscience*, 5(6):495–501, 2004.

[83] 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 *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[84] T. Wu, S. Tang, R. Zhang, and Y. Zhang. Cgnet: A light-weight context guided network for semantic segmentation. *arXiv preprint arXiv:1811.08201*, 2018.

[85] S. Xie and Z. Tu. Holistically-nested edge detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1395–1403, 2015.

[86] Q. Yan, L. Xu, J. Shi, and J. Jia. Hierarchical saliency detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, June 2013.

[87] 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.*, pages 3166–3173, 2013.

[88] H. Yin, P. Molchanov, J. M. Alvarez, Z. Li, A. Mallya, D. Hoiem, N. K. Jha, and J. Kautz. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 8715–8724, 2020.

[89] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *Eur. Conf. Comput. Vis.*, pages 325–341, 2018.

[90] Y. Zeng, P. Zhang, J. Zhang, Z. Lin, and H. Lu. Towards high-resolution salient object detection. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[91] G. Zhang, C. Wang, B. Xu, and R. Grosse. Three mechanisms of weight decay regularization. In *Int. Conf. Learn. Represent.*, 2019.

[92] G.-X. Zhang, M.-M. Cheng, S.-M. Hu, and R. R. Martin. A shape-preserving approach to image resizing. *Computer Graphics Forum*, 28(7):1897–1906, 2009.

[93] L. Zhang, J. Dai, H. Lu, Y. He, and G. Wang. A bi-directional message passing model for salient object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[94] 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.*, Oct 2017.

[95] P. Zhang, D. Wang, H. Lu, H. Wang, and B. Yin. Learning uncertain convolutional features for accurate saliency detection. In *Int. Conf. Comput. Vis.*, pages 212–221. IEEE, 2017.

[96] 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.*, June 2018.

[97] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 6848–6856, 2018.

[98] 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.*, October 2019.

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

[100] X. Zhao, Y. Pang, L. Zhang, H. Lu, and L. Zhang. Suppress and balance: A simple gated network for salient object detection. In *Eur. Conf. Comput. Vis.*, pages 35–51. Springer, 2020.

[101] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2921–2929, 2016.

[102] W. Zhu, S. Liang, Y. Wei, and J. Sun. Saliency optimization from robust background detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2814–2821, 2014.