C^{4}Net: Contextual Compression and Complementary Combination Network for Salient Object Detection

Hazarapet Tunanyan
hazarapet.tunanyan@picsart.com

Picsart AI Research (PAIR)
Yerevan, Armenia

Abstract

Deep learning solutions of the salient object detection problem have achieved great results in recent years. The majority of these models are based on encoders and decoders, with a different multi-feature combination. In this paper, we show that feature concatenation works better than other combination methods like multiplication or addition. Also, joint feature learning gives better results, because of the information sharing during their processing. We designed a Complementary Extraction Module (CEM) to extract necessary features with edge preservation. Our proposed Excessiveness Loss (EL) function helps to reduce false-positive predictions and purifies the edges with other weighted loss functions. Our designed Pyramid-Semantic Module (PSM) with Global guiding flow (G) makes the prediction more accurate by providing high-level complementary information to shallower layers. Experimental results show that the proposed model outperforms the state-of-the-art methods on all benchmark datasets under three evaluation metrics.

1 Introduction

Salient object detection (SOD) aims to detect the most visually attractive parts of images and videos, which is widely used in many applications like visual tracking [15], image retrieval [7], content-aware image editing [3], robot navigation [4], and many others. For many years, researchers have proposed some solutions [11, 14, 18, 27], which are based on hand-crafted features (*e.g.* color, contrast, and texture), however, these approaches can not take into account high-level semantic information, which is a big restriction for the solution of the problem. The recent development of deep convolutional neural networks (CNNs) demonstrated powerful capabilities in terms of the high (*e.g.* class, position) and low (*e.g.* high-frequency data, detailed information) level feature extraction, which promotes better results of the salient object detection problem.

Many modern solutions propose U-Net like architectures to solve the problem, still, the feature combination and their processing methods have big potential to improve. Many models observe feature fusing methods with different aggregation functions like multiplication and addition. We find these methods have some drawbacks, because of the choice of aggregation functions. Also, these models process the features of encoder and decoder separately, which promotes cross-necessary information loss. The processed information of encoders and decoders is complementary to each other and they need to be managed together.

© 2021. The copyright of this document resides with its authors.
It may be distributed unchanged freely in print or electronic forms.

By analyzing the visual results of different models, we have noticed there are some incorrect predictions like random holes or excessive parts. These types of issues often come from the lack of high-level semantic information. After a couple of comprehensive experiments, we have come up with a modification of the pyramid-pooling module [13] and called it pyramid-semantic module, which contains multi-scale context-aware feature representation and channel-wise shift and attention.

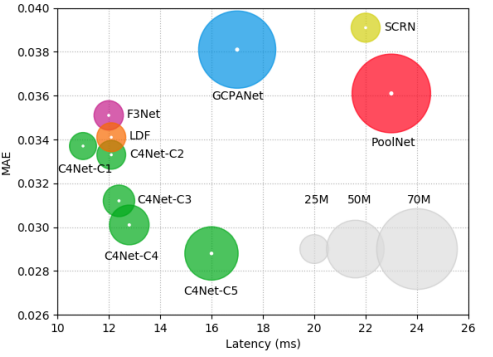

Figure 1: Performance comparison between our proposed and other state-of-the-art methods by their complexity, latency (ms) and MAE results on DUTS-Test dataset.

We proposed a complementary extraction module to combine and process low and high-level information by taking into account the feature representations of the edges. Contextual compression module is made to compress the connections between encoder and decoder, to maintain necessary features, and make the processing faster. Also, our proposed loss function helps to minimize false predictions and is formulated by false-positive values for each pixel. So our contributions are the following.

- Proposed a family of *C*[4]*Net* architectures with different complexities and performance.
- Proposed the *Complementary Extraction Module* to combine and process low and high-level information with edge preservation.
- Proposed *Excessiveness Loss* function to minimize false-positive predictions.

2 Related Work

During recent decades, researchers developed algorithms for the salient object detection problem, which are based on hand-crafted features (*e.g.* color or texture) [11, 14, 18, 27]. Most of these traditional methods have been surpassed by convolutional neural networks (CNN) in terms of quality and speed. Deep convolutional networks are capable to extract necessary semantic information and combine them properly. By bringing all known approaches together, we can formulate the following common factors.

Features Processing Methods. As the majority of the recent solutions are based on U-Net like architectures, one of the most important parts is the feature processing part at every layer of the decoder. Zuyao Chen *et al.* [2] considered separate processing. They applied multiplication operation between the encoder's and decoder's features, then concatenated the results of different branches. Jun Wei *et al.* [23] also proposed almost the same approach, where their model separately processes encoder's and decoder's features, multiplies them as a fusing operation then they apply an addition operation on it. They considered that the encoder's features are low-level representations and contain detailed information. In contrast, the decoder's features are high-level representations and contain rich semantic information. By multiplying them, they clean the information and add complementary features for high-quality detection. We find these approaches have some drawbacks, because of the separate processing of high and low-level features, which causes cross-necessary information loss.

Edge Preservation Approaches. The salient object detection problem requires a solution with high-quality detection, especially on edges, which need to be purified and exquisite. To maintain the high quality of edges, Jiang-Jiang Liu *et al*. [13] apply additional edge detection on the middle layers of the decoder. Mengyang Feng *et al*. [6] proposed the *Boundary-Enhanced Loss* function for shallower layers of the network, which is responsible for extracting the features and purifying the edges. All these methods tend to improve the quality of edges by using additional and separated edge-specific processing. We find that the core loss functions also must be adjusted for this manner and proposed weighted losses, where edge pixels have high weights. A similar solution has been done in methods like [23].

Global Features Extraction. As we have mentioned in the previous sections, one of the advantages of deep learning solutions is the capability of semantic information extraction, but the majority of these approaches do not have direct ties between global semantic modules and shallower layers. Jiang-Jiang Liu *et al*. [13] and Zuyao Chen *et al*. [2] gave solutions for this manner. They designed a module for high-level semantic information and a global guiding flow to distribute the information in top layers. Our contribution tends to improve global information extraction by channel-wise attention and shifting.

3 Proposed Method

In this section, we will describe the architecture of our model and all consisting parts of it. Before going into details, we first need to refer to the feature combination approach. As U-Net like architectures have shortcut connections between encoder and decoder, it is very important how their feature representations will be combined. Based on the conducted experiments and the design choice of our model, we found out that joint processing of encoder's and decoder's features works better than the branched-separated approach. Also, the concatenation aggregation function for skip connections and feature combinations leads better results compared with other aggregation functions like addition and multiplication. The details about those approaches can be found in the ablation study (Section 4.4).

3.1 Contextual Compression Module

As we noted in the previous section, shortcut connections between encoder and decoder are crucial for high-quality detection and exquisite boundaries. We defined a new term called Compression Factor (CF) to compress half of the channels of the network, which starts from the shortcut connections.

$$f_l = \delta(BN(Conv_{cf}(f_l^e))) \qquad (1)$$

Name	Encoder	CF	*MAE*	*mF*	E_ξ
C^{4}Net-C1	R34	64	0.033	0.872	0.925
C^{4}Net-C2	R50	32	0.033	0.867	0.928
C^{4}Net-C3	R50	64	0.031	0.872	0.929
C^{4}Net-C4	R50	128	0.030	0.875	0.929
C^{4}Net-C5	R101	64	0.029	0.886	0.934

Table 1: Architectures of our proposed model with different complexities. Results are based on *DUTS-Test* dataset.

where f_l^e is the feature representation of the encoder, $Conv_{cf}$ is a convolution layer with cf filters ({32, 64, 128} in our experiments), *BN* is a batch normalization layer, δ is the ReLU activation function. By conducting a couple of comprehensive experiments for shortcut connections, we came up with the following conclusion.

- Without any additional compression block, the encoder provides noisy information. The experimental results are reported in the fifth and last row of Table 2.

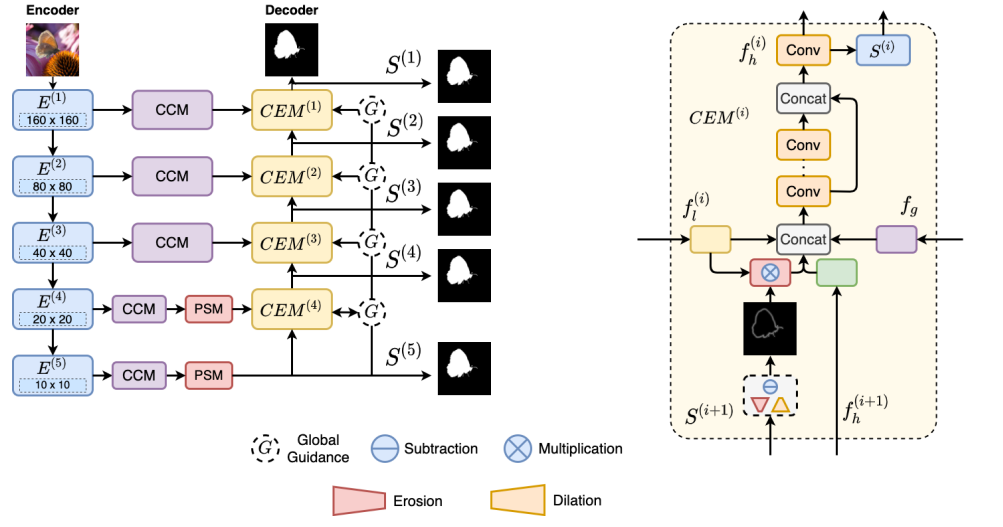

Figure 2: An overview of our proposed C[4]Net architecture. The model is based on a *ResNet* [10] encoder with multilevel supervision $S^{(i)}$. Contextual Compression Module (CCM) is used as compressed shortcut connection between encoder and decoder. Pyramid-Semantic Module (PSM) is used to extract high level semantic information, which is also used in Global Guidance flow (G) and Complementary Extraction Module (CEM) is used to combine three feature representations from the encoder, decoder and guiding flow.

- By using feature compression for half of the network, we have increased the speed of training and testing regimes, also, optimized the memory allocation of it.

We use the same compression factor for all layers of the CCM module and for the decoder of our network. Table 1 contains an ablation study for different compression factors and Table 2 shows the effectiveness of the CCM module. Each next row's configuration of Table 2 either is built on top of the previous row or is replaced with the mentioned module.

3.2 Pyramid-Semantic Module

We have noted the importance of low-level information, but high-level information is also very important for better detection. We have modified the PPM module proposed in [13] and designed the Pyramid-Semantic module, which is responsible to handle and maintain high-level information. PSM consists of two main parts, which are visualized in Figure 3. The first part is responsible for pyramid-feature extraction, which is made by branched pooling operation and the second part is a channel-wise attention module, which scales and shifts feature representations by channels. For the first part of this module, we use average pooling operation to get multi-resolution pyramid features. Also, It has an identical branch, which lets to maintain high-level semantic information of the same resolution.

$$\bar{f}_h^{(j)} = Up(Conv(Pool_k(f_l))) \tag{2}$$

where f_l is the input feature representation of deeper layers provided by the CCM module, $Pool_k$ is the average pooling operation with kernel size k. For each branch $(j \in \{2,3,4\})$ we use different kernel sizes: *{10, 5, 2}*. *Conv* is a combination of convolution, batch normalization, and a ReLU activation functions. Up is a *bilinear* up-sampling operation.

At the end of the first part, all four branches get concatenated into one feature representation.

The second part is the channel-wise attention module, which scales and shifts features. Let $f_h \in R^{H\times W\times C}$ be the input features. We apply global pooling operation on f_h and get $\tilde{f} \in R^C$.

$$w = \sigma(fc_2(\delta(fc_1(\tilde{f}, W_1^{(w)})), W_2^{(w)}))$$
$$v = \sigma(fc_2(\delta(fc_1(\tilde{f}, W_1^{(v)})), W_2^{(v)})) \quad (3)$$

$$f_h = w \odot f_h \oplus v, \quad w, v \in R^C \quad (4)$$

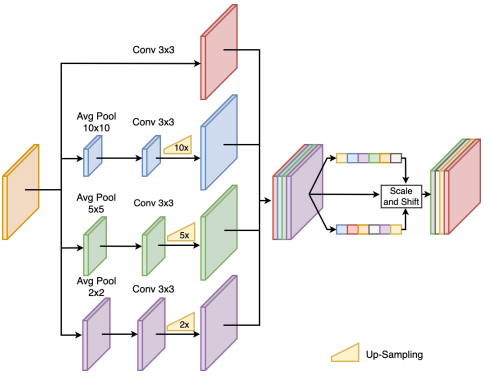

Figure 3: Illustration of our Pyramid-Semantic Module.

where fc_1 and fc_2 are fully connected layers, $W^{(w)}$ and $W^{(v)}$ are weights matrices for scaling and shifting respectively. δ is the ReLU activation function, and σ is the Sigmoid activation function, $\odot$ is channel-wise multiplication and $\oplus$ is channel-wise addition function.

To verify the effectiveness of our modification, we have conducted two other experiments as well with other similar solutions like *Pyramid-Pooling Module (PPM)* [13] and *Atrous Spatial Pyramid Pooling (ASPP)* [1]. Table 2 shows that our method outperforms other pyramid-based approaches mainly because of the channel-wise shift and attention.

3.3 Complementary Extraction Module

As we have referred to the structure of feature processing approach at the beginning of Section 3, it is crucial to choose a right extraction mechanism and a combination function, thus we have decided to choose the joint processing method (PipeMode) with concatenation combination function, as they perform better for the model design we have chosen (see Section 4.4). As a summary, we have designed a module, which is responsible for three different feature extraction. $f_l \in R^{H\times W\times C}$ is the low-level feature representation, which contains rich details with noise, in contrast to $f_h \in R^{H\times W\times C}$, which is a high-level feature and does not contain rich details for exquisite detection, but it is noisy-free. $f_g \in R^{H\times W\times C}$, where $f_g = f_h^4 + f_h^5$ is a global guidance flow using the fourth and fifth layer's features, which helps to complement high-level semantic information in shallow layers. As we seek high-quality detection, we also need to reduce the error on edges, because edges contain the most errors of the detection as shown in [24]. To adjust the model for multi-level supervision, we apply a convolution layer with the sigmoid function on the output of the previous layer and get $S^{(i)}$ binary mask. We compute edges by dilating and eroding the $S^{(i)}$ mask, then apply pixel-wise multiplication with f_l. f_{edge} will be the edge feature representation. We concatenate all four

Modules	*MAE*	*mF*	E_ξ
Baseline (w/o shortcut)	0.045	0.804	0.903
EL	0.043	0.813	0.904
EL + CCM	0.034	0.851	0.917
EL + CCM + CEM	0.032	0.867	0.921
EL + CCM + CEM + PSM	0.031	0.872	0.929
EL + CCM + CEM + PPM	0.033	0.865	0.923
EL + CCM + CEM + ASPP	0.033	0.867	0.924
CCM + CEM + PSM	0.033	0.867	0.922
EL + CEM + PSM†	0.033	0.874	0.924

Table 2: Ablation study for different modules of C[4]Net-C3 architecture. Symbol † indicates a model with straight shortcut connections w/o any operation. The results are based on *DUTS-Test* dataset.

features, f_l, f_h, f_{edge} and f_g and feed into a convolution block with six convolutions, batch normalization, ReLU layers, and a skip connection. Figure 2 contains the visualization of our proposed CEM module and Table 2 shows its effectiveness with three different metrics.

3.4 Excessiveness Loss Function

The binary cross-entropy function is one of the most widely used loss functions in salient object detection problem, however, the equal treatment of pixels and the ignorance of global structures make the function not effective, so we have used the approach proposed by Jun Wei *et al.* [23].

Modules	*MAE*	*mF*	*mFP*	*mFN*
CCM + CEM + PSM	0.033	0.867	0.022	0.016
EL + CCM + CEM + PSM	0.031	0.872	0.019	0.016

Table 3: Ablation study of *EL* contribution of C[4]Net-C3 architecture on *DUTS-Test* dataset.

First, we have used weighted losses by adding a pixel-wise weight matrix to increase the importance of edges.

$$\omega_{i,j} = 1 + \tilde{\lambda} \left| \frac{1}{N} \sum_{\tilde{i}=1,\tilde{j}=1}^{k,k} Gt_{\tilde{i},\tilde{j}} - Gt_{i,j} \right| \tag{5}$$

where Gt is the ground truth mask, $\tilde{\lambda}$ is a hyper parameter, $|.|$ is the absolute value, $N = k \times k$ is the number of pixels of the window with kernel size k.

$$L_{wbce} = - \frac{\sum_{i=1,j=1}^{W,H} \omega_{i,j} * (Gt_{i,j} * \log(S_{i,j}) + (1 - Gt_{i,j}) * \log(1 - S_{i,j}))}{\sum_{i=1,j=1}^{W,H} \omega_{i,j}} \tag{6}$$

where S_{ij} is the prediction of i,j-th pixel and W,H is the width and height of the output.
As we seek a solution with structural preservation of the objects, we use the following loss.

$$L_{wiou} = 1 - \frac{\sum_{i=1,j=1}^{W,H} S_{i,j} * Gt_{i,j} * \omega_{i,j}}{\sum_{i=1,j=1}^{W,H} (S_{i,j} + Gt_{i,j} - S_{i,j} * Gt_{i,j}) * \omega_{i,j}} \tag{7}$$

L_{wiou} is the weighted intersection over union loss function, which handles global structures of the foreground object and increases the impact of the pixels near the edges.
To improve the detection results, we proposed a new loss function, which is called *Excessiveness Loss* function. We observe *false positive* (FP), *false negative* (FN), and *true positive* (TP) predictions for each example. By analyzing the error for these values, we found out that *the error mostly accumulates by excessive predictions*, which means in most cases, FP is higher than FN. Table 3 contains an ablation study, where mFP and mFN are normalized by the resolution of the output mask. Our proposed weighted *excessiveness loss* function is

$$\omega TP = \sum_{i=1,j=1}^{W,H} S_{i,j} * Gt_{i,j} * \omega_{i,j} \quad and \quad \omega FP = \sum_{i=1,j=1}^{W,H} \delta(S_{i,j} - Gt_{i,j}) * \omega_{i,j} \tag{8}$$

$$L_{wel} = \frac{\omega FP}{\omega FP + \gamma \omega TP} \tag{9}$$

where $\omega FP, \omega TP$ are weighted *false positive* and *true positive* respectively and they are differentiable, δ is the ReLU activation function, γ is a hyper-parameter.
The overall loss of our model is a weighted combination of these three loss functions.

$$L = L_{wel}^{(1)} + \sum_{i=1}^{5} \frac{1}{2^{i-1}} (L_{wbce}^{(i)} + L_{wiou}^{(i)}) \tag{10}$$

Algorithms	ECSSD 1,000 images			HKU-IS 4,447 images			PASCAL-S 850 images			DUT-OMRON 5,168 images			DUTS-Test 5,019 images		
	MAE	mF	E_ξ	MAE	mF	E_ξ	MAE	mF	E_ξ	MAE	mF	E_ξ	MAE	mF	E_ξ
PiCANet (CVPR2018)	.046	.919	.951	.043	.900	.947	.075	.831	.893	.065	.759	.860	.050	.828	.909
DGRL (CVPR2018)	.045	.910	.945	.037	.897	.947	.074	.819	.882	.063	.738	.848	.051	.802	.892
EGNet (ICCV2019)	.041	.933	.953	.031	.917	**.956**	.074	.833	.885	.053	.767	.857	.039	.856	.915
SCRN (ICCV2019)	.037	**.935**	.954	.034	.917	.953	.063	**.850**	**.902**	.056	.772	**.863**	.039	**.860**	.915
CPD (CVPR2019)	.037	.923	.950	.034	.905	.948	.070	.828	.884	.056	.754	.850	.043	.836	.906
BASNet (CVPR2019)	.037	.927	.950	.032	.914	.950	.076	.824	.879	.056	**.773**	**.864**	.047	.832	.897
PoolNet (CVPR2019)	.038	.931	.955	.030	.917	.955	.065	.846	.900	.054	.756	.849	.036	.854	.917
MINet (CVPR2020)	.034	.931	**.956**	.029	**.918**	**.959**	.064	.836	.896	.056	.764	.861	.037	.854	**.920**
LDF (CVPR2020)	.034	.930	.926	**.027**	.914	.954	**.060**	.848	.866	**.052**	**.773**	.862	**.034**	.855	.910
GCPANet (AAAI2020)	.035	.929	.954	.031	.917	**.956**	.063	.840	.898	.057	.767	.857	.038	.858	.919
F^3Net (AAAI2020)	**.033**	.925	.930	**.028**	.912	.954	.062	.834	.886	.053	.770	.862	.035	.842	.903
C^4Net-C4 (Ours)	**.029**	**.939**	**.957**	**.027**	**.925**	**.956**	**.056**	**.854**	**.903**	**.051**	**.776**	**.863**	**.030**	**.875**	**.929**
C^4Net-C5 (Ours)	.030	**.939**	.956	**.025**	**.931**	**.961**	**.055**	**.861**	**.904**	**.047**	**.788**	**.865**	**.029**	**.886**	**.937**

Table 4: Performance comparison with 11 state-of-the-art methods on 5 benchmark datasets. MAE (smaller is better), mean F-measure (mF larger is better), E-measure (E_ξ larger is better) metrics are used to evaluate the results. All models are based on ResNet backbones. The best and the second best results among models with ResNet50 backbones highlighted in **green** and **blue** respectively. The **red** color indicates the best results of our model with ResNet101 backbone.

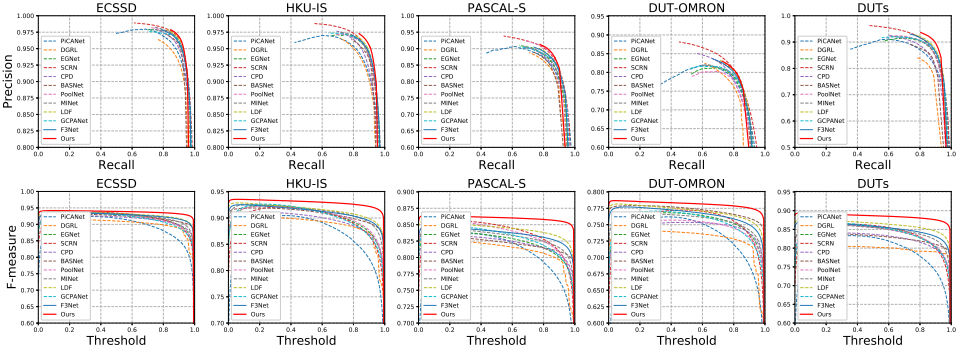

Figure 4: Illustration of PR curves (the first row), F-measure curves (the second row) on 5 benchmark datasets.

where $L^{(i)}_{wel}, L^{(i)}_{wbce}$ and $L^{(i)}_{wiou}$ are the corresponding loss functions for i-th layer. To verify the effectiveness of EL loss function, we show the results of two-way experiments in the first and last group of Table 2.

4 Experiments

4.1 Datasets and Evaluation Metrics

To evaluate our proposed method, we have used five popular benchmark datasets, including ECSSD [20] with 1000 images, PASCAL-S [29] with 850 images, HKU-IS [9] with 4447 images, DUT-OMRON [28] with 5168 images and DUTS [22] with 15572 images. DUTS is currently the biggest salient object detection dataset and contains 10553 training and 5019 testing examples. We used only the DUTS-Train dataset for training and others for testing. Three metrics are used to evaluate the performance of our model and other state-of-the-

art methods. The first metric is the *Mean Absolute Error (MAE)*, which is one of the most commonly used metrics for salient object detection, as shown in Eq. 11. Another widely used metric of SOD problem is the *mean F-measure* (mF) which is formulated with *precision* and *recall* as the F_β score, where $\beta = 0.3$. We also use the E-measure (E_ξ) [5] metric, which uses the combination of local pixel values and their means to evaluate the similarity between prediction and ground truth masks.

$$MAE = \frac{1}{W \times H} \sum_{i=1}^{W} \sum_{j=1}^{H} |P_{i,j} - Gt_{i,j}| \tag{11}$$

where P is the prediction and Gt is the ground truth mask. To show the robustness of our proposed method, we also plot the *Precision-Recall (PR)* curve, which is calculated by sliding the threshold from 0 to 1. The larger the area under the PR curve, the better is the performance.

4.2 Implementation Details

We use DUTS-Train as the training dataset, with randomly cropping and horizontal flipping augmentation techniques. Different architectures of ResNet [10] pre-trained on *ImageNet* are used as the encoder and other parts of the model are initialized randomly from the uniform distribution. We use an adaptive learning rate with a maximum value of 0.005 for the encoder and 0.05 for other parts. The network is trained with *stochastic gradient descent (SGD)* with 0.9 momentum and 0.0005 weight decay parameter values. The batch size is set to 20 with 50 epochs. Our network is implemented with Pytorch v1.6 and the training and testing processes are conducted on a *Nvidia RTX 2080 ti* GPU and *Intel Core i9-9900k* CPU device. The performance report of our proposed models can be found in Figure 1, where our fastest model has about 11ms (90 fps) inference time by surpassing other state-of-the-art solutions. All images are resized to 320×320 during training and testing, without any post-processing approach.

4.3 Comparison with State-of-the-Arts

We compare our proposed algorithm with 11 state-of-the-art methods, including PiCANet-R [16], DGRL [21], EGNet [12], SCRN [26], CPD [25], BASNet [19], PoolNet [13], MINet [17], LDF [24], GCPANet [2] and F³Net [23], which all have ResNet backbones. For fair comparison, we use the same three metrics for all methods with the same script and visualize their PR and F_β curves.

Quantitative Comparison. To compare all listed methods with the metrics mentioned in Section 4.1, we have created Table 4 with 11 state-of-the-art algorithms. Without bells and whistles, our proposed method works better on all benchmark datasets. One of the highest improvements among models with ResNet50 [10] backbone is on the DUTS-Test dataset, where our algorithm boosts the results by more than 11% in terms of the MAE. Our *C⁴Net-C1* model, which has ResNet34 [10] as a backbone, also surpasses many SOTA results. There is a comparison between our proposed architectures and other solutions by their latency (ms), MAE on DUTS-Test, and number of parameters in Figure 1. Some other SOTA models like MINet [17] or BASNet [19] are either too slow or too complex and are out of the chosen window. Also, the PR and F_β curves show the robustness of our models on the mentioned datasets.

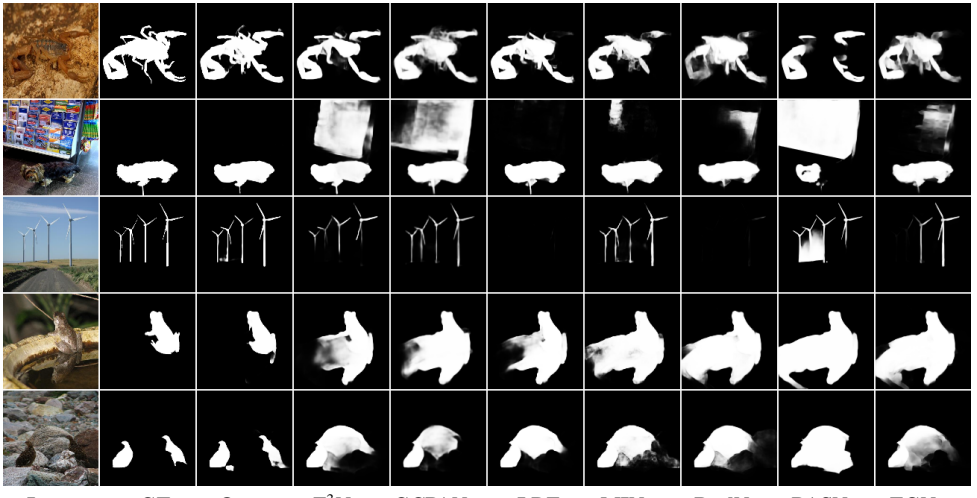

Figure 5: Qualitative comparison of the proposed model with other state-of-the-art methods. Our model has minimal false prediction and the edges are more exquisite than others.

Qualitative Comparison. The visual comparison examples are shown in Figure 5, where our model has better quality on the edges among all algorithms, because of our weighted loss functions for all layers and the complimentary edge-feature extraction in our CEM module, which helps to get exquisite boundaries. Our model is able to detect narrow parts and recover lost information. Another visible improvement is minimal false predictions, especially false-positive values. Our proposed EL function is able to minimize false-positive areas and maximize the true positive predictions, which is visible in the comparison figure.

4.4 Ablation Study

Features Combination Methods. Modern architectures of deep learning solutions for the SOD problem are based on encoders, decoders, and shortcut connections between them. Each layer of the decoder is responsible for combining features of decoder and encoder. There are two main approaches to using that information. The first one is by simply concatenating them, the second one is to fuse them by using other functions like multiplication or addition. Another very important thing is the features processing structure at every layer of the decoder. Methods like [2, 23] prefer to process encoder's and decoder's features separately and then fuse them by multiplication or addition.

Name	R_1	R_2	MAE	mF	E_ξ
BranchPP	Plus	Plus	0.0356	0.8514	0.9162
BranchMM	Mul	Mul	**0.0358**	**0.8496**	0.9165
BranchCC	Cat	Cat	**0.0353**	**0.8535**	**0.9168**
BranchMP	Mul	Plus	**0.0353**	0.8509	**0.9152**
PipeMM	Mul	Mul	-	-	-
PipePP	Plus	Plus	0.0348	**0.8528**	0.9164
PipeCC	Cat	Cat	**0.0342**	0.8512	**0.9171**
PipeCP	Cat	Plus	0.0347	0.8520	0.9159

Figure 6: Results of our proposed structures on *DUTS-Test* dataset with different aggregating functions, where Plus is addition, Mul is multiplication and Cat is concatenation.

We tried to find out the answers to two main questions

- How the features of the encoder and decoder need to be processed. (joint or separated)
- How they need to be combined. (fusing or concatenating)

To answer these questions, we proposed two main structures of a decoder's layer Figure 7. We designed a PipeMode layer, which is a joint processing of two feature representations, and BranchedMode layer, which is separated processing of the features. They both contain two aggregation functions R_1 and R_2.

The features of encoding layers, $f_l^{(i)}$ contain low-level information like edges, tiny areas, and high-frequency data, which is crucial for high-quality detection, especially on edges and they contain noisy information. The feature representations of decoding layers, $f_h^{(i)}$ contain noisy free high-level information like class, position, or shape of the object. These features representations helped Jun Wei *et al.* [23] to propose a solution like our BranchedMode, where they used multiplication function to fuse high and low-level features and to clean noisy parts, then they added a skip-like connection as complementary information. We find this approach has some drawbacks, because of the choice of the structure and aggregating functions. To find out the real behavior of these two approaches, we made different experiments with these structures by using different aggregating functions. We designed a simple architecture with *ResNet50* backbone, our proposed structures for the decoder, and shortcut connections between them. The BCE loss function was used on top of the decoder. Each model was trained three times with randomly initialized weights and the result was calculated by taking the average of the best performance at each run. Table 6 contains reports of our experiments. Based on those results and the architecture choice, we can say:

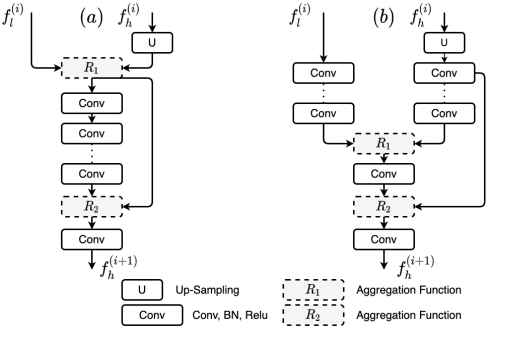

Figure 7: An overview of our proposed structures. (a) is joint-feature mode (PipeMode) and (b) is separate-feature mode (BranchedMode). R_1 and R_2 are aggregation functions.

- In general, PipeMode gives better results than BranchedMode, which shows that the sharing of information about different features leads to better results.
- The concatenation aggregation function works better for both structures.

PipeMM results are missing in Table 6, because it disturbs the training of the model with activation values pushed to zero [8].

5 Conclusion

In this paper, we proposed a new solution for the salient object detection problem. Our ablation study contains an investigation about feature combination approaches, where we showed that joint learning with pipe-mode works better than branched-mode. Also, based on our model design, the feature concatenation gives better results for both structures. Our proposed solution is able to extract and preserve feature representations on edges (*Complementary Extraction Module, CEM*) with high-level semantic information (*Pyramid-Semantic Module, PSM*), which leads to high-quality detection. Also, the proposed weighted *Excessiveness Loss (EL)* function helps to minimize false prediction values. Each module leads to significant improvement, which is shown in Table 2. The comparison with 11 state-of-the-art methods shows that our approach outperforms other solutions on all benchmark datasets under three evaluation metrics.

References

[1] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. In *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[2] Zuyao Chen, Qianqian Xu, Runmin Cong, and Qingming Huang. Global context-aware progressive aggregation network for salient object detection. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[3] Ming-Ming Cheng, Fang-Lue Zhang, Niloy J. Mitra, Xiaolei Huang, and Shi-Min Hu. Repfinder: Finding approximately repeated scene elements for image editing. *ACM Transactions on Graphics*, 29(4), 2010.

[4] C. Craye, D. Filliat, and J. Goudou. Environment exploration for object-based visual saliency learning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2303–2309, 2016.

[5] Deng-Ping Fan, Cheng Gong, Yang Cao, Bo Ren, Ming-Ming Cheng, and Ali Borji. Enhanced-alignment measure for binary foreground map evaluation. In *IJCAI*. AAAI Press, 2018.

[6] Mengyang Feng, Huchuan Lu, and Errui Ding. Attentive feedback network for boundary-aware salient object detection. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, 2019.

[7] Y. Gao, M. Wang, D. Tao, R. Ji, and Q. Dai. 3-d object retrieval and recognition with hypergraph analysis. *IEEE Transactions on Image Processing*, 21(9):4290–4303, 2012.

[8] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010.

[9] Li Guanbin and Yu Yizhou. Visual saliency based on multiscale deep features. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[11] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20 (11):1254–1259, 1998.

[12] Zhao Jia-Xing, Liu Jiang-Jiang, Fan Deng-Ping, Cao Yang, Yang Ju-Feng, and Cheng Ming-Ming. Egnet: Edge guidance network for salient object detection. In *International Conference on Computer Vision (ICCV)*, 2019.

[13] Jiang-Jiang Liu, Qibin Hou, Ming-Ming Cheng, Jiashi Feng, and Jianmin Jiang. A simple pooling-based design for real-time salient object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[14] T. Liu, J. Sun, N. Zheng, X. Tang, and H. Shum. Learning to detect a salient object. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[15] V. Mahadevan and N. Vasconcelos. Saliency-based discriminant tracking. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, page 1007:1013, 2009.

[16] Liu Nian, Han Junwei, and Yang Ming-Hsuan. Picanet: Learning pixel-wise contextual attention for saliency detection. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[17] Youwei Pang, Xiaoqi Zhao, Lihe Zhang, and Huchuan Lu. Multi-scale interactive network for salient object detection. In *CVPR*, June 2020.

[18] F. Perazzi, P. Krähenbühl, Y. Pritch, and A. Hornung. Saliency filters: Contrast based filtering for salient region detection. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 733–740, 2012.

[19] Xuebin Qin, Zichen Zhang, Chenyang Huang, Chao Gao, Masood Dehghan, and Martin Jagersand. Basnet: Boundary-aware salient object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[20] Yan Qiong, Xu Li, Shi Jianping, and Jiaya Jia. Hierarchical saliency detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[21] Wang Tiantian, Zhang Lihe, Wang Shuo, Lu Huchuan, Yang Gang, Ruan Xiang, and Borji Ali. Detect globally, refine locally: A novel approach to saliency detection. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[22] Lijun Wang, Huchuan Lu, Yifan Wang, Mengyang Feng, Dong Wang, Baocai Yin, and Xiang Ruan. Learning to detect salient objects with image-level supervision. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, 2017.

[23] Jun Wei, Shuhui Wang, and Qingming Huang. F3net: Fusion, feedback and focus for salient object detection. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[24] Jun Wei, Shuhui Wang, Zhe Wu, Chi Su, Qingming Huang, and Qi Tian. Label decoupling framework for salient object detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[25] Zhe Wu, Li Su, and Qingming Huang. Cascaded partial decoder for fast and accurate salient object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[26] Zhe Wu, Li Su, and Qingming Huang. Stacked cross refinement network for edge-aware salient object detection. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[27] Q. Yan, L. Xu, J. Shi, and J. Jia. Hierarchical saliency detection. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1155–1162, 2013.

[28] Chuan Yang, Lihe Zhang, Huchuan Ruan Xiang Lu, and Ming-Hsuan Yang. Saliency detection via graph-based manifold ranking. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, 2013.

[29] Li Yin, Hou Xiaodi, Koch Christof, M. Rehg James, and L. Yuille Alan. The secrets of salient object segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.