Enhancing Salient Object Segmentation Through Attention

Anuj Pahuja* Avishek Majumder* Anirban Chakraborty R. Venkatesh Babu
Indian Institute of Science, Bangalore, India
anujpahuja13@gmail.com avishek.alex15@gmail.com {anirban,venky}@iisc.ac.in

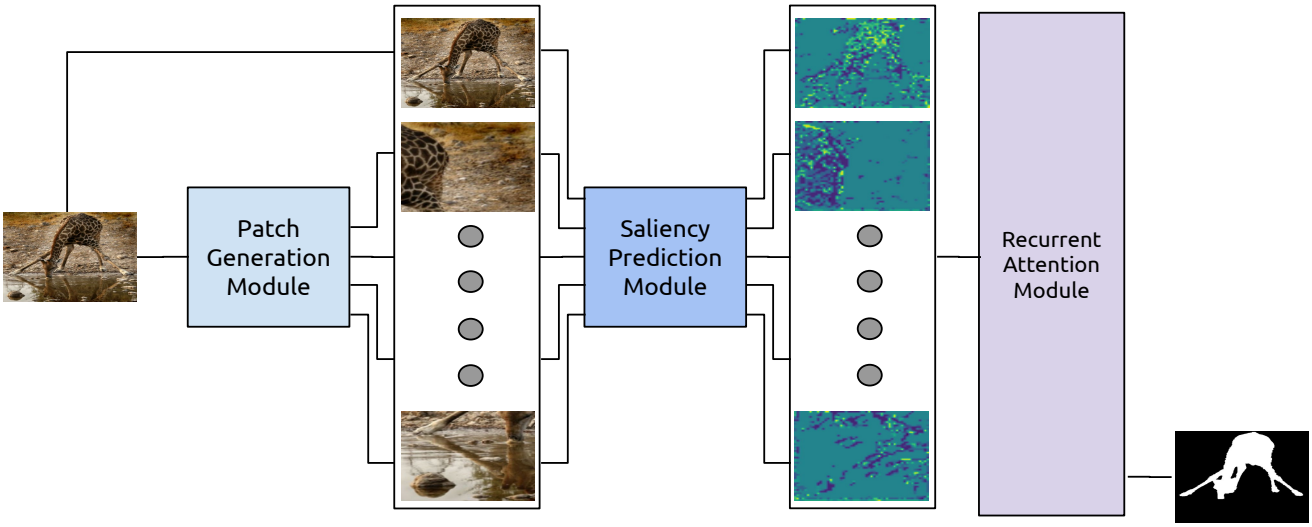

Figure 1: Overview of the proposed approach. An input RGB image goes through three different modules. Patch Generation Module learns to generate image patches in a differentiable manner. The Saliency Prediction Module operates on every patch and generates saliency feature maps. Finally, the Recurrent Attention Module aggregates the bag of features and iteratively refines the complete segmentation map.

Abstract

Segmenting salient objects in an image is an important vision task with ubiquitous applications. The problem becomes more challenging in the presence of a cluttered and textured background, low resolution and/or low contrast images. Even though existing algorithms perform well in segmenting most of the object(s) of interest, they often end up segmenting false positives due to resembling salient objects in the background. In this work, we tackle this problem by iteratively attending to image patches in a recurrent fashion and subsequently enhancing the predicted segmentation mask. Saliency features are estimated independently for every image patch which are further combined using an aggregation strategy based on a Convolutional Gated Recurrent Unit (ConvGRU) network. The proposed approach works in an end-to-end manner, removing background noise and false positives incrementally. Through extensive evaluation on various benchmark datasets, we show superior performance to the existing approaches without any post-processing.

1. Introduction

Saliency is an important aspect of human vision. It is the phenomenon that allows our brain to focus on some parts of a scene more than the rest of it. Thousands of years of evolution has optimized our brain usage by focusing only in the most important regions in our field of view and ignore the rest of it. Indeed, even in computer vision, saliency plays a huge role in many applications, including what humans use it for - compressed representation [16, 20]. Saliency can be exploited to improve agent navigation in the wild [10], image retrieval [5, 14, 17], content based object re-targeting [8, 37], scene parsing [51], object detection and segmentation [13, 32, 34] among many others. Due to its vast applications in vision, saliency prediction is a well established problem with decades of on-going research. De-

*Equal contribution

spite the efforts, the problem still remains open due to factors like cluttered background, multiple instances of non-salient objects, scattered salient regions, low contrast scene, and the definition of saliency varying from application to application.

Salient object detection (SOD) or segmentation is an immediate extension of saliency prediction, as it requires a precise pixel-wise segmentation of the object of interest in the scene. This is a harder task than saliency prediction due to the amount of precision required. We observe that background is the primary reason for poor segmentations. Lack of a well-defined boundary between the salient object and the background can make it very difficult for vision algorithms to segregate objects accurately. Besides often being similar to the foreground object, part of the background can also contribute to saliency, which further affects the segmentation performance due to false predictions. Cluttered or texture-rich background is often the reason why saliency models may focus on the background, failing to segment out the true object of interest. All these challenges are inherently associated to the task of salient object segmentation from images in the wild.

In recent past, Convolutional Neural Networks (CNNs) have shown impressive performance on this task, achieving significant improvements over existing approaches, both in speed and accuracy. Although existing approaches succeed in segmenting out majority of the salient object(s), they often miss out on finer details and/or segment partly salient background regions during the global optimization process. We believe that individually attending to finer image regions or regions separating object and background can refine the overall segmentation mask. Since handcrafting such regions is a very subjective and unscalable approach, we propose to use a learnable module for estimating these region locations. We would also like to incorporate learned features from a region$_k$ for predicting the next region$_{k+1}$, whilst maintaining the spatial context and improving the whole saliency map. This symbiotic relationship can be best exploited using a recurrent network, where different regional features can act as a temporal sequence. Such a strategy could impart important foreground/background distinction to the network along with fine object details that can be aggregated and improved upon iteratively.

We also take inspiration from recent approaches for the task of video object segmentation task. Motion patterns in a scene, specifically the differences between the foreground object motion and background motion may act as an important cue to segregate foreground from background. Moreover, the information flow within a temporal neighborhood often improves the segmentation accuracy especially in cases of occlusion and background clutter. Tokmakov et al. [41] leveraged such temporal dynamics in the video via feature aggregation using a gated recurrent network. Analogously, different regions/patches might emulate this behavior of spatio-temporal perturbations in a single image.

Our Contributions: In this work, we propose an end-to-end salient object detection architecture comprising of three modules that a) learns what image regions to attend, b) effectively aggregates the learned features from regions, and c) incrementally refines the overall segmentation in an interpretable way. Patch Generation Module (PGM) learns and crops the desired regions in a differentiable way, creating a bag of images (including the input image). Saliency Prediction Network (SPN) outputs the saliency features of each image in the bag independently. Recurrent Attention Module (RAM) combines these features using a novel aggregation strategy based on a pair of encoder-decoder Convolutional GRUs. Through our intuitive approach, we achieve state-of-the-art results on challenging SOD datasets.

2. Related Work

Given the ubiquity of the salient object segmentation problem, a lot of approaches have been tried and tested in the past literature. Earlier approaches mostly work on various low-level hand-crafted features in a data-independent setup. Object and background priors [38], global regional contrast [7, 23, 6] and boundary priors [7, 52] are some of the different techniques that have been extensively studied. A comprehensive survey of traditional techniques can be found in [4].

Convolutional Neural Networks have been the status quo for vision tasks in recent years, being present in all state-of-the-art methods for Salient Object Detection as well. DSS [19] is currently the best performing method on many benchmark datasets. It is a VGG-based network, where the authors use connections between the deeper and shallower layers, and fuse them in a way similar to HED [44]. These collective features have the ability to extract necessary high level information, without losing spatial acuity.

In another work by Liu et al. [31], a coarse-to-fine network is used to improve the features progressively. The authors implement a recurrent structure to gradually increase the spatial precision. Instead of using RNNs for interpolation, we use it to segment the common object(s) from the single image patches. Luo et al. [33] use a global segmentation branch, and various sub-branches at different levels to extract local features, which are then combined in a separate layer to refine the global prediction. Wang et al. [43] use a pyramid pooling scheme to extract multi-scale features before the final prediction layer.

Another work that uses recurrent structure in their architecture is RAN [26], where they used one RNN to predict segments from a local patch, and another RNN conditioned on the first one that proposes the next region in the image to focus on. This work is thematically the most similar to ours. We differ in the approach as we do not use a decoder STN

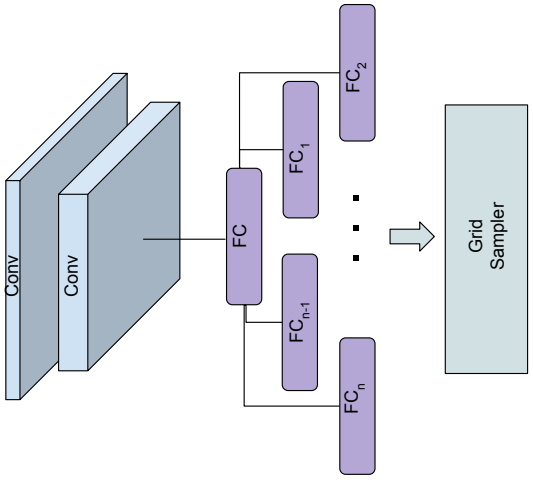

Figure 2: Patch Generation Module (PGM).

at the end to map the attended region back to input image. We also employ Convolutional GRUs (ConvGRU) instead of vanilla RNNs in our attention module which regresses all regions in one step and incorporates the inverse spatial mapping to input image within the module.

ConvGRU is an extension of a Gated Recurrent Unit [9] which was introduced in [2]. A fully convolutional GRU was used for video segmentation by Siam et al. [39]. A ConvGRU has been shown to perform well for a spatially structured task with fewer parameters than a traditional GRU.

In most of the other recent methods [49, 27, 42], we find a common practice to combine the lower and higher level features through convolution layers. Due to the lower level features being very noisy, and containing all edges and texture information, these are often combined with the higher level features and passed through convolution filters to suppress the noise. This preserves the stronger edge information, such as the boundary information of the salient object and the weaker noise signals are suppressed.

3. Method

We use a neural network based architecture comprising of CNNs, GRUs and fully connected layers for our method. The architecture is designed to be modular such that each module can be used independently (Figure 1), demonstrating both simplicity and interpretability. We describe the three modules in this section - a Patch Generation Module (PGM), a Saliency Prediction Module (SPM) and a Recurrent Attention Module (RAM).

3.1. Patch Generation Module (PGM)

PGM takes an RGB image I_0 of dimensions $[H, W, 3]$ as an input and generates N patches per image. It can be considered as a Multi-Spatial Transformer Network (STN) [21] with a shared localization network barring N fully connected layers (for each patch). The localization network is a small neural network with some convolutional and/or fully connected layers (Figure 2). For our experiments, we use 2 convolutional layers, each with 64 kernels of window size 7×7 and 5×5 respectively. Each $Conv$ layer is followed by a max pooling operation with a stride of 2. That is followed by a fully connected layer that outputs a 256-dimensional vector. The output is passed on to N fully connected layers. The N unshared FC layers regress to $4N$ outputs, representing N $[x_1, y_1, x_2, y_2]$ normalized image co-ordinates to crop.

Unlike a conventional spatial transformer network, we do not regress the parameters for a generic affine transformation. To specifically preserve the spatial appearance of the salient object, we use image crops. Image crops have proven to be good 'perturbations' for an iterative refinement task [12]. Even though affine perturbations add more degrees of freedom, they are well suited for an image-level classification task where preserving the spatial structure is less important. We further validated this experimentally and found the supervisory signal to be too weak to train the complete STN.

Spatial context is another important cue for good detection. Without a spatial context, the saliency features generated might not be optimal. Thus, we explicitly enforce $x_2 - x_1 \geq \epsilon$ and $y_2 - y_1 \geq \epsilon$, where ϵ is the percentage of image to crop. This also gives a good initialization to the model due to the lack of direct supervision on the crop parameters. We empirically choose $\epsilon = 0.6$ for our experiments as it is a good trade-off between the amount of zoom we want for a patch and the amount of overlap that could provide a good association of spatial context for our recurrent module.

The generated $4N$ crop parameters are used to crop I_0 and resized to fixed dimensions $[H, W, 3]$ through a differentiable grid sampling layer. The N generated patches are passed along with I_0, the full image, as a batch of $(N + 1)$ images $([I_0, I_1, \ldots, I_N])$ to the next module.

3.2. Saliency Prediction Module (SPM)

SPM is the primary saliency feature extractor in our architecture. We base our SPM network on the segmentation branch of Saliency Unified [25], which is a modified VGG-16 [40] network that achieved impressive results in saliency prediction task. The convolutional part of the original VGG-16 network makes the input image $1/32$ of the original size, which makes the task of spatially localizing the objects imprecise. To overcome this, the final two convolution layers of VGG-16 ($Conv4$ and $Conv5$) are replaced with dilated convolutional layers [48]. Dilated convolutions span a bigger field of view while effectively preserving the spatial resolution and maintaining the same

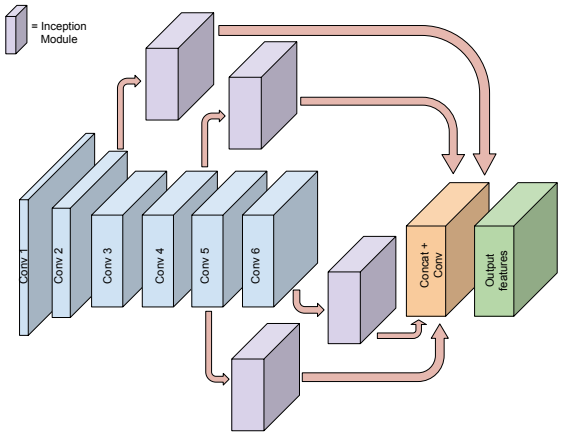

Figure 3: Saliency Prediction Module

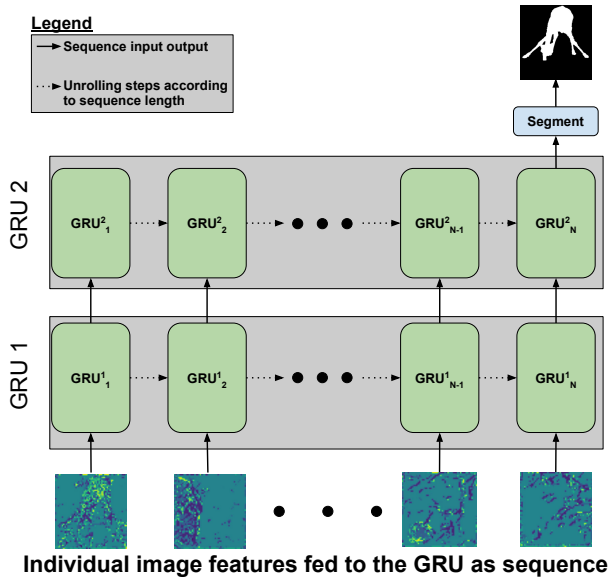

Figure 4: Recurrent Attention Module (RAM). The ConvGRUs in RAM have been unrolled for visualization.

number of parameters. Unlike [25], we introduce only one additional convolution layer abstraction after the $Conv5$ layer. Inception modules are used to fuse features from different layers at multiple scales. The network finally outputs a 512-dimensional feature blob at $1/8$ resolution of the original image.

A batch of $(N+1)$ 512-dimensional feature maps $([F_0, F_1, \ldots, F_N])$, corresponding to the $(N+1)$ image patches is generated and then passed on to the next module.

3.3. Recurrent Attention Module (RAM)

The task of RAM is to aggregate the learned bag of features from the previous module in a semantically coherent manner and improve the final segmentation map. RAM is implemented as two Convolutional GRUs [3] in an encoder-decoder style. A ConvGRU has fewer parameters than a traditional GRU and performs well on spatially structured data.

A ConvGRU is comprised of convolutional layers as opposed to fully connected layers in a traditional GRU. The function of hidden units is identical to the normal GRU and can be represented as:

$$z_t = \sigma_g(W_z * x_t + U_z * h_{t-1} + b_z), \quad (1)$$

$$r_t = \sigma_g(W_r * x_t + U_r * h_{t-1} + b_r), \quad (2)$$

$$h_t = (1-z_t)\odot h_{t-1} + z_t \odot \sigma_h(W_h * x_t + U_h * (r_t \odot h_{t-1}) + b_h), \quad (3)$$

where $*$ represents the convolution operation, z_t is the update gate and r_t is the reset gate. Unlike LSTM [18], GRU has only two gates and no internal state. W, U, b are the training parameters of the GRU, and x_t and h_t are input and output activation blobs respectively.

Figure 4 depicts a simplistic view and workflow of RAM. We use convolution filters of size 5×5 for the GRUs, slightly larger than the size of filters in SPM (3×3). This helps the ConvGRUs to learn a distinction between background and foreground regions at $1/8$ scale as a 5×5 filter will mostly see the background and foreground together. While the architecture is not exactly an encoder-decoder, 2 GRUs are used to make the learned representations co-dependent similar to what an encoder-decoder setup does. Since the 2^{nd} GRU directly outputs the saliency maps, it acts like a decoder. Furthermore, this specific architecture allows us to avoid the need for an inverse spatial transformer as the one used in [26]. We finish the network with a 1×1 convolution filter to scale the output of the decoder GRU to get the final pixel-wise predictions.

We order the incoming $(N+1)$ feature maps such that the input image feature map is fed first into the module (Figures 1, 4). This is done to ensure that RAM gets to learn the complete spatial context first. The task is then reduced to learning spatial associations of incoming feature maps and predict a saliency map for each recurrent step. Every feature map F_k, where $k \in [0, N]$ is iteratively fed into RAM where the decoder GRU outputs a saliency map $Pred_k$. We enforce the iterative improvement criterion by weighing the loss for $Pred_{k+1}$ more than the loss for $Pred_k$. Refer to next section (Sec 3.4) for more details.

3.4. Implementation

The initial layers of Saliency Prediction Module are initialized using pre-trained ImageNet weights of VGG-16. Rest of the its layers are initialized with Xavier initialization scheme [15]. PGM and RAM are also both initialized using the same scheme. The training is carried out in a stepwise manner. We first train SPM for object segmentation

using the training datasets. Since the proposed SPM (Figure 3) only outputs a feature blob, we add 3 convolutional layers after it that decode and predict a saliency map. We use this prediction to fully train SPM. We minimize the loss function -

$$L = \lambda_1 Loss_{CE} + \lambda_2 Loss_{IOU}, \tag{4}$$

where $Loss_{CE}$ is the standard sigmoid cross-entropy loss and $Loss_{IOU}$ is an IOU-based loss described in [36]. λ_1 and λ_2 are kept as 1.0. We use a batch size of 10 and optimize it using Adam [24] with a learning rate of $1e^{-5}$. We decrease the learning rate step-wise based on the validation performance. We train it for 10 epochs.

We then take out the added convolutional layers from SPM and plug the 512-dimensional feature blob to RAM. We also plug in PGM by placing it before SPM so that it outputs $(N+1)$ patches for input image I_0. These are then passed on to SPM as a batch. We use a batch size of 1 in this complete setup. We freeze SPM layers during the training. We optimize the whole network on an exponentially weighted loss on RAM's outputs-

$$L = \frac{1}{k^{N+1}} \sum_{i=0}^{N} k^{i+1} Loss_{CE_i}, \tag{5}$$

where $i \in [0, N]$ and value of k is chosen to be 2. $Loss_{CE_i}$ is the sigmoid cross-entropy loss between $Pred_i$ and ground truth label. The described loss gives more weight to every $i+1^{th}$ prediction compared to i^{th} as described in Section 3.3. Adam optimizer is used with a learning rate of $1e^{-4}$ for PGM and $5e^{-6}$ for the RAM. This setup is trained for 10 epochs. We further fine-tune the complete network end-to-end for 5 epochs. We use $N = 4$ for our experiments.

During testing, we adopt the same approach as our training mechanism. For a single image, we get $(N+1)$ predictions from RAM. We resize the predictions to the original image size using bilinear interpolation. We use $Pred_5$ (prediction after seeing all patches) for our final performance evaluations.

4. Experiments

4.1. Datasets

We use the Pascal VOC-2012 [11] and MSRA10K [1] datasets for training our model and ECSSD [45], HKU-IS [28], DUT-OMRON [46] and PASCAL-S [30] for evaluation. 300 random images from the training dataset are used for validation.

PASCAL-VOC 2012. Pascal-VOC dataset has semantic segmentations of 20 object classes. We convert these into binary segmentation maps and use for our task. This dataset has scenes containing complex background and multiple salient objects in the scene.

MSRA10K. This dataset contains 10000 image-label pairs of salient objects in varied scenes.

DUT-OMRON. This dataset consists of 5168 high quality images featuring one or more salient objects and relatively complex background.

ECSSD. ECSSD contains 1000 images of natural scenes, often comprising semantically meaningful and complex structures to segment.

HKU-IS. This dataset contains 4447 images with high-quality object annotations. Many images include multiple disconnected objects or objects touching the image boundary.

PASCAL-S. PASCAL-S is the testing subset of 850 images from the PASCAL VOC dataset.

4.2. Evaluation metrics

One of the evaluation metrics for the image dataset is Mean Absolute Error or MAE. The MAE computes the average pixel percent error. It is computed as:

$$MAE = \frac{1}{M \times N} \sum_{i,j} |G(x_{ij}) - P(x_{ij})| \tag{6}$$

where x_{ij} is the input image of width and height M and N, $G(\cdot)$ and $P(\cdot)$ are the ground truth mask and predicted mask of the input image respectively.

The other metric is F_β score, which is a weighted ratio of the Recall and Precisions. The recall is computed as $TP/(TP+FN)$ and the precision is computed as $TP/(TP+FP)$. Here TP, FP and FN hold their usual meanings of True Positive, False Positive and False Negative predictions respectively. Then F_β score can then be computed as:

$$F_\beta = \frac{(1+\beta^2) \times Precision \times Recall}{\beta^2 \times Precision + Recall}, \tag{7}$$

where $\beta^2 = 0.3$ to weigh precision more than recall rate [47]. This is done in accordance with the number of negative examples (non-salient pixels) typically being much bigger than the number of positive examples (salient object pixels) while evaluating SOD models. Hence, F_β score is a good indicator of an algorithm's detection performance [4]. We compare against the maximum F_β scores of all other approaches.

4.3. Comparison with existing approaches

In Table 1, we compare the quantitative performance of various state-of-the-art methods with ours based on the aforementioned evaluation criteria. We compare against DSS [19], DCL [29], DHS [31], Amulet [49], SRM [43],

Methods	DUT-OMRON		HKU-IS		ECSSD		PASCAL-S	
	MAE	$max.F_\beta$	MAE	$max.F_\beta$	MAE	$max.F_\beta$	MAE	$max.F_\beta$
BSCA [35]	-	-	0.175	0.719	0.182	0.758	0.223	0.667
DRFI [22]	-	-	0.145	0.777	0.164	0.786	0.207	0.698
RFCN [42]	-	-	0.079	0.892	0.107	0.890	0.118	0.837
DHS [31]	-	-	0.053	0.890	0.059	0.907	0.094	0.829
DCL [29]	0.084	0.733	0.054	0.892	-	-	0.113	0.815
UCF [50]	0.080	0.726	0.074	0.886	0.078	0.911	0.126	0.828
Amulet [49]	0.074	**0.741**	0.052	0.895	0.059	0.915	0.098	**0.837**
SRM [43]	0.069	0.707	**0.046**	0.874	**0.056**	0.892	-	-
NLDF [33]	0.085	0.724	0.060	0.874	0.075	0.886	0.108	0.804
DSS* [19]	**0.068**	0.736	**0.039**	**0.913**	**0.052**	**0.916**	**0.080**	0.830
Ours	**0.066**	**0.751**	0.054	**0.915**	0.063	**0.921**	**0.083**	**0.846**

Table 1: Quantitative comparison with other state-of-the-art methods on various datasets. Top two results are in **bold** numbers.

UCF [50], RFCN [42] and two non-deep methods - DRFI [22] and BSCA [35].

Our method consistently gets top F_β scores, implying a greater precision in the predicted map. The high precision showcases its effectiveness on suppressing false positives in cluttered backgrounds and partly salient objects. Metrics of other methods have either been reported by the respective authors or have been computed by us using available predictions/weights. For a fair comparison, we use the scores obtained without post-processing for all methods.

In Figure 5, we compare the qualitative results of the aforementioned methods with ours. We show results for a set of images with varying difficulties:

Cluttered background. Row 1 contains a textured background, making algorithms prone to background noise.

Shadows in background. Rows 2 and 4 include images with object shadows. While every method performs well on Row 4, our method is able to suppress much of the 'shadow saliency' in background of Row 2 that is easily thresholded during inference.

Low contrast. Row 7 contains an image with low contrast between object and background. We are able to segment better with fewer false positives than others.

Multiple Objects. Rows 5, 6, 7 and 10 contain multiple foreground objects. 6 and 10 contains multiple salient objects whereas 5 and 7 have only a single salient object. Our algorithm performs very well in these scenarios.

Complex foreground. Row 12 contains a complex salient object where most other algorithms create holes in the prediction. Our method is able to better understand the regional and global context.

Object within an object. Row 9 contains an interesting image which contains an image of a bird (with sharp contrast) within a poster (salient object). Our method is the only method that does not fail by trying to segregating these two objects.

*DSS also employs a CRF post-processing step.

Module	MAE (↓)	$max.F_\beta$ (↑)
SPM	0.080	0.870
SPM + RAM (Epoch 2)	0.0692	0.9193
PGM + SPM + RAM (Epoch 2)	0.0662	0.9196
SPM + RAM (Epoch 5)	0.0661	0.9205
PGM + SPM + RAM (Epoch 5)	0.0623	0.9204

Table 2: Incremental performance gains for different modules on ECSSD

4.4. Method Analysis

We analyze our network's performance by evaluating component-wise and step-wise results. The results shed light on our design choices and incremental gains. The evaluation metrics have been described in Section 4.2.

To better quantify the role of every module in our architecture, we do a component-wise performance analysis on ECSSD dataset (Table 2). We first compute the results using just SPM with added convolutional layers as described in Section 3.4. We can easily evaluate its performance independently since it is trained first. We then plug in RAM to SPM's 512-dimensional features and do an inference on trained SPM and RAM by only evaluating on $Pred_0$. We see an immediate performance boost with this setting. While this could just be attributed to increase in model complexity, we argue that the initial setup with SPM + 3 layers also has similar complexity. This observation shows that RAM not only predicts a better output for every $Pred_{k+1}(k \in [0, N-1])$, but also improves $Pred_i(i \in [0, k])$ in the process. We do a final evaluation by allowing a forward pass through all three modules.

In a recurrent network, we should ideally see perfor-

Figure 5: Qualitative comparison with various state-of-the-art approaches on some challenging images from ECSSD [45]. Most of the images where we perform better are the ones where global spatial context is important to distinguish between foreground and background.

mance improvements for every iterative step. To verify this, we evaluate the predicted saliency maps computed at every k^{th} step and compare the results in Table 3. We evaluate results after 2^{nd} and 5^{th}(final) epoch. For both the readings, we observe that the F_β scores do not vary much across the $(N+1)$ predictions. Higher F_β does not imply a lower

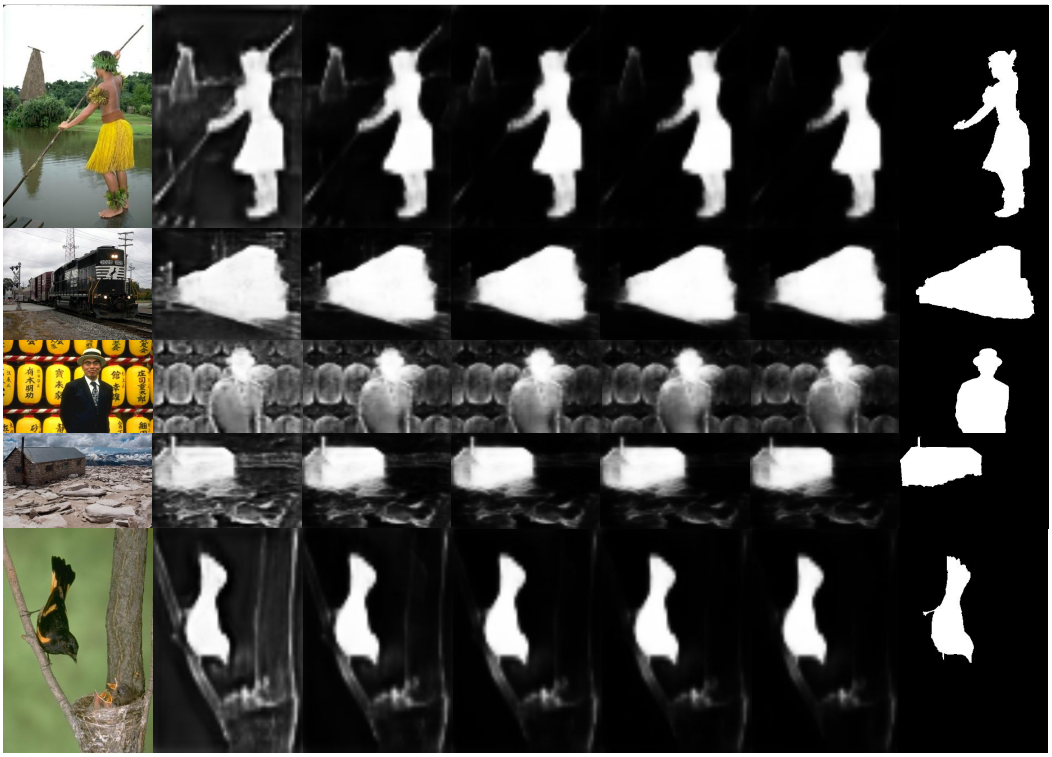

Figure 6: Recurrent step-wise qualitative performance analysis. We observe that $Pred_1$ captures a lot of 'pseudo' salient objects. As we go from left to right, we see a clear reduction in number of false positives that arise from background.

$Pred_k$ (Epoch 2)	MAE (↓)	F_β (↑)
$k = 1$	0.0692	0.9193
$k = 2$	0.0687	0.9196
$k = 3$	0.0666	0.9197
$k = 4$	0.0669	0.9197
$k = 5$	0.0662	0.9196
$Pred_k$ (Epoch 5)	MAE (↓)	F_β (↑)
$k = 1$	0.0661	0.9205
$k = 2$	0.0637	0.9210
$k = 3$	0.0629	0.9208
$k = 4$	0.0625	0.9206
$k = 5$	0.0623	0.9204

Table 3: Quantitative performance comparison of N predictions from the Recurrent Attention Module on ECSSD

MAE [4]. Often, a decrease in MAE also leads to a decrease in F_β. Thus, an important observation is the gradual decrease in MAE as we increase k. A decrease in MAE while maintaining the F_β scores affirms that RAM reduces the false positives incrementally without losing precision.

Qualitatively, we observed that visible differences in saliency maps are more noticeable during the initial epochs. Hence, we show the $(N + 1)$ predicted maps after the 1^{st} epoch in Figure 6. We can clearly notice the suppression of false positives in background for every subsequent prediction.

5. Conclusion

We present an intuitive, scalable and effective approach for detecting salient objects in a scene. Our approach is modular, resulting in interpretable results. We propose a Patch Generation Module, a Saliency Prediction Module and a Recurrent Attention Module that work in tandem to improve overall object segmentation by generating image patches, their corresponding feature maps and effectively aggregating them. Through our quantitative and qualitative performance on benchmark datasets, we show the importance of region-wise attention in saliency prediction. An easy and important extension to our work could be a dynamic improvement of predictions based on the number of allowed patches. This can reduce the inference time significantly for an accuracy trade-off. In future, we would also like to test our method's effectiveness on the task of video object segmentation in an unsupervised setting.

References

[1] Radhakrishna Achanta, Sheila Hemami, Francisco Estrada, and Sabine Susstrunk. Frequency-tuned salient region detection. In *CVPR*, 2009. 5

[2] Nicolas Ballas, Li Yao, Chris Pal, and Aaron Courville. Delving deeper into convolutional networks for learning video representations. *arXiv preprint arXiv:1511.06432*, 2015. 3

[3] Nicolas Ballas, Li Yao, Chris Pal, and Aaron C. Courville. Delving deeper into convolutional networks for learning video representations. *arXiv preprint arXiv:1511.06432*, 2015. 4

[4] Ali Borji, Ming-Ming Cheng, Huaizu Jiang, and Jia Li. Salient object detection: A benchmark. *IEEE Transactions on Image Processing*, 24(12):5706–5722, 2015. 2, 5, 8

[5] Ming-Ming Cheng, Qi-Bin Hou, Song-Hai Zhang, and Paul L Rosin. Intelligent visual media processing: When graphics meets vision. *Journal of Computer Science and Technology*, 32(1):110–121, 2017. 1

[6] Ming-Ming Cheng, Niloy J Mitra, Xiaolei Huang, Philip HS Torr, and Shi-Min Hu. Global contrast based salient region detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):569–582, 2015. 2

[7] Ming-Ming Cheng, Jonathan Warrell, Wen-Yan Lin, Shuai Zheng, Vibhav Vineet, and Nigel Crook. Efficient salient region detection with soft image abstraction. In *ICCV*, 2013. 2

[8] Ming-Ming Cheng, Fang-Lue Zhang, Niloy J Mitra, Xiaolei Huang, and Shi-Min Hu. Repfinder: Finding approximately repeated scene elements for image editing. In *ACM Transactions on Graphics (TOG)*, volume 29, page 83. ACM, 2010. 1

[9] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014. 3

[10] Celine Craye, David Filliat, and Jean-François Goudou. Environment exploration for object-based visual saliency learning. In *ICRA*, 2016. 1

[11] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html. 5

[12] Jianlong Fu, Heliang Zheng, and Tao Mei. Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition. In *CVPR*, 2017. 3

[13] Dashan Gao and Nuno Vasconcelos. Discriminant saliency for visual recognition from cluttered scenes. In *NIPS*, 2005. 1

[14] Yue Gao, Meng Wang, Dacheng Tao, Rongrong Ji, and Qionghai Dai. 3-d object retrieval and recognition with hypergraph analysis. *IEEE Transactions on Image Processing*, 21(9):4290–4303, 2012. 1

[15] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *ICAIS*, 2010. 4

[16] Chenlei Guo and Liming Zhang. A novel multiresolution spatiotemporal saliency detection model and its applications in image and video compression. *IEEE Transactions on Image Processing*, 19(1):185–198, 2010. 1

[17] Junfeng He, Jinyuan Feng, Xianglong Liu, Tao Cheng, Tai-Hsu Lin, Hyunjin Chung, and Shih-Fu Chang. Mobile product search with bag of hash bits and boundary reranking. In *CVPR*, 2012. 1

[18] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 4

[19] Qibin Hou, Ming-Ming Cheng, Xiao-Wei Hu, Ali Borji, Zhuowen Tu, and Philip Torr. Deeply supervised salient object detection with short connections. *CVPR*, 2017. 2, 5, 6

[20] Laurent Itti. Automatic foveation for video compression using a neurobiological model of visual attention. *IEEE Transactions on Image Processing*, 13(10):1304–1318, 2004. 1

[21] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *NIPS*, 2015. 3

[22] Huaizu Jiang, Jingdong Wang, Zejian Yuan, Yang Wu, Nanning Zheng, and Shipeng Li. Salient object detection: A discriminative regional feature integration approach. In *CVPR*, 2013. 6

[23] Zhuolin Jiang and Larry S Davis. Submodular salient region detection. In *CVPR*, 2013. 2

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[25] Srinivas SS Kruthiventi, Vennela Gudisa, Jaley H Dholakiya, and R Venkatesh Babu. Saliency unified: A deep architecture for simultaneous eye fixation prediction and salient object segmentation. In *CVPR*, 2016. 3, 4

[26] Jason Kuen, Zhenhua Wang, and Gang Wang. Recurrent attentional networks for saliency detection. In *CVPR*, 2016. 2, 4

[27] Guanbin Li and Yizhou Yu. Visual saliency based on multiscale deep features. In *CVPR*, 2015. 3

[28] Guanbin Li and Yizhou Yu. Visual saliency based on multiscale deep features. In *CVPR*, 2015. 5

[29] Guanbin Li and Yizhou Yu. Deep contrast learning for salient object detection. In *CVPR*, 2016. 5, 6

[30] Yin Li, Xiaodi Hou, Christof Koch, James M Rehg, and Alan L Yuille. The secrets of salient object segmentation. In *CVPR*, 2014. 5

[31] Nian Liu and Junwei Han. Dhsnet: Deep hierarchical saliency network for salient object detection. In *CVPR*, 2016. 2, 5, 6

[32] Nian Liu, Junwei Han, Dingwen Zhang, Shifeng Wen, and Tianming Liu. Predicting eye fixations using convolutional neural networks. In *CVPR*, 2015. 1

[33] Zhiming Luo, Akshaya Mishra, Andrew Achkar, Justin Eichel, Shaozi Li, and Pierre-Marc Jodoin. Non-local deep features for salient object detection. In *CVPR*, 2017. 2, 6

[34] Junting Pan, Elisa Sayrol, Xavier Giro-i Nieto, Kevin McGuinness, and Noel E O'Connor. Shallow and deep convolutional networks for saliency prediction. In *CVPR*, 2016. 1

[35] Yao Qin, Huchuan Lu, Yiqun Xu, and He Wang. Saliency detection via cellular automata. In *CVPR*, 2015. 6

[36] Md Atiqur Rahman and Yang Wang. Optimizing intersection-over-union in deep neural networks for image segmentation. In *ISVC*, 2016. 5

[37] Vidya Setlur, Saeko Takagi, Ramesh Raskar, Michael Gleicher, and Bruce Gooch. Automatic image retargeting. In *ICMUM*, 2005. 1

[38] Xiaohui Shen and Ying Wu. A unified approach to salient object detection via low rank matrix recovery. In *CVPR*, 2012. 2

[39] Mennatullah Siam, Sepehr Valipour, Martin Jagersand, and Nilanjan Ray. Convolutional gated recurrent networks for video segmentation. In *ICIP*, 2017. 3

[40] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 3

[41] Pavel Tokmakov, Karteek Alahari, and Cordelia Schmid. Learning video object segmentation with visual memory. *CVPR*, 2017. 2

[42] Linzhao Wang, Lijun Wang, Huchuan Lu, Pingping Zhang, and Xiang Ruan. Saliency detection with recurrent fully convolutional networks. In *European Conference on Computer Vision*, pages 825–841. Springer, 2016. 3, 6

[43] Tiantian Wang, Ali Borji, Lihe Zhang, Pingping Zhang, and Huchuan Lu. A stagewise refinement model for detecting salient objects in images. In *CVPR*, 2017. 2, 5, 6

[44] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *ICCV*, 2015. 2

[45] Qiong Yan, Li Xu, Jianping Shi, and Jiaya Jia. Hierarchical saliency detection. In *CVPR*, 2013. 5, 7

[46] Chuan Yang, Lihe Zhang, Huchuan Lu, Xiang Ruan, and Ming-Hsuan Yang. Saliency detection via graph-based manifold ranking. In *CVPR*, 2013. 5

[47] Chuan Yang, Lihe Zhang, Huchuan Lu, Xiang Ruan, and Ming-Hsuan Yang. Saliency detection via graph-based manifold ranking. In *CVPR*, 2013. 5

[48] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016. 3

[49] Pingping Zhang, Dong Wang, Huchuan Lu, Hongyu Wang, and Xiang Ruan. Amulet: Aggregating multi-level convolutional features for salient object detection. In *CVPR*, 2017. 3, 5, 6

[50] Pingping Zhang, Dong Wang, Huchuan Lu, Hongyu Wang, and Baocai Yin. Learning uncertain convolutional features for accurate saliency detection. In *ICCV*, 2017. 6

[51] Rui Zhao, Wanli Ouyang, Hongsheng Li, and Xiaogang Wang. Saliency detection by multi-context deep learning. In *CVPR*, 2015. 1

[52] Wangjiang Zhu, Shuang Liang, Yichen Wei, and Jian Sun. Saliency optimization from robust background detection. In *ICCV*, 2014. 2