# QUICKSAL: A small and sparse visual saliency model for efficient inference in resource constrained hardware

Vignesh Ramanathan[1], Pritesh Dwivedi[1], Bharath Katabathuni[1,2],
Anirban Chakraborty[1], Chetan Singh Thakur[1]
[1]Indian Institute of Science, [2]Archeron Group
{vigneshr, priteshd, katabathunib, anirban, csthakur}@iisc.ac.in, bharath@archerongroup.com

## Abstract

*Visual saliency is an important problem in the field of cognitive science and computer vision with applications such as surveillance, adaptive compressing, detecting unknown objects and scene understanding. In this paper, we propose a small and sparse neural network model for performing salient object segmentation that is suitable for use in mobile and embedded applications. Our model is built using depthwise separable convolutions and bottleneck inverted residuals which have been proven to perform very memory efficient inference and can be easily implemented using standard functions available in all deep learning frameworks. The multiscale features extracted along the layers with deep residuals allow our network to learn high quality saliency maps. We present the quantitative results of our QUICKSAL model with multiple levels of model sparsity ranging from 0% to ∼96%, with the non-zero parameter count varying from ∼3.3M to ∼0.14M respectively - on publicly available benchmark datasets - showing that our highly constrained approach is comparable to other state-of-the-art approaches (parameter count ∼35M). We also present qualitative results on camouflage images and show that our model can successfully distinguish between the salient and non-salient parts even when both seem blended together.*

## 1. Introduction

In higher organisms, the eye is a complex optical system which focuses the light from the surrounding environment through an adjustable assembly of lenses to form an inverted image on the retina. This image gets translated into electrical neural impulses which can travel through the optic nerve to the visual cortex and other areas of the brain to create visual perception. Even the most sophisticated brains cannot perform real time, parallel processing of the entire visual field, due to the high computational cost involved with processing information at such a large scale [4, 53]. The nervous system of organisms have evolved to filter this overwhelming amount of information and process only a subset of it. Two different approaches exist to implement this bottleneck. The first, top down attention, is controlled by the organism itself and biases attention based on the organisms internal state and goals. The second mechanism, bottom up attention, is based on different parts of a visual scene having different instantaneous saliency values.

Recent advances in computer vision use increasingly large deep neural networks to achieve state of the art results in various vision tasks such as image classification [50, 15, 21], segmentation [42, 40], detection [46, 47, 33]. Over-parameterization is a widely-recognized property of deep neural networks [9, 2], which results in high computational cost and memory footprint during inference. The model size, memory footprint, computational complexity and power usage are major factors to be considered when using deep neural networks for embedded and mobile applications. It is not feasible to directly implement these models in hardware, which are designed for real world applications such as robotics and autonomous driving.

To address this issue, many methods have been proposed such as low-rank approximation of weights [9, 24], weight quantization [7, 45], knowledge distillation [16, 48], network pruning [13, 27] and neural architecture search [66, 29, 37, 20]. Neural architecture search methods are computationally intensive and result in complex architectures. We choose to keep our model design intuitive and hence choose to adapt the principles stated in [49]. The depthwise separable convolution splits a standard convolution into depthwise convolution and pointwise convolution, thereby achieving benefits similar to low-rank approximation. This convolutional module is perfectly suited for use in mobile applications, because it significantly reduces the memory footprint during inference by never fully materializing large intermediate tensors [49]. This reduces the need for main memory access in many embedded systems, that have small amounts of very fast cache memory, thereby providing gains in computation time and power usage. Literature suggests that the memory fetching operation is much more power hungry than a basic compute operation [17].

The least significant weights of the network are pruned out using an iterative algorithm [12, 10] that masks them out, thereby further reducing the RAM,ROM and power requirements of our model. Our model can be easily implemented using any modern deep learning framework [43, 1].

In this paper, we present a small, sparse and efficient neural network model for salient object segmentation that is suitable for use in embedded and mobile applications. Section 2 reviews prior work in visual saliency. Section 3 describes our network architecture and its building blocks in detail. Section 4 describes the pruning strategy. Section 5 describes the implementation details. Section 6 describes the evaluation metrics and experimental results on benchmark datasets. Section 7 closes with a conclusion.

## 2. Related Work

Visual saliency models have shown significant progress over the last few years. Earlier approaches for visual saliency involved using various low-level features such as colors, intensity and orientation to generate coarse saliency maps [19]. Other approaches such as [32, 62, 35, 6] combine local, regional and global contrast-based features to detect salient parts of image. A neuromorphic visual saliency algorithm is implemented on digital hardware using stochastic computation(SC) with very low power and small area [52]. The SC hardware implementation of convolution filters is done using SC with simple logic gates. [39] demonstrate a real-time implementation of a proto-object based neuromorphic visual saliency model [38] on an embedded processing board. In [54], an undirected graph is created with all the image pixels as nodes, and the edges between nodes being weighted by colour/intensity differences (absolute gradients). A minimum spanning tree is constructed by sequentially removing edges with large weights, which is post processed to generate the saliency maps. A comprehensive survey on traditional techniques used to solve visual saliency is presented in [3].

Compared to the traditional methods that use hand-crafted features, CNN-based methods extract highly abstract features from images to achieve state-of-the-art results in various computer vision tasks, including salient object detection. The bottom up approach for visual saliency is implemented by training deep neural networks to either predict eye fixation maps [23, 31] or perform salient object segmentation [25, 56, 65] or both [22]. DSS [18] achieves state-of-the-art results in salient object segmentation across various benchmark datasets. Their model is a VGG/RESNET-based network, with multiple shortcut connections across layers which are fused in a way similar to HED [59]. By combining rich multi-scale feature maps from each layer, DSS is able to generate high quality saliency maps. DHSnet [30], uses a similar architecture where saliency maps are generated at various scales and the model is trained using binary cross-entropy loss between the generated saliency maps and ground truths, at various scales. [34] uses a grid like architecture, with initial layers using kernels of small size to capture lower level features and deeper layers using kernels of large size to capture global context information.These features are then combined in a separate layer to refine the final prediction. [58] uses a pyramid pooling scheme before the final prediction layer to extract multi-scale features for saliency map prediction. In [41], a Patch Generation Module, a Saliency Prediction Module and a Recurrent Attention Module are proposed that work in tandem to generate image patches, their corresponding feature maps and aggregate them to generate the saliency maps.

## 3. Network Architecture

We will first discuss about the building blocks of our model and then describe in detail the network architecture used to perform salient object segmentation.

### 3.1. Depthwise Separable Convolutions

Depthwise separable convolution is a form of factorized convolution, which factorizes a standard convolution into a depthwise convolution, where a single filter is applied to each channel and a $1\times1$ convolution called a pointwise convolution, which linearly combines the outputs of the depthwise convolution. Figure 1 visualizes the depthwise seperable convolution operation. A standard convolution takes as input a tensor of size $h_i \times w_i \times d_i$, and applies a convolutional kernel $K \in^{k\times k\times d_i\times d_j}$ to produce an output tensor of size $h_i \times w_i \times d_j$. The computational cost of a standard convolutional operation is given as,

$$h \cdot w \cdot d_i \cdot d_j \cdot k^2 \tag{1}$$

where for zero padding and stride one, $h = h_i - k + 1$ and $w = w_i - k + 1$. The computational cost involved with performing depthwise seperable convolution is given as,

$$h \cdot w \cdot d_i \cdot k^2 + h \cdot w \cdot d_i \cdot d_j \tag{2}$$

The improvement in computational performance can be calculated as,

$$\begin{aligned} & \frac{h \cdot w \cdot d_i \cdot k^2 + h \cdot w \cdot d_i \cdot d_j}{h \cdot w \cdot d_i \cdot d_j \cdot k^2} \\ = \quad & \frac{h \cdot w \cdot d_i \cdot (k^2 + d_j)}{h \cdot w \cdot d_i \cdot d_j \cdot k^2} \\ = \quad & \frac{1}{d_j} + \frac{1}{k^2} \end{aligned} \tag{3}$$

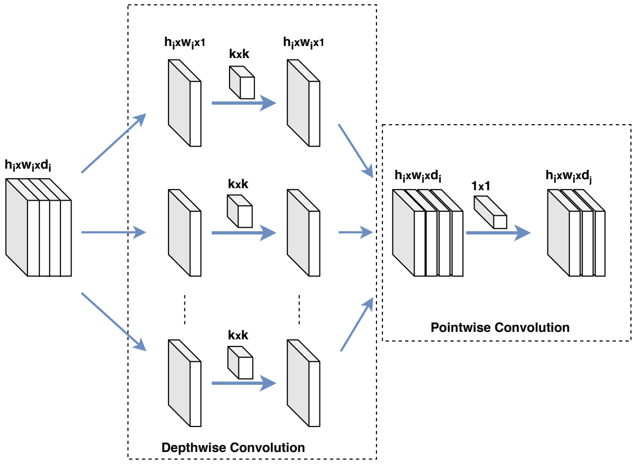


Figure 1. Depthwise seperable convolution factorized into depthwise convolution, which applies a filter to each input channel and a pointwise $1 \times 1$ convolution to linearly combine the output of the depthwise layer.

## 3.2. Bottleneck Inverted Residual Block

Consider a deep neural network consisting of n layers each of which has an activation tensor size $h_i \times w_i \times d_i$. For an input set of real images, the set of layer activations form a manifold of interest [49]. It has been hypothesized that manifolds of interest in neural networks could be embedded in low-dimensional subspaces. This assumption lets us insert linear bottleneck layers in between convolutional blocks. It is important to use linear layers in order to prevent non-linearities from destroying the information. [11, 49] have shown empirical evidence to support this statement. The bottlenecks contain the necessary information in a low-dimensional subspace and the expansion layers exist for the application of the non-linearities. Figure 2 shows the bottleneck inverted residual block.

## 3.3. Inception Blocks With Deep Residuals

Salient object segmentation involves detecting the closed contour which represents the object of interest. This means we require both the object level semantics and low level features such as edges, patterns, contrast etc. Neural networks are known to learn increasing complex abstractions along the layers. [14] have shown that information of interest for pixel-level tasks is spread across all the layers of a convolutional network. [25, 65] have observed that saliency can be captured better when semantics are considered across multiple scales by upsampling and downsampling image patches. As in [5, 51], we use multi-scale dilated convolutional blocks to capture this information. Dilated convolutions requires fewer parameters to cover the desired scale. These multi-scale dilated convolutional blocks are implemented using depth wise convolution and bottleneck inverted residuals.

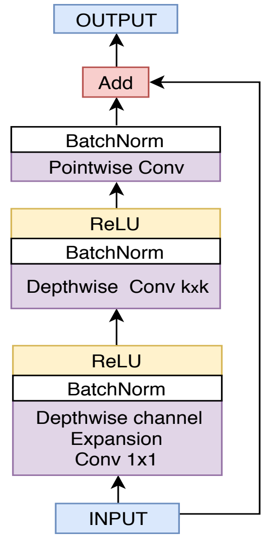


Figure 2. Bottleneck inverted residual block where ReLU function is applied to the output of expansion layers, and pointwise convolution embedding the feature vector to a low dimension space.

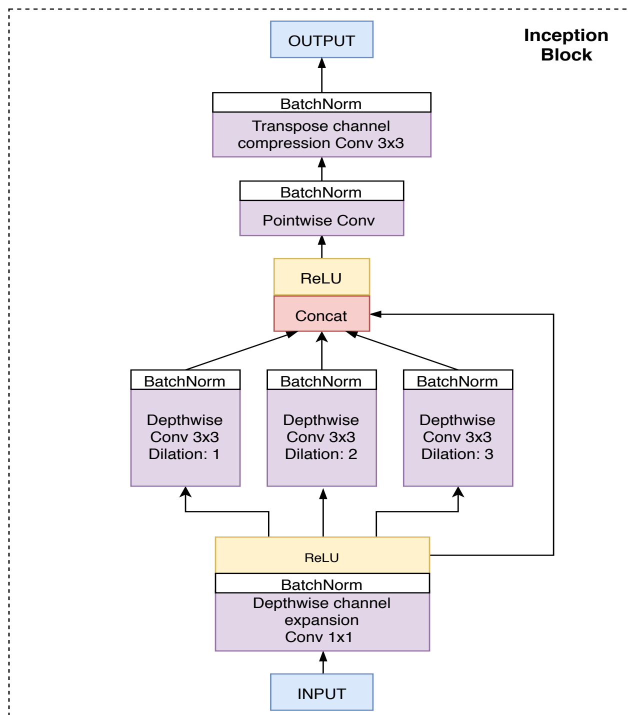


Figure 3. The inception block comprised of multi-scale convolutions. The effective field of view of the kernel operations are $1 \times 1$, $3 \times 3$, $5 \times 5$, $7 \times 7$. The $3 \times 3$, $5 \times 5$, $7 \times 7$ convolutions are implemented using dilated convolutions to maintain the same parameter count across all of them.

## 3.4. Detailed Architecture

Figure 4 shows the network architecture in detail. The network consists of multiple residual connections across various scales which combines the low level features with

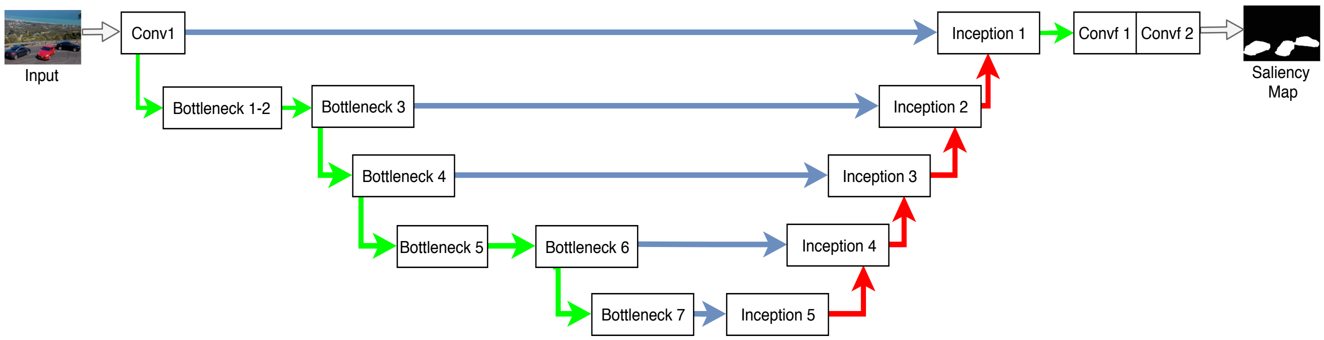


Figure 4. QUICKSAL Model

| Input | Operator | $t$ | $c$ | $n$ | $s$ |
|---|---|---|---|---|---|
| $224^2 \times 3$ | Conv1 | - | 32 | 1 | 2 |
| $112^2 \times 32$ | Bottleneck1 | 1 | 16 | 1 | 1 |
| $112^2 \times 16$ | Bottleneck2 | 6 | 24 | 2 | 2 |
| $56^2 \times 24$ | Bottleneck3 | 6 | 32 | 3 | 2 |
| $28^2 \times 32$ | Bottleneck4 | 6 | 64 | 4 | 2 |
| $14^2 \times 64$ | Bottleneck5 | 6 | 96 | 3 | 1 |
| $14^2 \times 96$ | Bottleneck6 | 6 | 160 | 3 | 2 |
| $7^2 \times 160$ | Bottleneck7 | 6 | 320 | 1 | 1 |

Table 1. The layers of encoder network where each line describes a sequence of 1 or more identical (modulo stride) layers, repeated $n$ times. All layers in the same sequence have the same number of output channels, $c$. The first layer of each sequence has a stride $s$ and all others use stride 1. All spatial convolutions use $3 \times 3$ kernels. The expansion factor $t$ is always applied to the input size.

| Input | Operator | $t$ | $c$ | $s$ |
|---|---|---|---|---|
| $7^2 \times 320$ | Inception5 | 6 | 96 | 1 |
| $14^2 \times 192$ | Inception4 | 6 | 32 | 1 |
| $28^2 \times 64$ | Inception3 | 6 | 24 | 1 |
| $56^2 \times 48$ | Inception2 | 6 | 16 | 1 |
| $128^2 \times 32$ | Inception1 | 6 | 16 | 1 |
| $224^2 \times 16$ | Convf1 | - | 8 | 1 |
| $224^2 \times 8$ | Convf2 | - | 1 | 1 |

Table 2. The layers of decoder network, where $c$ represents the number of output channels and $s$ represents the stride of all convolution operations inside the inception blocks. The expansion factor $t$ is the ratio between the number of output channels of the concatenation operation and input channels to the inception block.

the high level features to give refined saliency maps. The bottleneck is implemented as shown in figure 2 where ReLU non-linearity is applied to the output of the depthwise $1 \times 1$, $3 \times 3$ convolutions so that no information is destroyed. The pointwise convolution is a linear operation that embeds the feature vector to a low dimensional space. Table 1 shows the encoder module. Figure 3 shows the inception module, which consists of multi-scale convolutions to capture the local and global context in the image. Each inception module consists of a linear $3 \times 3$ transpose convolution operation that upsamples and compresses the number of channels at the output. The output of $3 \times 3$ and $1 \times 1$ convolution at the final layer is then squashed using a softmax to generate probability scores per pixel, which are nothing but the saliency maps. The last two convolution layers do not have a batch norm implementation. Table 2 shows the decoder module.

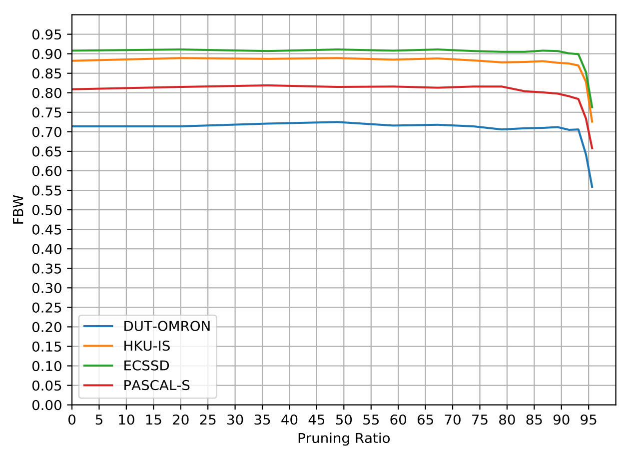


Figure 5. FBW vs pruning ratio curve for the QUICKSAL model trained on MSRA10k with a train/val/test split of 0.8/0.1/0.1 respectively. The model performance remains constant until the sharp drop at a pruning ratio of ∼*93% (elbow)*.

## 4. Pruning Strategy

Consider a network $f(x;\theta)$ with initial parameters $\theta = \theta_0 \sim \mathcal{D}_\theta$. Network pruning involves finding a mask $m \in \{0,1\}^{|\theta|}$ such that accuracy of the network $f(x; m \odot \theta)$ is comparable to that of $f(x;\theta)$, i.e we would like to uncover a sub network which has the same performance as the complete network. The network is allowed to train for $N$ iterations on the training data before the least significant weights are pruned out. Pruning out $p$

| Pruning Percentage | Non-Zero Parameters | DUT-OMRON [61] | | HKU-IS [25] | | ECSSD [60] | | PASCAL-S [28] | |
|---|---|---|---|---|---|---|---|---|---|
| | | MAE(↓) | $max.F_\beta$(↑) | MAE(↓) | $max.F_\beta$(↑) | MAE(↓) | $max.F_\beta$(↑) | MAE(↓) | $max.F_\beta$ (↑) |
| 0% | 3304705 | 0.084 | 0.714 | 0.053 | 0.882 | 0.061 | 0.908 | 0.117 | 0.809 |
| 20% | 2643765 | 0.081 | 0.714 | **0.051** | **0.889** | **0.059** | **0.911** | 0.114 | 0.815 |
| 36% | 2115014 | 0.079 | 0.721 | **0.051** | 0.887 | 0.060 | 0.907 | **0.113** | **0.819** |
| 48.8% | 1692013 | **0.078** | **0.725** | 0.052 | **0.889** | **0.059** | **0.911** | 0.115 | 0.815 |
| 59.04% | 1353612 | 0.084 | 0.716 | 0.053 | 0.885 | 0.060 | 0.908 | 0.116 | 0.816 |
| 67.24% | 1082892 | 0.081 | 0.718 | 0.052 | 0.888 | **0.059** | **0.911** | 0.116 | 0.813 |
| 73.79% | 866315 | 0.080 | 0.714 | 0.052 | 0.883 | 0.060 | 0.907 | 0.117 | 0.816 |
| 79.03% | 693054 | 0.086 | 0.706 | 0.054 | 0.878 | 0.061 | 0.905 | 0.117 | 0.816 |
| 83.23% | 554445 | 0.082 | 0.709 | 0.055 | 0.879 | 0.061 | 0.905 | 0.119 | 0.804 |
| 86.58% | 443558 | 0.082 | 0.710 | 0.055 | 0.881 | 0.062 | 0.908 | 0.119 | 0.801 |
| 89.27% | 354848 | 0.083 | 0.712 | 0.057 | 0.877 | 0.063 | 0.907 | 0.118 | 0.798 |
| 91.42% | 283881 | 0.086 | 0.705 | 0.060 | 0.875 | 0.066 | 0.901 | 0.117 | 0.791 |
| 93.12% | 227106 | 0.087 | 0.706 | 0.063 | 0.870 | 0.069 | 0.899 | 0.122 | 0.784 |
| 94.51% | 181687 | 0.109 | 0.642 | 0.081 | 0.828 | 0.093 | 0.853 | 0.133 | 0.734 |
| 95.64% | 145351 | 0.148 | 0.559 | 0.135 | 0.726 | 0.147 | 0.763 | 0.176 | 0.658 |

Table 3. Quantitative comparison of QUICKSAL trained on MSRA10k with multiple levels of pruning on various datasets. Each pruning iteration removes 20% of the remaining model parameters. Top results are in **bold** numbers.

| Pruning Percentage | Non-Zero Parameters | DUT-OMRON [61] | | HKU-IS [25] | | ECSSD [60] | | PASCAL-S [28] | |
|---|---|---|---|---|---|---|---|---|---|
| | | MAE(↓) | $max.F_\beta$(↑) | MAE(↓) | $max.F_\beta$(↑) | MAE(↓) | $max.F_\beta$(↑) | MAE(↓) | $max.F_\beta$ (↑) |
| 0% | 3304705 | 0.072 | 0.745 | **0.049** | **0.905** | **0.062** | **0.916** | **0.109** | **0.821** |
| 20% | 2643765 | 0.071 | 0.732 | 0.050 | 0.896 | 0.064 | 0.906 | 0.115 | 0.82 |
| 36% | 2115014 | 0.071 | **0.747** | 0.050 | 0.901 | 0.066 | 0.911 | 0.114 | 0.817 |
| 48.8% | 1692013 | 0.071 | 0.737 | 0.050 | 0.900 | 0.066 | 0.908 | 0.116 | 0.822 |
| 59.04% | 1353612 | **0.070** | 0.740 | 0.051 | 0.900 | 0.064 | 0.915 | 0.113 | 0.824 |
| 67.24% | 1082892 | **0.070** | 0.739 | 0.052 | 0.897 | 0.066 | 0.910 | 0.113 | 0.821 |
| 73.79% | 866315 | **0.070** | 0.735 | 0.051 | 0.901 | 0.066 | 0.914 | 0.115 | 0.814 |
| 79.03% | 693054 | 0.071 | 0.730 | 0.051 | 0.891 | 0.066 | 0.906 | 0.117 | 0.806 |
| 83.23% | 554445 | 0.075 | 0.727 | 0.053 | 0.893 | 0.066 | 0.903 | 0.113 | 0.808 |
| 86.58% | 443558 | 0.075 | 0.719 | 0.054 | 0.889 | 0.071 | 0.895 | 0.117 | 0.806 |
| 89.27% | 354848 | 0.088 | 0.670 | 0.063 | 0.166 | 0.084 | 0.869 | 0.128 | 0.785 |
| 91.42% | 283881 | 0.091 | 0.666 | 0.065 | 0.857 | 0.086 | 0.860 | 0.128 | 0.777 |
| 93.12% | 227106 | 0.105 | 0.631 | 0.077 | 0.829 | 0.098 | 0.841 | 0.138 | 0.748 |
| 94.51% | 181687 | 0.108 | 0.618 | 0.085 | 0.208 | 0.104 | 0.826 | 0.144 | 0.737 |
| 95.64% | 145351 | 0.266 | 0.337 | 0.276 | 0.478 | 0.300 | 0.515 | 0.295 | 0.496 |

Table 4. Quantitative comparison of QUICKSAL trained on MSRA-B with multiple levels of pruning on various datasets. Each pruning iteration removes 20% of the remaining model parameters. Top results are in **bold** numbers.

% of the model weights is carried out over k iterations. The model parameters are reinitialized to their initial value $\theta_0$ after every pruning iteration [10]. The pruning strategy can be summarized as follows,

Run for k iterations,

1. Randomly initialize a neural network $f(x;\theta_0)$ (where $\theta_0 \sim \mathcal{D}_\theta$).
2. Train the network for $j$ iterations, arriving at parameters $\theta_j$.
3. Prune bottom $p^{\frac{1}{k}}$% of the absolute value of parameters in $\theta_j$, creating a mask $m$.
4. Reset the remaining parameters to their values $\theta_0$.

# 5. Implementation Details

The weights of the encoder were initialized with those pretrained on the ImageNet [8] dataset. We train our model on MSRA10k dataset with a train/val/test split of 0.8/0.1/0.1 respectively. For fair comparison with DSS and DCL, we train our model separately on the MSRA-B dataset (5,000

| Methods | DUT-OMRON [61] | | HKU-IS [25] | | ECSSD [60] | | PASCAL-S [28] | |
|---|---|---|---|---|---|---|---|---|
| | MAE (↓) | $max.F_\beta$ (↑) | MAE (↓) | $max.F_\beta$ (↑) | MAE (↓) | $max.F_\beta$ (↑) | MAE (↓) | $max.F_\beta$ (↑) |
| BSCA | - | - | 0.175 | 0.719 | 0.182 | 0.758 | 0.223 | 0.667 |
| DRFI | - | - | 0.145 | 0.777 | 0.164 | 0.786 | 0.207 | 0.698 |
| RFCN | - | - | 0.079 | 0.892 | 0.107 | 0.890 | 0.118 | 0.837 |
| Amulet | 0.074 | 0.741 | 0.052 | 0.895 | 0.059 | 0.915 | 0.098 | **0.837** |
| SRM | 0.069 | 0.707 | **0.046** | 0.874 | **0.056** | 0.892 | - | - |
| NLDF | 0.085 | 0.724 | 0.060 | 0.874 | 0.075 | 0.886 | 0.108 | 0.804 |
| DSS* | **0.068** | 0.736 | **0.039** | **0.913** | **0.052** | **0.916** | **0.080** | 0.830 |
| ESOD | **0.066** | **0.751** | 0.054 | **0.915** | 0.063 | **0.921** | **0.083** | **0.846** |
| QUICKSAL-10k (*Best*) | 0.078 | 0.725 | 0.052 | 0.889 | 0.059 | 0.911 | 0.115 | 0.815 |
| QUICKSAL-10k (*Elbow*) | 0.087 | 0.706 | 0.063 | 0.87 | 0.069 | 0.899 | 0.122 | 0.784 |
| QUICKSAL-B (*Best*) | 0.072 | **0.745** | 0.049 | 0.905 | 0.062 | **0.916** | 0.109 | 0.821 |
| QUICKSAL-B (*Elbow*) | 0.075 | 0.719 | 0.054 | 0.889 | 0.071 | 0.895 | 0.117 | 0.806 |

Table 5. Quantitative comparison with other state-of-the-art methods on various datasets. QUICKSAL-10k *(Best)* and QUICKSAL-10k *(Elbow)* models are trained on MSRA10k and have a pruning ratio of 48.8% and 93.12% respectively. QUICKSAL-B *(Best)* and QUICKSAL-B *(Elbow)* models are trained on MSRA-B and have a pruning ratio of 0% and 86.58% respectively. Best and second best results are shown in **red** and **blue** respectively. Results best viewed in color.

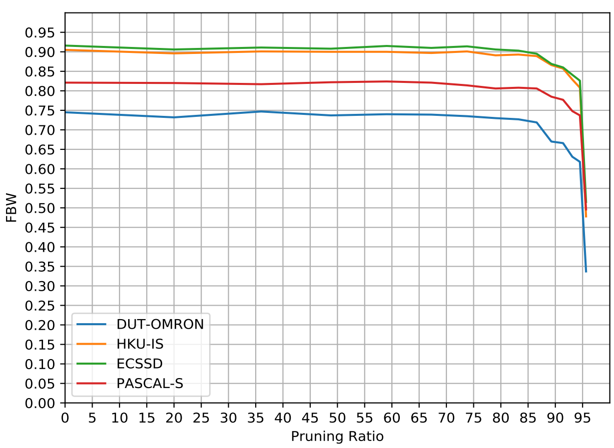


Figure 6. FBW vs pruning ratio curve for the QUICKSAL model trained on MSRA-B with a train/val/test split of 0.5/0.1/0.4 respectively. The model performance remains constant until the sharp drop at a pruning ratio of ∼*86% (elbow)*.

images) with a train/val/test split of 0.5/0.1/0.4 respectively. The images are rescaled to a size of $224 \times 224$ with no data augmentation. We use Adam optimizer, with CyclicLR scheduler [48] with the base and max learning rate set to 0.0001 and 0.001 respectively. We pruned out the bottom 20 % of the network parameters per pruning iteration, where each pruning iteration runs for 20 epochs. We choose not to prune the weights of the final convolution layer before the softmax in order to maintain maximum expressibility at the output layer.

*DSS also employs a CRF post-processing step.
†Non-zero parameters

| Methods | Model Parameters | Model Size (in MB) |
|---|---|---|
| DSS | 40M | 152 |
| DCL | 135M | 513 |
| DHS | 100M | 381 |
| Amulet | 35M | 133 |
| SRM | 30M | 115 |
| UCF | 17M | 65 |
| RFCN | 138M | 526 |
| ESOD | 16M | 64 |
| QUICKSAL-10k (Best) | **1.6M†** | **6.45** |
| QUICKSAL-10k (Elbow) | **0.22M†** | **0.86** |
| QUICKSAL-B (Best) | **3.3M†** | **12.6** |
| QUICKSAL-B (Elbow) | **0.44M†** | **1.69** |

Table 6. Comparison between the model sizes (approx) of state-of-the-art methods. The model implementation is assumed to be in float32, which is usually the case.

# 6. Experiments

## 6.1. Evaluation Metrics

**Mean Absolute Error:** MAE is computed as the mean of pixel-wise absolute difference between the continuous object saliency map and the binary ground-truth.

$$MAE = \frac{1}{H \times W} \sum_{i,j} |G(x_{ij}) - P(x_{ij})| \quad (4)$$

**Weighted $F_\beta$ Measure:** Weighted $F_\beta$-measure [36] evaluates a binarized map with respect to ground truth based on weighted harmonic mean of precision and recall values.


Figure 7. Qualitative comparison with various state-of-the-art approaches on some challenging images from ECSSD.

Similar to other works, $\beta^2 = 0.3$, thereby giving more importance to precision.

$$F_\beta = \frac{(1+\beta^2) \times Precision \times Recall}{\beta^2 \times Precision + Recall} \tag{5}$$

### 6.2. Results

In table 3 and table 4, we present the results for our QUICKSAL model with multiple levels of pruning trained on the MSRA10k and MSRA-B datasets respectively. As reported in [18], we observe that though MSRA10k is twice as big as MSRA-B, models trained on it have a competitive performance compared to those trained on the MSRA-B. We can observe that our model performance remains constant until it reaches a sparsity threshold, pruning beyond which results in performance degradation. This phenomenon is visualized in figure 5 and figure 6, where the curve has an elbow at a pruning ratio of 93.12% and 86.58% respectively. We will refer to these two models as QUICKSAL-10k (Elbow) and QUICKSAL-B (Elbow). Pruning seems to be also acting as a regularizer, allowing our model to generalize better. The performance of the QUICKSAL model trained on the MSRA-B dataset starts degrading much sooner with pruning than for the one trained on MSRA10k dataset. We observe the best performance of the QUICKSAL trained on MSRA10k and MSRA-B datasets at a pruning ratio of 48.8% and 0% (no pruning) respectively. We will refer to these two models as QUICKSAL-10k (Best) and QUICKSAL-B (Best).

In table 5, we compare the quantitative performance of various state-of-the-art methods with ours based on the aforementioned evaluation criteria. We compare against DSS [18], DCL [26], DHS [30], Amulet [63], SRM [58],UCF [64], RFCN [57], ESOD [41] and two non-deep methods -DRFI [55] and BSCA [44]. In table 6, we compare the model size of the different methods. We can see

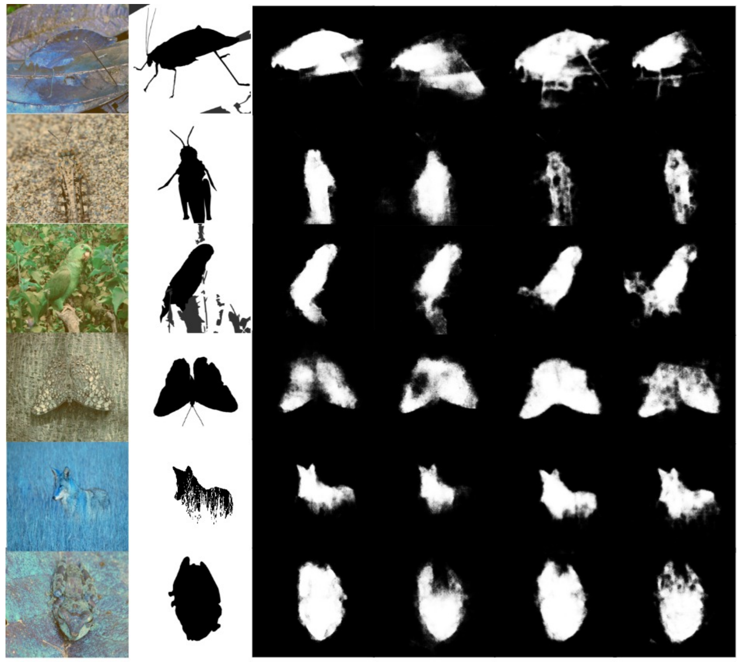


Figure 8. Qualitative results of our model on camouflage images.

that even though our model is significantly smaller than state-of-the-art, it is still able to achieve comparable results. It is to be noted that our model is designed with the constrain of being implementable on resource constrained hardware. 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 (except DSS).

In figure 7, we compare the qualitative results of the aforementioned methods with ours. We observe that our model performs well in identifying objects with a cluttered background (row 1) and low contrast (row 7) as well. It is able to identify the salient object even when there are multiple objects in the images (row 5, 6, and 7). It is able to perform well when there are objects within objects as in (row 9), which consists of a bird with a sharp contrast with a poster. It is able to distinguish to an extent objects from shadows (row 4). In figure 8 we show the results of our model on camouflage images. We observe that our model is able to distinguish between salient and non-salient regions in such complex scenario.

## 7. Conclusion

In this work, we have presented a small and sparse deep neural network model that performs efficient salient object segmentation. Our model was built using depthwise separable convolutions, bottleneck inverted residuals and inception blocks with deep residuals, with the least significant weights pruned out to induce sparsity. Our model design is 5-10 folds smaller than other state-of-the-art approaches. Pruning makes the network sparse and reduces the memory and power requirements even further. Due to sparsity, most of the weights are zero and we dont need to compute the activation function for most of the hidden nodes. The reduced memory read/write operations, saves the dynamic power significantly. Our network is tailored to be hardware realizable for edge computing devices. We have presented both quantitative and qualitative results on multiple public benchmark datasets which show that our QUICKSAL-10k *(Elbow)* and QUICKSAL-B *(Elbow)* with ∼0.22M and ∼0.44M non-zero parameters respectively have competitive results with other state-of-the-art methods. Our QUICKSAL-10k *(Best)* and QUICKSAL-B *(Best)* with ∼1.6M and ∼3.3M non-zero parameters respectively, have a performance almost in the top two.

## 8. Acknowledgement

Research facilities for this work were supported and funded by (i) INSPIRE faculty fellowship (DST/INSPIRE/04/2016/000216) from the Department of Science Technology, India, (ii) Funded by SERB (Science and Engineering Research Board), India: ECR/2017/002517 and IMP/2018/000550.

## References


[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[2] J. Ba and R. Caruana. Do deep nets really need to be deep? In *Advances in neural information processing systems*, pages 2654–2662, 2014.

[3] A. Borji, M.-M. Cheng, Q. Hou, H. Jiang, and J. Li. Salient object detection: A survey. *Computational Visual Media*, pages 1–34, 2014.

[4] D. E. Broadbent. *Perception and communication*. Elsevier, 2013.

[5] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015.

[6] M.-M. Cheng, N. J. Mitra, X. Huang, P. H. Torr, and S.-M. Hu. Global contrast based salient region detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):569–582, 2014.

[7] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[9] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in neural information processing systems*, pages 1269–1277, 2014.

[10] J. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

[11] D. Han, J. Kim, and J. Kim. Deep pyramidal residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5927–5935, 2017.

[12] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[13] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.

[14] 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.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[16] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[17] M. Horowitz. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 10–14. IEEE, 2014.

[18] Q. Hou, M.-M. Cheng, X. Hu, A. Borji, Z. Tu, and P. H. Torr. Deeply supervised salient object detection with short connections. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3203–3212, 2017.

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

[20] K. Kandasamy, W. Neiswanger, J. Schneider, B. Poczos, and E. P. Xing. Neural architecture search with bayesian optimisation and optimal transport. In *Advances in Neural Information Processing Systems*, pages 2016–2025, 2018.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[22] S. S. Kruthiventi, V. Gudisa, J. H. Dholakiya, and R. Venkatesh Babu. Saliency unified: A deep architecture for simultaneous eye fixation prediction and salient object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5781–5790, 2016.

[23] M. Kümmerer, L. Theis, and M. Bethge. Deep gaze i: Boosting saliency prediction with feature maps trained on imagenet. *arXiv preprint arXiv:1411.1045*, 2014.

[24] V. Lebedev and V. Lempitsky. Fast convnets using group-wise brain damage. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2554–2564, 2016.

[25] G. Li and Y. Yu. Visual saliency based on multiscale deep features. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5455–5463, 2015.

[26] G. Li and Y. Yu. Deep contrast learning for salient object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 478–487, 2016.

[27] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[28] Y. Li, X. Hou, C. Koch, J. M. Rehg, and A. L. Yuille. The secrets of salient object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 280–287, 2014.

[29] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 19–34, 2018.

[30] N. Liu and J. Han. Dhsnet: Deep hierarchical saliency network for salient object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 678–686, 2016.

[31] N. Liu, J. Han, D. Zhang, S. Wen, and T. Liu. Predicting eye fixations using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 362–370, 2015.

[32] T. Liu, Z. Yuan, J. Sun, J. Wang, N. Zheng, X. Tang, and H.-Y. Shum. Learning to detect a salient object. *IEEE Transactions on Pattern analysis and machine intelligence*, 33(2):353–367, 2010.

[33] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[34] Z. Luo, A. Mishra, A. Achkar, J. Eichel, S. Li, and P.-M. Jodoin. Non-local deep features for salient object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6609–6617, 2017.

[35] Y.-F. Ma and H.-J. Zhang. Contrast-based image attention analysis by using fuzzy growing. In *Proceedings of the eleventh ACM international conference on Multimedia*, pages 374–381. ACM, 2003.

[36] R. Margolin, L. Zelnik-Manor, and A. Tal. How to evaluate foreground maps? In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 248–255, 2014.

[37] G. F. Miller, P. M. Todd, and S. U. Hegde. Designing neural networks using genetic algorithms. In *ICGA*, volume 89, pages 379–384, 1989.

[38] J. L. Molin, A. F. Russell, S. Mihalas, E. Niebur, and R. Etienne-Cummings. Proto-object based visual saliency model with a motion-sensitive channel. In *2013 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pages 25–28. IEEE, 2013.

[39] S. Narayanan, Y. Bethi, J. Lottier, E. Niebur, R. Etienne-Cummings, and C. Thakur. Live demonstration: Real-time implementation of proto-object based visual saliency model. In *2019 IEEE International Symposium on Circuits and Systems, ISCAS 2019 - Proceedings*, Proceedings - IEEE International Symposium on Circuits and Systems. Institute of Electrical and Electronics Engineers Inc., 1 2019.

[40] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[41] A. Pahuja, A. Majumder, A. Chakraborty, and R. Venkatesh Babu. Enhancing salient object segmentation through attention. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 27–36, 2019.

[42] G. Papandreou, L.-C. Chen, K. P. Murphy, and A. L. Yuille. Weakly- and semi-supervised learning of a deep convolutional network for semantic image segmentation. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[43] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[44] Y. Qin, H. Lu, Y. Xu, and H. Wang. Saliency detection via cellular automata. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 110–119, 2015.

[45] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016.

[46] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[47] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 91–99. Curran Associates, Inc., 2015.

[48] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

[49] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[50] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[51] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[52] C. S. Thakur, J. L. Molin, T. Xiong, J. Zhang, E. Niebur, and R. Etienne-Cummings. Neuromorphic visual saliency implementation using stochastic computation. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4. IEEE, 2017.

[53] J. K. Tsotsos. Is complexity theory appropriate for analyzing biological systems? *Behavioral and Brain Sciences*, 14(4):770–773, 1991.

[54] W.-C. Tu, S. He, Q. Yang, and S.-Y. Chien. Real-time salient object detection with a minimum spanning tree. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2334–2342, 2016.

[55] J. Wang, H. Jiang, Z. Yuan, M. Cheng, X. Hu, N. Zheng, and S. Detection. A discriminative regional feature integration approach. *IEEE Int. J. Comput. Vis*, pages 1–18, 2017.

[56] L. Wang, H. Lu, X. Ruan, and M.-H. Yang. Deep networks for saliency detection via local estimation and global search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3183–3192, 2015.

[57] L. Wang, L. Wang, H. Lu, P. Zhang, and X. Ruan. Saliency detection with recurrent fully convolutional networks. In *European conference on computer vision*, pages 825–841. Springer, 2016.

[58] T. Wang, A. Borji, L. Zhang, P. Zhang, and H. Lu. A stagewise refinement model for detecting salient objects in images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4019–4028, 2017.

[59] S. Xie and Z. Tu. Holistically-nested edge detection. In *Proceedings of the IEEE international conference on computer vision*, pages 1395–1403, 2015.

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

[61] C. Yang, L. Zhang, H. Lu, X. Ruan, and M.-H. Yang. Saliency detection via graph-based manifold ranking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3166–3173, 2013.

[62] Y. Zhai and M. Shah. Visual attention detection in video sequences using spatiotemporal cues. In *Proceedings of the 14th ACM international conference on Multimedia*, pages 815–824. ACM, 2006.

[63] P. Zhang, D. Wang, H. Lu, H. Wang, and X. Ruan. Amulet: Aggregating multi-level convolutional features for salient object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 202–211, 2017.

[64] P. Zhang, D. Wang, H. Lu, H. Wang, and B. Yin. Learning uncertain convolutional features for accurate saliency detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 212–221, 2017.

[65] R. Zhao, W. Ouyang, H. Li, and X. Wang. Saliency detection by multi-context deep learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1265–1274, 2015.

[66] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.