Journal of the Brazilian Computer Society, 2026, 32:1, doi: 10.5753/jbcs.2026.5893

FLIM-based Salient Object Detection Networks with Adaptive Decoders

Gilson Junior Soares [**University of Campinas** | ***gilson.soares@students.ic.unicamp.br***]
Matheus Abrantes Cerqueira [**University of Campinas** | ***matheus.cerqueira@students.ic.unicamp.br***]
Jancarlo F. Gomes [**University of Campinas** | ***jgomes@ic.unicamp.br***]
Laurent Najman [**Univ Gustave Eiffel, CNRS, LIGM, Champs-sur-Marne, France and Department of Mathematics, Khalifa University, Abu Dhabi, United Arab Emirates** | ***laurent.najman@esiee.fr***]
Silvio Jamil F. Guimarães [**Pontificial Catholic University of Minas Gerais** | ***sjamil@pucminas.br***]
Alexandre X. Falcão [**University of Campinas** | ***afalcao@unicamp.br***]

Institute of Computing, University of Campinas (UNICAMP), Av. Albert Einstein, 1251, Cidade Universitária, Campinas, SP, 13083-852, Brazil.

Received: 12 April 2025 • **Accepted:** 10 September 2025 • **Published:** 16 April 2026

Abstract Salient Object Detection (SOD) methods can locate objects that stand out in an image, assign higher values to their pixels in a saliency map, and binarize the map outputting a predicted segmentation mask. A recent tendency is to investigate pre-trained lightweight models rather than deep neural networks in SOD tasks, coping with applications under limited computational resources. In this context, we have investigated lightweight networks using a methodology named *Feature Learning from Image Markers* (FLIM), which assumes that the encoder's kernels can be estimated from marker pixels on discriminative regions of a few representative images. This work proposes flyweight networks, hundreds of times lighter than lightweight models, for SOD by combining a FLIM encoder with an *adaptive decoder*, whose weights are estimated for each input image by a given heuristic function. Such FLIM networks are trained from three to four representative images only and without backpropagation, making the models suitable for applications under labeled data constraints as well. We study five adaptive decoders; two of them are introduced here. Differently from the previous ones that rely on one neuron per pixel with shared weights, the heuristic functions of the new adaptive decoders estimate the weights of each neuron per pixel. We compare FLIM models with adaptive decoders for two challenging SOD tasks with three lightweight networks from the state-of-the-art, two FLIM networks with decoders trained by backpropagation, and one FLIM network whose labeled markers define the decoder's weights. For one of the applications, we evaluate the generalization ability of the networks to six different datasets. The experiments demonstrate the advantages of the proposed networks over the baselines, revealing the importance of further investigating such methods in new applications.

Keywords: Salient Object Detection, FLIM, Adaptive Decoders, Decoding

1 Introduction

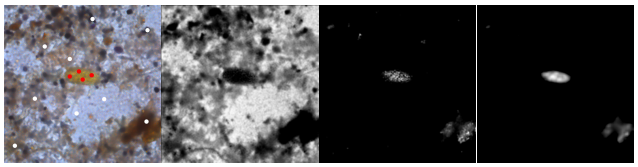
Salient Object Detection (SOD) methods [Wang *et al*., 2015; Zhao *et al*., 2015; Borji *et al*., 2019] can locate objects that stand out in an image and assign higher values to their pixels in a saliency map. They then binarize the map, showing a segmentation mask with the detected objects. Many works rely on deep learning methods, while a recent tendency is lightweight models that require much less computational resources [Liu *et al*., 2021; Lin *et al*., 2022; Liang and Luo, 2024]. Training such models requires extensive human effort to create segmentation masks (data annotation) by object delineation. Fine-tuning the models to new applications may also require considerable human effort in data annotation.

In this work, we investigate a recent methodology for SOD [Joao *et al*., 2023; Joao. *et al*., 2024], which combines a convolutional encoder trained from user-drawn markers on discriminative regions of a few representative images [De Souza *et al*., 2020] with an *adaptive decoder* that creates saliency maps by estimating its parameters for each input image. The methodology allows the construction of *flyweight* SOD models, hundreds of times more efficient than lightweight networks, trained with minimum human effort in data annotation and without backpropagation. The paper proposes and evaluates new adaptive decoders using a graph-based delineation algorithm to segment the objects.

Convolutional encoders are trained with *Feature Learning from Image Markers* (FLIM). The user draws markers on discriminative (object and background) regions of representative images (Figure 1a), and FLIM estimates the kernels of all convolutional blocks by clustering patches centered at marker pixels using the input feature map (activation channels) of each block and selecting the cluster's centers to constitute the kernels. A kernel derived from a background marker is expected to create a channel with background activation (Figure 1b), while a kernel derived from an object marker is expected to generate a foreground activation channel (Figure 1c). An adaptive decoder may be a simple point-wise convolution (i.e., a weighted average of the encoder's output channels) followed by activation, in which a heuristic function estimates the convolutional weights for each input image. The above characteristics of a FLIM encoder allow adaptive

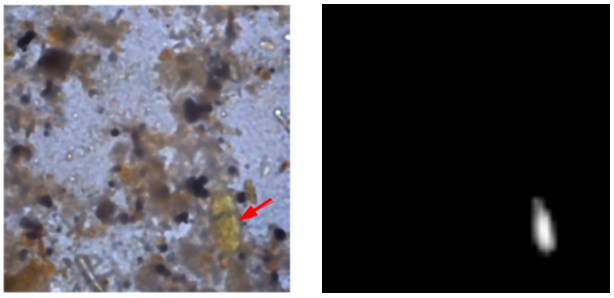
decoders to create reasonable saliency maps (Figure 1d).

In applications with redundant object and background properties, FLIM-based SOD models can be trained with only five images [Joao *et al*., 2023] and generalize to new ones (Figures 2a and 2b). Moreover, FLIM networks have shown promising results in other tasks, such as object delineation [De Souza *et al*., 2020], image classification [De Souza and Falcão, 2020], and instance segmentation [Cerqueira *et al*., 2023], surpassing the same architecture trained from scratch and, in some situations, deep models [De Souza and Falcão, 2020; Joao *et al*., 2023].



(a) (b) (c) (d)

Figure 1. Marker drawing for kernel estimation: (a) A training image with foreground (red) and background (white) markers; (b) background activation channel from a background kernel; (c) foreground activation channel from an object kernel; (d) resulting saliency map.



(a) (b)

Figure 2. A FLIM-based SOD model generalizing to new images. (a) A test image where the red arrow indicates the object (a parasite egg) and (b) the resulting saliency map.

This work studies five adaptive decoders: (i) a tri-state decoder, which is a slight modification of the original one in Joao. *et al*. [2024], (ii) an attention-based decoder, (iii) a label-based tri-state decoder, (iv) a probability-based decoder, and (v) a mean-based decoder. The present paper is an extended version of our previous work [Soares *et al*., 2024], in which decoders (i)-(iii) were introduced. Therefore, decoders (iv) and (v) are new contributions proposed here. The work compares FLIM-based SOD networks using the five adaptive decoders and three decoders with fixed weights (two of them learned by backpropagation). This comparative analysis also includes three lightweight SOD models from the state-of-the-art. The models were evaluated on two SOD tasks: (i) detection of parasite eggs in microscopy images, and (ii) brain tumor detection in magnetic resonance images. Given that recent lightweight networks adopt a delineation loss to improve saliency maps and segment the objects by thresholding, we explore a graph-based delineation algorithm for the same purpose. Graph-based object delineation was adopted only for parasite egg detection and was not used in Soares *et al*. [2024].

The experiments differ in several aspects from the ones in Soares *et al*. [2024]. FLIM-based SOD networks were created by two users, who followed an image selection procedure on a validation set to guarantee representative training images. The datasets were randomly divided three times into two parts with 50% of the images each; one part was used to select the training set, leaving the remaining images for validation, and the other was used for test. Three to four representative training images were selected per split. The previous work used only a single test set per dataset and five training images chosen by a single user based on experience. As the work focuses on the decoder part, the encoder's architecture was selected using the validation set since an unsuitable encoder can negatively impact the decodification process. Another difference is the rule that defines the number of filters per block. Rather than specifying an empirical number of kernels per block, which required reducing the total number of kernels derived from the markers, this work sets the number of kernels per block as the number of markers multiplied by the number of kernels per maker.

In summary, the contributions of this paper are: (1) two new adaptive decoders that introduce the concept of estimating one adaptive neuron per pixel for each image; (2) FLIM-based flyweight SOD networks that are hundreds of times more efficient than the current lightweight ones, and require very few images (three-four) to be trained, assuming the selected images are representative; (3) more extensive evaluation experiments involving two users, five adaptive decoders, six baselines, two datasets, and two metrics; (4) the use of a graph-based algorithm for object delineation; and (5) extended evaluation of the generalization ability of the proposed methods in a zero-shot experiment to six different datasets. The paper is organized in the following way: Related works are discussed in Section 2. Section 3 provides the background on FLIM encoders, while Section 4 presents the adaptive decoders. Section 5 describes the experimental setup, including the image selection and object delineation procedures. Results are presented and discussed in Section 6. Finally, Section 7 states the conclusion and discusses future work.

2 Related Work

Traditional SOD methods rely on local image features (texture, color, orientation) and a bottom-up approach to combine them into a saliency map [Ullah *et al*., 2020]. One of the first works decomposes an image into feature maps that are further processed and combined at various scales to generate a saliency map [Itti *et al*., 2002]. A similar strategy rescales the feature maps to the same domain of the input image before combining them into a saliency map with better resolution [Achanta *et al*., 2008]. There are also methods based on focusness [Cheng *et al*., 2014; Zhu *et al*., 2014] and objectness priors [Chang *et al*., 2011; Jiang *et al*., 2013].

Most recent SOD methods use a convolutional encoder for feature map extraction, due to its innate capacity to generate multi-level and multi-scale features [Borji *et al*., 2019]. In the stage of feature map combination, methods fall into two categories: the ones that rely on multi-layer perceptrons (MLPs) for pixel classification into object/background [Lee *et al*.,

2016; He *et al*., 2015], and those that generate a saliency map using a convolutional decoder [Liu *et al*., 2018; Qin *et al*., 2019, 2020]. The second category, Fully Convolutional Networks – FCNs, is the most common. PiCANet [Liu *et al*., 2018], for instance, resembles the U-Net model with a convolutional encoder from the Resnet/VGG network. BASNet [Qin *et al*., 2019] adopts a similar U-shape with the encoder of ResNet-34 as the backbone and with a focus on boundary refinement through supervision in different layers of the network, including a module to refine the saliency map. The idea was further enhanced by employing nested U-shaped structures across different network levels, enabling the capture of richer contextual information at varying scales [Qin *et al*., 2020]. Unlike previous deep learning-based approaches, this method does not rely on a pretrained backbone and is instead trained from scratch.

Although the deep FCNs achieve state-of-the-art results, they consume considerable computational resources. In this scenario, lightweight models emerge while maintaining equivalent results. The methods can create a lightweight network by design or compression techniques – e.g., knowledge distillation, quantization, and network simplification [Chen *et al*., 2024]. The former replaces convolution by group, separable, or dilated convolutions. These modified operations reduce the number of parameters and the computational complexity of the model. The expert usually does such changes. Still, it can be done automatically based on AutoML [Chen *et al*., 2024], which searches the best architecture for the network [He *et al*., 2021] – a subtopic also known as Network Architecture Search (NAS).

Lightweight models are usually pre-trained on ImageNet [Deng *et al*., 2009]. Several models [Lin *et al*., 2022; Li *et al*., 2023; Liang and Luo, 2024] use the encoder of MobileNet-V2 [Sandler *et al*., 2018] as the backbone to constrain the model's size. Two well-succeeded examples are MSCNet [Lin *et al*., 2022] and MEANet [Liang and Luo, 2024], both developed to cope with the multi-scale information of salient objects in remote sensing images. To address the problem, MSCNet and MEANet introduce a multi-scale context extraction module and a multi-scale edge-embedded attention module. Other works developed custom backbones [Wang *et al*., 2023a,b; Zhou *et al*., 2024; Liu *et al*., 2021]. They argue that utilizing pre-trained lightweight encoders from MobileNet-V2 [Sandler *et al*., 2018], ShuffleNet [Zhang *et al*., 2018], and GhostNet [Han *et al*., 2020] makes it more difficult to compress the model, generates redundant features, and affects the quality of the saliency maps. One well-succeeded example is SAMNet [Liu *et al*., 2021], which introduces a stereoscopically attentive multi-scale module to fuse image features at various scales.

Among lightweight networks that do not rely on pre-trained backbones, FLIM-based models [De Souza and Falcão, 2020] have proven effectiveness in detecting [Joao *et al*., 2023], classifying [De Souza and Falcão, 2020], and segmenting objects [De Souza *et al*., 2020; Cerqueira *et al*., 2023]. In all methods, the encoder is trained from user-drawn markers without backpropagation. However, backpropagation may improve the encoder when annotated examples are available in a reasonable number [Cerqueira *et al*., 2023]. For SOD tasks, FLIM-based networks use adaptive decoders [Joao *et al*., 2023; Joao. *et al*., 2024; Soares *et al*., 2024], which allow training the entire model from a few marked images without backpropagation.

This work introduces two adaptive decoders besides the three decoders presented in Soares *et al*. [2024] and adds a graph-based delineation algorithm to segment the object. Alternatively, one could improve the saliency map before segmentation by exploring cellular automata [Salvagnini *et al*., 2024]. We define the number $m' = m_1 \times m_2$ of kernels per block as the number m_1 of markers multiplied by the number m_2 of kernels per marker. Previous works define $m' < m_1 \times m_2$ requiring methods that reduce to m' the total number $m_1 \times m_2$ of estimated kernels, which may miss important information.

An essential step in FLIM-based networks is selecting representative training images. We wish to choose only a few representative examples from an unlabeled set to minimize human effort in data annotation. Supervised [Cerqueira *et al*., 2024a] and unsupervised [Cerqueira *et al*., 2024b] approaches are under investigation, and such representative images are crucial for the encoder's performance. Since the decoder is our focus, we adopt a supervised approach based on Cerqueira *et al*. [2024b], with the difference that we make the selection based on the result after the decoder, not on the feature maps and do not manually annotate the convolutional filters. We select one image at a time and update the model to verify an incremental performance on a validation set. The user retains the images that improve the model and stops with three or four images in the training set.

3 FLIM encoders

This section provides definitions related to FLIM encoders. An encoder is a sequence of blocks $b = 0, 1, \ldots, B-1$, where each block applies four operations: marker-based normalization, convolution with a kernel bank, activation, and pooling. The user draws markers (e.g., disks of radius 3) on a few training images at the input of block $b = 0$. These markers are mapped onto the domain of the input image of each block $b > 0$. Marker-based normalization involves correcting the input images of each block using parameters estimated from the marker pixels mapped onto their domain, thereby eliminating bias in the block. Convolution kernels are calculated using K-means clustering on the patch set extracted from each marker for all markers. The number of kernels in each block b will depend on the total number of markers drawn by the user on the training images, where each marker i contributes with K kernels. Hence, the total number m' of kernels in each block is fixed as the total number of markers multiplied by K. Since the markers are small disks, we did not see any need to use a different number of kernels per block. The details are provided next.

3.1 Images and patches

Let $\mathbf{I}^b = (D_{I_b}, \vec{I}_b)$ be an image with m_b channels ($m_b = m$ for $b = 0$ and $m_b = m'$ for $b > 0$) as presented at the input of an encoder's block b, where $D_{I_b} \subset \mathcal{Z}^2$ is the image domain of size $w_b \times h_b$ pixels and $\vec{I}_b(p) =$

$(I_{1,b}(p), I_{2,b}(p), \ldots, I_{m_b,b}(p)) \in \mathbb{R}^{m_b}$ assigns m_b features to every pixel $p = (x_p, y_p) \in D_{I_b}$. Let $A_b(p)$ be a set of $k_b \times k_b$ adjacent pixels $q = (x_q, y_q) \in D_{I_b}$, such that $x_q - x_p \in [-d_b \frac{k_b}{2}, d_b \frac{k_b}{2}]$ and $y_q - y_p \in [-d_b \frac{k_b}{2}, d_b \frac{k_b}{2}]$, within a squared region of size $d_b k_b \times d_b k_b$ centered at p with dilation factor $d_b \geq 1$. A patch $\vec{P}_b(p) \in \mathbb{R}^{k_b \times k_b \times m_b}$ centered at p results from the concatenation of feature vectors $\vec{I}_b(q)$ of all $q \in A_b(p)$.

The parameters k_b, K, d_b, and B are defined in the encoder's architecture file. The parameter k_b defines the shape of the filters in block b – i.e., patches and filters should have shape $k_b \times k_b \times m_b$.

3.2 Sets of marker pixels

Let $\mathcal{M}_i(\mathbf{I}^b)$ be the set of pixels in a marker i on the domain of $\mathbf{I}^b$. Let $\mathcal{M}(\mathbf{I}^b) = \bigcup_i \mathcal{M}_i(\mathbf{I}^b)$ and $\mathcal{M}^b$ be the union of all sets $\mathcal{M}(\mathbf{I}^b)$ derived from all training images $\mathbf{I}^b$. Recall that the user draws markers on $\mathbf{I}^0$ only (the original image) and the sets $\mathcal{M}_i(\mathbf{I}^b)$, $b > 0$, are obtained by mapping the pixels in $\mathcal{M}_i(\mathbf{I}^0)$ into $\mathcal{M}_i(\mathbf{I}^b)$.

3.3 Convolutional block

Each convolutional block contains marker-based normalization, convolution, activation, and pooling. However, one can incorporate other operations.

3.3.1 Marker-based normalization

Each channel $I_{j,b}$, $j = 1, 2, \ldots, m_b$, of $\mathbf{I}^b$ can be normalized by

$$I_{j,b}(p) \leftarrow \frac{I_{j,b}(p) - \mu_{j,b}}{\sigma_{j,b} + \epsilon}, \tag{1}$$

$$\mu_{j,b} = \frac{1}{\mid \mathcal{M}^b \mid} \sum_{\mathcal{M}(\mathbf{I}^b) \in \mathcal{M}^b, p \in \mathcal{M}(\mathbf{I}^b)} I_{j,b}(p), \tag{2}$$

$$\sigma^2_{j,b} = \frac{1}{\mid \mathcal{M}^b \mid} \sum_{\mathcal{M}(\mathbf{I}^b) \in \mathcal{M}^b, p \in \mathcal{M}(\mathbf{I}^b)} \left(I_{j,b}(p) - \mu_{j,b}\right)^2, \tag{3}$$

where $\epsilon > 0$ is a small value. This operation centralizes the patches around the origin of $\mathbb{R}^{k_b \times k_b \times m_b}$ and corrects distortions among different features, dismissing the need for estimating bias [Joao. *et al.*, 2024].

3.3.2 Convolution, activation, and pooling

Let $\mathcal{P}_i(\mathbf{I}^b)$ be a set of patches $\vec{P}_b(p) \in \mathbb{R}^{k_b \times k_b \times m_b}$ centered at pixels $p \in \mathcal{M}_i(\mathbf{I}^b)$, we execute K-means clustering on $\mathcal{P}_i(\mathbf{I}^b)$ to select the K patches in marker i closest to their respective cluster centers. Such patches are forced to have unit norm, $\vec{P}_b(p) \leftarrow \frac{\vec{P}_b(p)}{\|\vec{P}_b(p)\|}$, and used as kernels for block b.

The above kernel estimation procedure obtains m' kernels per block $b \geq 0$: $\vec{K}_{j,b} \in \mathbb{R}^{k_b \times k_b \times m_b}$, $j = 1, 2, \ldots, m'$. The marker labels may be explored for kernel estimation [De Souza and Falcão, 2020], but this option is not adopted in this work.

The convolutions between the input image $\mathbf{I}^b$ and each of the m' kernels $\vec{K}_{j,b}$ generate an image $\mathbf{J}^b$ with m' channels $J_{j,b}$, such that

$$J_{j,b}(p) = \langle \vec{P}_b(p), \vec{K}_{j,b} \rangle, \tag{4}$$

$j = 1, 2, \ldots, m'$. We apply ReLU followed by max-pooling to each channel, such that the output image $\mathbf{I}^{b+1}$ of block $b \geq 0$ has m' activation channels.

By drawing markers with labels $l \in \{1, 2\}$ (background, foreground) and estimating kernels from each marker, the label $\lambda(I_{j,b+1}) \in \{1, 2\}$ of channel $I_{j,b+1}$ is the same of the marker used to estimate kernel $\vec{K}_{j,b}$ that has generated it. This is valid for any block $b \geq 0$ and one can expect background or foreground activations in the respective channels. However, this might not occur due to the cross-influence among training images. For this reason, adaptive decoders rely on a heuristic function that estimates background and foreground activation channels for each input image.

4 Adaptive Decoders

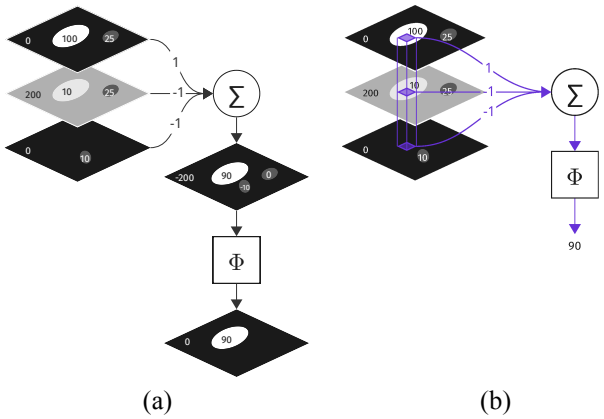


Figure 3. Adaptive decoders. (a) A point-wise convolution followed by activation, whose weights are estimated in $-1, +1$ by a heuristic function. (b) It defines one neuron per pixel, and all neurons share the same weights.

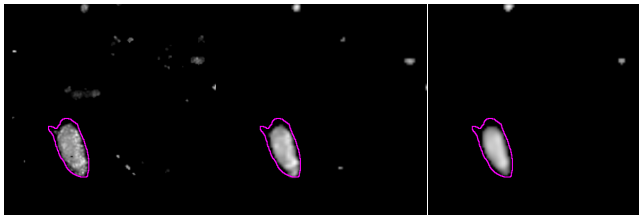
An adaptive decoder is a method that can combine the activation channels (image features) from the feature map of a given block into an object saliency map by adapting its parameters for each input image. The simplest example is a point-wise convolution between the feature map $\mathbf{I}^B$ and a point-wise kernel $\vec{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_{m'}) \in \mathbb{R}^{1 \times 1 \times m'}$, followed by ReLU activation Φ (Figure 3a-b). Such a decoder represents one neuron per pixel, with shared weights estimated by a given heuristic function. One may also re-estimate the weights of the point-wise convolution for each pixel.

In SOD, FLIM encoders are designed to either activate or deactivate object pixels in different channels. Although this expected behavior is not guaranteed, the presented adaptive decoders can create a reasonable saliency map $\mathbf{S}$ with image domain $D_S \subseteq D_I$ and values $S(p) \in \mathbb{R}$ for each $p \in D_S$,

such that

$$S(p) = \Phi\left(\langle \vec{I}_B(p), \vec{\alpha}\rangle\right) = \Phi\left(\sum_{j=1}^{m'} \alpha_j I_{j,B}(p)\right). \quad (5)$$

One may upsample $D_S = D_{I^B}$ to the size of D_{I^0}, whenever strides greater than one are used.



(a) Block 1 (b) Block 2 (c) Block 3

Figure 4. Saliencies generated by an adaptive decoder for different blocks of an architecture. The ground-truth's border is presented in magenta.

An interesting effect of combining an FLIM encoder with an adaptive decoder is that the saliency maps after each encoder's block show fewer false positives as the encoder adds blocks. However, object representation is more precise in the earlier blocks (Figure 4) due to the small receptive field to the input image. The deeper the layer of the network, the bigger the receptive field due to the effect of pooling operations with strides greater than one. This enables the network to transition from low-level to high-level features. Specifically, for a small kernel size, kernels that create channels with foreground activation can capture more of the object details (border and interior) at the initial blocks. However, such kernels also react to the elements in the background, generating more false positives in the channel. The cumulative effect of the blocks on those foreground activation channels can eliminate false positives as the block's depth increases. However, it misses details such as the object's border.

Next, we describe the proposed adaptive decoders. Different from the previous ones in Soares *et al*. [2024], the new adaptive decoders, probability-based and mean-based, adopt heuristic functions that estimate weights for each neuron per pixel. Hence, the neurons do not share weights.

4.1 Tri-state adaptive decoder (ts)

A tri-state adaptive decoder assumes the object occupies a considerably smaller portion of the image domain compared to the background. It identifies an object (background) activation channel when its mean activation is below (above) a threshold $\tau - \sigma$ ($\tau + \sigma$), and when the fraction of pixels above Otsu threshold at the channel is below (above) another threshold t_j. Such rules define each channel weight α_j^{ts}, with $\alpha_j^{ts} = 0$ when the rules are not satisfied.

Let $\mu_{I_{j,B}}$ be the mean activation of channel $I_{j,B}$, with τ denoting the Otsu threshold for the distribution $\{\mu_{I_{1,B}}, \mu_{I_{2,B}}, \ldots, \mu_{I_{m',B}}\}$, and σ^2 indicating the variance of that distribution. The number of pixels exceeding the Otsu threshold for channel $I_{j,B}$ divided by $|D_{I^0}|$ establishes a threshold t_j. The *tri-state adaptive decoder* (ts) defines α_j^{ts} as follows.

$$\alpha_j^{ts} = \begin{cases} -1, & \text{if } \mu_{I_{j,B}} \geq \tau + \sigma^2 \text{ and } t_j > 0.2, \\ +1, & \text{if } \mu_{I_{j,B}} \leq \tau - \sigma^2 \text{ and } t_j < 0.1, \\ 0, & \text{otherwise.} \end{cases} \quad (6)$$

The limits for the threshold, $t_j > 0.2$ and $t_j < 0.1$, are empirically defined for the datasets utilized in this work.

4.2 Attention-based adaptive decoder (at)

In Li *et al*. [2022], the authors introduce a method to calculate channel attention without utilizing trainable parameters. Channel attention should assign positive weights (more importance) to object channels and negative weights (less importance) to background channels. In our work, we adopt a slight variation of their technique. Let X and Y represent the outcomes of max-pooling and average pooling through all channels $I_{j,B}$, $j \in [1, m']$. X and Y have their values linearly scaled to $[0, 1]$. A spatial attention a is established by linearly normalizing the sum of X and Y within $[0, 1]$. Denote $\vec{a} \in \mathbb{R}^{w_B \times h_B}$ as the vectorized form of the spatial attention a, and $\vec{b}_j \in \mathbb{R}^{w_B \times h_B}$ as the vectorized form of channel $I_{j,B}$. Each channel's importance c_j is calculated by

$$c_j = \frac{\langle \vec{a}, \vec{b}_j \rangle}{\|\vec{a}\| \|\vec{b}_j\|}. \quad (7)$$

Let μ_c and σ_c be the mean and standard deviation of the distribution $\{c_1, c_2, \ldots, c_{m'}\}$. The weight vector for an attention-based decoder can be expressed as $\vec{\alpha}^{at} = (\alpha_1^{at}, \alpha_2^{at}, \ldots, \alpha_{m'}^{at})$, where

$$\alpha_j^{at} = \begin{cases} +1, & \text{if } c_j < \mu_c - \frac{\sigma_c}{2}, \\ -1, & \text{if } c_j > \mu_c + \frac{\sigma_c}{2} \\ 0, & \text{otherwise} \end{cases} \quad (8)$$

for $j \in [1, m']$.

4.3 Label-based tri-state adaptive decoder (lt)

Assuming that object and background activation channels are derived from kernels according to their labeled markers, the label-based tri-state adaptive decoder combines evidences from this information and the tri-state adaptive decoder. Let $\vec{\alpha}^{ts} = (\alpha_1^{ts}, \alpha_2^{ts}, \ldots, \alpha_{m'}^{ts})$ be the weight vector generated by the tri-state decoder for a particular input image, and $\lambda(I_{j,B}) \in \{1, 2\}$, $j \in [1, m']$, be the label for object and background activation channels. We establish

$$\alpha_j^{lt} = \begin{cases} 0, & \text{if } \lambda(I_{j,B}) = 2, \\ \alpha_j^{ts}, & \text{otherwise.} \end{cases} \quad (9)$$

Hence, we only consider the weights of the tri-state adaptive decoder for foreground-labeled kernels, eliminating channels where background-labeled kernels generate object activations.

4.4 Probability-based adaptive decoder (pb)

Let $\mathcal{A}(p)$ be a small neighborhood around p, $\mu_1(p)$ be the mean activation value of the adjacent pixels $q \in \mathcal{A}(p)$ in all the channels with $\lambda(I_{j,B}) = 1$, and $\mu_2(p)$ be the mean activation of the adjacent pixels $q \in \mathcal{A}(p)$ in all channels with $\lambda(I_{j,B}) = 2$. The values $\mu_l(p), l = 1, 2$ are:

$$\mu_l(p) = \frac{1}{N_l} \sum_{j,\lambda(I_{j,B})=l} \sum_{q \in \mathcal{A}(p)} I_{j,B}(q), \tag{10}$$

where N_l is the total number of adjacent pixels of p in channels with label l. Similarly, the variance $\sigma_l^2(p), l = 1, 2$ is:

$$\sigma_l^2(p) = \frac{1}{N_l} \sum_{j,\lambda(I_{j,B})=l} \sum_{q \in \mathcal{A}(p)} (I_{j,B}(q) - \mu_l(p))^2. \tag{11}$$

We estimate the probability of p be part of the object (background) given its activation $I_{j,B}(p)$ in a given channel by:

$$\phi_{j,l}(p) = \exp^{-\left(\frac{(I_{j,B}(p)-\mu_l(p))^2}{2\sigma_l^2(p)}\right)}. \tag{12}$$

The probability-based adaptive decoder assigns a weight vector $\vec{\alpha}^{pb}(p) = (\alpha_1^{pb}(p), \alpha_2^{pb}(p), \ldots, \alpha_{m'}^{pb}(p))$ per pixel rather than a single weight vector per channel, as the previous ones do. For each channel, it assumes consistency between the channel's label and those probability values.

$$\alpha_j^{pb}(p) = \begin{cases} +1, & \text{if } \lambda(I_{j,B}(p)) = 1 \ \& \ \phi_{j,1}(p) > \phi_{j,2}(p), \\ -1, & \text{if } \lambda(I_{j,B}(p)) = 2 \ \& \ \phi_{j,1}(p) < \phi_{j,2}(p), \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

4.5 Mean-based adaptive decoder (mb)

The mean-based decoder is a simplification of the probability-based adaptive decoder for the sake of efficiency, assuming that $\mu_1(p) > \mu_2(p)$ for object pixels and $\mu_1(p) < \mu_2(p)$ for background pixels, being this information consistent with the channel's label. The weight vector $\vec{\alpha}^{mb}(p) = (\alpha_1^{mb}(p), \alpha_1^{mb}(p), \cdots, \alpha_{m'}^{mb}(p))$ is defined by:

$$\alpha_j^{mb}(p) = \begin{cases} +1, & \text{if } \lambda(I_{j,B}(p)) = 1 \ \& \ \mu_1(p) > \mu_2(p), \\ -1, & \text{if } \lambda(I_{j,B}(p)) = 2 \ \& \ \mu_1(p) < \mu_2(p), \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

5 Experimental setup

Figure 5 illustrates the experimental setup. First, a dataset $\mathcal{Z}$ is randomly divided into two parts, $\mathcal{Z}_1$ and $\mathcal{Z}_2$, each with 50% of the samples (Figure 5a). $\mathcal{Z}_1$ is used to select the training set $\mathcal{T}$ image by image as the model (a FLIM CNN) is trained on $\mathcal{T}$, evaluated on $\mathcal{Z}_1 \backslash \mathcal{T}$, and a new image, among the ones not solved by the model, is selected for $\mathcal{T}$ (Figure 5b). Once $\mathcal{T}$ is defined with 3 or 4 images, depending on the split and user, the best CNN architecture on $\mathcal{Z}_1 \backslash \mathcal{T}$ is selected (Figure 5c). Its FLIM encoder and different adaptive decoders form the FLIM CNNs for testing on $\mathcal{Z}_2$ (Figure 5d). The set $\mathcal{T}$ is also used to train the decoders of two FLIM CNNs (Figure 5e) and fine-tune the lightweight models by backpropagation (Figure 5f), subsequently using the set $\mathcal{Z}_1 \backslash \mathcal{T}$ to select the best model for testing on $\mathcal{Z}_2$. At every stage, the saliency maps are post-processed to produce predicted binary masks and evaluate their metrics. The whole process is repeated three times for each user and dataset, with random splits of $\mathcal{Z}$, to obtain the mean and standard deviation of the metrics.

The following sections describe the datasets, the best architectures of the FLIM CNNs with adaptive decoders, the baselines, the evaluation metrics, the representative image selection procedure, and the post-processing of the saliency maps to obtain the predicted binary masks.

5.1 Datasets

Two SOD tasks were used to evaluate the methods: parasite egg detection from optical microscopy images and brain tumor detection from magnetic resonance (MR) images. The first task uses a dataset, *S. Mansoni*, of *Schistosoma Mansoni* eggs (see availability of data and materials). The second task uses 2D grayscale slices from the Brain Tumor Segmentation Challenge 2021, BraTS, a public dataset [Menze *et al*., 2014; Bakas *et al*., 2017, 2018]. A challenge in the *S. Mansoni* is the high amount of food debris (impurities) in the images, some of which are without eggs, and others contain impurities similar to the eggs in shape and color. In the BraTS dataset, the challenge comes from tumors with higher intensity, size, and shape variance.

S. Mansoni contains $1,219$ RGB images with dimensions of 400×400 pixels. BraTS contains $3,743$ 16-bit grayscale images with dimensions of 240×240 pixels.

To evaluate the generalization ability of the networks trained on the *S. Mansoni* dataset, we utilized six small datasets of different species of parasites: Ancylostoma (*Ancylostoma* spp.), Toxocara (*Toxocara* spp.), Trichuris (*Trichuris* spp.), Blastocystis (*Blastocystis hominis*), Entamoeba (*Entamoeba coli*), and Iodamoeba (*Iodamoeba bütschlii*). These datasets contain the following number of images: 320, 394, 391, 121, 100, and 134.

5.2 FLIM CNNs with adaptive decoders

Each user specifies the hyperparameters of each encoder block $b = 0, 1, \ldots, B-1$ as consisting of patch sizes, the number of kernels per marker, kernel-dilation factors, the number of desired kernels, pooling adjacency, pooling strides, and pooling type. The kernels are estimated as described in Section 3. The number of kernels per block equals the total number of markers multiplied by the number of kernels per marker. Note that the marker count varies between different splits and users, since it depends on how many kernels were annotated on each training image. The output of the last block is a feature map $\mathbf{I}^B$ to the input of one of the adaptive decoders, forming the following CNNs: FLIM$_{ts}$, FLIM$_{at}$, FLIM$_{lt}$, FLIM$_{pb}$, and FLIM$_{mb}$.

Both users start from an encoder with four blocks ($B = 4$) for parasite egg detection and three ($B = 3$) for brain tumor detection, as shown in Figure 6. Each block is composed of a

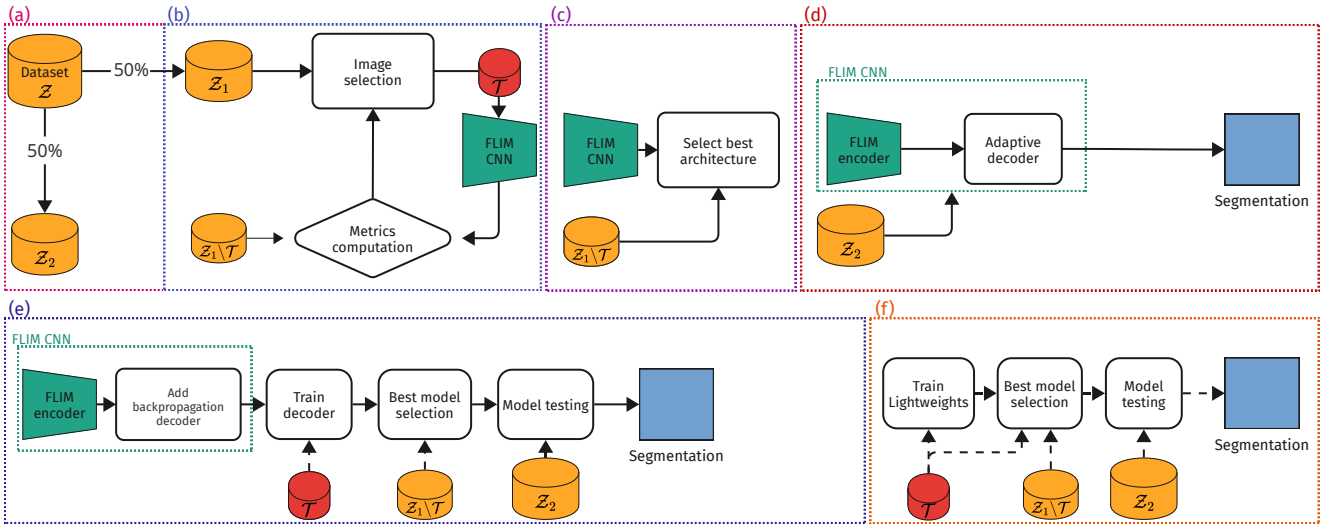


Figure 5. Pipeline used for the experiments. (a) The dataset $\mathcal{Z}$ is randomly divided into sets $\mathcal{Z}_1$ and $\mathcal{Z}_2$. (b) A few representative images are selected for the set $\mathcal{T}$, according to the model's performance on the validation set $\mathcal{Z}_1 \backslash \mathcal{T}$. (c) Once $\mathcal{T}$ is fixed, the best network architecture is found on the validation set. (d) The pre-trained FLIM encoder with each adaptive decoder is tested on $\mathcal{Z}_2$. (e) Each fixed-weight decoder is trained on $\mathcal{T}$, using the pre-trained FLIM encoder as the fixed backbone; the best models are found on $\mathcal{Z}_1 \backslash \mathcal{T}$ and tested on $\mathcal{Z}_2$. (f) Each pre-trained lightweight model is fine-tuned on $\mathcal{T}$; the best models are found on $\mathcal{Z}_1 \backslash \mathcal{T}$ and tested on $\mathcal{Z}_2$.

marker-based normalization layer, convolution with kernel size of 3×3, ReLU activation and pooling (average or max). Since the number of kernels per block varies for each user and split, the best CNN architecture per split is obtained by verifying the adaptive decoder and block $0 \leq b \leq B-1$ that produces the best metric F_β on $\mathcal{Z}_1 \backslash \mathcal{T}$ (Section 6.1).

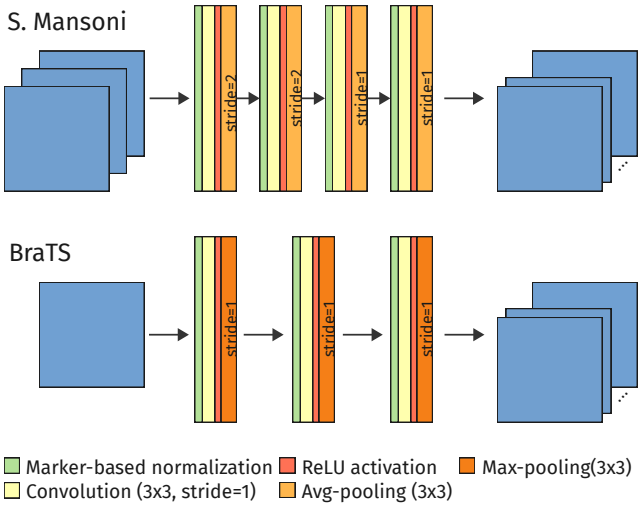


Figure 6. Maximum-depth encoders evaluated for parasite egg and brain tumor detection, with the operators used in each block.

5.3 Baselines

As baselines, we selected CNNs using the same FLIM encoder and substituting the adaptive decoders by decoders with fixed weights. We also selected three lightweight models from the state-of-the-art as baselines.

5.3.1 FLIM SOD models with fixed-weight decoders

Two decoders with fixed weights are point-wise convolutions followed by ReLU activation, one of them trained by backpropagation. The third decoder forms a U-shape network with the FLIM encoder, named U-Net$_{\text{FLIM}}$, trained by backpropagation.

Label-based fixed-weight decoder (lm) This decoder assumes that a kernel from a labeled marker will create a channel with the same label. Hence, we can assume that $\lambda(I_{j,B}) \in \{-1, +1\}$ indicates background (-1) or foreground $(+1)$ channel and define the weight vector of a label-based fixed-weight decoder as $\vec{\alpha}^{lm} = (\alpha_1^{lm}, \alpha_2^{lm}, \ldots, \alpha_{m'}^{lm})$, where $\alpha_j^{lm} = \lambda(I_{j,B})$, $j = 1, 2, \ldots, m'$. The CNN with lm is called FLIM$_{lm}$.

Backpropagation-based fixed-weight decoder (bp) We fixed the pre-trained FLIM encoder and optimized the weight vector $\vec{\alpha}^{bp} = (\alpha_1^{bp}, \alpha_2^{bp}, \ldots, \alpha_{m'}^{bp})$ of the fixed-weight decoder by backpropagation using the ground-truth and predicted masks of the images in $\mathcal{T}$. The weight vector $\vec{\alpha^{bp}}$ is initialized with the Xavier method. The loss function is the average between Dice and Binary Cross Entropy (BCE). Adam performed the optimization with a learning rate of 0.01 for 100 epochs. The CNN with bp is called FLIM$_{bp}$.

U-Net$_{\text{FLIM}}$ This CNN uses the pre-trained FLIM encoder frozen, considers a U-shape with skip connections from the output of each encoder's block to each decoder's block, and trains the model on $\mathcal{T}$. The FLIM encoder was set with four blocks for the *S. Mansoni* and three blocks for the brain tumor dataset (Figure 7). They use the same architecture and weights from the FLIM base architecture from Table 2, with each block being composed of marker-based normalization, convolution with kernel size 3×3 and stride 1, ReLU activation and pooling with size 3×3 and stride changed to

2. The decoder was composed of blocks consisting of a bilinear upsampling, convolution with kernel size 3×3 and stride 1, batch normalization, and ReLU activation. After the last block of the decoder, a sigmoid function was applied to generate the saliency map.

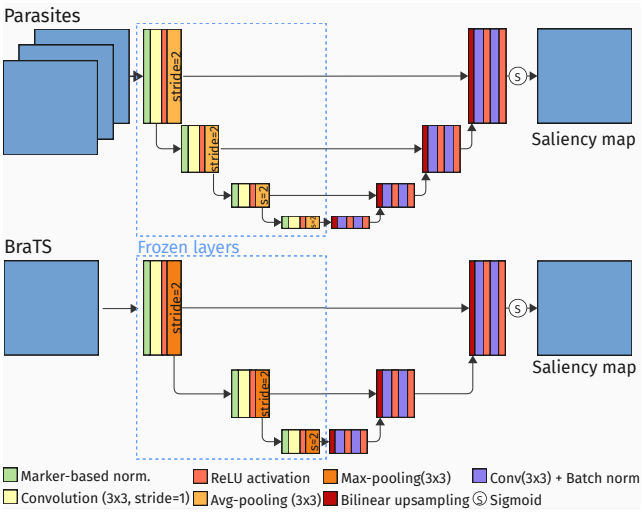


Figure 7. U-Net$_{\text{FLIM}}$ architectures with the operators used in each block.

The decoder's blocks used bilinear interpolation for upsampling, convolution, and batch normalization. The models were trained for 50 epochs, with a learning rate linearly decreasing from 0.01 to 0.00001. Horizontal and vertical flips were used for data augmentation.

5.3.2 Lightweight SOD models

We used the codes provided by the authors for the lightweight SOD models, SAMNet, MSCNet, and MEANet, with the default of almost all configurations. We used their pre-trained weights and fine-tuned them on set $\mathcal{T}$. The three models were trained for 100 epochs. For convergence, we had to increase their original learning rate. SAMNet, MSCNet, and MEANet were trained with learning rates of 0.005, 0.03, and 0.01, respectively. The batch size was set to $|\mathcal{T}|$. The best model for SAMNet was selected on $\mathcal{Z}_1 \backslash \mathcal{T}$, but MSCNet and MEANet can only select their best model on $\mathcal{T}$.

5.4 Evaluation metrics

The resulting saliency maps of the models were post-processed (Section 5.6) to obtain predicted binary masks. We used Mean Absolute Error (MAE) and F-measure (F_β) to compare those masks with the ground-truth binary masks of the images. The parameter β was set to 0.3, giving more weight to Precision, as it is usually done in the literature. Those metrics are obtained from Precision and Recall as follows.

$$\text{Precision} = \frac{TP}{TP+FP}, \text{Recall} = \frac{TP}{TP+FN}, \tag{15}$$

where TP, FP and FN are the true positive, false positive and false negative values obtained from the ground-truth G and predicted binary mask B, both in the resolution of the input image domain D_I.

F-measure (F_β) is given by:

$$F_\beta = \frac{(1+\beta^2)\text{Precision} \times \text{Recall}}{\beta^2 \text{Precision} + \text{Recall}}. \tag{16}$$

MAE is given by:

$$\text{MAE} = \frac{1}{|D_I|} \sum_{p \in D_I} |G(p) - B(p)|. \tag{17}$$

5.5 Representative image selection

We adopted a supervised approach to select a few representative images for $\mathcal{T}$ from the validation set $\mathcal{Z}_1 \backslash \mathcal{T}$, since our focus is on the decoders, and $\mathcal{T}$ is paramount to obtain effective CNN models. Our approach is similar to the one proposed in Cerqueira *et al*. [2024b]. The method selects one image per time, by evaluating the current model on $\mathcal{Z}_1 \backslash \mathcal{T}$. The selected images come from those not solved by the model trained on the current set $\mathcal{T}$ – i.e., the ones from $\mathcal{Z}_1 \backslash \mathcal{T}$ with the worst F_β measure. However, differently from Cerqueira *et al*. [2024b], if the last image inserted in $\mathcal{T}$ does not produce a better model, we remove it, and try another one.

The user decided when to stop the representative image selection process. In this work, the two users decided to stop the construction of $\mathcal{T}$ with three to four images, depending on the split. Given $\mathcal{T} = \{\}$ and $\mathcal{Z}_1$, this process can be described as follows (Algorithm 1).

Algorithm 1: Representative image selection

Input: $\mathcal{Z}_1, \mathcal{T} \leftarrow \{\}$
Output: $\mathcal{T} \neq \{\}$

Randomly select an image $z \in \mathcal{Z}_1$ and set $\mathcal{T} \leftarrow \mathcal{T} \cup \{z\}$;
Set $x_{prev} \leftarrow 0$ and $z_{prev} \leftarrow z$;
while *User is not satisfied* **do**
  Train a FLIM CNN with an adaptive decoder on $\mathcal{T}$;
  Evaluate the model on $\mathcal{Z}_1 \backslash \mathcal{T}$;
  Compute x as the average F_β on $\mathcal{Z}_1 \backslash \mathcal{T}$;
  Among the images with the lowest F_β, select a new image z from $\mathcal{Z}_1 \backslash \mathcal{T}$;
  if $x < x_{prev}$ **then**
    Set $\mathcal{T} \leftarrow \mathcal{T} \backslash \{z_{prev}\}$;
  else
    Set $\mathcal{T} \leftarrow \mathcal{T} \cup \{z\}$;
    Set $x_{prev} \leftarrow x$ and $z_{prev} \leftarrow z$;

For *S. Mansoni*, one user selected FLIM$_{pb}$ as the adaptive decoder, and the other selected FLIM$_{lm}$. This variation prevents bias and ensures that adaptive decoders do not have an unfair advantage over fixed-weight decoders. For the BraTS dataset, one user selected FLIM$_{ts}$ as the adaptive decoder, and the other selected FLIM$_{lm}$.

5.6 Post-processing

Post-processing differs as shown in Table 1. Since the delineation loss in the lightweight models aims to create predicted masks with reasonable boundary delineation, we applied Otsu's threshold to the saliency maps of both datasets. We then removed false positives by an area filter. The area range was estimated from the data observation in $\mathcal{Z}_1 \backslash \mathcal{T}$. This procedure was used for both datasets independently of the model, and it was enough for the FLIM CNNs in the case of the brain tumor dataset.

In the case of the *S. Mansoni*, we also removed components connected to the image's frame (mostly false positives). We noticed that the predicted binary masks from FLIM CNNs, except U-Net$_{\text{FLIM}}$ due to skip connection, are smaller than the object. We then applied morphological operations to create internal and external seeds and improve delineation by Dynamic Trees using the Lab color space of the original image [Bragantini *et al.*, 2018].

Table 1. Post-processing applied on the saliency maps of lightweight models, U-Net$_{\text{FLIM}}$, and the other FLIM models: OT is Otsu's Threshold; AF is Area Filtering using the range [minimum component area, maximum component area]; and DT is object delineation by Dynamic Trees [Bragantini *et al.*, 2018]

Dataset	Model	Post-processing
S. Mansoni	Lightweight	OT + AF[1000-9000]
	FLIM	OT + AF[1000-9000] + DT
	U-Net$_{\text{FLIM}}$	OT + AF[1000-9000]
BraTS	Lightweight	OT + AF[100-20000]
	FLIM	OT + AF[100-20000]
	U-Net$_{\text{FLIM}}$	OT + AF[100-20000]

6 Results and discussion

This section presents the results of our experiments for each user, A and B, using their respective training images and marker sets. The best FLIM network architectures (efficiency), quantitative results (effectiveness), and qualitative results are presented.

6.1 The best FLIM architectures

Table 2. Average number of parameters per user and model. FLIM networks that use a single decoder block are grouped for this measure.

Model	Users	# Parameters	
		S. Mansoni	BraTS
SAMNet	A	1.33 (M)	1.33 (M)
	B	1.33 (M)	1.33 (M)
MSCNet	A	3.26 (M)	3.26 (M)
	B	3.26 (M)	3.26 (M)
MEANet	A	3.27 (M)	3.27 (M)
	B	3.27 (M)	3.27 (M)
FLIM	A	195.05 ± 85.98 (K)	1405.55 ± 881.82 (K)
	B	57.12 ± 59.20 (K)	24.30 ± 26.51 (K)
U-Net$_{\text{FLIM}}$	A	2.29 ± 0.13 (M)	12.58 ± 1.17 (M)
	B	1.36 ± 0.51 (M)	00.34 ± 0.09 (M)

The numbers of blocks and kernels per block in the best FLIM encoder's architecture on $\mathcal{Z}_1 \backslash \mathcal{T}$ change for each dataset, user, and split, affecting the complexity of the CNN architecture. User A, for instance, selected more markers for the BraTS dataset, increasing the number of parameters of his FLIM networks. Table 2 shows the mean number of parameters in the best network architecture for each case. We are grouping all FLIM CNNs, except U-Net$_{\text{FLIM}}$ (due to its more complex decoder), for this efficiency measure with standard deviation. The lightweight models have a fixed number of parameters and they are hundreds of times more complex than FLIM CNNs, except U-Net$_{\text{FLIM}}$.

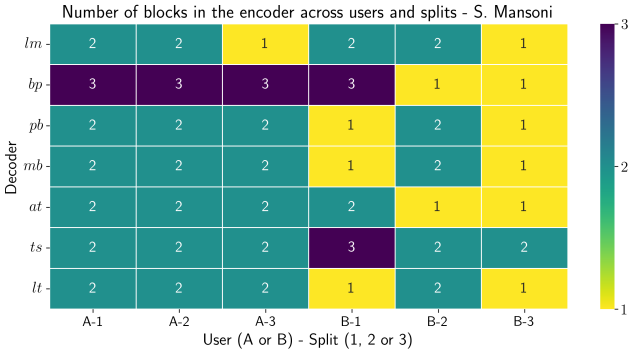


Figure 8. Number of blocks in the FLIM encoder for the *S. Mansoni* dataset.

Figures 8 and 9 show that the number of blocks in the best FLIM-CNN architecture varies a lot, depending on the dataset, split, decoder, and user. Note that an encoder with only two blocks was usually enough for *S. Mansoni*. For the BraTS dataset, the best encoder depth has more variance.

The encoder's depth also affects the decoders that combine the encoder's feature maps to create saliency maps. The deeper the encoder becomes, the fewer false positives in the saliency map are observed, while more details of the object's border are lost (Figure 4). This behavior may improve object detection up to a certain depth but requires object delineation to improve effectiveness, which was the case with *S. Mansoni*.

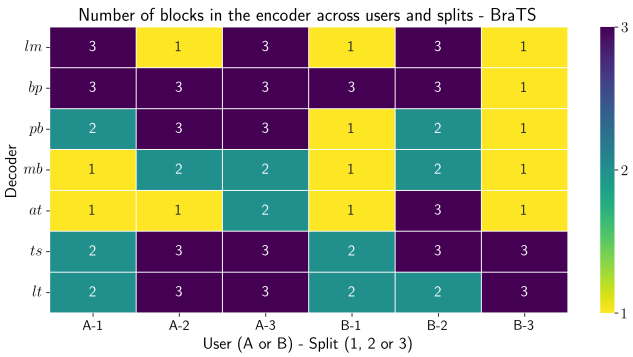


Figure 9. Number of blocks in the FLIM encoder for the BraTS dataset.

6.2 Quantitative results

Table 3 shows the mean and standard deviation of MEA and F_{beta} in the three splits per user on the validation sets $\mathcal{Z}_1 \backslash \mathcal{T}$. The best results are highlighted in green, the second

Table 3. Mean and standard deviation of MEA and F_β in the validation set. For each model and user, the best result is shown in **green**, the second best in **blue**, and the worst in **red**. The central horizontal line separates FLIM CNNs with adaptive decoders from the baselines. Arrows ↑ and ↓ denote higher and lower values are better, respectively.

Model	Users	*S. Mansoni*		BraTS	
		MAE↓	F_β ↑	MAE↓	F_β ↑
SAMNet	A	**0.038±0.036**	**0.428±0.234**	0.140±0.072	0.219±0.078
	B	**0.023±0.015**	**0.521±0.216**	0.121±0.062	0.225±0.029
MSCNet	A	0.017±0.009	0.615±0.147	**0.197±0.047**	0.246±0.032
	B	0.012±0.005	0.634±0.178	**0.161±0.020**	0.266±0.007
MEANet	A	0.023±0.017	0.538±0.120	0.049±0.003	**0.122±0.015**
	B	0.013±0.005	0.674±0.094	0.051±0.004	**0.131±0.022**
U-Net$_{\text{FLIM}}$	A	0.009±0.002	0.771±0.028	0.031±0.016	0.645±0.099
	B	0.011±0.004	0.710±0.087	0.049±0.037	0.630±0.097
FLIM$_{lm}$	A	**0.005±0.001**	**0.860±0.017**	0.018±0.001	0.659±0.021
	B	**0.005±0.001**	0.843±0.025	0.024±0.007	0.691±0.012
FLIM$_{bp}$	A	0.007±0.000	0.770±0.030	0.019±0.001	0.707±0.005
	B	0.008±0.001	0.737±0.025	0.023±0.007	0.693±0.014
FLIM$_{pb}$	A	**0.006±0.001**	**0.857±0.012**	0.019±0.000	0.703±0.011
	B	**0.006±0.001**	**0.847±0.015**	0.022±0.003	0.691±0.008
FLIM$_{mb}$	A	0.006±0.002	0.843±0.023	0.021±0.002	0.702±0.006
	B	0.006±0.001	**0.843±0.021**	0.025±0.007	0.694±0.017
FLIM$_{at}$	A	0.011±0.003	0.740±0.062	0.022±0.003	0.679±0.013
	B	0.014±0.006	0.660±0.108	0.024±0.006	0.694±0.017
FLIM$_{ts}$	A	0.010±0.000	0.747±0.015	**0.017±0.001**	**0.709±0.014**
	B	0.013±0.004	0.687±0.085	**0.020±0.004**	**0.697±0.037**
FLIM$_{lt}$	A	0.007±0.001	0.810±0.010	**0.018±0.001**	**0.721±0.004**
	B	0.009±0.004	0.760±0.089	**0.021±0.005**	**0.697±0.044**

Table 4. Mean and standard deviation of MEA and F_β in the test set. For each model and user, the best result is shown in **green**, the second best in **blue**, and the worst in **red**. The central horizontal line separates FLIM CNNs with adaptive decoders from the baselines. Arrows ↑ and ↓ denote higher and lower values are better, respectively.

Model	Users	*S. Mansoni*		BraTS	
		MAE↓	F_β ↑	MAE↓	F_β ↑
SAMNet	A	**0.038±0.036**	**0.422±0.241**	0.142±0.068	0.215±0.084
	B	**0.024±0.014**	**0.508±0.194**	0.123±0.063	0.221±0.038
MSCNet	A	0.016±0.009	0.604±0.125	**0.197±0.047**	0.248±0.028
	B	0.012±0.005	0.645±0.163	**0.161±0.020**	0.268±0.003
MEANet	A	0.023±0.018	0.538±0.137	0.050±0.004	**0.124±0.010**
	B	0.012±0.004	0.681±0.085	0.052±0.003	**0.126±0.019**
U-Net$_{\text{FLIM}}$	A	0.008±0.001	0.777±0.025	0.032±0.017	0.650±0.104
	B	0.011±0.004	0.706±0.063	0.049±0.038	0.636±0.103
FLIM$_{lm}$	A	**0.005±0.000**	**0.857±0.015**	0.018±0.001	0.678±0.040
	B	**0.005±0.000**	0.843±0.015	0.023±0.007	0.700±0.013
FLIM$_{bp}$	A	0.007±0.001	0.757±0.012	0.018±0.001	0.717±0.005
	B	0.008±0.001	0.733±0.025	0.023±0.007	0.704±0.016
FLIM$_{pb}$	A	**0.006±0.000**	**0.857±0.012**	0.019±0.001	0.711±0.010
	B	**0.006±0.001**	**0.850±0.010**	**0.021±0.003**	0.701±0.009
FLIM$_{mb}$	A	0.006±0.001	0.847±0.012	0.021±0.002	0.712±0.007
	B	0.006±0.001	**0.850±0.010**	0.025±0.007	0.702±0.019
FLIM$_{at}$	A	0.011±0.002	0.733±0.042	0.021±0.003	0.688±0.012
	B	0.015±0.006	0.647±0.101	0.024±0.006	0.703±0.018
FLIM$_{ts}$	A	0.010±0.001	0.747±0.015	**0.017±0.001**	**0.720±0.014**
	B	0.013±0.003	0.647±0.057	**0.019±0.004**	**0.706±0.039**
FLIM$_{lt}$	A	0.007±0.001	0.803±0.021	**0.017±0.002**	**0.731±0.005**
	B	0.009±0.005	0.753±0.086	0.021±0.005	**0.704±0.048**

best in blue, and the worst results in red. For *S. Mansoni*, FLIM$_{lm}$, FLIM$_{pb}$, and FLIM$_{mb}$ outperformed the other models in MAE and F_β for both users. According to both measures, these FLIM models are considerably more effective than the lightweight models, with SAMNet presenting the worst results. The good performance of FLIM$_{lm}$ indicates that the foreground and background activation channels created by the FLIM encoder consistently came from kernels with the same label, motivating the exploration of labeled markers for kernel estimation in future work.

Although FLIM$_{pb}$ and FLIM$_{mb}$, with the new decoders, present good MAE and F_β for BraTS, FLIM$_{ts}$ and FLIM$_{lt}$ obtained slightly better results, while MSCNet and MEANet presented the worst MAE and F_β, respectively. Similar results can be observed in the test set $\mathcal{Z}_2$ of both datasets (Table 4), demonstrating the adaptive decoders' generalization and robustness to address SOD problems with distinct characteristics.

Since the models were trained with very few images (three or four), a question may arise about the performance of the backpropagation-based baselines when a reasonable number of annotated images is available. For that, we trained the more complex baselines based on backpropagation on different subsamples of $\mathcal{Z}_1$. We tested them on $\mathcal{Z}_2$. More specifically, we create subsets $\mathcal{Z}_p \subset Z_1$, where $p \in \{0.05, 0.20, 0.40, 0.60, 0.80, 1.00\}$ represents the proportion of data used, such that $|D_p| = p \cdot |Z_1|$. For U-Net$_{\text{FLIM}}$, we use the encoder frozen and unfrozen to fine-tune the FLIM encoder with the higher number of annotated images in $\mathcal{Z}_1$.

Table 5 and Figure 10 (a)-(c) show that SAMNet and MSCNet could improve their results on *S. Mansoni*, achieving superior MAE and F_β to the others when trained with more than 20% of the $\mathcal{Z}_1$. However, they could not perform well on BraTS, as MEANet too, indicating that the number of annotated images in $\mathcal{Z}_1$ was still insufficient for these models to adapt to the different domain of data (MRI). U-Net$_{\text{FLIM}}$, both with the encoder frozen and unfrozen, improved F_β and

Table 5. Mean and standard deviation of MEA and F_β on the test set $\mathcal{Z}_2$ of the baseline backpropagation-based models, when trained with the entire set $\mathcal{Z}_1$. The results of the FLIM networks are presented with the encoder trained by each user, A and B. Arrows ↑ and ↓ denote higher and lower values are better, respectively.

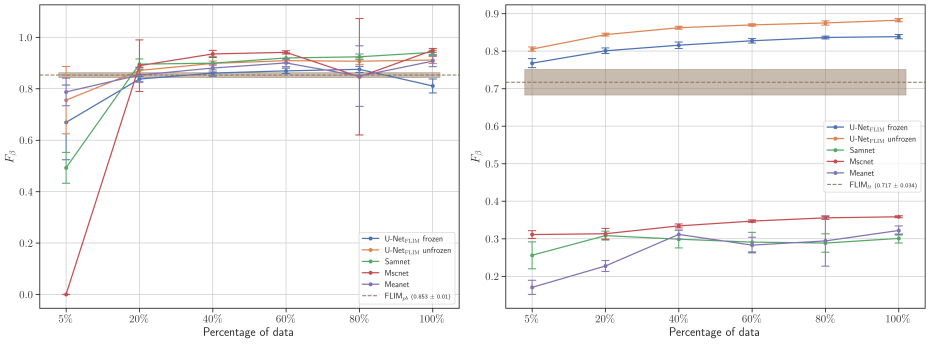
Model	*S. Mansoni*		BraTS	
	MAE↓	F_β ↑	MAE↓	F_β ↑
SAMNet	0.002±0.000	0.945±0.004	0.065±0.000	0.353±0.007
MSCNet	0.002±0.000	0.949±0.006	0.055±0.000	0.358±0.002
MEANet	0.003±0.000	0.907±0.019	0.048±0.001	0.322±0.011
U-Net$_{\text{FLIM}}$-Frozen (A)	0.004±0.000	0.798±0.024	0.009±0.000	0.841±0.005
U-Net$_{\text{FLIM}}$-Frozen (B)	0.004±0.000	0.824±0.016	0.010±0.000	0.836±0.004
U-Net$_{\text{FLIM}}$-Unfrozen (A)	0.002±0.000	0.912±0.014	0.007±0.000	0.885±0.000
U-Net$_{\text{FLIM}}$-Unfrozen (B)	0.002±0.000	0.910±0.010	0.007±0.000	0.879±0.003

MAE in both datasets, producing superior results to the others in BraTS.

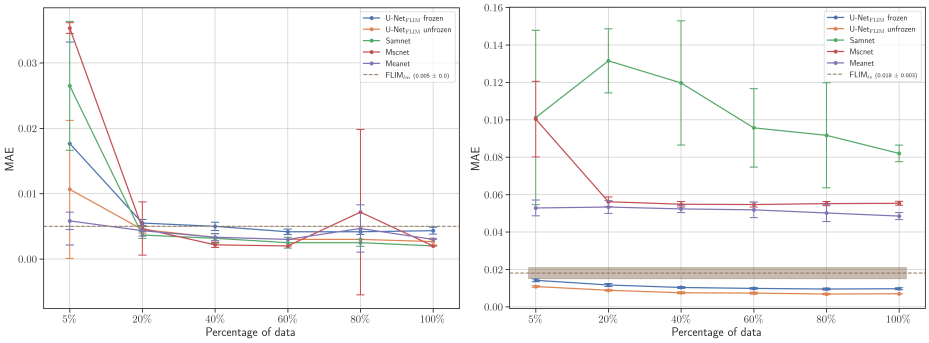
The results in Table 5 reveal the robustness of FLIM SOD networks with adaptive decoders trained from markers (weak annotation) on very few representative images. They also demonstrate that the small set $\mathcal{T}$ is representative to train effective FLIM SOD networks. The results for the U-Net$_{\text{FLIM}}$ also show that FLIM proves to be a suitable method for initializing the weights of the encoder, which is refined with more images and presents competitive results even in datasets that are harder for lightweight models (BraTS).

6.2.1 Generalization

To further assess our methodology and its ability to generalize, we extended the experimental evaluation to six additional small datasets. These evaluations followed a zero-shot approach, employing models that were trained on the *S. Mansoni* dataset without any additional training on the new datasets. The identical procedure was implemented for the baseline methods. The area filter has to be adjusted for this experiment: Ancylostoma [100,4000], Toxocara [1000,4000], Trichuris [1000,4000], Blastocystis [100,800], Entamoeba [1000,30000], and Iodamoeba [1000,3000]. We also modify the parameters of DT, with Ancylostoma, Toxocara, and Trichuris using dilation 30 and the others 20.



(a) *S. Mansoni* F_β (b) BraTS F_β



(c) *S. Mansoni* MAE (d) BraTS MAE

Figure 10. Backpropagation models trained in different amounts of data. (a) and (b) present the F_β for *S. Mansoni* and BraTs, respectively, while (c) and (d) presents the MAE.

The experimental results in zero-shot configuration are presented in Table 6 and Table 7. The findings demonstrate that lightweight architectures (SAMNet, MSCNet, and MEAnet) produce the lowest F_β across all baseline methods, with MSCNet exhibiting a performance gap of approximately 54 times compared to the optimal results in the dataset (FLIMts$_{ts}$). This pattern is consistently observed across all evaluated datasets.

Among the FLIM-based CNNs employing fixed decoders, we similarly observe challenges in generalization capability, with performance levels falling short of the peak results achieved within each dataset. Nevertheless, these results demonstrate superior performance compared to the lightweight architectures (e.g., showing only a 1.12-fold difference in the Iodamoeba dataset between FLIM$_{lm}$ and FLIM$_{pb}$), indicating that despite utilizing fixed weights acquired during encoder training, FLIM successfully extracted meaningful features from these datasets.

For FLIM-based CNNs incorporating adaptive decoders, Table 6 and Table 7 (lower section) reveal significantly enhanced generalization capabilities, demonstrated by the consistent achievement of first and second-best performance across all datasets by the adaptive decoders.

Analyzing the performance outcomes, no single decoder consistently delivers optimal results, as different decoders demonstrate superior performance on different datasets. Nevertheless, FLIM$_{pb}$ attains competitive results across four of the six datasets, comparable to the performance of FLIM$_{lt}$.

FLIM$_{pb}$ employs kernel labels while refining them through pixel-wise probability computation and local image region analysis. This approach enables the decoder to identify characteristics that remain invisible when using labels exclusively, supported by the observation that FLIM$_{lm}$ produces inferior results compared to FLIM$_{pb}$.

FLIM$_{lt}$ similarly demonstrates strong performance on four datasets, attributed to its focus on feature maps produced by kernels designated as foreground (object) while disregarding background feature maps. Given the balanced distribution of markers and objects in this study, this approach helps eliminate potentially spurious background activations while emphasizing object-related features.

The results obtained in these experiments highlight the necessity of different decoders, as they can deliver varying performance levels depending on the specific dataset characteristics. The results also demonstrate superior generalization ability of the FLIM plus adaptive decoders framework compared to baselines that rely on fixed weights. The implementation of adaptive weights that are modified based on input data provides enhanced network flexibility for generalization, enabling effective performance on related datasets containing classes absent from the original training set.

6.3 Qualitative results

Figure 11 illustrates the improvement in salient object detection for an example from *S. Mansoni* when post-processing is applied to the adaptive decoder's output. Note that Dynamic Trees is responsible for improving object delineation. Figure 11a shows the original image (above) and its ground-truth mask (below). The superior row shows the saliency maps at the adaptive decoder's output for each network, and the inferior row shows the respective results from post-processing for

Table 6. Mean and standard deviation of MEA in the additional datasets for 0-shot execution. For each model, the best result is shown in **green**, the second best in **blue**, and the worst in **red**. The central horizontal line separates FLIM CNNs with adaptive decoders from the baselines. Arrows ↑ and ↓ denote higher and lower values are better, respectively.

Model	Ancylostoma	Toxocara	Trichuris	Blastocystis	Entamoeba	Iodamoeba
	MAE↓	MAE↓	MAE↓	MAE↓	MAE↓	MAE↓
SAMNet	**0.055±0.030**	**0.032±0.008**	**0.024±0.016**	**0.039±0.010**	**0.129±0.078**	**0.030±0.014**
MSCNet	**0.015±0.007**	0.024±0.006	0.014±0.006	0.022±0.001	0.063±0.010	0.022±0.004
MEANet	0.022±0.011	0.026±0.005	0.013±0.002	0.023±0.003	0.064±0.017	0.021±0.006
U-Net$_{\text{FLIM}}$	0.021±0.007	0.026±0.009	0.012±0.006	0.022±0.001	0.044±0.013	0.016±0.003
FLIM$_{lm}$	**0.016±0.005**	0.012±0.004	0.007±0.003	**0.021±0.000**	0.053±0.013	0.012±0.003
FLIM$_{bp}$	0.016±0.009	0.013±0.007	0.007±0.002	**0.021±0.001**	0.052±0.016	0.025±0.005
FLIM$_{pb}$	0.017±0.005	**0.009±0.002**	**0.003±0.001**	0.022±0.001	**0.040±0.008**	**0.010±0.001**
FLIM$_{mb}$	0.018±0.006	**0.009±0.004**	**0.004±0.001**	0.022±0.000	**0.043±0.008**	**0.010±0.001**
FLIM$_{at}$	0.021±0.007	0.010±0.004	0.004±0.002	0.023±0.002	0.061±0.013	0.015±0.005
FLIM$_{ts}$	0.017±0.005	0.010±0.002	0.005±0.001	0.022±0.001	0.059±0.012	0.015±0.005
FLIM$_{lt}$	0.018±0.007	0.009±0.004	0.004±0.002	0.025±0.003	0.039±0.009	0.013±0.004

Table 7. Mean and standard deviation of F_β in the additional datasets for 0-shot execution. For each model, the best result is shown in **green**, the second best in **blue**, and the worst in **red**. The central horizontal line separates FLIM CNNs with adaptive decoders from the baselines. Arrows ↑ and ↓ denote higher and lower values are better, respectively.

Model	Ancylostoma	Toxocara	Trichuris	Blastocystis	Entamoeba	Iodamoeba
	F_β ↑	F_β ↑	F_β ↑	F_β ↑	F_β ↑	F_β ↑
SAMNet	0.033±0.013	0.086±0.111	0.237±0.176	**0.043±0.034**	0.402±0.171	**0.346±0.128**
MSCNet	**0.010±0.013**	0.086±0.132	**0.076±0.119**	0.059±0.066	**0.351±0.131**	0.402±0.181
MEANet	0.056±0.123	**0.060±0.072**	0.152±0.245	0.068±0.068	0.361±0.237	0.375±0.233
U-Net$_{\text{FLIM}}$	0.135±0.160	0.148±0.230	0.348±0.355	0.076±0.056	0.591±0.179	0.651±0.159
FLIM$_{lm}$	0.330±0.081	0.522±0.206	0.403±0.328	0.177±0.116	0.422±0.167	0.723±0.094
FLIM$_{bp}$	0.395±0.240	0.473±0.307	0.395±0.251	0.067±0.084	0.450±0.211	0.182±0.163
FLIM$_{pb}$	0.438±0.075	**0.667±0.073**	0.808±0.068	**0.293±0.087**	**0.605±0.112**	**0.787±0.029**
FLIM$_{mb}$	0.440±0.085	**0.672±0.117**	0.787±0.120	0.258±0.092	0.565±0.110	**0.783±0.032**
FLIM$_{at}$	0.497±0.095	0.642±0.130	**0.845±0.042**	0.162±0.119	0.345±0.168	0.668±0.119
FLIM$_{ts}$	**0.540±0.071**	0.627±0.101	0.725±0.169	0.162±0.127	0.370±0.155	0.630±0.171
FLIM$_{lt}$	**0.527±0.100**	0.662±0.127	**0.835±0.074**	**0.310±0.083**	**0.647±0.127**	0.727±0.084

each network (Figures 11b-11g). The border of the ground-truth mask is shown in color to illustrate accuracy. The presented results are good, but errors in delineation may occur when both the parasite egg and connected impurity components are enhanced in the saliency map (e.g., Figure 12j, above).

Figures 12 and 13 present qualitative results comparing baseline methods with FLIM networks using adaptive decoders. The networks were trained with markers of user A (top row) and B (bottom row) on *S. Mansoni* and BraTS datasets. These images demonstrate successful cases of the proposed method and the comparative methods for object delineation.

For the *S. Mansoni* dataset(Figure 12), both the comparative methods and the proposed approaches achieve satisfactory delineation results. However, the Lightweight models occasionally fail to converge when trained on limited data. Figure 14 illustrates these failure cases, with examples (c) and (e) representing the most critical convergent failures where the models cannot effectively learn from the available training data.

The BraTS (13) reveals a different behavior from the Lightweight models, which consistently fail to converge with the available training images. Among the FLIM-based models, U-Net$_{\text{FLIM}}$ (Figure 13e) demonstrates inconsistent performance: it successfully combines feature maps from one user's encoder (top row). However, it fails with the other user's data (bottom row), which suggests that the decoder requires additional training data for optimal performance. In contrast, other FLIM-based methods, including both comparative approaches (f-g) and adaptive decoders (h-l), successfully delineate tumors across the dataset, demonstrating robust delineation performance.

7 Conclusion

This work presented five adaptive decoders for FLIM networks, evaluating them in two SOD tasks: (a) parasite egg detection in optical microscopy images (dataset *S. Mansoni*) and (b) brain tumor detection in magnetic resonance slices (dataset BraTS). The adaptive decoders FLIM$_{ts}$, FLIM$_{at}$, and FLIM$_{lt}$ were previously presented in Soares *et al*. [2024], while FLIM$_{pb}$ and FLIM$_{mb}$ were introduced here. Unlike the previous ones, FLIM$_{pb}$ and FLIM$_{mb}$ estimate adaptive weights per pixel, being more complex than a simple point-wise convolution followed by activation.

The work demonstrated that one can create efficient and effective networks for SOD from very few representative images (three-four), by combining a FLIM encoder with an adaptive decoder. It described how representative images were selected, and compared FLIM networks with adaptive decoders against three pre-trained lightweight models, SAMNet, MSCNet, and MEANet, two FLIM networks with decoders trained by backpropagation, U-Net$_{FLIM}$ and FLIM$_{bp}$, and one network, FLIM$_{lm}$, whose marker labels define the decoder's weights. FLIM$_{pb}$, FLIM$_{mb}$, and FLIM$_{lm}$ outperformed the others in *S. Mansoni*, while FLIM$_{ts}$ and FLIM$_{lt}$ were the best models in BraTS. Indeed, compared to the lightweight models, all FLIM networks, including the baseline ones, presented better results in both datasets. Even when trained on the full dataset $\mathcal{Z}_1$, most lightweight models –except on *S. Mansoni*– failed to improve or match the performance of FLIM networks.

To conclude, the lightweight models likely require considerably more annotated samples to achieve meaningful performance gains. In contrast, FLIM networks equipped with adaptive decoders offer a superior alternative, particularly in resource-constrained settings.

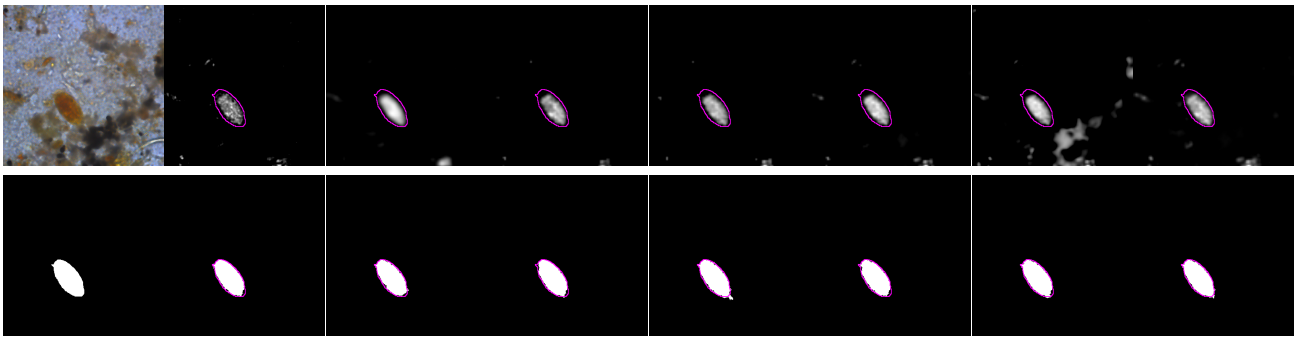
FLIM-based models with adaptive decoder presented superior generalization ability to different datasets of the same application domain when compared with the models with fixed weights, both FLIM-based and Lightweight ones.

The post-processing object delineation worked effectively for *S. Mansoni*, but not for BraTS, likely due to the choice of delineation algorithm–a subject warranting further investigation. Additionally, combining saliency maps from different encoder blocks merits deeper exploration, as early blocks often capture object boundaries more accurately, while later blocks reduce false positives by better localizing the object. Future work will also focus on developing new adaptive decoders for broader applications.

Declarations

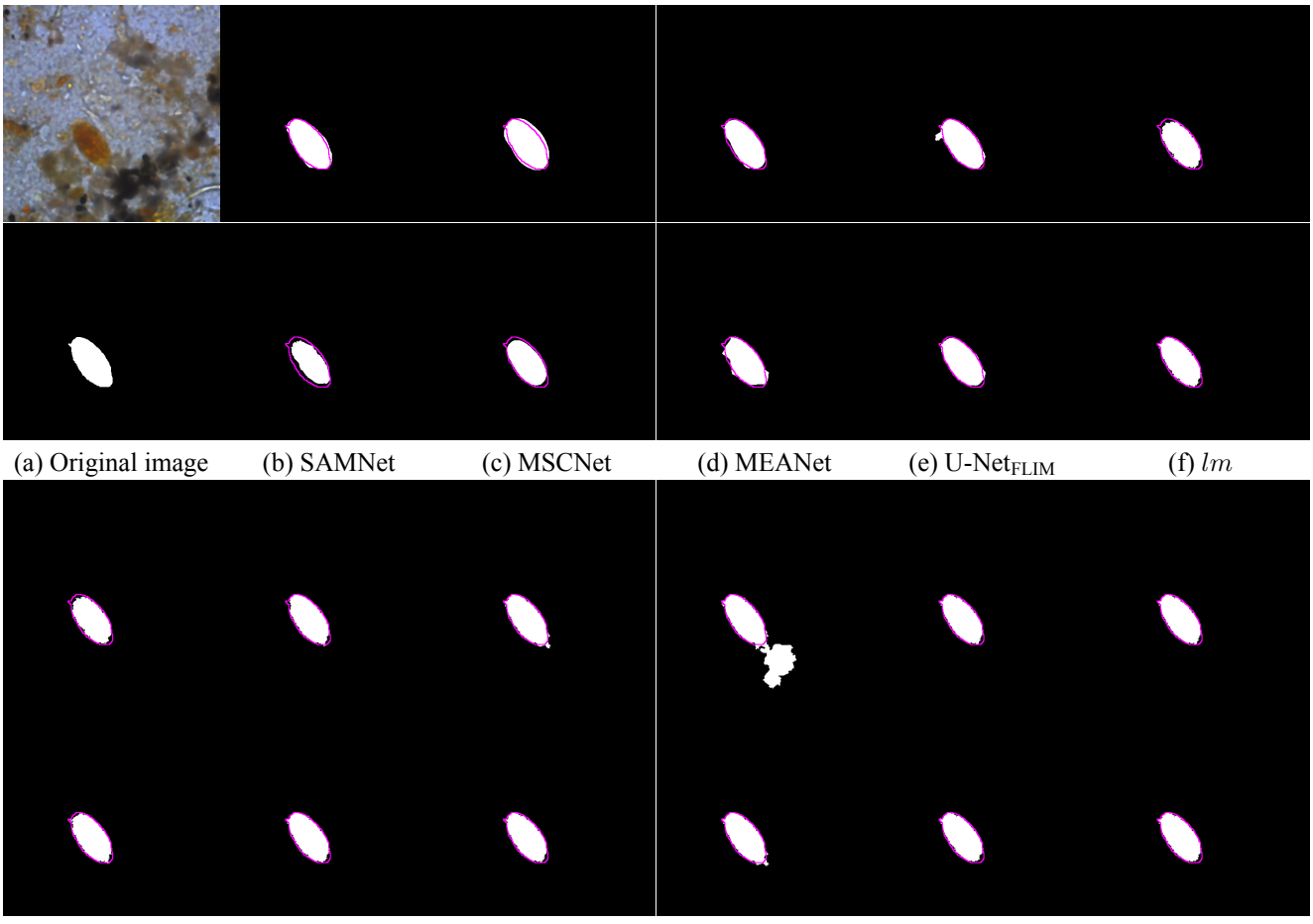
Authors' Contributions

GJS, SJFG, LN, and AXF conceptualized this study. JFG contributed to data curation. GJS and MAC performed the experiments. AXF supervised GJS. GJS wrote the original draft. AXF, LN, and SJFG reviewed and edited the text. All authors read and approved the final manuscript.



(a) Original image (b) lm (c) bp (d) pb (e) mb (f) at (g) ts (h) ts

Figure 11. (a) Original image (above) and ground-truth mask (below) of an example from the *S. Mansoni*. (b-g) show the improvement in salient object detection when comparing the adaptive decoders' output (above) with the results of post-processing (below) for each network.



(a) Original image (b) SAMNet (c) MSCNet (d) MEANet (e) U-Net$_{\text{FLIM}}$ (f) lm

(g) bp (h) pb (i) mb (j) at (k) ts (l) lt

Figure 12. (a) Original image (above) and ground-truth mask (below) of an example from the *S. Mansoni*. (b-g) show the qualitative results of the baseline networks trained by users A (above) and B (below). (h-l) show the qualitative results of the networks with adaptive decoders trained by users A (above) and B (below).

Competing interests

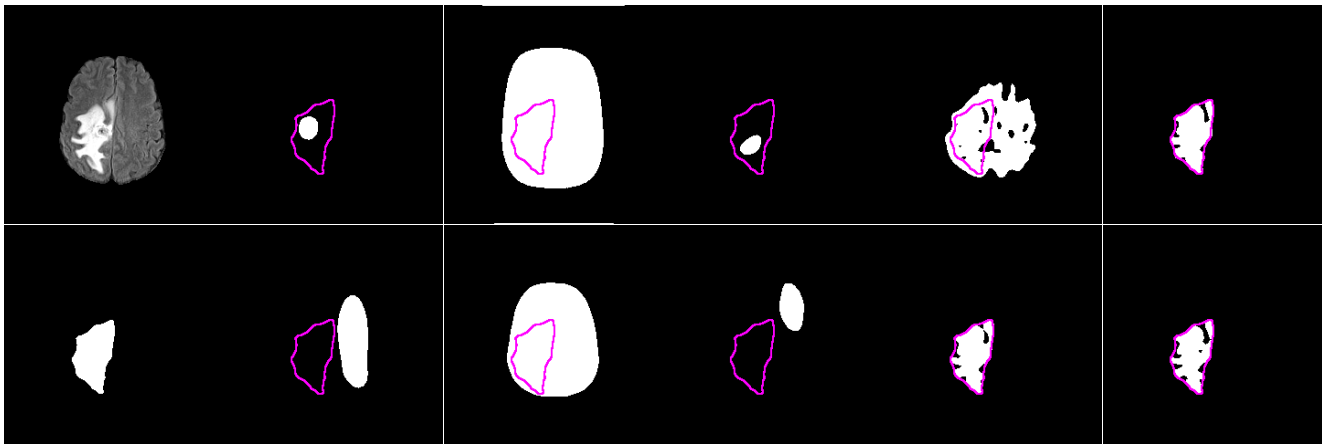
The authors declare that they have no competing interests.

Acknowledgements

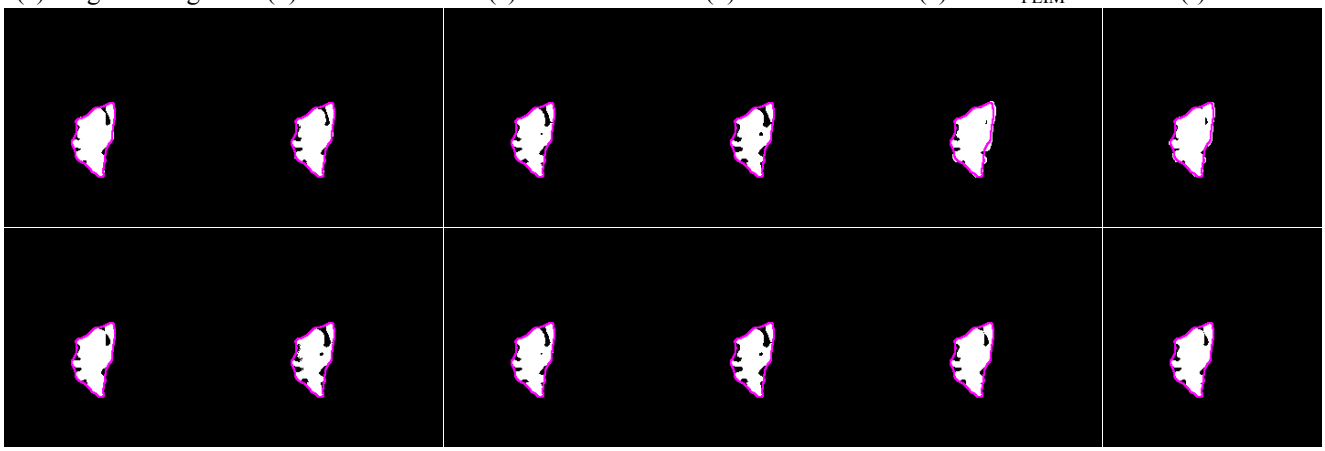
The authors acknowledge grants from Quinto Andar, FAPESP (2023/14427-8, 2023/09210-0 and 2013/07375-0), CNPq (407242/2021-0, 306573/2022-9, 442950/2023-3, 304711/2023-3), Labex Bézout/ANR, CAPES (88887.947543/2024-00, STIC-AMSUD 88887.878869/2023-00), and FAPEMIG (APQ-01079-23, PCE-00417-24 and APQ-05058-23).

Availability of data and materials

The *S. Mansoni* dataset is available at `https://github.com/LIDS-Datasets/schistossoma-eggs`. The implementation of the FLIM encoders and decoders is publicly available at: `https://github.com/LIDS-UNICAMP/flim_ad`.

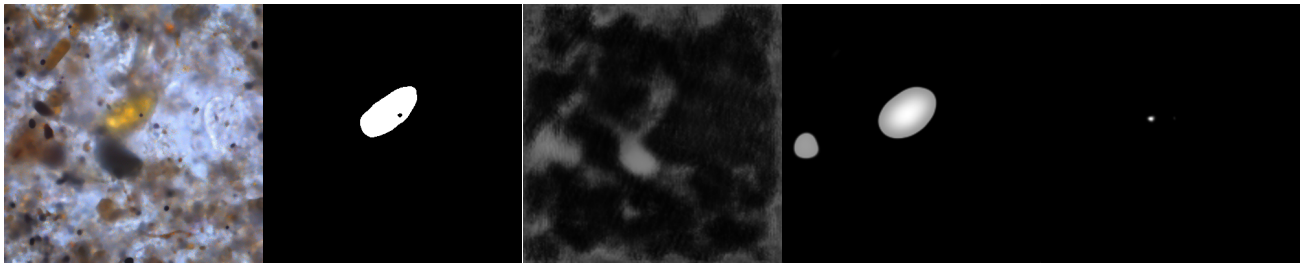


(a) Original image (b) SAMNet (c) MSCNet (d) MEANet (e) U-Net$_{\text{FLIM}}$ (f) lm



(g) bp (h) pb (i) mb (j) at (k) ts (l) lt

Figure 13. (a) Original image (above) and ground-truth mask (below) of an example from the BraTS dataset. (b-g) show the qualitative results of the baseline networks trained by users A (above) and B (below). (h-l) show the qualitative results of the networks with adaptive decoders trained by users A (above) and B (below).



(a) Original image (b) Ground truth (c) SAMNet (d) MSCNet (e) MEANet

Figure 14. This example illustrates typical convergence errors of the lightweight models in a training image.

References

Achanta, R., Estrada, F., Wils, P., and Süsstrunk, S. (2008). Salient region detection and segmentation. In *Computer Vision Systems: 6th International Conference, ICVS 2008 Santorini, Greece, May 12-15, 2008 Proceedings 6*, pages 66–75. Springer. DOI: 10.1007/978-3-540-79547-6_7.

Bakas, S., Akbari, H., Sotiras, A., Bilello, M., Rozycki, M., Kirby, J. S., Freymann, J. B., Farahani, K., and Davatzikos, C. (2017). Advancing the cancer genome atlas glioma mri collections with expert segmentation labels and radiomic features. *Scientific data*, 4(1):1–13. DOI: 10.1038/sdata.2017.117.

Bakas, S., Reyes, M., Jakab, A., Bauer, S., Rempfler, M., Crimi, A., Shinohara, R. T., Berger, C., Ha, S. M., Rozycki, M., *et al*. (2018). Identifying the best machine learning algorithms for brain tumor segmentation, progression assessment, and overall survival prediction in the brats challenge. *arXiv preprint arXiv:1811.02629*. DOI: 10.17863/CAM.38755.

Borji, A., Cheng, M.-M., Hou, Q., Jiang, H., and Li, J. (2019). Salient object detection: A survey. *Computational visual media*, 5:117–150. DOI: 10.1007/s41095-019-0149-9.

Bragantini, J., Martins, S. B., Castelo-Fernandez, C., and Falcão, A. X. (2018). Graph-based image segmentation using dynamic trees. In *Iberoamerican Congress on Pattern Recognition*, pages 470–478. Springer. DOI: 10.1007/978-3-030-13469-3_55.

Cerqueira, M. A., Sprenger, F., Teixeira, B. C., and Falcão, A. X. (2023). Building brain tumor segmentation networks with user-assisted filter estimation and selection. In *18th International Symposium on Medical Information Processing and Analysis*, volume 12567, pages 202–211. SPIE. DOI: 10.1117/12.2669770.

Cerqueira, M. A., Sprenger, F., Teixeira, B. C., and Falcão, A. X. (2024a). Interactive image selection and training for brain tumor segmentation network. In *2024 46th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 1–4. IEEE. DOI: 10.1109/EMBC53108.2024.10781962.

Cerqueira, M. A., Sprenger, F., Teixeira, B. C., Guimarães, S. J. F., and Falcão, A. X. (2024b). Interactive ground-truth-free image selection for flim segmentation encoders. In *2024 37th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 1–6. IEEE. DOI: 10.1109/SIBGRAPI62404.2024.10716300.

Chang, K.-Y., Liu, T.-L., Chen, H.-T., and Lai, S.-H. (2011). Fusing generic objectness and visual saliency for salient object detection. In *2011 International Conference on Computer Vision*, pages 914–921. IEEE. DOI: 10.1109/ICCV.2011.6126333.

Chen, F., Li, S., Han, J., Ren, F., and Yang, Z. (2024). Review of lightweight deep convolutional neural networks. *Archives of Computational Methods in Engineering*, 31(4):1915–1937. DOI: 10.1007/s11831-023-10032-z.

Cheng, M.-M., Mitra, N. J., Huang, X., Torr, P. H., and Hu, S.-M. (2014). Global contrast based salient region detection. *IEEE transactions on pattern analysis and machine intelligence*, 37(3):569–582. DOI: 10.1109/TPAMI.2014.2345401.

De Souza, I. E., Benato, B. C., and Falcão, A. X. (2020). Feature learning from image markers for object delineation. In *2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 116–123. IEEE. DOI: 10.1109/SIBGRAPI51738.2020.00024.

De Souza, I. E. and Falcão, A. X. (2020). Learning cnn filters from user-drawn image markers for coconut-tree image classification. *IEEE Geoscience and Remote Sensing Letters*, 19:1–5. DOI: 10.1109/LGRS.2020.3020098.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. IEEE. DOI: 10.1109/CVPR.2009.5206848.

Han, K., Wang, Y., Tian, Q., Guo, J., Xu, C., and Xu, C. (2020). Ghostnet: More features from cheap operations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1580–1589. DOI: 10.1109/CVPR42600.2020.00165.

He, S., Lau, R. W., Liu, W., Huang, Z., and Yang, Q. (2015). Supercnn: A superpixelwise convolutional neural network for salient object detection. *International journal of computer vision*, 115:330–344. DOI: 10.1007/s11263-015-0822-0.

He, X., Zhao, K., and Chu, X. (2021). Automl: A survey of the state-of-the-art. *Knowledge-based systems*, 212:106622. DOI: 10.1016/j.knosys.2020.106622.

Itti, L., Koch, C., and Niebur, E. (2002). A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 20(11):1254–1259. DOI: 10.1109/34.730558.

Jiang, P., Ling, H., Yu, J., and Peng, J. (2013). Salient region detection by ufo: Uniqueness, focusness and objectness. In *Proceedings of the IEEE international conference on computer vision*, pages 1976–1983. DOI: 10.1109/ICCV.2013.248.

Joao., L., Cerqueira., M., Benato., B., and Falcão., A. (2024). Understanding marker-based normalization for flim networks. pages 612–623. DOI: 10.5220/0012385900003660.

Joao, L. d. M., Santos, B. M. d., Guimaraes, S. J. F., Gomes, J. F., Kijak, E., Falcão, A. X., *et al*. (2023). A flyweight cnn with adaptive decoder for schistosoma mansoni egg detection. *arXiv preprint arXiv:2306.14840*. DOI: 10.48550/arXiv.2306.14840.

Lee, G., Tai, Y.-W., and Kim, J. (2016). Deep saliency with encoded low level distance map and high level features. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 660–668. DOI: 10.1109/CVPR.2016.78.

Li, G., Liu, Z., Zhang, X., and Lin, W. (2023). Lightweight salient object detection in optical remote-sensing images via semantic matching and edge alignment. *IEEE Transactions on Geoscience and Remote Sensing*, 61:1–11. DOI: 10.1109/TGRS.2023.3235717.

Li, W., Zhang, Y., Shi, W., and Coleman, S. (2022). A cam-guided parameter-free attention network for person re-identification. *IEEE Signal Processing Letters*, 29:1559–1563. DOI: 10.1109/LSP.2022.3186273.

Liang, B. and Luo, H. (2024). Meanet: An effective and lightweight solution for salient object detection in optical remote sensing images. *Expert Systems with Applications*, 238:121778. DOI: 10.1016/j.eswa.2023.121778.

Lin, Y., Sun, H., Liu, N., Bian, Y., Cen, J., and Zhou, H. (2022). A lightweight multi-scale context network for salient object detection in optical remote sensing images. In *2022 26th international conference on pattern recognition (ICPR)*, pages 238–244. IEEE. DOI: 10.1109/ICPR56361.2022.9956350.

Liu, N., Han, J., and Yang, M.-H. (2018). Picanet: Learning pixel-wise contextual attention for saliency detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3089–3098. DOI: 10.1109/CVPR.2018.00326.

Liu, Y., Zhang, X.-Y., Bian, J.-W., Zhang, L., and Cheng, M.-M. (2021). Samnet: Stereoscopically attentive multi-scale network for lightweight salient object detection. *IEEE Transactions on Image Processing*, 30:3804–3814. DOI: 10.1109/TIP.2021.3065239.

Menze, B. H., Jakab, A., Bauer, S., Kalpathy-Cramer, J., Farahani, K., Kirby, J., Burren, Y., Porz, N., Slotboom, J., Wiest, R., *et al*. (2014). The multimodal brain tumor image segmentation benchmark (brats). *IEEE transactions on medical imaging*, 34(10):1993–2024. DOI: 10.1109/TMI.2014.2377694.

Qin, X., Zhang, Z., Huang, C., Dehghan, M., Zaiane, O. R., and Jagersand, M. (2020). U2-net: Go-

ing deeper with nested u-structure for salient object detection. *Pattern recognition*, 106:107404. DOI: 10.1016/j.patcog.2020.107404.

Qin, X., Zhang, Z., Huang, C., Gao, C., Dehghan, M., and Jagersand, M. (2019). Basnet: Boundary-aware salient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7479–7489. DOI: 10.1109/CVPR.2019.00766.

Salvagnini, F. C. R., Gomes, J. F., Santos, C. A., Guimarães, S. J. F., and Falcão, A. X. (2024). Improving flim-based salient object detection networks with cellular automata. In *2024 37th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 1–6. IEEE. DOI: 10.1109/SIBGRAPI62404.2024.10716266.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520. DOI: 10.1109/CVPR.2018.00474.

Soares, G. J., Cerqueira, M. A., Guimaraes, S. J. F., Gomes, J. F., and Falcão, A. X. (2024). Adaptive decoders for flim-based salient object detection networks. In *2024 37th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 1–6. IEEE. DOI: 10.1109/SIBGRAPI62404.2024.10716333.

Ullah, I., Jian, M., Hussain, S., Guo, J., Yu, H., Wang, X., and Yin, Y. (2020). A brief survey of visual saliency detection. *Multimedia Tools and Applications*, 79:34605–34645. DOI: 10.1007/s11042-020-08849-y.

Wang, L., Lu, H., Ruan, X., and Yang, M.-H. (2015). 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. DOI: 10.1109/CVPR.2015.7298938.

Wang, Z., Zhang, Y., Liu, Y., Qin, C., Coleman, S. A., and Kerr, D. (2023a). Larnet: Towards lightweight, accurate and real-time salient object detection. *IEEE Transactions on Multimedia*, 26:5207–5222. DOI: 10.1109/TMM.2023.3330082.

Wang, Z., Zhang, Y., Liu, Y., Zhu, D., Coleman, S. A., and Kerr, D. (2023b). Elwnet: An extremely lightweight approach for real-time salient object detection. *IEEE Transactions on Circuits and Systems for Video Technology*, 33(11):6404–6417. DOI: 10.1109/TCSVT.2023.3269951.

Zhang, X., Zhou, X., Lin, M., and Sun, J. (2018). Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6848–6856. DOI: 10.1109/CVPR.2018.00716.

Zhao, R., Ouyang, W., Li, H., and Wang, X. (2015). Saliency detection by multi-context deep learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1265–1274. DOI: 10.1109/CVPR.2015.7298731.

Zhou, X., Shen, K., and Liu, Z. (2024). Admnet: Attention-guided densely multi-scale network for lightweight salient object detection. *IEEE Transactions on Multimedia*. DOI: 10.1109/TMM.2024.3413529.

Zhu, W., Liang, S., Wei, Y., and Sun, J. (2014). Saliency optimization from robust background detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2814–2821. DOI: 10.1109/CVPR.2014.360.