Adaptive Decoders for FLIM-based Salient Object Detection Networks

Gilson Junior Soares*, Matheus Abrantes Cerqueira*, Silvio Jamil F. Guimaraes†,
Jancarlo F. Gomes‡ and Alexandre X. Falcão*
*Institute of Computing, University of Campinas, Campinas, Brazil
†School of Medical Sciences, University of Campinas, Campinas, Brazil
‡Institute of Computing Sciences, Pontifical Catholic University of Minas Gerais, Belo Horizonte, Brazil

***Abstract*—Salient Object Detection (SOD) methods based on deep learning have succeeded, usually at the price of abundantly annotated data and intensive computational resources. Such limitations have motivated the development of lightweight models, but they are still pre-trained on large datasets, and their adaptation under labeled data scarcity is challenging. In this context, Feature Learning from Image Markers (FLIM) is a methodology under investigation to create convolutional encoders with minimal user effort in data annotation. Flyweight networks based on a FLIM encoder followed by an *adaptive decoder*, which is a point-wise convolution with adaptive weights for each image followed by activation, achieved state-of-the-art results for SOD recently. In this work, we propose four strategies for computing adaptive weights based on (i) channel-tri-state detection, (ii) labeled markers, (iii) channel attention, and (iv) a hybrid solution using the tri-state and labeled-marker decoders. An assessment on two medical datasets between FLIM-based SOD networks with the proposed adaptive decoders, three state-of-the-art lightweight models and a U-shaped network with a FLIM encoder has shown that the results favor FLIM networks, with the hybrid solution being the most promising option.**

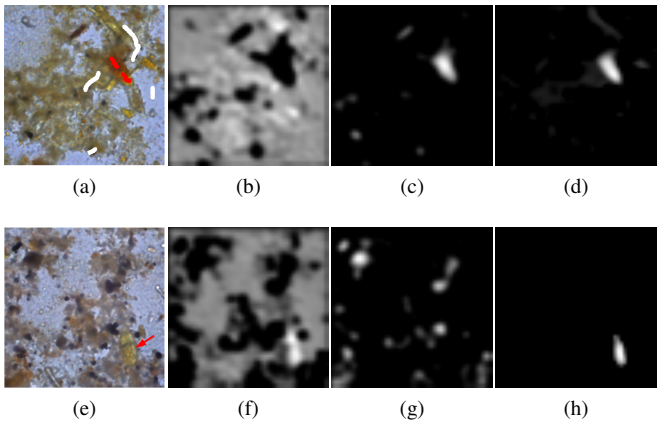


Fig. 1. Example with parasite eggs: (a) A training image with foreground (red) and background (white) markers; (b) Background activation from a background kernel; (c) Foreground activation from a foreground kernel; (d) Saliency map; (e) A test image with a red arrow indicating object location; (f) Foreground activation from a background kernel; (g) Background activation from a foreground kernel; and (h) Saliency map.

I. Introduction

Salient Object Detection (SOD) methods locate the most evident objects in an image [1], [2], with most recent works relying on deep learning models [3]–[6]. Despite their success, large labeled datasets and intensive computational resources are usually needed, leading to the investigation of lightweight models [4], [7], [8]. SAMNet [4] was presented as a lightweight model and introduced a novel operation called Stereoscopically Attentive Multiscale (SAM) that fuses the features at different scales. The network has 1.33 million parameters, considerably fewer than other deep models, achieving comparable results. MSCNet [7] is another lightweight example that explores multiple scales to refine salient objects. It is heavier than SAMNet, with 3.26 million parameters, yet much lighter than other deep models. Similarly light, MEANet [8] has 3.27 million parameters focusing on little memory consumption. However, a common characteristic of those pre-trained lightweight models is that their adaptation to new applications is challenging when labeled data are scarce. In this scenario, *Feature Learning from Image Markers* (FLIM) is a promising research direction [9] since it provides new ways to build convolutional encoders from scratch without backpropagation, requiring minimum human effort in data annotation.

In FLIM, the user selects a few representative images (e.g., five) and draws markers (scribbles or circles) on discriminative regions of them. All convolutional layers' filters (kernels) are estimated by clustering from patches centered at marker pixels. The kernels are derived from the clusters' centers with the exact shape of the patches. FLIM-based lightweight networks have shown promising results in object delineation [9], image classification [10], instance segmentation [11], and object detection [12], surpassing the same architecture trained from scratch and deep models in some applications [9], [10], [12].

A FLIM encoder extracts feature maps for SOD, and subsequently, those features are combined to produce a final object saliency map. Most works are concerned with the first part of this pipeline, producing FLIM encoders but still relying on backpropagation to train a decoder [11] or a classifier [10]. For SOD, recent work [12] has introduced the concept of an *adaptive decoder* – a point-wise convolution with weights estimated on the fly for each input image followed by ReLU activation. In this case, the entire network does not require backpropagation. Such FLIM-based SOD networks can be thousands of times smaller than deep models, being efficiently executed on CPU.

By drawing background (white) and foreground (red) mark-

ers (Figure 1a), a FLIM encoder can estimate kernels that produce feature maps with background (Figure 1b) and foreground (Figure 1c) activation channels. An adaptive decoder should assign positive weights to foreground channels and negative weights to background ones. Its point-wise convolution (a weighted average of the channels) followed by ReLU can eliminate false positive activations in foreground channels by subtracting background activations when generating an object saliency map (Figure 1d). However, in test images (Figure 1e), some background kernels may create foreground activations (Figure 1f) and vice versa (Figure 1g). Hence, an adaptive decoder cannot rely only on labeled markers. It should be able to differentiate between foreground and background activation channels for each input image when generating its object saliency map (Figure 1h).

In this work, we describe four new *adaptive decoders*: (i) a slight modification of the adaptive decoder in [13], named here the tri-state decoder, (ii) labeled markers, (iii) channel attention, and (iv) a combination of (i) and (ii). The tri-state decoder in (i) identifies three types of activation channels, assigning weight -1 for background, $+1$ for foreground, and 0 for undefined channels. In (ii), it is assumed that foreground (background) channels in a feature map are created by kernels derived from markers with the foreground (background) label. Channel weights are $+1$ for the foreground and -1 for the background channels. From Figure 1, we know this assumption may not always hold. However, it is worth evaluating. In (iii), channel attention is estimated to be higher for foreground than for background channels. Positive and negative weights are assigned to the feature map's channels by subtracting channel attention values from their mean value. In (iv), we use the channel weight from (i) if the weight from (ii) indicates a foreground channel; otherwise, set it to 0.

Section II discusses FLIM and the adaptive decoder. Section III presents the four adaptive decoders. Section IV provides qualitative and quantitative analyses. Finally, Section V presents final considerations and future works.

II. Theorectical Background

Figure 2 gives an overview of the overall FLIM pipeline, in which features are extracted by the FLIM-based encoder, and the feature maps are combined by a decoder to generate a saliency map. This section presents basic definitions while explaining a FLIM-based SOD network.

A. Images and patches

Let $\mathbf{I}$ be an image with m channels and domain $D_I \subset Z^2$ of size $w \times h$ pixels, such that $\vec{I}(p) = (I_1(p), I_2(p), \ldots, I_m(p)) \in \mathbb{R}^m$ assigns m features to every pixel $p = (x_p, y_p) \in D_I$. Let $A(p)$ be a set of $k \times k$ adjacent pixels $q = (x_q, y_q) \in D_I$, $x_q - x_p \in [-d\frac{k}{2}, d\frac{k}{2}]$ and $y_q - y_p \in [-d\frac{k}{2}, d\frac{k}{2}]$, within a squared region of size $dk \times dk$ centered at p with dilation factor $d \geq 1$. A patch $\vec{P}(p) \in \mathbb{R}^{k \times k \times m}$ results from the concatenation of feature vectors $\vec{I}(q)$ of all $q \in A(p)$.

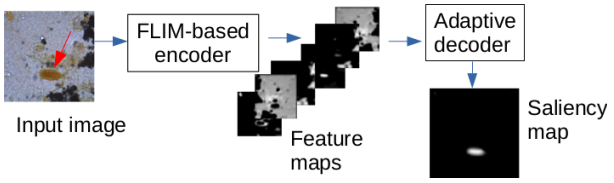


Fig. 2. FLIM-based network with an adaptive decoder for parasite egg (arrow) detection in optical microscopy images.

B. FLIM encoder

For the sake of simplicity, a FLIM encoder may be a sequence of convolutional layers, with each convolutional layer containing a sequence of four operations: marker-based normalization, convolution, ReLU activation, and max-pooling. These operations are well known, but marker-based normalization, convolution, and ReLU provide an important geometrical interpretation for understanding kernel estimation.

Let $\mathbf{I}^{l-1}$ be the input of a convolutional layer l, $l = 1, 2, \ldots, L$, for an encoder with L layers. Let $\mathcal{M}_i \subset D_I$ be a set of pixels from a given image marker and $\mathcal{M} = \cup_{i=1}^{N} \mathcal{M}_i$ be the union of pixels from N markers drawn by the user in multiple selected images. Patches $\vec{P}(p)$ are extracted from all marker pixels $p \in \mathcal{M}$ forming a patch dataset $\mathcal{P} = \cup_{i=1}^{N} \mathcal{P}_i$, where $\mathcal{P}_i$ is the subset of patches from marker pixels $p \in \mathcal{M}_i$.

We apply $z-$score normalization to the patches in $\mathcal{M}$ and use its parameters to normalize patches from any image $\mathbf{I}^{l-1}$. This operation centralizes the patches around the origin of $\mathbb{R}^{k \times k \times m}$ and corrects distortions among different features, dismissing the need for estimating bias. A c-means clustering is applied to each normalized patch dataset $\mathcal{P}_i$ and one kernel $\vec{K} \in \mathbb{R}^{k \times k \times m}$, with norm $\|\vec{K}\| = 1$, is derived from the center of each cluster. If the kernels total from $\mathcal{P}$ is higher than the desired number of kernels for layer l, the number of kernels can be reduced by principal component analysis (i.e., the kernels are eigenvectors) or by a clustering method (i.e., the kernels are cluster representatives) . Otherwise, each marker contributes for layer l with the same number c of kernels per marker.

The convolution between image $\mathbf{I}^{l-1}$ and a kernel $\vec{K}_j$, $j \in [1, m']$, of m' desired kernels for layer l results a channel J_j in the output image $\mathbf{J}^l$ with domain $D_J = D_I$ and $\vec{J}(p) = (J_1(p), J_2(p), \ldots, J_{m'}(p)) \in \mathbb{R}^{m'}$, such that

$$J_j(p) \quad = \quad \langle \vec{P}(p), \vec{K}_j \rangle, \tag{1}$$

$j = 1, 2, \ldots, m'$ and to every pixel $p \in D_I$. Hence, convolution computes a distance (inner product) between $\vec{P}(p)$ and a hyperplane at the origin of $\mathbb{R}^{k \times k \times m}$, whose unit normal vector is $\vec{K}_j$. Patches that fall on the positive side of the hyperplane cause activated values $J_j(p) > 0$ after ReLU, while those on the negative side generate $J_j(p) = 0$. Activation $J_j(p)$ indicates a "similarity" between the patch $\vec{P}(p)$ and the visual pattern of $\vec{K}_j$. By drawing markers on discriminative regions, each kernel represents a visual pattern inside or outside the object. After convolution and ReLU, image regions with similar patches to $\vec{K}_j$ should be activated with zeroes

elsewhere in channel J_i. After max-pooling, the output image $\mathbf{I}^l$ should contain m' channels with object or background activations.

The user provides an encoder architecture with patch sizes, number of kernels per marker, kernel-dilation factors, number of desired kernels, pooling adjacency, strides, and pooling type for each layer l. The above procedure is repeated for each layer $l = 1, 2, \ldots, L$ to estimate kernels from the output $\mathbf{I}^{l-1}$ of the previous layer using set $\mathcal{M}$ of marker pixels mapped to its domain. We are not using marker labels for kernel estimation, but this is an option [10]. Deeper the layer, the clusters are expected to be farther from the origin of $\mathbb{R}^{k\times k\times m}$, enhancing the filtering effect with fewer activations from nearby clusters. After that, the encoder is ready to extract a feature map $\mathbf{I}^L$ for any input image $\mathbf{I}^0$ from the validation/test set.

C. Adaptive decoder

A simple decoder is a point-wise convolution between a feature map $\mathbf{I}^L$ 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. The operation creates a saliency map $\mathbf{S}$ with image domain D_S and values $S(p) \in \mathbb{R}$ for each $p \in D_S$, such that

$$S(p) \quad = \quad \Phi\left(\langle \vec{I}^L(p), \vec{\alpha}\rangle\right) = \Phi\left(\sum_{i=1}^{m'} \alpha_i I_i^L(p)\right). \quad (2)$$

One may upsample D_S or D_{I^L} to the size of D_{I^0}, whenever strides are used. In this work, $D_S = D_{I^L} = D_{I^0}$.

According to the example illustrated in Figure 1, fixing the weights α_i by optimization is not an alternative since the same kernel may create background or foreground channels I_i^L depending on the input image. One may still want to discard a channel with an undefined label. Next, we present the proposed four adaptive decoders.

III. Adaptive decoders for SOD networks

This section presents four adaptive decoders for FLIM-based SOD networks. They are all implemented by Equation 2, with their difference relying on the definition of $\vec{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_{m'})$. Figure. 3 illustrates the results of the four adaptive decoders.

A. Tri-state adaptive decoder

Let $\mu_{I_i^L}$ be the mean activation of channel I_i^L, τ be the Otsu threshold of the distribution $\{\mu_{I_1^L}, \mu_{I_2^L}, \ldots, \mu_{I_{m'}^L}\}$, and σ be the standard deviation of that distribution. The number of pixels above the Otsu threshold of channel I_i^L divided by D_I defines another threshold t_i. The **tri-state adaptive decoder** is a slight modification of the one presented in [13], which defines three channel classes: background ($\alpha_i = -1$), foreground ($\alpha_i = +1$), and undefined ($\alpha_i = 0$). Unlikely of [13], we incorporate a threshold t_i for removing undesired regions. The values α_i are then defined as

$$\alpha_i = \begin{cases} -1, & \text{if} \quad \mu_{I_i^L} \geq \tau + \sigma \text{ and } t_i > 0.2, \\ +1, & \text{if} \quad \mu_{I_i^L} \leq \tau - \sigma \text{ and } t_i < 0.1, \\ 0, & \text{otherwise.} \end{cases} \quad (3)$$

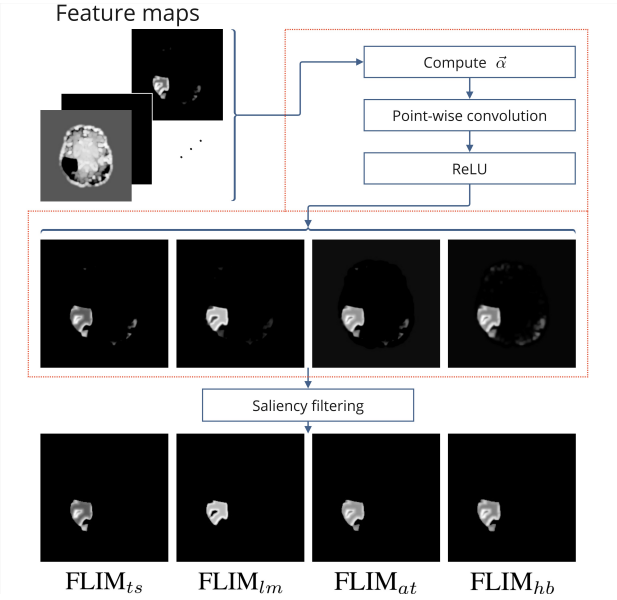


Fig. 3. Adaptive decoder overview(in red). Given the feature maps of each input image, the adaptive decoder computes a kernel $\vec{\alpha} = (\alpha_1, \alpha_2, \cdots, \alpha_{m'}) \in \mathbb{R}^{1\times 1\times m'}$ used in a point-wise convolution followed by a ReLU. Pos-processings are done after the decoding process.

B. Labeled-marker-based adaptive decoder

A **labeled-marker-based adaptive decoder** assumes that a kernel from a labeled marker will create a channel with the same label. Hence, we can assume that $\lambda(I_i^L) \in \{-1, +1\}$ indicates background (-1) or foreground ($+1$) channel and define the weight vector of a labeled-marker-based decoder as $\vec{\alpha}^{lm} = (\alpha_1^{lm}, \alpha_2^{lm}, \ldots, \alpha_{m'}^{lm})$, where $\alpha_i^{lm} = \lambda(I_i^L)$, $i \in [1, m']$.

C. Attention-based adaptive decoder

In [14], the authors propose a way to compute channel attention without trainable parameters. The method highlights the image regions that mostly contribute to the network's final output. In this work, we use a slight modification of this method. Consider X_i and Y_i the results of max-pooling and average pooling of each channel I_i^L, $i \in [1, m']$, respectively. The values in X_i and Y_i are linearly normalized within $[0, 1]$. A spatial attention channel a_i is defined by the linear normalization of $X_i + Y_i$ within $[0, 1]$. Let $\vec{a}_i \in \mathbb{R}^{w\times h}$ be the vectorization of channel a_i and $\vec{b}_i \in \mathbb{R}^{w\times h}$ be the vectorization of channel I_i^L. Each channel's importance c_i is computed by

$$c_i \quad = \quad \frac{\langle \vec{a}_i, \vec{b}_i\rangle}{\|\vec{a}_i\|\|\vec{b}_i\|}. \quad (4)$$

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 of an attention-based decoder can be defined as $\vec{\alpha}^{at} = (\alpha_1^{at}, \alpha_2^{at}, \ldots, \alpha_{m'}^{at})$, where

$$\alpha_i^{at} \quad = \quad \begin{cases} 1, & \text{if } c_i < \mu_c - \frac{\sigma_c}{2}, \\ -1, & \text{if } c_i > \mu_c + \frac{\sigma_c}{2} \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

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

D. Hybrid adaptive decoder

This decoder combines the tri-state adaptive decoder with the labeled-marker-based one. Let $\vec{\alpha}^{ts} = (\alpha_1^{ts}, \alpha_2^{ts}, \ldots, \alpha_{m'}^{ts})$ be the weight vector created by the tri-state decoder for some input image. We define

$$\alpha_i^{hb} = \begin{cases} 0, & \text{if } \alpha_i^{lm} = -1, \\ \alpha_i^{ts}, & \text{otherwise,} \end{cases} \tag{6}$$

for $i \in [1, m']$, as the weights in the vector $\vec{\alpha}^{hb} = (\alpha_1^{hb}, \alpha_2^{hb}, \ldots, \alpha_{m'}^{hb})$ of the hybrid adaptive decoder. Thus, we consider only the weights of the tri-state decoder if a kernel from a background-labeled marker did not create the corresponding channel. This decoder was proposed to correct the cases where kernels generated by background-labeled markers activate for the objects.

IV. Experiments, Results, and Discussion

This section presents the experiments for salient object detection. For a fair comparison, we have used the same FLIM encoder (architecture and parameters) for the proposed adaptive decoders and the adaptive decoder without the threshold. Moreover, we also compare the proposed decoder to the state-of-the-art of lightweight SOD models.

A. Datasets

Two datasets were used: Parasites, a private dataset with eggs of *Schistosoma Mansoni* in optical microscopy images, and Tumors, a public dataset with brain tumors from the Brain Tumor Segmentation Challenge (BraTS) 2021 [15]–[17]. The parasites dataset contains 1219 images of 400×400 pixels and three channels, from which 853 were divided into three splits of 5 images for training and 848 for validation. The remaining 366 were held for testing. The tumor dataset contains 3743 grayscale images of 240×240 pixels, and 1113 were held for testing. The remaining 2630 images were divided into three splits of 5 for training and 2625 for validation. The parasites dataset was converted from RGB to the normalized CIELAB color space for the network's training. The grayscale images from the BraTS dataset were also converted to CIELAB by applying a color table from blue to red and subsequently converting them to CIELAB.

B. FLIM encoder architectures

Figure 4 presents the FLIM encoder architectures for Parasites (above) and Tumors (below). These architectures were empirically defined by observing the results in the validation set, which leads to differences in the number of kernels per layer (increasing in one and decreasing in another). We use stride ($s = 1$), and the dilation factor d was used only in the parasite encoder's last convolutional layer ($d = 7$). These encoders create feature maps with eight and sixty-four activation channels for Parasites and Tumors.

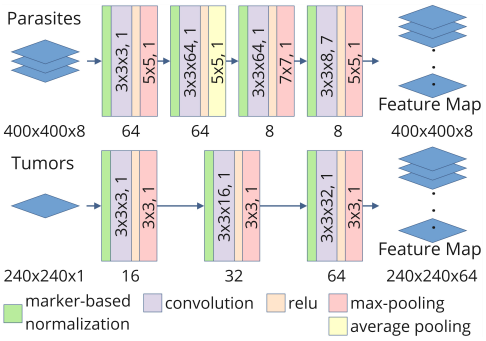


Fig. 4. FLIM encoder architectures for parasite egg (above) and brain tumor (below) detection. For convolution, the numbers indicate $k \times k \times m, d$, and the number of filters per layer is indicated below. For pooling the numbers indicate $k \times k, s$, where s is the stride in both directions.

C. Compared methods

The FLIM networks for comparison used the same encoder with each adaptive decoder: original adaptive decoder FLIM$_{oad}$ [13], tri-state (FLIM$_{ts}$), labeled-marker-based (FLIM$_{lm}$), attention-based (FLIM$_{at}$), and hybrid (FLIM$_{hb}$). For reference, we include a comparison with three pre-trained lightweight models, fine-tunned with the training set: SAM-Net [4], MSCNet [7], and MEANet [8]. We also compare the methods with a U-shaped network (based on [18]), using its encoder initialized by FLIM (U-Net$_{\text{FLIM}}$). The encoder's weights are frozen during the decoder's training. The FLIM encoder for parasites and tumors has three layers with 16, 32, and 64 kernels. For parasites, the kernels have size $5 \times 5 \times m$ and max-pooling uses size 3×3 with stride $s = 2$ in all layers. For tumors, the kernels have size $3 \times 3 \times m$ and max-pooling uses size 3×3 with stride $s = 2$ in all layers. The decoder uses skip connections from all layers and uses interpolation (rather than transpose convolution) for the upsampling of the feature maps, which further passes through convolutions with kernels of size $3 \times 3 \times m$ followed by ReLU as in the original U-Net. This method was trained for 100 epochs, using as loss function the average between Dice loss and Binary cross-entropy; the learning rate was decreased linearly from 1e-2 to 1e-5 with the Adam optimizer.

D. Evaluation metrics

We used three popular metrics to evaluate the models: Weighted F measure [19], Mean Absolute Error (MAE), Enhanced-aligment Measure (E-M) [20] and also Dice.

E. Experimental setup

The FLIM encoders were trained as described in Section II using only the five training images. With three splits, the model with the best validation result was used for testing. Similarly, each pre-trained lightweight network was fine-tuned on the three training sets, and the best model was used for testing.

For parasites, we applied grid search to determine the minimum and maximum sizes of the objects. This search used the validation set and possible bounds defined as lower bound $= \{1000, 2000, 3000\}$ and upper bound $=$

$\{10000, 15000, 20000\}$ pixels. Grid search considered the average of the above metrics, except MAE, in the validation sets to find the best parameters. Post-processing eliminated components out of the best size range from the results of all FLIM and lightweight models.

F. Quantitative results and discussion

Table I shows the mean results of all models in the validation sets for both datasets. Note that the FLIM models have from 56 to 251 times fewer parameters than the lightweight models and require considerably fewer GFLOPs than MSCNet and MEANet. Only SAMNet is the most efficient model. SAMNet obtained the best E-M for Parasites, while MEANet was the best model according to F_β^ω and DICE, and U-Net$_{FLIM}$ was the one with the best MAE. The parasite dataset exemplifies a case in which the adaptation of a pre-trained model works. Since the FLIM models are trained from scratch with only five images, these results also indicate that there are more suitable strategies than selecting a few training images randomly. We agree with the authors in [12] that more user involvement in selecting images, markers, and network architecture can improve the effectiveness of the FLIM models. None of the pre-trained models could adapt to Tumors, even having converted them to color images (the results with the gray-scale images were worse). U-Net$_{\text{FLIM}}$ was the only model trained with backpropagation (at least its decoder) which achieves satisfactory results on this dataset. In this case, FLIM$_{hb}$ presented the best result for all metrics except MAE, which was better with FLIM$_{at}$. Note that for tumors, some models present higher MAE values. This behavior stems from some of the splits, the model tends to generate saliencies of the whole brain, as in Figure 5 (d), which shows that the model could not learn with the data.

Table II shows the results in the test set. Such results agree with the average ones in the validation sets, demonstrating that none of the models overfitted or underfitted. MEANet was still the best model for Parasites for two metrics, except for MAE and E-M, which were better with U-Net$_{\text{FLIM}}$ and SAMNet. The FLIM models presented the best results for the tumor dataset, being FLIM$_{hb}$ the best one for the same three metrics, except MAE, which was better with FLIM$_{at}$.

Among the adaptive decoders, the tri-state one performed best in the parasite dataset, except for E-M, which was better with the hybrid decoder. Ou modification on this decoder also proves to be beneficial, improving the results in both datasets. The hybrid decoder was the best among the proposed ones in the tumor dataset. Its F_β^ω was from 1.05 to 5.34 times higher, MAE was from 0.04 to 0.31 times lower, E-M was from 1.01 to 1.87 times higher, and DICE was from 1.17 to 5.05 times higher than the values obtained by the three lightweight models. For the parasite dataset, improving the FLIM network may also improve results with the hybrid decoder.

G. Qualitative results and discussion

Figure 5 presents the saliency maps generated by all models in both datasets. The first row presents the qualitative results for the parasite dataset, while the second row presents the qualitative results for the tumor dataset. It can be seen that the lightweight models (Figures 5(c)- 5(e)) generated better results for Parasites, even in cases where impurities surround the egg. The FLIM models (Figures 5(g)- 5(j)), on the other hand, present better results for Tumors. The U-Net$_{\text{FLIM}}$ presents a good parasite saliency(Figure 5(f)), but that is not as good as the lightweight models, which explains the lower metrics. For Tumors(Figure 5(f)), the model presents saliencies better than the lightweights but has saliencies bigger than the actual size of the tumor, which explains the high MAE of this model.

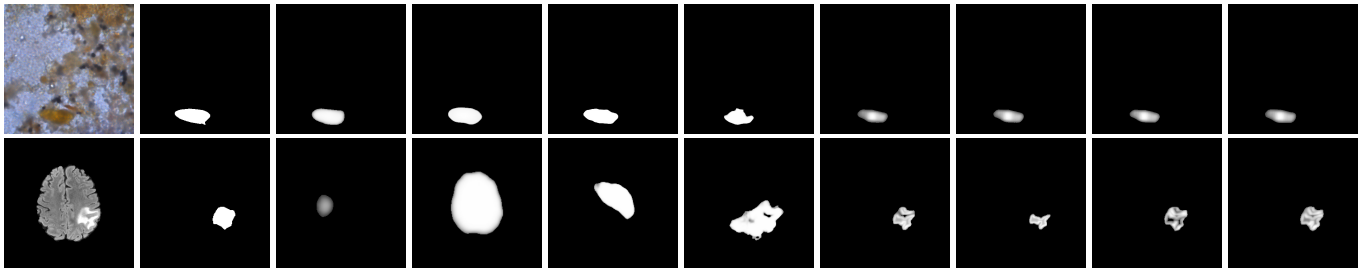
The lightweight models could have generated better saliency maps in the tumor dataset. MSCNet presents the worst results among these models, generating saliency maps with the whole brain. FLIM decoders, in contrast, generate better saliency maps, managing to detect the tumor with the right size and position. The hybrid decoder provides the best qualitative results, creating a better saliency map that fits the tumor area. The tri-state decoder provides a crisp border in the saliency map but misses parts of the object. Indeed, the lightweight models adopt a loss that improve object delineation, while the

TABLE I
MEAN RESULTS IN THE VALIDATION SETS. ARROWS ↑ AND ↓ DENOTE HIGHER AND LOWER VALUES ARE BETTER, RESPECTIVELY.

Parasites	#Params ↓	GFLOPs ↓	F_β^ω ↑	MAE ↓	E-M ↑	DICE ↑
SAMNet [4]	1.33(M)	**0.5**	0.520±0.086	5.105±1.484	**0.711±0.058**	0.524±0.087
MSCNet [7]	3.26(M)	9.62	0.712±0.075	2.050±0.548	0.584±0.048	0.737±0.069
MEANet [8]	3.27(M)	5.87	**0.774±0.010**	1.688±0.257	0.589±0.013	**0.788±0.010**
U-Net$_{\text{FLIM}}$	64.91(K)	3.27	0.731±0.037	**1.128±0.423**	0.589±0.057	0.743±0.037
FLIM$_{oad}$	**13.04(K)**	0.65	0.544±0.079	4.701±1.900	0.664±0.029	0.601±0.081
FLIM$_{at}$	**13.04(K)**	0.65	0.549±0.074	3.984±1.127	0.609±0.005	0.593±0.076
FLIM$_{lm}$	**13.04(K)**	0.65	0.638±0.054	2.554±1.428	0.593±0.070	0.687±0.046
FLIM$_{ts}$	**13.04(K)**	0.65	0.589±0.073	2.446±0.589	0.642±0.038	0.634±0.075
FLIM$_{hb}$	**13.04(K)**	0.65	0.596±0.061	2.578±0.570	0.637±0.047	0.645±0.063
Tumors	#Param ↓	GFLOPs↓	F_β^ω ↑	MAE ↓	E-M ↑	DICE ↑
SAMNet	1.33(M)	**0.5**	0.155±0.011	9.569±1.580	0.762±0.020	0.195±0.008
MSCNet	3.26(M)	9.62	0.146±0.001	39.342±1.122	0.452±0.007	0.273±0.001
MEANet	3.27(M)	5.87	0.119±0.011	6.602±2.150	0.739±0.034	0.129±0.014
U-Net$_{\text{FLIM}}$	64.91(K)	0.89	0.560±0.024	2.777±1.704	0.787±0.046	0.572±0.025
FLIM$_{oad}$	**23.58(K)**	1.18	0.407±0.070	13.999±3.652	0.687±0.059	0.462±0.053
FLIM$_{at}$	**23.58(K)**	1.18	0.423±0.062	**2.381**±1.849	0.746±0.031	0.476±0.040
FLIM$_{lm}$	**23.58(K)**	1.18	0.334±0.216	18.695±25.497	0.627±0.181	0.384±0.219
FLIM$_{ts}$	**23.58(K)**	1.18	0.483±0.064	2.867±1.776	0.742±0.042	0.523±0.075
FLIM$_{hb}$	**23.58(K)**	1.18	**0.562±0.052**	2.725±1.642	**0.819±0.012**	**0.634±0.048**

TABLE II
RESULTS IN THE TEST SET. ARROWS ↑ AND ↓ DENOTE HIGHER AND LOWER VALUES ARE BETTER, RESPECTIVELY.

Parasites	#Params ↓	GFLOPs ↓	F_β^ω ↑	MAE ↓	E-M ↑	DICE ↑
SAMNet [4]	1.33(M)	**0.5**	0.602	5.303	**0.737**	0.614
MSCNet [7]	3.26(M)	9.62	0.804	1.242	0.485	0.825
MEANet [8]	3.27(M)	5.87	**0.824**	1.430	0.553	**0.834**
U-Net$_{\text{FLIM}}$	64.91(K)	3.27	0.799	**0.637**	0.484	0.809
FLIM$_{oad}$	**13.04(K)**	1.18	0.681	2.048	0.575	0.731
FLIM$_{at}$	**13.04(K)**	0.65	0.664	2.198	0.584	0.707
FLIM$_{lm}$	**13.04(K)**	0.65	0.704	1.653	0.536	0.745
FLIM$_{ts}$	**13.04(K)**	0.65	0.711	1.331	0.536	0.749
FLIM$_{hb}$	**13.04(K)**	0.65	0.677	1.914	0.588	0.725
Tumors	#Param ↓	GFLOPs ↓	F_β^ω ↑	MAE ↓	E-M ↑	DICE ↑
SAMNet	1.33(M)	**0.5**	0.161	8.180	0.787	0.199
MSCNet	3.26(M)	9.62	0.150	36.619	0.454	0.279
MEANet	3.27(M)	5.87	0.116	5.982	0.727	0.137
U-Net$_{\text{FLIM}}$	64.91(K)	0.89	0.575	5.161	0.843	0.589
FLIM$_{oad}$	**23.58(K)**	1.18	0.475	9.743	0.741	0.512
FLIM$_{at}$	**23.58(K)**	1.18	0.550	**0.549**	0.772	0.595
FLIM$_{lm}$	**23.58(K)**	1.18	0.528	1.207	0.829	0.603
FLIM$_{ts}$	**23.58(K)**	1.18	0.558	2.005	0.805	0.621
FLIM$_{hb}$	**23.58(K)**	1.18	**0.620**	1.623	**0.852**	**0.692**



(a) Original image (b) Ground truth (c) SAMNet (d) MSCNet (e) MEANet (f) U-Net$_{\text{FLIM}}$ (g) FLIM$_{\text{at}}$ (h) FLIM$_{\text{lm}}$ (i) FLIM$_{\text{ts}}$ (j) FLIM$_{\text{hb}}$

Fig. 5. Qualitative results that show the FLIM networks can outperform the others. (a) Images from Parasites and BraTS. (b) Ground truth. Saliency map by (c) SAMNet, (d) MSCNet, (e) MEANet, and FLIM with the (f) U-Net$_{\text{FLIM}}$, (g) attention-based, (h) labeled-marker-based, (i) tri-state, and (j) hybrid adaptive decoders.

FLIM networks are not exploring object delineation methods. Their qualitative and quantitative results can improve in both datasets by using the saliency maps to estimate seeds for graph-based object delineation.

V. Conclusion

We present and evaluate adaptive decoders for FLIM-based networks by comparing them with lightweight SOD methods. The best adaptive decoder combines the tri-state with the labeled-marker-based decoder. Its competitive results confirm that adaptive decoding each image is a promising strategy for SOD in FLIM networks. The results of the U-shaped network also indicate that the adaptive decoders should explore feature maps from all layers in a future work.

The results of the FLIM networks depend heavily on selecting good training images. This work used a 3-fold split created by hand and methods for more representative image selection should be evaluated in the future. The encoder's architecture is another point for improvement. It was built empirically since our focus was to investigate the decoding process only.

Acknowledgment

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

References

[1] L. Wang, H. Lu, X. Ruan, and M.-H. Yang, "Deep networks for saliency detection via local estimation and global search," in *Proc. of the IEEE Conf. on Comput. Vis. and Pattern Recognit.*, 2015, pp. 3183–3192.

[2] R. Zhao, W. Ouyang, H. Li, and X. Wang, "Saliency detection by multi-context deep learning," in *Proc. of the IEEE Conf. on Comput. Vis. and Pattern Recognit.*, 2015, pp. 1265–1274.

[3] X. Qin, Z. Zhang, C. Huang, C. Gao, M. Dehghan, and M. Jagersand, "Basnet: Boundary-aware salient object detection," in *Proc. of the IEEE/CVF Conf. on Comput. Vis. and Pattern Recognit.*, 2019, pp. 7479–7489.

[4] Y. Liu, X.-Y. Zhang, J.-W. Bian, L. Zhang, and M.-M. Cheng, "Samnet: Stereoscopically attentive multi-scale network for lightweight salient object detection," *IEEE Trans. on Image Process.*, vol. 30, pp. 3804–3814, 2021.

[5] M. S. Lee, W. Shin, and S. W. Han, "Tracer: Extreme attention guided salient object tracing network (student abstract)," in *Proc. of the AAAI Conf. on Artif. Intell.*, vol. 36, no. 11, 2022, pp. 12 993–12 994.

[6] X. Qin, Z. Zhang, C. Huang, M. Dehghan, O. R. Zaiane, and M. Jagersand, "U2-net: Going deeper with nested u-structure for salient object detection," *Pattern recognition*, vol. 106, p. 107404, 2020.

[7] Y. Lin, H. Sun, N. Liu, Y. Bian, J. Cen, and H. Zhou, "A lightweight multi-scale context network for salient object detection in optical remote sensing images," in *2022 26th Int. Conf. on Pattern Recognit. (ICPR)*. IEEE, 2022, pp. 238–244.

[8] B. Liang and H. Luo, "Meanet: An effective and lightweight solution for salient object detection in optical remote sensing images," *Expert Systems with Applications*, p. 121778, 2023.

[9] I. E. de Souza, B. C. Benato, and A. X. Falcão, "Feature learning from image markers for object delineation," in *2020 33rd SIBGRAPI Conf. on Graph., Patterns and Images (SIBGRAPI)*. IEEE, 2020, pp. 116–123.

[10] I. E. De Souza and A. X. Falcão, "Learning cnn filters from user-drawn image markers for coconut-tree image classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 19, pp. 1–5, 2020.

[11] M. A. Cerqueira, F. Sprenger, B. C. Teixeira, and A. X. Falcão, "Building brain tumor segmentation networks with user-assisted filter estimation and selection," in *18th Int. Symp. on Med. Inf. Process. and Anal.*, vol. 12567. SPIE, 2023, pp. 202–211.

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

[13] L. Joao., M. Cerqueira., B. Benato., and A. Falcão., "Understanding marker-based normalization for flim networks," in *Proc. of the 19th Int. Joint Conf. on Comput. Vis., Imaging and Comput. Graph. Theory and Appl. - Volume 2: VISAPP*, INSTICC. SciTePress, 2024, pp. 612–623.

[14] W. Li, Y. Zhang, W. Shi, and S. Coleman, "A cam-guided parameter-free attention network for person re-identification," *IEEE Signal Processing Letters*, vol. 29, pp. 1559–1563, 2022.

[15] B. H. Menze, A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahani, J. Kirby, Y. Burren, N. Porz, J. Slotboom, R. Wiest *et al.*, "The multimodal brain tumor image segmentation benchmark (brats)," *IEEE Trans. on Med. imaging*, vol. 34, no. 10, pp. 1993–2024, 2014.

[16] S. Bakas, H. Akbari, A. Sotiras, M. Bilello, M. Rozycki, J. S. Kirby, J. B. Freymann, K. Farahani, and C. Davatzikos, "Advancing the cancer genome atlas glioma mri collections with expert segmentation labels and radiomic features," *Scientific data*, vol. 4, no. 1, pp. 1–13, 2017.

[17] S. Bakas, M. Reyes, A. Jakab, S. Bauer, M. Rempfler, A. Crimi, R. T. Shinohara, C. Berger, S. M. Ha, M. Rozycki *et al.*, "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*, 2018.

[18] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Med. Image Comput. and Computer-Assisted Interv.–MICCAI 2015: 18th Int. Conf., Munich, Germany, October 5-9, 2015, proc., part III 18*. Springer, 2015, pp. 234–241.

[19] R. Margolin, L. Zelnik-Manor, and A. Tal, "How to evaluate foreground maps?" in *Proc. of the IEEE Conf. on Comput. Vis. and Pattern Recognit.*, 2014, pp. 248–255.

[20] D.-P. Fan, C. Gong, Y. Cao, B. Ren, M.-M. Cheng, and A. Borji, "Enhanced-alignment measure for binary foreground map evaluation," in *Proc. of the Twenty-Seventh Int. Joint Conf. on Artif. Intell., IJCAI-18*. Int. Joint Conf. on Artif. Intelli. Organ., 7 2018, pp. 698–704.