Journal of the Brazilian Computer Society, 2026, 32:1, doi: 10.5753/jbcs.2026.5892
This work is licensed under a Creative Commons Attribution 4.0 International License.

Multi-level Cellular Automata for FLIM networks

Felipe Crispim R. Salvagnini [**University of Campinas** | ***felipe.salvagnini@students.ic.unicamp.br***]
Jancarlo F. Gomes [**University of Campinas** | ***jgomes@ic.unicamp.br***]
Cid A. N. Santos [**Eldorado Institute** | ***cid.santos@eldorado.org.br***]
Silvio Jamil F. Guimarães [**Pontifical Catholic University of Minas Gerais** | ***sjamil@pucminas.br***]
Alexandre X. Falcão [**University of Campinas** | ***afalcao@unicamp.br***]

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

Received: 12 April 2025 • **Accepted:** 31 August 2026 • **Published:** 23 November 2026

Abstract. The need for abundant annotated data and complex network architectures poses a significant challenge for deep-learning Salient Object Detection (deep SOD) and across the broader deep-learning landscape. This challenge is particularly acute in medical applications in developing countries with limited computational resources. Combining modern and classical techniques offers a path to maintaining competitive performance while enabling practical applications. The Feature Learning from Image Markers (FLIM) methodology empowers experts to design convolutional encoders using user-drawn markers, with filters learned directly from these annotations. Recent findings demonstrate that coupling a FLIM encoder with an adaptive decoder creates a flyweight network suitable for SOD, requiring significantly fewer parameters than lightweight models and eliminating the need for backpropagation. Cellular Automata (CA) methods have proven successful in data-scarce scenarios but require proper state initialization — typically through user input, priors, or randomness. We propose a practical intersection of these approaches: using FLIM networks to initialize CA states with expert knowledge, without requiring user interaction per image. By decoding features at each level of a FLIM network, we can simultaneously initialize multiple CAs, creating a multi-level framework. Our method leverages hierarchical knowledge across network layers, merging multiple saliency maps into a high-quality final output that serves as a CA ensemble. Benchmarks across two challenging medical datasets demonstrate the competitiveness of our multi-level CA approach compared to established models in the deep SOD literature.

Keywords: SOD, Cellular Automata, Multi-Level Cellular Automata, Cellular Automata Ensemble, Saliency Merging, Feature Learning from Image Markers, FLIM, Adaptive Decoders.

1 Introduction

When a specialist analyzes an image, such as a neuroradiologist investigating tumors in patients' Magnetic Resonance Imaging (MRI) data, abnormalities usually stand out and are easily spotted. Likewise, a parasitologist can quickly locate parasite eggs in microscopic images. In some images, even someone without training can identify the abnormalities. Additionally, we naturally perform this task when we observe a scene and the foreground becomes prominent. Salient Object Detection (SOD) methods propose to tackle the same problem, generating a saliency map where the foreground (or abnormality) is brighter than the background.

SOD methods are categorized into bottom-up or top-down approaches [Qin *et al*., 2018]. Bottom-up methods employ low-level features — local properties such as color, texture, and descriptors — and statistics to identify foreground regions [Tong *et al*., 2015]. Conversely, top-down techniques utilize high-level features extracted through supervised methods, which require annotated data [Qin *et al*., 2019, 2020; Zhu *et al*., 2024]. Some works also categorize those approaches into traditional manual feature learning and deep-learning-based methods [Yang *et al*., 2025].

Despite the recent success of deep-learning approaches applied to SOD, as models become even more sophisticated, they face two main challenges in real-world applications: the need for high-cost computational resources (GPUs) and the requirement for large amounts of annotated data (even when pre-trained). The literature proposes solving the former using *Lightweight Models*, in which the complexity of the models is reduced through their design or compression [Chen *et al*., 2024]. Although they speed up inference and reduce model size, usually lightweight models still have 1-5 million parameters and require substantial amounts of annotated data.

The FLIM methodology addresses deep learning challenges by enabling the design of flyweight convolutional networks (typically 2-5 layers) under expert control [Joao *et al*., 2023]. Flyweight is a term that describes networks even lighter than lightweight models, with around 500k parameters. Inserting a user in the interactive pipeline enables: (1) Selection of representative images (usually less than 10) [Cerqueira *et al*., 2024b]; (2) Design of a convolutional encoder from weak annotations inserted by a user (*e.g.,* scribbles or disks) [Salvagnini *et al*., 2024; Soares *et al*., 2024]; (3) Get the saliency map by adaptively decoding features extracted by the encoder [Soares *et al*., 2024]. FLIM is a recently proposed method, and each of those three steps is under investigation.

FLIM networks are suitable for data-scarcity scenarios where the entire network is backpropagation-free. Hence, there is no requirement for densely annotated ground truth;

only weak annotations on a few images are needed. Such an approach is not data-hungry and employs learned high-level features to generate saliency maps. The saliency map generated by FLIM is particularly well-suited for initializing a bottom-up method. The latter behaves as a post-processing operation, leveraging both low-level and high-level features.

Cellular Automata (CA) methods present an excellent candidate for integration with FLIM networks as a post-processing mechanism, where FLIM networks can significantly enhance the initialization phase. CA is commonly applied to data-scarce scenarios (*e.g.,* medical images), where the CA is executed for each input. CA also enables SOD methods using both low-level [Vezhnevets and Konouchine, 2005; Hamamci *et al*., 2012] and high-level features [Qin *et al*., 2018, 2015]. Nevertheless, CA methods require proper initialization. Common strategies are: initialize states from user inputs like a line drawn on image regions [Vezhnevets and Konouchine, 2005; Hamamci *et al*., 2012; Sompong and Wongthanavasu, 2017]; random initialization [Marinescu *et al*., 2018]; priors (as background or contrast priors) [Qin *et al*., 2018]. As far as we are concerned, no work takes advantage of high-level features during initialization; only user-based methods employ the expert's knowledge but require the user for every inference image. Therefore, investigating FLIM for CA initialization is of particular interest.

The intersection of FLIM networks with CA enables a novel initialization technique, characterized by three phases: (1) Design of a FLIM Encoder, which, together with an adaptive decoder, yields a FLIM network; (2) at inference time, initialize CA using a FLIM-Network saliency; (3) evolve the CA to generate a better saliency map. **The key aspect is, unlike other user-interaction-based SOD methods, user interaction is not required at inference time**.

Our previous research investigated the initialization of CA using FLIM networks [Salvagnini *et al*., 2024]. Specifically, with an L-layer FLIM encoder, we decoded feature maps from the last convolutional layer into a saliency map to initialize the CA. As established in [Salvagnini *et al*., 2024] and reiterated here, FLIM encoders detect descriptive and discriminative patterns beginning from the first layer. The initial layers activate strongly in edge regions, providing sharp features but generating numerous false positive activations. In contrast, deeper layers produce fewer or no false positives, though they yield blurrier activations in edge regions. These complementary characteristics suggest the value of decoding saliency maps from each FLIM-Encoder level to initialize multiple CA simultaneously. This multi-level initialization approach enables exploring a broader initialization space, combining the precision of early layers with the reliability of deeper representations, better exploring the edge and internal regions of the salient objects. Moreover, this novel method, called multi-level CA, opens room for investigation on combining the saliency of multiple CAs, which can be seen as an ensemble of CA models.

In improving our previous study, this work investigates the applicability and generalization of our multi-level CA towards two challenging datasets: brain tumors (glioblastomas) and parasite eggs (Schistosoma Mansoni) detection. It evaluates the proposed method towards gray-scale MRI images — with high heterogeneity from multiple institution acquisitions [Baid *et al*., 2021] — and RGB microscopy images [Suzuki *et al*., 2013]. Our main contributions are:

1. We demonstrate significant performance enhancements by initializing multi-level CA with saliencies from each level of the FLIM network. Measurable improvements are observed across each hierarchical layer;
2. We extend our comparison to SOD literature, including lightweight models to address the criticality of deploying salient object detection in low-cost devices;
3. We propose a simple yet elegant fusion mechanism — only three convolutional filters — that effectively integrates multi-level saliency maps into a final saliency. Our experimental results show that our method achieves performance comparable to, and in some cases surpasses, specialized lightweight models and state-of-the-art deep learning SOD architectures (pre-trained on thousands to millions of images);
4. We statistically validate the gains achieved by the CA-FLIM integration using nonparametric significance tests with multiple-comparison correction and effect-size analysis, while also investigating its generalization capabilities through zero-shot experiments;
5. We characterize the computational complexity of the proposed method and demonstrate its scalability through a GPU implementation, achieving a $\sim 27\times$ speed-up, while the CPU baseline remains practical for resource-constrained environments.

This work is organized as follows: Section 2 provides a comprehensive review of SOD literature, covering both classical bottom-up models and modern top-down deep learning approaches. Additionally, it introduces the FLIM methodology and explores CA applications in both SOD and semantic segmentation problems, establishing the foundation necessary for understanding our proposed method. Section 3 presents our multi-level CA method in detail, examining the integration of FLIM networks into hierarchical CA initialization, the CA evolution process, and the ensemble technique that combines multiple CAs to yield enhanced saliency maps. Section 4 describes our experimental setup, guiding the reader through creating a multi-level CA. At the same time, we also discuss the methodology for benchmarking our approach against state-of-the-art methods. Section 5 thoroughly discusses our results across each stage of the proposed method, providing a quantitative statistical analysis of CA-FLIM integration gains using nonparametric significance tests and effect-size analysis, and characterizing the computational complexity of the proposed method and its GPU scalability. Finally, Section 6 concludes with our findings, acknowledges the limitations of our approach, and outlines directions for future research.

2 Related Works

This section reviews the bottom-up and deep-learning SOD methods and builds the intuition behind FLIM methodology and Cellular Automata, setting the foundation for our approach.

2.1 Bottom-up Methods

Bottom-up methods are rooted in cognitive psychology and human perception to predict which areas of an image capture people's attention [Itti *et al*., 1998]. This process typically consists of two stages: first, identifying the region that stands out, and then accurately segmenting that region by fitting a saliency map to the object's interior and edges [Borji *et al*., 2019; Wang *et al*., 2022; Zhu *et al*., 2024].

These classical methods operate directly on low-level features, such as image intensity, color, contrast, and gradients [Itti *et al*., 1998; Wang *et al*., 2024]. Hence, they do not need to learn semantic information, alleviating the requirement for abundant training data. Intrinsic features (derived from the image itself) and/or extrinsic features (*e.g.,* statistics from images or user interaction) are employed, where computational heuristics or classical supervision techniques yield the final saliency map.

Itti *et al*. [1998] drew attention to the topic by taking inspiration from the efficiency of primates' visual mechanism for complex scene understanding. Their seminal work operates over colors, intensity, and orientations at multiple scales (*i.e.,* gaussian pyramids) to generate a salience map. The first SOD models were based on local contrast — the difference between a pixel, or a patch, and its neighbors — to compute salient regions [Ma and Zhang, 2003; Liu and Gleicher, 2006; Liu *et al*., 2007; Achanta, Radhakrishna *et al*., 2008; Rosin, 2009].

For example, Ma and Zhang [2003] employs a perceive field (same size as the input image), where each pixel is a perception unit operating on a neighborhood, measuring the contrast as the color distance in LUV space. This local contrast saliency map is then processed by a region-growing process (fuzzy growing), which splits the pixels into two classes: attended or unattended areas. Liu *et al*. [2007] further expands by applying Conditional Random Field (CRF) learning to combine local, regional, and global features: multi-scale contrast, center-surround histogram, and color spatial distribution. Achanta, Radhakrishna *et al*. [2008] adds and normalizes multi-scale saliency map, where a thresholded image generates the salient object map. Later, Rosin [2009] improves the use of image features by applying edge detection, threshold decomposition, and distance transformation [Borgefors, 1984]. Remarkably, most of the preliminary works on the field employed intrinsic features (input image only) and center-pixel neighborhoods despite being computed at multiple scales.

Other works then explored whole image regions, such as the work of Liu and Gleicher [2006], where a hybrid approach first extracts an intermediary saliency using contrast and later improves by averaging the saliency value of the image's regions, where the mean shift image segmentation algorithm extracts the regions. Another example is the work of Cheng *et al*. [2015], which efficiently computes pixel-level global contrast (using a global histogram) and later improves the intermediary results with a region-level analysis and spatial weighting. This work extracts regions using Felzenszwalb and Huttenlocher's segmentation method [Felzenszwalb and Huttenlocher, 2004]. Concurrently, as more images became available, researchers began incorporating extrinsic features, enabling SOD methods to leverage similar images as references to generate saliency maps [Wang *et al*., 2011].

A deeper review of SOD literature will also reveal that some assumptions improve the overall results. As is the case of *background prior* — assuming that pixels/blocks/regions at the image frame are the background, or conversely the *center prior*. However, they do not stand for all categories of images (*e.g.,* parasites in microscopy images can appear near the image's frame). For readers seeking a more comprehensive review of the techniques employed by bottom-up methods, we strongly recommend the survey conducted by Borji *et al*. [2019].

Nevertheless, despite the vast amount of bottom-up methods, its literature has consistently struggled with a critical weakness: poor generalization. As most approaches rely on low-level, hand-crafted features, applying them across domains proved highly challenging, particularly in complex scenes. The emergence of CNNs promoted a significant shift in the field, as researchers began leveraging deep learning ability to automatically extract and integrate low-level and high-level semantic features, producing more accurate and robust saliency maps across diverse visual contexts.

2.2 Deep Learning & Lightweight Models

Categorized as top-down approaches, deep-learning-based SOD methods (*deep SOD*) focus on learning high-level semantic features, commonly through supervision. Deep SOD employs an encoder that learns both low-level (*e.g.,* edges, corners) and high-level features (*e.g.,* complex textures, shapes) and, subsequently, a decoder that combines extracted features into a saliency object map — the pixel-wise prediction of the most visually important objects in the scene — with the size of the input image [Zhu *et al*., 2024].

Since 2015, when deep SOD was introduced, the majority of works in the field apply Deep Neural Networks (DNNs) to explore local and global features [Wang *et al*., 2015; Li and Yu, 2015; Zhao *et al*., 2015]. Classical methods in the past had struggled in both tasks, where methods based on local features are usually better at detecting salient objects' edges than their interior; conversely, global methods do not perform well when the object's textures are similar to the background.

Building this evolutionary shift in SOD methodology, Wang *et al*. [2015] proposed a saliency detection approach called LEGS (Local Estimation and Global Search) utilizing two deep-neural networks (DNNs), a local (DNN-L) and a global one (DNN-G). DNN-L focuses on local features, classifying a central pixel (of a 51×51 patch) as salient/non-salient. DNN-L is applied to the input image through a sliding window at inference and later refined through geodesic object proposal. Globally, a feature vector describes each region's global contrast, geometric information, and local saliency measurements (from DNN-L). DNN-G gets the global feature vector of each region as input and outputs the region's salient value, which is further sorted and yields the final saliency map through a weighted sum of the top K candidate masks. DNN-L comprises convolutional, max-pooling, and fully connected layers, while DNN-G is built only with fully connected layers.

In addition to exploring local and global features, the work

of Li and Yu [2015] proposes a multiscale approach using CNN features. Employing a neural network architecture with fully connected layers on top of CNNs, they extract features at three different spatial scales, generating saliency maps at different scales, which are then linearly combined. A key innovation was leveraging convolutional weights pretrained on the ImageNet dataset, enabling rich semantic understanding for visual saliency detection. Similarly, Zhao *et al*. [2015] also employed a single network to derive a saliency map, but through a dual branch architecture, a local-context branch and a global-context branch, which outputs are employed to classify a super-pixel (the super-pixel's center is the network's input) as salient/non-salient. They also carried out a more in-depth analysis of using pre-trained models.

The onset of larger datasets also played a central role in deep-learning adoption, such as MSRA-5000 [Liu *et al*., 2007], SOD [Movahedi and Elder, 2010], ECCSD [Yan *et al*., 2013], PASCAL-S [Li *et al*., 2014], MSRA10K [Cheng *et al*., 2015], and DUTS [Wang *et al*., 2017].

The seminal deep SOD works based on multi-layer perceptrons (*i.e.,* fully connected layers) leveraged image subunits and object proposals [Wang *et al*., 2015]. Image subunits include super-pixels and patches, while object proposals encompass regions and bounding boxes. These subunits are then scored as salient or non-salient. Despite outperforming bottom-up approaches, these methods cannot fully exploit spatial information and are computationally expensive at inference time because they must be applied to each image subunit independently.

To tackle the aforementioned problems of representation learning and inference time, the literature employed fully convolutional networks (FCN) [Liu and Han, 2016]. FCNs enable a single model capable of learning multi-level features simultaneously, which better explores spatial information. Furthermore, FCNs operate over the whole input image, requiring a single feed-forward process, significantly speeding up inferences. The seminal work of Liu and Han [2016] proposed the DHSNet (Deep Hierarchical Saliency Network), which is composed of a two-step architecture. First, they coarsely detect salient objects from a global perspective (VGG Convolutional encoder), and then, through recurrent convolutions, they refine the details of the saliency map in a step-by-step fashion.

Recent SOD models further explore multi-scale feature extractions to improve results and also loss functions to better guide the representational learning phase. Great examples are the works of Qin *et al*. [2019, 2020], which proposes the BasNet and U^2-Net architectures. BasNet is a deeply supervised encoder-decoder (*i.e.,* it employs deep supervision, computing loss, and backpropagating errors for each decoder level) [Qin *et al*., 2019]. Its weights are learned through a hybrid boundary-aware loss to learn at pixel, patches, and map levels (*i.e.,* BCE, SSIM, and IoU, respectively). Extending this approach, U^2-Net takes advantage of nested U-shaped architectures to improve high-level feature extraction at multiple resolutions (intra-stage multi-scale features) [Qin *et al*., 2020].

Still, there is an ongoing trend to enhance model architecture for better feature extraction across multiple scales. A good example is the work of Huang *et al*. [2023], which proposes the Multilevel Diverse Feature Aggregation Network (MDFNet), composed of different modules to capture saliency information in multiple directions and channels, learn the correlation between different salient objects, and improve the extraction of boundary information. The proposed method combines multilevel features and identifies suitable feature spaces and channels to produce better saliency maps. Although this trend yields models suitable for complex scenarios, the downside is increased complexity and, consequently, longer inference time.

The problem of high inference time is worst in scenarios where computing resources are limited, such as our problem of detecting parasite eggs in a tropical country (*e.g.,* Brazil), where laboratories usually do not have computers with GPUs. To address this issue, one could refer to literature on lightweight models concerned with lowering hardware requirements and speeding up inferences with competitive performance [Chen *et al*., 2024]. The first approach to generate a lightweight model focuses on architecture design, where fewer model parameters and computational complexity are possible through convolution paradigms (*e.g.,* dilated separable convolutions [Liu *et al*., 2021] and depth-wise convolutions [Sandler *et al*., 2018]) or AutoML [Chu *et al*., 2020]. On the other hand, the second approach employs model compression by optimizing larger models through model pruning, quantization, knowledge distillation, or low-rank decomposition [Phan *et al*., 2020; Chariton *et al*., 2022; Huang *et al*., 2024]. Despite those efforts, lightweight models still have around 1M to 5M parameters and require considerable training data. Furthermore, under constrained scenarios, high inference times are still prohibitive (*e.g.,* running for a slicing window of a microscopy slide).

More recent SOD research (2024–2026) has largely pushed representational power further rather than data efficiency. Three directions stand out: (i) sequence/state-space backbones, such as the Mamba-based *Samba* and its extension *Samba+*, which unify multiple SOD tasks under a single architecture with strong accuracy [He *et al*., 2025, 2026]; (ii) promptable and weakly-supervised formulations built on foundation models, such as *SSFam*, which adapts the Segment Anything Model to scribble supervision [Liu *et al*., 2024]; and (iii) efficiency-oriented designs, such as the difference-convolution network *SDNet*, which reaches real-time inference with fewer than one million parameters [Su *et al*., 2025]. A very recent line even couples diffusion priors with state-space models — *DGSSM* formulates multimodal SOD as a progressive denoising process [Ghosh *et al*., 2026] — underscoring how quickly the field is moving. These works confirm that the field keeps advancing, but they still assume either abundant annotated data (typically trained on DUTS-TR) or a large pre-trained foundation model, and most target natural-image or multi-modal (RGB-D/RGB-T) settings. Our contribution is orthogonal: rather than competing on representational capacity, we address the regime of scarce annotation and constrained compute with a flyweight ($\sim$400k-parameter), backpropagation-free network designed from 3–4 weakly-annotated images. We benchmark against representatives of these recent trends in Section 5; we do not include *DGSSM*, as no public implementation was available at the time of submission, which precludes the few-image fine-tuning that our

evaluation protocol requires.

To summarize, key questions remain for deep SOD models, mainly in scenarios where we do not have large annotated datasets and inference time is mandatory. Related to datasets, when few annotated data are available, how can we efficiently train a network? Should we consider training from scratch, or would transfer learning be a more practical approach? Regarding the model architecture, how can we discover the lightest network that can solve our problem? How many convolutional layers are required? And the correct number of filters? Those questions permeate not only the deep SOD field but deep learning in general.

2.3 Feature Learning from Image Markers

Over recent years, works employing the FLIM methodology have consistently demonstrated comparable, and even superior, results under a data-scarcity regime [Joao *et al*., 2023; de Souza and Falcão, 2022; de Souza *et al*., 2020]. Such scenarios are particularly challenging for deep learning models. Deep SOD models commonly overfit and do not generalize to unseen data when learning with little data. For instance, in Wang *et al*. [2022], they argue that SOD methods struggle when applied to unseen datasets, even when trained on thousands of data points. Hence, at least fine-tuning deep SOD methods for the target task is a must, but learning a different domain remains a problem.

The FLIM methodology (Figure 1) is a recently proposed approach [de Souza *et al*., 2020], which is under active investigation. It integrates user knowledge into the design of a feature extractor (*i.e.,* a convolutional encoder). User knowledge is exploited in different methodology phases, encompassing the selection of training images, identification of discriminative regions for filter estimation, and filter evaluation and selection. The methodology allows the design of a CNN encoder layer by layer. The user can create the encoder architecture by selecting representative images and drawing markers on descriptive and discriminative image regions. Conventional filters are then directly estimated from marked regions. Each FLIM encoder holds operations like marker-based normalization, the convolution with a filter bank, the activation function (*e.g.,*, ReLU), and pooling (*e.g.,*, max-pooling or averaging pooling). All operations, except marker-based normalization, are well-known in the literature. However, the below definition and geometrical interpretation in light of the FLIM methodology provide vital insights.

Let $\mathbf{I} = (D_I, \vec{I}) \in \mathcal{D}$ be an m-channel image from dataset $\mathcal{D}$, where $D_I \subset Z^2$ is the image domain and $\vec{I}(p) \in \mathbb{R}^m$ assigns feature values to pixel $p \in D_I$. A patch vector $\vec{P}_p \in \mathbb{R}^{k\times k\times m}$ vectorizes features in a $k \times k \times m$ square region centered at p. The same holds for grayscale ($m = 1$), color ($m = 3$), and feature maps ($m \geq 1$) from filter bank convolutions. It is important to understand that in Figure 1 we represented the FLIM's filter estimation and the extraction of features given an image as input (first convolutional layer). However, given features from a past layer, markers are projected into the feature space for a subsequent layer, so filters are learned directly from features.

The selection of training images is possible through multiple approaches: (1) random selection of a fixed number of images, (2) comprehensive dataset inspection to identify representative images — as demonstrated in our previous research [Salvagnini *et al*., 2024] —, or (3) iterative selection using validation metrics to build the training set progressively. We implemented the iterative approach in this work, although recent studies have demonstrated the effectiveness of supervised and unsupervised techniques for training image selection [Cerqueira *et al*., 2024a,b]. This step yelds a subset $C_I \subseteq \mathcal{D}$ (usually, $|C_I| << |\mathcal{D}|$) of a few representative images.

Given C_I, users mark discriminative regions (disks, scribbles) on each image $\mathbf{I} \in C_I$ while specifying architectural parameters—including filter counts per marker and image, and the target kernel bank size. This process extracts overlapping patches from marked regions, creating a set $\mathcal{M}_I$ of patch vectors $\vec{P}_p$ for pixels p marked in image $\mathbf{I}$. These user interactions, required only for training images, demand domain knowledge of both $\mathcal{D}$ and the application context.

With $\mathcal{M} = \bigcup_{\mathbf{I}\in C_I} \mathcal{M}_I$, patch vectors in $\mathcal{M}$ are normalized by z-score, and these normalization parameters are applied to all patch vectors from any image $\mathbf{I} \in \mathcal{D}$. This *marker-based image normalization* centralizes the dataset $\mathcal{M}$ and corrects distortions along the main axes of $\mathbb{R}^{k\times k\times m}$.

For each marker, the user specifies n filters, and k-means clustering of normalized patch vectors produces n clusters whose centers define kernel weight vectors $\vec{K}_i \in \mathbb{R}^{k\times k\times m}$. These kernels compose the filter bank, which may require further reduction (via k-means or PCA) if the total filter count exceeds either the per-image filter limit or the target number of filters specified for the layer. Convolving a normalized image $\mathbf{I}$ with kernel $\vec{K}_i$ produces an image $\mathbf{J}$ where $J_i(p) = \langle \vec{P}_p, \vec{K}i \rangle$. With a filter bank $\{\vec{K}_i\}_{i=1}^{n\times M}$ from M markers and n filters per marker, $\mathbf{J} = (D_J, \vec{J})$ becomes a feature map with $n\times M$ channels and $\vec{J}(p) = (J_1(p), J_2(p), \ldots, J_{n\times M}(p)) \in \mathbb{R}^{n\times M}$.

Geometrically, each kernel $\vec{K}_i$ represents a vector normal to a hyperplane through the origin of $\mathbb{R}^{k\times k\times m}$. The patch vector $\vec{P}_p$ exists as a point in this space, with $J_i(p)$ measuring the signed distance from this point to the hyperplane — positive or negative depending on which side the point falls. ReLU operations remove points on the negative side, while max-pooling consolidates nearby activations. Enforcing unit norm $\|\vec{K}_i\| = 1$ across all kernels prevents $J_i(p)$ amplification by kernel magnitude. The marker-based normalization eliminates bias by centering clusters around the origin of $\mathbb{R}^{k\times k\times m}$.

The operations described so far—patch extraction, marker-based normalization, and subsequent clustering for filter estimation from cluster centers—will repeat until the whole encoder architecture is designed. Layers other than layer one will use feature maps of the previous layer (extracted from $\mathcal{C}_I$ images) and project markers into the features' domain. At this point, we have a FLIM encoder suitable for extracting feature maps for any image in $\mathcal{D}$.

A key property of FLIM encoders highlighted in Joao *et al*. [2023] involves foreground-background differentiation. When markers are placed in distinct regions (Figure 2a), the resulting feature maps show channel-specific activation patterns for either foreground or background elements

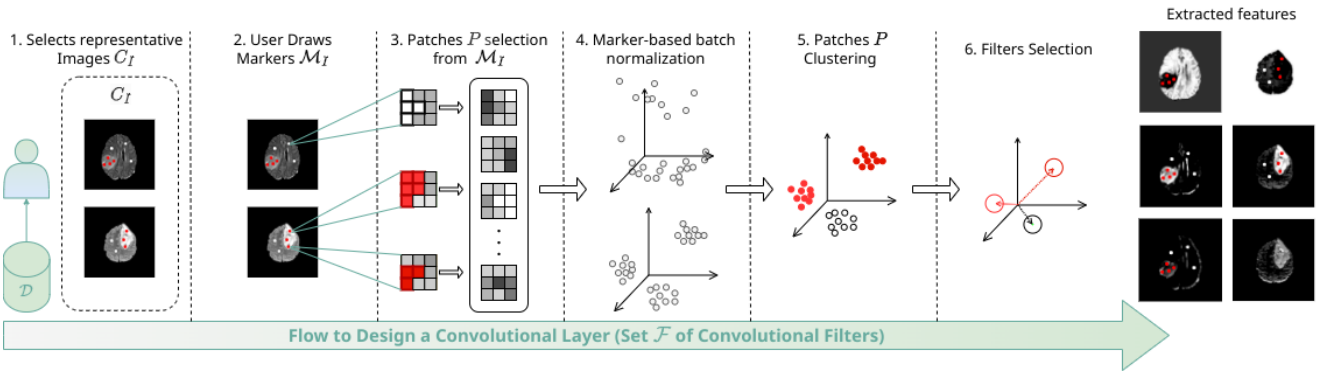
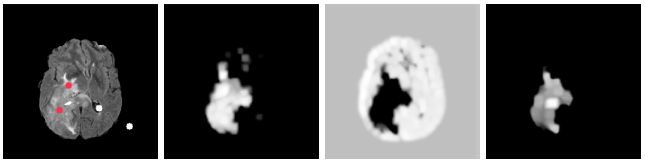


Figure 1. Flow to design a convolutional layer via FLIM: (1) A subset $C_I \subseteq \mathcal{D}$ of representative images is selected from dataset $\mathcal{D}$; (2) The user draws markers $\mathcal{M}_I$ on descriptive and discriminative regions of each image $\mathbf{I} \in C_I$; (3) Overlapping patch vectors $\vec{P}_p \in \mathbb{R}^{k\times k\times m}$ are extracted from the marked regions, forming the set $\mathcal{M} = \bigcup_{\mathbf{I}\in C_I} \mathcal{M}_I$; (4) Patch vectors undergo marker-based z-score normalization, centering $\mathcal{M}$ and correcting distortions in $\mathbb{R}^{k\times k\times m}$; (5) K-means clustering of normalized patches produces n cluster centers, defining the kernel weight vectors $\vec{K}_i$; (6) Filter selection consolidates the kernel bank $\{\vec{K}_i\}$, yielding the final set of convolutional filters. The rightmost panel shows the resulting extracted feature maps.



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

Figure 2. Adaptive decoder for FLIM-Encoder layer 3 (same across all l layers): (a) input with foreground/background markers; (b) channel i with $w_i = 1$ (foreground); (c) channel j with $w_j = -1$ (background); (d) resulting saliency map $\mathcal{S}_3$.

(Figures 2b-2c). For each layer l in L (total encoder layers), saliency maps $\mathcal{S}_l$ can be generated by applying a weighted channel averaging followed by activation function ϕ (Figure 2d). Optimal saliency maps with enhanced object visibility result from decoders assigning positive weights to foreground channels and negative weights to background channels, eliminating false positives from foreground maps. Since channel weight polarity varies with input images, adaptive decoders become necessary. The work of Joao *et al*. [2023] presents dataset-specific adaptive decoders, but we will follow with a modified version from Joao *et al*. [2024], employed in our previous work [Salvagnini *et al*., 2024], that aligns with our requirements for saliency map $\mathcal{S}_l$ estimation (Equation 1), where

$$\mathcal{S}_l(p) = \phi(\langle \vec{J}_l(p), \vec{w} \rangle), \tag{1}$$

where $\vec{w} = (w_1, w_2, \ldots, w_{n\times M})$, $\vec{J}_l(p)$ is the feature vector at position p for layer l, and ϕ is a given activation function (*e.g.,* ReLU). Each feature map channel weight is estimated as $w_i \in \{-1, 0, 1\}$, by

$$w_i = \begin{cases} +1, & \text{if} \quad \mu_{J_i} \leq \mathcal{T} - \sigma^2 \text{ and } a_i < A_1, \\ -1, & \text{if} \quad \mu_{J_i} \geq \mathcal{T} + \sigma^2 \text{ and } a_i > A_2, \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

with μ_{J_i} denoting the mean activation value of channel i of a given feature map, $\mathcal{T}$ the Otsu threshold computed over the vector of per-channel mean activations $\{\mu_{J_i}\}_{i=1}^{n\times M}$, and σ^2 the variance of the same vector. The parameter a_i measures the relative activated area of channel i — that is, the fraction of pixels whose activation exceeds the channel's own Otsu threshold — while A_1 and A_2 are area thresholds that distinguish spatially selective channels from broadly responding ones. A channel is assigned $w_i = +1$ (foreground) when its mean response is low, and its activation is spatially concentrated ($a_i < A_1$), reflecting the tendency of foreground filters to respond selectively to small, discriminative regions. Conversely, $w_i = -1$ (background) is assigned when the channel exhibits high mean activation spread over a large area ($a_i > A_2$), consistent with background filters that respond diffusely across uninformative regions. Channels that satisfy neither condition receive $w_i = 0$ and are excluded from the saliency map computation. Additionally, before weight estimation, each feature map channel undergoes min-max normalization to the $[0, 1]$ range, ensuring that the thresholds and mean activations operate on a consistent scale across channels.

At this point, we have a FLIM network trained without backpropagation and consequently without requiring many pixel-wisely annotated ground truths. We can now run it in inference for training, validation, and testing images, and extract saliency maps at any encoder level (*i.e.,* layer).

2.4 Cellular Automata

First introduced in 1951 by John Von Neumann [von Neumann, 1951], Cellular Automata (CA) — or Cellular Automaton in singular — refers to a discrete evolving model capable of mapping complex behaviors through relatively simple evolution rules. At its core, a CA consists of a regular arrangement of cells, typically organized in a lattice structure. Common examples include grids of cells (analogous to pixels) for 2D images or cubes of cells (similar to voxels) for 3D images. Nevertheless, cells can have formats other than squares and cubes, as complex surfaces can be represented by irregular tessellations such as Voronoi diagrams or superpixels [Qin *et al*., 2015]. This adaptability, combined with the ability to model local iterations, which produce global emergent patterns, is suitable for SOD, where identifying visually distinct regions requires both local contrast analysis and global contextual understanding. Hence, by combining CA with FLIM networks, we create a hybrid approach suitable to tackle SOD challenges, where a FLIM network (FLIM convolutional encoder + adaptive decoder) initializes the CA. Thus, this approach leverages both the hierarchical feature

learning capabilities of a FLIM network and the iterative, rule-based refinement of a CA.

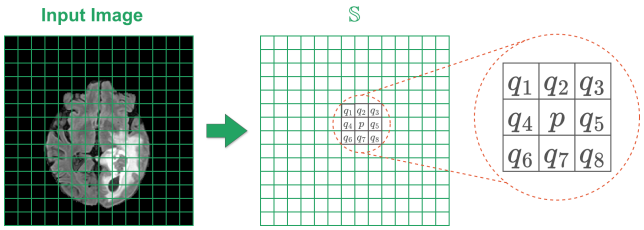


Figure 3. Brain image as a CA's lattice of cells, showing individual cells and their neighboring relationships.

When operating over images, the CA requires a set D_I, representing the image domain containing all cells (lattice of cells). Accordingly, we define the CA as a triple $(\mathbb{S}, N, \delta)$, which operates over the lattice of cells, dictating changes over time given each cell's state $\mathbb{S}_p$ based on the cells's neighborhood $N(p)$ and a defined local transition function $\delta : \mathbb{S}_p^t \rightarrow \mathbb{S}_p^{t+1}$. The cell's state evolves through discrete time steps t, where the state of a cell p at time $t+1$ depends on the configuration of states in its neighbors $N(p)$ at time t. Common neighborhoods are Von Neumann (4-neighbors) and Moore (8-neighbors).

CA applications on images range from classical image processing operations to more complex image analysis tasks. For example, in Hernández and Herrmann [1996], they show the application of CA towards sharpening and smoothing operations, establishing that CA's methods can converge fast and are also stable in the presence of noise, hence proposing CA as a first-level elementary image enhancement technique. They also discuss the inherently parallel nature of CA and demonstrate results on 2D-Images comparable to widely used software back then, without requiring a priori information. Fast forwarding some years, Rosin [2006] further experimented with CA over additional operations — noise filtering, thinning, and convex hulls — applied on binary and grayscale images. Additionally, the main focus of their work was to draw a method to enable easy training CA, which means designing a set of rules to compose the transition function δ. The authors discuss how even simple rules, combined with a matrix of cells with only local iterations, drive the system to map sophisticated global behaviors.

Then, building upon fundamental image processing techniques, other works have pushed CA applications to SOD. As a more complex task, SOD involves visual understanding, not only pixel-wise transformation (but also local and object-level understanding). A representative CA-SOD method is the work of Qin *et al*. [2015], which proposes two CA as post-processing operations capable of improving prior maps from all state-of-the-art methods at the time. First, the authors argue that most SOD methods exploit background priors, treating image boundaries as background. Despite this assumption holding for most images, the entire method will produce incorrect saliency when it fails. Hence, to overcome this issue, they begin by segmenting the input image into N small superpixels — using the SLIC algorithm [Achanta *et al*., 2012] — described by their mean color features and coordinates. Through the K-means algorithm, boundary superpixels are then clustered into K clusters based on the CIE LAB color feature. Background seeds are computed from clustered superpixels by combining geodesic and color information. Once background seeds are available, they evolve them using CA. Their approach treats each superpixel as a cell, and each cell has as neighbors the adjacent superpixels and cells with shared boundaries. An impact factor matrix and a coherence matrix then drive evolution, where the impact factor matrix measures the similarity in the CIE LAB color space between every cell, and the coherence matrix assesses the level of agreement between the state of the cell and its neighbors. As a final contribution, they propose a multi-level cellular automaton that combines the saliencies of multiple state-of-the-art methods, generating an improved saliency map. In this Multi-layer CA, each cell represents a pixel. Moreover, unlike the Single-layer CA, neighborhoods are also defined across saliency maps rather than only within a single map - specifically, pixels with the exact coordinates in different saliency maps are considered neighbors. They demonstrated results surpassing those of current methods using a Bayesian framework, effectively integrating the complementary strengths of different SOD methods.

Additionally, Qin *et al*. further contributed to the field by exploring a hierarchical CA towards SOD [Qin *et al*., 2018], again using a single- and a multi-level CA. This time, they extracted super-pixels for different image scales and deep features — extracted using FCN-32s [Shelhamer *et al*., 2017] — to guide evolution.

Semantic segmentation, another closely related image analysis domain where CA has been successfully applied, shares significant methodological overlap with SOD, where semantic segmentation is also concerned with classifying image pixels. However, it classifies every pixel/voxel between N classes (e.g., health tissue/tumor tissue), where SOD only concerns detecting the most salient object of the scene (.e.g., tumor tissue). Nevertheless, given the overlapping between the fields, we can drive inspiration from CA for semantic segmentation, mainly when applied to medical image analysis.

For example, in Vezhnevets and Konouchine [2005], the authors proposed a CA algorithm for semantic segmentation called GrowCut. Given an N-dimensional image as a lattice of cells, they describe each cell by a triple $(lb, \theta, \vec{C})$, where lb is the label of the cell, θ the strength, and $\vec{C}$ the RGB vector. Then, the user initializes seed cells by adding markers on images with labels (e.g., background or foreground, or more classes); simultaneously, the cell's strength (θ) is initialized. From these segmentation seeds, evolution happens, where the neighbors attack cells at each iteration. The attack force is controlled by the cell strength weighted by a monotonous decreasing function, which computes the distance between RGB vectors. If an adjacent cell has a higher strength value and is very similar to the central cell, it will conquer the central cell. Moreover, they also enable boundary smoothness by adding the concept of enemies (surrounding cells with different labels). Thresholding the number of enemies, they forbid cells to attack or conquer them automatically. Following the same trend, Kauffmann and Piché [2010] explores seeded algorithms but embeds the Ford-Bellman algorithm as an evolution rule using GPU optimization. From K labeled seeds, initialized by a user, they evolve the CA to compute multiple shortest-path trees. Once CA converges, K-cuts generate the

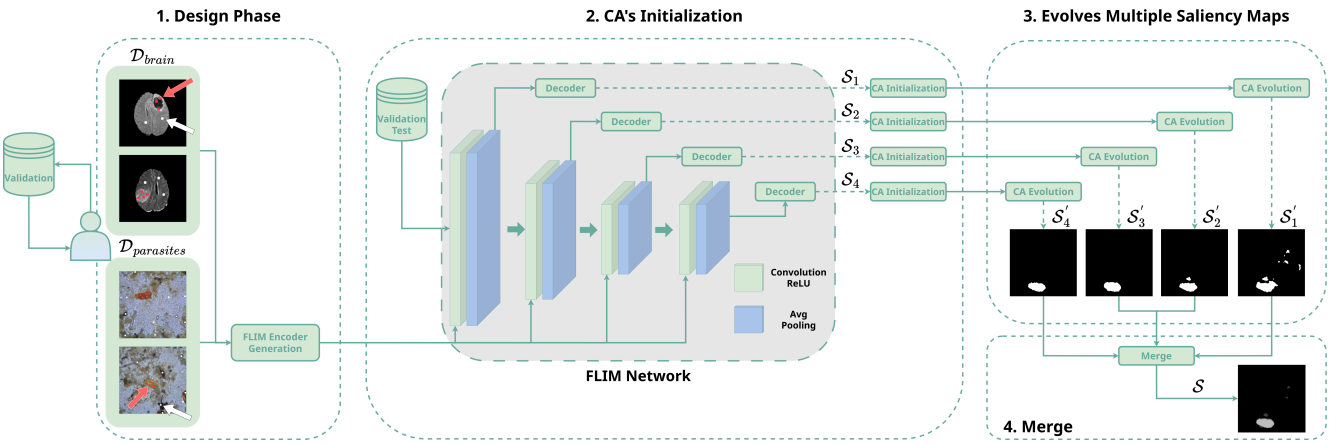


Figure 4. Multi-Level Cellular Automata for FLIM networks (example for a 4-layer FLIM Network): (1) Users iteratively select training images and define FLIM-Encoder architecture. (2) During inference, images enter the encoder, and features decode into intermediary saliency maps, initializing CA cells. (3) Multi-level saliency maps evolve from this initialization, and (4) merge into a final saliency.

final segmentation. Most interestingly, the authors integrate medical experts in the tests and show competitive performance (on low-cost graphics hardware suitable in 2010) for clinical routine. The possibility of implementing algorithms through CA is particularly interesting, as there is evidence that a CA with the proper rules can emulate a universal Turing machine and, therefore, implement anything computable [Cook *et al*., 2004]. Thus, it is possible to implement a more complex algorithm through carefully designed rules and parallelize it (given the characteristics of straightforward parallelization for CA).

The GrowCut algorithm was then adapted towards brain tumors in Hamamci *et al*. [2012] through the Tumor-Cut algorithm. Considering the characteristic necrotic core of brain tumor MRI images, they propose a two-step method, where the whole tumor is first segmented. Then, the necrotic core is segmented through a CA-based approach. This time, user input must be a line drawn over the largest visible diameter of the tumor, which initializes cells with labels and strengths for the foreground (tumor). Additionally, it generates a volume of interest (VOI) over a sphere, which is 35% larger than the drawn line. The VOI's border then initializes the background's labels and strengths. By evolving the foreground and background strengths, a tumor probability map is then calculated, which is used to evolve a level set surface to impose spatial smoothness, generating the final tumor segmentation. The necrotic core is segmented by computing two thresholds, one for necrotic voxels and the other for enhanced voxels. These thresholded voxels are initial seeds for a cellular automata algorithm that labels the remaining mid-intensity voxels. Years ago, the tumor cut algorithm was expanded to consider a second-order statistic (gray-level co-occurrence matrix) to evolve CA [Sompong and Wongthanavasu, 2017].

Despite further modifications, all methods derived from GrowCut share the same evolving characteristic: a cell q propagates its state to a cell p at time $t+1$ if both p and q cells are similar (close pixel color, intensity, or features) and if q has more strength than p given the monotonous decreasing function g, which means $g(p,q) \times \theta^t_{lb}(q) > \theta^t_{lb}(p)$.

A review of CA for both SOD and semantic segmentation methods reveals that CA enables applications across natural and medical images. Nevertheless, most works initialize the CA via user interaction, which is acceptable for a few images but burdensome, costly, and error-prone when running on many images. Given the work previously discussed on FLIM, the possibility of integrating FLIM networks with CA initialization arises. Interestingly, we can initialize CA states ($\mathbb{S}$) using prior user knowledge without requiring user interaction per image. Hence, initializing CA employing a FLIM network is of particular interest.

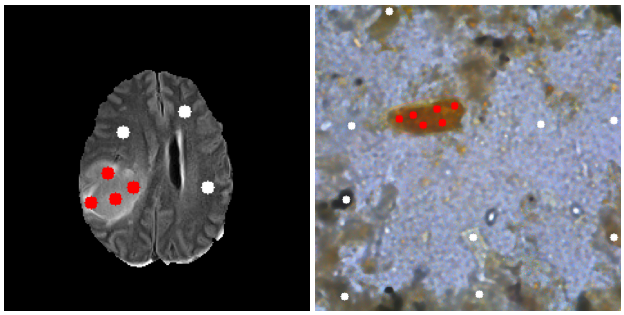
3 Multi-level Cellular Automata

Our method leverages FLIM to address the limitations of the deep SOD literature. Figure 4 illustrates our proposed method implemented via a 4-level FLIM network, though the framework is adaptable to FLIM architectures with any number of layers (L). This section details each step of our method. We begin with the design phase, guiding the reader through selecting images and defining the architecture of a FLIM encoder. Then, we address how the decoded features are employed to initialize the CA's states. Once the multi-level CAs are initialized, we detail how they evolve and yield a probability map. Finally, we describe our approach to merging multiple CAs' outputs into a single, refined saliency map.

3.1 Design Phase

The design phase focuses on training a FLIM Convolutional Encoder, the core component of the FLIM network. This first step is the only stage in our method that requires user interaction, which occurs in two specific, well-defined steps: First, as discussed in Section 2.3, the user selects a set of representative images (C_I) following an iterative procedure. Second, given the selection of an initial image, the user places markers (dots) in descriptive and discriminative image regions. Figure 5 shows two examples of user markers inserted on selected images.

The designed FLIM encoder is then combined with the adaptive decoder (Equation 1), which generates saliency maps for each network layer, such as $S_l = \{\mathcal{S}_l\}_{l=1}^{L}$, in which L is the number of layers of the convolutional encoder. Then, the quality of saliency maps is assessed on a validation subset.



(a) Markers on BraTS' data **(b)** Markers on parasites' data

Figure 5. Examples of user annotations: red markers indicate tumors or parasite eggs; white markers show healthy brain tissue, background, or debris.

The worst cases on validation may be selected to compose the training data, repeating the previous two steps. The iterative procedure is then stopped when acceptable performance is achieved, and this usually happens around 3 to 4 images for the evaluated datasets. Additionally, to supervise which image regions compose the convolutional filters, the user has complete control of the encoder's architecture through a configuration file.

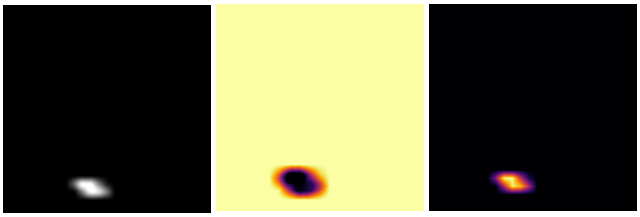
3.2 CA's Initialization

Once we train the FLIM encoder, we can extract features at each encoder level by feeding it an input image. User interaction is only needed during the design phase. Now, the FLIM encoder is run in inference mode for remaining validation (excluding training images) and test images; validation is used to guide CA parametrization and test to assess our final method. We then sequentially decode features at each level, generating an intermediate saliency map to initialize the state of the CA. Combining the FLIM encoder and adaptive decoders forms a *FLIM network* [Soares *et al*., 2024; Salvagnini *et al*., 2024], an architecture designed for extracting saliency maps. This L-layers network is developed end-to-end without requiring backpropagation and eliminates the need for pixel-wise annotated ground truths.

The CA's initialization corresponds to the second block in Figure 4, exemplified for a 4-layer FLIM network towards parasite egg detection; nevertheless, the method is suitable for encoders with any number of layers. We decode each encoder's features and generate an intermediary saliency map for each encoder level. Consequently, we generate a set of L intermediary saliency maps: $S_l = \{\mathcal{S}_l\}_{l=1}^{L}$. Each intermediary saliency map initializes one CA, which represents a significant improvement over our previous work by leveraging the delineation-to-detection capability of FLIM-Networks (see Figures 14b to 14e). Using each FLIM network level, we can initialize multiple Cellular Automata (CAs) — one for each network layer — which expands our initialization space. This multi-level CA approach better captures both edge and internal regions of salient objects while effectively reducing false positives (see Figures 14 and 15).

Formally, given the set of intermediary saliency maps $S = \{\mathcal{S}_l\}_{l=1}^{L}$ decoded from the L encoder levels, we instantiate a *family* of L cellular automata $\{\mathcal{A}_l\}_{l=1}^{L}$. Following the definition of Section 2, each automaton $\mathcal{A}_l = (\mathbb{S}, N, \delta)$ shares the same state space $\mathbb{S}$, the same Moore neighborhood N, and the same transition rule δ (Equation 3); the automata differ *only* in their initial state, derived from the corresponding saliency map $\mathcal{S}_l$. The state of a cell p is the triple $\mathbb{S}_p = \big(LM(p),\, \theta_0(p),\, \theta_1(p)\big)$, where $LM(p) \in \{0, 1\}$ is the current label and $\theta_0(p), \theta_1(p) \in [0, 1]$ are the background and foreground strengths, respectively. Crucially, the L automata evolve *independently* — there is no exchange of state across levels during evolution; the hierarchical information from different levels is combined only at the merging stage (Section 3.4). Each $\mathcal{A}_l$ yields an evolved saliency map $\mathcal{S}'_l$, and the multi-level CA is defined as this ensemble $\{\mathcal{A}_l\}_{l=1}^{L}$ together with the merge operator $\mathcal{F}$ that fuses their outputs into the final saliency.

To decode each layer feature map into an intermediary saliency map, we adopt a recently proposed adaptive decoder (Equation 2). Exploring decoders is outside the scope of this work, but we refer the reader to recent works that discuss decoders for FLIM networks [Joao *et al*., 2023; Soares *et al*., 2024; Joao *et al*., 2024].



(a) S_4 **(b)** θ_0^0 **(c)** θ_1^0

Figure 6. Example of initialization procedure for parasite eggs detection. FLIM-Network saliency from layer 4 (a) initializes background (b) and foreground strength (c).

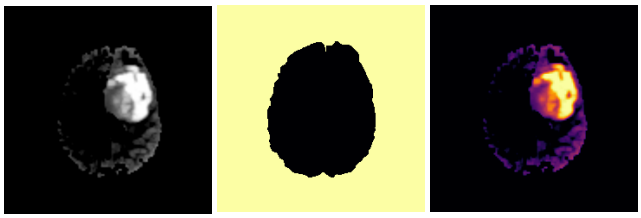
From the saliency map of each layer $S_l = \{\mathcal{S}_l\}_{l=1}^{L}$, we initialize the CA cell's states, where θ_0^0 and θ_1^0 represents background and foreground strengths, respectively, at $t = 0$. For θ_1^0, we simply normalize S_l within $[0, 1]$ and set cells' foreground strengths $\theta_1^0(p)$. However, the FLIM encoder may have pooling operations with strides larger than 1 for both datasets. If necessary, we upsample the saliency map S_l (bilinear upsampling) to the same size as the input image, as image color/intensity are employed to guide CA's evolution, and the loss of edge information through downsampling operations may degrade CA's performance.

Initializing the background strengths, θ_0^0, is more challenging; hence, our method considers two approaches based on the dataset at hand.

The first approach assumes that FLIM networks are good at detection and that the CAs can improve salience delineation while eliminating false positives. Hence, we initialize θ_0^0 as the complement of saliency maps (S_l) posterior to a dilation operation of radius 10. Figure 6 shows an example where no prior assumption can be made about where the salient object (parasite eggs) may appear. For example, background priors [Qin *et al*., 2018] are unsuitable, as eggs can be spotted at image borders.

The second approach concerns applications where a reasonable assumption about the background can be made. In such cases, we initialize the background strength as $\theta_0^0 \leftarrow 1$ based on a mask. Figure 7 exemplifies such an example, where the salient object is brain tumors. As brain tumors only appear inside the brain, we can employ a *brain prior* and use a brain

mask; background strengths are easily defined.



(a) S_1 **(b)** θ_0^0 **(c)** θ_1^0

Figure 7. Example of the initialization procedure for brain tumor detection. FLIM-Network saliency from layer 1 (a) initializes background (b) and foreground strength (c).

Lastly, we need to initialize the label map LM at $t = 0$, where we set $LM(p) \leftarrow 1$, if $\theta_1^0(p) > 0$, and 0 otherwise. The described initialization procedure is the same as that applied to each layer, and once it is done, we evolve each CA (one for each encoder level).

3.3 CA's Evolution

The CA's evolution is detailed in Algorithm 1, where the same algorithm details the evolution of each CA. Hence, what we describe here is the same as that applied to each CA; the only difference is how they were initialized. Once the cell's state is set, we evolve each CA state (*i.e.,* foreground and background) until convergence is reached — when there is little to no update on a given evolution step.

The evolution happens based on a Moore Neighborhood (8-neighbors), where we took inspiration from the tumor cut algorithm proposed by Hamamci *et al*. [2012]. We evolve for each object label, $ol \in \{0, 1\}$, background and foreground, respectively. The execution occurs first for $ol = 1$ (foreground) and later for $ol = 0$ (background).

Given an input image I, initial strength θ_{ol}^0, and label map LM, the algorithm begins by initializing a control variable $dist$ (Line 3) used for convergence detection in the main loop (Line 4). In each iteration, the future cell's strength θ_{ol}^{t+1} is copied from the current strength (Line 5) to keep track of unconquered cells.

The core evolution process (Lines 6-16) evolves each cell p in the domain. For each cell, the algorithm sets the q_{max} as the current strength of p (Line 7) and then evaluates all neighboring pixels q in the Moore neighborhood (Line 8). For each neighbor, it calculates q_{aux} as the product of a similarity function $g(p, q)$ and the neighbor's current strength $\theta_l^t(q)$ (Line 9). If this calculated value exceeds the current maximum strength q_{max} (Line 9), the cell p is conquered (Lines 11-13).

After processing all pixels, the algorithm computes the L2-norm between current and future strengths normalized by domain size to measure convergence (Line 17). The current strength θ_l^t is updated with the future strength (Line 18), and the iteration counter t is incremented (Line 19). This process continues until the convergence is reached ($dist$ distance falls below 10^{-8}, indicating convergence.

Algorithm 1 realizes the local transition rule δ in closed form. For a given object label $ol \in \{0, 1\}$, the strength of

Algorithm 1 Cellular Automata initialized by FLIM

Input:
$\mathbf{I}$ - input image, where $\hat{\mathbf{I}} = (D, \mathbf{I})$;
θ - Cells strength, initialized by FLIM, where $\hat{\theta} = (D, \theta)$;
ol - object label being evolved (0 | 1, background | foreground);
LM - Cells' current label, where $\hat{LM} = (D, LM)$.
Output: Evolved Cells' strength.

1: **procedure** CA($\mathbf{I}$, θ_{ol}^0, ol, LM)
2: $t \leftarrow 0$
3: $dist \leftarrow +\infty$
4: **while** $dist > 10^{-8}$ **do**
5: $\theta_{ol}^{t+1} \leftarrow \theta_{ol}^t$ ▷ Copy previous θ
6: **for** $\forall p \in D$ **do**
7: $q_{max} \leftarrow \theta_{ol}^t(p)$
8: **for** $\forall q \in N(p)$ **do** ▷ Moore
9: $q_{aux} = g(p, q) \times \theta_{ol}^t(q)$
10: **if** $q_{aux} > q_{max}$ **then**
11: $\theta_{ol}^{t+1}(p) \leftarrow q_{aux}$
12: $LM(p) \leftarrow LM(q)$
13: $q_{max} \leftarrow q_{aux}$
14: **end if**
15: **end for**
16: **end for**
17: $dist \leftarrow ||\theta_{ol}^t - \theta_{ol}^{t+1}||_2/|D_I|$
18: $\theta_{ol}^t \leftarrow \theta_{ol}^{t+1}$ ▷ Update step
19: $t \leftarrow t + 1$
20: **end while**
21: **return** θ_{ol}^t
22: **end procedure**

each cell p is updated as

$$\theta_{ol}^{t+1}(p) = \max\left(\theta_{ol}^t(p), \max_{q \in N(p)} g(p, q)\, \theta_{ol}^t(q)\right), \quad (3)$$

and the label is propagated from the conquering neighbor,

$$LM^{t+1}(p) = \begin{cases} LM^t(\hat{q}), & \text{if } g(p, \hat{q})\, \theta_{ol}^t(\hat{q}) > \theta_{ol}^t(p), \\ LM^t(p), & \text{otherwise,} \end{cases} \quad (4)$$

where $\hat{q} = \arg\max_{q \in N(p)} g(p, q)\, \theta_{ol}^t(q)$. A cell is thus *conquered* by its strongest sufficiently-similar neighbor, exactly as in the GrowCut family (Section 2). The similarity function $g(p, q)$ is computed according to Equation 5, which incorporates image features (intensity/color) to guide the evolution process. The β parameter smooths the weight reduction during evolution if evolving inside the object (see Section 4.2), which means that a lower value will cause foreground regions to leak. In contrast, higher values will penalize the saliency map on edge regions, as cells will lose strength fast and may not fit correctly to the object border.

$$g(p, q) = \begin{cases} e^{-\beta||\vec{I}(p) - \vec{I}(q)||_2}, & \text{if evolving object} \\ e^{-||\vec{I}(p) - \vec{I}(q)||_2}, & \text{otherwise,} \end{cases} \quad (5)$$

The input image features $\vec{I}(\cdot)$ are normalized to $[0, 1]$ (grayscale intensities divided by their maximum; color images in a normalized LAB space), and the cell strengths are

likewise initialized in $[0, 1]$. Since $\|\vec{I}(p) - \vec{I}(q)\|_2 \geq 0$ and $\beta > 0$, the monotonically decreasing weight satisfies $g(p, q) \in (0, 1]$, so a conquered cell never amplifies its neighbor's strength. Consequently, each cell's strength is monotonically non-decreasing along the evolution and upper-bounded by the global maximum of the initial strengths (itself non-increasing over time, as $g \leq 1$). This bounded-monotone behavior guarantees that the CA reaches a fixed point, which is detected when the mean strength update $dist$ falls below 10^{-8} (Algorithm 1). The normalization further keeps g bounded away from zero and makes β and the color-distance threshold comparable across datasets. Section 5.5 reports the empirical counterpart of this argument: the measured number of iterations until $dist$ falls below 10^{-8} across both datasets (Figure 21), confirming the monotone, bounded convergence behavior predicted here.

Once convergence is reached, combining foreground (θ_1^t) and background (θ_0^t) strengths generates a object probability map O_l (Equation 6), and we define a binary saliency map S_l' by thresholding O_l. Figure 8 exemplifies the whole evolution pipeline, where from FLIM-Network initialization, we evolve the foreground and background strengths (middle column), which enables us to compute the object probability map (O_l) and thresholding the parasite egg.

$$O_l(p) = \frac{ln(\theta_0^t(p))}{ln(\theta_0^t(p)) + ln(\theta_1^t(p))} \tag{6}$$

As exemplified in Figure 4, the CA evolution is performed for each FLIM-Network level, producing L saliency maps, which explores multi-level hierarchical features. Through multiple initializations, we now have an ensemble of complementary saliency maps that can be combined to produce a more robust final segmentation result.

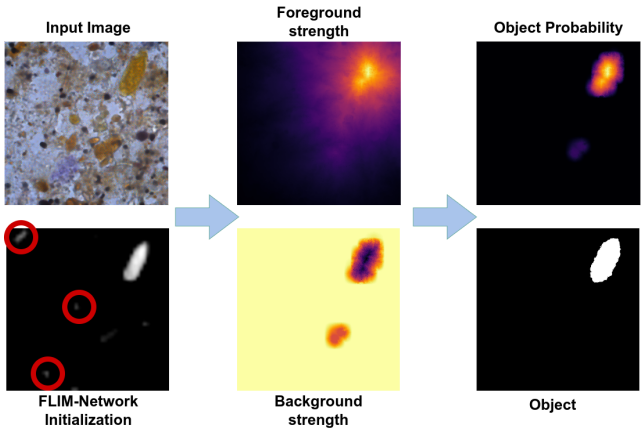


Figure 8. The first column shows the input image and FLIM initialization to CA. The middle column shows the evolved foreground and background strengths. The last column shows the object probability map and binary saliency. Our method evolves correctly even with false positives (red circles).

3.4 Merge Multiple Saliency Maps

Given the improved saliency maps from CA Evolution, we want to merge them into a single final salience map. As each evolved saliency corresponds to a hierarchical layer in our FLIM network, we want to leverage the best of each saliency without increasing the method's complexity. The inputs of the merging network are all evolved saliency maps $S_l' = \{\mathcal{S}_l'\}_{l=1}^{L}$ and the input image, which guides the merge procedure. During the initialization phase, if necessary, we upsample the saliencies to the size of the input image; then, the evolved saliencies are the same size. We designed a straightforward and efficient fully convolutional neural network to merge all evolved saliency maps into a final saliency map (guided by the input image). Figure 9 shows an example of such a merge network for a 4-layer FLIM network.

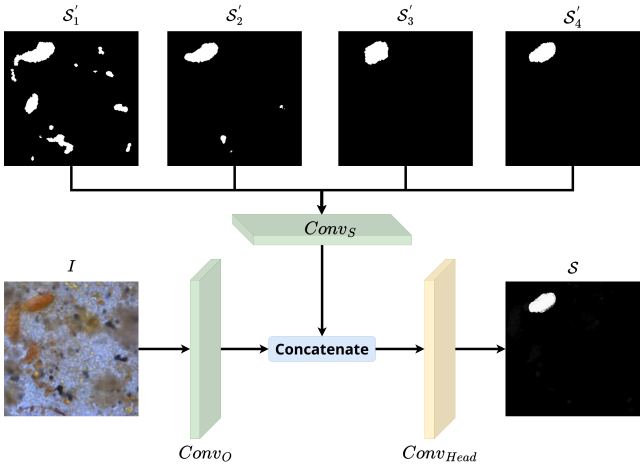


Figure 9. Merging multiple saliency maps from CAs supervised by input image.

The simplicity of this merging network is well-suited to our problem, where we face two main challenges: learning from a few images (3-4, the same used to train the FLIM convolutional encoder) and learning from imperfect saliency maps. We want to combine multiple-level saliency maps, merging correct regions and eliminating incorrect ones. We train this simple model for 2000 epochs, with a learning rate of (10^{-2}), Adam optimizer, and L1 regularization (with $\lambda = 10^{-3}$); we also employed the cosine annealing scheduler to take advantage of lower learning rates and learning resets. The key aspect of our training procedure was aggressive data augmentation, including random cropping, horizontal and vertical flipping, rotation, sharpness, affine, and perspective transformations. Our implementation guarantees that the same augmentations are applied to the multiple evolved saliencies and the input image.

The network is composed of 3 convolutional filters. First, a single $3 \times 3 \times c$ kernel ($Conv_O$), in which c is the number of channels of the input image (*e.g.,* 3 for RGB images and 1 for gray-scale ones), extracts features from the input image (I). In parallel, a single $3 \times 3 \times L$ kernel ($Conv_S$) combines the L saliency maps. Both convolutions generate a single-channel feature map, to which we apply a sigmoid activation function. The two feature maps are concatenated and fed to $Conv_{head}$, which generates the final saliency map with a point-wise convolution of $1 \times 1 \times 2$ followed by a sigmoid.

Formally, the merge operator $\mathcal{F}$ maps the L evolved saliency maps $\{\mathcal{S}_l'\}_{l=1}^{L}$ and the input image I to the final saliency $\mathcal{S}_{\text{final}} = \mathcal{F}\big(\{\mathcal{S}_l'\}_{l=1}^{L}, I\big)$. Because the network has only three small kernels, $\mathcal{F}$ can be written explicitly, pixel-wise. Let $\mathcal{W}(p)$ denote the 3×3 window centered at pixel p, and let $\sigma(\cdot)$ be the sigmoid. The two parallel branches

produce single-channel maps

$$h_S(p) = \sigma\Big(\sum_{u \in \mathcal{W}(p)} \sum_{l=1}^{L} K_S(u-p,\, l)\, \mathcal{S}'_l(u) + b_S \Big), \tag{7}$$

$$h_O(p) = \sigma\Big(\sum_{u \in \mathcal{W}(p)} \sum_{j=1}^{c} K_O(u-p,\, j)\, I_j(u) + b_O \Big), \tag{8}$$

where $K_S \in \mathbb{R}^{3\times3\times L}$ and $K_O \in \mathbb{R}^{3\times3\times c}$ are the saliency- and image-branch kernels, I_j is the j-th image channel, and b_S, b_O are biases. The point-wise head then fuses the two branches at each pixel independently,

$$\mathcal{S}_{\text{final}}(p) = \sigma\big(w_1\, h_S(p) + w_2\, h_O(p) + b_h\big), \tag{9}$$

with scalar weights w_1, w_2 and bias b_h of the $1 \times 1 \times 2$ head. This makes the multi-level fusion an explicit, differentiable operator over the CA ensemble — reducible to per-pixel arithmetic at inference — rather than a fixed layer-selection heuristic.

We empirically selected this architecture after observing that networks with fewer parameters (*e.g.,* a single shared convolution) cannot capture the complementary relationships between the multi-level saliency maps. In contrast, deeper or wider architectures tend to overfit to the limited training data (3–4 images) and fail to generalize to unseen images (*e.g.,*, on validation).

At this stage, our approach resembles an ensemble of CAs initialized through multi-level features, namely a multi-level CA. We can replicate the proposed method for multi-channel images, focusing on scenarios where collecting large datasets and pixel-wisely annotating images is problematic.

4 Experimental Setup

This section details the setup to validate the proposed method on two challenging datasets. We will guide the reader through building a FLIM network and its integration into the multi-level CA approach, discussing how to initialize, evolve, and merge the saliency maps. Additionally, this section also presents the SOD models used for comparison with our method, as well as the evaluation metrics employed for assessment.

4.1 Designing a FLIM Network

We choose two medical datasets to validate the proposed method: the BraTS 2021 [Baid *et al*., 2021] for brain tumor segmentation and a private dataset of parasite eggs. For BraTS, we extract 2D axial slices of brain tumors (T2-FLAIR), three slices for every image (given the tumor volume, we select axial indexes representing the median, 1st and 3rd quartile). For parasites, images were collected through a method (DAPI) developed in our lab focused on the automated diagnosis of human and pet parasites [Suzuki *et al*., 2013].

The problem of parasites is intrinsically related to the deep SOD problems, mainly inference time and annotation cost. Given an input slide, DAPI performs the screening and processing of around 2000 images with four million pixels each, segmenting and identifying objects of interest, where the whole pipeline must take less than 5 minutes. Recent works show that deep neural networks improve the diagnosis of gastrointestinal parasites in humans, cats, and dogs [Osaku *et al*., 2020; de Melo Joao *et al*., 2023]. However, given the need for less than 5-minute processing times, applying those methods without considerably increasing costs is impossible. Moreover, annotating the volume of ground-truth data required for training is both timely and financially expensive. Hence, we need lighter networks, and at the same time, we want to take advantage of weak annotations.

With the goal of broader validation of our method, we chose to explore the BraTS 2D dataset, as it enables us to assess our image in a different domain (MRI images), with grayscale images, and with a more heterogeneous dataset (multi-institutional dataset).

Once the datasets are prepared, we split them into three splits, each composed of a validation and a test subset, with a 50/50 ratio. For parasites, after the splitting, we also filter only images that contain parasite eggs. Hence, we end up with the following split configurations: 336/296, 308/324, and 326/306 (validation/test images). For the brain tumor dataset, the three splits have 1882/1882 validation and test images, as we guarantee during dataset construction that all images hold tumor regions.

The next step is to generate the architecture of the FLIM Encoder. As we previously detailed in Section 2.3, there are two steps where the FLIM methodology performs filter reduction and, hence, harnesses user control. First, when the number of filters extracted from one image (from all markers) is larger than the upper limit of filters set for each image, and second, if the total number of filters learned is larger than the number of filters specified in the convolutional layer. To avoid those reductions, we took the following decisions:

1. We only use disks as marks (*i.e.,* user clicks, see Figure 5), so we control the exact position that generates filters;
2. Each marker generates four filters;
3. The specified number of filters for each convolutional layer is 200, and given the number of selected images and drawn markers, we never surpass this number.

We proceed to image selection after specifying these requirements in the FLIM network. First, one image is randomly chosen from the validation subset, and we draw markers on descriptive and discriminative regions (Figure 5 exemplifies those markers). One can see this procedure as weak supervision once there is no need for a densely annotated ground truth, and filters are learned directly from marked regions. After a FLIM encoder is created, we apply the adaptive decoder described in Equation 1, which is empirically configured with $A_1 = 0.1$ and $A_2 = 0.2$ for the area thresholds. Then, we compute metrics (F-Score, Dice, MAE) for all validation data and select a new image for annotation based on the worst predictions, increasing the training subset. We repeat this procedure until the desired performance is achieved. It is a manual procedure, but there is parallel work on automating image selection for training [Cerqueira *et al*., 2024b,a].

A suitable architecture for CA initialization is typically achieved within 3 to 4 training images. Table 1 presents the encoder architecture for parasites, while Table 2 displays the 2D BraTS encoder architecture.

Table 1. FLIM Encoder for Parasites dataset (s for stride)

Layer	Conv	Activation	Pooling
1	(3×3)	ReLU	Avg $(3 \times 3 \vert s2)$
2	(3×3)	ReLU	Avg $(3 \times 3 \vert s2)$
3	(3×3)	ReLU	Avg $(3 \times 3 \vert s1)$
4	(3×3)	ReLU	Avg $(3 \times 3 \vert s1)$

Table 2. FLIM Encoder for BraTS 2D dataset (s for stride)

Layer	Conv	Activation	Pooling
1	(3×3)	ReLU	Max $(3 \times 3 \vert s2)$
2	(3×3)	ReLU	Max $(3 \times 3 \vert s1)$
3	(3×3)	ReLU	Max $(3 \times 3 \vert s1)$

As the number of filters depends on the number of markers on training images, they differ slightly across splits. On parasites, we have 129, 147, and 141 filters for the three splits, respectively. The number of filters across splits for brain tumors is 60, 48, and 68. All layers within each split maintain the same number of filters for both datasets.

We develop the FLIM-Encoder architectures empirically through an experimental analysis. A deeper study of encoder design is outside our scope, as our goal was to validate multi-level CA initialization using FLIM. Once this step is complete, we have a convolutional encoder (FLIM-based) ready for inference and for initializing CA cell states. By combining these encoders with our adaptive decoder (the same across all feature levels), we have a FLIM network.

The FLIM network consists of different numbers of layers for each dataset. Our method is evaluated using a 4-layer FLIM network for parasites and a 3-layer FLIM network for brain tumors. For the parasites, the saliency maps are represented as $[S_1, S_2, S_3, S_4]$, as illustrated in the method pipeline. For brain tumors, we generate three saliency maps $[S_1, S_2, S_3]$ from the 3-layer convolutional encoder.

4.2 Multi-level CA Parametrization and Execution

Given the two defined datasets and the extracted saliency maps, the foreground strengths of both datasets are initialized identically. However, as depicted in Section 3.2, background initialization demands domain-specific reasoning. For BraTS, tumors are anatomically constrained within the brain, making the *brain prior* straightforward and reliable — given a brain mask, background strengths are easily defined as $\theta_0^0 \leftarrow 0$ (inside the brain) and $\theta_0^0 \leftarrow 1$ (outside the brain). For parasites, eggs can appear anywhere in the field of view, including near image borders, making background priors unsuitable. Instead, we exploit the detection capability of FLIM networks directly: θ_0^0 is set as the complement of a dilation operation of radius 10 applied to the saliency map, reinforcing regions the network already considers background while creating a soft boundary around salient regions, giving the foreground CA room to grow and correctly delineate egg boundaries. For initialization examples, see Figures 6 and 7.

After all CA are initialized, we evolve them according to the Algorithm 1. To experiment with CA algorithm parametrization, we explored the validation data, and once we had selected its parameters (i.e., different thresholding on brain images), we used our test subsets to assess our results. Since we are dealing with gray-scale and RGB images, the first configuration specific for each dataset is the β parameter in Equation 5. For the brain, we smooth evolution when evolving in the foreground ($LM(q) = 1$), and if $I_1(p) > I_1(q)$ (gray-scale images), given the fact that the tumor core is darker than the surrounding tumor (edematous tissue). For parasites, we also verify if evolving on the foreground and the Euclidean distance in LAB color space ($\|\vec{I}(p) - \vec{I}(q)\|_2 < 0.2$). For both datasets, when smoothing $\beta = 0.6$.

Once evolution converged, we now have multiple object probability maps, $O_l = \{\mathcal{O}_l\}_{l=1}^{L}$, which must be thresholded to generate the improved saliency maps $S'_l = \{\mathcal{S}'_l\}_{l=1}^{L}$. Thresholding required special attention for each domain. For parasites, S'_l is obtained by first computing the morphological residual of O_l after a dome opening, followed by a morphological opening, and thresholding the result with Otsu (Figure 8 exemplifies the evolution end-to-end). For brain tumors, a different approach was required: rather than applying Otsu directly on O_l, a histogram-based statistical analysis of O_l values within the brain mask performs better. We focus on the upper-probability spectrum (top 10%) of O_l, thereby enhancing the detection of the characteristic peak corresponding to the entire tumor on T2-FLAIR images. After identifying this peak, we compute the mean (μ) and standard deviation (σ) of O_l values within a 20% window around it. The final threshold is then defined as $\tau = \mu - k\sigma$ (empirically, $k = 2.5$), classifying pixels with $O_l(p) \geq \tau$ as tumor tissue in S'_l. This approach better accommodates the distinctive high-probability characteristics of tumor regions while adaptively adjusting to image-specific distributions.

Our final step in multi-level CA now concerns merging the improved saliency maps $S'_l = \{\mathcal{S}'_l\}_{l=1}^{L}$. As shown in Figure 4, the CA evolution is performed for each FLIM-Network level, producing four saliency maps for parasite segmentation and three saliency maps for brain tumor detection, exploring multi-level hierarchical features, which now composes an ensemble of CA, or the multi-level CA, through the merging network. Through multiple initializations, we now have an ensemble of complementary saliency maps that can be combined to produce a more robust final segmentation result. The only difference between the merging network for the two chosen datasets happens in the input filters, where the input convolution has $c = 3$ (RGB) and $L = 4$ for parasites and $c = 1$ (gray-scale) and $L = 3$ for brain tumors.

For reproducibility, and to make explicit which choices are dictated by the method and which are dataset-specific, Table 3 consolidates all hyperparameters of the pipeline, reporting the value used for each dataset and how it was selected: *principled* (dictated by the method or the literature), *validation* (tuned on the validation subset), *empirical* (fixed by preliminary inspection), or *domain* (set from domain knowledge/priors). Most dataset-specific values concern only the background initialization and the object-probability thresholding, reflecting the different appearance of parasite eggs and brain tumors; the core CA formulation is shared across both applications.

We stress the methodological discipline behind these choices. All dataset-specific values were fixed using only the validation subset (Section 4); the test subsets remained untouched until the final evaluation, so every reported met-

Table 3. Consolidated hyperparameters of the multi-level CA pipeline. Selection: *P*rincipled, *V*alidation, *E*mpirical, *D*omain. "—" means not applicable to that dataset.

Stage	Parameter	Parasites	BraTS 2D	Sel.
FLIM design	Marker type	disks	disks	E
	Filters per marker	4	4	E
	Filters per layer (cap)	200	200	E
	Number of layers L	4	3	V
Adaptive decoder	Channel weights w_i	$\{-1, 0, 1\}$	$\{-1, 0, 1\}$	P
	Area thresholds A_1, A_2	0.1, 0.2	0.1, 0.2	E
CA initialization	FG strength θ_1^0	$\mathrm{norm}(\mathcal{S}_l)$	$\mathrm{norm}(\mathcal{S}_l)$	P
	BG strength θ_0^0	compl. of dilation	brain mask	D
	Dilation radius	10	—	E
CA evolution	Neighborhood N	Moore (8)	Moore (8)	P
	Convergence threshold	10^{-8}	10^{-8}	P
	β (evolving object)	0.6	0.6	V
	β (otherwise)	1.0	1.0	P
	Similarity feature	LAB color	gray intensity	D
	"inside object" cond.	$\|\vec{I}(p)-\vec{I}(q)\| < 0.2$	$I(p) > I(q)$	D
$O_l \to \mathcal{S}'_l$ thresh.	Method	dome+open+Otsu	histogram-based	V
	Peak window / factor k	—	20%, $k = 2.5$	E
	Component area filter	$[1000, 9000]$	$[100, 20000]$	E
Merge network	Convolutional filters	3	3	V
	Epochs	2000	2000	E
	Learn. rate / optim.	10^{-2} / Adam	10^{-2} / Adam	E
	L1 regularization λ	10^{-3}	10^{-3}	E
	Training images	3–4	3–4	P
Evaluation	F-Score / μWF β^2	0.3	0.3	P
	s-measure α	0.5	0.5	P

ric reflects held-out performance. Two design decisions act specifically as overfitting controls under extreme data scarcity: the CA is parameter-free — it only refines FLIM saliencies and adds no learnable weights — and the merge network is deliberately restricted to three convolutional filters, the smallest architecture that still captures the complementary information across levels, since larger merge networks overfit the 3–4 training images and degrade on validation (Section 5.1). Combined with a FLIM encoder learned without backpropagation from those same 3–4 images, the full pipeline exposes very few degrees of freedom, which is the intended behavior for scarce-annotation scenarios rather than a limitation.

4.3 Evaluation Metrics & Benchmark

To evaluate the efficacy of multi-level CA for FLIM-Networks, we employed the following metrics: F-Score, μWF, Dice, e-measure, s-measure, and MAE. All metrics, except for MAE, range from 0% to 100%, with higher values indicating better saliency maps.

The F-Score and μWF metrics assess the quality of the binarized saliency map (threshold set as 0.5) by combining precision and recall [Achanta *et al*., 2009; Margolin *et al*., 2014]. Following the literature, we empirically set $\beta^2 = 0.3$ to give more emphasis on precision. Similarly, Dice measures the overlapping ratio between the binary saliency map and the binary ground truth, making it particularly effective for evaluating improvements in edge regions [Qin *et al*., 2019]. The s-measure [Fan *et al*., 2017] evaluates the structural similarity between the normalized saliency map and the binary ground truth (with $\alpha = 0.5$). After subtracting their global means, the e-measure [Fan *et al*., 2018] quantifies the correlation between the binary saliency map and ground truth. The e-measure metric simultaneously accounts for global statistics and local pixel-level matching. MAE [Perazzi *et al*., 2012] computes the pixel-wise absolute difference between the normalized saliency map and the binary ground-truth mask, directly measuring prediction deviation. Before metrics computing, we filter components by area: $[1000, 9000]$ for parasites and $[100, 20000]$ for brain.

Leveraging the metrics above, we then compare our model to three recent lightweight models: SAMNet [Liu *et al*., 2021], MSCNet [Lin *et al*., 2022], and MEANet [Liang and Luo, 2024]; and to two recent heavier models: BasNet [Qin *et al*., 2019] and U^2-Net [Qin *et al*., 2020]. We further extend the comparison to four state-of-the-art methods published in 2024–2026: the efficiency-oriented SDNet [Su *et al*., 2025], the Mamba-based state-space models Samba [He *et al*., 2025] and Samba+ [He *et al*., 2026], and the SAM-based SSFam [Liu *et al*., 2024] (evaluated in its standard and scribble-click variants). Our goal is to benchmark our model against deep SOD, comparing the performance of those models under constrained scenarios where annotated data are scarce. Therefore, we fine-tune those models on the same training images employed in the FLIM design phase (3-4 images). Remarkably, FLIM networks are significantly smaller than lightweight networks.

To fine-tune the models, we utilized the weights released by the authors. For BasNet and U^2, trained on the augmented DUTs-TR dataset (21,106 images), we fine-tuned with a lower learning rate (10^{-4}) and additional augmentations (vertical/horizontal flips, rotation, color jittering, Gaussian noise). For lightweight models, we used the authors' pretrained models and code, where even the backbone had been previously fine-tuned on ImageNet. We adopted the same fine-tuning recipe — released weights, a lower learning rate, and the same

set of augmentations — for all heavier 2024–2026 models (Samba, Samba+, and SSFam). We stress that this augmentation was not a minor implementation detail but essential: with only 3–4 training images, fine-tuning these high-capacity models without augmentation led them to overfit almost immediately and fail to produce usable saliency maps. Applying the augmentations was therefore decisive for obtaining competitive results from the heavier models, and we made sure to grant them this advantage so that the comparison against FLIM remains as favorable to the baselines as possible.

5 Results & Discussion

This section presents and discusses our results, along with an in-depth statistical analysis of the proposed method's implications for FLIM networks. We then explore the proposed method's zero-shot generalization to unseen parasite species. Since our method targets real-world scenarios with scarce annotated data and constrained computing resources, we deliberately evaluate on domain-specific medical datasets; assessing performance on standard SOD benchmarks such as DUTS [Wang *et al*., 2017], ECSSD [Yan *et al*., 2013], or PASCAL-S [Li *et al*., 2014] is outside our current scope: these datasets belong to different visual domains (natural images), involve different object categories, and would require annotating new training images to design FLIM encoders, departing from our focus on scarce-annotation medical scenarios. For the same practical reasons, among the most recent methods we could not benchmark DGSSM [Ghosh *et al*., 2026], a 2026 diffusion-guided state-space model, as no public implementation was available at the time of submission, which precludes the few-image fine-tuning that our evaluation protocol requires. Moreover, there is parallel work on scaling FLIM to larger datasets with progressive filter reduction, which will enable those comparisons in future work. Finally, we analyze the computational complexity of our approach and present a GPU implementation that can leverage low-cost, off-the-shelf hardware to accelerate CA evolution substantially.

5.1 Per-Layer and Multi-Level CA Results

We evaluate our novel CA initialization method and explore an ensemble of multi-level CAs across two datasets: parasite eggs and brain tumors. For parasites, we seek saliency maps that correctly locate and delineate individual eggs. For brain tumors, we process T2-FLAIR images aiming to detect and delineate the whole tumor (a hyperintense signal on T2-FLAIR sequences).

The first step in enabling the proposed method involves selecting a random image to train an initial FLIM encoder. By decoding extracted features, we can compute metrics for all validation data, where images with bad performance are candidates for increasing the training data. Figure 10 exemplifies this procedure for building the FLIM encoder for brain tumor split 1. We see significant improvements over the analyzed metrics across each iteration. For instance, at iteration 1, with one image, F-Score values are around 0.3 to 0.35 for layers 1 to 3. However, as we add one more training image, there is a slight decrease in F-Score for layer 1, while we see an

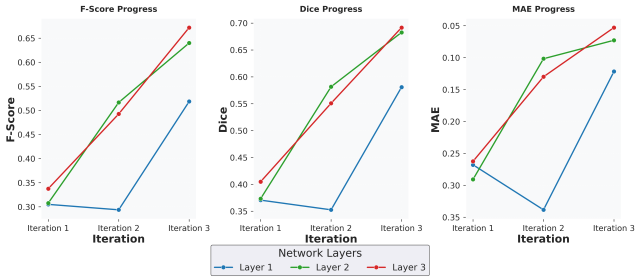


Figure 10. Metrics improvement on validation data across FLIM design. We start with one training image and at each iteration we add one more image.

increase of around 0.15 for layers 2 and 3. An improvement for all layers is observable once we add a third training image. Since layers 2 and 3 show metric values around 0.65 at this iteration, we assume the FLIM encoder is suitable for CA initialization, and the training procedure is done. This iterative procedure has two key benefits:

1. It allows choosing the right amount of training data;
2. It also allows the identification of the suitable network architecture for the given problem. For instance, if the performance of layer 3 is worse than that of layer 2, we can modify our FLIM encoder to have only two convolutional layers.

Once the design phase is done, we will have a FLIM network, and we can run inferences. We first extract each validation and test image's features, decode them into an intermediary saliency map (one for each encoder level), and then initialize our multiple CAs, where evolutions happen until convergence. We refer the reader to Algorithm 1 (Section 3) for the complete pseudocode of the CA initialization and evolution procedure, and to Equation 1 (Section 2.3) for the formal definition of the adaptive decoder.

Tables 4 and 6 summarize our results on test sets (average and standard deviation across splits) for parasites and BraTS 2D, respectively.

Table 4 demonstrates that multi-level CA significantly enhances saliency across all layers of parasites. Comparing intermediary saliency to CA output, we observed improvements in F-Score (by 2.8-17.7%), μWF (by 26.7-43.4%), Dice coefficient (by 26.2-41.3%), e-measure (by 22.4-34.4%), and s-measure (by 28.1-37.4%). For MAE, our method reduced error values by 0.091-0.31, significantly improving pixel-wise accuracy. These results confirm that our approach enhances precision, recall, delineation quality, structural similarity, and correlation between predictions and ground-truth masks while reducing overall error. Our method exhibits greater consistency, as demonstrated by lower standard deviation values across all metrics.

The improvements across all layers yield suitable saliency maps for merging, which can be characterized as a Multi-Level CA or a CA's ensemble (FLIM_{MCA}). On the merged column on Table 4, we visualize that by a simple network composed of 3 convolutional filters, we can generate a saliency map that combines multiple saliencies into a better one. Except for MAE, the merged saliency surpasses metrics on all layers. Nevertheless, even for MAE, the average metric is closer to the better layer (layers 2 and 3).

We empirically set the merging network to three convolutional filters. Increasing the number of filters consistently led

to overfitting, as the merge network was trained on only 3-4 images with imperfect input saliencies (the same 3-4 images used to train the FLIM encoder). Under such extreme data scarcity, a minimal architecture avoids memorizing training artifacts (*i.e.,* noisy) while still capturing the complementary information across layers. This design choice reflects the same principle that underlies our entire approach: simplicity and robustness in constrained data scenarios.

The merging network's potential is also confirmed when applied to FLIM-Network saliencies (saliencies employed to initialize the CAs). Even before CA, the merge network can improve the saliency map (e.g., 17.5% improvement on Dice). Besides exploring each saliency map's best features, the merge network also facilitates inferences using FLIM networks. When extracting multi-level saliency maps, we need to verify which layer yields better saliency (*e.g.,* we have a higher average dice on layer 3). Merging multiple saliencies eliminates this need. Also, since inference time is required, these three convolutions can be converted to simple arithmetic operations executed pixel-wise.

When we analyze a split, the same conclusions arise. Figure 11 shows metrics distribution (μWF) across each layer, and after merging, for split 1. An expressive improvement in the metric's average and quartiles is observed, ensuring better stability after running CA. More interestingly, we see the FLIM key capability to extract features and, consequently, saliencies, which shows a delineation-to-detection tendency. Initial layers have more evident edges that match ground-truth masks but may yield many false positives, resulting in a higher standard deviation for layer 1. As we go deeper into the network, edges get blurrier. However, false positives tend to disappear and, consequently, are better for CA initialization, as evolution tends to correct edge regions with lower confidence (i.e., blurrier). This behavior is evident in image 14 (14b-14e).

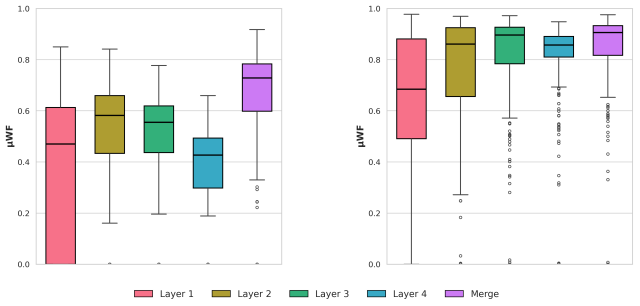


Figure 11. Metrics distribution across all layers (parasites data) on split 1. FLIM distribution on the left against FLIM_{MCA} on the right.

Table 5 then compares merged metrics (both FLIM_{M} and FLIM_{MCA}) with lightweight and conventional deep SOD. We see that FLIM-merged saliencies (FLIM_{M}) are comparable to both deep SOD methods, whereas FLIM_{MCA} outperforms every lightweight and conventional deep SOD method. Among the recent 2024–2026 approaches, only the state-space models Samba and Samba+ surpass FLIM_{MCA} on this dataset, and they do so at a substantially higher parameter and computational cost (Section 5.4); as shown next, their advantage does not carry over to the more heterogeneous BraTS 2D setting (Table 7). Moreover, lightweight models (MSCNet and MEANet) also outperform deeper models, showing that, under constrained scenarios (few training images), optimization over a more complex surface (more model parameters) is problematic, even with models pretrained on larger datasets (*i.e.,* DUTS-TR or ImageNet). As previously stated, lightweights target scenarios with limited computing resources, but, compared to FLIM-Networks, models are still larger. Take, for instance, MSCNet, with 3.6M parameters, against a FLIM network with 400k (11.11%) parameters for split 1, which substantially outperforms MSCNet with the multi-level CA framework. FLIM_{MCA} is a post-processing operation that does not increase the parameter count.

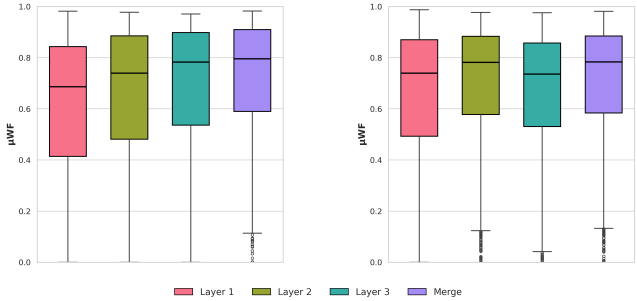


Figure 12. Metrics distribution across all layers (BraTS 2D data) on split 2. FLIM distribution on the left against FLIM_{MCA} on the right.

Continuing our analysis of saliency detection, now focusing on whole tumors in the BraTS 2D dataset, the same steps were used to design the FLIM encoder. Despite being grayscale, the BraTS 2D dataset is challenging due to data heterogeneity (the original 3D dataset is multi-institutional) and the need for domain adaptation during fine-tuning of deep-learning approaches (pretrained models were trained on RGB images).

Contrary to parasite experiments, except for F-Score and MAE, Table 6 shows improvements on BraTS 2D only on the first and second layers. Nevertheless, gains in metrics are substantial: F-Score (by 5.6-10.1%), μWF (by 2.2-6.5%), Dice coefficient (by 2.5-7.1%), e-measure (by 4.8-11.5%), and s-measure (by 6.1-13%), and on MAE we see and error reduction of 0.035 to 0.122. An analysis of a split's distribution (Figure 12) shows that the method saturates when it reaches layer 3. However, for all other layers, there is an improvement in both average metric value and stability (standard deviation), while some outliers remain (dots on the image). Still, we see a higher standard deviation across all layers for every split, which we believe is strongly related to data heterogeneity; hence, selecting more representative images for the FLIM encoder design is important.

Furthermore, as we see saturation on layer 3, experimenting with minor modifications to architecture design is also important. Examples are adding stride on layers 2 and 3 (pooling) and experimenting with average pooling. As overconfidence in edge regions causes the CA to leak the background into the foreground, blurrier edges would be of great importance and have the potential to overcome saturation. Also, a significant difference from our previous work is how we threshold the object probability map [Salvagnini *et al.*, 2024]. Instead of a fixed threshold, we employed a histogram-based threshold, significantly improving our results. We omitted those results for conciseness but experimented with fixed thresholds, the Otsu threshold computed using the brain mask, and different histogram-based thresholds. In summary, all approaches

Table 4. Metrics for parasites on test.

Metric	Model	Layer_1	Layer_2	Layer_3	Layer_4	Merged
F-Score	FLIM	0.467 ± 0.053	0.690 ± 0.022	0.726 ± 0.020	0.602 ± 0.027	0.821 ± 0.010
(↑)	FLIM_{CA}	0.604 ± 0.019	0.718 ± 0.020	0.780 ± 0.011	0.779 ± 0.011	0.834 ± 0.017
μWF	FLIM	0.310 ± 0.042	0.472 ± 0.019	0.480 ± 0.019	0.350 ± 0.018	0.660 ± 0.011
(↑)	FLIM_{CA}	0.632 ± 0.021	0.739 ± 0.017	0.795 ± 0.009	0.784 ± 0.011	0.816 ± 0.017
Dice	FLIM	0.322 ± 0.043	0.494 ± 0.020	0.505 ± 0.020	0.378 ± 0.019	0.680 ± 0.010
(↑)	FLIM_{CA}	0.657 ± 0.021	0.756 ± 0.015	0.805 ± 0.007	0.791 ± 0.009	0.823 ± 0.015
e-measure	FLIM	0.518 ± 0.058	0.688 ± 0.015	0.701 ± 0.011	0.598 ± 0.013	0.848 ± 0.014
(↑)	FLIM_{CA}	0.862 ± 0.005	0.912 ± 0.007	0.935 ± 0.007	0.936 ± 0.008	0.940 ± 0.004
s-measure	FLIM	0.404 ± 0.043	0.558 ± 0.014	0.584 ± 0.011	0.517 ± 0.012	0.756 ± 0.012
(↑)	FLIM_{CA}	0.778 ± 0.010	0.840 ± 0.009	0.867 ± 0.006	0.850 ± 0.006	0.879 ± 0.008
MAE	FLIM	0.337 ± 0.061	0.146 ± 0.013	0.110 ± 0.006	0.168 ± 0.014	0.057 ± 0.007
(↓)	FLIM_{CA}	0.030 ± 0.001	0.019 ± 0.001	0.019 ± 0.002	0.032 ± 0.003	0.031 ± 0.008

Table 5. Benchmark on parasites test set (FLIM Metrics stands for merged model).

Model	F-Score (↑)	μWF (↑)	Dice (↑)	e-measure (↑)	s-measure (↑)	MAE (↓)
SAMNet	0.518 ± 0.209	0.431 ± 0.262	0.440 ± 0.256	0.516 ± 0.276	0.478 ± 0.257	0.418 ± 0.242
MSCNet	0.730 ± 0.032	0.695 ± 0.009	0.709 ± 0.008	0.854 ± 0.017	0.775 ± 0.009	0.086 ± 0.023
MEANet	0.715 ± 0.039	0.655 ± 0.034	0.666 ± 0.034	0.797 ± 0.044	0.733 ± 0.035	0.150 ± 0.047
SDNet	0.741 ± 0.077	0.545 ± 0.060	0.663 ± 0.070	0.825 ± 0.036	0.786 ± 0.045	0.056 ± 0.016
FLIM_{M}	0.821 ± 0.010	0.660 ± 0.011	0.680 ± 0.010	0.848 ± 0.014	0.756 ± 0.012	0.057 ± 0.007
FLIM_{MCA}	0.834 ± 0.017	0.816 ± 0.017	0.823 ± 0.015	0.940 ± 0.004	0.879 ± 0.008	0.031 ± 0.008
BasNet	0.701 ± 0.118	0.654 ± 0.128	0.663 ± 0.126	0.786 ± 0.111	0.737 ± 0.109	0.155 ± 0.094
$\text{U}^{\mathbf{2}}$-Net	0.687 ± 0.085	0.617 ± 0.089	0.631 ± 0.089	0.760 ± 0.074	0.716 ± 0.074	0.158 ± 0.062
Samba	0.872 ± 0.036	0.849 ± 0.033	0.865 ± 0.036	0.958 ± 0.015	0.911 ± 0.024	0.016 ± 0.009
Samba+	0.872 ± 0.035	0.848 ± 0.039	0.866 ± 0.037	0.962 ± 0.013	0.914 ± 0.022	0.014 ± 0.004
SSFam	0.721 ± 0.033	0.715 ± 0.015	0.722 ± 0.028	0.877 ± 0.023	0.817 ± 0.024	0.043 ± 0.011
SSFam-FC	0.749 ± 0.045	0.698 ± 0.025	0.731 ± 0.032	0.884 ± 0.017	0.828 ± 0.022	0.036 ± 0.005

Table 6. Metrics for BraTS 2D on test set.

Metric	Model	Layer_1	Layer_2	Layer_3	Merged
F-Score	FLIM	0.516 ± 0.015	0.633 ± 0.022	0.673 ± 0.016	0.699 ± 0.034
(↑)	FLIM_{CA}	0.617 ± 0.007	0.718 ± 0.013	0.730 ± 0.015	0.712 ± 0.005
μWF	FLIM	0.577 ± 0.021	0.675 ± 0.020	0.688 ± 0.011	0.697 ± 0.013
(↑)	FLIM_{CA}	0.642 ± 0.012	0.698 ± 0.012	0.656 ± 0.025	0.710 ± 0.013
Dice	FLIM	0.574 ± 0.021	0.675 ± 0.018	0.689 ± 0.011	0.700 ± 0.015
(↑)	FLIM_{CA}	0.645 ± 0.012	0.701 ± 0.011	0.664 ± 0.024	0.713 ± 0.011
e-measure	FLIM	0.707 ± 0.035	0.822 ± 0.018	0.848 ± 0.015	0.841 ± 0.018
(↑)	FLIM_{CA}	0.822 ± 0.001	0.870 ± 0.005	0.833 ± 0.020	0.877 ± 0.007
s-measure	FLIM	0.624 ± 0.033	0.726 ± 0.013	0.756 ± 0.012	0.771 ± 0.017
(↑)	FLIM_{CA}	0.754 ± 0.006	0.787 ± 0.008	0.757 ± 0.017	0.799 ± 0.009
MAE	FLIM	0.166 ± 0.057	0.082 ± 0.012	0.069 ± 0.016	0.075 ± 0.021
(↓)	FLIM_{CA}	0.044 ± 0.001	0.027 ± 0.002	0.034 ± 0.004	0.032 ± 0.001

generated saliencies worse than FLIM. It emphasizes the key role of thresholding for tumor salient object detection, and there is room for improvement.

When we merge the multi-level saliencies, the same improvements observed on parasites stand out (see also Figure 12). However, on FLIM saliencies, for e-measure and MAE, merge statistics are closer to the best layer but do not surpass it. Merging saliencies also yields better metrics, except for F-Score and MAE, which are comparable to the best layer. Both results show that in the worst scenario, merging yields comparable results. Merging results on both datasets lay down the floor for further experiments with merging, as it shows that taking advantage of all layers' saliencies is possible and could improve overall results. Experiments with more complex networks, taking the proper care to avoid overfitting, have the potential to further improvements. Moreover, using the probability map instead of the binarized saliency could eliminate the necessity to compute thresholds. We tried the same merge networks using the probability map but without success.

This behavior also clarifies the relationship between the two merged variants on BraTS 2D. As the statistical analy-

Table 7. Benchmark on BraTS 2D test set (FLIM Metrics stands for merged model).

Model	F-Score (↑)	μWF (↑)	Dice (↑)	e-measure (↑)	s-measure (↑)	MAE (↓)
SAMNet	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	1.000 ± 0.000
MSCNet	0.431 ± 0.012	0.234 ± 0.003	0.269 ± 0.001	0.459 ± 0.028	0.509 ± 0.003	0.158 ± 0.017
MEANet	0.118 ± 0.020	0.116 ± 0.013	0.123 ± 0.014	0.680 ± 0.054	0.446 ± 0.036	0.149 ± 0.078
SDNet	0.598 ± 0.076	0.565 ± 0.048	0.576 ± 0.050	0.778 ± 0.054	0.699 ± 0.053	0.099 ± 0.055
$FLIM_M$	0.699 ± 0.034	0.697 ± 0.013	0.700 ± 0.015	0.841 ± 0.018	0.771 ± 0.017	0.075 ± 0.021
$FLIM_{MCA}$	0.712 ± 0.005	0.710 ± 0.013	0.713 ± 0.011	0.877 ± 0.007	0.799 ± 0.009	0.032 ± 0.001
BasNet	0.775 ± 0.022	0.751 ± 0.020	0.756 ± 0.019	0.880 ± 0.028	0.830 ± 0.011	0.024 ± 0.007
U^2-Net	0.728 ± 0.026	0.684 ± 0.017	0.692 ± 0.015	0.834 ± 0.033	0.785 ± 0.012	0.037 ± 0.012
Samba	0.240 ± 0.043	0.195 ± 0.012	0.211 ± 0.016	0.760 ± 0.067	0.518 ± 0.004	0.065 ± 0.011
Samba+	0.336 ± 0.024	0.231 ± 0.009	0.257 ± 0.007	0.652 ± 0.091	0.534 ± 0.007	0.079 ± 0.016
SSFam	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.011 ± 0.011	0.011 ± 0.010	0.978 ± 0.020
SSFam-FC	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.011 ± 0.007	0.011 ± 0.006	0.977 ± 0.012

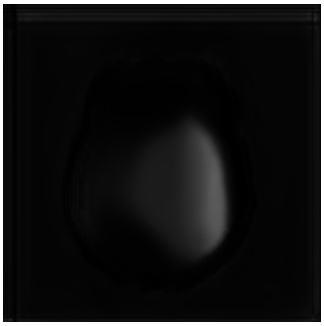


Figure 13. Sample of SAMNet's predictions.

sis in Section 5.2 shows, $FLIM_M$ and $FLIM_{MCA}$ are statistically comparable on this dataset, differing significantly in only one of the six metrics. This follows directly from the deep-layer degradation reported above: because CA evolution slightly worsens the third layer for several metrics on brain tumors (*e.g.,* μWF and Dice drop from 0.688 and 0.689 to 0.656 and 0.664 in Table 6), the merge of the CA-evolved layers ($FLIM_{MCA}$) inherits a weakened deep-layer input and no longer improves over the merge of the raw FLIM layers ($FLIM_M$). More broadly, this exposes a limitation of the current design rather than of the method itself: CA evolution and merging are treated as two independent, sequential stages, so the merge network cannot recover a saliency that CA has already degraded, and CA receives no feedback from the merging objective. Two directions seem promising for future work. The first is to optimize the multi-level CA and the merge step jointly, making evolution aware of the final fusion so that the two stages reinforce rather than constrain each other. The second is to replace the deterministic binary saliencies with a probabilistic formulation that propagates per-pixel confidence through both CA evolution and merging, allowing uncertain regions (such as the hypo-intense areas that trigger leakage) to be down-weighted instead of committed early. Both would address the fragility of the merge under data scarcity while preserving the backpropagation-free, low-parameter nature of the approach.

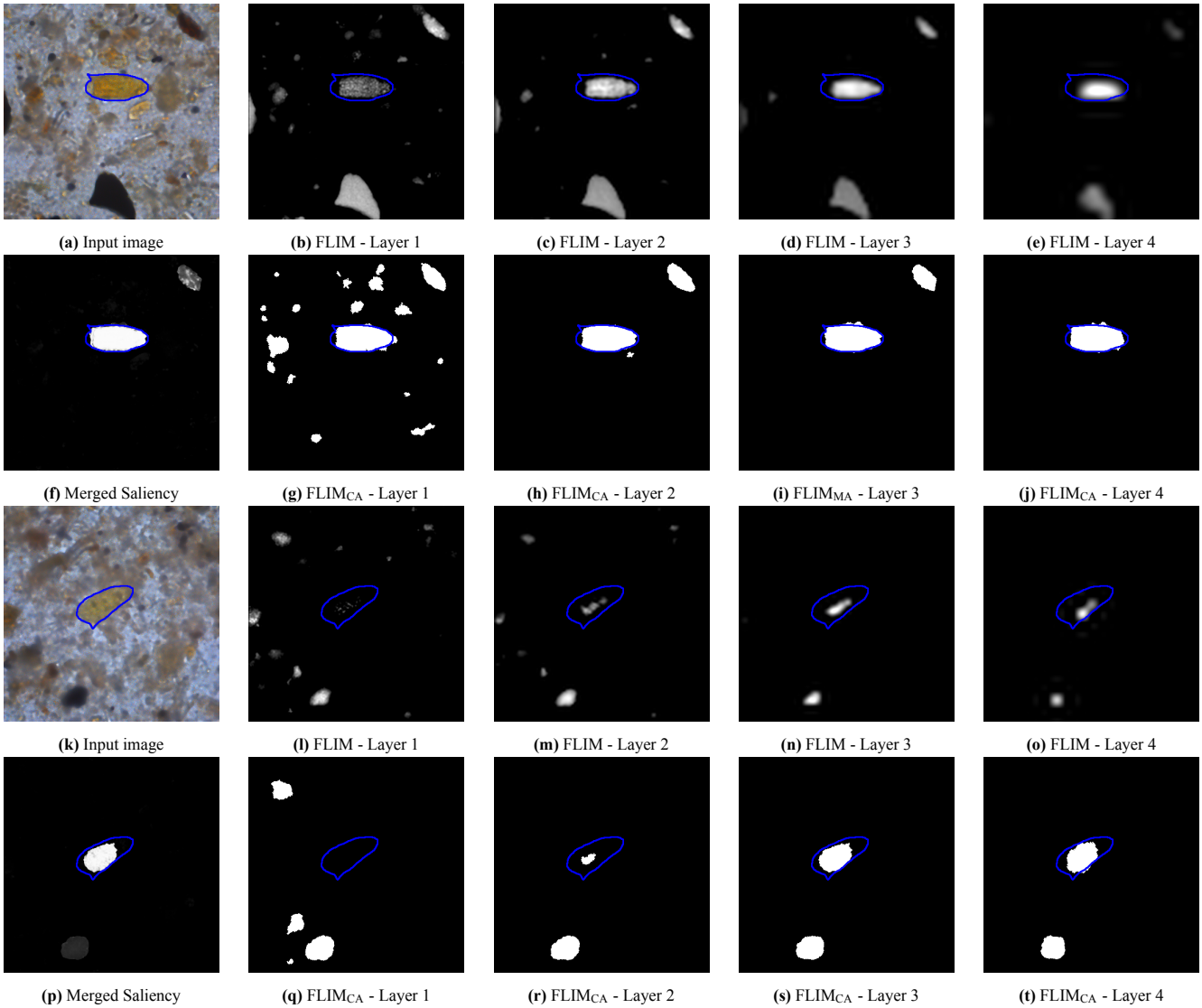
We emphasize that Tables 4 and 6 may also be interpreted as an ablation study of the multi-level approach. Each layer row isolates the contribution of a single-level CA, demonstrating the impact of initialization at different hierarchical levels. The merged column then shows the ensemble benefit of combining all levels. Comparing FLIM rows (before CA) with $FLIM_{CA}$ rows (after CA) directly quantifies the CA contribution at each level, while comparing individual layers against the merged result demonstrates the value of the multi-level ensemble. This analysis addresses both the impact of the number of CA levels and the role of FLIM-based initialization.

Comparing merged results with deep SOD literature shows a different behavior than the parasites' benchmark. First, lightweight networks fail fine-tuning to a different domain — pre-trained weights were learned on RGB images — when few training images are available. All networks fail to learn, even with our efforts to experiment with different learning rates to reach the best performance. SAMNet could not predict saliency maps, which detect tumors correctly, and saliency maps had all values below the threshold for metric computation (Figure 13). Thus, SAMNet has the worst metrics possible.

Distinctly from lightweight methods, deeper networks learned a different domain (gray-scale images), in which BasNet and U^2-Net generate better saliency maps than all lightweight models. BasNet has the best results. Nevertheless, $FLIM_M$ and $FLIM_{MCA}$ showed competitive results compared to BasNet, significantly outperforming U^2-Net on most metrics.

From a qualitative perspective, Figures 14 and 15 illustrate both the **strengths** and **limitations** of our method. We present successful and problematic cases to provide a thorough understanding of the method's behavior. Both images present the FLIM output (CA initialization), the CA output, and the merged saliency. We also provide two examples: one demonstrating a successful case and the other illustrating a problematic one. The blue line shows ground-truth edges.

Figure 14 (14a-14j) shows what we have already discussed. First, on layer 1, we see much noise (false positives), while the parasite's egg itself is darker than some of them. An analysis of deeper saliencies shows that values for the egg exceed false-positive values, while false positives also disappear. Finally, we observe a blurrier characteristic in the last layer and only two false-positive components. Despite the noise of initialization, we see that both the initialization and evolution procedures are robust and can eliminate most false positives (dark debris at the bottom of the image). On



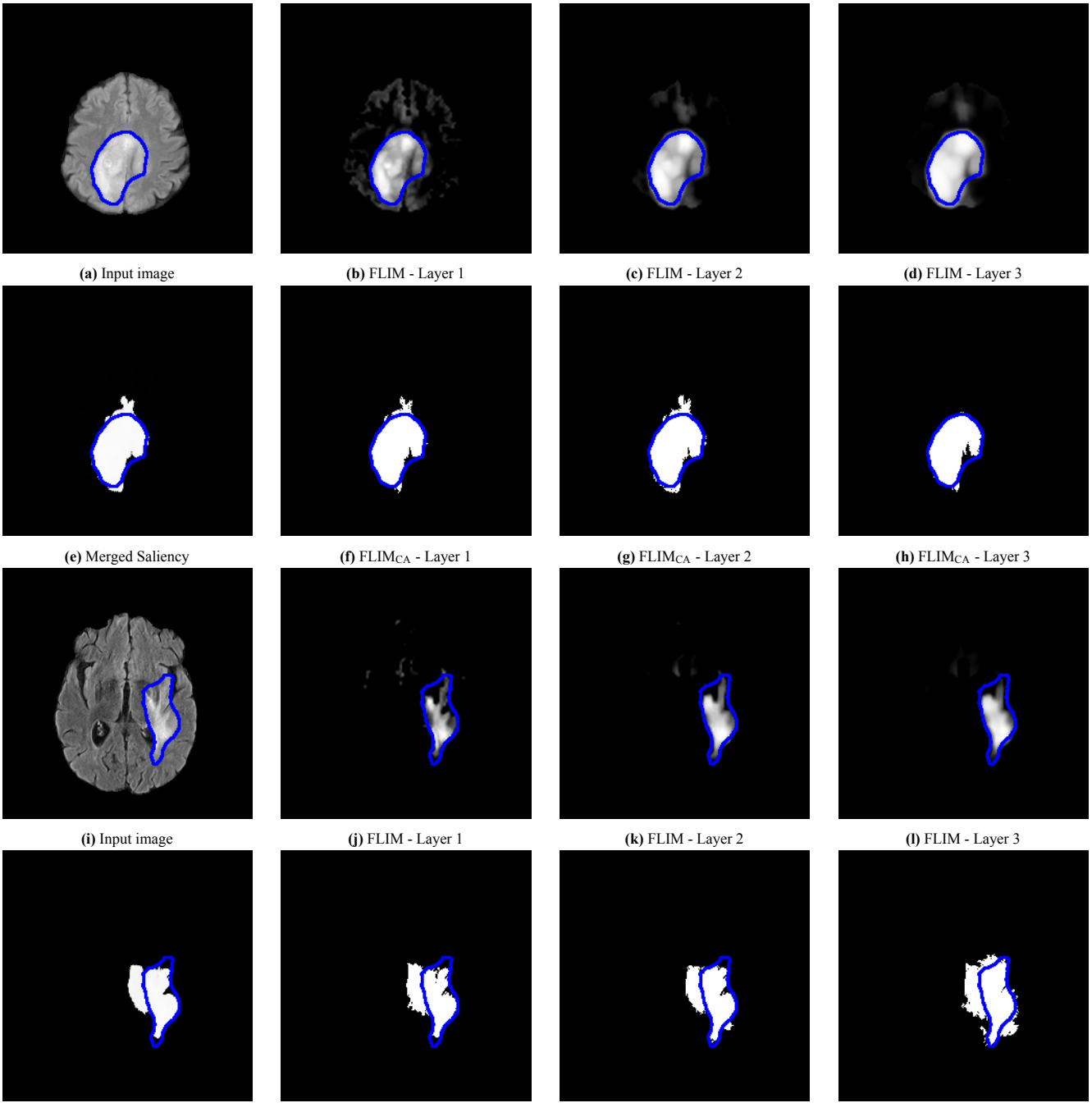
(a) Input image **(b)** FLIM - Layer 1 **(c)** FLIM - Layer 2 **(d)** FLIM - Layer 3 **(e)** FLIM - Layer 4

(f) Merged Saliency **(g)** FLIM_{CA} - Layer 1 **(h)** FLIM_{CA} - Layer 2 **(i)** FLIM_{MA} - Layer 3 **(j)** FLIM_{CA} - Layer 4

(k) Input image **(l)** FLIM - Layer 1 **(m)** FLIM - Layer 2 **(n)** FLIM - Layer 3 **(o)** FLIM - Layer 4

(p) Merged Saliency **(q)** FLIM_{CA} - Layer 1 **(r)** FLIM_{CA} - Layer 2 **(s)** FLIM_{CA} - Layer 3 **(t)** FLIM_{CA} - Layer 4

Figure 14. Multi-level CA on parasite data. The first two rows show images with superior performance, while the bottom rows show problematic ones. The blue line represents ground-truth mask edges.



(a) Input image **(b)** FLIM - Layer 1 **(c)** FLIM - Layer 2 **(d)** FLIM - Layer 3

(e) Merged Saliency **(f)** FLIM_{CA} - Layer 1 **(g)** FLIM_{CA} - Layer 2 **(h)** FLIM_{CA} - Layer 3

(i) Input image **(j)** FLIM - Layer 1 **(k)** FLIM - Layer 2 **(l)** FLIM - Layer 3

(m) Merged Saliency **(n)** FLIM_{CA} - Layer 1 **(o)** FLIM_{CA} - Layer 2 **(p)** FLIM_{CA} - Layer 3

Figure 15. Multi-level CA on BraTS 2D data. The first two rows show images with superior performance, while the bottom rows show problematic ones. The blue line represents ground-truth mask edges.

the last layer, CA eliminates all false positives. The merged saliency has a single false-positive component, and the only remaining component (which appears on layers 1 to 3) has a lower saliency value and is eliminated in the thresholding step. To support our discussion of the method's robustness against false positives, Figures 14k to 14t show how initialization and evolution eliminate most of the incorrect saliencies. We clearly see that the darker component will be thresholded out on merged saliency. However, it shows a limitation of our method in evolution when lower saliency values are used for initialization or when only a small portion of the parasite egg is salient in FLIM saliencies. The CA cannot evolve correctly to fit most parasites' eggs in such cases. A possible approach to overcome this problem is to extract component statistics (e.g., the mean LAB value of the salient component) and use them to propagate cells' strengths better.

For brain tumor examples (Figures 15a to 15h), we observe that despite false positives, our method correctly fits the whole tumor, eliminating false-positive values. In the brain, false positives consistently have lower values than tumors. Furthermore, in FLIM saliencies for layers 2 and 3, we can spot higher saliency values outside the tumor. Nevertheless, as we initialize background strength on the brain border and evolve it into the brain, we correctly eliminate those regions. The second case shows a more problematic example. If we analyze the input image, we can easily see that the tumor ground truth encompasses both a hyper-intense region and a hypo-intense region. As it has intensity values similar to those of brain tumor tissue, FLIM saliencies also detect the dark region as a tumor, and evolution occurs, leaking into the background. The leakage is more pronounced in deeper layers, as more regions are initialized with higher values, which explains the FLIM_{CA} degradation in the third layer.

Leaking may occur when the CA propagates foreground strength into healthy brain tissue or background regions on parasites due to incorrect initialization. More typically, when using deeper network layers (like layer 3), their saliency maps are blurrier and cover larger areas than those from earlier layers. This saliency causes healthy brain regions (hypo-intense regions in Figure 15i) to be incorrectly initialized as foreground (tumor). During evolution, these misclassified regions act as "seeds" that spread and "conquer" neighboring tissues with similar characteristics, resulting in false positives extending beyond the tumor boundaries. A possible approach for better robustness would be experimenting with different pooling (average or minimum), which could provide a better initialization. It is important to note that the ground-truth tumor masks annotated by specialists also include these hypo-intense regions (blue line on Figure 15i).

From the results discussed, a conclusion arises. With a fraction of parameters, FLIM_{MCA} outperforms deep SOD methods for parasites and reaches competitive performances under a more heterogeneous problem (BraTS 2D). Nevertheless, there is plenty of room for improvement in every phase of the proposed method, from image selection to decoding, initialization, evolution, and merging. We also note that the implemented solution uses OpenMP, and the entire pipeline (CA initialization, evolution, and merging) takes around 3s. There is significant potential to improve efficiency when working with constrained computing resources, and section 5.4 provides a complexity analysis and discusses a GPU implementation that can leverage off-the-shelf GPUs, even low-cost ones, to speed up the entire pipeline significantly.

5.2 Statistical Analysis

As can be seen, our approach is inherently multi-step: CA evolution builds on FLIM initialization, and multiple evolved CA saliencies are merged into a unified final saliency map. To better assess and quantify our results, we conducted additional analyses using statistical tests to verify the significance of each stage. Our evaluation spans multiple network layers (4 for parasites and 3 for brain tumors) and uses 6 complementary metrics, requiring statistical testing to ensure robust conclusions. We first assess normality using the Shapiro-Wilk test, which yields extremely low p-values (*e.g.,* 1.82×10^{-15}) for all metrics across all layers on both datasets, rejecting the null hypothesis of normally distributed data. Both FLIM and CA evolved from FLIM saliencies — per-layer and merged — exhibit plateau behavior at distribution extremes, with FLIM clustering near zero and evolved saliencies near optimal values, justifying the use of non-parametric tests throughout.

To clarify the naming convention used throughout this analysis: for per-layer comparisons (Wilcoxon tests and Cliff's Delta effect sizes), we measure the improvement that CA evolution brings by comparing, for each layer, the CA-evolved saliency with the FLIM saliency used to initialize it. For benchmark comparisons, FLIM_{M} denotes the merged saliency obtained from combining FLIM saliencies across all layers, while FLIM_{MCA} denotes the merged saliency obtained from combining the CA-evolved saliencies. Both merged variants employ the lightweight merge network described in Sections 3.4 and 5.1.

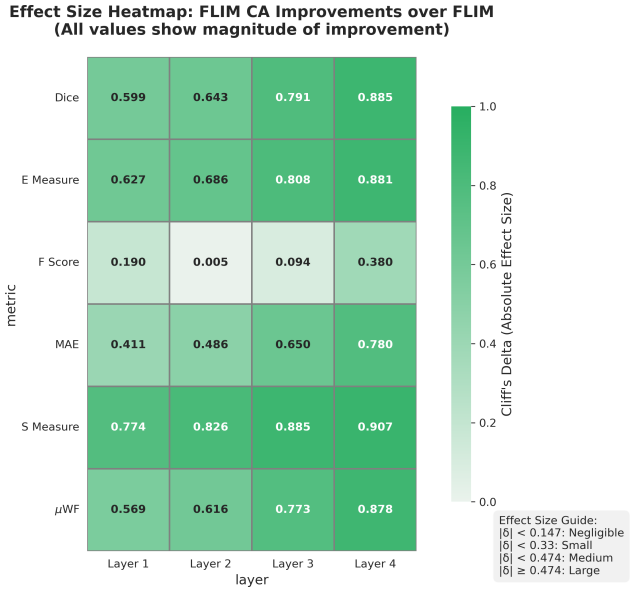
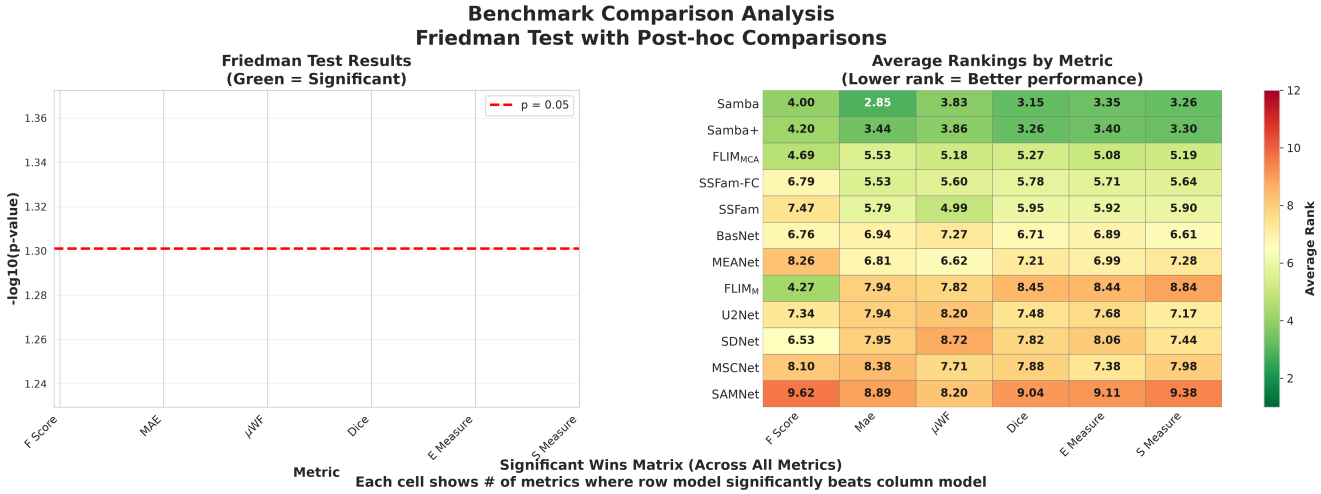


Figure 16. Cliff's Delta effect size heatmap showing evolved CA saliencies improvements over FLIM saliencies across all layers and metrics for the parasites dataset. Darker green indicates larger improvements. Most metrics demonstrate substantial effects ($|\delta| \geq 0.474$).

We then apply the Wilcoxon signed-rank test to compare paired samples (metrics for each test image) between FLIM saliencies and CA saliencies after evolution. For parasites,

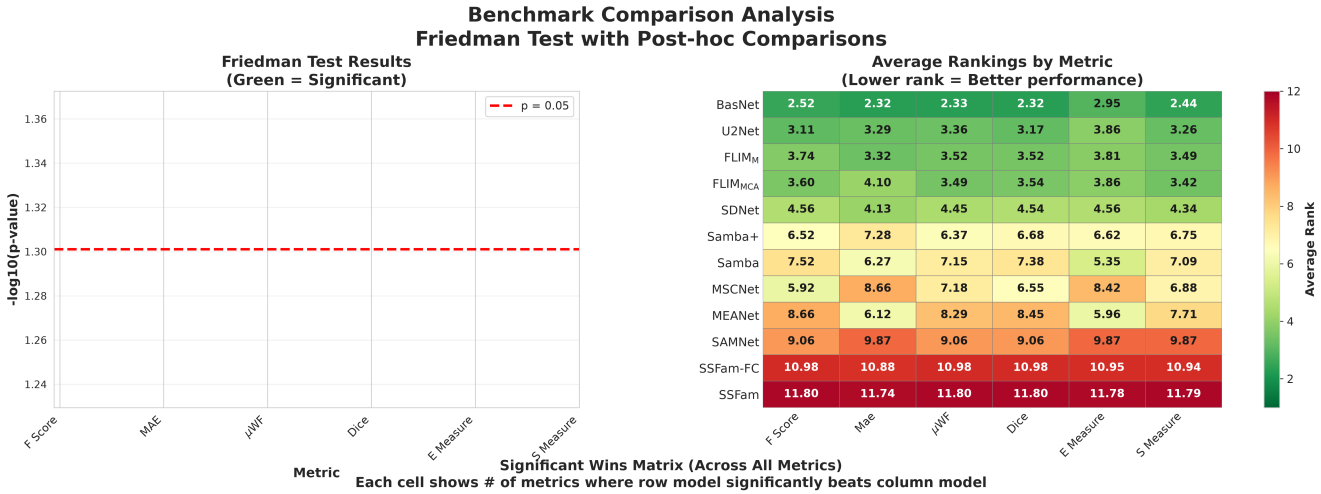


Significant Wins Matrix (Across All Metrics)
Each cell shows # of metrics where row model significantly beats column model

	BasNet	FLIM$_M$	FLIM$_{MCA}$	MEANet	MSCNet	SAMNet	U^2-Net	Samba	Samba+	SDNet	SSFam	SSFam-FC	Total Wins
BasNet	0	4	0	2	4	5	6	0	0	5	0	0	**26**
FLIM$_M$	2	0	0	1	1	4	3	0	0	2	1	1	**15**
FLIM$_{MCA}$	6	5	0	6	6	6	6	0	0	6	3	2	**46**
MEANet	1	5	0	0	4	6	3	0	0	4	0	0	**23**
MSCNet	0	3	0	0	0	5	2	0	0	2	0	0	**12**
SAMNet	1	0	0	0	0	0	1	0	0	1	0	0	**3**
U^2-Net	0	1	0	1	1	4	0	0	0	0	0	0	**7**
Samba	6	5	6	6	6	6	6	0	1	6	6	6	**60**
Samba+	6	5	5	6	6	6	6	0	0	6	6	6	**58**
SDNet	0	2	0	1	1	5	1	0	0	0	0	0	**10**
SSFam	6	5	1	6	6	6	5	0	0	6	0	0	**41**
SSFam-FC	5	5	0	6	6	6	6	0	0	5	1	0	**40**
Total Losses	**33**	**40**	**12**	**35**	**41**	**59**	**45**	**0**	**1**	**43**	**17**	**15**	**341**

Statistical significance determined using Friedman test followed by Nemenyi post-hoc test (α = 0.05)

(a) Parasites dataset. Friedman test results (all $p < 0.001$), average rankings heatmap where Samba and Samba+ achieve the lowest ranks and FLIM$_{MCA}$ ranks third, and significant wins matrix showing Samba with 60, Samba+ with 58, and FLIM$_{MCA}$ with 46 wins.



Significant Wins Matrix (Across All Metrics)
Each cell shows # of metrics where row model significantly beats column model

	BasNet	FLIM$_M$	FLIM$_{MCA}$	MEANet	MSCNet	SAMNet	U^2-Net	Samba	Samba+	SDNet	SSFam	SSFam-FC	Total Wins
BasNet	0	6	6	6	6	6	6	6	6	6	6	6	**66**
FLIM$_M$	0	0	1	6	6	6	0	6	6	6	6	6	**49**
FLIM$_{MCA}$	0	0	0	6	6	6	0	6	6	5	6	6	**47**
MEANet	0	0	0	0	2	6	0	0	2	0	3	3	**16**
MSCNet	0	0	0	4	0	6	0	2	1	0	6	6	**25**
SAMNet	0	0	0	0	0	0	0	0	0	0	0	0	**0**
U^2-Net	0	1	3	6	6	6	0	6	6	6	6	6	**52**
Samba	0	0	0	5	2	6	0	0	2	0	6	6	**27**
Samba+	0	0	0	4	3	6	0	4	0	0	6	6	**29**
SDNet	0	0	0	6	6	6	0	6	6	0	6	6	**42**
SSFam	0	0	0	3	0	0	0	0	0	0	0	0	**3**
SSFam-FC	0	0	0	3	0	0	0	0	0	0	0	0	**3**
Total Losses	**0**	**7**	**10**	**49**	**37**	**54**	**6**	**36**	**35**	**23**	**51**	**51**	**359**

Statistical significance determined using Friedman test followed by Nemenyi post-hoc test (α = 0.05)

(b) BraTS 2D dataset. Friedman test results ($p = 0$ for all metrics), average rankings with BasNet 1st, U^2-Net 2nd, FLIM$_M$ 3rd, and FLIM$_{MCA}$ 4th, and wins matrix showing BasNet with 66 wins, U^2-Net with 52, FLIM$_M$ with 49, and FLIM$_{MCA}$ with 47.

Figure 17. Benchmark comparison using the Friedman test with Nemenyi post-hoc analysis for both datasets. Each panel shows (top-left) Friedman test p-values, (top-right) average rankings heatmap, and (bottom) significant wins matrix.

66 out of 72 tests (91.7%) — four layers, six metrics, three splits — show significant improvement. After applying the Holm-Bonferroni method to control for Type I error across multiple comparisons, 65 tests (≈90.3%) remain significant, demonstrating the robustness of CA evolution. To quantify the magnitude of these improvements, we compute Cliff's Delta (δ) effect sizes (Figure 16). Most metrics demonstrate substantial improvements ($|\delta| \geq 0.474$), with F-Score being the primary exception (small to medium effects). MAE exhibits progressively larger effects in deeper layers. Beyond metric improvements, CA also recovers from failure cases: we observe an 85.2% overall reduction in complete detection failures (zero Dice coefficients), with per-layer reductions ranging from 69.1% to 75.6%.

The merged layer consistently outperforms individual layers for both FLIM_M and FLIM_MCA. FLIM_M achieves 100% significance with an average effect size of 0.665, while FLIM_MCA maintains 96% significance with an average effect size of 0.259. Although FLIM_MCA shows smaller gains — due to already improved per-layer performance — the analysis confirms that merging enhances both approaches.

We then compare FLIM_M and FLIM_MCA against ten benchmarked SOD methods (twelve models total), now including recent 2024–2026 approaches (SDNet, Samba, Samba+, and the two SSFam variants), through the Friedman test and Nemenyi post-hoc analysis (Figure 17a). The left panel shows the Friedman test, which evaluates whether performance distributions across all twelve models differ significantly for each metric; extremely low p-values ($p < 0.001$) confirm substantial differences. The right panel presents average rankings: for each test image, all twelve models are ranked per metric (1 = best, 12 = worst), and these ranks are averaged across all images and splits. Lower average rank indicates more consistent superiority. The recent state-space models Samba and Samba+ achieve the lowest average ranks, with FLIM_MCA ranking third overall and ahead of every pre-2024 baseline. The bottom panel shows the Nemenyi post-hoc significant wins matrix. With twelve models, there are $\binom{12}{2} = 66$ possible pairwise comparisons per metric, yielding $66 \times 6 = 396$ total comparisons across all metrics, of which 341 were statistically significant. Each model is involved in $11 \times 6 = 66$ pairwise comparisons (eleven opponents, six metrics). Of FLIM_MCA's 66 comparisons, 46 result in significant wins, 12 in losses, and the remaining 8 show no significant difference; only Samba (60 wins) and Samba+ (58 wins) surpass it.

For the more challenging BraTS 2D dataset, the Wilcoxon test with Holm-Bonferroni correction yields 40 out of 54 (74.1%) significant comparisons (three layers, six metrics, three splits). While this confirms that FLIM-initialized CA consistently produces improved saliencies, the effect sizes are notably smaller: Cliff's Delta values are predominantly small ($|\delta| < 0.33$) or negligible ($|\delta| < 0.147$), with the strongest effects in layer 1 and some negative values (degradation) in layer 3 (Figure 18). Despite these smaller improvements, CA still recovers from failure cases, with reduction rates ranging from 57.5% to 69.3%. The merged layer significantly improves over the first layer. However, it yields results comparable to, or even lower than, those of deeper layers, indicating that CA error propagation limits improvement when FLIM saliencies are already adequate.

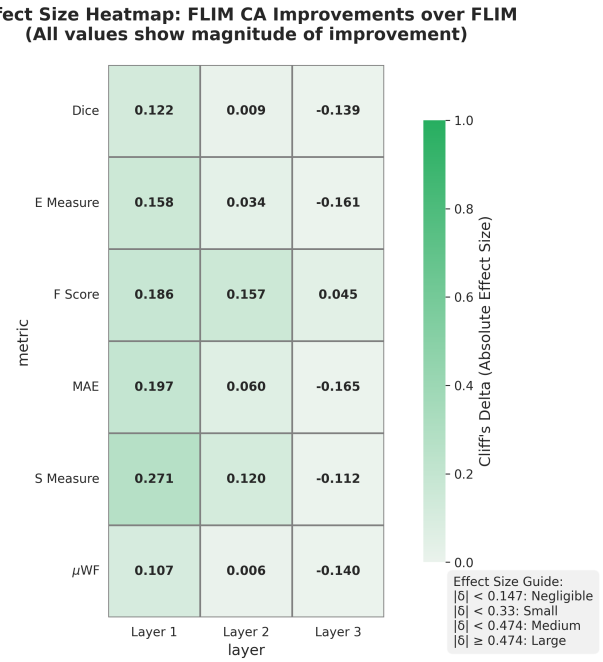


Figure 18. Cliff's Delta effect size heatmap for the BraTS 2D dataset. Most improvements are classified as small ($|\delta| < 0.33$) or negligible, with the strongest effects in layer 1. Layer 3 shows negative values for several metrics, indicating greater difficulty in brain tumor detection.

Friedman test analysis for the BraTS 2D benchmark (Figure 17b) also confirms significant performance differences between models ($p \approx 0$ across all metrics). Rankings show that, on this out-of-domain and heterogeneous data, both the lightweight models and the recent state-space baselines largely fail under few-shot fine-tuning, while BasNet and U^2-Net rank first and second, respectively. Following the same Nemenyi post-hoc procedure, 359 out of 396 total pairwise comparisons are significant. FLIM_M and FLIM_MCA rank third and fourth overall, with 49 and 47 significant wins respectively; of FLIM_MCA's 66 comparisons, 47 result in significant wins, 10 in losses, and 9 show no significant difference. The two merged variants are themselves statistically comparable here — they differ significantly in only one of the six metrics — so this ordering reflects marginal differences against the other models rather than one dominating the other, a behavior we attribute to CA's third-layer degradation on this dataset (Section 5.1). Crucially, both merged variants outperform every 2024–2026 method evaluated here — SDNet (42 wins), Samba+ (29), Samba (27), and the two SSFam variants (3 each) — and trail only the pre-2024 deep models BasNet (66 wins) and U^2-Net (52 wins), while using a fraction of their parameters.

In summary, the statistical evidence strongly supports the effectiveness of our multi-level CA framework. For parasites, 90.3% of comparisons remain significant after multiple comparison correction, with large effect sizes and an 85.2% reduction in complete failures. In the twelve-model benchmark, FLIM_MCA ranks third, behind only the recent state-space models Samba and Samba+, and outperforms every pre-2024 baseline. For BraTS 2D, while effect sizes are smaller (74.1% of significant comparisons, predominantly small effects), the framework proves markedly more robust to domain shift: FLIM_M and FLIM_MCA rank third and fourth and surpass every 2024–2026 method — all of which col-

lapse on this heterogeneous data — trailing only BasNet and U^2-Net while using significantly fewer parameters. Both datasets demonstrate substantial failure recovery capabilities, validating CA's ability to improve initially inadequate FLIM predictions through spatial context evolution.

5.3 Zero Shot Generalization

To further assess the generalization capability of our method, we conduct zero-shot experiments in which the FLIM encoder, originally trained on Schistosoma mansoni parasite eggs, is applied without retraining to morphologically distinct organisms. Specifically, we evaluate two groups of unseen species: helminths from canine fecal samples (Toxocara spp., Trichuris spp., Ancylostoma spp.) and protozoans from human fecal samples (Blastocystis, Entamoeba, Iodamoeba). Figure 19 shows examples of each dataset. These organisms vary substantially in size, shape, and texture, providing a rigorous test of whether FLIM features and CA evolution generalize across morphologically different organisms.

For each species, we evaluate across three independent splits and report the mean and standard deviation of metrics. Each split corresponds to a distinct FLIM encoder designed on a different S. mansoni training set, along with its associated CA parametrization and merge network. All deep learning and lightweight baselines are also evaluated under the same zero-shot protocol: models trained on the corresponding S. mansoni split are applied directly to the unseen species without any retraining. Thus, the reported mean $\pm$ std across splits reflects the variability introduced by different FLIM designs (and different DL trainings).

The helminth datasets comprise 396 images of Toxocara spp., 392 of Trichuris spp., and 394 of Ancylostoma spp. The protozoan datasets comprise 121 images of Blastocystis, 100 of Entamoeba, and 134 of Iodamoeba. As in Section 4.3, we filter connected components by area before computing metrics. Since each species differs in object size, we define per-class filter ranges (in pixels at 400×400 resolution): $[1000, 4000]$ for Toxocara spp. and Trichuris spp., $[100, 4000]$ for Ancylostoma spp., $[100, 800]$ for Blastocystis, $[1000, 30000]$ for Entamoeba, and $[1000, 3000]$ for Iodamoeba. In the benchmark tables, $\text{FLIM}^{\text{L}i}$ denotes the best single-layer FLIM saliency (selected by Dice), FLIM_{M} the merged FLIM saliency, $\text{FLIM}_{\text{CA}}^{\text{L}i}$ the best single-layer CA-evolved saliency, and FLIM_{MCA} the merged CA-evolved saliency. We also choose to add the best-layer saliency because, as we will discuss later, the merge network mostly degrades the performance of the separate saliencies, mainly because it relies on backpropagation during training.

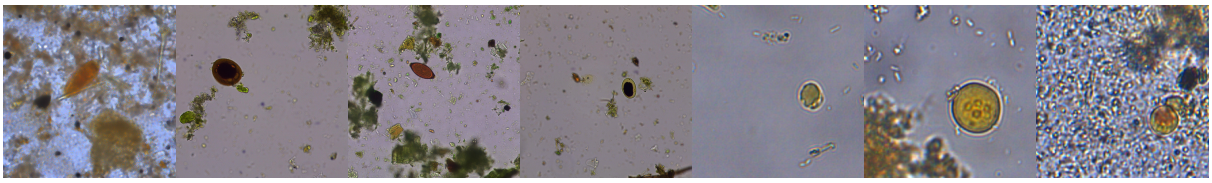
Tables 8-13 present per-layer results comparing FLIM and FLIM_{CA} saliencies, while Tables 14-19 consolidate the benchmark comparison across all evaluated models. We organize the discussion around three key findings: the effect of CA evolution on unseen species, the competitiveness of FLIM-based methods against literature models, and the failure of merge networks.

The CA evolution was parametrized — including initialization thresholds and transition rules — on S. mansoni eggs. When applied to morphologically similar organisms, this parametrization can transfer well. A row-by-row analysis of the per-layer tables reveals the extent of these improvements. For Trichuris spp. (Table 9), CA evolution improves *every* layer across *all* metrics. Focusing on F-Score, Layer 1 jumps from 0.743 to 0.877, Layer 2 from 0.776 to 0.873, Layer 3 from 0.803 to 0.872, and Layer 4 from 0.752 to 0.861. The pattern is consistent across other metrics: Dice improves from a best of 0.782 (Layer 3) to 0.892 ($\text{FLIM}_{\text{CA}}^{\text{L1}}$), s-measure from 0.816 to 0.939 ($\text{FLIM}_{\text{CA}}^{\text{L2}}$), and MAE is reduced by roughly $8\times$, from 0.104 (Layer 3) to 0.012 ($\text{FLIM}_{\text{CA}}^{\text{L2/L3}}$). These gains are not restricted to individual metrics or layers: the CA evolution uniformly refines saliencies regardless of the encoder depth, indicating strong morphological compatibility between Trichuris and the S. mansoni-derived parametrization. Notably, the merged column also improves substantially — from 0.328 (FLIM) to 0.707 (FLIM_{CA}) in F-Score — though it still lags the best individual CA layer (0.877), a pattern discussed later.

Similarly, for Entamoeba (Table 12), CA evolution yields dramatic improvements at every layer and metric. In F-Score, the gains are larger: Layer 1 rises from 0.082 to 0.554, Layer 2 from 0.339 to 0.682, and Layer 3 from 0.416 to 0.700. MAE is reduced from 0.328 (best FLIM layer) to 0.069 (best FLIM_{CA} layer), and e-measure increases from 0.406 to 0.872. These consistent improvements confirm that CA evolution can generalize when the target organism provides sufficient contrast and spatial structure for the automaton rules to exploit.

However, for other species, a row-by-row analysis of Table 8 reveals a more nuanced picture. For Toxocara spp., CA evolution degrades F-Score at every layer (e.g., Layer 1 from 0.908 to 0.834, Layer 2 from 0.899 to 0.858) and increases MAE (Layer 1 from 0.024 to 0.093). Yet, interestingly, s-measure *improves* at all layers: Layer 1 from 0.843 to 0.870, Layer 2 from 0.838 to 0.889, Layer 3 from 0.837 to 0.891. It suggests that while the CA-evolved saliencies lose pixel-level precision, they gain structural coherence. The merged column is also telling: FLIM_{M} collapses to 0.070 in F-Score (compared to individual layers above 0.877), while FLIM_{MCA} reaches 0.610 — still degraded relative to the best CA layer (0.859) but a dramatic improvement over the vanilla merge, indicating that CA evolution partially mitigates the merge network's domain shift.

For Ancylostoma spp. (Table 10), The degradation is more pronounced: CA evolution reduces F-Score at the upper layers (Layer 1 from 0.601 to 0.404, Layer 2 from 0.586 to 0.400) while slightly improving the deeper layers in Dice (Layer 4 from 0.448 to 0.509) and s-measure (Layer 4 from 0.645 to 0.699). Nonetheless, the merged column again shows that FLIM_{MCA} (0.371) outperforms FLIM_{M} (0.069), reinforcing the pattern that CA evolution, even when detrimental at the per-layer level, can rescue the merge output. The most striking case is Iodamoeba (Table 13), where CA evolution catastrophically fails across every metric and layer: F-Score drops from a best of 0.401 (Layer 3) to 0.074 ($\text{FLIM}_{\text{CA}}^{\text{L1}}$), and MAE worsens from 0.359 to 0.494. Here, the merged CA output (0.138) also falls well below FLIM_{M} (0.448), as the CA parametrization is entirely inadequate. We believe the main reasons are: Iodamoeba images exhibit backgrounds with substantially different color distributions, smaller cyst structures, and granular textures that cause the automaton to



(a) S. mansoni **(b)** Toxocara spp. **(c)** Trichuris spp. **(d)** Ancylostoma spp. **(e)** Blastocystis **(f)** Entamoeba **(g)** Iodamoeba

Figure 19. Representative images from each species used in the zero-shot experiments. S. mansoni (a) was used for FLIM encoder design; all other species (b–g) are unseen at design time. Helminth eggs (b–d) share similar preparation and background characteristics with S. mansoni, while protozoans (e–g) exhibit markedly different stained appearance, smaller structures, and distinct morphologies.

Table 8. Zero-shot metrics for toxocara_spp (mean ± std across 3 splits).

Metric	Model	Layer_1	Layer_2	Layer_3	Layer_4	Merged
F-Score	FLIM	0.908 ± 0.006	0.899 ± 0.010	0.890 ± 0.014	0.877 ± 0.035	0.070 ± 0.014
(↑)	$FLIM_{CA}$	0.834 ± 0.037	0.858 ± 0.031	0.859 ± 0.030	0.816 ± 0.013	0.610 ± 0.312
μWF	FLIM	0.911 ± 0.006	0.901 ± 0.009	0.890 ± 0.013	0.874 ± 0.037	0.072 ± 0.018
(↑)	$FLIM_{CA}$	0.817 ± 0.038	0.842 ± 0.034	0.842 ± 0.032	0.798 ± 0.012	0.607 ± 0.310
Dice	FLIM	0.865 ± 0.009	0.861 ± 0.008	0.851 ± 0.015	0.837 ± 0.032	0.055 ± 0.009
(↑)	$FLIM_{CA}$	0.844 ± 0.037	0.868 ± 0.029	0.869 ± 0.029	0.826 ± 0.013	0.579 ± 0.311
e-measure	FLIM	0.943 ± 0.005	0.933 ± 0.011	0.927 ± 0.015	0.926 ± 0.034	0.245 ± 0.143
(↑)	$FLIM_{CA}$	0.894 ± 0.025	0.914 ± 0.012	0.916 ± 0.019	0.877 ± 0.012	0.674 ± 0.301
s-measure	FLIM	0.843 ± 0.009	0.838 ± 0.010	0.837 ± 0.016	0.837 ± 0.029	0.159 ± 0.073
(↑)	$FLIM_{CA}$	0.870 ± 0.029	0.889 ± 0.020	0.891 ± 0.023	0.851 ± 0.013	0.609 ± 0.297
MAE	FLIM	0.024 ± 0.003	0.040 ± 0.014	0.045 ± 0.016	0.045 ± 0.035	0.703 ± 0.155
(↓)	$FLIM_{CA}$	0.093 ± 0.021	0.075 ± 0.009	0.071 ± 0.015	0.108 ± 0.013	0.286 ± 0.297

Table 9. Zero-shot metrics for trichuris_spp (mean ± std across 3 splits).

Metric	Model	Layer_1	Layer_2	Layer_3	Layer_4	Merged
F-Score	FLIM	0.743 ± 0.013	0.776 ± 0.017	0.803 ± 0.019	0.752 ± 0.022	0.328 ± 0.147
(↑)	$FLIM_{CA}$	0.877 ± 0.025	0.873 ± 0.029	0.872 ± 0.021	0.861 ± 0.013	0.707 ± 0.248
μWF	FLIM	0.742 ± 0.012	0.771 ± 0.018	0.798 ± 0.018	0.748 ± 0.022	0.320 ± 0.150
(↑)	$FLIM_{CA}$	0.853 ± 0.029	0.847 ± 0.034	0.848 ± 0.026	0.838 ± 0.017	0.693 ± 0.248
Dice	FLIM	0.727 ± 0.014	0.762 ± 0.018	0.782 ± 0.019	0.721 ± 0.022	0.315 ± 0.135
(↑)	$FLIM_{CA}$	0.892 ± 0.022	0.889 ± 0.025	0.887 ± 0.017	0.871 ± 0.011	0.705 ± 0.250
e-measure	FLIM	0.811 ± 0.008	0.840 ± 0.025	0.867 ± 0.023	0.852 ± 0.027	0.466 ± 0.080
(↑)	$FLIM_{CA}$	0.932 ± 0.017	0.934 ± 0.013	0.936 ± 0.009	0.928 ± 0.008	0.781 ± 0.217
s-measure	FLIM	0.750 ± 0.015	0.789 ± 0.019	0.816 ± 0.020	0.777 ± 0.025	0.401 ± 0.094
(↑)	$FLIM_{CA}$	0.938 ± 0.014	0.939 ± 0.011	0.938 ± 0.008	0.925 ± 0.010	0.755 ± 0.235
MAE	FLIM	0.165 ± 0.009	0.133 ± 0.026	0.104 ± 0.024	0.107 ± 0.030	0.480 ± 0.069
(↓)	$FLIM_{CA}$	0.017 ± 0.007	0.012 ± 0.001	0.012 ± 0.002	0.022 ± 0.016	0.177 ± 0.208

Table 10. Zero-shot metrics for ancylostoma_spp (mean ± std across 3 splits).

Metric	Model	Layer_1	Layer_2	Layer_3	Layer_4	Merged
F-Score	FLIM	0.601 ± 0.005	0.586 ± 0.008	0.550 ± 0.010	0.488 ± 0.011	0.069 ± 0.020
(↑)	$FLIM_{CA}$	0.404 ± 0.039	0.400 ± 0.061	0.439 ± 0.063	0.477 ± 0.033	0.371 ± 0.141
μWF	FLIM	0.572 ± 0.007	0.561 ± 0.001	0.517 ± 0.010	0.447 ± 0.007	0.065 ± 0.021
(↑)	$FLIM_{CA}$	0.338 ± 0.042	0.333 ± 0.062	0.368 ± 0.064	0.401 ± 0.037	0.325 ± 0.123
Dice	FLIM	0.545 ± 0.003	0.527 ± 0.015	0.498 ± 0.013	0.448 ± 0.011	0.063 ± 0.015
(↑)	$FLIM_{CA}$	0.436 ± 0.032	0.433 ± 0.053	0.472 ± 0.055	0.509 ± 0.026	0.380 ± 0.148
e-measure	FLIM	0.836 ± 0.004	0.819 ± 0.020	0.824 ± 0.013	0.823 ± 0.019	0.613 ± 0.036
(↑)	$FLIM_{CA}$	0.646 ± 0.032	0.651 ± 0.052	0.690 ± 0.054	0.728 ± 0.028	0.683 ± 0.063
s-measure	FLIM	0.684 ± 0.005	0.674 ± 0.015	0.666 ± 0.012	0.645 ± 0.015	0.437 ± 0.031
(↑)	$FLIM_{CA}$	0.656 ± 0.019	0.654 ± 0.032	0.679 ± 0.033	0.699 ± 0.016	0.615 ± 0.079
MAE	FLIM	0.023 ± 0.007	0.024 ± 0.011	0.021 ± 0.008	0.024 ± 0.022	0.134 ± 0.074
(↓)	$FLIM_{CA}$	0.023 ± 0.003	0.022 ± 0.005	0.019 ± 0.005	0.025 ± 0.013	0.046 ± 0.012

propagate foreground states into background regions. Figure 20 illustrates these contrasting behaviors: for Entamoeba,

Table 11. Zero-shot metrics for blastocystis (mean ± std across 3 splits).

Metric	Model	Layer_1	Layer_2	Layer_3	Layer_4	Merged
F-Score	FLIM	0.011 ± 0.007	0.057 ± 0.019	0.059 ± 0.029	0.021 ± 0.009	0.086 ± 0.040
(↑)	FLIM_{CA}	0.043 ± 0.012	0.019 ± 0.011	0.001 ± 0.002	0.000 ± 0.000	0.005 ± 0.008
μWF	FLIM	0.013 ± 0.007	0.061 ± 0.019	0.063 ± 0.031	0.023 ± 0.010	0.080 ± 0.035
(↑)	FLIM_{CA}	0.036 ± 0.011	0.019 ± 0.011	0.001 ± 0.002	0.000 ± 0.000	0.006 ± 0.008
Dice	FLIM	0.006 ± 0.004	0.036 ± 0.014	0.036 ± 0.018	0.011 ± 0.005	0.066 ± 0.036
(↑)	FLIM_{CA}	0.037 ± 0.009	0.015 ± 0.009	0.001 ± 0.001	0.000 ± 0.000	0.004 ± 0.006
e-measure	FLIM	0.182 ± 0.058	0.324 ± 0.015	0.265 ± 0.036	0.129 ± 0.043	0.406 ± 0.061
(↑)	FLIM_{CA}	0.599 ± 0.010	0.371 ± 0.039	0.064 ± 0.009	0.021 ± 0.009	0.096 ± 0.057
s-measure	FLIM	0.224 ± 0.064	0.337 ± 0.013	0.281 ± 0.030	0.126 ± 0.043	0.358 ± 0.047
(↑)	FLIM_{CA}	0.418 ± 0.010	0.299 ± 0.019	0.061 ± 0.008	0.020 ± 0.007	0.085 ± 0.051
MAE	FLIM	0.547 ± 0.130	0.330 ± 0.027	0.441 ± 0.055	0.749 ± 0.086	0.309 ± 0.078
(↓)	FLIM_{CA}	0.148 ± 0.015	0.396 ± 0.033	0.876 ± 0.017	0.959 ± 0.013	0.830 ± 0.100

Table 12. Zero-shot metrics for entamoeba (mean ± std across 3 splits).

Metric	Model	Layer_1	Layer_2	Layer_3	Layer_4	Merged
F-Score	FLIM	0.082 ± 0.023	0.339 ± 0.053	0.416 ± 0.050	0.307 ± 0.038	0.514 ± 0.069
(↑)	FLIM_{CA}	0.554 ± 0.039	0.682 ± 0.066	0.700 ± 0.058	0.620 ± 0.048	0.623 ± 0.057
μWF	FLIM	0.089 ± 0.025	0.349 ± 0.054	0.426 ± 0.051	0.317 ± 0.039	0.524 ± 0.071
(↑)	FLIM_{CA}	0.517 ± 0.037	0.679 ± 0.067	0.705 ± 0.058	0.624 ± 0.048	0.631 ± 0.058
Dice	FLIM	0.051 ± 0.015	0.237 ± 0.039	0.294 ± 0.036	0.201 ± 0.027	0.397 ± 0.058
(↑)	FLIM_{CA}	0.524 ± 0.042	0.609 ± 0.062	0.619 ± 0.056	0.538 ± 0.046	0.532 ± 0.051
e-measure	FLIM	0.094 ± 0.027	0.340 ± 0.046	0.406 ± 0.047	0.300 ± 0.036	0.513 ± 0.064
(↑)	FLIM_{CA}	0.872 ± 0.017	0.813 ± 0.039	0.765 ± 0.057	0.654 ± 0.059	0.661 ± 0.068
s-measure	FLIM	0.103 ± 0.028	0.332 ± 0.041	0.397 ± 0.047	0.326 ± 0.032	0.494 ± 0.060
(↑)	FLIM_{CA}	0.677 ± 0.025	0.720 ± 0.040	0.693 ± 0.051	0.589 ± 0.050	0.604 ± 0.061
MAE	FLIM	0.802 ± 0.053	0.414 ± 0.065	0.328 ± 0.076	0.425 ± 0.054	0.275 ± 0.076
(↓)	FLIM_{CA}	0.069 ± 0.004	0.078 ± 0.017	0.145 ± 0.049	0.262 ± 0.058	0.223 ± 0.079

Table 13. Zero-shot metrics for iodameba (mean ± std across 3 splits).

Metric	Model	Layer_1	Layer_2	Layer_3	Layer_4	Merged
F-Score	FLIM	0.060 ± 0.039	0.273 ± 0.023	0.401 ± 0.051	0.128 ± 0.047	0.448 ± 0.029
(↑)	FLIM_{CA}	0.074 ± 0.025	0.071 ± 0.036	0.069 ± 0.014	0.058 ± 0.006	0.138 ± 0.047
μWF	FLIM	0.065 ± 0.042	0.282 ± 0.024	0.412 ± 0.052	0.132 ± 0.049	0.454 ± 0.034
(↑)	FLIM_{CA}	0.062 ± 0.020	0.071 ± 0.037	0.069 ± 0.014	0.057 ± 0.006	0.138 ± 0.048
Dice	FLIM	0.041 ± 0.027	0.200 ± 0.019	0.289 ± 0.040	0.087 ± 0.032	0.361 ± 0.017
(↑)	FLIM_{CA}	0.072 ± 0.025	0.062 ± 0.031	0.059 ± 0.012	0.048 ± 0.005	0.119 ± 0.043
e-measure	FLIM	0.064 ± 0.035	0.280 ± 0.019	0.403 ± 0.051	0.153 ± 0.046	0.506 ± 0.012
(↑)	FLIM_{CA}	0.411 ± 0.058	0.182 ± 0.038	0.126 ± 0.026	0.096 ± 0.008	0.181 ± 0.057
s-measure	FLIM	0.063 ± 0.035	0.255 ± 0.014	0.382 ± 0.043	0.153 ± 0.050	0.451 ± 0.015
(↑)	FLIM_{CA}	0.267 ± 0.036	0.136 ± 0.032	0.101 ± 0.021	0.071 ± 0.005	0.153 ± 0.051
MAE	FLIM	0.887 ± 0.059	0.565 ± 0.021	0.359 ± 0.067	0.730 ± 0.083	0.341 ± 0.028
(↓)	FLIM_{CA}	0.494 ± 0.060	0.771 ± 0.039	0.840 ± 0.031	0.879 ± 0.009	0.783 ± 0.066

CA evolution expands an undersized FLIM detection to accurately delineate the cyst, while for Iodamoeba, CA propagates foreground across a noisy background region and also loses regions correctly initialized.

Remarkably, even when CA evolution degrades results relative to FLIM alone, the degraded FLIM_{CA} and FLIM_{MCA} saliencies still outperform all literature models in most cases. Analysing F-Score, on Toxocara spp. (Table 14), FLIM_{MCA} at 0.610 ± 0.312 — despite being degraded by both CA and merging — still exceeds BasNet (0.419), U^2-Net (0.181) by $3.4\times$, and all lightweight models. On Ancylostoma spp. (Table 16), FLIM_{MCA} at 0.371 ± 0.141 surpasses BasNet (0.050), and outperforms U^2-Net (0.011) in F-Score, while maintaining a substantially lower MAE (0.046 vs. 0.367 for U^2-Net). Even on Iodamoeba (Table 19), where CA catastrophically fails, FLIM_{MCA} at 0.138 ± 0.047 still exceeds BasNet (0.039) and U^2-Net (0.001), and loses for MEANet (0.271) and MSCNet(0.358); it is important to notice FLIM generalization here, where its merged and best layer saliencies yet score in first accross all metrics. For Blastocystis, all method assessments are near zero, with FLIM_{MCA} (0.005) falling below BasNet (0.009) and MEANet (0.052), indicat-

Table 14. Zero-shot benchmark for toxocara_spp (mean ± std across 3 splits).

Model	F-Score (↑)	μWF (↑)	Dice (↑)	e-measure (↑)	s-measure (↑)	MAE (↓)
SAMNet	0.007 ± 0.010	0.007 ± 0.009	0.006 ± 0.009	0.064 ± 0.067	0.037 ± 0.040	0.927 ± 0.075
MEANet	0.005 ± 0.007	0.005 ± 0.007	0.004 ± 0.005	0.124 ± 0.091	0.071 ± 0.047	0.854 ± 0.102
MSCNet	0.005 ± 0.004	0.005 ± 0.004	0.004 ± 0.003	0.101 ± 0.057	0.060 ± 0.033	0.878 ± 0.069
SDNet	0.144 ± 0.031	0.135 ± 0.029	0.152 ± 0.032	0.254 ± 0.051	0.209 ± 0.035	0.726 ± 0.055
FLIM^{L1}	0.908 ± 0.006	0.911 ± 0.006	0.865 ± 0.009	0.943 ± 0.005	0.843 ± 0.009	0.024 ± 0.003
FLIM_{M}	0.070 ± 0.014	0.072 ± 0.018	0.055 ± 0.009	0.245 ± 0.143	0.159 ± 0.073	0.703 ± 0.155
$\text{FLIM}_{\text{CA}}^{\text{L3}}$	0.859 ± 0.030	0.842 ± 0.032	0.869 ± 0.029	0.916 ± 0.019	0.891 ± 0.023	0.071 ± 0.015
FLIM_{MCA}	0.610 ± 0.312	0.607 ± 0.310	0.579 ± 0.311	0.674 ± 0.301	0.609 ± 0.297	0.286 ± 0.297
BasNet	0.419 ± 0.088	0.412 ± 0.091	0.418 ± 0.086	0.494 ± 0.079	0.456 ± 0.081	0.480 ± 0.081
$\text{U}^{\mathbf{2}}$-Net	0.181 ± 0.118	0.163 ± 0.106	0.188 ± 0.123	0.425 ± 0.182	0.323 ± 0.156	0.523 ± 0.197
Samba	0.887 ± 0.009	0.878 ± 0.009	0.900 ± 0.009	0.932 ± 0.010	0.913 ± 0.010	0.064 ± 0.011
Samba+	0.870 ± 0.039	0.859 ± 0.040	0.884 ± 0.036	0.917 ± 0.030	0.898 ± 0.033	0.078 ± 0.029
SSFam	0.790 ± 0.011	0.778 ± 0.012	0.800 ± 0.012	0.843 ± 0.009	0.821 ± 0.009	0.146 ± 0.007
SSFam-FC	0.712 ± 0.119	0.699 ± 0.122	0.723 ± 0.118	0.776 ± 0.103	0.750 ± 0.107	0.210 ± 0.098

Table 15. Zero-shot benchmark for trichuris_spp (mean ± std across 3 splits).

Model	F-Score (↑)	μWF (↑)	Dice (↑)	e-measure (↑)	s-measure (↑)	MAE (↓)
SAMNet	0.139 ± 0.197	0.136 ± 0.192	0.142 ± 0.200	0.189 ± 0.258	0.167 ± 0.231	0.803 ± 0.267
MEANet	0.096 ± 0.105	0.094 ± 0.105	0.093 ± 0.100	0.189 ± 0.101	0.147 ± 0.095	0.791 ± 0.113
MSCNet	0.022 ± 0.016	0.020 ± 0.015	0.023 ± 0.017	0.112 ± 0.046	0.075 ± 0.020	0.870 ± 0.054
SDNet	0.564 ± 0.041	0.509 ± 0.045	0.611 ± 0.039	0.735 ± 0.013	0.730 ± 0.017	0.160 ± 0.014
FLIM^{L3}	0.803 ± 0.019	0.798 ± 0.018	0.782 ± 0.019	0.867 ± 0.023	0.816 ± 0.020	0.104 ± 0.024
FLIM_{M}	0.328 ± 0.147	0.320 ± 0.150	0.315 ± 0.135	0.466 ± 0.080	0.401 ± 0.094	0.480 ± 0.069
$\text{FLIM}_{\text{CA}}^{\text{L1}}$	0.877 ± 0.025	0.853 ± 0.029	0.892 ± 0.022	0.932 ± 0.017	0.938 ± 0.014	0.017 ± 0.007
FLIM_{MCA}	0.707 ± 0.248	0.693 ± 0.248	0.705 ± 0.250	0.781 ± 0.217	0.755 ± 0.235	0.177 ± 0.208
BasNet	0.317 ± 0.050	0.311 ± 0.047	0.317 ± 0.052	0.422 ± 0.080	0.377 ± 0.072	0.554 ± 0.099
$\text{U}^{\mathbf{2}}$-Net	0.002 ± 0.001	0.001 ± 0.000	0.002 ± 0.001	0.109 ± 0.060	0.065 ± 0.034	0.870 ± 0.071
Samba	0.934 ± 0.021	0.928 ± 0.022	0.938 ± 0.019	0.975 ± 0.006	0.960 ± 0.012	0.015 ± 0.005
Samba+	0.911 ± 0.020	0.896 ± 0.024	0.926 ± 0.013	0.963 ± 0.005	0.955 ± 0.006	0.015 ± 0.002
SSFam	0.905 ± 0.010	0.887 ± 0.014	0.918 ± 0.009	0.950 ± 0.008	0.948 ± 0.005	0.021 ± 0.002
SSFam-FC	0.880 ± 0.037	0.858 ± 0.043	0.897 ± 0.034	0.931 ± 0.032	0.934 ± 0.030	0.031 ± 0.027

Table 16. Zero-shot benchmark for ancylostoma_spp (mean ± std across 3 splits).

Model	F-Score (↑)	μWF (↑)	Dice (↑)	e-measure (↑)	s-measure (↑)	MAE (↓)
SAMNet	0.002 ± 0.002	0.001 ± 0.002	0.001 ± 0.002	0.092 ± 0.064	0.059 ± 0.037	0.879 ± 0.075
MEANet	0.008 ± 0.006	0.007 ± 0.005	0.007 ± 0.006	0.397 ± 0.130	0.277 ± 0.095	0.433 ± 0.196
MSCNet	0.001 ± 0.001	0.001 ± 0.001	0.001 ± 0.001	0.333 ± 0.082	0.218 ± 0.060	0.554 ± 0.123
SDNet	0.346 ± 0.014	0.288 ± 0.019	0.387 ± 0.010	0.610 ± 0.010	0.586 ± 0.010	0.192 ± 0.018
FLIM^{L1}	0.601 ± 0.005	0.572 ± 0.007	0.545 ± 0.003	0.836 ± 0.004	0.684 ± 0.005	0.023 ± 0.007
FLIM_{M}	0.069 ± 0.020	0.065 ± 0.021	0.063 ± 0.015	0.613 ± 0.036	0.437 ± 0.031	0.134 ± 0.074
$\text{FLIM}_{\text{CA}}^{\text{L4}}$	0.477 ± 0.033	0.401 ± 0.037	0.509 ± 0.026	0.728 ± 0.028	0.699 ± 0.016	0.025 ± 0.013
FLIM_{MCA}	0.371 ± 0.141	0.325 ± 0.123	0.380 ± 0.148	0.683 ± 0.063	0.615 ± 0.079	0.046 ± 0.012
BasNet	0.050 ± 0.030	0.049 ± 0.028	0.047 ± 0.029	0.308 ± 0.131	0.243 ± 0.096	0.544 ± 0.176
$\text{U}^{\mathbf{2}}$-Net	0.011 ± 0.004	0.008 ± 0.004	0.012 ± 0.003	0.435 ± 0.114	0.313 ± 0.086	0.367 ± 0.178
Samba	0.744 ± 0.037	0.730 ± 0.040	0.750 ± 0.033	0.889 ± 0.022	0.837 ± 0.025	0.057 ± 0.018
Samba+	0.721 ± 0.023	0.693 ± 0.031	0.744 ± 0.015	0.876 ± 0.013	0.839 ± 0.011	0.052 ± 0.005
SSFam	0.643 ± 0.011	0.601 ± 0.032	0.665 ± 0.011	0.811 ± 0.011	0.789 ± 0.018	0.057 ± 0.031
SSFam-FC	0.634 ± 0.028	0.584 ± 0.046	0.664 ± 0.018	0.808 ± 0.021	0.792 ± 0.013	0.054 ± 0.019

ing that all methods failed to generalize. Those results also show that, for datasets where FLIM fails to learn, it often propagates low-quality or inadequate saliencies during initialization.

These results highlight that while the FLIM encoder generalizes well — detecting salient structures even in unseen species — the CA component is sensitive to its parametrization since the CA evolution was designed for S. Mansoni eggs,

Table 17. Zero-shot benchmark for blastocystis (mean ± std across 3 splits).

Model	F-Score (↑)	μWF (↑)	Dice (↑)	e-measure (↑)	s-measure (↑)	MAE (↓)
SAMNet	0.003 ± 0.002	0.004 ± 0.003	0.002 ± 0.001	0.018 ± 0.014	0.021 ± 0.017	0.957 ± 0.033
MEANet	0.052 ± 0.042	0.052 ± 0.039	0.038 ± 0.034	0.251 ± 0.110	0.232 ± 0.084	0.549 ± 0.151
MSCNet	0.023 ± 0.022	0.025 ± 0.023	0.013 ± 0.013	0.161 ± 0.061	0.154 ± 0.066	0.690 ± 0.131
SDNet	0.028 ± 0.001	0.027 ± 0.003	0.023 ± 0.001	0.280 ± 0.061	0.254 ± 0.047	0.492 ± 0.096
$FLIM^{L2}$	0.057 ± 0.019	0.061 ± 0.019	0.036 ± 0.014	0.324 ± 0.015	0.337 ± 0.013	0.330 ± 0.027
$FLIM_{M}$	0.086 ± 0.040	0.080 ± 0.035	0.066 ± 0.036	0.406 ± 0.061	0.358 ± 0.047	0.309 ± 0.078
$FLIM_{CA}^{L1}$	0.043 ± 0.012	0.036 ± 0.011	0.037 ± 0.009	0.599 ± 0.010	0.418 ± 0.010	0.148 ± 0.015
$FLIM_{MCA}$	0.005 ± 0.008	0.006 ± 0.008	0.004 ± 0.006	0.096 ± 0.057	0.085 ± 0.051	0.830 ± 0.100
BasNet	0.009 ± 0.006	0.011 ± 0.006	0.005 ± 0.004	0.155 ± 0.092	0.165 ± 0.084	0.665 ± 0.170
U^2-Net	0.006 ± 0.002	0.006 ± 0.002	0.005 ± 0.001	0.344 ± 0.110	0.299 ± 0.082	0.391 ± 0.168
Samba	0.015 ± 0.010	0.015 ± 0.010	0.011 ± 0.009	0.197 ± 0.007	0.183 ± 0.009	0.632 ± 0.025
Samba+	0.016 ± 0.006	0.017 ± 0.007	0.012 ± 0.004	0.086 ± 0.008	0.084 ± 0.009	0.835 ± 0.020
SSFam	0.013 ± 0.002	0.015 ± 0.002	0.008 ± 0.001	0.239 ± 0.075	0.221 ± 0.068	0.552 ± 0.139
SSFam-FC	0.014 ± 0.008	0.016 ± 0.010	0.009 ± 0.005	0.278 ± 0.029	0.257 ± 0.030	0.480 ± 0.062

Table 18. Zero-shot benchmark for entamoeba (mean ± std across 3 splits).

Model	F-Score (↑)	μWF (↑)	Dice (↑)	e-measure (↑)	s-measure (↑)	MAE (↓)
SAMNet	0.265 ± 0.214	0.269 ± 0.215	0.215 ± 0.193	0.261 ± 0.216	0.248 ± 0.189	0.657 ± 0.219
MEANet	0.466 ± 0.157	0.471 ± 0.157	0.383 ± 0.139	0.470 ± 0.157	0.447 ± 0.146	0.401 ± 0.170
MSCNet	0.381 ± 0.107	0.384 ± 0.107	0.306 ± 0.089	0.390 ± 0.113	0.357 ± 0.104	0.482 ± 0.140
SDNet	0.623 ± 0.048	0.556 ± 0.048	0.663 ± 0.047	0.784 ± 0.038	0.731 ± 0.038	0.144 ± 0.030
$FLIM^{L3}$	0.416 ± 0.050	0.426 ± 0.051	0.294 ± 0.036	0.406 ± 0.047	0.397 ± 0.047	0.328 ± 0.076
$FLIM_{M}$	0.514 ± 0.069	0.524 ± 0.071	0.397 ± 0.058	0.513 ± 0.064	0.494 ± 0.060	0.275 ± 0.076
$FLIM_{CA}^{L3}$	0.700 ± 0.058	0.705 ± 0.058	0.619 ± 0.056	0.765 ± 0.057	0.693 ± 0.051	0.145 ± 0.049
$FLIM_{MCA}$	0.623 ± 0.057	0.631 ± 0.058	0.532 ± 0.051	0.661 ± 0.068	0.604 ± 0.061	0.223 ± 0.079
BasNet	0.445 ± 0.234	0.423 ± 0.213	0.428 ± 0.250	0.518 ± 0.289	0.479 ± 0.257	0.412 ± 0.273
U^2-Net	0.020 ± 0.021	0.016 ± 0.017	0.020 ± 0.020	0.127 ± 0.087	0.104 ± 0.070	0.799 ± 0.132
Samba	0.810 ± 0.043	0.775 ± 0.052	0.831 ± 0.035	0.919 ± 0.027	0.866 ± 0.028	0.053 ± 0.020
Samba+	0.865 ± 0.020	0.833 ± 0.027	0.886 ± 0.014	0.949 ± 0.013	0.905 ± 0.011	0.036 ± 0.009
SSFam	0.755 ± 0.015	0.712 ± 0.037	0.763 ± 0.000	0.892 ± 0.009	0.816 ± 0.002	0.058 ± 0.001
SSFam-FC	0.723 ± 0.045	0.668 ± 0.058	0.746 ± 0.033	0.859 ± 0.044	0.794 ± 0.032	0.091 ± 0.034

Table 19. Zero-shot benchmark for iodameba (mean ± std across 3 splits).

Model	F-Score (↑)	μWF (↑)	Dice (↑)	e-measure (↑)	s-measure (↑)	MAE (↓)
SAMNet	0.096 ± 0.026	0.098 ± 0.027	0.072 ± 0.022	0.112 ± 0.030	0.104 ± 0.028	0.834 ± 0.038
MEANet	0.271 ± 0.027	0.273 ± 0.029	0.222 ± 0.017	0.346 ± 0.023	0.309 ± 0.005	0.544 ± 0.008
MSCNet	0.358 ± 0.022	0.356 ± 0.020	0.296 ± 0.024	0.433 ± 0.050	0.370 ± 0.034	0.455 ± 0.048
SDNet	0.031 ± 0.012	0.030 ± 0.011	0.024 ± 0.010	0.109 ± 0.047	0.082 ± 0.029	0.849 ± 0.056
$FLIM^{L3}$	0.401 ± 0.051	0.412 ± 0.052	0.289 ± 0.040	0.403 ± 0.051	0.382 ± 0.043	0.359 ± 0.067
$FLIM_{M}$	0.448 ± 0.029	0.454 ± 0.034	0.361 ± 0.017	0.506 ± 0.012	0.451 ± 0.015	0.341 ± 0.028
$FLIM_{CA}^{L1}$	0.074 ± 0.025	0.062 ± 0.020	0.072 ± 0.025	0.411 ± 0.058	0.267 ± 0.036	0.494 ± 0.060
$FLIM_{MCA}$	0.138 ± 0.047	0.138 ± 0.048	0.119 ± 0.043	0.181 ± 0.057	0.153 ± 0.051	0.783 ± 0.066
BasNet	0.039 ± 0.025	0.040 ± 0.026	0.031 ± 0.020	0.103 ± 0.089	0.080 ± 0.065	0.858 ± 0.119
U^2-Net	0.001 ± 0.001	0.001 ± 0.001	0.001 ± 0.001	0.093 ± 0.018	0.063 ± 0.013	0.869 ± 0.027
Samba	0.012 ± 0.005	0.011 ± 0.004	0.011 ± 0.005	0.070 ± 0.029	0.050 ± 0.019	0.909 ± 0.037
Samba+	0.015 ± 0.006	0.014 ± 0.006	0.013 ± 0.005	0.057 ± 0.010	0.040 ± 0.010	0.931 ± 0.014
SSFam	0.099 ± 0.020	0.102 ± 0.022	0.086 ± 0.016	0.151 ± 0.031	0.125 ± 0.030	0.812 ± 0.042
SSFam-FC	0.084 ± 0.029	0.086 ± 0.030	0.073 ± 0.024	0.150 ± 0.023	0.125 ± 0.024	0.802 ± 0.030

achieving better zero-shot performance with $FLIM_{CA}$ would require adapting the CA parameters (initialization thresholds, transition rules, number of iterations) to the target domain, which falls outside the scope of zero-shot evaluation.

A central finding of the zero-shot experiments is the consistent superiority of FLIM-based methods over both lightweight and deep learning SOD models. Across all six species, every lightweight model (SAMNet, MEANet, MSCNet) and

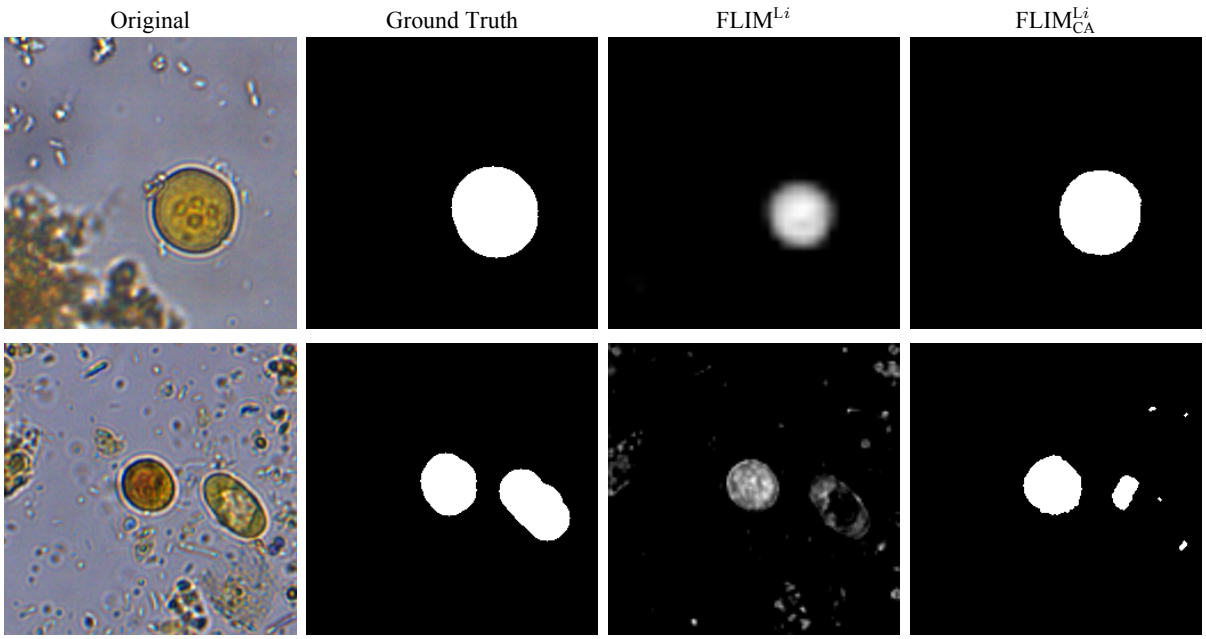


Figure 20. Zero-shot saliency examples illustrating contrasting CA behavior on protozoan species. Each row shows a FLIM saliency and its corresponding FLIM_{CA} evolution at the best-performing layer. Top row: *Entamoeba*, where FLIM^{L3} detects the cyst but undersizes it (F-Score $= 0.755$), and CA evolution expands the saliency to match the ground truth closely (F-Score $= 0.970$). Bottom row: *Iodamoeba*, where FLIM^{L1} correctly localizes two cysts (F-Score $= 0.517$), but CA evolution over-expands to background while also losing detail of the right-most cysts, degrading segmentation quality (F-Score $= 0.320$).

both deep learning architectures (BasNet, U^2-Net) largely fail, with most achieving F-Scores below 0.1 and MAE values above 0.5. This finding concerns the lightweight models and the two pre-2024 deep baselines evaluated in the original benchmark. The recent 2024–2026 baselines added to Tables 14–19 break this pattern on most helminth and protozoan species, as discussed below.

For the helminth datasets, the contrast is stark. On Toxocara spp. (Table 14), all three lightweight models collapse almost entirely: SAMNet, MEANet, and MSCNet all achieve F-Scores below 0.01 with MAE above 0.85, indicating that their saliencies are essentially random noise on this unseen species. The deep learning models are better: BasNet reaches 0.419 ± 0.088 in F-Score and U^2-Net 0.181 ± 0.118, both with MAE above 0.48. In contrast, FLIM^{L1} achieves 0.908 ± 0.006 in F-Score, 0.943 ± 0.005 in e-measure, and a remarkably low MAE of 0.024 ± 0.003 — outperforming the best literature model (BasNet) by a factor of $2.2\times$ in F-Score and $20\times$ in MAE. On Trichuris spp. (Table 15), the gap widens further: $\text{FLIM}^{\text{L1}}_{\text{CA}}$ achieves 0.877 ± 0.025 in F-Score and 0.938 ± 0.014 in s-measure, while BasNet manages only 0.317 ± 0.050 ($2.8\times$ lower), SAMNet 0.139 ± 0.197, MEANet 0.096 ± 0.105, and U^2-Net completely fails at 0.002 ± 0.001. On Ancylostoma spp. (Table 16), FLIM^{L1} at 0.601 ± 0.005 dominates all baselines by an order of magnitude: BasNet achieves only 0.050 ± 0.030, U^2-Net 0.011 ± 0.004, and all lightweight models remain below 0.01. Even in MAE, FLIM^{L1} (0.023) is $16\times$ better than U^2-Net (0.367) and $24\times$ better than BasNet (0.544).

Among the protozoans, where all methods struggle more due to the domain gap, FLIM-based approaches still outperform every literature model. On Entamoeba (Table 18), $\text{FLIM}^{\text{L3}}_{\text{CA}}$ achieves 0.700 ± 0.058 in F-Score, surpassing the best literature model MEANet (0.466 ± 0.157) by 50% and BasNet (0.445 ± 0.234) by 57%. MSCNet (0.381 ± 0.107) and SAMNet (0.265 ± 0.214) trail further behind. In MAE, $\text{FLIM}^{\text{L3}}_{\text{CA}}$ (0.145) is $2.8\times$ better than MEANet (0.401) and $3.3\times$ better than MSCNet (0.482). Notably, even FLIM_{MCA} (0.623 ± 0.057) — a merged and thus degraded output — outperforms all literature models on this species. On Iodamoeba (Table 19), where CA evolution fails, FLIM_{M} at 0.448 ± 0.029 outperforms the best literature model MSCNet (0.358 ± 0.022) by 25%, and MEANet (0.271 ± 0.027) by 65%, while BasNet (0.039 ± 0.025) and U^2-Net (0.001 ± 0.001) essentially produce no useful output. The only dataset where all approaches fail is Blastocystis (Table 17).

These results reveal a fundamental limitation of deep learning SOD models under zero-shot transfer. However, BasNet was pre-trained on large-scale natural image datasets before fine-tuning on S. Mansoni, this pre-training does not prevent catastrophic failure on unseen microscopy species. Their millions of parameters become overfitted to the fine-tuning distribution; not only to the target morphology but also to the specific image characteristics (staining, background texture, contrast) of the training data. The lightweight models exhibit an even more severe collapse: SAMNet, MEANet, and MSCNet consistently produce near-zero F-Scores on the helminth species, suggesting that their compact architectures lack the capacity to retain any transferable representation after fine-tuning. In contrast, the FLIM encoder, trained on only a handful of annotated images and without backpropagation, produces features that generalize better across species precisely because it captures fundamental visual patterns (edges, textures, contrast boundaries) rather than memorizing species-specific appearance. This limitation, however, is not intrinsic to backpropagation itself: as shown next, three recent 2024–2026 baselines fine-tuned under the identical budget avoid it entirely on four of the six species, which redirects the explanation from backpropagation *per se* to the specific pretrained prior each model carries into fine-tuning.

The recent 2024–2026 baselines, fine-tuned under the identical schisto few-shot budget (released pretrained weights, three FLIM design images per split; Section 4.3), substantially revise this picture (Tables 14–19). On Trichuris spp., Ancylostoma spp., and Entamoeba, Samba and Samba+ exceed not only FLIM_{CA} and FLIM_{MCA} but also the best individual FLIM layer: F-Score $0.934/0.911$ versus $\text{FLIM}_{\text{CA}}^{\text{L1}}$'s 0.877 on Trichuris spp., $0.744/0.721$ versus FLIM^{L1}'s 0.601 on Ancylostoma spp., and $0.810/0.865$ (Samba/Samba+) versus $\text{FLIM}_{\text{CA}}^{\text{L3}}$'s 0.700 on Entamoeba. On Toxocara spp. the ordering is closer: Samba (0.887) and Samba+ (0.870) exceed $\text{FLIM}_{\text{CA}}^{\text{L3}}$ (0.859) and FLIM_{MCA} (0.610) but trail the best single FLIM layer, FLIM^{L1} (0.908). SSFam and its click-prompted ablation, SSFam-FC, follow the same ranking one tier below Samba/Samba+ on all four species. On Blastocystis and Iodamoeba, in contrast, every recent baseline collapses to F-Score ≤ 0.10, below FLIM_{M}'s 0.086 and 0.448 — the same two species where the original benchmark already showed near-universal failure. SDNet, the one recent baseline in the same parameter range as FLIM (Section 5.4), does not reproduce this success: on the four species above it matches or trails FLIM_{MCA} (e.g., 0.144 vs. 0.610 on Toxocara spp., 0.623 vs. 0.623 on Entamoeba), consistent with it also ranking below FLIM_{MCA} in the in-domain benchmark (Section 5.2).

The species where the recent baselines succeed are precisely those morphologically closest to *S. mansoni*: an opaque, well-defined helminth egg or an *Entamoeba* cyst on a relatively homogeneous background is, for a model pretrained across large and diverse saliency corpora, an instance of the generic "salient object" concept its pretraining already covers. This is not a data-budget confound — BasNet and U^2-Net are fine-tuned under the exact same protocol (released weights, the same three images per split; Section 4.3) and still fail on these species — so the difference lies in what each pretrained prior already contains, not in how much schisto-specific data it received. A simple ablation confirms the mechanism: a Samba variant trained under the identical protocol and data but *without* loading the released pretrained weights collapses to F-Score 0.000 and Dice 0.000 (mean over 394 non-degenerate test images, Toxocara spp. split 1) on the same zero-shot species, versus $0.887/0.900$ once the released weights are loaded. Same architecture, same three training images, same optimization budget; only the pretrained prior is missing. Normalization compounds the effect: Samba+ replaces BatchNorm with GroupNorm, whose statistics do not depend on batch composition, whereas BasNet and U^2-Net keep BatchNorm throughout, and fine-tuning on only three homogeneous schisto images is known to distort batch statistics in a way a new species' images do not match.

A consistent pattern across all zero-shot experiments is the severe performance degradation introduced by the merge networks (FLIM_{M} and FLIM_{MCA}), clearly visible by comparing the merged column against individual layer values in each per-layer table. On Toxocara spp. (Table 8), FLIM individual layers range from 0.877 to 0.908 in F-Score, yet FLIM_{M} collapses to 0.070 ± 0.014. The same catastrophic drop appears in all metrics: Dice falls from 0.865 to 0.055, s-measure from 0.843 to 0.159, and MAE rises from 0.024 to 0.703. On Ancylostoma spp. (Table 10), FLIM_{M} achieves only 0.069 ± 0.020 in F-Score versus the best individual layer at 0.601. For FLIM_{MCA}, the degradation pattern persists but is sometimes less extreme: on Trichuris spp., FLIM_{MCA} (0.707) retains 81% of the best CA layer's F-Score (0.877).

This failure is expected. The merge network, although consisting of only three convolutional filters, is the sole component of our pipeline that relies on backpropagation. It was trained on S. mansoni images to combine multi-level saliencies into a final output, learning not only how to weight different saliency maps but also implicitly encoding the spatial and intensity characteristics of the training images. When applied to unseen species with different image properties (background textures, object sizes, contrast profiles), these learned weights become inadequate. This limitation parallels the failure mode observed in deep learning models: any component trained via backpropagation on a specific dataset risks overfitting to that distribution. The merge network acts as a bottleneck, collapsing the otherwise transferable FLIM features into a domain-specific combination. This observation reinforces that the generalization strength of our method lies in its backpropagation-free components — the FLIM encoder and the adaptive decoder — while the merge step, despite its simplicity, inherits the domain-dependency characteristic of supervised learning. Nevertheless, with few data, the merged network can be adapted to the new data.

An important practical finding is that FLIM saliencies alone provide a reliable fallback when CA evolution degrades results. On Toxocara spp., Ancylostoma spp., and Iodamoeba, where CA evolution is counterproductive, the best single-layer FLIM saliency consistently outperforms all literature models. Even on Iodamoeba, where CA catastrophically fails (F-Score dropping from 0.401 to 0.074), FLIM^{L3} still surpasses every deep learning and lightweight baseline. This robustness suggests a practical deployment strategy: in zero-shot scenarios, one could rely on FLIM saliencies directly or, at most, apply CA only when the target domain is sufficiently similar to the training domain. Detecting such a domain mismatch automatically or increasing CA robustness and adaptability across different domains remains an open question for future work. This single-layer fallback still holds against the recent 2024–2026 baselines on Toxocara spp. and Iodamoeba (FLIM^{L1}'s 0.908 and FLIM^{L3}'s 0.401 remain the best F-Score on each), but not on Ancylostoma spp.: Samba, Samba+, and SSFam all exceed FLIM^{L1}'s 0.601 there, so a strong pretrained saliency prior can substitute for CA evolution where one is available.

Blastocystis (Table 17) represents the most challenging case, where all methods — including FLIM — essentially fail, with the best result being FLIM_{M} at 0.086 ± 0.040 in F-Score. This near-complete failure across all approaches is explained by examining the data's visual characteristics. Blastocystis cysts are very small, translucent structures with low contrast against the background, producing images that differ fundamentally from S. mansoni eggs in every aspect: size, shape, opacity, and background texture. The FLIM encoder, designed to detect the relatively large, opaque, and well-defined eggs of S. mansoni, cannot produce meaningful activations on these faint structures.

More broadly, the protozoan datasets exhibit a larger domain gap than the helminths. While helminth eggs share

more similarities with S. mansoni, resulting in comparable background textures and contrast profiles, the protozoan images exhibit more distinct color distributions and background characteristics. Additionally, the morphological differences are substantial: S. mansoni eggs are elongated with a characteristic lateral spine, whereas protozoans present round or amorphous cysts. These differences in both the imaging domain and the target morphology compound the challenge for zero-shot transfer.

Despite these challenges, the zero-shot experiments validate the core premise of our approach. The FLIM encoder, designed interactively on a few images of a single species, produces features that transfer to morphologically distinct organisms without any retraining. While the CA and merge components require domain-specific tuning, the encoder itself demonstrates a generalization capacity that fundamentally exceeds that of deep learning models fine-tuned on the same source data with orders of magnitude more parameters. This conclusion holds unconditionally against the lightweight models and the two pre-2024 deep baselines. Measured against the recent 2024–2026 methods, it is more specific: a sufficiently broad pretrained saliency prior, not additional schisto-specific data, lets Samba, Samba+, and SSFam match or exceed FLIM on four of six species, while collapsing below it on the two protozoans with the largest domain gap — a pattern we contextualize next in terms of cost.

Taken together, the zero-shot results argue for reframing this comparison as a cost–generalization trade-off rather than settling it in either method's favor. Samba and Samba+ win on four of the six species here, and already rank first and second in the in-domain parasite benchmark (Section 5.2), ahead of FLIM_{MCA}; the latter's zero-shot advantage is confined to the two protozoan species with the largest domain gap from *S. mansoni*. That advantage, however, is comes with 61.65M/76.65M parameters and a CUDA-only selective-scan kernel that cannot run on CPU at all (Section 5.4), against roughly 400K parameters for the full FLIM+CA pipeline, which runs on a CPU in a few seconds per image. SSFam adds a 317.97M-parameter, 4.5GB SAM ViT-L backbone that is locked to a native 1024×1024 input and cannot even process the parasite images at their original resolution. Whether the recent baselines' pretrained prior is preferable to FLIM's data efficiency therefore depends on which resource is scarce in a given deployment — annotated data and compute, or a GPU and a pretrained checkpoint already close to the target domain — rather than on which method is unconditionally superior.

5.4 Complexity & GPU Optimization

As discussed throughout this work, our method targets real-world scenarios with scarce annotated data and constrained computing resources. To substantiate those claims, we now provide a quantitative analysis of computational complexity and present a GPU-accelerated implementation of the CA evolution — the most expensive component of our pipeline. All experiments in this subsection were conducted on an Intel Core i9-14900KF with an NVIDIA RTX 5080 GPU and 128GB of DDR5 RAM (4200MT/s).

Table 20 compares parameter counts, inference times, and peak GPU memory across all evaluated models on 416×416 inputs. We chose this resolution because some pretrained architectures employ encoder-decoder structures that constrain valid input dimensions to multiples of 32; the value $416 = 32 \times 13$ satisfies this requirement while remaining close to the largest original image dimensions (*i.e.,* parasites with 400×400). All timings are averaged over the 1218 sample data, and we report mean $\pm$ standard deviation to provide a robust estimate of per-image inference cost.

The first observation is that the FLIM encoder alone, with only 588K parameters, is faster than every compared method on both CPU (56.70 ± 1.06ms) and GPU (2.05 ± 0.01ms), even the lightweight models: SAMNet (1.33M), MSCNet (3.26M), and MEANet (3.27M). This efficiency stems from the FLIM encoder's design, which learns filters via clustering rather than backpropagation, resulting in compact architectures in which each filter carries meaningful discriminative information derived from user markers.

This gap widens sharply against the recent 2024–2026 methods, which sit at the opposite end of the cost spectrum. Among them, only SDNet is designed for efficiency: at 745K parameters it is the sole new baseline in the same order of magnitude as FLIM, yet it is still larger than the full FLIM encoder–decoder pipeline and, as Section 5.2 showed, ranks below FLIM_{MCA} on both datasets. The state-space and foundation-model baselines are far heavier: Samba and Samba+ carry 61.65M and 76.65M parameters, while SSFam, built on a SAM ViT-L backbone, reaches 317.97M — roughly $540\times$ the FLIM encoder. Their memory footprints scale accordingly, from 498–575MB for the Samba variants to 4.46GB for SSFam, against 221MB for the FLIM encoder and 31.68MB for the CA. Beyond raw size, these models are also less deployable: the Samba variants rely on a CUDA-only selective-scan kernel and therefore *cannot run on CPU at all*, and SSFam is locked to a fixed 1024×1024 input by SAM's positional grid, so it cannot even process the parasite images at their native resolution. In contrast, the entire FLIM + CA pipeline runs on commodity CPUs. This contrast is the crux of our argument: the recent methods that outperform FLIM_{MCA} on parasites do so only in a high-resource regime, and, as the benchmark showed, that advantage collapses under the domain shift of BraTS 2D — precisely the scarce-data, constrained-hardware scenario our method targets.

The adaptive decoder introduces no learnable parameters, as its channel weights are computed analytically from feature statistics (Equation 2). However, its execution time (106.22 ± 20.39ms on CPU, 4.02 ± 0.02ms on GPU) is non-negligible because it must process all feature channels produced by the encoder. Take, for instance, a three-layer FLIM encoder producing 147 feature channels across its hierarchy: each channel must be evaluated for foreground/background selectivity and combined into a saliency map. This per-channel processing explains why the full encoder-decoder pipeline (153.80 ± 33.04ms on CPU) becomes comparable to lightweight models such as SAMNet (115.32 ± 210.25ms) and MSCNet (135.63 ± 187.38ms), despite using a fraction of their parameters. On the GPU, the combined time (9.95 ± 0.04ms) remains competitive with MEANet (9.61 ± 0.14ms) and is faster than both deep models.

The merge network, with only 68 parameters, adds negli-

Table 20. Parameter count, inference time, and peak GPU memory comparison across models (416×416 input). A dash (—) in the CPU column marks models whose selective-scan kernels run only on GPU (Samba, Samba+). *SSFam is measured at its native 1024×1024 resolution, as the SAM ViT-L backbone uses a fixed positional grid and cannot accept 416×416 inputs.

Model	Parameters	CPU (ms)	GPU (ms)	GPU Mem (MB)
SAMNet	1.33M	115.32 ± 210.25	6.45 ± 0.21	53.26
MSCNet	3.26M	135.63 ± 187.38	5.99 ± 0.06	178.54
MEANet	3.27M	135.91 ± 231.02	9.61 ± 0.14	128.12
SDNet	745K	121.99 ± 83.71	4.53 ± 0.20	202.90
BASNet	87.06M	834.80 ± 26.63	25.38 ± 0.07	913.19
U^2-Net	44.01M	316.26 ± 17.66	15.15 ± 0.05	658.52
Samba	61.65M	—	44.90 ± 1.27	498.21
Samba+	76.65M	—	48.45 ± 0.81	574.63
SSFam*	317.97M	10446.10 ± 130.63	381.82 ± 3.15	4464.12
FLIM Encoder	588K	56.70 ± 1.06	2.05 ± 0.01	221.33
FLIM Decoder	0	106.22 ± 20.39	4.02 ± 0.02	437.13
FLIM (Enc+Dec)	588K	153.80 ± 33.04	9.95 ± 0.04	441.35
Merge Network	68	0.60 ± 0.01	0.05 ± 0.00	25.54

Table 21. CA execution time comparison between CPU and GPU. GPU memory is computed analytically.

Platform	Initialization (s)	Evolution (s)	Probability Map (s)	Get Saliency Map (s)	Total (s)	GPU Mem (MB)
CPU	0.0371 ± 0.0144	2.8743 ± 1.9370	0.0030 ± 0.0029	0.0141 ± 0.0044	2.9284 ± 1.9432	—
GPU	0.0393 ± 0.0122	0.0555 ± 0.0208	0.0001 ± 0.0000	0.0124 ± 0.0046	0.1073 ± 0.0255	31.68

gible overhead (0.60 ± 0.01ms on CPU, 0.05 ± 0.00ms on GPU), consistent with the design philosophy of simplicity under data scarcity we discussed in Section 5.1.

Table 20 also reports peak GPU memory during inference. Deep architectures dominate memory consumption: BASNet requires 913.19MB and U^2-Net 658.52MB, reflecting their large parameter counts and deep feature hierarchies. Among lightweight models, SAMNet is the most frugal at 53.26MB, while MSCNet (178.54MB) and MEANet (128.12MB) occupy intermediate positions. The FLIM encoder and decoder exhibit higher memory usage (221.33MB and 437.13MB, respectively) than their parameter counts would suggest; this overhead arises from intermediate activation maps, particularly the 147 feature channels the decoder must process.

The merge network consumes only 25.54MB. For the CA, we compute GPU memory analytically from the `GPUMemory` struct allocated by our CUDA implementation: each CA instance allocates nine float arrays (original pixel values at $|D| \times c$, foreground and background strengths with their update buffers at $4|D|$, a difference array at $|D|$, and a probability map at $|D|$) and three integer arrays (label, new label, and saliency, each at $|D|$), plus three small adjacency offset arrays. For a 416×416 image with $c = 3$ channels, this totals 7.92MB per CA instance. In the worst case — the parasite dataset with an 4-layer encoder — four CA instances run sequentially, yielding a peak allocation of $4 \times 7.92 = 31.68$MB (Table 21), which is smaller than even the lightest compared model.

The CA evolution constitutes the computational bottleneck. Table 21 breaks down the CA execution time by component. On CPU, the total time per image is 2.93 ± 1.94 seconds, with the evolution step dominating at 2.87 ± 1.94 seconds — roughly 98% of the total. Initialization (0.037s), probability map computation (0.003s), and saliency extraction (0.014s) are comparatively negligible.

Importantly, the CA evolution exhibits **linear complexity** $\mathcal{O}(|D| \cdot |N(p)| \cdot t)$ with respect to the number of pixels $|D|$, where $|N(p)| = 8$ is the Moore neighborhood size and t is the number of iterations until convergence. Each iteration performs a single pass over all pixels, computing the similarity function $g(p, q)$ for each of the $|N(p)|$ neighbors — an $\mathcal{O}(1)$ operation per neighbor — and updating strengths via a simple comparison. The number of iterations t (convergence at L2-norm distance below 10^{-8}) is data-dependent but empirically bounded, typically happening around 200-300 iterations (Section 5.5 reports precise per-dataset measurements and their sensitivity to β). This linear scaling is a fundamental advantage: deep learning convolutions exhibit $\mathcal{O}(|D| \cdot k^2 \cdot c_{\text{in}} \cdot c_{\text{out}})$ complexity per layer, compounded across tens or hundreds of layers, meaning that increasing resolution has a superlinear effect on total inference cost. In contrast, our CA scales proportionally with the number of pixels.

To address the CA's overhead, we developed a GPU implementation using CUDA. The parallelization strategy assigns one thread per pixel, exploiting the *embarrassingly parallel* nature of the evolution: each pixel's state update depends only on its local neighborhood, with no global data dependencies between pixels within an iteration. The implementation uses three CUDA kernels: (i) an evolution kernel that computes similarity-weighted strength propagation for all pixels simultaneously; (ii) a convergence kernel that evaluates per-pixel differences; and (iii) a probability map kernel that generates the final object probability. The GPU implementation achieves a **27× speed-up**, reducing total CA time from 2.93s to 0.107 ± 0.026s (Table 21).

Nevertheless, the first GPU implementation has known limitations and room for further optimization. The primary bottleneck is the convergence check: we could limit transfer from device to host for CPU-side reduction; we could explore evolving foreground and background strengths concurrently on separate CUDA streams, or evolving a single strength map, yielding further improvements. Other opportunities include replacing the fixed-thread block configuration with dynamic, occupancy-based tuning and overlapping host-device transfers with kernel execution via pinned memory.

An important consideration when interpreting Table 20 is that all compared models benefit from years of PyTorch-level optimizations. It means the reported FLIM timings reflect an implementation with substantial room for engineering optimization — the decoder, for instance, could benefit from

batched GPU execution across feature channels. In contrast, the baseline models are closer to their practical performance ceiling. These are engineering improvements that do not alter the method's algorithmic properties but would further reduce measured execution times.

From a deployment perspective, the full FLIM_{MCA} pipeline on GPU completes in approximately 120ms per image ($\sim$10ms for encoder-decoder, $\sim$107ms for CA evolution, $\sim$0.05ms for merging), which is practical for interactive applications. Even on CPU, the combined time of $\sim$3.1s per image remains feasible for batch processing in resource-constrained environments such as parasitology laboratories in developing regions, where the ability to operate without specialized hardware outweighs processing speed. The linear complexity of the CA ensures that scaling to larger images increases cost proportionally, without the superlinear growth that characterizes deeper architectures, yielding results that surpass or are comparable to those of deep-learning models.

Lastly, from the perspective of decoder memory usage, we are currently working on FLIM filter selection mechanisms to select discriminative and descriptive filters, while reducing redundancy; it will enable us to start with fewer feature maps in the first layers, while increasing filter count as the feature domain gets reduced by strided operations.

5.5 Convergence and Sensitivity to β

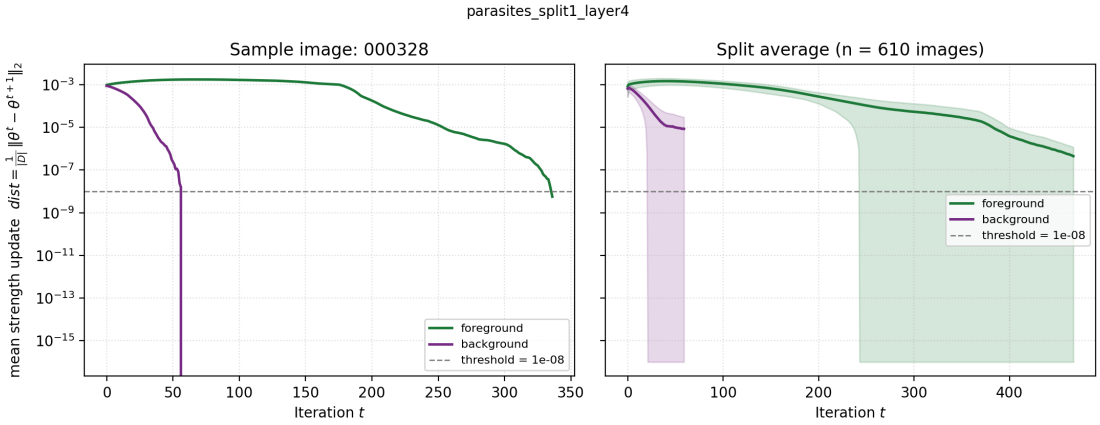
Section 3.3 argued that the CA's evolution is a bounded, monotone process that must reach a fixed point. We now characterize this behavior empirically and assess how stable the final result is to the evolution's only free parameter, β.

Figure 21 plots the mean strength update $dist$ (Algorithm 1) against the iteration t, for a representative test image and averaged over the full test split, for both datasets. In both cases, $dist$ decreases monotonically (background much faster than foreground) until it crosses the 10^{-8} convergence threshold, directly confirming the argument of Section 3.3. Averaged over the β-sensitivity sweep below (Table 22), the foreground CA converges after 301–331 iterations for parasites and 266–268 iterations for BraTS 2D. In contrast, the background CA converges much faster and is essentially insensitive to β (39.8 iterations for parasites, 83.3 for BraTS). The runtime measurements reported in Table 21 are direct empirical timings of our implementation, which reflect this iteration count: the 2.87s CPU / 0.107s GPU evolution times and the resulting 27$\times$ speed-up.

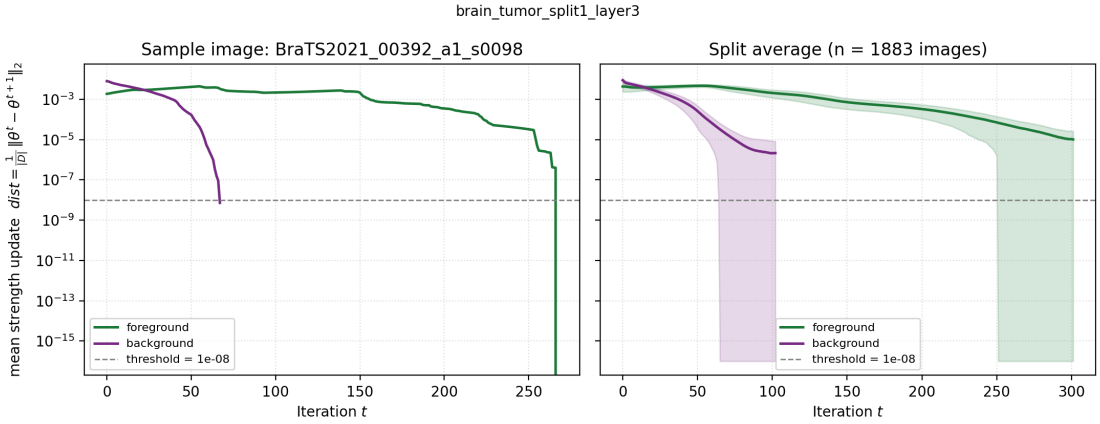
We next sweep $\beta \in [0.2, 1.0]$ on the evolving-object branch of Equation 5, holding the other branch at its unsmoothed value $\beta = 1$, on the same layer/split used in Figure 21 (Table 22). For BraTS, every metric saturates smoothly: the change from $\beta = 0.8$ to $\beta = 1.0$ is at most 0.003 for all five metrics, an order of magnitude below the $\beta = 0.2 \rightarrow 0.4$ step, and the dataset's operating point ($\beta = 0.6$ for the evolving-object case, 1.0 otherwise, Table 3) falls inside this flat region for every metric. For parasites, the picture is more nuanced: Dice and MAE peak at or before $\beta = 0.6$ and change by no more than 0.003 per step from $\beta = 0.6$ onward (Dice even decreases slightly past $\beta = 0.8$), while E-measure peaks at $\beta = 0.6$ and changes by at most 0.004 per step thereafter, and S-measure's step-to-step change likewise collapses to 0.001 by $\beta = 0.8$–1.0 — four of five metrics are effectively flat in the region containing the chosen β. Only the weighted F-measure (μWF) keeps improving at a non-negligible, though decelerating, rate through $\beta = 1.0$ (successive steps of $0.096, 0.040, 0.021, 0.013$). The sweep nevertheless supports $\beta = 0.6$: on parasites it is where Dice (tied with $\beta = 0.8$), E-measure, and MAE all attain their best value in the range, and only μWF and S-measure prefer $\beta = 1.0$, by 0.034 and 0.004 respectively. That disagreement is structural rather than a matter of thousandths. At $\beta = 1$ the two branches of Equation 5 coincide and the object-adaptive smoothing is switched off altogether (see the "inside object" predicate below), so the direction μWF favors is the one that removes the mechanism, while the metrics most directly measuring region overlap and boundary agreement move against it. On BraTS the question does not arise, since $\beta = 0.6$ already lies inside the flat region for every metric. A single value therefore serves both datasets, a constraint we imposed rather than a per-dataset fit. Overall, varying β across its full $5\times$ range moves Dice by only 0.056 (parasites) and 0.015 (BraTS). Table 22 also shows the foreground iteration count decreasing monotonically with β for parasites ($331 \rightarrow 301$) and mildly for BraTS ($268 \rightarrow 266$) — tying this sensitivity analysis directly to the convergence behavior above: a higher β lies within the performance plateau while also converging marginally faster.

Two quantities govern the evolution rule (Equation 5, Algorithm 1): the smoothing factor β and the "inside object" predicate that selects which of its two branches applies — $\|\vec{I}(p) - \vec{I}(q)\|_2 < 0.2$ in normalized LAB space for parasites, $I(p) > I(q)$ for BraTS 2D (Table 3). The predicate acts inside the evolution loop, at every iteration, so it is part of the rule and not an initialization heuristic. We configured the CA empirically on validation data (Section 4.2), and β is what we varied there and selected: it is the rule's only continuous quantity, since the parasites' cutoff picks between two exponents rather than setting one and the BraTS predicate is a sign comparison with no scalar at all. Table 22 repeats that variation on the test split.

Varying β also covers the predicate. Because the two branches differ only in that exponent, changing the predicate can only reassign neighbor pairs between $\beta = 0.6$ and $\beta = 1.0$, and both limits are rows of Table 22: with the LAB cutoff at 0 the predicate never fires and the branches collapse to $e^{-\|\vec{I}(p)-\vec{I}(q)\|_2}$, exactly the $\beta = 1.0$ row — a complete ablation of the predicate — while widening the cutoff applies the smoothed branch to every foreground neighbor; the value we selected (0.2) is the $\beta = 0.6$ row. Across those two rows all five metrics change by at most 0.006 on both datasets, the single exception being μWF on parasites ($0.764 \rightarrow 0.798$), which bounds how much the predicate can move the final saliency. The remaining values sit outside Algorithm 1: the background dilation radius (10) configures the initial state θ^0 (Section 3.2), and the brain histogram threshold ($\tau = \mu - 2.5\sigma$, top-10%) the post-evolution binarization of O_l into $\mathcal{S}'_l$ (Section 4.3); Table 3 documents each alongside its dataset-specific value and selection criterion. The Moore neighborhood and the 10^{-8} convergence threshold are shared by both datasets, the latter being a stopping tolerance rather than a parameter of the rule — by the argument of Sec-



(a) Parasites, layer 4 (the layer selected for the final saliency, Section 5.1).



(b) BraTS 2D, layer 3 (representative; convergence behavior is qualitatively similar across layers, Section 5.1).

Figure 21. Convergence of the mean strength update $dist$ (Algorithm 1) for a representative test image (left panels) and averaged over the full test split (right panels, shaded band = ± 1 std), for the foreground and background CA. Both datasets use the default β configuration (Equation 5) and the 10^{-8} convergence threshold.

Table 22. Sensitivity of the final saliency to the CA's smoothing parameter β (same layer/split as Figure 21). "Iters." is the foreground CA's mean iterations to convergence; background iterations are omitted as they are essentially constant across β (39.8 for parasites, 83.3 for BraTS).

Dataset	β	μWF	Dice	S-measure	E-measure	MAE	Iters.
Parasites	0.2	0.627	0.744	0.804	0.905	0.035	331.3
(Layer 4,	0.4	0.724	0.789	0.845	0.941	0.028	323.4
Split 1)	0.6	0.764	0.800	0.858	0.948	0.027	315.3
	0.8	0.785	0.800	0.861	0.946	0.030	308.7
	1.0	0.798	0.797	0.862	0.942	0.029	301.2
BraTS 2D	0.2	0.699	0.703	0.788	0.858	0.028	267.8
(Layer 3,	0.4	0.709	0.712	0.795	0.871	0.025	267.2
Split 1)	0.6	0.713	0.717	0.797	0.878	0.023	266.5
	0.8	0.716	0.719	0.799	0.881	0.022	266.0
	1.0	0.715	0.718	0.799	0.884	0.021	265.7

tion 3.3 it sets when the evolution halts, not the fixed point it approaches.

6 Conclusion

We expanded previous work on improving the CA initialization problem by exploring knowledge distilled at different hierarchical levels of FLIM networks. CA methods have shown competitive performance, particularly in medical domains where abundant training data and annotations are difficult to obtain. While CA typically requires user interaction for each image, our approach incorporates user knowledge into a FLIM network that automatically initializes the CA. By initializing a separate CA for each convolutional layer of the FLIM Network, we create a multi-level CA that explores a broader initialization space, consistently improving saliencies at each level (up to 43.4%).

The multi-level approach generates multiple complementary saliency maps that form a CA ensemble, leveraging each

encoder level's unique characteristics — from sharp-edged features with false positives to blurrier edges with fewer false positives. Our experiments demonstrate that these saliency maps can be effectively combined using a straightforward merging network (just three convolutional filters) trained on only 3-4 images. The merging network eliminates the need to determine which network layer yields the best saliency while maintaining high performance.

Our method showed consistent cross-domain improvements when benchmarked across both public (BraTS) and private (parasite eggs detection) datasets. A twelve-model statistical benchmark (Friedman test with Nemenyi post-hoc analysis, Section 5.2) confirms that FLIM_{MCA} significantly outperforms every evaluated lightweight model and the pre-2024 deep SOD baselines (BasNet, U^2-Net) on parasite detection, trailing only the recent (2024–2026) state-space models Samba and Samba+, which reach this advantage at a substantially higher parameter and computational cost (Section 5.4). On the more heterogeneous BraTS 2D dataset this cost trade-off reverses: FLIM_{MCA} outperforms every evaluated 2024–2026 baseline, including Samba and Samba+, and trails only the pre-2024 deep baselines BasNet and U^2-Net. Zero-shot experiments on six unseen parasite species (Section 5.3) further show that the recent baselines' advantage on parasites does not generalize uniformly: it holds for species morphologically close to *S. mansoni*, whereas FLIM_{M} remains the only evaluated method that produces usable saliency on the two most dissimilar protozoan species. For brain tumor detection specifically, we outperformed every evaluated lightweight model, requiring only a fraction of the parameters (400k versus 3.6M in lightweight models, and orders of magnitude fewer than the 2024–2026 baselines) and only 3-4 weakly-annotated training images compared to the thousands or millions used by benchmarked approaches.

As a limitation, we see the separate evolution of foreground and background CA as an additional computational overhead that could be optimized. The method also shows some performance variability across datasets, with more substantial results on parasite detection than on more heterogeneous brain tumor data.

For future work, we plan to evolve a single CA instead of separate foreground and background automata, which would reduce computational overhead and simplify rule design. This approach could enable more complex cellular interactions and facilitate 3D multi-label CA development for semantic tumor segmentation - correctly evolving a CA to simultaneously segment the whole tumor, enhanced tumor, and edematous tissue, which could be tried under a clinical environment. The simplicity of our merging network offers substantial room for investigating optimal methods to combine multi-level salience/segmentation maps. Additionally, exploring methods for smoothing border surfaces could further improve edge regions [Sompong and Wongthanavasu, 2017; Vezhnevets and Konouchine, 2005].

From an implementation perspective, our CUDA-based GPU acceleration of the CA evolution achieves a $27\times$ speed-up over the CPU baseline, reducing the full FLIM_{MCA} pipeline to approximately 120ms per image — practical for interactive applications. Even on CPU, the method remains feasible for batch processing in resource-constrained environments such as parasitology laboratories in developing regions. Further GPU optimizations and future improvements remain, such as eliminating device-host memory transfer, concurrent stream execution, and occupancy-based tuning.

Declarations

Authors' Contributions

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

Competing interests

The authors declare that they have no competing interests.

Funding

We thank the Eldorado Institute. Moreover, this work was financially supported by the Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq – (407242/2021-0, 306573/2022-9, 442950/2023-3, and 304711/2023-3), the Fundação de Amparo a Pesquisa do Estado de Minas Gerais – FAPEMIG – (APQ-01079-23 and APQ-05058-23), the Fundação de Amparo a Pesquisa do Estado de São Paulo – FAPESP – (2023/14427-8, 2013/07375-0, and 2025/23020-4) and the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – CAPES – (STIC-AMSUD 88887.878869/2023-00).

Availability of data and materials

The parasite egg dataset is publicly accessible on GitHub at `https://github.com/LIDS-Datasets/schistossoma-eggs`. The code for FLIM encoders and decoders is available at `https://github.com/LIDS-UNICAMP/flim-python-demo`, while the code for initializing CAs, the CAs' evolution processes, and multi-level saliency merging is available at `https://github.com/LIDS-UNICAMP/flim_mca`, along with instructions for execution.

References

Achanta, R., Hemami, S., Estrada, F., and Susstrunk, S. (2009). Frequency-tuned salient region detection. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1597–1604. DOI: 10.1109/CVPR.2009.5206596.

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Süsstrunk, S. (2012). Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282. DOI: 10.1109/TPAMI.2012.120.

Achanta, Radhakrishna, Estrada, Francisco, Wils, Patricia, and Süsstrunk, Sabine (2008). Salient region detection and segmentation. In Gasteratos, Antonios, Vincze, Markus, and Tsotsos, John K., editors, *Computer Vision Systems*,

pages 66–75, Berlin, Heidelberg. Springer Berlin Heidelberg. DOI: 10.1007/978-3-540-79547-6_7.

Baid, U. *et al*. (2021). The RSNA-ASNR-MICCAI BraTS 2021 Benchmark on Brain Tumor Segmentation and Radiogenomic Classification. *arXiv e-prints*, page arXiv:2107.02314. DOI: 10.48550/arXiv.2107.02314.

Borgefors, G. (1984). Distance transformations in arbitrary dimensions. *Computer Vision, Graphics, and Image Processing*, 27(3):321–345. DOI: 10.1016/0734-189X(84)90035-5.

Borji, A., Cheng, M.-M., Hou, Q., Jiang, H., and Li, J. (2019). Salient object detection: A survey. *Computational Visual Media*, 5(2):117–150. DOI: 10.1007/s41095-019-0149-9.

Cerqueira, M. A., Sprenger, F., Teixeira, B. C. A., 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. DOI: 10.1109/EMBC53108.2024.10781962.

Cerqueira, M. A., Sprenger, F., Teixeira, B. C. A., 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. DOI: 10.1109/SIBGRAPI62404.2024.10716300.

Chariton, A., Passalis, N., and Tefas, A. (2022). Bag-of-features-based knowledge distillation for lightweight convolutional neural networks. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 1541–1545. DOI: 10.1109/ICIP46576.2022.9897390.

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. S., and Hu, S.-M. (2015). Global contrast based salient region detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):569–582. DOI: 10.1109/TPAMI.2014.2345401.

Chu, X., Zhang, B., and Xu, R. (2020). Moga: Searching beyond mobilenetv3. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4042–4046. DOI: 10.1109/ICASSP40776.2020.9054428.

Cook, M. *et al*. (2004). Universality in elementary cellular automata. *Complex systems*, 15(1):1–40. DOI: 10.25088/ComplexSystems.15.1.1.

de Melo Joao, L., Proença, L. R., Loiola, S. H. N., Inácio, S. V., dos Santos, B. M., Rosa, S. L., Soares, F. A., Stefano, V. C., Osaku, D., Suzuki, C. T. N., Bresciani, K. D. S., Gomes, J. F., and Falcão, A. X. (2023). Toward automating the diagnosis of gastrointestinal parasites in cats and dogs. *Computers in Biology and Medicine*, page 107203. DOI: 10.1016/j.compbiomed.2023.107203.

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. DOI: 10.1109/SIBGRAPI51738.2020.00024.

de Souza, I. E. and Falcão, A. X. (2022). 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.

Fan, D.-P., Cheng, M.-M., Liu, Y., Li, T., and Borji, A. (2017). Structure-measure: A new way to evaluate foreground maps. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 4558–4567. DOI: 10.1109/ICCV.2017.487.

Fan, D.-P., Gong, C., Cao, Y., Ren, B., Cheng, M.-M., and Borji, A. (2018). Enhanced-alignment measure for binary foreground map evaluation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, page 698–704. AAAI Press. DOI: 10.5555/3304415.3304515.

Felzenszwalb, P. F. and Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. *International journal of computer vision*, 59:167–181. DOI: 10.1023/B:VISI.0000022288.19776.77.

Ghosh, S., Sur, A., and Mitra, P. (2026). Dgssm: Diffusion guided state-space models for multimodal salient object detection. *arXiv preprint arXiv:2604.17585*. DOI: 10.48550/arXiv.2604.17585.

Hamamci, A., Kucuk, N., Karaman, K., Engin, K., and Unal, G. (2012). Tumor-cut: Segmentation of brain tumors on contrast enhanced mr images for radiosurgery applications. *IEEE Transactions on Medical Imaging*, 31(3):790–804. DOI: 10.1109/TMI.2011.2181857.

He, J., Fu, K., Liu, X., and Zhao, Q. (2025). Samba: A unified mamba-based framework for general salient object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. DOI: 10.1109/CVPR52734.2025.02357.

He, J., Fu, K., Liu, X., and Zhao, Q. (2026). Samba+: General and accurate salient object detection via a more unified mamba-based framework. *arXiv preprint arXiv:2602.01593*. DOI: 10.48550/arXiv.2602.01593.

Hernández, G. and Herrmann, H. J. (1996). Cellular automata for elementary image enhancement. *Graphical Models and Image Processing*, 58(1):82–89. DOI: 10.1006/gmip.1996.0006.

Huang, H., Liu, P., Wang, Y., Zhou, T., Qu, B., Tao, A., and Zhang, H. (2023). Multi-feature aggregation network for salient object detection. *Signal, Image and Video Processing*, 17(4):1043–1051. DOI: 10.1007/s11760-022-02310-3.

Huang, M., Liu, Y., Zhao, L., and Wang, G. (2024). A lightweight deep neural network model and its applications based on channel pruning and group vector quantization. *Neural Computing and Applications*, 36(10):5333–5346. DOI: 10.1007/s00521-023-09332-z.

Itti, L., Koch, C., and Niebur, E. (1998). 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.

Joao, L., Cerqueira, M., Benato, B., and Falcao., A. (2024). Understanding marker-based normalization for flim networks. In *Proceedings of the 19th International Joint Conference on Computer Vision, Imaging and Com-*

puter Graphics Theory and Applications - Volume 2: VISAPP, pages 612–623. INSTICC, SciTePress. DOI: 10.5220/0012385900003660.

Joao, L. d. M., Santos, B. M. d., Guimaraes, S. J. F., Gomes, J. F., Kijak, E., Falcao, 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.

Kauffmann, C. and Piché, N. (2010). Seeded nd medical image segmentation by cellular automaton on gpu. *International Journal of Computer Assisted Radiology and Surgery*, 5(3):251–262. DOI: 10.1007/s11548-009-0392-0.

Li, G. and Yu, Y. (2015). Visual saliency based on multiscale deep features. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5455–5463. DOI: 10.1109/CVPR.2015.7299184.

Li, Y., Hou, X., Koch, C., Rehg, J. M., and Yuille, A. L. (2014). The secrets of salient object segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 280–287. DOI: 10.1109/CVPR.2014.43.

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. DOI: 10.1109/ICPR56361.2022.9956350.

Liu, F. and Gleicher, M. (2006). Region enhanced scale-invariant saliency detection. In *2006 IEEE International Conference on Multimedia and Expo*, pages 1477–1480. DOI: 10.1109/ICME.2006.262821.

Liu, N. and Han, J. (2016). Dhsnet: Deep hierarchical saliency network for salient object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 678–686. DOI: 10.1109/CVPR.2016.80.

Liu, T., Sun, J., Zheng, N.-N., Tang, X., and Shum, H.-Y. (2007). Learning to detect a salient object. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. DOI: 10.1109/CVPR.2007.383047.

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.

Liu, Z., Deng, S., Wang, X., Wang, L., Fang, X., and Tang, B. (2024). Ssfam: Scribble supervised salient object detection family. *IEEE Transactions on Multimedia (TMM)*. DOI: 10.1109/TMM.2025.3543092.

Ma, Y.-F. and Zhang, H.-J. (2003). Contrast-based image attention analysis by using fuzzy growing. In *Proceedings of the Eleventh ACM International Conference on Multimedia*, MULTIMEDIA '03, page 374–381, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/957013.957094.

Margolin, R., Zelnik-Manor, L., and Tal, A. (2014). How to evaluate foreground maps. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. DOI: 10.1109/CVPR.2014.39.

Marinescu, A.-I., Balint, Z., Diosan, L.-S., and Andreica, A.-M. (2018). Unsupervised and fully autonomous 3d medical image segmentation based on grow cut. In *2018 20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pages 401–408. DOI: 10.1109/SYNASC.2018.00068.

Movahedi, V. and Elder, J. H. (2010). Design and perceptual validation of performance measures for salient object segmentation. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, pages 49–56. DOI: 10.1109/CVPRW.2010.5543739.

Osaku, D., Cuba, C., Suzuki, C., Gomes, J., and Falcão, A. (2020). Automated diagnosis of intestinal parasites: A new hybrid approach and its benefits. *Computers in Biology and Medicine*, 123:103917. DOI: 10.1016/j.compbiomed.2020.103917.

Perazzi, F., Krähenbühl, P., Pritch, Y., and Hornung, A. (2012). Saliency filters: Contrast based filtering for salient region detection. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 733–740. DOI: 10.1109/CVPR.2012.6247743.

Phan, A.-H., Sobolev, K., Sozykin, K., Ermilov, D., Gusak, J., Tichavský, P., Glukhov, V., Oseledets, I., and Cichocki, A. (2020). Stable low-rank tensor decomposition for compression of convolutional neural network. In Vedaldi, A., Bischof, H., Brox, T., and Frahm, J.-M., editors, *Computer Vision – ECCV 2020*, pages 522–539, Cham. Springer International Publishing. DOI: 10.1007/978-3-030-58526-6_31.

Qin, X., Zhang, Z., Huang, C., Dehghan, M., Zaiane, O. R., and Jagersand, M. (2020). U2-net: Going 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 *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7471–7481. DOI: 10.1109/CVPR.2019.00766.

Qin, Y., Feng, M., Lu, H., and Cottrell, G. W. (2018). Hierarchical cellular automata for visual saliency. *International Journal of Computer Vision*, 126(7):751–770. DOI: 10.1007/s11263-017-1062-2.

Qin, Y., Lu, H., Xu, Y., and Wang, H. (2015). Saliency detection via cellular automata. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 110–119. DOI: 10.1109/CVPR.2015.7298606.

Rosin, P. (2006). Training cellular automata for image processing. *IEEE Transactions on Image Processing*, 15(7):2076–2087. DOI: 10.1109/TIP.2006.877040.

Rosin, P. L. (2009). A simple method for detecting salient regions. *Pattern Recognition*, 42(11):2363–2371. DOI: 10.1016/j.patcog.2009.04.021.

Salvagnini, F. C. R., Gomes, J. F., Santos, C. A. N., 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. 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 *IEEE conference on computer vision and pattern recognition*, pages 4510–4520. DOI: 10.1109/CVPR.2018.00474.

Shelhamer, E., Long, J., and Darrell, T. (2017). Fully convolutional networks for semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):640–651. DOI: 10.1109/TPAMI.2016.2572683.

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. DOI: 10.1109/SIBGRAPI62404.2024.10716333.

Sompong, C. and Wongthanavasu, S. (2017). An efficient brain tumor segmentation based on cellular automata and improved tumor-cut algorithm. *Expert Systems with Applications*, 72:231–244. DOI: 10.1016/j.eswa.2016.10.064.

Su, Z., Liu, L., Muller, M., Zhang, J., Wofk, D., and Cheng, M.-M. (2025). Rapid salient object detection with difference convolutional neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*. DOI: 10.1109/TPAMI.2025.3583968.

Suzuki, C. T. N., Gomes, J. F., Falcão, A. X., Papa, J. P., and Hoshino-Shimizu, S. (2013). Automatic segmentation and classification of human intestinal parasites from microscopy images. *IEEE Transactions on Biomedical Engineering*, 60(3):803–812. DOI: 10.1109/TBME.2012.2187204.

Tong, N., Lu, H., Zhang, Y., and Ruan, X. (2015). Salient object detection via global and local cues. *Pattern Recognition*, 48(10):3258–3267. Discriminative Feature Learning from Big Data for Visual Recognition. DOI: 10.1016/j.patcog.2014.12.005.

Vezhnevets, V. and Konouchine, V. (2005). Growcut: Interactive multi-label nd image segmentation by cellular automata. In *proc. of Graphicon*, volume 1, pages 150–156. Citeseer. Available at:`https://www.scienceopen.com/book?vid=20aab770-8991-4e2b-b02c-a364566d51c0`.

von Neumann, J. (1951). The general and logical theory of automata. In *Cerebral mechanisms in behavior*, pages 1–41. DOI: 10.4324/9781315130569-15.

Wang, L., Lu, H., Ruan, X., and Yang, M.-H. (2015). Deep networks for saliency detection via local estimation and global search. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3183–3192. DOI: 10.1109/CVPR.2015.7298938.

Wang, L., Lu, H., Wang, Y., Feng, M., Wang, D., Yin, B., and Ruan, X. (2017). Learning to detect salient objects with image-level supervision. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3796–3805. DOI: 10.1109/CVPR.2017.404.

Wang, M., Konrad, J., Ishwar, P., Jing, K., and Rowley, H. (2011). Image saliency: From intrinsic to extrinsic context. In *CVPR 2011*, pages 417–424. DOI: 10.1109/CVPR.2011.5995743.

Wang, W., Lai, Q., Fu, H., Shen, J., Ling, H., and Yang, R. (2022). Salient object detection in the deep learning era: An in-depth survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(6):3239–3259. DOI: 10.1109/TPAMI.2021.3051099.

Wang, X., Yu, S., Lim, E. G., and Wong, M. D. (2024). Salient object detection: a mini review. *Frontiers in Signal Processing*, 4:1356793. DOI: 10.3389/frsip.2024.1356793.

Yan, Q., Xu, L., Shi, J., and Jia, J. (2013). Hierarchical saliency detection. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1155–1162. DOI: 10.1109/CVPR.2013.153.

Yang, Q., Zheng, J., and Chen, J. (2025). Multilevel diverse feature aggregation network for salient object detection. *Neurocomputing*, 628:129648. DOI: 10.1016/j.neucom.2025.129648.

Zhao, R., Ouyang, W., Li, H., and Wang, X. (2015). Saliency detection by multi-context deep learning. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1265–1274. DOI: 10.1109/CVPR.2015.7298731.

Zhu, G., Wang, L., and Tang, J. (2024). Learning discriminative context for salient object detection. *Engineering Applications of Artificial Intelligence*, 131:107820. DOI: 10.1016/j.engappai.2023.107820.