A Flyweight CNN with Adaptive Decoder for *Schistosoma mansoni*'s Egg Detection

Leonardo de Melo Joao[a,**], Azael de Melo e Sousa[a], Bianca Martins dos Santos[b], Silvio Jamil Ferzoli Guimarães[c], Jancarlo Ferreira Gomes[b], Alexandre Xavier Falcao[a]

[a]*Institute of Computing State University of Campinas Campinas 13083-872 S£o Paulo Brazil*
[b]*School of Medical Sciences State University of Campinas Campinas 13083-872 S£o Paulo Brazil*
[c]*Institute of Computing Sciences Pontifical Catholic University of Minas Gerais Belo Horizonte 30535-901 Minas Gerais Brazil*

Article history:

flyweight-cnn, object-detection, schistosoma-mansoni-detection, adaptive-decoder, flim

ABSTRACT

Schistosomiasis mansoni is an endemic parasitic disease in more than seventy countries, whose diagnosis is commonly performed by visually counting the parasite's eggs in microscopy images of fecal samples. State-of-the-art (SOTA) object detection algorithms are based on heavyweight neural networks, unsuitable for automating the diagnosis in the laboratory routine. We circumvent the problem by presenting a flyweight Convolutional Neural Network (CNN) that weighs thousands of times less than SOTA object detectors. The kernels in our approach are learned layer-by-layer from attention regions indicated by user-drawn scribbles on very few training images. Representative kernels are visually identified and selected to improve performance with reduced computational cost. Another innovation is a single-layer adaptive decoder whose convolutional weights are automatically defined for each image on-the-fly. The experiments show that our CNN can outperform three SOTA baselines according to five measures, being also suitable for CPU execution in the laboratory routine, processing approximately four images a second for each available thread.

2023

1. Introduction

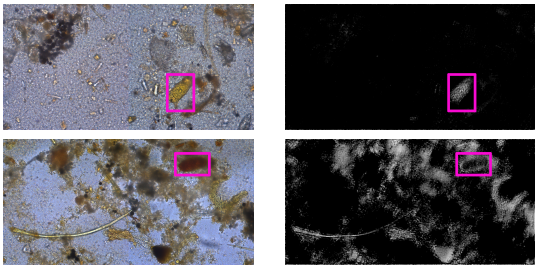
Parasitic infection is a severe problem in developing tropical countries Santos et al. (2019), and schistosomiasis is one of the most frequent, with two hundred million people infected in 2019 World Health Organisation (2019). Schistosomiasis mansoni is the most common schistosomiasis, infecting around seven million people in Brazil alone MDS (2010). This disease is caused by *Schistosoma mansoni*, a parasite trematode effectively endemic in seventy-six countries. Its diagnosis is commonly performed by visually counting the parasite's eggs in microscopy images of fecal samples.

TF-Test Quantified technique Santos et al. (2019) is a recent parasitological procedure developed to facilitate the detection of *S. mansoni*'s eggs. It can improve the diagnostics' sensitivity and egg counting compared to the traditional Kato-Katz exam Katz et al. (1972), as discussed in the literature Santos et al. (2019); Carvalho et al. (2016); Kongs et al. (2001). On the other hand, the amount of fecal impurities is high since it is difficult to eliminate them while preserving the number of eggs in the microscopy slide.

This work focuses on the image analysis of microscopy slides from the *TF-Test Quantified* technique. Most state-of-the-art (SOTA) algorithms for object detection rely on deep neural networks, requiring large annotated sets to train models with millions of parameters (heavyweight) that demand powerful computational resources in GPUs Zaidi et al. (2022). Even few-shot-learning-based models, which require a small number of training images, are heavyweight Huang et al. (2021). Such models are unsuitable for automating the detection of *S. mansoni*'s eggs in laboratory routines. One exam can produce thousands of images with millions of pixels, and the result should take very few minutes. Not requiring expensive GPUs to perform the diagnosis in time would also decrease the monetary cost significantly, enabling the solution for most laboratories.

Constructing lightweight neural networks from weak user supervision and very few training images is possible through a recently proposed method named *Feature Learning from Image Markers* (FLIM) de Souza et al. (2020). The user indicates attention regions by drawing scribbles on training images, and the kernels of a feature extractor (a sequence of convolutional layers) are learned from those markers without backpropagation or weight initialization. Although backpropagation with annotated images might improve performance, a FLIM-based feature extractor is usually adequate as designed for classification and segmentation Sousa et al. (2021); de Souza et al. (2020).

**Corresponding author
e-mail: 1228118@dac.unicamp.br (Leonardo de Melo Joao), a180784@dac.unicamp.br (Azael de Melo e Sousa), b120382@dac.unicamp.br (Bianca Martins dos Santos), sjamil@pucminas.br (Silvio Jamil Ferzoli Guimarães), jgomes@ic.unicamp.br (Jancarlo Ferreira Gomes), afalcao@ic.unicamp.br (Alexandre Xavier Falcao)



(a) (b)

Fig. 1. The activation maps of a kernel in two training images. (a) Original images; (b) Kernel activation maps. In the proposed adaptive decoder, the kernel is considered a foreground dominant kernel for the top image, and background dominant for the bottom one.

The deep-learning paradigm of training a model for a highly complex problem (using datasets such as Imagenet Deng et al. (2009), and MSCoco Lin et al. (2014)) and then specializing it for a simpler one creates oversized models that are hard to simplify. Instead, this work builds on top of FLIM to propose a methodology for creating few-shot and weakly supervised Flyweight (tiny) Convolutional Neural Networks (CNN) for detecting *S. mansoni*'s eggs. The presented Flyweight CNNs combine FLIM encoders with an innovative adaptive and unsupervised single-layer decoder, that uses prior information of the image domain for estimating weights on-the-fly.

For the encoder, we use *FLIM-Builder* [1] for learning kernels layer-by-layer, where kernel outputs are visually inspected by the network designer and manually selected to create a small and sufficient kernel set suitable for the adaptive decoder. The decoder weights the selected kernels as foreground or background dominant according to their mean activation and fixed threshold, making our decoder adapt to each input image (Figure 1). By adapting the feature's roles for each image, the decoder requires less feature diversity to perform well. The decoder performs a point-wise convolution with the adaptive weights, adding activation maps from foreground kernels and subtracting activation maps from background kernels to output a single object saliency map. During training, the network designer can inspect this saliency map to assist decision-making (adding more layers or re-doing the kernel selection). The objects are detected by thresholding the map and filtering components by area. A minimum bounding box containing each component describes its position. It is important to highlight that the unsupervised and adaptive decoder is essential for enabling the kernel selection strategy proposed in this work, as presented in Section 3.

The proposed CNNs are more closely related to saliency estimation than to object detection, but we compare them with SOTA baselines from both areas: U^2Net Qin et al. (2020), which uses nested U-Nets to capture object saliency in multiple scales; SelfReformer Yun and Lin (2022), which uses a transformer backbone and a patch-based decoder with a global-context module to create high-resolution saliency maps; and *DETReg* Bar et al. (2021), a few-short-learning object detector implemented as an extension of the Detection Transformer Carion et al. (2020). Such baselines are heavyweight neural networks, previously trained with thousands of images and fine-tuned in the same training set used to learn our FLIM encoder.

Our approach was trained from scratch (no pre-training) using only five images; it does not require pixel-wise annotation, as it was trained using only user-drawn scribbles; it is flyweight, meaning it is thousands of times smaller than lightweight models, being efficiently executed on CPU (around 200ms per image in a single thread); and it outperformed deep heavyweight methods for saliency estimation and few-shot object detection considering five metrics in our in-house *Schistosoma mansoni*'s egg detection dataset. Two executions in parallel would fulfill the speed requirement for the laboratory routine. Additionally, we show kernel selection's importance and each added layer's impact in our ablation studies, and we discuss future work and improvements in the conclusion.

As contributions, we present: (i) a strategy to build a FLIM encoder layer-by-layer with kernel selection; (ii) the first unsupervised and adaptive decoder for CNNs; and (iii) a novel flyweight CNN for the detection of *S. mansoni*'s eggs that do not require GPU to run and outperforms large deep models. Moreover, it is suitable for CPU execution in the laboratory routine.

2. Related Work

Deep-learning-based object detectors often use a backbone pre-trained on the ImageNet dataset followed by fine-tuning the model with training samples of the given application Liu et al. (2020); Huang et al. (2021). Few-shot-learning-based approaches additionally pre-train the model in large object-detection datasets before fine-tuning it with a few samples for the given application Kang et al. (2019); Wu et al. (2020); Yan et al. (2019). In both cases, the backbone is pre-trained on ImageNet using a loss function suitable for classification. Alternatively, current object detectors create weak labels to adopt a self-supervised approach for object detection Bar et al. (2021); O Pinheiro et al. (2020); Wei et al. (2021); Yang et al. (2021). They execute an unsupervised object detector on ImageNet to create weak labels and then pre-train the model on ImageNet using the weak labels and a detection loss. Among these self-supervised approaches, the Detection with Transformer using Region priors (DETReg) Bar et al. (2021) has achieved the best results on MSCOCO Lin et al. (2014) – the most popular dataset for object detection. DETReg uses Selective Search Uijlings et al. (2013) to create weak labels and adopts a class-agnostic model (i.e., estimates bounding boxes around objects without class knowledge). The authors adapt the model for the expected classes by incorporating a classification head that outputs the probability of a bounding box containing a given object class. Two problems with this method concerning our application are the model's size, which comprises around 40 million parameters and requires expensive GPUs to execute, and the fixed number of bounding boxes per image, which results in many false positives – higher the number of false positives, higher will be the processing time to classify parasites and impurities. Even though the authors state it is not required, non-maximum sup-

[1] `https://github.com/LIDS-UNICAMP/FLIM-Builder`.

pression is a solution for the second problem, but the model's size makes it unsuitable for the laboratory routine and budget.

Class-agnostic approaches can also be implemented by combining a Salient Object Detection (SOD) method with an object classifier. SOD methods highlight objects that stand out given observer-defined salient features. The state-of-the-art approaches are based on deep learning Wang et al. (2021a). They usually adopt a backbone pre-trained on ImageNet, and then the model is trained on a large annotated dataset – DUTS Wang et al. (2017) is the most popular with ten thousand images. Among the recent approaches, U²Net Qin et al. (2020) achieved state-of-the-art performance when considering a reduced number of parameters and can be trained using a small number of training images. The method proposes multiple U-shaped blocks to explore multiple scales. By reducing the image size at every step, the number of parameters can be kept low. We also evaluated the pre-published SelfReformer (SR) Yun and Lin (2022), which has achieved the highest scores in most SOD benchmarks. SelfReformer uses the Pyramid Vision Transformer Wang et al. (2021b) as its backbone for improving long-range information dependency, uses Pixel Shuffle Shi et al. (2016) instead of pooling for keeping fine segmentation details, and frames the saliency detection problem in a patch-wise manner, which relies on a global-context branch to feed information to the local-context patch-based branch. However, both U²Net and SR are still unsuitable for our application because they also require expensive GPUs to execute in a viable time.

Our application requires a tiny model that can execute fast on multiple images without expensive GPUs. One exam may produce thousands of images with millions of pixels for object detection (eggs and impurities), followed by classification. This requirement and the absence of large annotated datasets led us to investigate Feature Learning from Image Markers (FLIM) – a method that learns the kernels of convolutional layers from image regions indicated as relevant by the user de Souza et al. (2020). The user draws markers on a few training images (typically two-three images per class), and the method finds the kernels by grouping patches of the same shape extracted from all marker pixels. Such kernels should activate the relevant regions of each object/class, being a suitable feature extractor for classification De Souza and Falcão (2020); Sousa et al. (2021) and segmentation de Souza et al. (2020) tasks. To the best of our knowledge, no published works have used FLIM for object detection.

3. Flyweight CNN with Adaptive Decoder

This section presents a method to construct a CNN with the designer in the learning loop. The CNN estimates saliency maps for object detection. Particularly, the CNN is flyweight and suitable for *S. mansoni*'s egg detection. Combined with a fast classification network, the solution is suitable for distinguishing between parasite and impurity, counting the number of eggs in a given exam during the laboratory routine. Our method (Figure 2) involves the following steps.

1. The designer annotates a minimal training set with scribbles in relevant parts of the images.
2. The kernels of the current layer are estimated by the FLIM algorithm.
3. The designer manually selects relevant kernels based on the visualization of their activations.
4. A single-layer unsupervised decoder converts the output of the current encoder with the selected kernels into an object saliency map. Visualizing the saliency allows the designer to decide whether to continue the architecture learning.
5. If the designer opts to add a next convolutional layer, steps 2 to 4 are repeated until the designer is satisfied with the architecture of the CNN.

The final CNN consists of a sequence of convolutional layers (a feature extractor as the encoder) followed by a single-layer unsupervised adaptive decoder. The network architecture is also defined by the designer. Our CNN is a flyweight and effective network with only three convolutional layers.

3.1. FLIM - Feature Learning from Image Markers

FLIM extracts image patches from marker pixels with the same shape of the kernels in a given layer. The designer can specify the number of kernels (n_m), with m channels, per marker and the number of kernels per layer (b_l). Let $\mathcal{I}$ be an image, P be the set of all pixels in $\mathcal{I}$, $\mathcal{M} \subset P$ be a marker, $\mathcal{P}(p)$ be a patch of size $j \times j \times m$ centered on p, and $K^*_{\mathcal{M}} = \bigcup_{\forall p \in \mathcal{M}} \mathcal{P}(p)$ be all candidate kernels extracted from marker $\mathcal{M}$.

Then, all patches are clustered using K-means in the $\mathbb{R}^{j \times j \times m}$ feature space to find n_m clusters per marker. The centroid of each cluster is selected as a candidate kernel. Because the number of clusters is bigger than the number b_l of desired kernels for that layer, another K-means is executed to find the b_l kernels that compose the kernel bank of the current layer. Note that this process does not require a full feed through the network, allowing each layer to be built separately. Each convolutional layer contains marker-based normalization (i.e., a batch normalization using only the parameters of the markers), convolution, ReLU activation, and max-pooling Sousa et al. (2021). The above process is repeated using the same markers and output of the previous layer to estimate the kernels of the next layer.

3.2. Single-layer unsupervised and adaptive decoder

Let $\mathbf{I}_k(p)$ be the FLIM encoder's output activation of a kernel k for a pixel p. We normalize these activation values within $[0, 1]$ and define $\mu_k = \frac{1}{|P|} \sum_{\forall p \in P} \mathbf{I}_k(p)$ as the mean activation of a kernel k. Assuming the objects of interest are considerably smaller than background structures, kernels that activate for most pixels are likely activating for background structures. So, a kernel is considered a background kernel if it has high activation values for most pixels; otherwise, it is considered a foreground kernel — *i.e.*, if $\mu_k \geq 0.5 \rightarrow k \in \mathcal{B}$, and conversely, if $\mu_k < 0.5 \rightarrow k \in \mathcal{F}$, where $\mathcal{F}$ and $\mathcal{B}$ the subsets of foreground and background kernels in the last encoder's layer. The decoder combines the activation maps for each subset (foreground and background), creating a foreground activation map $\mathbf{A}_f(p) = \frac{1}{|\mathcal{F}|} \sum_{k \in \mathcal{F}} \mathbf{I}_k(p)$, and a background activation map $\mathbf{A}_b(p) = \frac{1}{|\mathcal{B}|} \sum_{k \in \mathcal{B}} \mathbf{I}_k(p)$. The two activation maps are finally

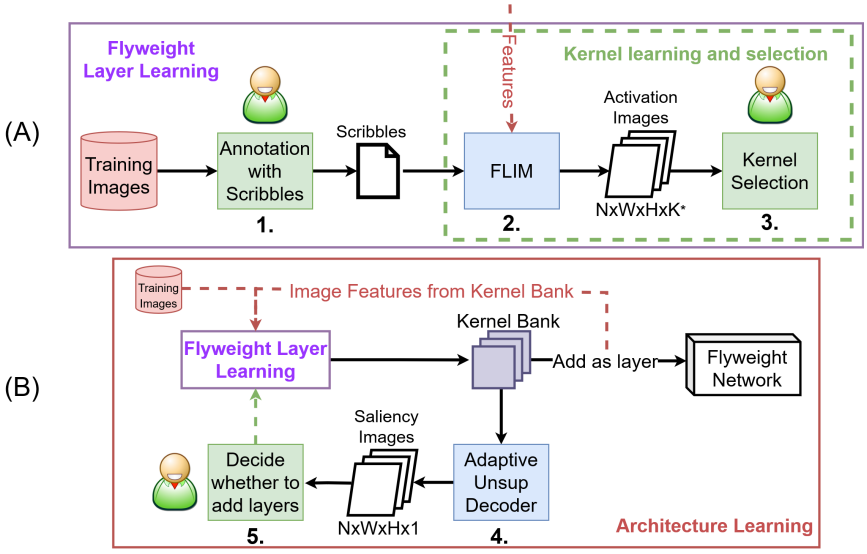


Fig. 2. Diagram of the proposed learning strategy of Flyweight CNNs using FLIM. The network is learned layer by layer following the "Flyweight Layer Learning" steps (A), and more layers are added iteratively until the network designer decides to stop during "Architecture Learning" (B). The Flyweight Network created at the end of (B) is a concatenation of multiple Encoder Layers Learned in (A) followed by the Adaptive and Unsupervised Decoder. In the diagram, blue boxes are automatic processes, and green are interactive ones.

subtracted to filter out false positives, creating a decoder activation score, $\mathbf{D}(p) = \mathbf{A}_f(p) - \mathbf{A}_b(p)$, followed by ReLU activation: *i.e.,* $\mathbf{D}(p) < 0 \rightarrow \mathbf{D}(p) = 0$. This way, impurities with high saliency values in both maps cancel each other. The object saliency map $\mathbf{D}$ is normalized between $[0 - 255]$. The objects of interest are detected by binarizing $\mathbf{D}$, using the Otsu's threshold Otsu (1979), and removing components out of the expected object area. The eggs should satisfy a known area interval, then we discard components out of the interval $[0.5\%, 5\%]$ of $|P|$.

Note that our decoder is equivalent to performing a $1 \times 1 \times B$ convolution (point-wise) with kernel weights $w_k = \{-1, 1\}$, $k \in [1, B]$, followed by *ReLU* activation. These weights are set for each activation map as 1 if derived from a foreground kernel or -1 otherwise, making our decoder adaptive. Depending on the image, a kernel may be considered a foreground or a background kernel (Figure 1). Such classification is only related to the mean activation value and does not imply the kernel has to activate the object of interest.

3.3. *Layer-by-layer encoder construction using kernel selection*

With no backpropagation, FLIM allows the layer-by-layer construction of the encoder. After annotating the training set with scribbles, FLIM uses the image colors as the first layer's features and estimates kernels considering the annotation. Using FLIM-Builder, the network designer can visualize the activation maps of each estimated kernel and select which kernels to keep in that layer.

Two observations can guide Kernel selection: (i) foreground kernels that together activate all eggs in the training images and (ii) pairs of foreground and background kernels that activate the same impurities. Since a foreground kernel may not activate all eggs (Figure 1), multiple kernels are required to satisfy observation (i). Given that our decoder subtracts foreground and background activation maps, observation (ii) implies that both kernel types should be selected to reduce the number of false positives (Figure 3).

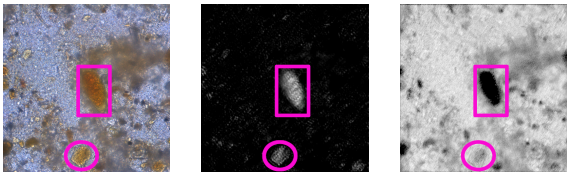


Fig. 3. Activation maps of a pair of foreground and background kernels for an image. (a) Original image; (b) Foreground kernel activation; (c) Background kernel activation. The egg (in the box) is activated in (b) and is not activated in (c). However, the impurity (in the ellipse) is activated in both and can be eliminated by the subtraction (c) from (b).

Understanding the decoder's dynamics, the user can select a combination of foreground kernels that is enough to detect all eggs in the training images and a combination of background kernels that reduce the number of impurities to a minimum while keeping the eggs. To verify the impact of such decisions, the user can analyze the output of the decoder using the selected kernels on a validation set.

After analyzing the decoded result, if the user identifies any object lacking in the training set, another layer can be added, in which the features from the previous layer are used as input, and a new kernel selection must be performed. When satisfied, the training pipeline finishes, the Flyweight model's encoder is the concatenation of all learned layers, and its decoder is the proposed unsupervised adaptive decoder.

4. Experimental Results

4.1. Experimental Setup

Computer Setup: Our approach was trained and executed on an i7-7700 (CPU), while the heavyweight models used an NVIDIA RTX A6000 (GPU).

FLIM parameters: Our network has two convolutional layers with kernel sizes $j = 3$, no strides, number of kernels per marker $n_m = 5$ in all layers, and number of kernels per layer $b_1 = 27$ and $b_2, b_3 = 64$. We adopted atrous convolution with dilation factors $\{3, 5, 7\}$ for the first, second, and third layers, respectively. After ReLU, we used max-pooling with $j = 3$ in the first and second layers and $j = 5$ in the third one. As explained, the decoder has no parameters. The number of kernels selected for the first two layers is 10, and 12 for the third.

DETReg overlapping bounding boxes: DETReg outputs a fixed number (30) of bounding boxes per image. For our application, that number is too high. We executed non-maximum suppression using an IoU threshold of 0.1 due to the low chance of overlapping objects.

Saliency-based bounding boxes: To estimate bounding boxes from object saliency maps, we binarize the decoded image using Otsu's thresholding Otsu (1979) and place minimum bounding boxes around every connected component of the binary image. For our solution, we increased the bounding boxes' width and height by 10%, after observing that the boxes were consistently smaller than the object. The bounding box increase did not improve the other methods.

Size filtering: Because the objects of interest (Schistosoma mansoni eggs) are of a particular size, we implemented a size filter to discard bounding boxes outside the given range. The same filtering was applied to all methods.

Dataset: The dataset used is proprietary and is composed of 631 images containing *S. mansoni*'s eggs. The images are annotated by specialists that provide a pixel-wise segmentation mask. The images often have a cluttered background, and fecal impurities sometimes occlude the eggs (Figure 5). We separated 20% of the dataset for testing and 80% for training and validation. We used 3-fold cross-validation in our experiments, using 1% (5 images) for training and the rest for validation in each split. The test set was only executed once with the best-performing model considering the validation results of each method in all splits. The training images were selected manually to guarantee a good representation of the real problem.

Evalutation metrics: We used multiple object detection metrics to evaluate our solution, which depends on the *Intersection over Union* (IoU), the *Precision*, and the *Recall*. The *IoU* is the ratio between intersection and union of the pixels of the predicted bounding boxes and the expected ones. Formally, let $bb = \{x1, x2, y1, y2\}$ be a bounding box, such that $x2 > x1, y2 > y1$. Let $\mathcal{B}$ be the set of all ground-truth bounding boxes, and bb_p be a predicted box. The $\mathbf{IoU}(bb_p)$ can be defined as:

$$\mathbf{IoU}(bb_p) = \max_{\forall bb_t \in \mathcal{B}} \{\frac{|bb_p \cap bb_t|}{|bb_p \cup bb_t|}\} \quad (1)$$

Bounding boxes with $IoU > \tau$ are considered true positives, and false positives otherwise. Let TP^τ, FP^τ, FN^τ be the total number of true positives, false positives, and false negatives considering τ. The precision can be described as $P^\tau = \frac{TP^\tau}{TP^\tau + FP^\tau}$, and the recall as $R^\tau = \frac{TP^\tau}{TP^\tau + FN^\tau}$.

The metrics we used for evaluation are the Precision-Recall (PR) curve, considering the mean precision and recall on varying IoU thresholds ranging from $\tau \in [0.50, 0.55, ..., 0.95]$, the mean average precision (μAP) as proposed for MS Coco considering the same threshold range, the average precision ($AP^{\tau'}$), and the $F_2^{\tau'}$-score, considering two fixed IoU thresholds $\tau' \in \{0.5, 0.75\}$. The $AP^{\tau'}$ is the Area Under the Curve (AUC) of the PR-curve in a fixed threshold, and the μAP is the mean of all AUCs considering the threshold range. The $F_2\text{-score}^{\tau'} = (1+\beta)^2 \frac{P^{\tau'} \cdot R^{\tau'}}{\beta^2 \cdot P^{\tau'} \cdot R^{\tau'}}$ is the harmonic mean of average precision and recall.

4.2. Ablation on manual kernel selection and number of layers

We executed the experiments with and without kernel selection to compare both approaches' performance and model size differences. As presented in Table 1, kernel selection increased the μAP by 0.28 and $F_2^{0.75}$-score by 0.494. Regarding the number of parameters, the model with kernel selection is more than 20x smaller than the full model (Table 4), but the full model is still tens of times smaller than the lightweight models.

Table 1. Results of the proposed method with all components (FLIM), the same model but using only one ($FLIM_1$) and two ($FLIM_2$) layers, and the model without manual kernel selection ($FLIM*$).

	$F_2^{0.5}$	$AP^{0.5}$	$F_2^{0.75}$	$AP^{0.75}$	μAP
Adaptive-$FLIM_1$	0.778	0.456	0.248	0.070	0.185
Adaptive-$FLIM_2$	0.792	0.506	0.295	0.089	0.199
Adaptive-$FLIM*$	0.559	0.643	0.136	0.151	0.241
Adaptive-FLIM	**0.799**	**0.929**	**0.630**	**0.488**	**0.525**

4.3. Quantitative Results

Table 2 presents the validation-set results of all methods in all three splits. The proposed **Adaptive-FLIM** outperformed all other methods in all metrics apart from *Split 1*, where U²Net had better AP for all threshold values. U²Net had the highest standard deviation across splits, with a $\sigma_{\mu AP} = 0.114$, while our solution had $\sigma_{\mu AP} = 0.014$, meaning our method is more consistent and less dependent on the choice of training images.

The model with the highest μAP in the validation set was selected for each method. Table 3 presents the models' results on the test set (images never before seen). Adaptive-FLIM had the best results in all metrics. DETReg had the lowest values, primarily due to lack of precision (the model output many boxes in wrong objects), as seen in the PR-curves (Figure 4). Self-Reformer had competitive performance when considering a lower IoU threshold ($AP^{0.5}$, $F_2^{0.5}$), but loses performance significantly when the IoU requirement is increased; that decrease is also represented in the PR curves, and, as discussed in Section 4.4, it is mostly due to sub-par segmentation (represented by poorly delineated bounding-boxes). U²Net had the second-best results overall, but more components were missed, which hampered their recall values.

Regarding model size, Table 4 compiles each network's number of parameters. We also present a size comparison to Lightweight Networks architectures to illustrate the difference. The proposed solution is over 17,000 times smaller than DETReg and almost 40,000 times smaller than Self-reformer. Compared to lightweight networks, our approach is 543 times smaller than SqueezeNet and 1,526 times smaller than MobileNet.

Table 2. Validation set results for each split considering our proposed solution (FLIM), U²Net, Self-Reformer and DETReg. The two best results for each metric in each split are highlighted in blue and green, respectively.

Split 1	$F_2^{0.5}$	$AP^{0.5}$	$F_2^{0.75}$	$AP^{0.75}$	μAP
DETReg	0.623	0.329	0.523	0.276	0.208
U²Net	0.748	0.853	0.579	0.661	0.544
Self-Reformer	0.786	0.682	0.161	0.034	0.199
Adaptive-FLIM	**0.793**	**0.676**	**0.629**	**0.536**	**0.457**
Split 2	$F_2^{0.5}$	$AP^{0.5}$	$F_2^{0.75}$	$AP^{0.75}$	μAP
DETReg	0.444	0.151	0.397	0.129	0.102
U²Net	0.628	0.518	0.513	0.423	0.351
Self-Reformer	0.697	0.596	0.095	0.013	0.167
Adaptive-FLIM	**0.821**	**0.688**	**0.627**	**0.525**	**0.438**
Split 3	$F_2^{0.5}$	$AP^{0.5}$	$F_2^{0.75}$	$AP^{0.75}$	μAP
DETReg	0.619	0.273	0.437	0.145	0.148
U²Net	0.672	0.413	0.503	0.309	0.265
Self-Reformer	0.648	0.495	0.097	0.017	0.129
Adaptive-FLIM	**0.763**	**0.667**	**0.570**	**0.498**	**0.423**

Table 3. Test set results considering our proposed solution with kernel selection (Adaptive-FLIM), U²Net, Self-Reformer and DETReg. The two best results for each metric are highlighted in blue and green, respectively.

Test set	$F_2^{0.5}$	$AP^{0.5}$	$F_2^{0.75}$	$AP^{0.75}$	μAP
DETReg	0.634	0.279	0.421	0.146	0.155
U²Net	0.740	0.531	0.609	0.405	0.335
Self-Reformer	0.747	0.688	0.114	0.024	0.227
Adaptive-FLIM	**0.799**	**0.929**	**0.630**	**0.488**	**0.525**

4.4. Qualitative Results

The first two rows of Figure 5 present images where our solution detected the correct component. The last row presents a fail case, where the object of interest was not detected.

In the first row, the object is obscured by impurities, and our method could still detect the component with a well-fitted bounding box. U²Net failed to detect the component, and SR detected it with a poorly defined bounding box. DETReg detected the object with a larger than necessary bounding box, with two additional false-positive boxes. This image is a good example of the recall loss of U²Net, the poor definition of bounding boxes by SR, and the decreased precision of DETReg.

The second row presents an image where Adaptive-FLIM detected the object with three false positives, but U²Net and SR failed to detect it. DETReg also detected the object, but it also predicted four false positives. The last row presents an image heavily obscured by impurities where no methods were able to detect the parasite. However, FLIM did not estimate any false positives.

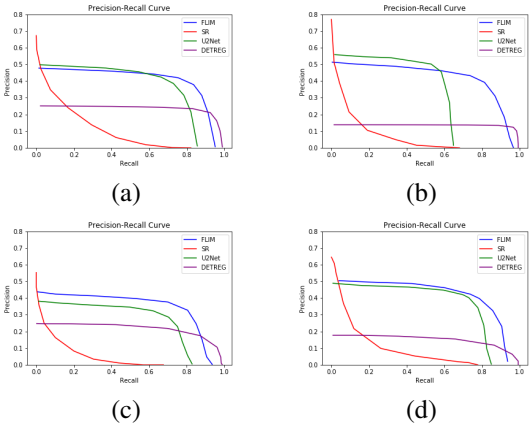


Fig. 4. Precision-Recall curves from splits one to three (a-c) and test set (d).

Table 4. Comparison among the number of parameters of different models. The deep models presented are used in our comparisons, and the lightweight models are standard backbones used for object detection. Adaptive-FLIM* is our model without manual kernel selection

Deep Models	#Parameters
DETReg	39.847.265
U²Net	44.009.869
Self-Reformer	91.585.457
Lightweight Models	#Parameters
MobileNetv2	3.504.872
SqueezeNet	1.248.424
Flyweight Models	#Parameters
Adaptive-FLIM*	53,333
Adaptive-FLIM	**2,296**

5. Conclusion

We presented a user-in-the-learning-loop methodology to improve FLIM-based encoders and build flyweight networks layer-by-layer with kernel selection. Using this methodology, we introduced **Adaptive-FLIM** – a flyweight network for *S. mansoni*'s egg detection using a novel unsupervised and adaptive decoder. We showed that **Adaptive-FLIM** outperformed the baselines in five object detection metrics, using only five images to train from scratch without segmentation masks while being thousands of times smaller, which is a relevant contribution towards embedded solutions for this application.

In future work, we intend to extend our approach to other problems, handle multi-class object detection, test architectures with more layers, and explore multi-scale feature extraction.

References

Bar, A., Wang, X., Kantorov, V., Reed, C.J., Herzig, R., Chechik, G., Rohrbach, A., Darrell, T., Globerson, A., 2021. Detreg: Unsupervised pretraining with region priors for object detection. arXiv preprint arXiv:2106.04550 .

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S., 2020. End-to-end object detection with transformers, in: European conference on computer vision, Springer. pp. 213–229.

Carvalho, J.B.d., Santos, B.M.d., Gomes, J.F., Suzuki, C.T.N., Hoshino Shimizu, S., Falcão, A.X., Pierucci, J.C., Matos, L.V.S.d., Bresciani, K.D.S., 2016. Tf-test modified: New diagnostic tool for human enteroparasitosis. Journal of clinical laboratory analysis 30, 293–300.

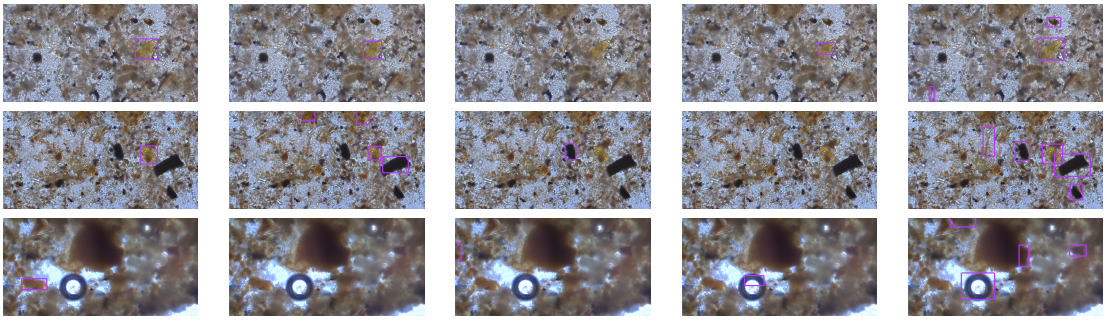


Fig. 5. Comparison among FLIM-Select and the baseline methods. (a) Ground-truth, (b) Adaptive-FLIM, (c) U²Net, (d) Self-Reformer, (e) DETReg.

De Souza, I.E., Falcão, A.X., 2020. Learning cnn filters from user-drawn image markers for coconut-tree image classification. IEEE Geoscience and Remote Sensing Letters .

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

Huang, G., Laradji, I., Vazquez, D., Lacoste-Julien, S., Rodriguez, P., 2021. A survey of self-supervised and few-shot object detection. arXiv preprint arXiv:2110.14711 .

Kang, B., Liu, Z., Wang, X., Yu, F., Feng, J., Darrell, T., 2019. Few-shot object detection via feature reweighting, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 8420–8429.

Katz, N., Chaves, A., Pellegrino, J., et al., 1972. A simple device for quantitative stool thick-smear technique in schistosomiasis mansoni. Revista do instituto de medicina tropical de São Paulo .

Kongs, A., Marks, G., Verle, P., Van Der Stuyft, P., 2001. The unreliability of the kato-katz technique limits its usefulness for evaluating s. mansoni infections. Tropical Medicine & International Health 6, 163–169.

Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L., 2014. Microsoft coco: Common objects in context, in: European conference on computer vision, Springer. pp. 740–755.

Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., Pietikäinen, M., 2020. Deep learning for generic object detection: A survey. International journal of computer vision .

MDS, B., 2010. Doenças infecciosas e parasitárias: guia de bolso/ministério da saúde, secretaria de vigilância em saúde, departamento de vigilância epidemiológica.–8. ed. rev. Brasília: Ministério da Saúde .

O Pinheiro, P.O., Almahairi, A., Benmalek, R., Golemo, F., Courville, A.C., 2020. Unsupervised learning of dense visual representations. Advances in Neural Information Processing Systems .

Otsu, N., 1979. A threshold selection method from gray-level histograms. IEEE Trans. on Systems, Man, and Cybernetics .

Qin, X., Zhang, Z., Huang, C., Dehghan, M., Zaiane, O.R., Jagersand, M., 2020. U2-net: Going deeper with nested u-structure for salient object detection. Pattern Recognition .

Santos, B.M., Soares, F.A., Rosa, S.L., Gomes, D.d.C.F., Oliveira, B.C.M., Peixinho, A.Z., Suzuki, C.T.N., Bresciani, K.D.S., Falcao, A.X., Gomes, J.F., 2019. Tf-test quantified: a new technique for diagnosis of schistosoma mansoni eggs. Tropical Medicine & International Health .

Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A.P., Bishop, R., Rueckert, D., Wang, Z., 2016. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1874–1883.

Sousa, A.M., Reis, F., Zerbini, R., Comba, J.L., Falcão, A.X., 2021. Cnn filter learning from drawn markers for the detection of suggestive signs of covid-19 in ct images, in: 2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC), IEEE. pp. 3169–3172.

de Souza, I.E., Benato, B.C., Falcão, A.X., 2020. Feature learning from image markers for object delineation, in: 2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), IEEE. pp. 116–123.

Uijlings, J.R., Van De Sande, K.E., Gevers, T., Smeulders, A.W., 2013. Selective search for object recognition. International journal of computer vision .

Wang, L., Lu, H., Wang, Y., Feng, M., Wang, D., Yin, B., Ruan, X., 2017. Learning to detect salient objects with image-level supervision, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 136–145.

Wang, W., Lai, Q., Fu, H., Shen, J., Ling, H., Yang, R., 2021a. Salient object detection in the deep learning era: An in-depth survey. IEEE Trans. on Pattern Analysis and Machine Intelligence .

Wang, W., Xie, E., Li, X., Fan, D.P., Song, K., Liang, D., Lu, T., Luo, P., Shao, L., 2021b. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 568–578.

Wei, F., Gao, Y., Wu, Z., Hu, H., Lin, S., 2021. Aligning pretraining for detection via object-level contrastive learning. Advances in Neural Information Processing Systems .

World Health Organisation, 2019. Schistosomiasis, fact sheet. `https://www.who.int/news-room/fact-sheets/detail/schistosomiasis`, Last accessed on 2022-05-09.

Wu, J., Liu, S., Huang, D., Wang, Y., 2020. Multi-scale positive sample refinement for few-shot object detection, in: European conference on computer vision, Springer. pp. 456–472.

Yan, X., Chen, Z., Xu, A., Wang, X., Liang, X., Lin, L., 2019. Meta r-cnn: Towards general solver for instance-level low-shot learning, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 9577–9586.

Yang, C., Wu, Z., Zhou, B., Lin, S., 2021. Instance localization for self-supervised detection pretraining, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 3987–3996.

Yun, Y.K., Lin, W., 2022. Selfreformer: Self-refined network with transformer for salient object detection. arXiv preprint arXiv:2205.11283 .

Zaidi, S.S.A., Ansari, M.S., Aslam, A., Kanwal, N., Asghar, M., Lee, B., 2022. A survey of modern deep learning based object detection models. Digital Signal Processing .