Recurrent Existence Determination Through Policy Optimization

Baoxiang Wang[1], **Shengyu Zhang**[1],
[1] The Chinese University of Hong Kong
bxwang@cse.cuhk.edu.hk, syzhang@cse.cuhk.edu.hk,

Abstract

Binary determination of the presence of objects is one of the problems where humans perform extraordinarily better than computer vision systems, in terms of both speed and preciseness. One of the possible reasons is that humans can skip most of the clutter and attend only on salient regions, which makes the process more efficient and more precise. Recurrent attention models (RAMs) are the first computational models to imitate the way humans process images via the REINFORCE algorithm. Despite the fact that RAM is originally designed for image recognition, we extend it and present recurrent existence determination model (RED) which shows that attention based mechanisms solve existence determination problems well. Our algorithm employs a novel k-maximum aggregation layer and a new reward mechanism to address the issue of delayed rewards, which would have caused the instability of the training process. The experimental analysis demonstrates massive efficiency improvement and competitive accuracy over existing approaches, on both synthetic and real-world datasets.

1 Introduction

Object existence determination[1] (ED) focuses on deciding if certain visual patterns exist in an image. As the basis of many computing vision tasks including, ED's quality affects further processing such as locating certain patterns (apart from telling the existence), segmentation of certain patterns, object recognition and object tracking in consecutive image frames. However, while ED is conducted by humans rapidly and effortlessly [Das *et al.*, 2016; Borji *et al.*, 2014], the performance of computer vision algorithms are surprisingly poor especially when the image is of large size and low quality. Hence, it is desirable to develop more efficient and noise-proof systems to deal with object detection tasks with large and noisy images.

In fact, the way humans process images is not similar to recent prevailing approaches such as detecting objects via convolution networks (ConvNets) and residual networks (ResNets) [He *et al.*, 2016]. Instead of taking all pixels from the image in parallel, humans perform sequential interactions with the image. Humans may deploy a few saccades and gather the information to make the final decision. Such behavior accomplishes ED tasks efficiently, especially for large images as it depends only on the number of saccades. Meanwhile, as the approach selectively learns to skip the clutter[2], it tends to be less sensitive to noise compared with those that take all pixels into the computation.

The process can be naturally interpreted as a Partially Observable Markov Decision Process (POMDP) where each image represents an environment. At the beginning of the process, the agent conducts an action which is represented by a 2-dimension Cartesian coordinate. When the environment receives the action, it calculates the retina-like representation of the image at the corresponding location, and returns that representation to the agent as the agent's observation. Repeatedly until the last step, the agent predicts the detection result based on the trajectory and receives the evaluation of its prediction as the reinforcement signal.

Recurrent attention models (RAMs) [Mnih *et al.*, 2014] are the first computational models to imitate the process with a reinforcement learning (RL) algorithm. The success of RAMs leads to enormous studies on attention-based computer vision solutions [Yeung *et al.*, 2016; Gregor *et al.*, 2015]. However, RAMs and their extensions [Ba *et al.*, 2015; 2014] are designed to solve object recognition tasks such as handwritten digit classification. Those models largely ignore the trajectory information which causes massively delayed rewards. Indeed in RAMs, the reward function is associated with only the last step of the process (and is otherwise zero).

[1] Certain literatures [Zagoruyko *et al.*, 2016] may refer the problem using the terminology *object detection*. More commonly object detection refers to both deciding the existence of certain patterns and subsequently locating them if so.

[2] Clutter are the irrelevant features of the visual environment, defined in RAMs [Mnih *et al.*, 2014].

The actions before that, which deploy the attention for the model, do not receive direct feedback and are therefore not efficiently guided. Especially in ED (and in general, object detection) tasks, delayed rewards fail to provide reinforcement signals to the choice of locations when the glimpse at certain locations may provide explicit information for the existence of the object.

We present recurrent existence determination models (RED), which inherit the advantage of RAMs that the attention is only deployed on locations that are deemed informative. Our approach involves a new observation setting which allows the agent to have access to explicit visual patches. Unlike previous trails which blur the pixels that are far from the saccade location, we acquire the exact patches which help to detect the existence of specific patterns. We employ gated recurrent units (GRUs) [Chung *et al.*, 2014] to encode the historical information acquired by the agent and generates temporary predictions at each time step. The temporary predictions over the time horizon are aggregated via a novel k-maximum aggregation layer, which averages the k-greatest value to provide the final decision. It allows the rewards to be backpropagated to the early and middle stages of the processing directly apart from through the recurrent connections of the RNN unit. It provides immediate feedback which guides the agent to allocate its attention, and therefore addresses the issues caused by delayed rewards.

RED is evaluated empirically on both synthetic datasets and real-world datasets. *Stained MNIST* is a set of handwritten digits from MNIST. Additionally, the resolution has been enlarged and each digit may be added dot stains around the writings. The dataset is designed to compare the performance of RED and existing algorithms on images with high-resolution settings. The results show that attention based models run extraordinarily faster than traditional methods [Dieleman *et al.*, 2015; Graham, 2014], such as ConvNets, and have better accuracy as well. Experiments on real-world dataset show superior speed improvement and competitive accuracy on retinopathy screening, compared to existing approaches. Our proposed approach shows to be promising in terms of both speed and preciseness, on both datasets.

2 Preliminaries

2.1 Policy Gradients

In an RL episode, at each time step t, the agent takes an action a_t from the set $\mathcal{A}_t$ of feasible actions. Receiving the action from the agent, the environment updates its internal state, and returns an observation x_t and a scalar reward r_t to the agent accordingly. In most of the problems, the observation does not fully describe the internal state of the environment, and the agent has to develop its policy using only the partial observations of the state. This process continues until the time horizon T. Let $R_t = \sum_{t'=1}^{t'=t} r_{t'}$ denote the cumulative rewards up to time t, the policy is trained to maximize its cumulative rewards R_T. Let π_θ be the policy function, the REINFORCE algorithm [Williams, 1992; Mnih *et al.*, 2016] estimates the policy gradient using

$$g = \mathbb{E}_\pi[\nabla_\theta \log \pi(a_t|s_t)(R_t - b_t)] \tag{1}$$

where s is the agent state and b_t is an arbitrary function for variance reduction.

RED is an model-based RL algorithm where the agent is an RNN. It decides the action based on the trajectory $\tau_t = (a_1, x_1, r_1, \ldots, a_{t-1}, x_{t-1}, r_{t-1})$ which includes past actions, observations, and rewards. To handle the growing dimensionality of τ_t, the agent maintains an internal state[3] s_t which encodes the trajectory, and updates it repeatedly until the end of the time horizon. In this way, the action is decided by the policy function, based only on the internal state s_t of the agent Note that the full state of the environment is τ_t and the image to be processed, and the agent observes τ_t only. Define a rollout $\hat{\tau_T}$ of the trajectory within an episode to be a sample drawn from the probability distribution $\mathbb{P}(\tau_T|\pi_\theta(\cdot))$.

The training of RL repeats the above process from time 1 to time T for a certain number of episodes. At the beginning of each episode, the agent resets its internal state while the environment resets its internal state as well. The model parameters are maintained across multiple episodes and are updated gradually as the occurrence of the reinforcement signals at the end of each episode. Note that different from general online learning framework, in RED the agent does not receive the reinforcement signal in the middle stages of an episode. Hence the reinforcement signals are inevitably heavily delayed, and RED need to address temporal credit assignment problem [Sutton, 1984], which evaluates individual action within a sequence of actions according to a single reinforcement signal.

2.2 Glimpse and Retina-Like Representations

A retina-like representation is the visual signal humans receive when glimpsing at a point of an image. The visual effect is that regions close to the focused location tend to retain their original, high-resolution form, while regions far from the focused location are blurred and passed to the human brain in their low-resolution form. In RED, the environment calculates the retina-like representations and returns it as the observation. Existing approaches to mimic such visual effects have been used in RAMs and RAMs' variants. They can be categorized into two classes: soft attention [Gregor *et al.*, 2015; Xu *et al.*, 2015] and hard attention [Eslami *et al.*, 2016; Xu *et al.*, 2015; Mnih *et al.*, 2014].

Soft attention [Hermann *et al.*, 2015] applies a filter centered at the focused location. It imitates human behaviors, which downsamples the image gradually as it approaches far away from the focused point, resulting in a smooth representation. The approach is fully differentiable and is hence amenable to be trained straightforwardly using gradient descent. Despite those merits, soft attention is in general computationally expensive as it involves the convolution operation over all pixels, which

[3] In our paper, s_t is defined to be the state of the agent instead of the state of the environment.

deviates from the idea of RED which only examines parts of the image and subsequently making the process relatively inefficient.

Hard attention, on the other hand, extracts pixels with predefined sample rates. Fewer pixels are extracted as the region approaches further away from the focused location, making the process cost only constant time. Hard attention fits the idea of RED well though it is non-differentiable as it indexes and extracts pixels. To address the non-differentiability, we develop our training algorithm via policy gradient and use the rollouts of the algorithm to estimate the gradient. Formally, let x_t be a list of c channels and the i-th channel extracts the squared region centered at a_t with size $n_i \times n_i$, and down sample the patch to $n_1 \times n_1$. The channels are incorporated with the location information (known as the what and where pathways) with the patch information by adding the affine transformation $\tanh(W_{xa}a_t)$ of a_t. Note that the value of each entry W_{xa} will be restricted to be relatively small compared to the pixel values to retain the original patch information.

2.3 Convolutional Gated Recurrent Units

We use RNNs to encode the trajectory and update the internal states of the agent. While both LSTMs and GRUs are prevailing RNN variants in sequential data processing [Chung *et al.*, 2014], GRUs are preferred to LSTMs in RED. The reason is that the input passes through the unit more explicitly as a GRU merges the memory cell state and the output state in an LSTM unit. The explicit information helps with making the temporary detection decisions and enables our design of k-maximum aggregation layer. Meanwhile, as the merge happens, the agent updates its internal state more efficiently. The speed improvement is important especially for real-time applications such as surveillance anomaly detection when instant detection decision is required.

We use convolutional GRUs, a variant of GRUs where the matrix product operations between the output state s_t, the input x_t, and the model parameters are replaced with convolution operations, and s_t and x_t are kept in their 2-dimension matrix shapes [Shi *et al.*, 2015]. A graphical illustration of GRUs is shown in Fig. 1, where lines in orange represent convolution operation. The gate mechanism in convolutional GRUs is formulated as follows

$$\begin{aligned} z_t &= \sigma(W_{zh} * s_{t-1} + W_{zx} * x_t) \\ v_t &= \sigma(W_{rh} * s_{t-1} + W_{rx} * x_t) \\ \tilde{s}_t &= \tanh(W_{sh} * (v_t \circ s_{t-1}) + W_{sx} * x_t) \\ s_t &= (1 - z_t)s_{t-1} + z_t\tilde{s}_t, \end{aligned} \tag{2}$$

where "$*$" denotes convolution, "$\circ$" denotes Hadamard product, $\sigma(\cdot)$ denotes the sigmoid function, and z_t and v_t are the update gate and the reset gate, respectively. W_{zh}, W_{zx}, W_{rh}, W_{rx}, W_{sh} and W_{sx} are trainable parameters. Convolutional GRUs retains the spacial information in the output state so that temporary detection decisions can be made well before the end of an episode.

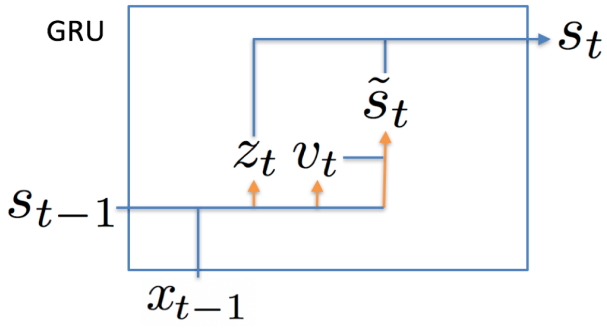

Figure 1: Graphical Illustration of Convolutional Gated Recurrent Units

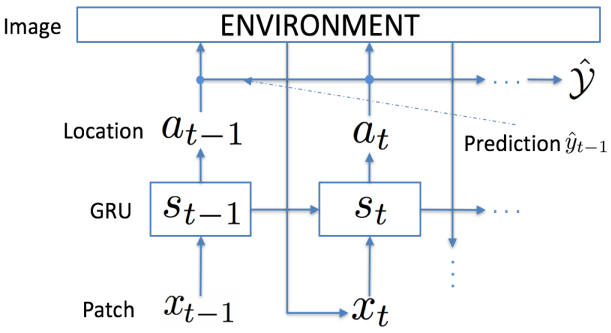

Figure 2: The Attention Mechanism and The Reward Mechanism in RED

3 Recurrent Existence Determination

3.1 Attention Mechanism in RED

We formulate the attention mechanism within each of the episodes, that is, within the processing of one image. Let $\mathcal{I}$ denote the image, the agent has x_0, which is the low-resolution form of $\mathcal{I}$, as the initial observation. The internal state s_0 of the agent is initialized as $\mathbf{0}$. Repeatedly, at each time step t, the agent calculates its action $a_t \in \mathbb{R}^2$, according to

$$a_t = \tanh(W_{as}s_t) + \epsilon_t, \tag{3}$$

where W_{as} is a trainable parameter of the model and ϵ_t is a random noise to improve exploration. The action a_t refers to a Cartesian Coordinate on the image, with $(-1, -1)$ corresponding to the bottom-left corner of $\mathcal{I}$ and $(1, 1)$ corresponding to the top-right corner of $\mathcal{I}$. Each entry of ϵ_t is sampled from a normal distribution with fixed standard deviation β, independently. The environment returns the retina-like representation x_t via the hard attention model described in Sec. 2.2.

The agent employs a single convolutional GRUs and uses the output states s_t as the agent's internal state, defined in Eq. (2). The state s_t has the same shape $n_1 \times n_1$ as each channel of the observation does, which is ensured by the convolution operation in Eq. (2). We take advantage of GRUs that the reset gate v_t in Eq. (2) controls the choice between long-term dependencies and short-term observations. The former is important for exploring

future attention deployment within an episode, and the later is important for exploiting currently available information to make temporary decisions. By training over a large number of episodes, the agent learns to balance exploration and exploitation from the reinforcement signals by updating its gate parameters.

With Eq. (2), Eq. (3) and the hard attention mechanism each rollout is computed in constant time with respect to the image size as T is fixed. As a result, a trained RED model is able to make predictions very efficiently.

3.2 Prediction Aggregation

We present the framework to generate temporary predictions, and subsequently, aggregate the temporary predictions into the final prediction of an image. At each time step t, the agent has access to the output state s_t of the GRU which carries the information from the current patch x_t. Based on s_t the agent makes a temporary prediction $\hat{\mathcal{Y}}$ using a feed-forward network followed by a non-linear operation $\hat{y}_t = \frac{1}{2}(1 + \tanh(W_{ys} s_t))$, where $\hat{y}_t$ is the estimated probability that the object exists in $\mathcal{I}$. W_{ys} is a trainable parameter which represents the 1-layer feed-forward network.

The temporary predictions are aggregated over time using our newly proposed k-maximum aggregation layer. The layer calculates the weighted average of the top k largest values among $\hat{y}_{t_0}, \cdots, \hat{y}_T$, where $t_0 \geq 1$ is a fixed threshold of the model. The output $\hat{\mathcal{Y}}$ of the k-maximum layer is formulated as

$$\hat{\mathcal{Y}} = \frac{1}{Z} \sum_{t \in K} (1 - \gamma^t) \hat{y}_t, \tag{4}$$

where $K = \text{argmax-}k_{\leq t \leq T}\{\hat{y}_t\}$ is the set of the indexes of the top k-largest temporary predicted probabilities, and $Z = \sum_{t \in K}(1 - \gamma^t)$ is the normalizer to guarantee $0 \leq \hat{\mathcal{Y}} \leq 1$. In Eq. (4) we elaborate a time discount factor $1 - \gamma^t$ which assigns a larger value toward the late stages of the process than the early stages of the process, where γ is fixed through the process. The factor γ is a trade-off between RAMs where all previous steps are used to benefit the prediction at the end of the episode and majority voting where all observation contributes to the binary determination.

The benefit of the k-maximum layer is to guide the model to balance between exploration and exploitation[4]. Consider that only steps t with top k largest $\hat{y}_t$ are taken into account of the final prediction, the model has a sufficient number of time steps to explore different locations on $\mathcal{I}$ and does not need to worry about affecting the final prediction. In fact, exploring the context of the image is important to collect information and locate the detection objective in late stages. The time discount factor further reinforces that by assigning larger weights toward late stages, which encourages the agent to explore at the early stages of the process and exploit at the late stages of the process.

Viewing our proposed prediction aggregation mechanism from an RL perspective, it addresses the credit assignment problem. Existing studies on applications via policy learning, e.g. AlphaGo [Silver *et al.*, 2016] and Neural Fictitious Self-Play [Heinrich and Silver, 2016], equally assign the feedback of an episode toward all actions the agent has made. The large variance of estimating the quality of a single action using the outcome of the entire episode is neutralized by training the agent for millions of episodes. However, in our settings the state of the environment is diverse as each different image $\mathcal{I}$ corresponds to a unique initial state of the environment. The variance cannot be reduced by simply training on a large dataset of images without a fixed observation function with respect to $\mathcal{I}$. In this way, our proposed aggregation mechanism is necessary to help the algorithm to converge and it is the key component for RED to make detection decisions.

3.3 Policy Gradient Estimation

Let W denote the set of trainable parameters including θ, W_{as}, W_{xa}, W_{ys} and the trainable parameters in Eq. (2). Also let $\mathcal{Y} \in \{0, 1\}$ be the ground truth of the detection result, where 0 and 1 correspond to existence and non-existence of the object, respectively. During training, the agent generates its rollouts $\hat{\tau}_T$ and predictions $\hat{\mathcal{Y}}$ on an iterator of $(\mathcal{I}, \mathcal{Y})$ pairs, where each pair of the image and the ground truth corresponds to one episode of RL. Define the regret L_T to be the squared error between the predicted probability and the ground truth

$$L_T = (\hat{\mathcal{Y}} - \mathcal{Y})^2. \tag{5}$$

The model updates W after the conclusion of each episode in order to minimize L_T. Note that $L_T + R_T = 1$.

We utilize policy gradient to address the non-differentiability. Let $\boldsymbol{a} = (\hat{a}_1, \ldots, \hat{a}_T)$ be the sequence of actions in $\hat{\tau}_T$, we have the expected regret

$$\mathbb{E}[L_T|W] = \sum\nolimits_{\boldsymbol{a}} \mathbb{P}(\boldsymbol{a}|W)(\hat{\mathcal{Y}}_{\boldsymbol{a}} - \mathcal{Y})^2, \tag{6}$$

where the deterministic variable $\hat{\mathcal{Y}}_{\boldsymbol{a}}$ denotes the model's counterfactual prediction under the condition that $\boldsymbol{a}$ is sampled with probability 1. Since there is no randomness involved on the environment side, the expectation above is calculated over the actions only. Taking derivative with respect to W, the gradient is

$$\begin{aligned} \nabla_W \mathbb{E}[L_T|W] = \mathbb{E}_{\boldsymbol{a} \sim \mathbb{P}(\boldsymbol{a}|W)}[(\hat{\mathcal{Y}}_{\boldsymbol{a}} - \mathcal{Y})^2 \\ \nabla_W \log \mathbb{P}(\boldsymbol{a}|W) + 2(\hat{\mathcal{Y}}_{\boldsymbol{a}} - \mathcal{Y}) \nabla_W \hat{\mathcal{Y}}_{\boldsymbol{a}}], \end{aligned} \tag{7}$$

where the immediate partial derivative from chain Eq. (3) is

$$\nabla_W \log \mathbb{P}(a_t|W) = \frac{1}{\beta^2} \cdot (a_t - \mathbb{E}[a_t|W]) s_{t-1}^T. \tag{8}$$

Further, deduct the reward the value function

$$b_T = \mathbb{E}_{\boldsymbol{a} \sim \mathbb{P}(\boldsymbol{a}|W)}[(\hat{\mathcal{Y}}_{\boldsymbol{a}} - \mathcal{Y})^2] \tag{9}$$

[4]It also helps to address the problems of vanishing gradient at the same time, though, it's out of the scope of this paper.

which calculates the expected regret from the rollouts. We count only the difference between the actual reward and the value function. The value function brings no bias into the expectation in Eq. (7), while reduces variance when it is estimated using sampled $\boldsymbol{a}$. At the end of each episode, update W according to

$$W := W - \alpha \mathbb{E}_{\boldsymbol{a}\sim\mathbb{P}(\boldsymbol{a}|W)}[((\hat{\mathcal{Y}}_{\boldsymbol{a}} - \mathcal{Y})^2 - b_T) \nabla_W \log \mathbb{P}(\boldsymbol{a}|W) + 2(\hat{\mathcal{Y}}_{\boldsymbol{a}} - \mathcal{Y})\nabla_W \hat{\mathcal{Y}}_{\boldsymbol{a}}], \tag{10}$$

where α is the learning rate.

To estimate the expectation in Eq. (10), the agent generates a rollout $\hat{\tau}_T$ which samples $\boldsymbol{a}$ according to $\boldsymbol{a} \sim \mathbb{P}(\boldsymbol{a}|W)$. The expectation is then estimated using the generated $\boldsymbol{a}$ value by Eq. (8) and REINFORCE's back-propagation [Wierstra *et al.*, 2007]. Note that the second part $2(\hat{\mathcal{Y}}_{\boldsymbol{a}} - \mathcal{Y})\nabla_W \hat{\mathcal{Y}}_{\boldsymbol{a}}$ of the gradient is useful, though, it is sometimes ignored in previous studies [Mnih *et al.*, 2014]. It connects the regret to the early stages which allows the regret to be back-propagated directly those stages and to guide the exploitation of the agent. It can be regarded as retrospectively assigning credits after the rollout has been generated, making the reward r_t in RED is no longer 0 when $t < T$ during the training phase, which addresses the issues caused by delayed rewards.

4 Experimental Evaluation

4.1 Stained MNIST

We first examine RED on our synthetic dataset, *stained MNIST*. Stained MNIST contains a set of handwritten digits, which have very high resolution and much thinner writings than MNIST does. Each digit may be associated with multiple "stains" on the edge of its writing, which are dot-shaped regions with high tonal value. The algorithms are required to predict if such stains exist in the images. The task is very challenging since the image resolution is very high while the writings are thin and unclear. Hence it is hard to locate the stains or recognize the stains from the writings.

Stained MNIST is constructed by modifying the original MNIST dataset as follows. Each image from MNIST is first resized to 7168×7168 by bilinear interpolation, and rescaled to 0 to 1 tonal value. The enlarged images are then smoothed using a Gaussian filter with a 20×20 kernel. After that, it calculates the central differences of each pixel and finds out the set C of pixels with 0.2 or larger gradient. The tonal values of pixels that are within 500 pixels of C are set to 0. The operation makes the writings of the digits much thinner in the high-resolution images. After removing those pixels, the gradient of each pixel is calculated again, and 10 to 15 stains with radius 12 are randomly drawn at pixels with high gradient[5].

[5]we are unable to include examples of stained MNIST in this paper since the writings are so thin that they can not be observed under paper resolution.

Table 1: Performances of Different Approaches on Stained MNIST

Approach	Time(s)	Accuracy
RED	0.06	84.43%
Random $\boldsymbol{a}$	0.06	51.79%
RAMs	0.06	62.35%
ConvNets2	1.95	81.49%
ConvNets3	3.30	82.92%

The hyper-parameters of RED are set to be $c = 3, n_1 = 18, n_2 = 36, n_3 = 54$ for attention mechanism and $\gamma = 0.95, k = 25$ for prediction aggregation, through a random search on a training subset. The horizon is fixed to $T = 350$, where no significant improvement can be observed by further increasing it. When evaluating RED, we rule out the stochastic components ϵ_t in the actions.

We evaluate both the accuracy and the average time cost to make a prediction by RED with by other approaches. We compare RED with both a 3-layer ConvNet and a 4-layer ConvNet. The ConvNets takes the full image as input and outputs the probability of the existence of the objects. We also include a base model, *Random* $\boldsymbol{a}$, which replaces Eq. (3) with sampling $\hat{a}_t$ uniformly from $\mathcal{A}_t = [-1, 1]^2$. The baseline is used to show the necessity of the attention mechanism. As shown on Tbl. 1, RED outperforms all other models in terms of accuracy, and all attention based models have extraordinarily better speed compared with ConvNets based algorithms.

4.2 Diabetic Retinopathy Screening

Diabetic Retinopathy (DR) [Fong *et al.*, 2004] is an interesting problem in medical image analysis. It screens the disease by determining the existence of several of the leisure patterns from the fundus image. We take advantage of RED that it processes the high-resolution fundus images very efficiently and skips most of the clutter. We validate the performance using a dataset publicly available on Kaggle[6]. The fundus images are originally rated with five levels, and we merge level 0 and level 1 as a negative result $\mathcal{Y} = 0$, and merge level 2, 3 and 4 to a positive result $\mathcal{Y} = 1$. We use the same hyper-parameters settings as is in stained MNIST. The results are shown in Tbl. 2.

The performance of our RED approach is compared with RAMs and ConvNets with both four layers and five layers. Also, we test ConvNets with fractional max-pooling layers [Graham, 2014] and cyclic pooling layers [Dieleman *et al.*, 2015] which have solid performances on the original Kaggle challenge. Note that their original network architecture involves fourteen convolutional layers and over 6 million parameters, which is unfair for RED to compete with. We re-implement their approach (*ConvNets4+* and *ConvNets5+*) with 4 and 5 layers and the comparison is shown in Tbl. 2.

[6]https://www.kaggle.com/c/diabetic-retinopathy-detection

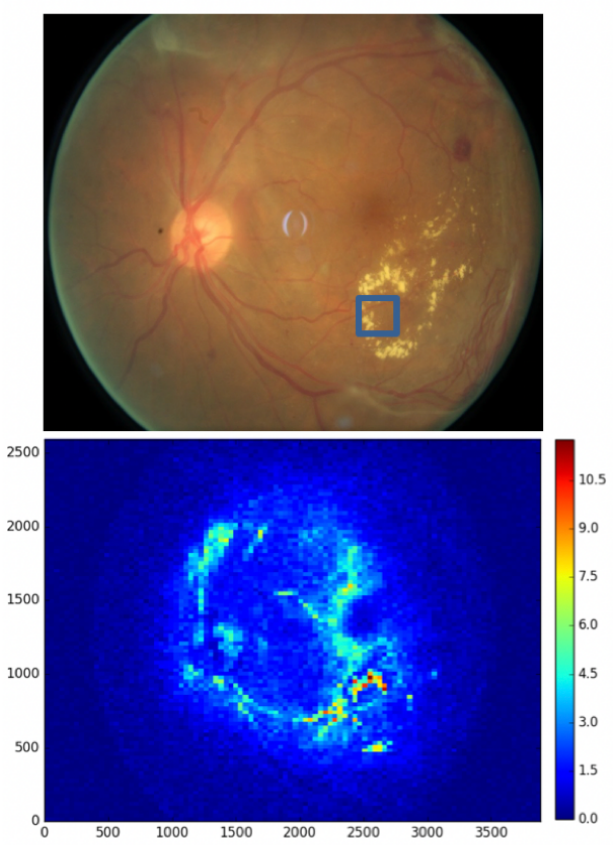

Figure 3: Distribution of the Attentions in a Rollout of RED

As shown in Tbl. 2, our RED approach achieves extraordinary speed performance while achieving competitive accuracy. Notably, compared with the ConvNets-based methods which usually take many seconds to process each image, RED provides a way to trade marginal accuracy to significant speed improvement. That could be critical especially for algorithms that are designed to be applied on population-wise datasets or required to give instant decisions. Apart from the speed improvement, it is worth to note that RED is light weighted, that is, the number of parameters in W is small, as RED process patches with size $n_1 \times n_1$ only at any time t. Our employment of Convolution GRUs further guarantees this because it has fewer states and gates than LSTM and variants. The experiments on DR screening indicate the application values of RED.

4.3 Intuitive Demonstration of The Trajectory

To understand the policy the agent deploy its attention, we present a graphical demonstration of the trajectory, which imitates the way humans process existence detection tasks. As shown in Fig. 3 top, the trained agent predicts if patterns related to DR exist in the fundus image. To observe the trajectory, we put the limit $T \to \infty$ on the time horizon while keeping the stochastic components in Eq. (3) active. We then illustrate the distribution of attention, in the form of a heat map, in Fig. 3 bottom.

We first observe that the attention are majority crowded in the bottom right part of the image, which coincides with the lesion patterns (yellow stains on the fundus image). Within the small blue box marked on Fig. 3 top concentrates 30 out of the first 250 saccades. It shows the ability of the trained model to locate regions of interest and to deploy its limited attention resource selectively. Notably, only 6 of them happen in the first 100 time steps, and the density of attention for $T \to \infty$ becomes even higher (≥ 9 heat value on Fig. 3 bottom). On the other hand, we observe that the model tends to put attentions on around the vessels especially at the early stages of the process. Such behavior helps the agent to gain information about the context of the image and predict and locate the region of interest in the late stages of the process. Also, it is worth to note that the agent does not stuck in a small region even when we set the time horizon to be arbitrage large. Instead, the agent put increasing but not all resource on exploitation and keep exploring the image. The way the agent automatically balance exploitation and exploration is what we have been expecting an RL algorithm to learn.

Table 2: Performances of Different Approaches on Retinopathy Screening

Approach	Time(s)	Accuracy
RED	0.04	91.55%
Random $\boldsymbol{a}$	0.04	53.44%
RAMs	0.04	81.35%
ConvNets4	2.32	90.61%
ConvNets4+	2.32	91.97%
ConvNets5	2.92	91.84%
ConvNets5+	2.92	92.29%

5 Conclusion and Future Works

We present RED, a novel RL algorithm for existence detection. RED imitate the attention mechanism that humans elaborate to process object detection, which is both efficient and precise. We propose k-maximum aggregation layer and other components in RED which help to ease the delayed reward problem and automatically learns to balance exploration and exploitation. RED employs hard attention which boosts the test-time speed and solves the non-differentiability via policy optimization. Experimental analysis shows massive speed improvement compared with previous approaches while keeping competitive accuracy performance.

Future works include applying a value network as the critic mechanism in actor-critic [Mnih *et al.*, 2016] to help further address the delayed reward. Especially, as the states are partially observable, the actor-critic could be asymmetric where the critic should have access to the full image. Training such an actor-critic pair induces large variance to both the policy network and the value network because of the partially observable nature. If train properly, the critic network is pronely a proper replacement of the aggregation layer and is expected to send the agent the feedback at any step.

References

[Ba *et al.*, 2014] Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. Multiple object recognition with visual attention. *arXiv preprint arXiv:1412.7755*, 2014.

[Ba *et al.*, 2015] Jimmy Ba, Ruslan R Salakhutdinov, Roger B Grosse, and Brendan J Frey. Learning wake-sleep recurrent attention models. In *Advances in Neural Information Processing Systems*, pages 2593–2601, 2015.

[Borji *et al.*, 2014] Ali Borji, Ming-Ming Cheng, Huaizu Jiang, and Jia Li. Salient object detection: A survey. *arXiv preprint arXiv:1411.5878*, 2014.

[Chung *et al.*, 2014] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[Das *et al.*, 2016] Abhishek Das, Harsh Agrawal, C Lawrence Zitnick, Devi Parikh, and Dhruv Batra. Human attention in visual question answering: Do humans and deep networks look at the same regions? *arXiv preprint arXiv:1606.03556*, 2016.

[Dieleman *et al.*, 2015] Sander Dieleman, Kyle W Willett, and Joni Dambre. Rotation-invariant convolutional neural networks for galaxy morphology prediction. *Monthly Notices of the Royal Astronomical Society*, 450(2):1441–1459, 2015.

[Eslami *et al.*, 2016] SM Ali Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, David Szepesvari, Geoffrey E Hinton, et al. Attend, infer, repeat: Fast scene understanding with generative models. In *Advances In Neural Information Processing Systems*, pages 3225–3233, 2016.

[Fong *et al.*, 2004] Donald S Fong, Lloyd Aiello, Thomas W Gardner, George L King, George Blankenship, Jerry D Cavallerano, Fredrick L Ferris, and Ronald Klein. Retinopathy in diabetes. *Diabetes care*, 27(suppl 1):s84–s87, 2004.

[Graham, 2014] Benjamin Graham. Fractional max-pooling. *arXiv preprint arXiv:1412.6071*, 2014.

[Gregor *et al.*, 2015] Karol Gregor, Ivo Danihelka, Alex Graves, and Daan Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[Heinrich and Silver, 2016] Johannes Heinrich and David Silver. Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121*, 2016.

[Hermann *et al.*, 2015] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701, 2015.

[Mnih *et al.*, 2014] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems*, pages 2204–2212, 2014.

[Mnih *et al.*, 2016] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy P Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, 2016.

[Shi *et al.*, 2015] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun WOO. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems*, pages 802–810, 2015.

[Silver *et al.*, 2016] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[Sutton, 1984] Richard Stuart Sutton. Temporal credit assignment in reinforcement learning. 1984.

[Wierstra *et al.*, 2007] Daan Wierstra, Alexander Foerster, Jan Peters, and Juergen Schmidhuber. Solving deep memory pomdps with recurrent policy gradients. In *Artificial Neural Networks–ICANN 2007*, pages 697–706. Springer, 2007.

[Williams, 1992] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pages 5–32. Springer, 1992.

[Xu *et al.*, 2015] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, volume 14, pages 77–81, 2015.

[Yeung *et al.*, 2016] Serena Yeung, Olga Russakovsky, Greg Mori, and Li Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2678–2687, 2016.

[Zagoruyko *et al.*, 2016] Sergey Zagoruyko, Adam Lerer, Tsung-Yi Lin, Pedro O Pinheiro, Sam Gross, Soumith Chintala, and Piotr Dollár. A multipath network for object detection. *arXiv preprint arXiv:1604.02135*, 2016.