TYPE Original Research
PUBLISHED 18 November 2022
DOI 10.3389/fncom.2022.1012559

BIAS-3D: Brain inspired attentional search model fashioned after what and where/how pathways for target search in 3D environment

Sweta Kumari[1], V. Y. Shobha Amala[2], M. Nivethithan[1] and V. Srinivasa Chakravarthy[1]*

[1]Computational Neuroscience (CNS) Lab, Department of Biotechnology, IIT Madras, Chennai, India, [2]IIT BHU, Varanasi, India

Check for updates

OPEN ACCESS

EDITED BY
Yuqi Han,
Tsinghua University, China

REVIEWED BY
Tianshan Liu,
Hong Kong Polytechnic University, Hong Kong SAR, China
Zbigniew Kowalewski,
AGH University of Science and Technology, Poland

*CORRESPONDENCE
V. Srinivasa Chakravarthy
schakra@ee.iitm.ac.in

RECEIVED 05 August 2022
ACCEPTED 26 October 2022
PUBLISHED 18 November 2022

CITATION
Kumari S, Shobha Amala VY, Nivethithan M and Chakravarthy VS (2022) BIAS-3D: Brain inspired attentional search model fashioned after what and where/how pathways for target search in 3D environment. *Front. Comput. Neurosci.* 16:1012559. doi: 10.3389/fncom.2022.1012559

We propose a brain inspired attentional search model for target search in a 3D environment, which has two separate channels—one for the object classification, analogous to the "what" pathway in the human visual system, and the other for prediction of the next location of the camera, analogous to the "where" pathway. To evaluate the proposed model, we generated 3D Cluttered Cube datasets that consist of an image on one vertical face, and clutter or background images on the other faces. The camera goes around each cube on a circular orbit and determines the identity of the image pasted on the face. The images pasted on the cube faces were drawn from: MNIST handwriting digit, QuickDraw, and RGB MNIST handwriting digit datasets. The attentional input of three concentric cropped windows resembling the high-resolution central fovea and low-resolution periphery of the retina, flows through a Classifier Network and a Camera Motion Network. The Classifier Network classifies the current view into one of the target classes or the clutter. The Camera Motion Network predicts the camera's next position on the orbit (varying the azimuthal angle or "θ"). Here the camera performs one of three actions: move right, move left, or do not move. The Camera-Position Network adds the camera's current position (θ) into the higher features level of the Classifier Network and the Camera Motion Network. The Camera Motion Network is trained using Q-learning where the reward is 1 if the classifier network gives the correct classification, otherwise 0. Total loss is computed by adding the mean square loss of temporal difference and cross entropy loss. Then the model is trained end-to-end by backpropagating the total loss using Adam optimizer. Results on two grayscale image datasets and one RGB image dataset show that the proposed model is successfully able to discover the desired search pattern to find the target face on the cube, and also classify the target face accurately.

KEYWORDS

attention, memory, human visual system, what and where pathway, convolutional neural network, search in 3D, flip-flop neurons

1. Introduction

Human visual system (HVS) processes a restricted field of view of about 150° in the horizontal line and 210° in the vertical line (Knapp, 1938). However, the eye orientates itself in such a manner that the image of the region of interest falls inside the central part of the retina or fovea to obtain precise information from that part of the visual field. Information from the fovea in high resolution and periphery in low resolution is passed through the visual hierarchy, and the features related to the form, color, and motion are analyzed by respective visual cortical areas. Due to this anatomical constraint, the eye does not process the entire scene at once: the eye makes darting movements called saccades and attends the salient parts of the scene sequentially and integrates the pieces of the image to get a more comprehensive understanding of the scene.

Visual attention is a popular topic in both computer vision and visual neuroscience. Many computational models of visual attention, proposed in the past couple of decades, may be divided into two categories: bottom-up approaches (Le Meur et al., 2006; Gao et al., 2008), and top-down approaches (Gao et al., 2009; Kanan et al., 2009; Borji et al., 2012). The models are basically developed to predict the saliency map, where a brighter pixel has a higher probability of receiving human attention and vice versa. Bottom-up attention is considered to be stimulus driven whereas top-down attention is considered to be task driven, which receives human attention based on the explicit understanding of the image content. Prior attempts in the field of top-down attention mechanisms (Gao et al., 2009; Kanan et al., 2009; Borji et al., 2012) have mainly used non-deep approaches such as the Bayesian approach (Borji et al., 2011), based on a limited understanding of visual attention. In a recent model of visual attention, Mnih et al. (2014) have developed a recurrent attention model (RAM) which takes a glimpse of the attention window as input and uses the internal state of the network to find the next location to focus on in a non-static environment. Their proposed network processes multiple glimpses of windows to attend to a part of the image at different levels of resolutions. Training of their model is done by using the reinforcement learning approach for classification of MNIST dataset for modeling task-driven visual attention. Design of their network is based on fully connected layers, which leads to a rapid increase in computational cost with image size, and therefore the network is perhaps not feasible for more complex real world tasks such as search in a 3D environment.

There is an extensive number of research studies that demonstrate the application of attentional search methods to solve real world problems in 2D space such as image cropping (Xu et al., 2019), object recognition (Gao and Vasconcelos, 2004), object segmentation (Shen et al., 2014; Wang et al., 2015a,b), video understanding (Zhang et al., 2015, 2017; Yang et al., 2016a,b,c), and egocentric activity recognition (Liu et al., 2021, 2022). These models are based on covert attention, where the mental shift of attention occurs at the output activity map without explicit eye movement. But the use of overt visual attention in a 3D environment is still relatively under-explored. Earliest work in 3D target search is the Shape Nets (Wu et al., 2015) where the objective was to voxelize the target and use deep belief networks for training and prediction. Minut and Mahadevan (2001) used Q learning to identify the next movement of the camera (action) out of the eight possible actions in order to focus on the object of interest. At a lower level, this approach uses histogram back projection color maps and symmetry map to identify the objects. Unlike reinforcement learning based approaches, the model proposed by Kanezaki et al. (2018) named RotationNet, focuses on convolutional neural networks (CNNs) based approaches where multiple views of the object are taken into consideration for learning. The model predicts the class and the pose (orientation) of the object of consideration. This was an improvement over the previous CNN based networks, that recognized the object but failed to predict the pose. The model yielded an accuracy of 94% on Modelnet40 dataset (Wu et al., 2015) consisting of 40 categories including chair, airplane, etc. Multiview CNN (Su et al., 2015) was one of the earliest attempts in 3D object recognition that acts as a precursor of the RotationNet.

In the model known as the SaccadeNet developed by Lan et al. (2020), a model closest in approach to ours, four module classifiers are used to recognize objects. These modules are—center attentive module, the corner attentive module, the attention transitive module, and the aggregation attentive module. Each module works on identifying the main key points of the object of interest, perhaps the center, corners, attend object centers, and bounding boxes. This technique works similar to the proposed saccade approach inspired by human visual search. The drawback is that it works mainly on 2D inputs. While performing a target search in a 3D environment, the model needs to predict the next location of the camera and identify the object that the camera is looking for. To perform such search tasks in 3D space, time is one of the constraints which depends on the network design and input.

We propose a Brain Inspired Attentional Search model in 3D space (BIAS-3D) that takes the attentional glimpse instead of the entire image. The design of the model contains convolutional layers instead of fully connected layers to extract features and contains Elman and Jordan recurrence layers as well as JK-flip-flop recurrence layer (Sweta et al., 2021) instead of Long Short Term Memory (LSTM) to integrate the temporal attention history in the network. To generate the attentional glimpse, a set of concentric attention windows is used by taking the inspiration from Ba et al. (2014), Mnih et al. (2014), and Kahou et al. (2017).

The proposed model has the following brain-inspired features: (1) it has separate channels for image classification and camera movement, analogous to the "What pathway" and

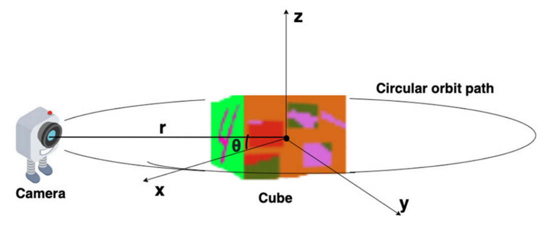

FIGURE 1
Simulated environment: "*r*" is the radius of the circular orbit around the cube or the line connecting the camera position and origin of the spherical coordinate system. Polar angle "ϕ" is assumed to be fixed at 0, and azimuth angle "θ" is varying.

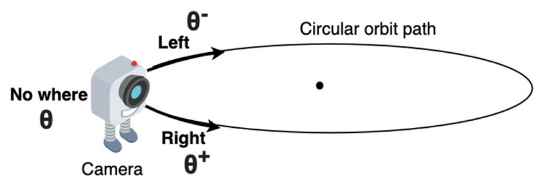

FIGURE 2
Direction of all three movements of the camera on the orbital path, supposed to be predicted by the model.

"Where pathway" in HVS; (2) it incorporates three types of recurrence connections: (a) Local recurrence connection of Elman type (Elman, 1991), (b) Global recurrence connection of Jordan type (Jordan, 1986), (c) Flip-flop neurons (Holla and Chakravarthy, 2016) that are capable of storing information for a long time. In this study, we show that the BIAS-3D is effectively able to learn task-specific strategies and identify the targets. Our simulation results successfully show that an attention-based network can be an efficient approach in dealing with target search tasks in a 3D environment, which is demonstrated by using 3D Cluttered MNIST Cube dataset, 3D Cluttered QuickDraw Cube dataset, and 3D Cluttered RGB MNIST Cube dataset.

2. The proposed approach

2.1. Environment overview

The virtual environment used in this study is created using OpenGL (Segal and Akeley, 2010) (Figure 1). The environment contains a cube placed at the origin of a spherical coordinate system and a camera placed on a circular orbit around the cube. On this orbit of radius "r," the camera revolves around the cube, always looking inwards toward the center of the cube (Figure 2). As the camera moves on the orbit, it processes the views of the cube it captures and searches for the face that has a target pattern displayed on it (Figure 3B). The possible movements of the camera on the orbit are: "move right" (θ^{+}), "move left" (θ^{-}), or "do not move" (θ; Figure 2).

2.2. Architecture overview

The architecture design of the proposed brain inspired attentional search model in 3D space (BIAS-3D) is depicted in Figure 3D. The model takes two inputs: (i) the *attentional glimpse* which consists of the contents at different resolutions and sizes of the attended region, where multiple concentric attention windows are applied to the center location of the camera view, and (ii) the *camera-position* in the form of a point on the unit circle at an angle θ or the azimuth angle of the camera position on its circular orbit. The model predicts two outputs at each timestep: (i) the next location of the camera on the orbit, and (ii) the class of the object seen in the camera view. The model consists of three parallel pipelines (Figure 3D): (i) the upper pipeline processes the class information of the object seen in the view, called the Classifier Network, (ii) the middle pipeline processes the location of the target object over the cube and predicts the next position of the camera, called the Camera Motion Network, and (iii) the lower pipeline, which incorporates the camera position into the high level features of the Classifier Network and the Camera Motion Network, is called the Camera-Position Network. Outputs of all the three pipelines are concatenated in one flattened layer which connects with a fully connected layer, and the output of the fully connected layer passes through one linear output layer and one softmax output layer in parallel. Linear output layer computes the *Q*-values corresponding to the three actions that can be taken by the camera, and softmax output layer computes the classification probabilities of the object present inside the attentional glimpse. A Deep Q-learning algorithm is applied to train the model and learn the optimal policy for camera control (Fan et al., 2020). As the model takes the sequential input, the network requires memory to store the past information of the following details: (i) the extracted features of the attentional glimpse, (ii) its corresponding location on the cube, and (iii) the camera position. For storing this input history, the model uses three recurrent neural features: the flip-flop neuron layer (Holla and Chakravarthy, 2016), Elman and Jordan recurrence layers.

2.3. The BIAS-3D

The proposed attention model is a deep neural network, which has three pipelines: Classifier Network, Camera Motion Network, and Camera-Position Network (Figure 3). The classifier network consists of three convolutional layers (Convs), three maxpool layers, and one fully connected (FC) Elman Jordan recurrence layer (FCEJ). The camera

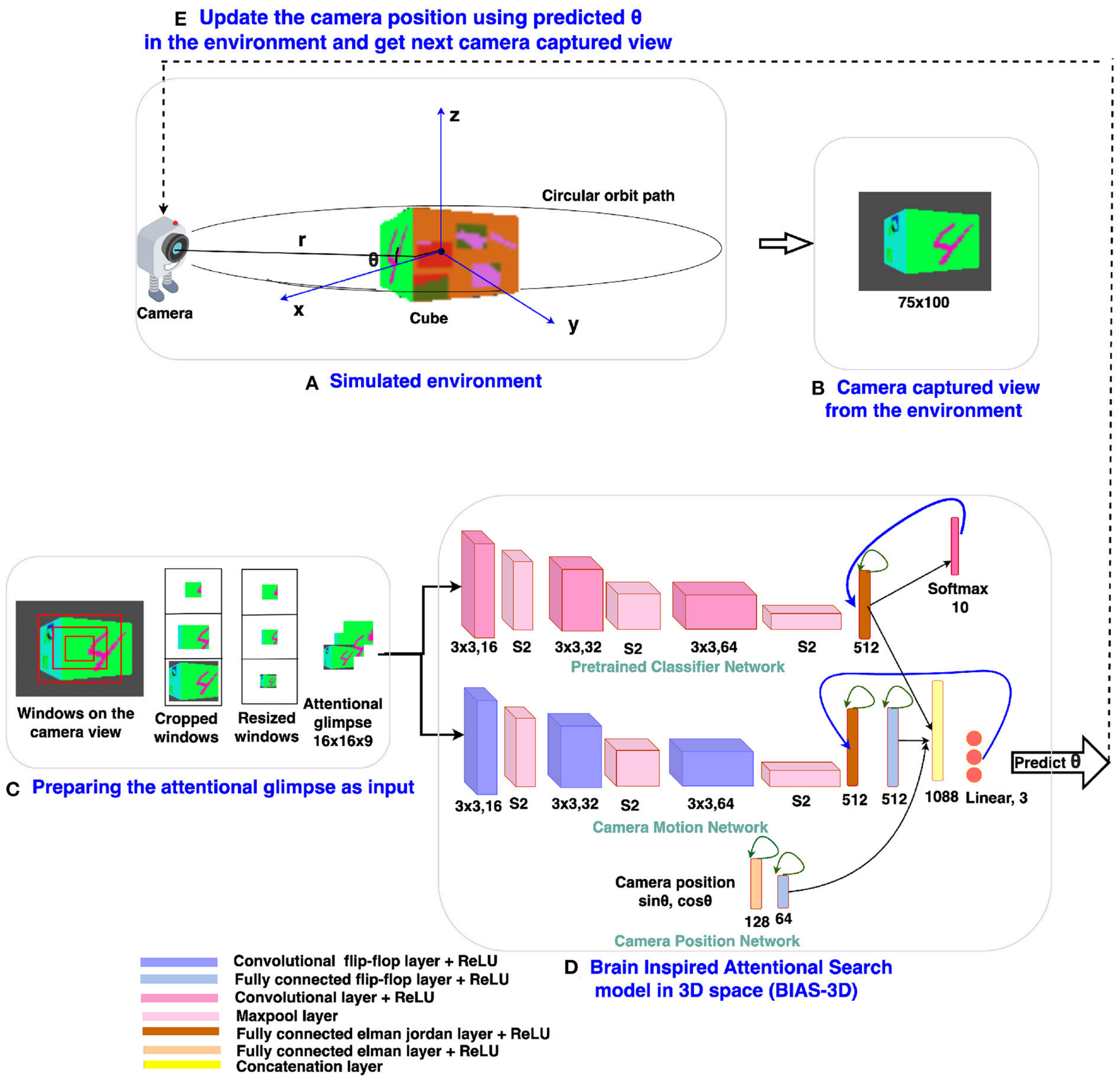
E Update the camera position using predicted θ in the environment and get next camera captured view
z
Circular orbit path
r
θ
Camera
x
Cube
y
A Simulated environment
75x100
B Camera captured view from the environment
Windows on the camera view
Cropped windows
Resized windows
Attentional glimpse 16x16x9
C Preparing the attentional glimpse as input
3x3,16
S2
3x3,32
S2
3x3,64
S2
512
Softmax 10
Pretrained Classifier Network
3x3,16
S2
3x3,32
S2
3x3,64
S2
512
512
1088
Linear, 3
Predict θ
Camera Motion Network
Camera position sinθ, cosθ
128
64
Camera Position Network
D Brain Inspired Attentional Search model in 3D space (BIAS-3D)
Convolutional flip-flop layer + ReLU
Fully connected flip-flop layer + ReLU
Convolutional layer + ReLU
Maxpool layer
Fully connected elman jordan layer + ReLU
Fully connected elman layer + ReLU
Concatenation layer

FIGURE 3
The design of the BIAS-3D: **(A)** Simulated environment with the 3D Cluttered RGB MNIST cube and the camera, **(B)** camera captures the image from the environment, **(C)** the attentional glimpse generated from the camera captured view, **(D)** the BIAS-3D predicts the class of the target and position of the camera or "θ," and **(E)** Update the camera position using the predicted "θ."

motion network consists of three convolutional flip-flop layers (ConvJKFF), three maxpool layers, one FCEJ layer, and one FC flip-flop layer (FCJKFF). The camera-position network consists of one FCEJ layer, and one FCJKFF layer; this network encodes the revolving direction of the camera. The aforementioned layers are discussed in greater detail in the following paragraphs.

Convolutional layers (Convs) are used to extract features by sharing the weights across different spatial locations. Input and output to the Conv layer are 3D tensors, called feature maps. The output feature map is calculated by convolving the input feature map with 3D linear filters. Then a bias term is added up into the convolved output. In this paper, the bold notations in all the equations stands for the matrix or the matrices. If $\mathbf{X}^{l-1}$ is the input feature map of *lth* Conv layer and $\mathbf{W}^{l}$ and $\mathbf{b}^{l}$ are filter weights and bias terms, respectively, then the output feature map $\mathbf{X}^{l}$ of *lth* layer is calculated *via* Equation-1:

$$\mathbf{X}^{l} = \mathbf{X}^{l-1}\mathbf{W}^{l} + \mathbf{b}^{l}, \tag{1}$$

$$l = 1, ..., L,$$

$$\mathbf{X}^{0} = \mathbf{I},$$

In the above equation, L is the total number of layers, $\mathbf{X}^0$ is the input image **I** to the first Conv layer. The output feature maps from each Conv layer are passed through a non-linear ReLU activation function (Nair and Hinton, 2010) (equation-2).

$$\mathbf{f}(\mathbf{X}) = max(0, \mathbf{X}) \tag{2}$$

The output feature maps from the activation function, are normalized using local response normalization (LRN) (Krizhevsky et al., 2012). LRN normalizes the feature maps within the channels and is a form of lateral inhibition (Equation 3).

$$\mathbf{N}^f_{x,y} = \mathbf{X}^f_{x,y} / \left(k + \alpha \sum_{j=max(0,f-c/2)}^{min(C-1,f+c/2)} (\mathbf{X}^j_{x,y})^2 \right)^{\beta} \tag{3}$$

where $\mathbf{X}(x, y)$ and $\mathbf{N}(x, y)$ are the pixel values at (x, y) position before and after normalization, respectively, f denotes the filter. C stands for the total number of channels. The constants k, α, β, and c are hyperparameters. k is used to avoid "division by zero," α is a normalization constant, while β is used as a contrasting constant. The constant c is used to define the length of the neighborhood, that is, the number of consecutive pixel values need to be considered while calculating the normalization. $(k, \alpha, \beta, c) = (0, 1, 1, C)$ case is considered as the standard normalization. Normalized features from the Conv layer are passed through the maxpool layer (Scherer et al., 2010). Several convolutional layers and pooling layers are assembled alternately across depth in the first three Conv or ConvJKFF layers in both classifier and camera motion networks (Figure 3D).

To implement the Elman recurrence layer (Elman, 1991), the output vector of the FC layer at time "$t - 1$" is stored in a context layer and the content of the context layer is fed back to the same FC layer at time "t," named as FC Elman recurrence layer which is a short range storage connection. The Elman recurrence layer is implemented only in the first FC layer of all three networks. Similarly, to implement the Jordan recurrence layer (Jordan, 1986), the output vector of the last FC layer at time step "$t - 1$" is stored in a context layer and this context layer is fed back to the first FC layer at time step "t" in their corresponding pipeline, named as FC Jordan recurrence layer which is a long-range storage connection. In this way, the first FC layer in Classifier and Camera Motion Networks has both Elman and Jordan recurrences; so we call this layer a FCEJ layer. The computation of FCEJ is shown in the following (Equation 4)

$$\mathbf{X}^l_t = f\left(\mathbf{X}^{l-1}_t \mathbf{W}^{l-1,l} + \mathbf{X}^l_{t-1} \mathbf{W}^{l,l} + \mathbf{X}^L_{t-1} \mathbf{W}^{L,l} + \mathbf{b}_l\right) \tag{4}$$

In Equation (4), $\mathbf{X}^{l-1}_t$ is the output of the $l-1th$ layer at time "t" and going as input to the lth layer at time "t" (FC layer). $\mathbf{X}^l_{t-1}$ is the output of the lth layer at time "$t - 1$" and going as input to the same lth layer at time "t" (Elman recurrence layer). $\mathbf{X}^L_{t-1}$ is the output of the Lth layer at time "$t - 1$" and going as input to the lth layer at time "t" (Jordan recurrence layer). $\mathbf{W}'s$ and b are the corresponding weights and bias, respectively. f is the ReLU activation function.

Memory of the past information in the layers of the proposed network is stored using a third mechanism—the flip-flop neurons (Holla and Chakravarthy, 2016). A flip-flop is a digital electronic circuit to store state information. There are four types of digital implementations of flip-flops: D flip-flops, Toggle flip-flops, SR flip-flops, and JK flip-flops (Roth et al., 2020). In the proposed network, JK flip-flop neurons are used in place of LSTM neurons because of the performance advantage shown in Holla and Chakravarthy (2016) and Sweta et al. (2021). In both of these papers, the experiments conducted on the sequential data shows that flip-flop neurons outperform the LSTM neurons, using only half the number of training parameters in comparison to LSTM. Likewise, to get the advantage of fewer parameters and better performance, in the current study we used the JK flip-flop neuron. The JK flip-flop neuron uses two gating variables with "**J** and **K**" nodes, whereas LSTM uses four gating variables. In this paper, the term flip-flop will be used to refer to JK-flip-flop. Furthermore, the flip-flop neurons are considered similar to the UP/DOWN neurons found in the prefrontal cortex (PFC), responsible for working memory (Gruber et al., 2006).

In the proposed model, the flip-flop layer is designed in two ways: flip-flop neurons in convolutional layer (named as "convolutional flip-flop layer" or ConvJKFF), and flip-flop neurons in the FC layer (named as "fully connected flip-flop layer" or FCJKFF). Training rules of these flip-flop neurons in the network were also developed. The two gate outputs "**J**" and "**K**," the hidden state of the JK flip-flops, and the final flip-flops output are computed by using Equations (5–7, respectively) below.

$$\mathbf{J} = \sigma\left(In_t \mathbf{W}_j\right), \mathbf{K} = \sigma\left(In_t \mathbf{W}_k\right) \tag{5}$$

$$\mathbf{H}_t = \mathbf{J}.\left(\mathbf{1} - \mathbf{H}_{t-1}\right) + (\mathbf{1} - \mathbf{K}).\mathbf{H}_{t-1} \tag{6}$$

$$\mathbf{O}_t = \tanh(\mathbf{H}_t \mathbf{W}_{out}) \tag{7}$$

In Equation (6), "." stands for the pointwise multiplication. $In_t = (\mathbf{X}_t; \mathbf{H}_t)$ is the input to the flip-flop layer, where $\mathbf{X}_t$ is the output from the previous layer and $\mathbf{H}_t$ is the hidden state at time "t," which initialize with ones at time 0. **1** is a matrix of ones. **J** and **K** are the gate variables, which has weight parameters $\mathbf{W}_j$ and, $\mathbf{W}_k$, respectively. $\mathbf{O}_t$ is the output of the flip-flop layer at time 't'. To train the flip-flop neurons, the partial derivatives w.r.t **J** and **K** were used to backpropagate the corresponding **J** and **K** nodes (Equation 8).

$$\frac{\partial \mathbf{H}_t}{\partial \mathbf{J}} = 1 - \mathbf{H}_{t-1}; \frac{\partial \mathbf{H}_t}{\partial \mathbf{K}} = -\mathbf{H}_{t-1} \tag{8}$$

2.4. Implementation detail

2.4.1. Camera motion network

The camera motion network takes the attentional glimpse of size $h \times w \times a$ as input, where "h" is the height, "w" is the width, and "a" is the number of the cropped attention windows. Here, the number of attention windows is chosen to be 3 (i.e., $a = 3$). The size of one attention window is twice the previous attention window's size. Similar multiscale concentric attention windows were used in other models (Mnih et al., 2014; Haque et al., 2016; Shaikh et al., 2019). All the attention windows, except the smallest one, get resized to the size of the smallest attention window. For example, to generate the attentional glimpse where $h = 16$, $w = 16$, and $a = 3$ from location $y = 35$ and $x = 50$ in the given image of size 75×100, the first, second, and third attention windows are cropped out of size 16×16, 32×32, and 50×50 from pixel location $(y, x) = (27$ to $43, 42$ to $58)$, $(y, x) = (19$ to $51, 34$ to $66)$, and $(y, x) = (10$ to $60, 25$ to $75)$, respectively. The second and third cropped attention windows are resized to the size of the first cropped attention window, which is 16×16. After resizing, all the three attention windows are stacked together, which finally becomes an attentional glimpse of size $16 \times 16 \times 3$. This type of attentional glimpse having a size of $h \times w \times a$ shown in Figure 3C is passed to the first ConvJKFF layer of 16 kernels, each of size 3×3, of the classifier network (shown in the top pipeline of the BIAS-3D in Figure 3D). The spatial dimension of the features generated from the first ConvJKFF layer is $h \times w \times 16$, which are normalized using LRN, and passed into ReLU activation function. Output from ReLU activation function is passed to the maxpool layer with a window of size 2×2 and stride by 2, which translates the feature's spatial dimensions into $h/2 \times w/2 \times 16$. The translated feature maps are passed as input to the second ConvJKFF layer of 32 kernels, each of size 3×3, to extract the higher level features of size $h/2 \times w/2 \times 32$. Then, similar to the previous layer, features generated from the second ConvJKFF layer are passed through the LRN layer, ReLU activation function, and maxpool layer with a window of size 2×2 and stride 2. After passing into the maxpool layer, feature maps of size $h/4 \times w/4 \times 32$ are generated, which are further converted into a flattened layer to reshape the 3D features into 1D vectors. The flattened vectors are passed through one FCEJ layer of 512 neurons, which is followed by one FCJKFF layer of 512 neurons. Output from the FCJKFF layer of the camera motion network is concatenated with the output vectors of the last layer of the other two channels.

2.4.2. Classifier network

The classifier network gets the same attentional glimpse as input which has been passed to the camera motion network. This network predicts the class of the object present in the attentional glimpse. The object present in the attentional glimpse may belong to one of the "$n + 1$" classes, where "n" classes are the object or target class and one is the nontarget or clutter class. The network consists of 3 Conv layers followed by one FCEJ layer. The first Conv layer of 16 kernels of size 3×3 generates the feature maps of spatial dimension $h \times w \times 16$. Generated features are passed through the LRN layer and ReLU activation function. After this, the maxpool layer with a window of size 2×2 and stride by 2 has been applied to the output of ReLU activation function, which gives the feature maps of spatial dimension $h/2 \times w/2 \times 16$. Then, the feature maps are passed through a second Conv layer of 32 kernels, each of size 3×3, LRN layer, ReLU activation function, and maxpool layer with a window of size 2×2 and stride by 2. Feature maps of spatial dimension $h/4 \times w/4 \times 32$ are passed through a third Conv layer of 64 kernels each of size 3×3 with ReLU activation function, which further generates the feature maps of size $h/4 \times w/4 \times 64$. Then the flattened layer reshapes the 3D tensor of feature maps into vectors, and these vectors are input to the FCEJ layer of 512 neurons. The output of the FCEJ layer gets concatenated with the output vectors of the last layer of the camera motion network and the camera position network.

2.4.3. Camera position network

The camera's position in the environment is inferred from the spherical coordinates, where the camera is assumed, as described before, on a circular object centered on the origin, and the center of the cube is located at the origin. The camera's position is defined by three variables: ("r," "θ," "ϕ"), where "$r = R$" is the radius of the circular orbit of the camera or line connecting the camera point and the origin of the spherical coordinate system, "θ" is the azimuth angle and "ϕ" is the polar angle of the spherical coordinate system. In the current simulated environment, the camera moves only in one degree of freedom, that is "θ." Therefore, "$r = R$" and "$\phi = 0$" are considered to be constant. Only "θ" varies as the camera moves on a circular orbital path around the origin of the spherical coordinate system or the cube. In the camera position network, sinusoidal functions of "θ" are passed as input to the first FC Elman (FCE) layer having 128 neurons, followed by one FCJKFF of 64 neurons. Output from the FCJKFF layer is concatenated to the output vectors of the last layer of the classifier network and the camera motion network.

Outputs from three pipelines are concatenated in one common flattened layer, which further connects with two output layers in parallel. One output layer with linear activation function is responsible to predict one direction out of the three considered directions in which the camera will move on the orbit to look and locate the target face present in the given cube. The other output layer with softmax activation function is responsible to predict the class of the object seen on the view of the camera.

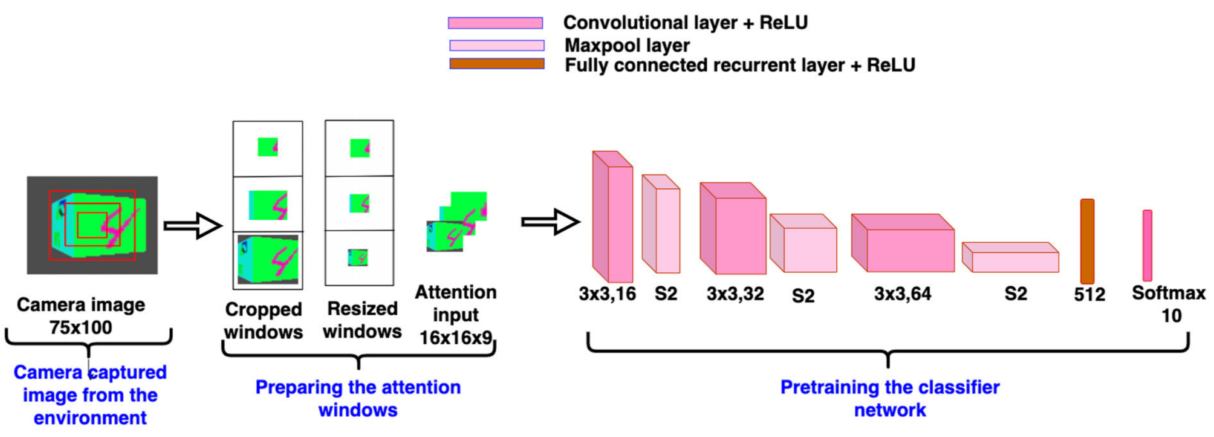
Convolutional layer + ReLU
Maxpool layer
Fully connected recurrent layer + ReLU
Camera image 75x100
Camera captured image from the environment
Cropped windows
Resized windows
Attention input 16x16x9
Preparing the attention windows
3x3,16
S2
3x3,32
S2
3x3,64
S2
512
Softmax 10
Pretraining the classifier network

FIGURE 4
Pretrained classifier network.

2.4.4. Training and testing

Tensorflow framework is used to implement the proposed attention model. Xavier's initialization (Glorot and Bengio, 2010; Equation 9) with random normal distribution is used to initialize the weights for each layer of the three networks. The Xavier initialization is able to avoid the exploding or vanishing gradients (Bengio et al., 2001) problem by fixing the variance of the activations across each layer as the same.

$$\mathbf{W}^{l} = \mathbf{N}\left(0, \frac{2}{m^{l-1} + m^{l}}\right) \tag{9}$$

where, **N** stands for the normal distribution. m^{l-1} and m^{l} is the number of neurons in the previous layer and current layer, respectively. $\mathbf{W}^{l}$ denotes the weights at *lth* layer with Xavier initialization.

Before training the model, the classifier network is pretrained on the camera captured views. To pretrain the classifier network, we collect views of the simulated environment by explicitly revolving the camera from 0 to 360°, where 0° is assumed to be exactly at the front of the face containing the target object. Advancing in steps of 9 degrees over the range of 0–360°, a total of 40 views is collected for each cube in the dataset. Views between −45 to +45 range are labeled as one of the "*n* classes" and views between +46 to +180 and −46 to −180 range are labeled as "background class." Therefore, the total number of classes present in the dataset is $n + 1$. To make the views data uniform, the same number of views of the background class are chosen randomly as the number of views of the other class. The classifier network is pretrained on such views of targets and background or nontarget class. We assume that the camera's focus is always fixed in the center of the view. Therefore, we create a glimpse of three concentric windows from the center location of the camera view. Detailed architecture of the pretraining classifier network is shown in Figure 4. The classifier network without recurrent layers in the BIAS-3D is pretrained on the glimpse of the camera views. Total loss of the model is calculated in two parts: one is classification loss, calculated using the cross-entropy loss function (Goodfellow et al., 2016) and the other one is prediction loss, calculated using mean square error of temporal difference (Sutton and Barto, 1998). Equations of the both loss functions are shown in Equations (10) and (13).

$$\mathbf{L}_{ce} = -\sum_{i=1}^{n+1} \mathbf{d}_i \log\left(\mathbf{p}_i\right) \tag{10}$$

In Equation (10), $\mathbf{d}_i$ denotes the desired classification probability and $\mathbf{p}_i$ denotes the predicted classification probability of *ith* class. "$n + 1$" is the total number of classes that are present in the dataset including background class. Here, the camera is assumed as an agent and the agent learns a defined policy of the reward function (Equation 11) (Armstrong and Murlis, 2007). When the agent is in the current state, *Q*-values of all three actions are predicted by passing the information of the current state (like the attention input and the θ value of the camera) into the deep neural network. Based on the predicted *Q*-value of all the actions in the current state, the agent makes an action decision using a softmax action selection policy (Abed-alguni, 2018). In this policy, the predicted *Q*-values are passed through a softmax activation function to produce the action probabilities. The action with the highest probability is selected and performed by the agent in the current state of the environment. After performing the action, the current state is updated to the next state and then the agent receives a reward, either "1" or "0" depending on the reward policy shown in Equation (11).

$$\mathbf{r} = \begin{cases} 1 \ \arg\max_{i\in n}(\mathbf{p_i}) == \arg\max_{i\in n}(\mathbf{d_i}) \\ \quad max(\mathbf{p}) \geq \lambda \\ \\ 0 \ \ otherwise \end{cases} \tag{11}$$

Apart from the softmax action selection policy, we used a race model (Rowe et al., 2010) which ensures that the selected action is correct. Race models have been applied in many behavioral, perceptual, and oculomotor decisions and such decisions are based on trial-to-trial modifications in a race among all the responses (Carpenter and McDonald, 2007). Race model works based on two neurophysiological evidences to show the relatedness. Firstly, if monkeys are trained to make their decision on coherently moving direction of dots, accumulating neuronal activity is formed that mirrors the decision even when there is no coherent motion. Here, both choices are equally rewarded (Churchland et al., 2008). Secondly, the decision threshold is considered constant for a selected action, regardless of its being a specifically cued action (Roitman and Shadlen, 2002). We have taken the motivation to apply the race model based on the second evidence. The action predicted by the network is the action which crosses the threshold, λ, first and if the action predicted is correct, the agent gets reward "1"; it otherwise gets reward "0."

The Q-values of the actions in the next step are estimated by passing the next state information into the target network, where the target network is the separate copy of the networks of the model. Target Q-value is calculated by adding the current state reward and maximum of the next state Q-values multiplied with a discount factor γ. Discount factor defines how much the current state Q-value depends on the future reward. Now, the temporal difference (TD) is calculated by calculating the difference between the target Q-value and the predicted Q-value (Equation 12).

$$\mathbf{TD} = \left(\mathbf{r} + \gamma * \mathbf{Q_{max}}\left(S_{t+1}\right)\right) - \mathbf{Q}\left(S_t\right) \tag{12}$$

$$\mathbf{L}_{mse} = \frac{1}{n}\sum_{i=0}^{n} \mathbf{TD}^2 \tag{13}$$

$$\mathbf{l}_{total} = \mathbf{L}_{ce} + \mathbf{L}_{mse} \tag{14}$$

where, $\mathbf{r}$ is the reward which the agent gets while going from the state $\mathbf{S}_t$ to the state $\mathbf{S}_{t+1}$. $\mathbf{Q}\left(S_{t+1}\right)$ and $\mathbf{Q}\left(S_t\right)$ is the Q-value of the state $\mathbf{S}_{t+1}$ and, $\mathbf{S}_t$, respectively. γ is the discount factor. Then these two losses, the cross-entropy loss of the classifier network (Equation 10), $\mathbf{L}_{ce}$ and the mean square error of temporal difference of the camera motion network (Equation 13), $\mathbf{L}_{mse}$, are added up to get the total loss (Equation 14). The total loss is back propagated into the network (Voleti, 2021). The network parameters are updated by using the mini-batch Adam optimizer (Kingma and Ba, 2014). L2 regularization (Kratsios and Hyndman, 2020) is used to avoid the overfitting problem of the network. During inference, the camera starts from a random location and moves toward the target face of the cube. Once it finds the target face, the camera continues to fixate around that face. The number of trainable parameters of the model are 2, 668, 362. The model achieves a processing speed of 0.0001 s per input image on a workstation with an NVIDIA GeForce GTX 1, 080Ti 11 GB GPU, i7-8700 CPU @ 3.20 GHz 3.19 GHz, 64-bit operating system, and 32.0 GB RAM.

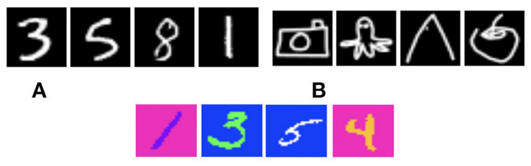

FIGURE 5
A sample of image datasets are shown here. **(A)** 28 × 28 MNIST digits. **(B)** 28 × 28 QuickDraw. **(C)** 28 × 28 RGB MNIST digits.

3. Simulation results

We evaluate our model on "painted cube" data, where each cube has a target object on one vertical face and nontarget objects on the other three vertical faces. The model is supposed to move the camera around the cube on a circular orbit and search the target object image present on one of the four vertical faces of the cube. For target object image, we used image datasets. Totally three 3D Cluttered Cube datasets were considered in the experiment. The cube datasets were generated using their related image data. Grayscale MNIST digit image dataset, QuickDraw image dataset, and RGB MNIST digit image dataset (Samples are shown in Figures 5A–C, respectively) were used to generate cube datasets like 3D Cluttered Grayscale MNIST Cube dataset, 3D Cluttered QuickDraw Cube dataset, and 3D Cluttered RGB MNIST Cube dataset respectively. The first two of these are cube datasets with grayscale images, and the last one is a cube dataset with RGB images. Based on the grayscale and RGB cube datasets, we designed the experiments in two parts: one part of the experiment shows the target search capability of the proposed model on the cubes which has all four vertical faces of grayscale images (called grayscale cubes) and the other part of the experiment shows the target search capability of the model on the cubes which has all vertical faces of RGB images (called RGB cubes).

3.1. Searching on grayscale cubes

In the first part of the experiment, we evaluated our model on two datasets of Grayscale cubes. For that, we used two different datasets of grayscale images: MNIST handwritten digits (LeCun et al., 1998) and QuickDraw (Jongejan et al., 2016). Both datasets with 10 different classes contain 48, 000 examples in the training set, 12, 000 examples in the validation set, and 10, 000

TABLE 1 Accuracy on testing set of all three datasets.

Dataset	Testing accuracy (%)
3D cluttered grayscale MNIST cube dataset	95.6
3D cluttered grayscale QuickDraw cube dataset	83
3D cluttered RGB MNIST cube dataset	91.5

examples in the testing set. We have generated a 3D Cluttered MNIST Cube dataset using MNIST dataset. To generate such a cube dataset, each of the cubes were created with a 28 × 28 MNIST digit image (target) on one vertical face and 28 × 28 a random clutter image (nontarget) on the other three vertical faces. In this experiment, the bottom and top faces of the cube are not considered for searching. Similarly, a 3D Cluttered QuickDraw Cube dataset was generated using QuickDraw image dataset.

Once the cube datasets are generated, we place the cube in the environment in such a way that the center of the cube is at the origin of the spherical coordinate system. Then the camera is placed at a random value of azimuth angle "θ" at initial time ($t = 0$). The polar angle "ϕ," and radius "r" are set to 0, and 2.5, respectively. The camera placed at (r, θ, ϕ) captures the view of size 75 × 100. Then a glimpse is extracted from the center location of the captured view. To extract the glimpse, three concentric windows of size 16 × 16, 32 × 32, and 50 × 50 are cropped out from the center of the view. After cropping out, windows of size 32 × 32 and 50 × 50 are resized into the size of 16 × 16. Then resized windows with the smallest window of size 16 × 16 are arranged together across depth to generate an attentional glimpse of dimension 16 × 16 × 3. Since the image size in the QuickDraw image dataset is same as the image size in the MNIST dataset, the same dimensions of the camera view and attentional glimpse were considered in case of the 3D Cluttered QuickDraw Cube dataset.

The proposed model takes the attentional glimpse of size 16 × 16 × 3 from the center location of the image view of the camera of size 75 × 100. Achieved accuracy on both grayscale datasets are listed in Table 1. The results of the camera's movement predicted by our model in the testing set are shown in Figures 6–9. In this figure, images of the camera view of dimension 75 × 100 are shown in one row and their corresponding plots for predicted classification probabilities for that view (dotted dashed-blue curve) and ground truth target classification probabilities (green curve) are shown in the row just below. At the bottom of the plots, timestep and ground truth target class labels are denoted by using variables "t" and "c," respectively. In the row of images of the camera view, three concentric red windows depict the glimpse.

The model has the ability to move the camera to the position where the target face of the cube is visible from the camera. For example, in Figure 6, the class of digit 2 in the fourth image of the first row has the view of nontarget or clutter face at timestep $t = 0$ and its corresponding predicted classification probabilities shown in the plot just below that image is low for all classes. But at timestep $t = 1$ (θ is decided by the model), the camera has moved toward the right and has seen some part of the target face that has the digit 2. At the same time, the highest of the predicted classification probabilities is for digit 2. The camera again moved to the right at timestep $t = 2$, where an adequate part of the digit 2 on the cube face is visible (first image in fourth row of Figure 6) and therefore, the maximum value of predicted classification probabilities is close to 1 for digit 2, which crosses a testing threshold of value 0.95. Similarly, for the other digits, the camera starts moving appropriately, searching for the target. The camera stops moving when the maximum value of the predicted classification probabilities crosses the testing threshold. The testing threshold is set based on the feature complexity of the image datasets.

In the case of the 3D Cluttered QuickDraw Cube dataset, we can observe the same search behavior of the camera. For example, in Figure 8, class 5 (bicycle) in the third image of the tenth row has the camera view showing non-target objects on the cube face at timestep $t = 0$ and its corresponding predicted classification probabilities shown in the plot just below the that image is low for all classes. At the next timestep ($t = 1$), the camera has moved to the left and the camera continues to move in the left direction 3 more times even though the target is not visible. At timestep $t = 4$, a very small part of the bicycle is visible (second image in the thirteenth row of Figure 9) and at this time the classification probability for class 5 or bicycle becomes the highest. The camera stops moving once the maximum value of the predicted classification probabilities crosses a testing threshold of value 0.85.

3.2. Searching on RGB cubes

In the second part of the experiment, we evaluated our model on RGB cubes to investigate that the model is able to search for the target object on the cube face even in the case of color images. To this end, we generated a cube dataset using RGB MNIST image dataset. Here, we first create the RGB MNIST digit image dataset by assigning different colors to the digits and the background of the images available in Grayscale MNIST digit image dataset (LeCun et al., 1998). Our created RGB MNIST handwriting digit dataset is available in this link. The dataset with 10 different classes contains 48,000 examples in the training set, 12,000 examples in the validation set, and 10,000 examples in the testing set. Once the image dataset is ready, we generate a 3D Cluttered RGB MNIST Cube dataset using RGB MNIST image dataset. To generate a 3D Cluttered RGB MNIST Cube dataset, each of the cubes is created with a 28 × 28 × 3 RGB MNIST image (target) on one vertical face and

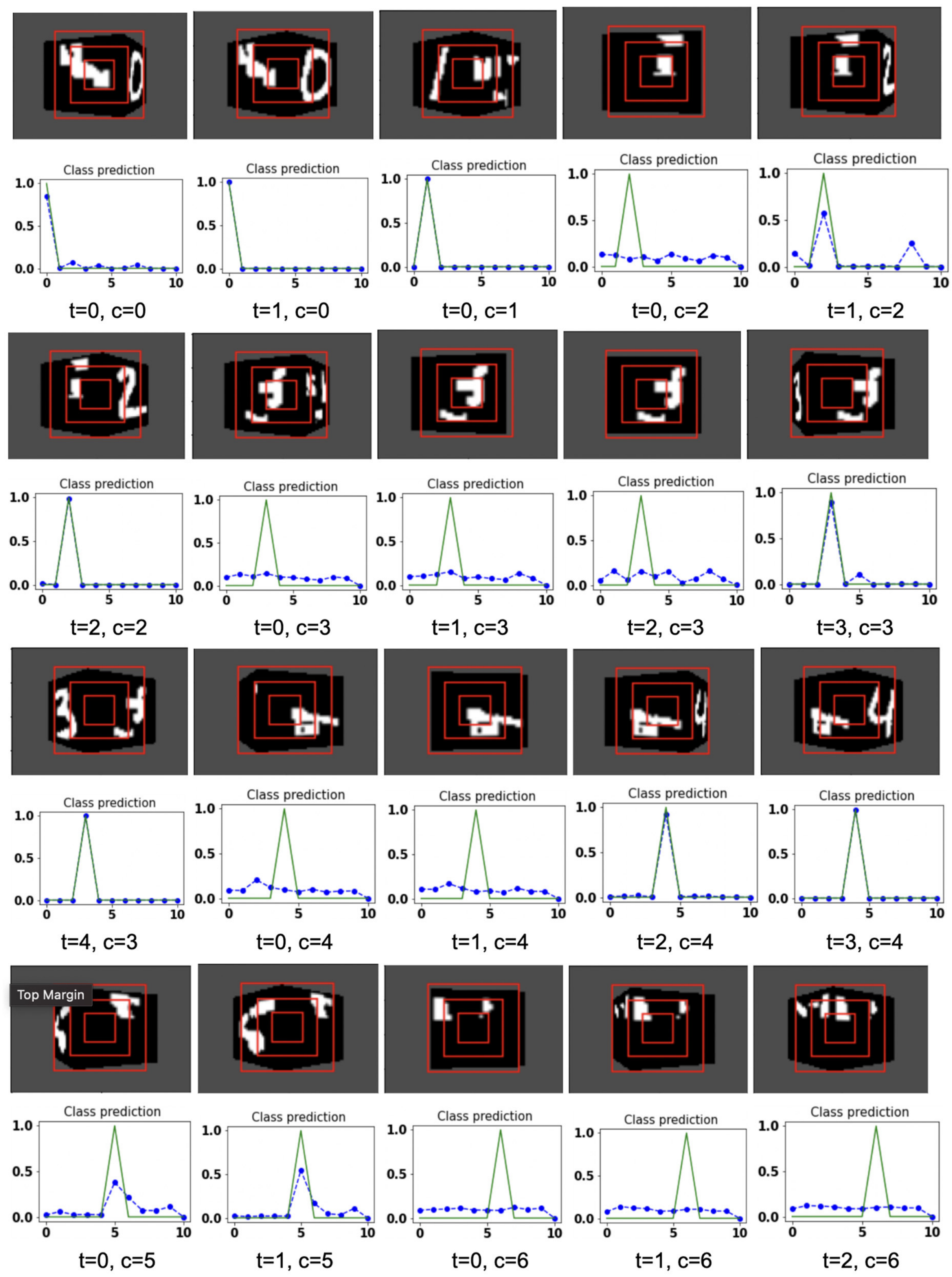

FIGURE 6
Illustrates the camera movements around the cube to search the target face in the view of size 75 × 100, predicted by our model in 3D Cluttered Grayscale MNIST Cube dataset. For each class at time *t*, there is a movement (shown in the row of camera view images) and corresponding classification probabilities (shown in the row of plots). In the row of camera view images, the three concentric red windows depict the glimpse at the center of the view image. In the plot corresponding with the above view image, the green curve is the desired classification probabilities and the dotted dashed-blue curve is the predicted classification probabilities at time *t*.

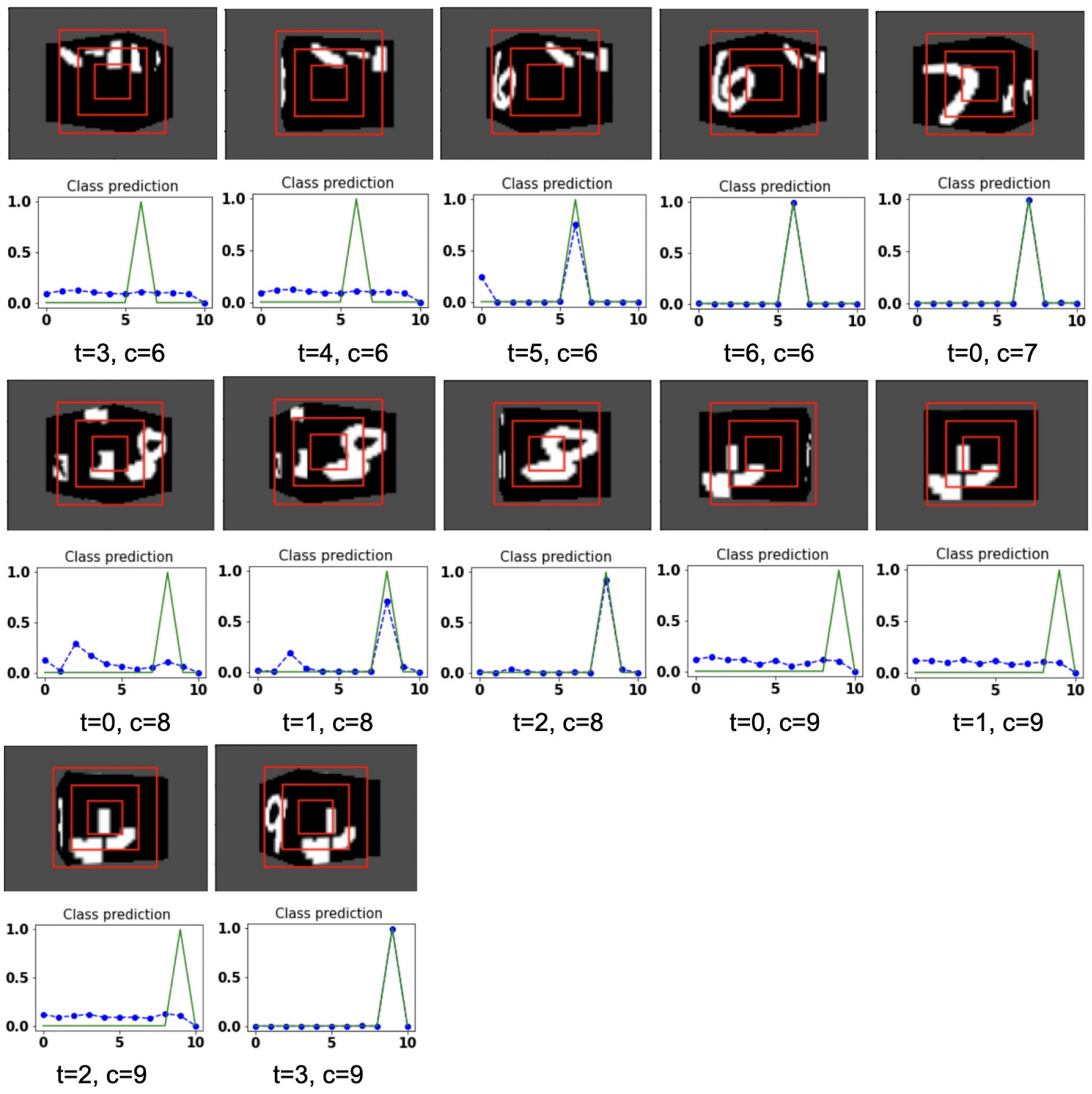

FIGURE 7
Illustrates the camera movements around the cube to search the target face in the view of size 75 × 100, predicted by our model in 3D Cluttered Grayscale MNIST Cube dataset. For each class at time *t*, there is a movement (shown in the row of camera view images) and corresponding classification probabilities (shown in the row of plots). In the row of camera view images, the three concentric red windows depict the glimpse at the center of the view image. In the plot corresponding with the above view image, the green curve is the desired classification probabilities and the dotted dashed-blue curve is the predicted classification probabilities at time *t*.

$28 \times 28 \times 3$ random clutter image (non-target) on the other three vertical faces.

The model is evaluated by placing the RGB cube in the environment in the same way of grayscale cube datasets. The camera captures the view of size 75 × 100 of the 3D Cluttered RGB MNIST Cube. The camera extracts the glimpse from the center of the captured view. To extract the glimpse, three concentric windows of size 16 × 16, 32 × 32, and 50 × 50 are cropped out from the center of the view to generate an attentional glimpse of size 16 × 16 × 9. The proposed model takes the attentional glimpse of size 16 × 16 × 9 from the center location of the image view of the camera of size 75 × 100 in case of 3D Cluttered RGB MNIST Cube dataset. The achieved accuracy on the RGB cube dataset is listed in Table 1. The results of the camera's movement predicted by our attention model in the testing set are shown

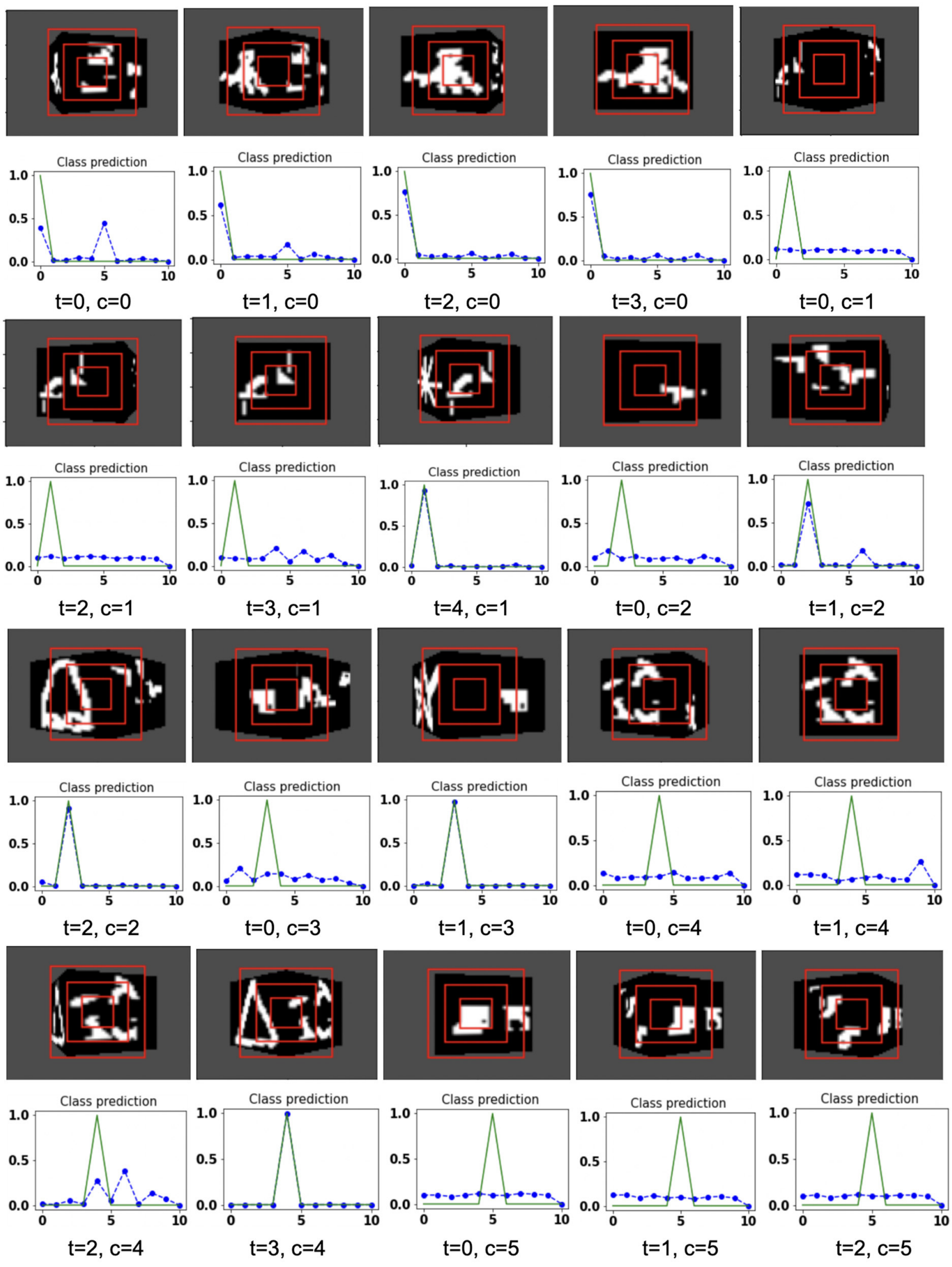

FIGURE 8
Illustrates the camera movements around the cube to search the target face in the view of size 75 × 100, predicted by our model in 3D Cluttered Grayscale QuickDraw Cube dataset. For each class at time *t*, there is a movement (shown in the row of camera view images) and corresponding classification probabilities (shown in the row of plots). In the row of camera view images, the three concentric red windows depict the glimpse at the center of the view image. In the plot corresponding with the above view image, the green curve is the desired classification probabilities and the dotted dashed-blue curve is the predicted classification probabilities at time *t*.

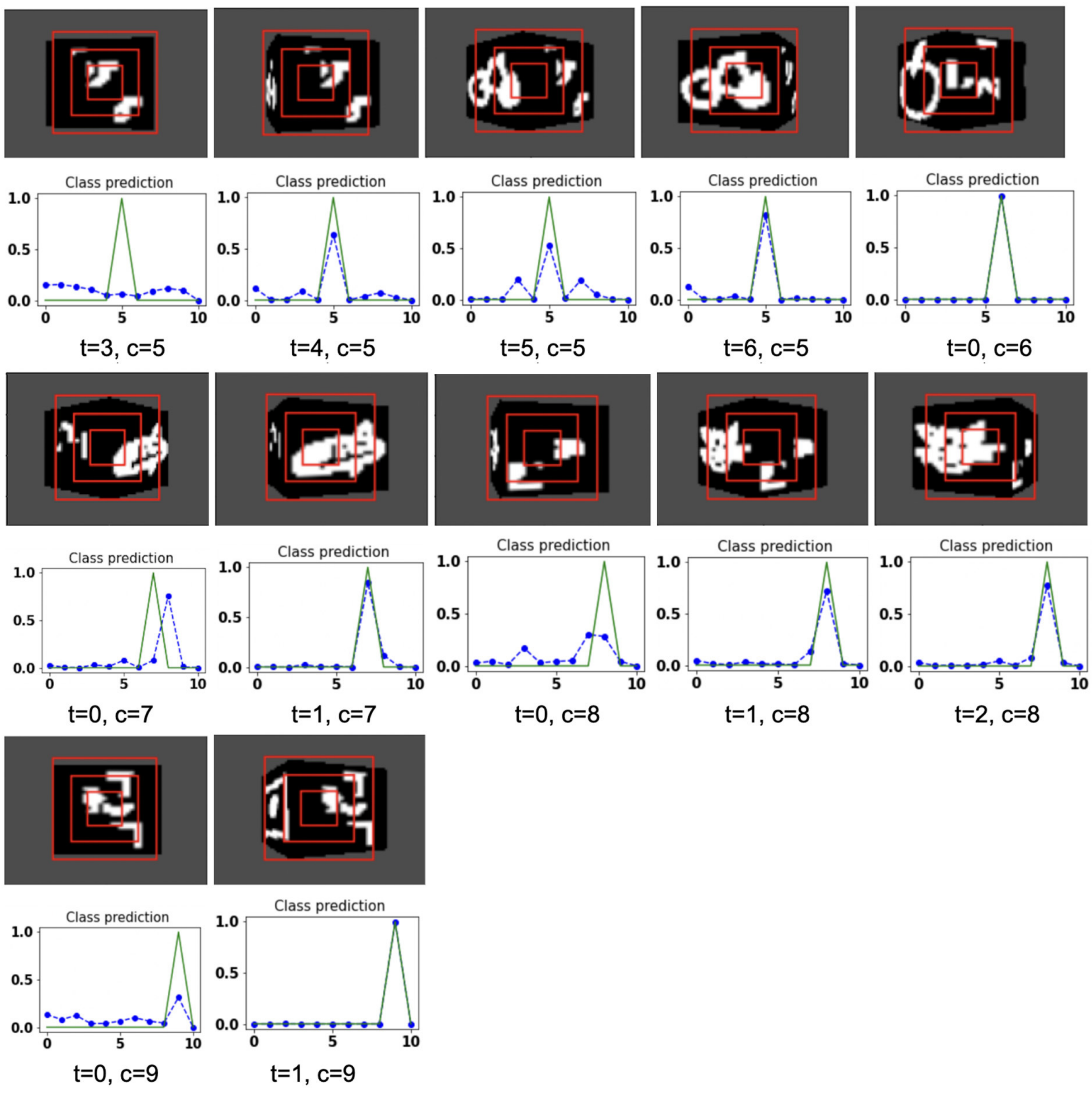

FIGURE 9
Illustrates the camera movements around the cube to search the target face in the view of size 75 × 100, predicted by our model in 3D Cluttered Grayscale QuickDraw Cube dataset. For each class at time *t*, there is a movement (shown in the row of camera view images) and corresponding classification probabilities (shown in the row of plots). In the row of camera view images, the three concentric red windows depict the glimpse at the center of the view image. In the plot corresponding with the above view image, the green curve is the desired classification probabilities and the dotted dashed-blue curve is the predicted classification probabilities at time *t*.

in Figures 10–12. Plots of accuracy, and reward vs. epoch, are shown in Figure 13.

The hyperparameters of the model are tuned and chosen as follows: 0.0001 learning rate, 0.43 discount factor, 0.85 threshold (λ), and 0.1 regularization factor (β) with the best performance in case of 3D Cluttered Grayscale MNIST Cube dataset. The model explores the actions with ϵ equal to 0.99 and the exploration gets reduced by a decay factor of 0.999 while training. The minimum value of ϵ is set with 0.1. The model is trained for 25 epochs and 50 timesteps per cube, in case of 3D Cluttered Grayscale MNIST Cube dataset. In the case of the 3D Cluttered QuickDraw Cube dataset, the model is trained for 20 epochs and 50 timesteps. During the inference, time-steps are varied depending upon the classification probabilities. Prediction is considered to be done as soon as the maximum value of the

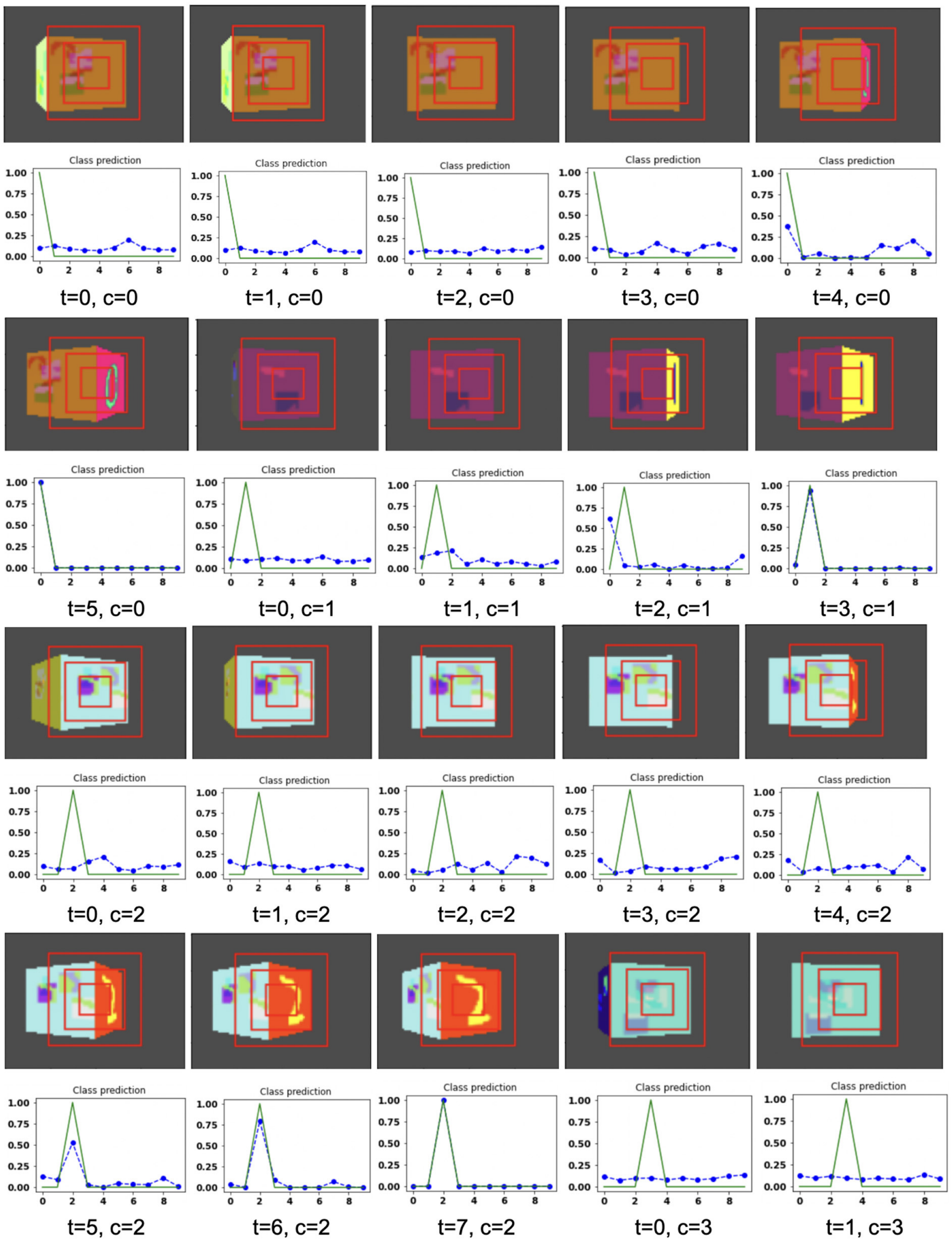

FIGURE 10
Illustrates the camera movements around the cube to search the target face in the view of size 75 × 100, predicted by our model in 3D Cluttered RGB MNIST Cube dataset. For each class at time *t*, there is a movement (shown in the row of camera view images) and corresponding classification probabilities (shown in the row of plots). In the row of camera view images, the three concentric red windows depict the glimpse at the center of the view image. In the plot corresponding with the above view image, the green curve is the desired classification probabilities and the dotted dashed-blue curve is the predicted classification probabilities at time *t*.

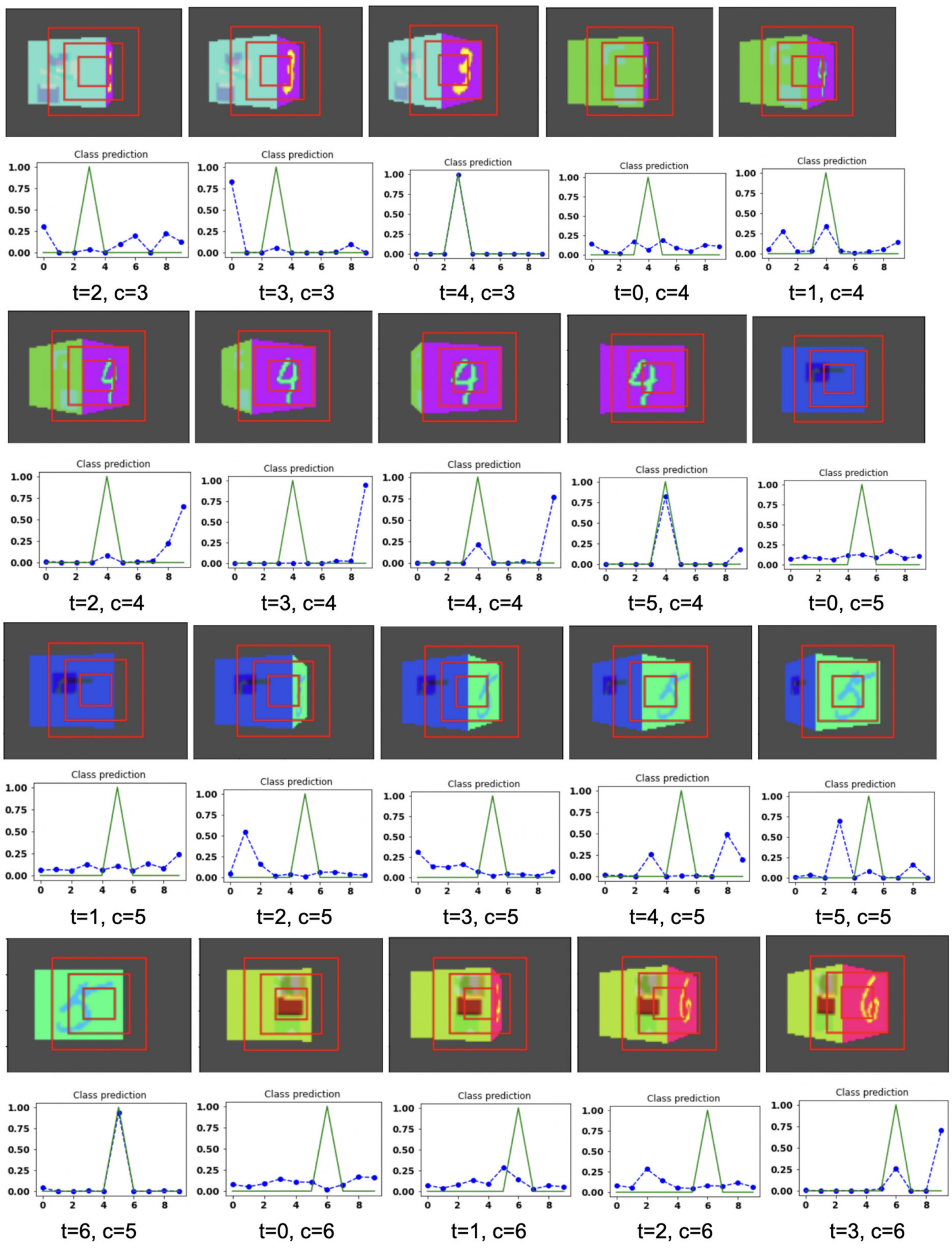

FIGURE 11
Illustrates the camera movements around the cube to search the target face in the view of size 75 × 100, predicted by our model in 3D Cluttered RGB MNIST Cube dataset. For each class at time *t*, there is a movement (shown in the row of camera view images) and corresponding classification probabilities (shown in the row of plots). In the row of camera view images, the three concentric red windows depict the glimpse at the center of the view image. In the plot corresponding with the above view image, the green curve is the desired classification probabilities and the dotted dashed-blue curve is the predicted classification probabilities at time *t*.

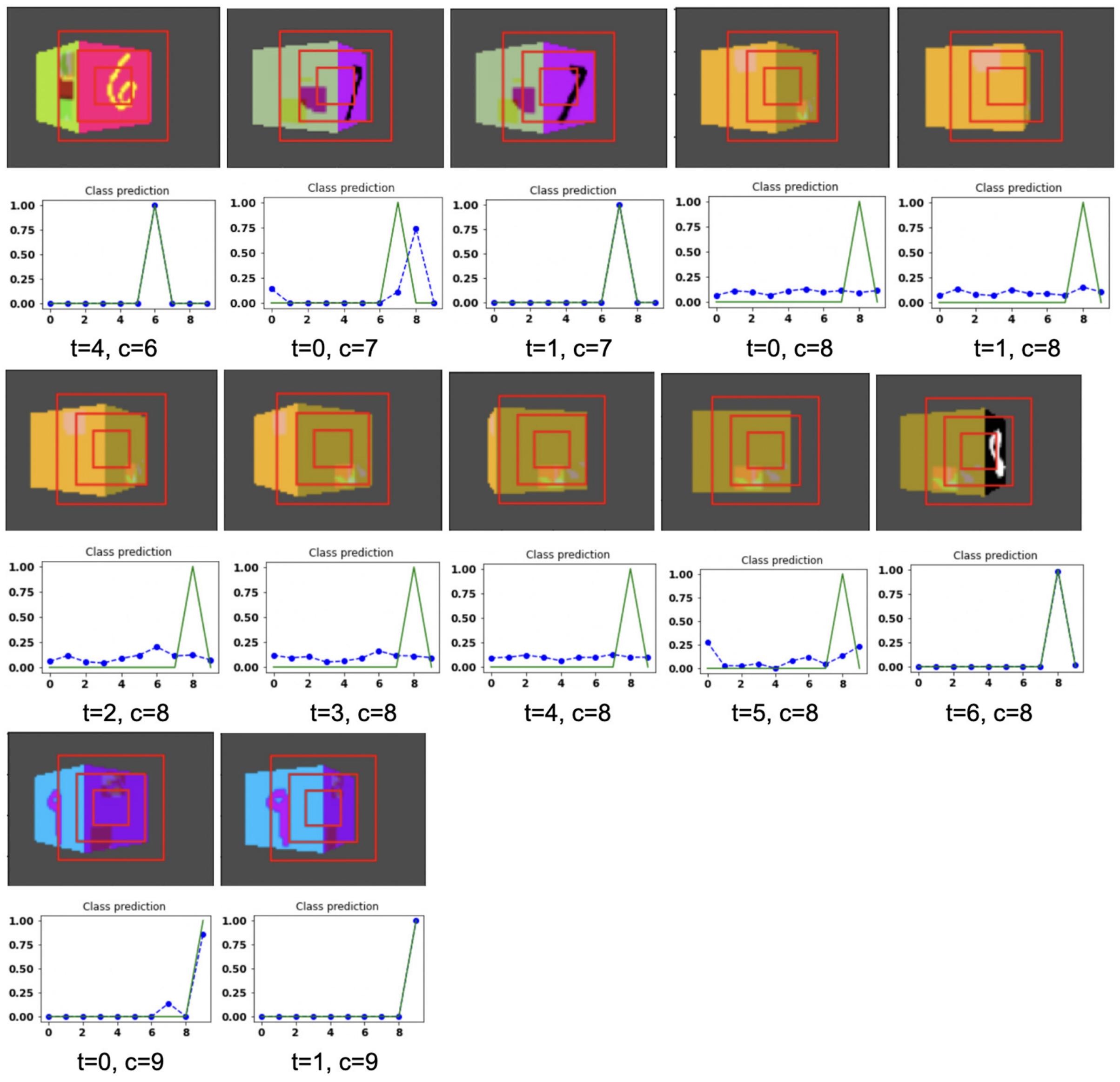

FIGURE 12
Illustrates the camera movements around the cube to search the target face in the view of size 75 × 100, predicted by our model in 3D Cluttered RGB MNIST Cube dataset. For each class at time *t*, there is a movement (shown in the row of camera view images) and corresponding classification probabilities (shown in the row of plots). In the row of camera view images, the three concentric red windows depict the glimpse at the center of the view image. In the plot corresponding with the above view image, the green curve is the desired classification probabilities and the dotted dashed-blue curve is the predicted classification probabilities at time *t*.

classification probabilities crosses a certain testing threshold (= 0.95). A slight variation in values of the hyperparameters is used for the 3D Cluttered RGB MNIST Cube dataset after tuning.

Jump length is the displacement from one location to the next location. The jump length of the camera from one location to the next location on the orbit is considered as a predefined parameter. The jump length of the camera is 12 in case of Grayscale 3D Cluttered MNIST Cube dataset, and 20 in case of 3D Cluttered QuickDraw Cube dataset and 3D Cluttered RGB MNIST Cube dataset.

4. Discussion

To search for the entrance of a building, where there is neither a boundary wall, nor a clear path leading to the entrance, we usually move on the circular path around the building in

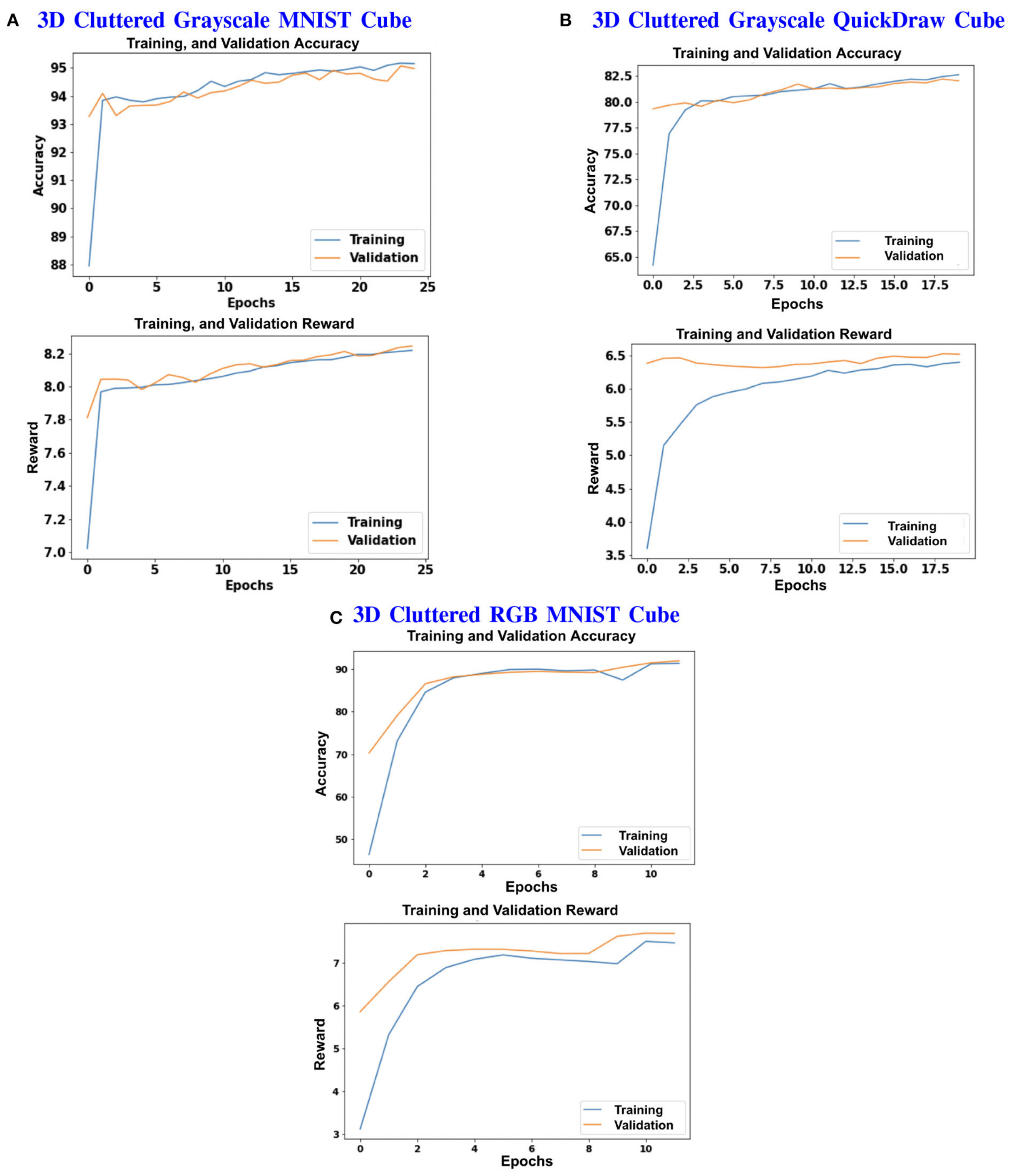

FIGURE 13
(A–C) Shows the plots of Accuracy (1*st* and 3*rd* row) and Reward (2*nd* and 4*th* row) vs. Epochs of 3D Cluttered Grayscale MNIST Cube dataset, 3D Cluttered Grayscale QuickDraw Cube dataset, and 3D Cluttered RGB MNIST Cube dataset, respectively.

either clockwise or anticlockwise direction until we find the entrance. While performing such a task, we also take care that the movement should not involve rapid alternation between the two directions, and must progress continuously in one direction. The best application of the current model can be in space. For example, geostationary satellites and spy satellites revolving around the earth in a circular orbit require a searching capability of one specific large area of the earth to get a bird's eye view or to obtain information about various weather, natural calamities, deforestation, and similar activities. From the results of camera movement shown in Figures 6–12, the proposed model is able to avoid alternative movements and is always able to follow the continuous movements to search the target face of the cube. There are three major components to consider the proposed model biologically inspired. First, the model takes the input of multiple concentric windows of different scales, which resembles the differential spatial resolution of the central fovea and the peripheral regions of the retinal. Second, the model processes the view and its corresponding functions of the camera's location, θ, which is analogous to determining the position using path integration and using it to navigate the world. The classifier and camera motion networks are analogous to the processing of visual information along the "what and where/how" pathways (Schenk and McIntosh, 2010), respectively. Third, the model uses Elman, Jordan, JK-flip-flop recurrence layers as memory to store the history of the view and corresponding location, which resemble the feedback loops present among the visual cortical areas, for example from V_1 to thalamus or from V_2 to V_1, (Angelucci and Sainsbury, 2006). The output layers of the classifier and the camera motion network are used to attribute a specialized role to both of the networks for classification and searching tasks, by feeding the outputs back into the first fully connected Elman and Jordan layers in their corresponding channels. The output vector of the camera motion network (Q-values) which has information about the action to be taken by the camera is fed back into the fully connected Elman and Jordan layer and the output vectors of this layer passed through fully connected flip-flop layer and gets concatenated with the output of the last layer of the camera position network; this wide loop is responsible for storing the history of location and view.

5. Conclusions

In the proposed model, we have shown how the "classifier" and "camera motion" networks coordinate with each other to perform the 3D visual search task. The BIAS-3D successfully performed the classification task on a 3D environment on three datasets (Table 1). As shown in the results, movements generated by the model to search a target in the given cube always aim at the target face and take meaningful movements so that the camera looks at the target and classifies it correctly. Based on the results described herewith, we want to extend the model to more complicated full 3D searches in a 3D environment like, for example, searching for defects on the surface of a 3D structure. The model can then be applied to full scale object detection and recognition in 3D space.

Data availability statement

The datasets generated during the current study are available in the RGB MNIST-dataset repository: https://github.com/sweta111/RGB_MNIST-dataset.

Author contributions

The experiments were conducted by SK. SK wrote the entire text. VA and MN created the setup of the environment. VC contributed in providing the key ideas, editing the manuscript drafts, and providing insight into structure. All authors contributed to the article and approved the submitted version.

Acknowledgments

We acknowledge the support from Pavan Holla and Vigneswaran for the implementation of flip-flop neurons. We also acknowledge Sowmya Manojna to generate RGB MNIST dataset. SK acknowledges the financial support of Ministry of Human Resource Development for graduate assistantship.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

Supplementary material

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fncom.2022.1012559/full#supplementary-material

References

Abed-alguni, B. H. (2018). Action-selection method for reinforcement learning based on cuckoo search algorithm. *Arab. J. Sci. Eng.* 43, 6771–6785. doi: 10.1007/s13369-017-2873-8

Angelucci, A., and Sainsbury, K. (2006). Contribution of feedforward thalamic afferents and corticogeniculate feedback to the spatial summation area of macaque V1 AND LGN. *J. Compar. Neurol.* 498, 330–351. doi: 10.1002/cne.21060

Armstrong, M., and Murlis, H. (2007). *Reward Management: A Handbook of Remuneration Strategy and Practice*. London: Kogan Page Publishers.

Ba, J., Mnih, V., and Kavukcuoglu, K. (2014). Multiple object recognition with visual attention. *arXiv preprint arXiv:1412.7755*. doi: 10.48550/arXiv.1412.7755

Bengio, Y., Frasconi, P., urgen Schmidhuber, J., and Elvezia, C. (2001). *Gradient Flow in Recurrent Nets: The Difficulty of Learning Long-Term Dependencies*. Fakultat fur Informatik.

Borji, A., Sihite, D. N., and Itti, L. (2011). "Computational modeling of top-down visual attention in interactive environments," in *BMVC, Vol. 85* (Los Angeles, CA), 1–12. doi: 10.5244/C.25.85

Borji, A., Sihite, D. N., and Itti, L. (2012). "Probabilistic learning of task-specific visual attention," in *2012 IEEE Conference on Computer Vision and Pattern Recognition* (Los Angeles, CA), 470–477. doi: 10.1109/CVPR.2012.6247710

Carpenter, R., and McDonald, S. A. (2007). Later predicts saccade latency distributions in reading. *Exp. Brain Res.* 177, 176–183. doi: 10.1007/s00221-006-0666-5

Churchland, A. K., Kiani, R., and Shadlen, M. N. (2008). Correction: Corrigendum: decision-making with multiple alternatives. *Nat. Neurosci.* 11, 851–851. doi: 10.1038/nn0708-851c

Elman, J. L. (1991). Distributed representations, simple recurrent networks, and grammatical structure. *Mach. Learn.* 7, 195–225. doi: 10.1007/BF00114844

Fan, J., Wang, Z., Xie, Y., and Yang, Z. (2020). "A theoretical analysis of deep q-learning," in *Learning for Dynamics and Control* (Sardinia), 486–489.

Gao, D., Han, S., and Vasconcelos, N. (2009). Discriminant saliency, the detection of suspicious coincidences, and applications to visual recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* 31, 989–1005. doi: 10.1109/TPAMI.2009.27

Gao, D., Mahadevan, V., and Vasconcelos, N. (2008). On the plausibility of the discriminant center-surround hypothesis for visual saliency. *J. Vis.* 8, 13–13. doi: 10.1167/8.7.13

Gao, D., and Vasconcelos, N. (2004). Discriminant saliency for visual recognition from cluttered scenes. *Adv. Neural Inform. Process. Syst.* 17, 481–488.

Glorot, X., and Bengio, Y. (2010). "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (California), 249–256.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep Learning, Vol. 1*. Cambridge: MIT Press.

Gruber, A. J., Dayan, P., Gutkin, B. S., and Solla, S. A. (2006). Dopamine modulation in the basal ganglia locks the gate to working memory. *J. Comput. Neurosci.* 20:153. doi: 10.1007/s10827-005-5705-x

Haque, A., Alahi, A., and Fei-Fei, L. (2016). "Recurrent attention models for depth-based person identification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (Stanford), 1229–1238. doi: 10.1109/CVPR.2016.138

Holla, P., and Chakravarthy, S. (2016). "Decision making with long delays using networks of flip-flop neurons," in *2016 International Joint Conference on Neural Networks (IJCNN)* (Vancouver, BC), 2767–2773. doi: 10.1109/IJCNN.2016.7727548

Jongejan, J., Rowley, H., Kawashima, T., Kim, J., and Fox-Gieg, N. (2016). *The Quick, Draw!-AI Experiment*. Available online at: http://quickdraw.withgoogle.com

Jordan, M. (1986). *Serial Order: A Parallel Distributed Processing Approach*. Technical Report, California University, Institute for Cognitive Science, San Diego, CA.

Kahou, S. E., Michalski, V., Memisevic, R., Pal, C., and Vincent, P. (2017). "RATM: recurrent attentive tracking model," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (Honolulu, HI), 1613–1622. IEEE. doi: 10.1109/CVPRW.2017.206

Kanan, C., Tong, M. H., Zhang, L., and Cottrell, G. W. (2009). Sun: top-down saliency using natural statistics. *Vis. Cogn.* 17, 979–1003. doi: 10.1080/13506280902771138

Kanezaki, A., Matsushita, Y., and Nishida, Y. (2018). "Rotationnet: joint object categorization and pose estimation using multiviews from unsupervised viewpoints," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (Tokyo), 5010–5019. doi: 10.1109/CVPR.2018.00526

Kingma, D. P., and Ba, J. (2014). Adam: a method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. doi: 10.48550/arXiv.1412.6980

Knapp, A. (1938). An introduction to clinical perimetry. *Arch. Ophthalmol.* 20, 1116–1117. doi: 10.1001/archopht.1938.00850240232021

Kratsios, A., and Hyndman, C. (2020). Deep arbitrage-free learning in a generalized HJM framework *via* arbitrage-regularization. *Risks* 8:40. doi: 10.3390/risks8020040

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Adv. Neural Inform. Process. Syst.* 25, 1097–1105. doi: 10.1145/3065386

Lan, S., Ren, Z., Wu, Y., Davis, L. S., and Hua, G. (2020). "SaccadeNet: a fast and accurate object detector," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (Seattle, WA), 10397–10406. doi: 10.1109/CVPR42600.2020.01041

Le Meur, O., Le Callet, P., Barba, D., and Thoreau, D. (2006). A coherent computational approach to model bottom-up visual attention. *IEEE Trans. Pattern Anal. Mach. Intell.* 28, 802–817. doi: 10.1109/TPAMI.2006.86

LeCun, Y., Cortes, C., and Burges, C. J. C. (1998). *The MNIST Database of Handwritten Digits* (New York, NY). Available online at: http://yann.lecun.com/exdb/mnist/

Liu, T., Lam, K.-M., Zhao, R., and Kong, J. (2021). Enhanced attention tracking with multi-branch network for egocentric activity recognition. *IEEE Trans. Circuits Syst. Video Technol.* doi: 10.1109/TCSVT.2021.3104651

Liu, T., Zhao, R., Jia, W., Lam, K.-M., and Kong, J. (2022). Holistic-guided disentangled learning with cross-video semantics mining for concurrent first-person and third-person activity recognition. *IEEE Trans. Neural Netw. Learn. Syst.* 32, 1–15. doi: 10.1109/TNNLS.2022.3202835

Minut, S., and Mahadevan, S. (2001). "A reinforcement learning model of selective visual attention," in *Proceedings of the fifth international conference on Autonomous agents* (New York, NY), 457–464. doi: 10.1145/375735.376414

Mnih, V., Heess, N., Graves, A., and Kavukcuoglu, K. (2014). Recurrent models of visual attention. *arXiv preprint arXiv:1406.6247*. doi: 10.48550/arXiv.1406.6247

Nair, V., and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. *Icml.* 10, 807–814.

Roitman, J. D., and Shadlen, M. N. (2002). Response of neurons in the lateral intraparietal area during a combined visual discrimination reaction time task. *J. Neurosci.* 22, 9475–9489. doi: 10.1523/JNEUROSCI.22-21-09475.2002

Roth, C. H. Jr., Kinney, L. L., and John, E. B. (2020). *Fundamentals of Logic Design*. Boston, MA: Cengage Learning.

Rowe, J. B., Hughes, L., and Nimmo-Smith, I. (2010). Action selection: a race model for selected and non-selected actions distinguishes the contribution of premotor and prefrontal areas. *Neuroimage* 51, 888–896. doi: 10.1016/j.neuroimage.2010.02.045

Schenk, T., and McIntosh, R. D. (2010). Do we have independent visual streams for perception and action? *Cogn Neurosci.* 1, 52–62. doi: 10.1080/17588920903388950

Scherer, D., Müller, A., and Behnke, S. (2010). "Evaluation of pooling operations in convolutional architectures for object recognition," in *International Conference on Artificial Neural Networks* (Heidelberg: Springer), 92–101. doi: 10.1007/978-3-642-15825-4_10

Segal, M., and Akeley, K. (2010). *The openGLr Graphics System: A Specification (Version 4.0 (Core Profile))*. Beaverton, OR: The Khronos Group Inc.

Shaikh, M., Kollerathu, V. A., and Krishnamurthi, G. (2019). "Recurrent attention mechanism networks for enhanced classification of biomedical images," in *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)* (Venice), 1260–1264. IEEE. doi: 10.1109/ISBI.2019.8759214

Shen, J., Du, Y., Wang, W., and Li, X. (2014). Lazy random walks for superpixel segmentation. *IEEE Trans. Image Process.* 23, 1451–1462. doi: 10.1109/TIP.2014.2302892

Su, H., Maji, S., Kalogerakis, E., and Learned-Miller, E. (2015). "Multi-view convolutional neural networks for 3D shape recognition," in *Proceedings of the IEEE International Conference on Computer Vision* (Massachusetts), 945–953. doi: 10.1109/ICCV.2015.114

Sutton, R. S., and Barto, A. G. (1998). *Introduction to Reinforcement Learning, Vol. 135*. Cambridge: MIT Press.

Sweta, K., Vigneswaran, C., and Chakravarthy, V. S. (2021). The flip-flop neuron - a memory efficient alternative for solving challenging sequence processing and decision-making problems. *BioRxiv*. doi: 10.1101/2021.11.16.468605

Voleti, R. (2021). Unfolding the evolution of machine learning and its expediency. *IJCSMC*. 10, 1–7. doi: 10.47760/ijcsmc.2021.v10i01.001

Wang, W., Shen, J., and Porikli, F. (2015a). "Saliency-aware geodesic video object segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (Boston, MA), 3395–3402.

Wang, W., Shen, J., and Shao, L. (2015b). Consistent video saliency using local gradient flow optimization and global refinement. *IEEE Trans. Image Process*. 24, 4185–4196. doi: 10.1109/TIP.2015.2460013

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., et al. (2015). "3D shapenets: a deep representation for volumetric shapes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (Boston, MA), 1912–1920.

Xu, B., Liu, J., Hou, X., Liu, B., Garibaldi, J., Ellis, I. O., et al. (2019). Attention by selection: A deep selective attention approach to breast cancer classification. *IEEE Trans. Med. Imaging* 39, 1930–1941. doi: 10.1109/TMI.2019.2962013

Yang, Y., Deng, C., Gao, S., Liu, W., Tao, D., and Gao, X. (2016a). Discriminative multi-instance multitask learning for 3d action recognition. *IEEE Trans. Multim*. 19, 519–529. doi: 10.1109/TMM.2016.2626959

Yang, Y., Deng, C., Tao, D., Zhang, S., Liu, W., and Gao, X. (2016b). Latent max-margin multitask learning with skelets for 3-D action recognition. *IEEE Trans. Cybern*. 47, 439–448. doi: 10.1109/TCYB.2016.2519448

Yang, Y., Liu, R., Deng, C., and Gao, X. (2016c). Multi-task human action recognition *via* exploring super-category. *Signal Process*. 124, 36–44. doi: 10.1016/j.sigpro.2015.10.035

Zhang, D., Han, J., Jiang, L., Ye, S., and Chang, X. (2017). Revealing event saliency in unconstrained video collection. *IEEE Trans. Image Process*. 26, 1746–1758. doi: 10.1109/TIP.2017.2658957

Zhang, L., Zhang, Q., and Xiao, C. (2015). Shadow remover: image shadow removal based on illumination recovering optimization. *IEEE Trans. Image Process*. 24, 4623–4636. doi: 10.1109/TIP.2015.2465159