Continuous-time on-policy neural reinforcement learning of working memory tasks

Davide Zambrano
CWI
Amsterdam, The Netherlands
D.Zambrano@cwi.nl

Pieter R. Roelfsema
Department of Vision & Cognition
Netherlands Institute for Neuroscience (KNAW)
Amsterdam, The Netherlands
p.roelfsema@nin.knaw.nl

Sander M. Bohte
CWI
Amsterdam, The Netherlands
S.M.Bohte@cwi.nl

***Abstract*—As living organisms, one of our primary characteristics is the ability to rapidly process and react to unknown and unexpected events. To this end, we are able to recognize an event or a sequence of events and learn to respond properly. Despite advances in machine learning, current cognitive robotic systems are not able to rapidly and efficiently respond in the real world: the challenge is to learn to recognize both *what* is important, and also *when* to act. Reinforcement Learning (RL) is typically used to solve complex tasks: to learn the *how*. To respond quickly – to learn *when* – the environment has to be sampled often enough. For "enough", a programmer has to decide on the step-size as a time-representation, choosing between a fine-grained representation of time (many state-transitions; difficult to learn with RL) or to a coarse temporal resolution (easier to learn with RL but lacking precise timing). Here, we derive a continuous-time version of on-policy SARSA-learning in a working-memory neural network model, AuGMEnT. Using a neural working memory network resolves the *what* problem, our *when* solution is built on the notion that in the real world, instantaneous actions of duration dt are actually impossible. We demonstrate how we can decouple action duration from the internal time-steps in the neural RL model using an action selection system. The resultant *CT-AuGMEnT* successfully learns to react to the events of a continuous-time task, without any pre-imposed specifications about the duration of the events or the delays between them.**

I. Introduction

A self-driving car travels along the way when suddenly a man crosses the street. The car has to stop immediately to avoid the impact. This is an example of the complexity of the environment that we live in, where many unknown and unexpected events have to be recognized and processed rapidly and continuously. Despite advances in machine learning, current robotic systems are not able to rapidly and efficiently respond in the real world: the challenge is to learn to recognize both *what* is important, and also *when* to act. Reinforcement Learning (RL) algorithms are commonly used as a learning paradigm to learn *what* to respond to in complex environments. In RL, the agent changes its behavior according to experience collected during the exploration of the world [1]. In typical RL tasks however, ad hoc abstractions are used: the actual relevant events are provided in compact representations. Much attention has been given recently to learning compact state representations from high-dimensional observations, where deep learning is the most well-known approach for this, including approaches like Long Short-Term Memory (LSTM) [2], [3], that capture state from both present and past observations in memory structures. Memory allows such networks to solve the *what* problem as posed by working memory tasks, by transforming certain classes of partially observable Markov decision problems (POMDPs) into Markov Decision Problems (MDPs) [4].

Here, we observe that in standard RL not only the representation is abstracted, but also the *timing* of events that is sampled in the ordered presentation. State-transitions in RL are defined in discrete steps and the agent state is updated every step. Effectively, the agent is given the information on *when* a decision has to be taken. To respond quickly however, the environment has to be sampled often. A programmer has to decide on the step-size as a time-representation, choosing between a fine-grained representation of time or to a coarse temporal resolution. The former corresponds to many state-action transitions that are difficult to learn, while in the latter correct action sequences are easier to learn but lack precise timing. For a learning self-driving car, this means it will either be very difficult to learn to avoid unexpected obstacles, or it will respond too late and hit the man. A number of approaches have been developed to deal with continuous-time in RL. Baird proposed *Advantage Learning* [5] as a continuous-time formalization of Q-learning. An RL neural working memory approach like [3] is in principle compatible with continuous-time Advantage Learning. However, like Q-learning, Advantage Learning is an off-policy method, and especially in tasks where exploration may incur large penalties, on-policy learning is preferred [1], [6]. Another common approach for continuous-time RL is based on actor-critic models, e.g. [7]. We find however that for the latter, there is no current work that includes methods for learning compact working memory representations, and our own efforts had great difficulty with reaching convergence when using an actor-critic method.

AuGMEnT (Attention-Gated MEmory Tagging) [8], [9] is a recent biologically plausible neural network framework, that is trained with on-policy SARSA. AuGMEnT includes working memory and shares a number of features with Long Short-Term Memory (LSTM) [2], [3]. AuGMEnT is not compatible with Advantage Learning for two reasons: AuGMEnT updates only those weights that are responsible for the selection of the winning action, and it does not use information about the not-selected actions. Here, we propose to solve the fast-responding self-driving car problem by presenting a continuous-time model of AuGMEnT. This solution is based on the idea that in biological brains, instantaneous actions of infinitesimal duration are actually impossible. Here, we introduce an action selection system that controls the action execution, by keeping active the selected action for the needed time. In biological brains, an action selection network can be mapped onto specific neural substrates, including the basal ganglia, as shown in [10]. Current actions can be interrupted if another is more important or urgent. Moreover, the exploratory system has been re-defined allowing enough time for execution and thus for spatial and temporal credit assignment. The continuous-time framework changes the way standard RL problems are presented: it defines *time* as an intrinsic property of the task, and it considers unavoidable delays in action selection and execution. In the next sections, the continuous-time AuGMEnT model is described including the description of the action selection network and the exploratory system. Results from three case studies are presented, which show the ability of the network to reach convergence within a remarkable short number of epochs.

II. Methods

A. Continuous-Time AuGMEnT Model

AuGMEnT [8], [9] is a biologically plausible RL framework for solving discrete-time MDPs that require learnable working memory to construct Markov states in the hidden layer of the neural network model. A continuous-time generalization of MDPs is known as Semi-Markov Decision Problems (SMDP's), as defined in [11]. The following description considers discrete-space continuous-time algorithm in which one output neuron codes for one discrete action, where we describe the standard AuGMEnT framework for time-steps of size dt. By decreasing this time-step we approximate continuous-time, where we denote the resultant algorithm as *CT-AuGMEnT*.

B. Feedforward model

The AuGMEnT network is composed of three layers of units connected by modifiable synapses (see Fig. 1). The sensory layer represents stimuli with instantaneous and transient units. Instantaneous units, $x(t)$, are active as long as the stimuli is present. Transient units represent positive and negative changes in sensory input:

$$\begin{aligned} x^{+}(t) &= \tfrac{1}{dt}[x(t) - x(t-dt)]_{+}, \\ x^{-}(t) &= \tfrac{1}{dt}[x(t-dt) - x(t)]_{+}, \end{aligned} \tag{1}$$

where $[.]_{+}$ is a threshold operation that returns 0 for all negative inputs. Here we assume backward Euler approximation of the time derivative of $\dot{x}^{+}(t)$ and $\dot{x}^{-}(t)$ for small dt. This approximation will be used for the rest of the text. Instantaneous units i are fully connected to regular units j in the association layer, through connections v^{R}_{ij}. Activations for regular units are thus computed as:

$$inp^{R}_{j}(t) = \sum_{i} v^{R}_{ij} x_i(t), \tag{2}$$

$$y^{R}_{j}(t) = \sigma(inp^{R}_{j}(t)), \tag{3}$$

σ is the sigmoidal activation function, with derivative:

$$\sigma'(inp^{R}_{j}(t)) = y^{R}_{j}(t)(1 - y^{R}_{j}(t)). \tag{4}$$

Transient units l are fully connected to memory units m through connections v^{M}_{lm}:

$$\begin{aligned} inp^{M}_{m}(t) &= inp^{M}_{m}(t-dt) + \textstyle\sum_{l} v^{M}_{lm} x'_{l}(t) \\ y^{M}_{m}(t) &= \sigma(inp^{M}_{m}(t)), \end{aligned} \tag{5}$$

where $x'(t) = [x^{+}(t)\ x^{-}(t)]$. The third (output) layer receives input from the association layer by the connections w^{R}_{jk} and w^{M}_{mk}. This layer computes Q-values for every possible action a in the current state s:

$$\begin{aligned} Q^{\pi}(s,a) &= E_{\pi}[R_t | s_t = s, a_t = a], \\ \text{with } R_t &= \int_{t}^{\infty} \mathrm{e}^{-\frac{\zeta - t}{\tau}}\ r(s(\zeta), a(\zeta)) d\zeta, \end{aligned} \tag{6}$$

where E_{π} is the expected discounted future reward R_t given a and s, under action-selection policy π. The action value $q_k(t)$ is computed as:

$$q_k(t) = \sum_{m} w^{M}_{mk} y^{M}_{m}(t) + \sum_{j} w^{R}_{jk} y^{R}_{j}(t). \tag{7}$$

C. Action selection

In AuGMEnT, as defined in [8], action selection follows a policy π known as Max-Boltzmann rule to ensure a good balance between exploration and exploitation. It selects the greedy action (highest $q_k(t)$) with probability $1 - \epsilon$, and, with a small probability ϵ, a random action sampled from the Boltzmann distribution [12].

When we consider discrete-size time steps dt, any exploration rate ϵ^{dt} implies that, potentially, a new action can be selected every dt, lasting also only for a duration of dt. Baird in 1993 [5] demonstrated that for standard RL algorithms like Q-learning, reduction of the time-step duration in tasks affects convergence rate. The difficulty in learning the corresponding many state-action transitions can be explained intuitively: the shorter the time-step duration, the less is the effect of that action on the final reward. In principle, for instantaneous actions no effect can be seen at the end, and it becomes impossible to assign the right credit to the right action. This is exacerbated when state-approximators are used, like artificial neural networks, as they introduce their own imprecision in the computed Q-values. To resolve this problem, we observe that real-world actions have a typical duration: they take some time to execute from start to finish, and our everyday experience is that this execution is to some degree (at least initially) non-interruptible. Such action-duration behavior can be implemented as an action selection mechanism, for example as the basal ganglia model described in [10]. This is achieved by connecting the Z-layer to the Q-layer with off-center on-surround connectivity: each neuron in the Q-layer transmits its value q_i to all output neurons but sends inhibition to only one output neuron (see the connections between the two layers in Fig. 1). The input to the Z-layer neuron is:

$$u_i(t) = -w^{-} q_i(t) + w^{+} \sum_{j \neq i}^{n} q_j(t), \tag{8}$$

where the balance between inhibition and excitation has been chosen as $\nu = w^{-}/w^{+} = 3$. Then, the activity of the neuron can be modeled as a leaky integrator:

$$\dot{a}_i(t) = -\rho(a_i(t) - u_i(t)), \tag{9}$$

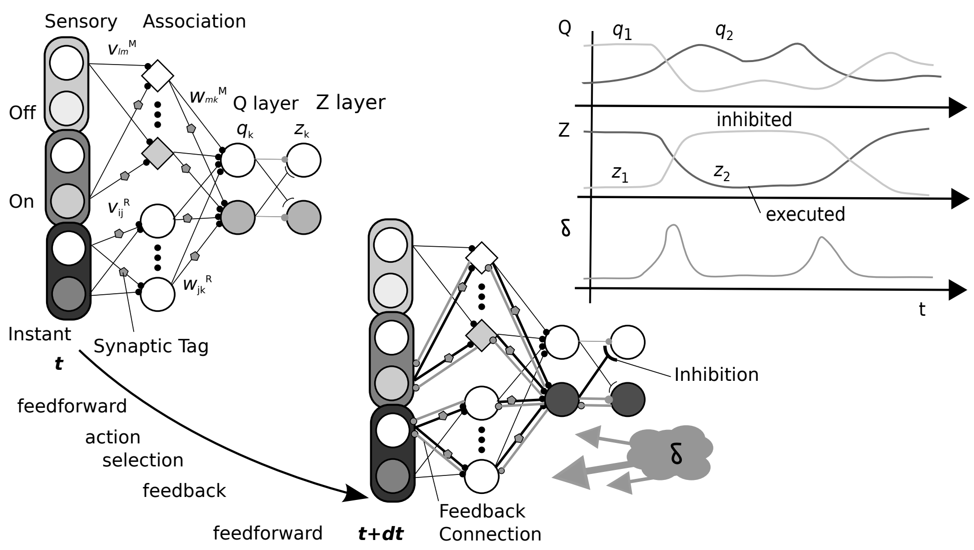

Fig. 1: Continuous-Time AuGMEnT network with the winner-take-all neural implementation in the Z-layer for action selection, implementing off-center on-surround in the action space. The dynamics of the network are illustrated in the inset: while the Q-values in the Q-layer can vary over time, the action selection mechanism in the Z-layer ensures a clean switching of actions with minimum action-duration. The corresponding δ TD-error are computed continuously.

where ρ is a rate constant which determines the speed at which the activity reaches the equilibrium (with $\rho = 1$ the activity is instantaneous). Finally, the output activation of the Z-layer is bounded using the same sigmoidal activation function used in Eq. (3):

$$y_i(t) = \sigma(a_i(t)). \tag{10}$$

Winner-take-all is guaranteed if the minimum activation is selected at every dt, thus the actions are selected continuously. In this case the Q-layer determines the degree of inhibition in the action space, which means that all the actions that are not selected are inhibited, and the action that is selected receives less inhibition. Fig. 1 (up-right) shows an example of the selection system: while the Q-values are free to change continuously, only one action is selected at a time. The action selection system responses on the basis of the mutual distance between the Q-values. Farther Q-values yield to a faster shift. The reaction time is controlled by the model parameters and mimic the accumulation of evidences before taking a decision. When exploration takes place, an external current is added to the explorative action in eq. (8). Note that such exploration does not change the computed Q-values, but only the input to the action selection system. The external current follows an exponentially decay function such that it does not have a fixed duration, and the duration of activation depends on the current Q-values. For an exploratory action ei, Eq. (8) is modified as:

$$u_{ei}(t) = -w^-(q_{ei}(t) + I_{ex}e^{-\tau_{ex}dt}) + w^+ \sum_{j \neq ei}^{n} q_j(t). \tag{11}$$

Where I_{ex} is the magnitude of the extra inhibition (we chose 5) and τ_{ex} is the time constant of the decay which determines the effective duration of the exploration (here we set to 0.01).

D. Feedback model

During learning, two factors modulate the network plasticity: a global neuromodulatory signal and an attentional feedback signal. Once an action is selected, the unit that codes the winning action a feeds back to earlier processing levels to create synaptic Tags (equivalent to eligibility traces) on the responsible synapses. The decaying Tag update for Tags k is defined as:

$$Tag_{Jk}(t + dt) = (1 - \frac{dt}{\phi})Tag_{Jk}(t) + dt[y_J(t)z_k(t)]. \tag{12}$$

With $z_k = 1$ for the selected action, $z_k = 0$ elsewhere, and J stands either for j or m indexes. Thus, the association units that provided strong input to the winning action a also receive strongest feedback. Equivalently, Tags on connections between regular units and instantaneous units are computed as:

$$Tag_{ij}(t + dt) = (1 - \frac{dt}{\phi})Tag_{ij}(t) + dt[x_i(t)\sigma'(inp_j^R(t))w'_{aj}] \tag{13}$$

(note that feedback connections w'_{aj} and feedforward connections w_{ja} have the same strength, as in [13]). Synaptic traces between sensory units and memory cells are used for learning working memory:

$$\begin{aligned} sTrace_{lm}(t + dt) &= sTrace_{lm}(t) + dt[x'_l(t)] \\ Tag_{lm}(t + dt) &= (1 - \frac{dt}{\phi})Tag_{lm}(t) \\ &\quad + dt[sTrace_{lm}(t)\sigma'(inp_m^M(t))w'_{am}]. \end{aligned} \tag{14}$$

With the SARSA temporal difference learning rule, the network compares the predicted outcome $q_a(T-1)$ to the sum of the reward $r(t)$ and the discounted action-value $q_{a'}(T)$ of the unit a' that wins the subsequent competition:

$$\delta(T) = r + \gamma q_{a'}(T) - q_a(T-1), \tag{15}$$

where T is the time-step defined in standard compound definition of RL algorithms. However, in the case of continuous-time TD learning, the estimate of the predicted outcome is defined as in [7]:

$$\dot{Q}^\pi(s, a) = \frac{1}{\tau}Q^\pi(s, a) - r(t). \tag{16}$$

If the equivalence in (16) is not satisfied, the prediction should be adjusted to decrease the inconsistency:

$$\delta(t) = r(t) - \frac{1}{\tau} q_{a'}(t) + \dot{q}_a(t). \tag{17}$$

Using the backwards Euler approximation for dt [7] gives the following discrete TD update:

$$\delta(t) = r(t) + \frac{1}{dt}\left[\left(1 - \frac{dt}{\tau}\right) q_{a'}(t) - q_a(t - dt)\right]. \tag{18}$$

In this case we consider the discount factor $\gamma = 1 - \frac{dt}{\tau}$ and the Q-values have been rescaled as $q_a(T) = \frac{1}{dt} q_a(t)$. Moreover, to be consistent with the previous representation of AuGMEnT which defines the Tag decay as $\alpha = (1 - \lambda\gamma)$, we have: $\lambda = \frac{1 - \frac{dt}{\phi}}{1 - \frac{dt}{\tau}}$; the reward $r(t)$ also has to be rescaled as $r(t)/dt$. Note that by defining $dt = 1$ we obtain the same equations of AuGMEnT.

Finally, the plasticity of all synapses (either R or M units) is defined as:

$$\begin{aligned} v_{ij}(t + dt) &= v_{ij}(t) + dt[\beta\delta(t)Tag_{ij}(t)] \\ w_{jk}(t + dt) &= w_{jk}(t) + dt[\beta\delta(t)Tag_{jk}(t)]. \end{aligned} \tag{19}$$

It is worth mentioning that the Tags and synaptic weights are updated *every* dt and that the order of updating is important for the correct execution of the algorithm. In this sense, every time step, a new error δ is computed (eq. (15)) and the synapses are updated consequently with eq. (19). Then, traces and Tags are updated through eq. (12), (13), and (14). At the end of a trial, the activity of memory units, traces, Tags, and Q-values are set to zero, after updating of the weights with a δ that reflects the transition to the terminal state (the expected reward is null).

III. RESULTS

In this section, we demonstrate the continuous-time model of AuGMEnT outlined above. Firstly, we show that our continuous-time formalization of AuGMEnT minimizes the temporal difference error by stochastic gradient decent as in the standard implementation [8], [9]. Then, we demonstrate the method in three case studies. The first one shows the ability of the continuous-time model to learn to react to the input even in case of fast updates. The second is an extension of the previous task in which working memory units are used to remember a specific input and to select the right answer at the end of the trial. The third one is a task used in neuroscience useful to demonstrate the ability of the network to represent contextual information and to remember previously seen input.

A. Continuous-Time AuGMEnT stochastically minimizes the reward-prediction error (RPE)

We can show that the continuous-time formulation of AuGMEnT outlined above reduces the reward-prediction error (RPE) as the original AuGMEnT formulation. The objective function is defined as:

$$E(t) = \frac{1}{2}\left|\delta(t)^2\right|. \tag{20}$$

Given (20) and (17), the gradient of the objective function with respect to the weights w^R_{ja} becomes:

$$\begin{aligned} \frac{\partial E(t)}{\partial w^R_{ja}} &= \delta(t)\frac{\partial}{\partial w^R_{ja}}\left[r(t) - \frac{1}{\tau}q_{a'}(t) + \dot{q}_a(t)\right] \\ &= \delta(t)\left[-\frac{1}{\tau}\frac{\partial q_{a'}(t)}{\partial w^R_{ja}} + \frac{\partial \dot{q}_a(t)}{\partial w^R_{ja}}\right]. \end{aligned} \tag{21}$$

Since the boundary condition for the Q-function, defined in (6), is given at $t \to \infty$, it is more appropriate to update the past estimates without affecting the future estimates [7]. Thus, recalling (18) and discretizing (21), a reduction of the gradient is guaranteed if:

$$\begin{aligned} -\frac{\partial E(t)}{\partial w^R_{ja}} &= \delta(t)\frac{\partial q_a(t - dt)}{\partial w^R_{ja}} \\ &= \delta(t) y^R_j(t - dt), \end{aligned} \tag{22}$$

which is consistent with the trace update in (12) for $a = 1$. The same can be shown for the synapses between memory units M and Q-values:

$$-\frac{\partial E(t)}{\partial w^M_{ma}} = \delta(t) y^M_m(t - dt). \tag{23}$$

Note that in the latter equations the update of the synapses has to be consistent with the neuron activity at the previous dt, which is stored in the Tags (see (12) and (13)). Thus, Tags have to be updated after the weights. Gradient decent for the weights v^R_{ij} is similarly computed:

$$\begin{aligned} -\frac{\partial E(t)}{\partial v^R_{ij}} &= \delta(t)\frac{\partial q_a(t - dt)}{\partial v^R_{ij}} \\ &= \delta(t)\frac{\partial q_a(t - dt)}{\partial y^R_j(t - dt)}\frac{\partial y^R_j(t - dt)}{\partial inp^R_j(t - dt)}\frac{\partial inp^R_j(t - dt)}{\partial v^R_{ij}} \\ &= \delta(t) w'^R_{aj}\sigma'(inp^R_j(t - dt)) x_i(t - dt) \end{aligned} \tag{24}$$

and for v^M_{lm}:

$$\begin{aligned} -\frac{\partial E(t)}{\partial v^M_{lm}} &= \delta(t)\frac{\partial q_a(t - dt)}{\partial v^M_{lm}} \\ &= \delta(t)\frac{\partial q_a(t - dt)}{\partial y^M_m(t - dt)}\frac{\partial y^M_m(t - dt)}{\partial inp^M_m(t - dt)}\frac{\partial inp^M_m(t - dt)}{\partial v^M_{lm}} \\ &= \delta(t) w'^M_{am}\sigma'(inp^M_m(t - dt)) sTrace_{lm}(t - dt), \end{aligned} \tag{25}$$

where we assume for simplicity that the strength of the feedback from the motor layer back to the association layer w'^R_{aj} is equal to w^R_{ja} and, analogously, $w'^M_{am} = w^M_{ma}$.

B. L-maze task

The L-Maze task is a simple fast-response scenario that mimics the case where a self-driving car has to stop as soon as a man appears in front of it. In the task, the agent has to move in a corridor and turn right as quickly as possible at the end of it. The agent has actions N, E, S, W to move in all compass directions. When the agent remains in the same place (e.g. by moving into a wall), it receives a negative reward (-0.1). The correct decision at the end of the maze is worth 4, and the wrong decision -1. The task also has a time-out condition: after $1.5N + 2$ time-steps we automatically stop the trial, and start a new one. In the input space a wall is

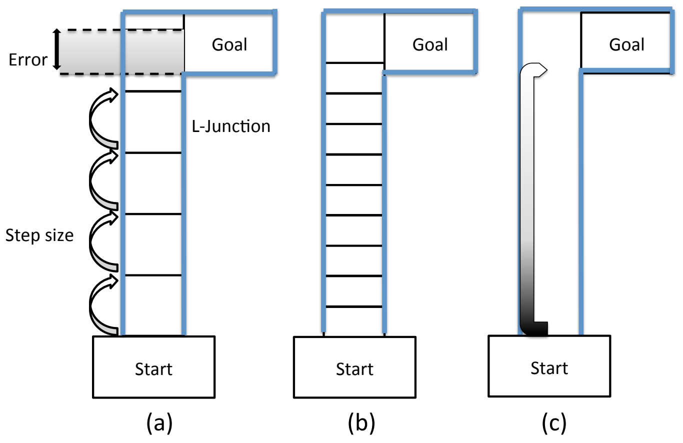

Fig. 2: L-Maze task: the agent has to reach the goal at the end of the corridor. On the left the standard time-step representation (a); in the middle an increased number of updates reduces the error (b); on the right the continuous-time version used (c). The gradient in the arrow illustrates the ease with which the selected action can be interrupted.

encoded as 0 and an empty space as 1. Thus the agent observes $[1, 0]$ at the beginning of the trial and $[0, 1]$ in the turning point. Continuous actions do not have a fixed finite duration: if the network selects the wrong action, only one negative reward is given in total, at the beginning of the selection. Then nothing happens until a new decision is made or the time-out is reached. We define learning to have converged when the model makes 80% optimal choices over the last 50 examples. For the simulations we give each network at most 10000 trials to learn the task. For each condition 10 different networks were tested. The parameters for the simulation are: $\beta = 0.15$; $\lambda = 0.20$; $\gamma = 0.90$; $\epsilon = 0.025$ and $\Theta = 2.5$. Initial synaptic weights are drawn from a uniform distribution $U[-0.25, 0.25]$.

Rather than the *how*, here the difficulty of the task is the *when*. Suppose we update the network every 1 second and the agent travels with constant velocity such that it takes 10 seconds to reach the turning point. The turning signal appears just after that time, and the agent has to wait the next update to change its direction. It will make an error equal to the space it covers during that second (see Fig. 2 a). If we want to reduce this error we can increase the number of updates. In standard time-step implementations, this is equivalent to a task with longer corridor length: we tested standard time-step AuGMEnT with corridor lengths of $10, 20, 40, 100$. Standard time-step AuGMEnT is obtained by setting $dt = 1$. In contrast to standard time-step AuGMEnT, in CT-AuGMEnT we can decrease the dt size to update the network more often without affecting the duration of actions (Fig. 2). To compare with the previous example we chose $dt = 0.5, 0.25, 0.1$. This is true since the number of updates (or decisions) required to cross the corridor is the same. Furthermore, we also tested for $dt = 0.05, 0.02, 0.01, 0.005$, which correspond to a corridor length of $200, 500, 1000, 2000$ respectively (for these corridor lengths standard AuGMEnT does not reach the convergence criterium).

Fig. 3 shows the results of the simulations for different lengths of the maze and equivalently reduced dt. The figure shows the median and the standard deviation of the median for the number of trial to reach convergence for 10 networks for

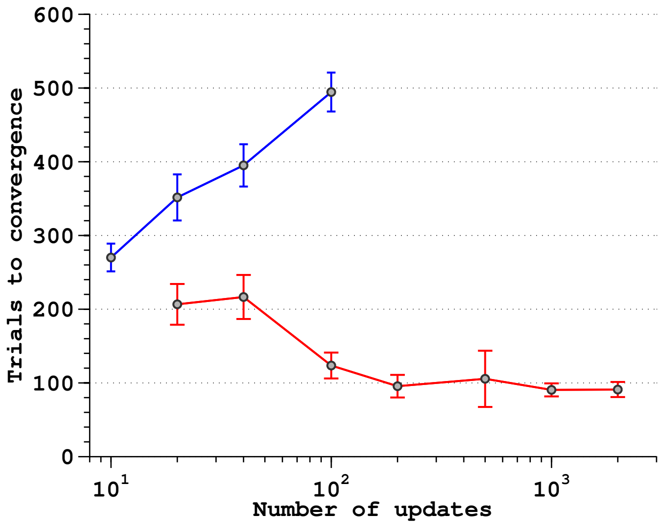

Fig. 3: Median and standard deviation of the median of the trials to convergence for different maze lengths for standard AuGMEnT (blue) and different dts for CT-AuGMEnT (red)

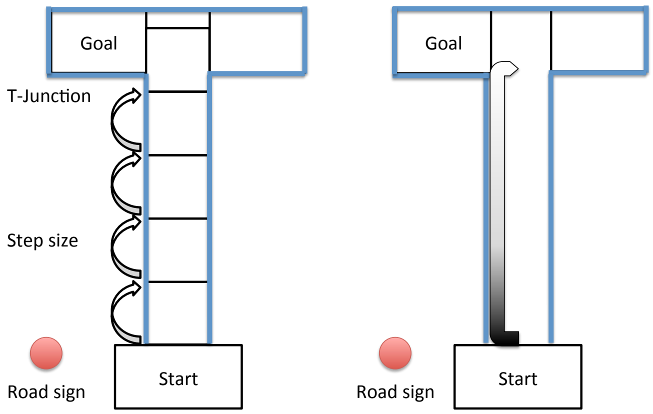

Fig. 4: T-Maze task. On the left the standard time-step representation, while on the right the continuous-time version used. The road-sign (red) is presented for 1 second at the beginning of the trial.

every point in the graph. In standard AuGMEnT the number of trials needed to reach convergence increases with the corridor length (note the log-scale of the x-axis). Instead, with CT-AuGMEnT the opposite is true: increasing the number of updates in the corridor does not affect the time required to reach convergence. The estimate of the Q-values in the corridor state actually becomes more precise leading to a decrease in convergence-time. Of course this effect is mainly due to the simplicity of the task, but it demonstrates that CT-AuGMEnT can reach convergence even with a large number of updates.

C. *T-maze task*

The T-Maze task is a working memory task based on [3] and [14]. Information presented at the beginning of the maze has to be remembered to make optimal decisions at the end of the corridor. As for the L-Maze, the agent has actions N, E, S, W to move in all compass directions; task difficulty is scaled by increasing the corridor length N (see Figure 4). The

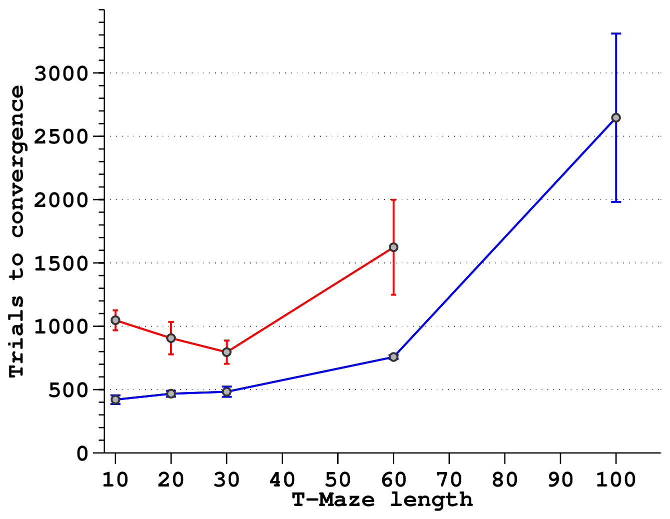

Fig. 5: Mean and standard deviation of the mean of the trials to convergence for different T-maze lengths and different dt. Blue: standard AuGMEnT, $dt = 1$. Red: CT-AuGMEnT, $dt = 0.1$

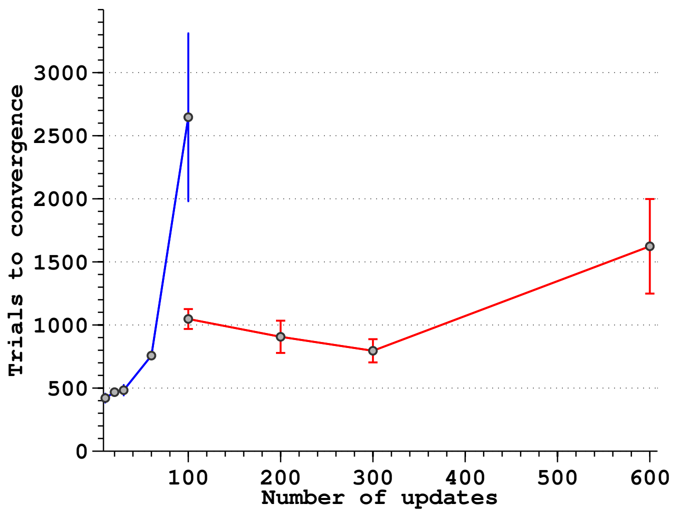

Fig. 6: Mean and standard deviation of the mean of the trials to convergence for the number of updates: standard AuGMEnT, blue, $dt = 1$, CT-AuGMEnT, red, $dt = 0.1$

same reward and time-out conditions as in the L-Maze were applied. With respect to [14], we increased the time limit (from $1.2N + 2$ to $1.5N + 2$), because the new exploration system could lead to longer time to reach the end of the corridor. Also, compared to [14], a different input representation is used: a wall is encoded as 0, a road-sign as 2 and an empty space as 1. Thus the agent observes $[2, 1, 0]$ or $[0, 1, 2]$ for the first second. For the simulations, we gave each network at most 50000 trials to learn the task. Convergence was determined by checking at 80% optimal choices as in [3] and [14] for each condition.

Fig. 5 shows the results of the simulations for different lengths of the maze and (corresponding) dt. For every corridor length, 10 networks were tested. The figure also shows the standard deviation of the number of epochs for every length in order to give an idea about the distribution of the results. Standard AuGMEnT is a version of continuous-time AuGMEnT in which $dt = 1$ and the action selection mechanism is not active. As in the L-Maze task we need to consider the same number of updates for each implementation. This means that a corridor length of 10 for CT-AuGMEnT with $dt = 0.1$ corresponds to a standard AuGMEnT with a corridor of 100 (see Fig. 6 for a direct comparison with respect the number of updates).

D. Saccade/antisaccade task

We also apply CT-AuGMEnT to the working memory saccade/anti-saccade task as in [8]. This task introduces the effect of context in selecting a specific strategy to follow Fig. 7. In this case, the agent has to learn that the color of the a fixation mark determines the strategy. The sequence of time events in this task is shown in Figure 8. Every trial started with an empty screen, shown for one second. Then a fixation mark was shown that was either black or white, indicating that a pro- or anti-saccade would be required. The model had to fixate within ten seconds, otherwise the trial was terminated without reward. If the model fixated for two consecutive seconds, we presented a cue on the left or the right side of the screen for one second and gave the fixation reward r_{fix}. This was followed by a memory delay of two seconds during which only the fixation point was visible. At the end of the memory delay the fixation mark turned off. To collect the final reward r_{fin} in the pro-saccade condition, the model had to make an eye-movement to the remembered location of the cue and to the opposite location on anti-saccade trials. The trial was aborted if the model failed to respond within eight seconds. With respect to the standard implementation of AuGMEnT [8], we kept the same temporal sequence of the events, while the update (time-steps) of the network can be done independently at an increased rate: in Fig. 8 the time-steps have been substituted with seconds and the network update has been chosen 1, 0.5 and 0.1 (respectively 2 and 10 times the original update). For all tasks we used $r_{fix} = 0.2$ and $r_{fin} = 1.5$.

Fig. 9a-c shows the distribution of the trials needed to reach convergence for each dt. These results are based on runs with 100 randomly initialized networks. The convergence criterium has been chosen as in [8]. We defined that learning

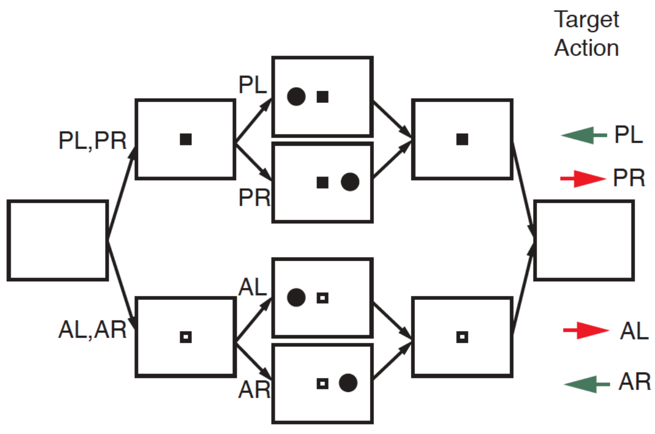

Fig. 7: Standard Saccade/antisaccade task with all the possible conditions: Prosaccade Left and Right; Antisaccade Left and Right. Picture from [9]

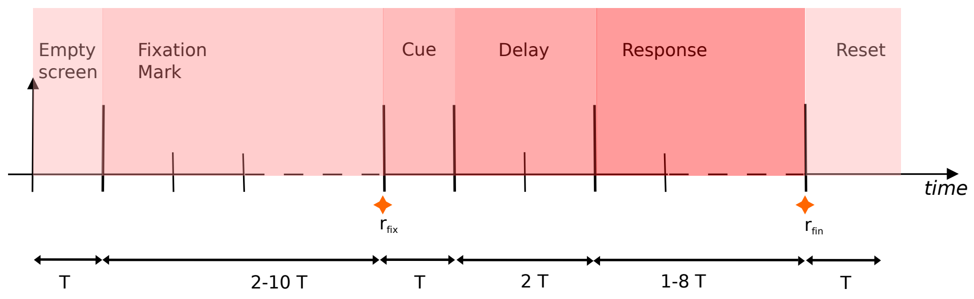

Fig. 8: Saccade/antisaccade task with time step representation. Every time step can be interpreted as seconds and the network can be be updated several time in between this interval.

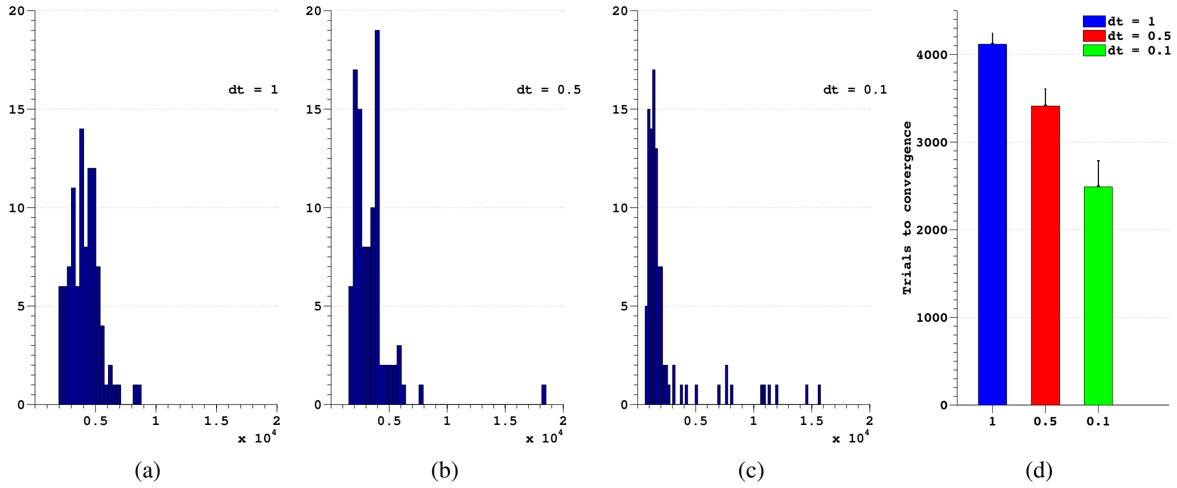

Fig. 9: Distribution of the number of trials needed to reach convergence in saccade/antisaccade task for different dt's.

was complete when the model made 90% optimal choices for each of the four possible conditions over the last 50 examples of each. Every network had 2.5×10^4 trials to learn the task. We obtained a convergence rate of 100%, 97% and 98% respectively. The mean of the convergence rate for the standard AuGMEnT was 4120. Also in this task decreasing the dt duration helps the time-to-convergence: for $dt = 0.5$, CT-AuGMEnT needed on average 3400 trials, and for $dt = 0.1$ it required 2490 (Fig. 9d).

IV. Conclusion

We derived a continuous-time version of SARSA-learning in a working-memory neural network model, AuGMEnT. Our main contribution is that we decouple action duration from the internal time-steps of the model. An action-selection mechanisms was defined, inspired by modern models of the brain's basal ganglia. This action-selection mechanism enforces a minimum temporal duration of a selection action by inhibiting other action-units in the network. As this inhibition decays with time, the selected action is effectively interruptible when novel and urgent inputs are encountered. A continuous-time SARSA-learning algorithm has potential applications when learning fast responses using RL, as the interruptible action selection effectively keeps the time-action space compact. Moreover, as AuGMEnT is considered biologically plausible [9], the continuous-time equivalent presented here allows us to directly compare unit activations in the network with electro-physiological measurements. CT-AuGMEnT solves the credit-assignment problem with the same 'attentional' feedback mechanisms described in the standard version [9]. Feedback connections form the output layer to the earlier levels Tag the relevant synapses that were responsible for the action selection [13]. AuGMEnT thus provides a biological explanation of the eligibility traces toward the Tag mechanisms [15], which permits learning if time passes between the action and the reward-prediction error [16].

In this work, we show that the derived algorithm, CT-AuGMEnT, computes the same stochastic gradient as the original serial-compound model. In experiments, we further show that it performs well on the continuous-time representation of a simple fast-response tasks and two standard (and hard) working-memory tasks. We show that the algorithm reaches convergence with a similar number of trials with respect to standard AuGMEnT, where in the simple fast-response task, the more precise estimation of the Q-values actually improves the time-to-convergence. Consequently, this model can successfully be applied to non-linear continuous-time problems without significant losses in terms of convergence rate and/or speed of learning: CT-AuGMEnT successfully learns to react to the events of a continuous-time task, without any pre-imposed specifications about the duration of the events or the delays between them. Remarkably, the AuGMEnT model computes a function approximator that learns non-linear mappings between inputs and Q-values in a single neural network model. Neural network function approximators for Q-learning in general are held to be unstable, and various algorithmic variations have been developed to deal with this [17]; an actor-critic version of AuGMEnT exhibited similar stability problems. In contrast, we find no such instability for SARSA-based AuGMEnT and the continuous-time version presented here, at least for the presented tasks.

It is worth noting that, while the rapid alternation of forward and feedback activity every dt in the network seems biologically implausible, we can see from the update equations that in principle, a separate feedback network can be constructed to carry the feedback signal, similar to [18]. This leaves unresolved the issue that in real networks, both forward and feedback activations take time. Such delays will introduce a fixed offset between forward and feedback signals. One solution we can envision here is to have neurons match this delays through delayed plasticity windows, similar to what is observed in cerebellum [19]. Whit respect to other approaches, CT-AuGMEnT provides a continuous-time on-policy neural reinforcement learning network with working memory. Recent works based on actor-critic architecture emphasize the role of basal ganglia in action selection with spiking neurons in continuous-time [20], [21]. These models are based on the Doya's work [7]. However, these approaches do not include working memory, and thus do not have the ability to keep track of past events. A remarkable approach of working memory units in continuous-time is shown in [22] and [23] based on LSTM; this somewhat related working-memory approach however does not consider action selection.

It is important to note that we show here that a continuous-time representation changes the way in which we define the task for the network. For example, in the standard time-step representation a new action can be selected every time-step, exactly during specific changes in the task. The time-step representation requires the network to give an answer at specific moments of the trial. This is even more important in the case of explorative actions: here actions can be interrupted in every dt due to a change of the inputs or an explorative action. The action selection system we introduce here accounts for the fact that before deciding on an action, especially when sampling often and in a noisy environment, the optimal decision requires accumulation of evidence [24]. The action-selection mechanism we implemented adheres to this notion: for large relative changes in computed Q-values, the mechanism quickly switches to a new action. For small relative changes this process is slower, and may even be reversed before switching actions when the computed Q-values change back again, as could be the case for noisy observations.

Taken together, CT-AuGMEnT is able to learn how to solve complex non-linear working memory tasks, thus making an actual step into a more realistic representation of learning.

Acknowledgment. This work was carried out during the tenure of an ERCIM "Alain Bensousan" Fellowship Programme.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, ser. A Bradford book. Bradford Book, 1998.

[2] S. Hochreiter and J. Schmidhuber, "Long short-term memory." *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[3] B. Bakker, "Reinforcement learning with long short-term memory," in *Advances in Neural Information Processing Systems 14*, T. Dietterich, S. Becker, and Z. Ghahramani, Eds. MIT Press, 2002, pp. 1475–1482. [Online]. Available: http://papers.nips.cc/paper/1953-reinforcement-learning-with-long-short-term-memory.pdf

[4] M. T. Todd, Y. Niv, and J. D. Cohen, "Learning to Use Working Memory in Partially Observable Environments through Dopaminergic Reinforcement," in *Advances in Neural Information Processing Systems 21*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds. Curran Associates, Inc., 2009, pp. 1689–1696.

[5] L. C. Baird, III, "Advantage updating," 1993.

[6] G. A. Rummery and M. Niranjan, "On-line Q-learning Using Connectionist Systems," 1994.

[7] K. Doya, "Reinforcement learning in continuous time and space." *Neural Computation*, vol. 12, no. 1, pp. 219–245, Jan. 2000.

[8] J. Rombouts, S. M. Bohte, and P. R. Roelfsema, "Neurally Plausible Reinforcement Learning of Working Memory Tasks." in *Advances in Neural Information Processing Systems 25 (NIPS 2012)*, 2012, pp. 1880–1888.

[9] J. O. Rombouts, S. M. Bohte, and P. R. Roelfsema, "How Attention Can Create Synaptic Tags for the Learning of Working Memories in Sequential Tasks," *PLoS computational biology*, Jan. 2015.

[10] K. N. Gurney, T. J. Prescott, and P. Redgrave, "A computational model of action selection in the basal ganglia. I. A new functional anatomy," *Biological cybernetics*, Jan. 2001.

[11] S. J. Bradtke and M. O. Duff, "Reinforcement Learning Methods for Continuous-Time Markov Decision Problems," in *Advances in Neural Information Processing Systems 7*, G. Tesauro, D. S. Touretzky, and T. K. Leen, Eds. MIT Press, 1995, pp. 393–400.

[12] M. Wiering and J. Schmidhuber, "HQ-Learning," *Adaptive Behavior*, vol. 6, no. 2, pp. 219–246, Sep. 1997.

[13] P. R. Roelfsema and A. van Ooyen, "Attention-gated reinforcement learning of internal representations for classification." *Neural Computation*, vol. 17, no. 10, pp. 2176–2214, Oct. 2005.

[14] J. O. Rombouts, P. R. Roelfsema, and S. M. Bohte, "Learning Resets of Neural Working Memory," in *ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, Bruges (Belgium), 23-25 April 2014, Apr. 2014.

[15] U. Frey and R. G. M. Morris, "Synaptic tagging and long-term potentiation," *Nature*, vol. 385, no. 6616, pp. 533–536, Feb. 1997.

[16] S. Cassenaer and G. Laurent, "Conditional modulation of spike-timing-dependent plasticity for olfactory learning." *Nature*, vol. 482, no. 7383, pp. 47–52, Feb. 2012.

[17] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[18] D. Zipser and D. E. Rumelhart, "The neurobiological significance of the new learning models," in *Computational neuroscience*. MIT Press, 1993, pp. 192–200.

[19] M. R. Carey, "Synaptic mechanisms of sensorimotor learning in the cerebellum." *Current Opinion in Neurobiology*, vol. 21, no. 4, pp. 609–615, Aug. 2011.

[20] N. Frémaux, H. Sprekeler, and W. Gerstner, "Reinforcement learning using a continuous time actor-critic framework with spiking neurons." *PLoS computational biology*, vol. 9, no. 4, pp. e1 003 024–e1 003 024, Mar. 2013.

[21] E. Vasilaki, N. Frémaux, R. Urbanczik, W. Senn, and W. Gerstner, "Spike-based reinforcement learning in continuous state and action space: when policy gradient methods fail." *PLoS computational biology*, vol. 5, no. 12, p. e1000586, Dec. 2009.

[22] F. Rivest, J. F. Kalaska, and Y. Bengio, "Conditioning and time representation in long short-term memory networks." *Biological cybernetics*, vol. 108, no. 1, pp. 23–48, Jan. 2014.

[23] ——, "Alternative time representation in dopamine models." *Journal of computational neuroscience*, vol. 28, no. 1, pp. 107–130, Feb. 2010.

[24] T. R. Stanford, S. Shankar, D. P. Massoglia, M. G. Costello, and E. Salinas, "Perceptual decision making in less than 30 milliseconds," *Nature Neuroscience*, vol. 13, no. 3, pp. 379–385, Jan. 2010.