Basal Ganglia Dynamics in Structure Learning

Matthew Clapp

August 2024

Neuroscience Institute
Dietrich College of Humanities and Social Sciences
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Dr. Timothy Verstynen, Chair
Dr. Jonathan E. Rubin
Dr. Xaq Pitkow
Dr. Blake Richards, McGill University

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research was supported by NIH awards R01DA053014 and R01DA059993 as part of the CRCNS program.

Abstract

We frequently encounter new challenges that necessitate action to achieve positive outcomes. To navigate these challenges, our biological brains must utilize environmental feedback from past experiences, combined with knowledge of patterns and structures in the environment, to guide the selection of the most rewarding actions. Central to this process is the basal ganglia, which is crucial for facilitating actions in response to dopaminergic signaling [1] [2] [3]; however, the precise relationships between synaptic plasticity, large-scale network dynamics, and reinforcement learning computations are not fully understood. Additionally, it remains unclear how this system interacts with other brain regions, particularly the hippocampus, which may provide information regarding environmental structure [4] [5] [6] [7].

In this dissertation, I investigate the neural mechanisms underlying reinforcement learning in the basal ganglia and its interaction with structure learning systems. First, I introduce and demonstrate the capabilities of CBGTPy, a framework designed to simulate biologically realistic neural dynamics and plasticity within the cortico-basal ganglia-thalamic (CBGT) system. This flexible framework allows researchers to explore the system's internal dynamics at multiple scales across various simulated behavioral tasks. Second, I examine the computational value of hippocampal representations in reinforcement learning by employing a model of place cells derived from successor representation [8]. These place cells, along with alternative handcrafted representations, are used to bias cortical inputs within the CBGTPy network throughout its performance of an example structured task [9].

The combined model exhibits enhanced flexibility and improved performance in structured reinforcement learning tasks, mirroring observed human behaviors in similar environments. The extent of this facilitation, however, heavily depends on the properties of the supplied place cells. These findings suggest that hippocampal computations are well-suited for learning task structures, though the current CBGTPy model has limited capacity to fully leverage the information present in biological place cells. This research underscores the importance of multi-region interactions in the brain's ability to solve structured tasks, offering significant insights into the neural basis of decision-making and learning.

Contents

Dedication

To my sister, my mother, and my father.
This work is a testament to your love.
Thank you.

Acknowledgements

To my advisor, Tim Verstynen, for his unwavering support no matter the circumstances.

To the CBGTPy team, without whom this project would have been impossible: Jyotika Bahuguna, Cristina Giossi, Jon Rubin, and Cati Vich.

To all the members of the Exploratory Intelligence group, especially Eric Yttri and Julia Badyna, for their helpful insights.

To the uPNC Summer Undergraduate Research Program for introducing me to computational neuroscience.

To my friends for their companionship.

Chapter 1

Introduction and Aims

1.1 General Background

Life is filled with decisions that are simultaneously novel yet repetitive. Consider the adventure of fine dining at an unfamiliar restaurant: while you are certain to encounter new faces, new foods, and new scenery, the procedures for dining out are relatively consistent. You do not know which menu items will taste the best, but you do know to place an order only after being seated, that the sequence in which the order is placed does not affect what food you receive, and that the available payment methods are the same regardless of whether you ate any dessert. This scenario illustrates two fundamental problems: 1) how do you learn to value choices made in new environments, and 2) how do you leverage invariant structure from prior experiences to generalize to value-based decisions in new contexts? In this dissertation, I approach these two problems from a computational perspective based on the circuit logic of the neural substrates of reinforcement and structure learning in the mammalian brain. For value learning, I developed a biologically-realistic spiking neural network model of the mammalian cortico-basal ganglia-thalamic circuit, and I illustrated how this circuit naturally implements dynamic reinforcement learning that can adapt in response to changes in environmental contingencies. I then explored how relational structure is learned through the processing of the available sensory data streams to identify large-scale patterns [10] in order to act as a world model for subsequent reinforcement learning. To accomplish this, I integrated existing models of relational learning in medial temporal lobe systems with basic reinforcement learning, connecting hippocampal-dependent computations with basal-ganglia-dependent computations. Together, these projects illuminate how the mammalian brain can leverage similarities between the past and present experiences to infer the presence of state relationships relevant to immediate value-based decisions, facilitating the type of complex volitional behavior for which most mammals are known.

1.1.1 Biological Implementation of Reinforcement Learning

The above process, referred to herein simply as structure learning [10], represents a particularly challenging reformulation of the standard reinforcement learning (RL) paradigm, and biological agents are demonstrably able to perform such learning with high accuracy [11] [12]. While the exact neural mechanisms which facilitate this behavior remain unclear, experimental evidence has cemented the cortico-basal ganglia-thalamic (CBGT) circuit as a key contributor to action selection [13], particularly in RL tasks [2] [14]. The CBGT contains two core pathways, often referred

to as the direct and indirect pathways, which serve similar yet opposing functions [1]. The direct pathway includes projections from the striatum to the internal globus pallidus (GPi) and is associated with disinhibition of actions, while the indirect pathway, associated with inhibition of actions, includes projections from the striatum to the external globus pallidus (GPe) and from the GPe to the GPi [15]. Although these pathways were classically interpreted to perform mutually exclusive tasks, it is now understood that they are simultaneously active and that the balance of these pathways during the decision process is what determines whether specific actions are performed or instead suppressed in favor of alternatives [1] [15]. Crucially, the striatal spiny projection neurons (SPNs) which originate these pathways selectively express different dopamine receptors, with the direct and indirect SPNs expressing D1 and D2 receptors respectively [16]. As a result, positive dopaminergic feedback from the substantia nigra pars compacta (SNc) can strengthen the direct pathway while weakening the indirect pathway, leading to future disinhibition of the selected action, as expected from a reinforcement learning system [1].

While the concept of pathway competition is simple, the mechanics of the full CBGT network are surprisingly complex. Hierarchical models such as that of Frank and Badre (2012) show that the looping CBGT circuit can efficiently learn complex conditional rules [17], while Caligiore *et al.* (2019) highlight how the interplay between multiple learning systems is fundamental to the performance of the system as a whole, a process they deem "super-learning" [18]. Beyond describing the underlying computations, a truly successful model would capture the process by which dopaminergic feedback signals drive the basal ganglia towards the optimal action selection policy. While it is believed that striatal activity patterns are key drivers of the competitive process of action selection [19] [20], the striatal neurons involved must necessarily perform their learning using only the information locally available to them [21]. As a result, the precise interplay between global network dynamics and the spike-timing-dependent plasticity of corticostriatal synapses is crucial for the effective learning of the overall system [21]. Although there are numerous models of dopaminergic plasticity [22] [19], a system capable of simulating the complete CBGT loop's interplay with complex reinforcement learning environments, in a manner true to the known neuroanatomy and plasticity mechanisms, is needed. Such a system would allow for the creation of complete simulated time-courses of neural activity, spike-timing-dependent plasticity, and agent behavior, enabling the formation and evaluation of specific testable hypotheses about the CBGT system and its performance in various environments.
Problem: While it is clear that the CBGT pathways are involved in reinforcement learning, it remains unclear how the system's dynamics regulate the flow of information during the decision-making process and how this information is transferred across decisions.

1.1.2 Structure Learning with the CBGT Circuit

Current computational models of the basal ganglia are unable to fully explain the performance of humans and animals in reinforcement learning tasks featuring latent structure. For example, Frank and Badre's hierarchical model [17] fails to describe how structural knowledge is generalized and transferred between environments. Furthermore, many examples of latent structures, such as spatial grids, do not elegantly lend themselves to representation via a set of hierarchical conditional rules. For this reason, it is intuitive to speculate that additional neural mechanisms, complementary to the basal ganglia, could greatly facilitate structural learning. An improved model, containing the computational processes necessary for structure learning, is clearly required to successfully explain the known decision-making behavior of mammals.

While it is not obvious *a priori* which additional neural systems would enable the

CBGT loop to generate accurate behavior in structure learning tasks, the hippocampal-entorhinal cortical system, and the neural representations present therein, are proposed herein as a prime candidate for such a mechanism. Recent research has shown the importance of hippocampal representations in reinforcement learning, including the existence of "value place" neurons [7], and the relevance of the conjunctive nature of hippocampal representations to reinforcement learning [4]. Furthermore, imaging data supports the notion that the entorhinal cortex encodes information about reinforcement learning tasks in a manner that reflects the underlying task structure [5]. It has been hypothesized that structure learning involves the formation of "mental maps" much akin to the spatial maps for which the HC-EC system is famous, enabling the behavioral agent to perform an analog of path integration to keep track of the environmental state [7]. Recently, it has been proposed that the HC-EC system learns to encode state as a successor representation (SR), a form of predictive map in which hippocampal cells are tuned both towards the current state and towards expected future states in a time-discounted manner [6] [23] [8]. The ability of the hippocampus to learn the successor representation, and thereby encode the long-term statistical relationships between states, is precisely the sort of computational capability that could facilitate efficient learning in structured RL tasks.

To evaluate the potential role of the HC-EC system in RL tasks, a joint model was constructed. As outlined in the following sections, this model combined an SR model of the hippocampus with a physiologically-realistic model of action selection in the CBGT loop. This combination was applied to a structured multi-arm bandit task to evaluate the model's ability to learn the latent structure and generalize across experiences in similar environments. After thorough training, it was expected that the neural representations derived from SR would greatly facilitate the performance of the basal ganglia in this structured task. **Problem: How can structure learning via hippocampal circuits coordinate with CBGT computations to regulate reinforcement learning?**

1.2 Overview of Specific Aims

This dissertation is divided into two core components, each further subdivided into a few main goals. The first component focuses on modeling the processes by which the basal ganglia's dynamics and synaptic plasticity can lead to the successful formation of an action policy, while the second component explores how the hippocampal system provides useful cortical representations over which the action policy can be formed.

Aim 1a: The development of CBGTPy, a physiologically-realistic model of the cortico-basal ganglia-thalamic system, incorporating dopaminergic plasticity and go/no-go dynamics. A new and highly flexible simulation framework was created, enabling the adaptation of the core CBGT dynamics and dopaminergic plasticity mechanisms to a variety of behavioral tasks, including tasks with very complex environmental dynamics. The first step involved creating a model with accurate single-channel dynamics based on the known anatomy of the circuit, featuring plasticity of corticostriatal synapses and the go/no-go dynamics characteristic of the direct and indirect pathways.

Aim 1b: The implementation of action selection dynamics, including cross-channel competition and effective learning through credit assignment. The CBGTPy model was extended with multiple action channels to incorporate the winner-take-all dynamics characteristic of action selection, enabling its application to complex RL environments. The model's network dynamics allowed for dopaminergic feedback to guide the selective strengthening of certain corticostriatal synapses, facilitating the learning of the correct action policy. The completed CBGTPy framework

enables the creation of complete simulated experimental time-courses, providing useful predictions about the relationship between neural dynamics and behavioral properties.

Aim 2a: The development of a successor representation model of structured reinforcement learning environments. A structured RL task was formulated, translating a real experiment into symbolic sequences. A recurrent neural network (RNN) was trained on these sequences, and successor features were extracted from the RNN's internal state. Using these successor features, simulated time-courses of hippocampal activity were produced. The successor representation, when trained on these environments, even with only unguided random action selection behavior, was able to encode a significant portion of the latent structure.

Aim 2b: The creation of an interface between the SR model and CBGTPy in which the SR influences the competitive dynamics. A combined model leveraged the SR by using the derived hippocampal representations to bias the cortical inputs to the full CBGT system model. To enable the SR to guide reinforcement learning, a method was developed for estimating state values and using the resulting reward prediction errors to guide action policy formation. When applied to a structured RL task, the combined SR-CBGTPy model demonstrated enhanced flexibility in response to changes in environmental state, leading to improved behavioral performance.

1.3 CBGTPy: Creating an Extensible Framework

The primary objective of CBGTPy is to serve as an extensible Python package for the simulation of a physiologically-realistic decision-making agent with internal dynamics directly modeled after the cortico-basal ganglia-thalamic loop. The construction, simulation, and analysis of the CBGTPy model reflect contributions from myself and a diverse team of interdisciplinary researchers. This model achieves high realism through the implementation of multiple underlying neural pathways, the use of spiking neural populations, and the application of a realistic spike-timing-dependent plasticity rule. Furthermore, the framework is designed around the principle of flexibility, exemplified by a clear division in the model between the dynamics of the behavioral agent and the dynamics of the external environment. This structure enables the straightforward adaptation of the CBGT system to a wide range of simulated experimental protocols, including complex multi-choice reinforcement learning tasks, with minimal additional developmental effort.

The network architecture, dynamics, and implementation details of CBGTPy are outlined in detail in Clapp *et al.* (in prep) [24], the text of which is reproduced herein as Chapter 2. The architecture is structured around the notion of cross-channel competition, where the competition is driven primarily by the within-channel balance of direct and indirect pathway activity. An overview of the relevant cellular populations and their respective pathways is shown in Figure 2.1. Stimulation of the cortical populations serves as the input to the network, while the firing rates of the thalamic populations serve as the output. When the thalamic firing rates exceed a threshold, it triggers the selection of the corresponding action and the removal of stimulation for the non-selected cortical populations. Dopaminergic feedback, representing a reward prediction error, triggers plasticity, affecting the corticostriatal weights and biasing the activity patterns of the action channels in future trials accordingly. In this way, the model is capable of learning to consistently select the most rewarding action, even when the values of the actions change over time, though it is unable to learn the more complex latent structures of the tasks.

1.4 SR-CBGT: Applying Successor Representation to RL Environments

Building a bridge between successor representation (SR) and structured reinforcement learning tasks was a multistage process. First, an example task was precisely characterized as a sequence of discrete events, and an initial non-SR representation of environmental state was learned from this sequence. Then, successor features, upon which the SR is computed, were extracted from this non-SR state representation. The result of this process was a successor representation of state along with a corresponding model of place cell activity. These place cell representations were then used to control the cortical inputs and dopaminergic feedback of the full CBGT model, allowing the network to alter its behavioral policy in response to changes in environmental state.

1.4.1 Background: The Hippocampus and Successor Representation

Successor representation is an algorithm for estimating the values of states by combining two structures: a vector which estimates the immediate reward value for each state, and a matrix which encodes the expected discounted future occupancy of each state given the current state [25]. The value of any particular state is thus estimated as the dot product of the reward vector and the corresponding row of the future occupancy matrix. The successor representation balances flexibility and efficiency of model-based and model-free learning, as the experience of reward can boost the value of states likely to precede the current state, as with model-based learning, without imposing any other assumptions as to the structure of the state space [26].

There are many observed similarities between the SR transition matrix and the activity patterns of hippocampal place cells in CA1, with these cells appearing to represent a predictive map influenced by transition probabilities between locations [6] [23] [8]. Grid cells, in turn, have been proposed to serve as a low-dimensional encoding of the SR matrix [23] [8]. This description of the hippocampus as the SR agent is a unifying principle which can successfully explain CA1 activity in a variety of spatial environments, including environments with reward [23] [8]. Furthermore, as the SR paradigm is based on the notion of state transitions, it is able to predict the formation of place cell representations of non-spatial states as well. For this reason, this project models the HC-EC system as encoding a successor matrix representation of the state of structured RL tasks.

It is also important to consider the processes by which place cell representations could potentially interact with the basal ganglia system. The cells in CA1, which are argued to encode the SR matrix [6], form the primary output of the hippocampal system and are known to project to the prefrontal cortex [27]. The prefrontal cortex, in turn, is a component of the hierarchical looping structure of the CBGT network [28]. From here, the place cell representations can influence every level of the CBGT hierarchy, an idea supported by the observation of place-dependent activity in primate premotor and primary sensorimotor cortex [29]. The activity of "place-value" cells could also serve as an important input for downstream calculations of expected value, and it is possible that this computation involves the orbitofrontal cortex [7]. These areas could influence the calculation of dopaminergic reward prediction errors in the substantia nigra pars compacta (SNc) through intermediate regions such as the subthalamic nucleus [3] [30].

1.4.2 Creating a Structured RL Task

A prime example of a structured RL task is that of a multi-armed bandit task with a hidden reward structure. In this project, a variation on a four-arm bandit task was employed, adapted from Bond (2022) [9]. In this task, the agent is challenged to make a sequence of six decisions. After each decision, the agent receives either a reward, a penalty, or neither, and after each block of six decisions, the agent is prompted to begin the next block. Here, the multi-armed bandit serves to obscure a latent structure, in which choosing an arm is equivalent to traveling in a certain direction on a grid. Each grid square contains either a reward or a penalty, and attempting to leave the boundaries of the map or revisit the same reward twice results in no reward. Each phase of the task uses a different map and runs for 68 blocks, in which the agent attempts to learn the structure of the environment and maximize reward. If an agent is capable of learning a latent structure which generalizes across environments, it is expected that learning in future phases will be facilitated depending on the similarity of those environments with those in previous phases. Indeed, human behavioral data indicates that learning is facilitated greatly when the new map is a rotation of the previous map, meaning that the environmental structure is preserved with simply a different action binding [9]. When the new map features a perturbation in the shape of the optimal path, which alters the underlying structure of the environment, there is much less facilitation [9].

As this task features a rich latent structure, whose grid-like nature bears more similarity to a map than to a hierarchy, it serves as an ideal example of the sort of structured RL problem for which HC-EC computations could be useful. A set of artificial place cells were developed that encoded this grid structure while obeying the physiologically-realistic predictive coding properties of SR. These cells were then used as inputs to the CBGTPy model to determine the extent to which this structural information facilitates the RL process.

1.4.3 Developing an SR Model of Place Cells

To develop a model of place cell activity patterns during the RL task, it was necessary to (1) process sequences of sensory events into a notion of state, (2) extract successor features from the state with which the SR matrix can be formed, and (3) describe the precise relationship between the SR matrix and place cell activity. To apply the SR model, the task was adapted to a sequential and symbolic format, where the symbols represented the sensory experiences associated with cues, motor actions, or rewards. To process the sequence of observed events into an encoding of environmental state, a recurrent neural network (RNN) was used. RNNs are a form of artificial neural network which possess a hidden state vector, such that the output and new state of the network at the current time step are contingent on both the current input as well as the hidden state of the network from the previous timestep [31]. As outlined in Section 3.2.3, the RNN was trained to predict future sensory events, such as rewards, from past sensory events generated by the RL task. The RNN thus learned to encode an estimate of environmental state into its hidden state vector, using only the sensory information available to the agent, albeit in a non-physiologically-realistic manner.

Once an embedding of environmental state was learned by the RNN, the next task was to extract a suitable set of basis features for the successor model. For simplicity, the basis features were formed as a linear transformation of the RNN state. Then, using the mathematical relationship between place cells and successor features described by de Cothi and Barry [8], a place cell activity pattern can be computed for each RNN state. Through gradient descent training, the final place cells were optimized to maximize their ability to predict future sensory events. Since these place cell activation patterns

are derived from an SR matrix, they are expected to behave similarly to real CA1 place cells. When these trained models were combined, the final network was capable of processing sequences of sensory events and producing realistic place cell representations at each step of the RL task.

1.4.4 Merging the SR Model with CBGTPy

The final objective of this project was to augment the complete CBGT network model with the place cell activation patterns governed by the successor representation model. To accomplish this, the SR model and the RL environment were adapted to run within the framework provided by CBGTPy, as outlined in Section 3.2.4 and Figure 3.4. Execution passed between the agent (CBGT network) and environment/SR, and a complete experimental time-course was produced as a result. During each trial, the cortical populations of the CBGT model received input signals corresponding to the current place cell vector. These cortical areas, through the corticostriatal pathways, drive activity downstream to select a single action. Furthermore, using a simple gated mixture-of-experts model, a value estimate of the state is learned and used to calculate reward prediction errors (RPEs). As the basal ganglia receives these RPEs over the course of a simulation, dopaminergic plasticity in the corticostriatal synapses shape the action policy. The behavior generated by the combined CBGT network was expected to show similar accuracy to human behavior in the latent grid task.

1.5 Summary

New environments bring new decisions, and to make these decisions in an effective manner, mammals must overcome two core challenges: first, they must assign values to their choices and use these values to guide their selection process, and second, they must leverage structural similarities to generalize from their past experiences to new contexts. While the biological goals of value learning and structure learning are simple to describe, it is unclear how the necessary computations are performed by the physiology of the cortico-basal ganglia-thalamic system. This project addresses these questions using a two-pronged computational modeling approach. First, the CBGTPy framework provides a platform for studying how realistic neural dynamics and plasticity rules in the CBGT system can lead to the proper selection of rewarding actions. Second, the computational value of hippocampal representations in RL is explored through the development of a combined model in which these representations act to bias the cortical input streams of the full CBGTPy network. While the CBGT loop alone is likely incapable of efficiently learning environmental structures, such as the structure of the latent grid task, the HC-EC system appears ideal for learning structural information and providing this information in a form that is useful for augmenting learning in other brain regions. The CBGT network model, when provided with place-cell-like cortical inputs, is designed to flexibly alter its action policy in response to changes in environmental state, reproducing the observed capabilities of humans in structured RL environments.

Chapter 2

CBGTPy: An extensible cortico-basal ganglia-thalamic framework for modeling biological decision making

This chapter is a reproduction of the text of the paper of the same name, which is currently undergoing revision for publication [24]. I would also like to acknowledge the programming and writing contributions of the co-authors: Dr. Jyotika Bahuguna, Cristina Giossi, Dr. Jonathan E. Rubin, Dr. Timothy Verstynen, and Dr. Catalina Vich.

2.1 Introduction

With the rise of fields like cognitive computational neuroscience [32], there has been a resurgence of interest in building biologically realistic models of neural systems that capture prior observations of biological substrates and generate novel predictions at the cellular, systems, and cognitive levels. In many cases, researchers rely on off-the-shelf machine learning models that use abstracted approximations of biological systems (e.g., rate-based activity and rectified linear unit gating, among others) to simulate properties of neural circuits [33–35]. For researchers interested primarily in cortical sensory pathways, these systems work well enough at making behavioral and macroscopic network predictions [36], but they often fail to provide biologically realistic predictions about underlying cellular dynamics that can be tested *in vivo*. Although there are a wealth of biologically realistic simulations of cortical and non-cortical pathways that have helped to significantly advance our understanding of BG function, these are often designed to address very narrow behaviors and lack flexibility for testing predictions across multiple experimental contexts [1, 37–40].

Here we present a scientifically-oriented tool for creating model systems that emulate the control of information streams during decision making in mammalian brains. Specifically, our approach mimics how cortico-basal ganglia-thalamic (CBGT) networks are hypothesized to regulate the evidence accumulation process as agents evaluate response options. The goal of this tool, called CBGTPy, is to provide a simple and easy-to-use spiking neural network simulator that reproduces the structural and functional properties of CBGT circuits in a variety of experimental environments. The core aim of CBGTPy is to enable researchers to derive neurophysiologically-realistic

predictions about basal ganglia dynamics under hypothesis-driven manipulations of experimental conditions.

A key advantage of our CBGTPy framework is that it separates most properties of the behaving agent from the parameters of the environment, such that experimental parameters can be tuned independently of the agent properties and vice versa. We explicitly distinguish the agent (Section 2.2.3) from the environment (Section 2.2.3). The agent generates two behavioral features – action choice and decision time – that match the behavioral data typically collected in relevant experiments and affords users the opportunity to analyze the simultaneous activity of all CBGT nuclei under experimental conditions. The flexibility of the environment component in CBGTPy allows for the simulation of both simple and complex experimental paradigms, including learning tasks with complex feedback properties, such as volatility in action-outcome contingencies and probabilistic reward scenarios, as well as rapid action control tasks (e.g., the stop signal task). On the biological side, CBGTPy incorporates biologically-based aspects of the underlying network pathways and dynamics, a dopamine-dependent plasticity rule [40], and the capacity to mimic targeted stimulation of specific CBGT nuclei (e.g., optogenetic stimulation). CBGTPy also allows the easy addition of novel pathways, as well as modification of network and synaptic parameters, so as to enable modeling new developments in the CBGT anatomy as they emerge in the literature. After a brief review of the CBGT pathways in the next subsection, in Section 2.2 we provide a full description of the structure, use, and input parameters of CBGTPy. In Section 2.3, we go on to present examples of its usage on a variety of standard cognitive tasks, before turning to a discussion in Section 2.4. Various appendices (S1, S2, S4, S5) present additional details about the CBGT model and CBGTPy toolbox, including the implementation of synaptic plasticity and a guide for CBGTPy installation.

Recent findings have suggested that the simple concepts of rigidly parallel feedforward basal ganglia (BG) pathways may be outdated [41, 42]*, and part of the motivation for CBGTPy is to provide a tool for developing and exploring more nuanced, updated theories of CBGT dynamics as new discoveries are made. Indeed, achieving a full understanding of CBGT circuit-level computations requires the development of theoretical models that can adapt with and complement the rapidly expanding empirical evidence on CBGT pathways. The fundamental goal of the CBGTPy toolbox is to provide a framework for this rapid theoretical development, which balances biological realism with computational flexibility and extensibility.

2.2 The toolbox

The core of the CBGTPy toolbox comprises an implementation of a spiking model CBGT network tuned to match known neuronal firing rates and connection patterns that have been previously used to study various aspects of basal ganglia function in cognitive tasks [21, 43–45]. The CBGT network model is composed of 6 different regions/nuclei shown in Figure 2.1: a cortical component, segregated into excitatory (Cx) and inhibitory (CxI) subpopulations; striatum, containing two subpopulations of spiny projection neuron (dSPNs involved in the so-called direct pathway, and iSPNs, involved in the indirect pathway) and also fast-spiking interneurons (FSI); external globus pallidus (GPe), which is divided into prototypical (GPeP) and arkypallidal (GPeA) subpopulations; subthalamic nucleus (STN); internal segment of globus pallidus

*We use traditional terminology of "direct" and "indirect" pathways and SPNs (e.g, Figure 2.1. While we recognize that the idea of a unified indirect pathway is outdated, it is useful to maintain a term to refer to the complement of the direct projection from dSPNs to GPi and the ascending pallidostriatal connections.

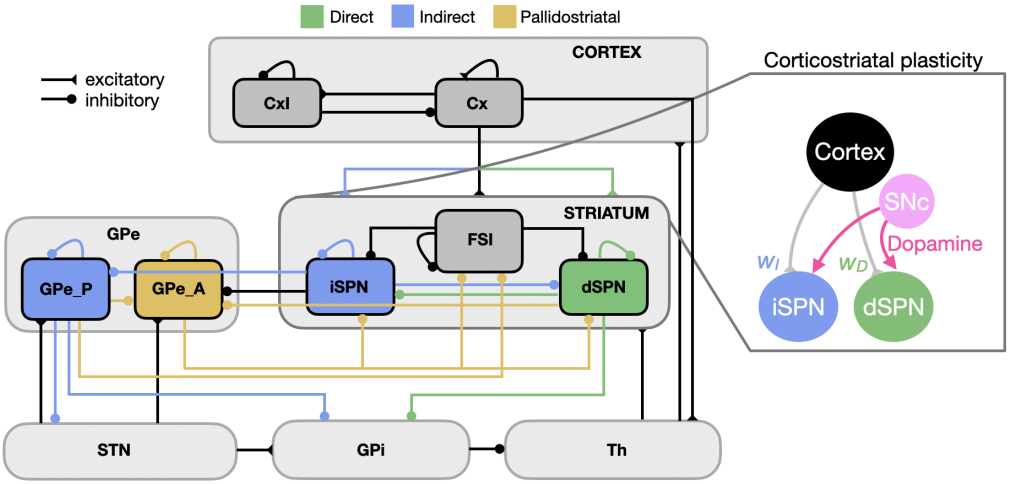


Fig 2.1. Overview of the CBGT network. The connectivity and cellular components of each CBGT channel in a CBGTPy agent are based on known biology. Direct pathway connections are shown in green, indirect pathway connections are shown in blue, and pallidostriatal connections are represented in gold, with arrows ending in circles marking the postsynaptic sites of inhibitory connections and those ending in triangles for excitatory connections. Dopaminergic feedback signals associated with rewards following actions induce plastic changes in corticostriatal synapses (pink arrows). A trial begins when the Cx population receives a stimulus and the model assumes a decision is made when the activity of the Th population reaches a certain threshold. In the current implementation of the network, the pallidostriatal pathways (gold colors) are only considered for the stop signal task.

(GPi); and a pallidal-receiving thalamic component (Th), which receives input from GPi and Cx and projects to cortical and striatal units.

Within each region, we model a collection of spiking point neurons, modeled in a variant of the integrate-and-fire framework [46] to include the spiking needed for synaptic plasticity while still maintaining computational efficiency. Numerical integration is performed via custom Cython code, rather than relying on existing frameworks, such as NEURON [47], BRIAN [48], or NetPyNE [49], a design choice which simplified the overall software stack. The core strengths of these frameworks are in the simulation of multi-scale or multi-compartment models, whereas one of the strengths of the CBGTPy model is the high level of direct control that can be exerted over the neural parameters throughout the interactions between the network and its environment (see Section 2.2.1). The integration is performed in a partially-vectorized manner, in which each variable is represented as a list of Numpy arrays, one array per neural population. Further details of the implementation of this network, including all relevant equations and parameter values, are provided in S1.

CBGTPy allows for the simulation of two general types of tasks that cover a variety of behavioral experiments used in neuroscience research. The first of these tasks is a discrete decision-making paradigm (*n-choice task*) in which the activity of the CBGT network results in the selection of one choice among a set of options (see Section 2.3.1). If plasticity is turned on during simulations, phasic dopamine, reflecting a reward prediction error, is released at the corticostriatal synapses and can modify their efficacy, biasing future decisions. We note that the inclusion of a biologically-realistic, dopamine-based learning mechanism, in contrast to the error gradient and backpropagation schemes present in standard artificial agents, represents an important

feature of the model in CBGTPy. We present the details of this learning mechanism in S2.

The second of these tasks is a stop signal paradigm (*stop signal task*), where the network must control the execution or suppression of an action, following the onset of an imperative cue (see Section 2.3.2). Here activity of the indirect and pallidostriatal pathways, along with simulated hyperdirect pathway control, determines whether a decision is made within a pre-specified time window. The probability of stop and the relevant RT distributions can be recorded across different values of parameters related to the stop signal.

2.2.1 Agent-Environment Paradigm

We have adopted an environment-agent implementation architecture, where the internal properties of the CBGT network (the agent) are largely separated from the external properties of the experiment (the environment). Interaction between the agent and environment is limited in scope, as shown in Figure 2.2, and occurs only at key time points in the model simulation. The core functionality of the agent is the mapping from stimuli to decisions and the implementation of post-decision changes (e.g., synaptic strength updates) to the CBGT network, while the environment serves to present stimuli, cues, and rewards.

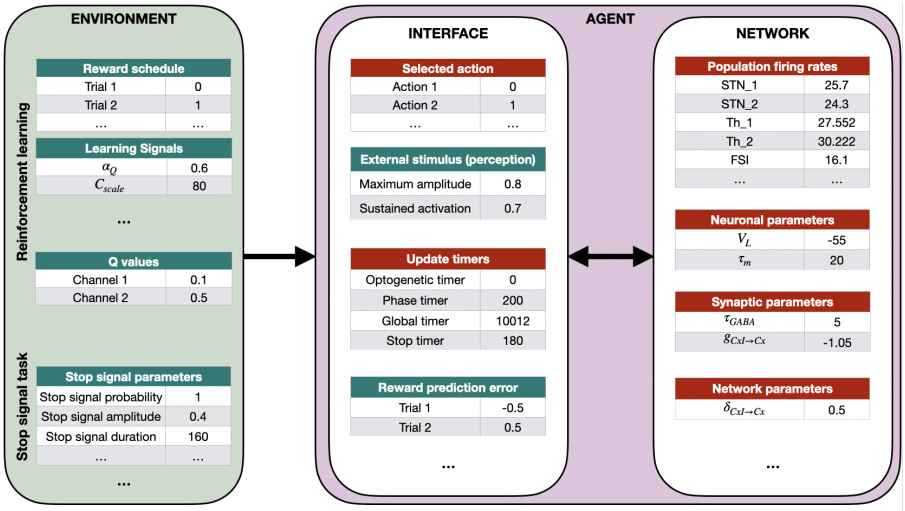


Fig 2.2. Example of interactions between the environment and the agent (CBGT network). The segregation of tasks between the environment and agent allows for independent modification of both. Tables with green title box depict some of the variables that can be easily modified by the user (see Section 2.2.3) while those in red are automatically updated or set internally. We have divided these variables into two sections: agent-related (Section 2.2.3) and environment-related (Section 2.2.3). The arrow from the environment to the agent block is unidirectional because once the stimulation starts it is not possible to change the environment. The arrow between the interface and network is bidirectional because they are always in constant interaction with each other, the interface controlling the simulation while the actual agent evolves the CBGT network.

CBGTPy uses a data-flow programming paradigm, in which the specification of computing steps is separated from the execution of those steps [50]. Internally, the initialization and simulation of the agent-environment system is divided into a large number of specialized functions, each addressing specific tasks. These code blocks are

then organized into sequences, referred to as pipelines. Only after a pipeline is constructed is it executed, transforming any input data into output data. The use of pipelines allows for individual code blocks to be rearranged, reused, and modified as necessary, leading to efficient code reuse.

One of the main benefits of the data-flow design is its synergy with the Ray multiprocessing library for Python. Ray operates on a client-server model and allows for the easy distribution of tasks and worker processes based on the available resources [51]. While the sequence of steps for running a simulation can be constructed locally, those same steps can be distributed and performed remotely on the Ray server. As a result, CBGTPy directly supports running on any system that can support a Ray server, which includes high-core-count computing clusters, while maintaining the exact same interface and ease-of-use as running simulations on a local machine. The user, however, can choose to run the model without any multiprocessing library or an alternative multiprocessing library to Ray. These options are explained in detail in the Section 2.2.2.

In the following subsections, we explain all the details of the toolbox by separately describing the agent and environmental components that can be changed by the user. The CBGTPy toolbox can be found in the Github repository `https://github.com/CoAxLab/CBGTPy/tree/main`. The instructions to install it and the list of functions contained in the toolbox can be found in S3 and S4, respectively.

2.2.2 Setting up a simulation

One of the objectives of CBGTPy is to enable end users to easily run simulations with default experimental setups. Furthermore, users can specify parameter adjustments with minimal effort, specifically through use of a `configuration` variable, which we describe in greater detail in the following sections.

The following list contains a mandatory set of instructions to be executed in order to implement the entire process associated with running a simulation. These instructions will proceed with a default set of parameters. We also provide two example notebooks (n-choice task and stop signal task), which can be found in the repository and include these steps and commands.

- Import all relevant functions.
- Create the main pipeline.
- Import the relevant `paramfile` for the selected experiment type.
- Create `configuration` dictionary with default values.
- Run the simulation, specifying which multiprocessing library to use.
- Extract relevant data frames (e.g., firing rates, reaction times, performance).
- Save variables of interest as pickle files.
- Plot variables of interest (e.g., firing rates and reward data frames).

We explain each of these steps in detail. Note that if Ray multiprocessing is being used, changing the local IP node (e.g., when the underlying Wi-Fi/LAN network has changed) requires stopping the previous instance of the Ray server and restarting it with the newly assigned IP. We also explain how to shut down the Ray server at the end of this section.

Import relevant functions. All the relevant imports can be implemented with the following commands:

```
import pandas as pd
import numpy as np
import cbgt as cbgt
import pipeline_creation as pl_creat
import plotting_functions as plt_func
import plotting_helper_functions as plt_help
import postprocessing_helpers as post_help
```

Create the main pipeline. Here the user can choose to run either the n-choice task or the stop signal task by assigning a variable `experiment_choice`[†]. Depending on the choice of this variable a relevant pipeline is created. A pipeline consists of all the modules required to run a task and returns a pipeline object that can be used.

For a basic n-choice task, the `experiment_choice` needs to be set as

```
experiment_choice = "n-choice"
```

while for a basic stop signal task, it is set as

```
experiment_choice = "stop-signal"
```

In both cases, the user also should decide how many choices or action channels the current instance of CBGTPy should create and run, using the variable `number_of_choices`:

```
number_of_choices = 2
```

While a common version of this decision making task is run with `number_of_choices` $= 2$, it can also be run for an arbitrary number of choices (i.e., `number_of_choices` ≥ 1). The change in the number of choices and corresponding action channels requires scaling of some of the parameters in order to ensure maintenance of the same amount of input to certain shared CBGT nuclei irrespective of the number of action channels. We explain these scaling schemata in S5. Please note that some parameters have to be appropriately updated in the configuration variables (e.g., Q data frame, channel names) according to the `number_of_choices` selected. We explicitly mention which parameters should be updated with the number of choices as we describe them below, and we include example notebooks in the repository for different cases.

The pipeline is created with commands

```
pl_creat.choose_pipeline(experiment_choice)
pl = pl_creat.create_main_pipeline(runloop=True)
```

Import the relevant `paramfile` for the selected experiment type. The `paramfile` contains dictionaries of default parameter values for the neural populations and plasticity model based on the choice of the experiment.

```
if experiment_choice == 'stop-signal':
```

[†]If the variable `experiment_choice` is not set, the pipeline creation gives an error because of the ambiguity of which experiment to run.

```
    import stopsignal.paramfile_stopsignal as paramfile
elif experiment_choice == 'n-choice':
    import nchoice.paramfile_nchoice as paramfile
```

The imported attributes, which can be listed out using `dir(paramfile)`, can be modified according to the user's preferences. For example, setting the cellular capacitance value to 0.5 is accomplished with

```
paramfile.celldefaults['C'] = 0.5
```

Create `configuration` dictionary with default values. The `configuration` variable is a dictionary in which some parameters take internally set default values, whereas others need to be assigned values to run a simulation. A minimal `configuration` variable consists of the following parameters, the details of which are described in S9 Table, S10 Table, S11 Table, S12 Table, S13 Table, S14 Table and explained separately in Section 2.2.3.

```
configuration = {
    "experimentchoice": experiment_choice,
    "seed": 0,
    "inter_trial_interval": None, # default = 600ms
    "thalamic_threshold": None, # default 30sp/s
    "movement_time": None, # default sampled from N(250,1.5)
    "choice_timeout": None, # default 1000
    "params": paramfile.celldefaults,
    "pops": paramfile.popspecific,
    "receps": paramfile.receptordefaults,
    "base": paramfile.basestim,
    "dpmns": paramfile.dpmndefaults,
    "dSPN_params": paramfile.dSPNdefaults,
    "iSPN_params": paramfile.iSPNdefaults,
    "channels": pd.DataFrame([["left"], ["right"]], columns=["action"]),
    "number_of_choices": number_of_choices,
    "newpathways": None,
    "Q_support_params": None,
    "Q_df": None,
    "n_trials": 3,
    "volatility": [1,"exact"],
    "conflict": (1.0, 0.0),
    "reward_mu": 1,
    "reward_std": 0.1,
    "maxstim": 0.8,
    "corticostriatal_plasticity_present":True,
    "record_variables": ["weight", "optogenetic_input"],
    "opt_signal_present": [True],
    "opt_signal_probability": [[1]],
    "opt_signal_amplitude": [0.1],
    "opt_signal_onset": [20.],
    "opt_signal_duration": [1000.],
    "opt_signal_channel": ["all"],
    "opt_signal_population": ["dSPN"],
    "sustainedfraction": 0.7
}
```

Note that the parameter `corticostriatal_plasticity_present` does not have to

be introduced in the configuration dictionary when running the stop signal task (see reference on the example notebook). Additionally, it is important to include the stop signal parameters within the `configuration` dictionary when executing the stop signal task.

```
configuration = {
    "stop_signal_present": [True, True],
    "stop_signal_probability": [1., 1.],
    "stop_signal_amplitude": [0.6, 0.6],
    "stop_signal_onset": [60.,60.],
    "stop_signal_duration": ["phase 0",165.],
    "stop_signal_channel": ["all","left"],
    "stop_signal_population":["STN","GPeA"],
}
```

More details will be provided in the corresponding sections. For reference, you can find an example notebook on github.

Run the simulation At this stage, the user can choose the number of cores to be used (`num_cores`) and the number of parallel simulations that should be executed with the same `configuration` variable but a different random seed (`num_sims`). Moreover, the user can optionally specify one of the two supported multiprocessing libraries, Ray and Pathos, to use to run the simulation. Ray is a library providing a compute layer for parallel processing. To start the Ray server, on the command line, run Ray server to execute the head node and obtain the local IP node, in the following way:

```
ray start --head --port=6379 --redis-password="cbgt"
```

This command should list a local IP along with a port number. Hence, to initiate a Ray client that connects to the server started above, the user will have to substitute the local IP node, obtained from the previous command line, in place of `<local ip node>` in

```
ray start --address=<local ip node>:6379 --redis-password="cbgt"
```

Any port number that is free to use in the machine can be used. Here port number 6379 is used, which is the default port number for Ray. To use Ray, the last step consists of setting the variable `use_library` in the notebook to `'ray'`. As an alternative to Ray, Pathos is a library that distributes processing across multiple nodes and provides other convenient improvements over Python's built-in tools. To use Pathos, no additional setup is required beyond setting the variable `use_library` to `'pathos'`. If the user does not want to use any of the above-mentioned libraries, this should be specified by setting the variable `use_library` to `'none'`. The simulation is performed by filling in values for these variables in the following command and executing it:

```
results = cbgt.ExecutionManager
(cores=num_cores,use=use_library).run([pl]*num_sims,[configuration]*num_sims))
```

To ensure the simulation results are both reproducible and robust to minor changes in initial conditions, CBGTPy offers control over the pseudorandom number generator seed. The random seed controls the initial conditions of the network, including precisely which neurons connect together, under the constraint of the given connection probabilities and the baseline background activity levels of the CBGT nuclei. The simulation returns a `results` object, which is a dictionary containing all the data produced by the model, typically organized into data frames [52].

Extract relevant data frames Once the `results` object has been returned, specific variables and tables of interest can be extracted (see S2 Fig). All the variables available can be listed by accessing the keys of the variable `results`, which is done by executing the following command:

```
results[0].keys()
```

All environmental variables passed to the simulation can also be accessed here for cross-checking.

Some additional data frames related to the simulation are also returned. One of these data frames is `results[0]["popfreqs"]`, which returns the population firing rate traces of all nuclei, with each neuronal subpopulation as a column and each time bin of simulated time as a row (see S3 Fig). This data frame can be addressed directly by executing its name in a command line.

Another relevant data frame is `datatables[0]`, which contains a list of chosen actions, optimal actions, reward outcomes, and decision times for all of the trials in the simulation (see S4 Fig). When running multiple simulations in parallel (i.e., `num_sims` > 1), `datatables[i]` is returned, where i indicates the corresponding thread. This data frame can be extracted by first executing the command

```
datatables = cbgt.collateVariable(results,"datatables")
```

and next typing `datatables[i]` on the command line, to access the results of the i^{th} simulation.

As part of the model's tuning of dopamine release and associated dopamine-dependent corticostriatal synaptic plasticity, the model maintains Q-values for each action, updated according to the Q-learning rule (details in S2). These values are available in `results["Q_df"]`, where each column corresponds to one of the possible choices (see S5 Fig). These data frames are designed for easy interpretation and use in later data processing steps. It should be however noted that while Q-values are updated and maintained by CBGTPy, the q-values do not influence the selection of decision choice. The selection of decision choice is solely dependent on the corticostriatal weights.

CBGTPy also provides a function to extract some of these data frames in a more processed form. The specific command for the n-choice task is given by

```
firing_rates, reward_q_df, performance, rt_dist, total_performance =
plt_help.extract_relevant_frames(results, seed, experiment_choice)
```

where `firing_rates` provides a stacked up (pandas command `melt`) version of the `results[0]["popfreqs"]` that can be used in the *seaborn.catplot()* plotting function, `reward_q_df` compiles data frames for reward and q-values, `performance` returns the percentage of each decision choice, `rt_dist` returns the reaction time distribution for the simulation, and `total_performance` compares the `decision` and `correctdecision` in `datatables[i]` and calculates the performance of the agent. Depending on the experiment choice, the function returns relevant data frames. Note that for the stop signal task, the data frames returned are just `firing_rates` and `rt_dist`.

The time-dependent values of the recorded variables can also be extracted for both the n-choice task and the stop signal task using the following command:

```
recorded_variables =
post_help.extract_recording_variables(results,
                                      results[0]["record_variables"],
                                      seed)
```

Presently, for the n-choice task, CBGTPy only allows recording the variable `weight` or `optogenetic_input`. The former can be used to track the evolution of corticostriatal weights during a n-choice experiment. The variable `optogenetic_input` can be recorded and plotted to check if the optogenetic input was applied as intended to the target nuclei. The list of variables to be recorded should be specified in the configuration variable. In the example of the configuration variable used above, both `weight` and `optogenetic_input` are recorded.:

```
configuration = {
..
"record_variables": ["weight", "optogenetic_input"],
..
}
```

An example of plotting these data frames is included in the example python notebook in the GitHub repository. In addition, for the stop signal task, CBGTPy also allows the recording of the variable `"stop_input"`, which can be used to check if the stop signal inputs were applied correctly to the target nuclei.

Save variables of interest as pickle files All the relevant variables can be compiled together and saved in a single pickle file. Pickle files provide a method for saving complex Python data structures in a compact, binary format. The following command saves the object `results` with additional data frames of `popfreqs` and `popdata` into a pickle file `network_data` in the current directory:

```
cbgt.saveResults(results, "network_data", ["popfreqs","popdata"])
```

Basic plotting functions (plot firing rates and reward data frame) CBGTPy provides some basic plotting functions. The `firing_rates` data frame from the above functions can be passed to function `plot_fr`, which returns a figure handle that can be saved (see Figure 2.3), as follows:

```
FR_fig_handles = plt_func.plot_fr(firing_rates, datatables, results,
    experiment_choice, display_stim)
```

In addition to the `firing_rates` data frame, the plotting function `plot_fr` requires the `datatables` (also extracted along with `firing_rates` data frame), the original `results` variable, `experiment_choice` and `display_stim`. The `experiment_choice` ensures that relevant nuclei are plotted and the `display_stim` is a boolean variable that can be set to True/False. When set to True, the stimulation information (e.g., optogenetic or stop signal) is indicated over all trials during which the stimulation was applied. Note that, for a longer simulation, this may slow the plotting function, because the function checks if a stimulation is applied on every trial before indicating the result in the figure. The stop signal application is indicated as a bright horizontal red bar above firing traces of the stimulated nuclei (e.g., Fig 2.7). The optogenetic stimulation is indicated as a blue bar for excitatory stimulation and yellow for inhibitory stimulation (e.g., Fig 2.8).

The reward and Q-values data frame can be plotted with the function `plot_reward_Q_df` as follows; note that a figure is shown in the example notebook:

```
reward_fig_handles = plt_func.plot_reward_Q_df(reward_q_df)
```

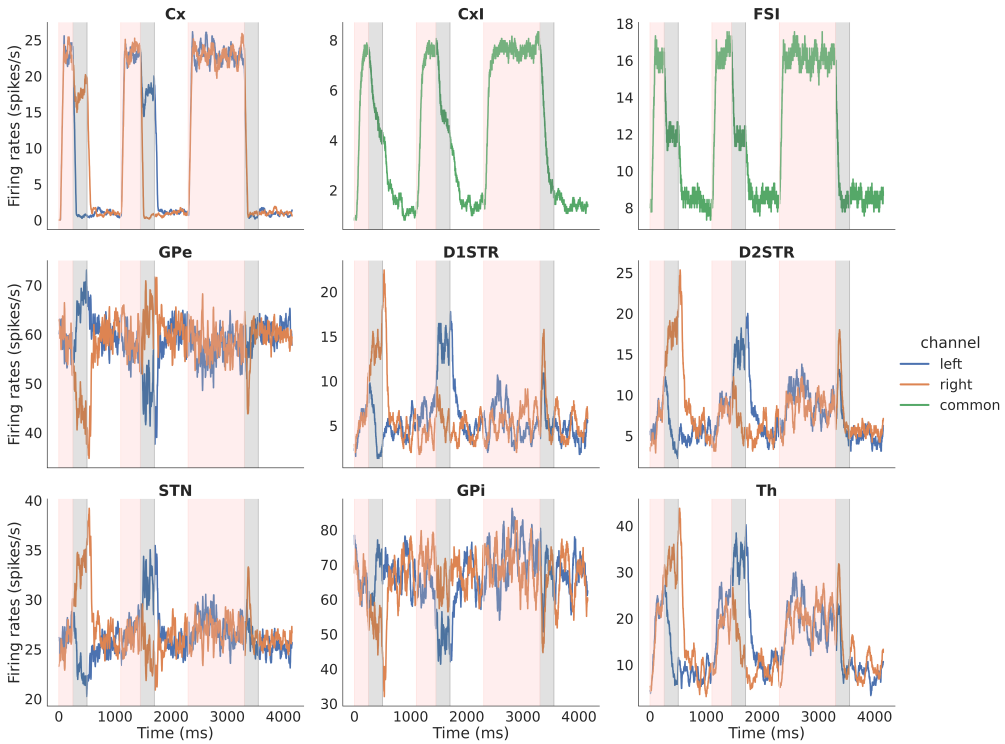


Fig 2.3. Example figure showing firing rates for all the nuclei in the n-choice task. Example figure showing firing rates for all nuclei for three consecutive trials of a 2-choice task, color-coded to distinguish times associated with decision making (pink, *decision phase*) and subsequent times of sustained activity in the selected channel (grey, *consolidation phase*) of each trial. The unshaded regions between each pair of trials are the (*inter-trial interval*). In each panel, the blue (orange) trace corresponds to activity in the left (right) action channel. In this example, the model chooses right on the first trial and left on the second; in the third trial, where decision making times out, no sustained activation is applied to the cortical channel (top left subplot) during the *consolidation phase* (grey region).

Shut down Ray server In case the Ray server was selected as the multiprocessing library to use to run the simulation, when the user is done working with the network, the Ray server can be shut down via the terminal with the command

```
ray stop
```

or, if the processes have not all been deactivated, with the command

```
ray stop --force
```

2.2.3 User level modifications

CBGTPy allows for modifications of several parameters that a user can easily perform. All the parameters in the `configuration` variable can be modified. A modified value is usually in the form of a data frame or a dictionary. The underlying function (`ModifyViaSelector`) in `frontendhelpers.py` iterates through all the features listed in the data frame/dictionary and updates the default values with the new

values passed to the `configuration` variable. If the user wants to use the default value of a parameter, it is essential to specify the parameter value as `None`, as also shown in Section 2.2.2. We can subdivide the parameters that can be modified into two major classes, which we will discuss separately: a) parameters related to the agent, which we call the agent parameters (Section 2.2.3); b) parameters related to the experimental environment, which we call the environmental parameters (Section 2.2.3).

Through the adjustment of the appropriate parameters, the user can adapt the model to study a variety of important scientific questions. For example, one could introduce new neural pathways and vary their connectivity to study the effects on the system's dynamics and behavior (e.g., addition of a cortico-pallidal pathway, which may complement stopping mechanisms in the BG pathways [53]). Alternatively, the user could study the effects of specific neural parameters on decision-making and stop signal tasks. Additionally, by introducing optogenetic stimulation to different populations, such as dSPN and iSPN striatal populations, the user could model how the timing and intensity of this stimulation influences the plasticity and learning processes. For example, it has been shown that inhibition of dSPNs during learning impairs performance in a goal-directed learning task [54]. CBGTPy allows stimulation of multiple populations at different phases of a task, which enhances the options for exploring possible functional pathways and their roles in task performance. Lastly, CBGTPy allows many environmental parameters to be modified for an n-choice or stop-signal task while simulating the activity of the CBGT network. For example, one could test the idea that the slowing down of decision times in healthy humans [55] when there is a high conflict or similarity between choices is related to increased activity in the STN [56]. Taken together, the large number of both network and environmental parameters available for the end user to control is a strength of the CBGTPy framework and greatly increases its ultimate scientific utility.

A detailed list of features of the CBGTPy package that can be easily modified by the user can be found in table S8 Table.

Agent parameters

General neuron parameters The parameters common to all neurons can be modified using the `params` field in the dictionary `configuration`. A complete list of editable neuronal parameters is listed in S9 Table. For example, the following dictionary entry can be modified to change the capacitance (`C`) of all neurons:

```
"params": pd.DataFrame([[30]], columns=["C"]),
```

Population-specific neuronal parameters The neuronal parameters of a specific population can be modified using the field `pops` in the dictionary `configuration`. These parameters will override the default values set by the `params` field. A complete list of editable parameters is given in S10 Table. In the following example, the membrane time constant (`Taum`) of the neurons in the FSI population is specified:

```
"pops": {"FSI": {"Taum": [60]}},
```

Synaptic parameters The parameters of the synapses (GABA, AMPA or NMDA) can be modified through the field `receps` in the `configuration` variable. A complete list of editable synaptic parameters is given in S11 Table. In the following example, the membrane time constants of AMPA and GABA synapses (`Tau_AMPA`, `Tau_GABA`) are specified:

```
"receps": pd.DataFrame([[100, 100]], columns=["Tau_AMPA", "Tau_GABA"]),
```

Population-specific baseline input parameters Each neuron receives a background input from a random Gaussian process with a specified mean frequency and variance equal to 1. Depending on the nature of the background input, the Gaussian process can be excitatory (AMPA and NMDA) or inhibitory (GABA). The user can specify the mean frequency, the efficacy, and the number of connections from the background Gaussian process to the neurons in the population with the dictionary `base`. A complete list of editable input parameters is given in S12 Table. In the following example, the frequency of the external AMPA inputs (`FreqExt_AMPA`) applied to FSI neurons is specified:

```
"base": {"FSI": {"FreqExt_AMPA": [100]}},
```

Dopamine-specific parameters The dopamine-related parameters can be modified via the `dpmns` parameter, which takes as its input a data frame containing the field name and the names of the parameters to be updated. A complete list of these editable parameters is given in S13 Table. In the following example, the dopamine decay rate (`dpmn_tauDOP`) is specified:

```
"dpmns": pd.DataFrame([[5]], columns=["dpmn_tauDOP"]),
```

SPN-specific dopaminergic parameters The SPN-specific dopaminergic parameters for corticostriatal projections to dSPNs and iSPNs can be modified via `d1` and `d2`, respectively, which take as their input a data frame containing the field name and the parameters to be updated. The complete list of these editable parameters is given in S14 Table. In the following example, the learning rate (`dpmn_alphaw`) and the maximal value for the corticostriatal weights (`dpmn_wmax`) are specified:

```
"dSPN_params": pd.DataFrame([[39.5, 0.08]], columns=["dpmn_alphaw",
    "dpmn_wmax"]),
"iSPN_params": pd.DataFrame([[-38.2, 0.06]], columns=["dpmn_alphaw",
    "dpmn_wmax"]),
```

New pathways The parameters of a specific pathway can be changed by using the variable `newpathways`. This variable can also be used to add new connections. This takes as its input a data frame that lists the following features of the pathway: source population (`src`), destination population (`dest`), receptor type (`receptor`), channel-specific or common (`type`), connection probability (`con`), synaptic efficacy (`eff`) and the type of the connection (`plastic`), which can be plastic (`True`) or static (`False`). An example of a cortico-pallidal pathway involving AMPA synapses with 50% connection probability, synaptic strength 0.01, and no plasticity is presented in the following dictionary entry:

```
"newpathways": pd.DataFrame([["Cx", "GPe", "AMPA", "syn", 0.5, 0.01, False]])
```

If the user wishes to change multiple pathways at once, then the variable can be given a list of data frames as input.

Q-learning process CBGTPy uses Q-learning to track the internal representations of the values of the possible choices, which depend on the rewards received from the environment. The parameters of this process can be modified via the variable `Q_support_params`. The two parameters that can be modified are `C_scale` and `q_alpha`. The former controls the scaling between the change in phasic dopamine and the change in weights, and the latter controls the change in choice-specific q-values with reward feedback from the environment. An example of how to specify these two parameters is presented in the following dictionary entry:

```
"Q_support_params": pd.DataFrame([[30, 0.1]], columns=["C_scale", "q_alpha"]),
```

The equations showing the roles of these parameters are described in detail in S2.

Q-values data frame The choices available to the agent can be initialized with identical values (e.g., 0.5) representing an unbiased initial condition. Alternatively, non-default values for the Q-values data frame (such as values biased towards one choice) can be initialized using the variable `Q_df`. An example of how to specify this variable is presented in the following dictionary entry:

```
"Q_df_set": pd.DataFrame([[0.3, 0.7]], columns=["left", "right"]),
```

Note that this parameter should be updated according to the number of choices specified in the variable `number_of_choices`. The above example shows an initialization for a 2-choice task.

Cortical activity In addition to the background inputs that generate the baseline activity of all of the CBGT nuclei, the cortical component provides a ramping input to the striatal and thalamic populations, representing the presence of some stimulus or internal process that drives the consideration of possible choices. The maximum level of this input can be defined by the parameter `maxstim` and specified using the following dictionary line:

```
"maxstim": 0.8,
```

Corticostriatal plasticity The corticostriatal plasticity in the n-choice experiment can be switched on and off using the `corticostriatal_plasticity_present` parameter. When it is set to `True`, the corticostriatal weights change based on rewards and dopaminergic signals (for more details see S2). When it is set to `False`, the simulation proceeds without any update in the corticostriatal weights and the Q-values. The value of this parameter is set to `True` using the following dictionary line:

```
"corticostriatal_plasticity_present": True,
```

Sustained activation to the action channel for the selected choice In order to resolve the temporal credit assignment problem [21], we rely on post-decision sustained activation to keep the selected channel active during the phasic dopaminergic activity [57, 58]. After the choice has been made, following the onset of a trial (*decision phase*), the cortical component of the action channel associated with the selected choice continues to receive inputs, while the unselected channels do not (*consolidation phase*); see Figure 2.3. This phase may also represent the movement time of the agent. The assumption here is that this activation provides an opportunity for corticostriatal

plasticity that strengthens the selected choice. The parameter `sustainedfraction` is the fraction of input stimulus maintained during the consolidation phase in the cortical channel corresponding to the action selected by the agent, and it can be specified using the following dictionary entry:

```
"sustainedfraction": 0.7,
```

Once the dopamine signal has been delivered at the end of the *consolidation phase*, all cortical inputs are turned off for the *inter-trial interval*. See Section 2.3.1 and Fig 2.4 for more details on the trial phases.

Thalamic threshold In the default set-up, when the thalamic firing rate of either choice reaches 30 Hz, that choice is selected. This threshold can be specified by the user by setting the parameter `thalamic_threshold` in the following way:

```
"thalamic_threshold": 30,
```

Environment parameters

Experiment choice The parameter `experiment_choice` is set at the beginning of the simulation (see Section 2.2.2). It also needs to be sent as a configuration variable, so that the specific functions and network components relevant to the appropriate experiment are imported.

Inter-trial interval The parameter `inter_trial_interval` allows the user to specify the inter-trial interval duration. The inter-trial interval also corresponds to the duration of the *inter-trial-interval phase* of the simulation, where the network receives no external input and shows spontaneous activity. When no value is specified (`None`), a default value of 600 ms is used. The user can set the value of this parameter using the following dictionary entry:

```
"inter_trial_interval": 600,
```

Movement time After a choice is made, the chosen action channel receives sustained activation at some fraction (with a default value of 70%) of the initial cortical input strength. As noted in the previous section, this phase of the simulation (*consolidation phase*) represents the movement time, which is distinct from the reaction time, provides a key window for corticostriatal synaptic plasticity to occur, and remains unaffected by the selected choice. The length of this phase can be controlled with the parameter `movement_time`. The default value of the movement time (when this parameter is set to `None`) is sampled from a normal distribution $\mathcal{N}(250, 1.5)$. However, the user can choose to set it to a constant value by passing a list `["constant", N]`, where `N` represents the constant value of movement time for all trials. The other option is to sample from a normal distribution of specified mean `N` using `["mean", N]`. The movement time can be set to a fixed value as follows:

```
"movement_time": ["constant", 300],
```

Choice time out The parameter `choice_timeout` controls the duration of the time interval in which a choice can be made. The default value of this parameter (when it is set to `None`) is 1000 ms. This parameter can be changed as follows:

```
"choice_timeout": 300,
```

Choice labels The data frame `channels` allows the labels for the action channels to be changed. The new labels can be used to access information about the action channels. An example is shown below:

```
"channels": pd.DataFrame([["left"], ["right"]], columns=["action"]),
```

Note that this parameter should be updated according to the number of choices specified in the variable `number_of_choices`. The above example shows an initialization for a 2-choice task.

Number of trials The `n_trials` parameter sets the number of trials to be run within a simulation. Note that this number should be greater than the `volatility` parameter (described in the following paragraph). However, if only 1 trial is to be simulated, then `volatility` parameter should be set to `None`. Examples of how to set this parameter are as follows:

```
"n_trials": 2,
```

```
"n_trials": 1,
...
"volatility": [None, "exact"]
```

For more details about setting `volatility` parameter, please refer to the following paragraph.

Volatility The parameter `volatility` indicates the average number of trials after which the reward contingencies switch between the two choices. The volatility parameter is a list consisting of two values, [λ, 'option'], where `option` can be set as `exact` or `poisson`. The λ parameter generates a reward data frame where the reward contingency changes after an average of λ trials. The option `exact` ensures that the reward contingency changes exactly after λ trials whereas the option `poisson` samples the change points from a Poisson distribution with parameter λ. However, note that this parameter cannot be 0 or the total number of trials. To perform a simulation in which the reward contingencies do not change until the end of the simulation, set this parameter to `n_trials`-1 and drop the last trial from the analysis. An example of how to define and specify the volatility is shown in the following command line:

```
"volatility": [2, "exact"],
```

Note that for a 1-choice task or stop signal task, the volatility parameter is not applicable and hence should be defined differently; specifically, the parameter λ should be set to None as follows:

```
"volatility": [None, "exact"],
```

Reward probability The parameter `conflict` represents the reward probability of the reward data frame and is defined as a tuple of reward probabilities for the n choices. In the following example, for a 2-choice task, the first reward probability corresponds to

the first choice listed in the `channels` parameter (e.g., `"left"`). The reward probabilities for the choices are independent, thereby allowing reward structure to be set in the format (p1, p2) as in the two following examples, representing unequal and equal reward probabilities, respectively:

```
"conflict": (0.75, 0.25),
"conflict": (0.75, 0.75),
```

Note that this parameter should be updated according to the number of choices specified in variable `number_of_choices`. The above example shows an initialization for a 2-choice task. For example, for a 3-choice task the reward probabilities can be defined as:

```
"conflict": (1.0, 0.5, 0.2),
```

Reward parameters The trial-by-trial reward size is generated by a random Gaussian process, with a mean (`reward_mu`) and a standard deviation (`reward_std`) that can be assigned. To simulate binary rewards, choose mean = 1 and standard deviation = 0, as follows:

```
"reward_mu": 1,
"reward_std": 0.0,
```

Optogenetic signal If the experiment requires an optogenetic signal to be applied, then this should be indicated in the `configuration` variable by setting `opt_signal_present` to `True`, with each boolean variable corresponding to each nucleus specified in the list of populations to be stimulated, as shown below:

```
"opt_signal_present": [True],
...
..
"opt_signal_population": ["dSPN"],
```

The above example shows the case, when only one population (i.e dSPNs) is stimulated. More than one population can be stimulated simultaneously as shown below:

```
"opt_signal_present": [True, True],
...
..
"opt_signal_population": ["dSPN", "iSPN"],
```

Optogenetic signal probability The `opt_signal_probability` parameter accepts either a float or a list. The float represents the probability of the optogenetic signal being applied in any given trial. For example, the user wants all the trials in the simulation to be optogenetically stimulated, i.e `opt_signal_probability` = 1.0, it should be defined as shown below:

```
"opt_signal_probability": [1.0],
```

Alternatively, a specific list of trial numbers during which optogenetic stimulation should be applied can also be passed. An example is shown below:

```
"opt_signal_probability": [[0, 1]],
```

In the above example, a list of trial numbers `[0, 1]` indicates that the optogentic stimulation is applied to trial numbers 0 and 1.

Please note that if more than one nucleus will be stimulated, the `opt_signal_probability` expects a list of floats (probabilities) or list of list (list of trial numbers). For example, as mentioned in an example above, say the user wants to stimulate two populations (dSPN and iSPN) at trial numbers [0,1] and [1,2] respectively:

```
"opt_signal_present": [True, True],
..
"opt_signal_population": ["dSPN", "iSPN"],
"opt_signal_probability": [[0, 1],[1, 2]],
```

Optogenetic signal amplitude The amplitude of the optogenetic signal can be passed as a list of floats to the parameter `opt_signal_amplitude`. A positive value represents an excitatory optogenetic signal, whereas a negative value represents an inhibitory optogenetic signal. An example of excitation is shown below:

```
"opt_signal_amplitude": [0.1],
```

If we want to send different amplitudes of optogenetic signals to different populations, for example, an excitatory (0.3) to dSPNs and inhibitory (-0.25) to iSPNs, then the amplitudes should be specified as :

```
"opt_signal_present": [True, True],
..
"opt_signal_population": ["dSPN", "iSPN"],
"opt_signal_amplitude": [0.3, -0.25],
```

Optogenetic signal onset The `opt_signal_onset` parameter sets the onset time for the optogenetic signal. The onset time is measured relative to the start of the *decision phase*. For example, to specify that optogenetic stimulation will start $20\,ms$ after the *decision phase* begins, the appropriate command is:

```
"opt_signal_onset": [20.],
```

Optogenetic signal duration The duration for which the optogenetic signal is applied can be controlled by the parameter `opt_signal_duration`. This parameter accepts a numerical value in *ms* as well as phase names as strings. For example, to apply the optogenetic stimulation for $1000\,ms$ after the signal onset, the command is:

```
"opt_signal_duration": [1000.],
```

However, in order to apply optogenetic stimulation during the whole *decision phase*, the duration variable should be the string "phase 0" as shown below:

```
"opt_signal_duration": ["phase 0"],
```

This allows the user to specifically target *decision* ("phase 0"), *consolidation* ("phase 1") and *inter-trial interval* ("phase 2") phases with optogenetic stimulation.

Optogenetic signal channel The user can also control whether the optogenetic signal is applied globally to all action channels (`all`), to a randomly selected action channel (`any`), or to a specific action channel (for instance, `left`). To do so, the parameter `opt_signal_channel` needs to be specified as in the example below:

```
"opt_signal_channel": ["all"],
```

Optogenetic signal population The optogenetic stimulation can be applied to a single or multiple populations in the same simulation. In either case, the population names should be defined as a list. The target population can be specified using the parameter `opt_signal_population`. In the example below, the dSPN population is set as the target population:

```
"opt_signal_population": ["dSPN"],
```

Although the optogenetic-related parameters can be used to mimic the effect of a stop signal manifested as the application of a step input current to a population, the network also has a number of modifiable parameters that are specific to injecting the stop signal to target nuclei via a box-shaped current.

Recorded variables CBGTPy allows recording of time-dependent values of both the corticostriatal weights and any optogenetic inputs to any CBGT nuclei that are being stimulated by using the parameter `record_variables`. The first component of `record_variables` can be used to track the evolution of the weights from cortex to dSPNs or iSPNs during an n-choice experiment. The latter component records the optogenetic input applied to the target population and is especially useful for debugging purposes. Both of these variables can be extracted as a data frame by calling the function `extract_recording_variables` as described in Section 2.2.2. Note that, for the parameter `weight`, cortical weights to both dSPNs and iSPNs for all choice representations (channels) are recorded. In addition, when running the stop signal task, the stop signal inputs can be recorded as well. Here, we can see an example of how to extract the weights and the optogenetic input:

```
"record_variables": ["weight", "optogenetic_input"],
```

In addition, for the stop signal task, CBGTPy also allows the recording of the variable `"stop_input"`, which can be used to check if the stop signal inputs were applied correctly to the target nuclei.

```
"record_variables": ["stop_input"],
```

Stop signal If the experiment requires the stop signal to be applied, then this option should be selected in the `configuration` variable by setting `stop_signal_present` to `True`. Different stop signals can be applied to different target populations during the same execution. For these, the `stop_signal_present` variable is defined as a list whose length depends on the number of target populations. For example, if we apply stop signals to two different populations, we have to set this variable as follows:

```
"stop_signal_present": [True, True],
```

Note that the user can apply the stop signals to as many nuclei as desired.

Stop signal populations The target populations for the stop signal can be specified using the parameter `stop_signal_population`. In the example below, the STN and GPeA populations are set as the target populations:

```
"stop_signal_populations": ["STN", "GPeA"],
```

All the examples presented below corresponding to the stop signal task are designed taking into account that the stop signal is injected into these two populations.

Stop signal probability The stop signal probability can be specified using the parameter `stop_signal_probability`, which is a list whose entries can take a float or a list as input. If a float (between 0 and 1) is introduced, then this value represents the probability to which the stop signal is applied. These trials are picked randomly from the total number of trials. Alternatively, if a sublist is specified, then it must contain the numbers of those trials where the user wants the stop signal to be applied. Note that the entries in the main list refer to the populations specified in the same order as in the variable `stop_signal_populations`. For example, within the statement

```
"stop_signal_probability": [1.0, [2, 3, 6]],
```

the first float value represents a 100% probability of applying the "first" stop signal to the first specified nucleus (STN), while the subsequent list of values represents the numbers of the trials on which the "second" stop signal will be applied to the other target region (GPeA).

Stop signal amplitude The amplitude of the stop signal can be passed as a float to the parameter `stop_signal_amplitude`. Note that this parameter is a list and that every value refers to the corresponding population. The order should follow the order of the populations. For example,

```
"stop_signal_amplitude": [0.4, 0.6],
```

Stop signal onset The parameter `stop_signal_onset` sets the times when the stop signals are injected into the target nuclei. The onset time is measured with respect to the start of the *decision* phase. In the example proposed below, the stop signal stimulation at the STN starts 30 ms after the *decision* phase begins, while that applied to the GPeA starts 60 ms after the *decision* phase begins:

```
"stop_signal_onset": [30., 60.]
```

Stop signal duration How long each stop signal is maintained can be controlled using the parameters `stop_signal_duration`. As in the previous cases, this is a list whose order must follow that of the target populations. In the following example, a stop signal lasting 100 ms is applied to the STN while another with duration 160 ms is applied to the GPeA:

```
"stop_signal_duration": [100., 160.]
```

In order to apply the stop signal throughout an entire phase, the duration variable should be set to a string containing the name of the phase when the user wants to apply the stop signal, as shown below:

```
"opt_signal_duration": ["phase 0", "phase 1"],
```

This allows the user to specifically target *decision* ("phase 0"), *consolidation* ("phase 1") and *inter-trial interval* ("phase 2") phases with persistent stop signal stimulation.

Stop signal channel The user can also control if the stop signal applied to a specific population is presented to all action channels (`all`), to a uniformly and randomly picked action channel (`any`), or to a specific action representation (for instance, `left`). These can be specified using the parameter `stop_signal_channel`. In the following example, one stop signal is presented to the STN populations of all of the action channels, while the second stop signal is applied only to the GPeA population corresponding to the "left" action channel:

```
"stop_signal_channel": ["all", "left"]
```

2.3 Experiments

In this section, we present some details about examples of the two primary experiments that CBGTPy is designed to implement.

2.3.1 An n-choice task in an uncertain environment

This task requires the agent to select between n choices (e.g., left/right in a 2-choice task). The selection of each choice leads to a reward with a certain probability. Moreover, the reward probability associated with each choice can be abruptly changed as part of the experiment. Thus, there are two forms of environmental uncertainty associated with this task: a) *conflict*, or the degree of similarity between reward probabilities; and b) *volatility*, or the frequency of changes in reward contingencies. Higher conflict would represent a situation where the reward probabilities are more similar across choices, making detection of the optimal choice difficult, whereas a lower conflict represents highly disparate values of reward probabilities and easier detection of the optimal choice. Conflict is not specified directly in CBGTPy; rather, the reward probabilities are explicitly set by the user (Section 2.2.3). An environment with high (low) volatility corresponds to frequent (rare) switches in the reward contingencies. The volatility can be set by the parameter λ, which determines the number of trials before reward probabilities switch. The user can choose between whether the trials are switched after exactly λ trials or whether switches are determined probabilistically, in which case λ represents the rate parameter of a Poisson distribution that determines the number of trials before reward probabilities switch (Section 2.2.3).

Using the reward probabilities and volatility, the backend code generates a reward data frame that the agent encounters during the learning simulation. The reward data frame is used in calculating the reward prediction error and the corresponding dopaminergic signals, which modulate the plasticity of the corticostriatal projections.

At the beginning of the simulation, the CGBT network is in a resting phase during which all CBGT nuclei produce their baseline firing rates. When a stimulus is presented, the network enters a new phase, which we call *phase 0* or *decision* phase. We assume that at the start of this stage a stimulus (i.e., an external stimulus, an internal process, or a combination of the two) is introduced that drives the cortical activity above baseline. Cortical projections to the striatal populations initiate ramping dynamics there, which in turn impacts activity downstream in the rest of the BG and Th. We also assume that when the mean firing rate of a thalamic population exceeds a

designated threshold value of 30 spikes per second (the so-called decision boundary), the CBGT network, and hence the agent, has made a choice. This event designates the end of *phase 0*, and the duration of this phase is what we call the *reaction time*. If a decision is not made within a time window $\Delta_{max}\, ms$ after the start of the phase, then we say that none of the available choices have been selected and the decision is recorded as "none". Such trials can be excluded from further analysis depending on the hypothesis being investigated.

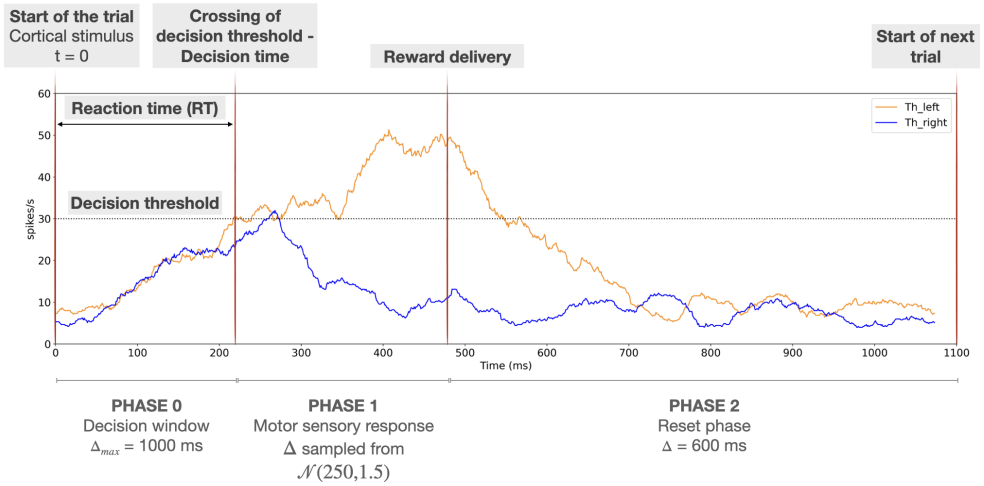


Fig 2.4. Representation of the different phases of the simulated decision process. This sketch represents the simulation of one trial of the 2-choice task in which a left choice is made. The first red vertical line, at time 0, represents the time of onset of a ramping stimulus to Cx_i for both i, which indicates the start of the trial. The second red vertical thick line depicts the decision time (end of *phase 0*). The third red vertical line depicts the end of the motor response period associated with the decision (end of *phase 1*), which is also the time when reward delivery occurs and hence dopamime level is updated. After this time, the reset phase (*phase 2*) starts; this ends after $600\, ms$ (inter trial interval), when a new trial starts (right-hand vertical red line). Orange and blue traces represent the mean thalamic firing rates Th_i for $i \in \{\text{left, right}\}$, respectively, and the horizontal black dotted line highlights the decision threshold.

To allow for selection between n different choices we instantiate n copies of all CBGT populations except FSI and CxI. This replication sets up action channels representing the available choices that can influence each other indirectly through the shared populations and otherwise remain separate over the whole CBGT loop. To distinguish between the firing ratse of the populations within channels, we will call them Pop_i, where Pop refers to the corresponding CBGT region and i refers to the channel name (e.g., Cx_{left}, Cx_{right} for $n = 2$).

The presentation of a stimulus to the cortical population is simulated by increasing the external input frequency in all copies of the cortical Cx populations that ramp to a target firing rate I_{target}. The ramping current $I_{ramp}(t)$ is calculated as

$$I_{ramp}(t) = I_{ramp}(t - dt) + 0.1\big[I_{target}(t) - I_{ramp}(t - dt)\big]$$

where dt is the integrator time step, and the external input frequency also changes according to

$$f_{ext,x}(t) = f_{ext,x,baseline}(t - dt) + I_{ramp}(t).$$

After a Th population reaches the threshold and hence a decision is made, the ramping input to Cx is extinguished and a subsequent period that we call *phase 1* or *consolidation* phase begins, which by default has duration sampled from a normal distribution $\mathcal{N}(\mu = 250ms, \sigma = 1.5ms)$ but can also be fixed to a given duration. This phase represents the period of the motor implementation of the decision. Activity during *phase 1* strongly depends on what happened in *phase 0* such that, if a decision i occurred at the end of *phase 0*, then Cx_i will be induced to exhibit sustained activity during *phase 1* [57] (see S2), while in any non-selected action channels, the cortical activity returns to baseline. If no decision has been made by the network (within a time window $\Delta_{max}\, ms$, with a default value of 1000 ms), then no sustained activity is introduced in the cortex (see Figure 2.3, 3rd trial, the top left subplot showing cortical activity).

Finally, each trial ends with a reset phase of duration $600\, ms$ (although this can be adjusted by the user), which we call *phase 2* or the *inter-trial interval* phase, when the external input is removed and the network model is allowed to return to its baseline activity, akin to an inter-trial interval.

A visualization of the decision phases is shown in Figure 2.4, where two different options, *right* and *left*, are considered. The blue trace represents the thalamic activity for the *right* channel, Th_{right}, while the orange trace represents that for the *left* channel, Th_{left}. At the end of *phase 0*, we can see that Th_{left} reaches the decision threshold of 30 spikes per second before Th_{right} has done so, resulting in a left choice being made. During *phase 1*, the Th_{left} activity is maintained around 30 spikes per second by the sustained activity in Cx_{left}.

In this task, critical attention should be paid to *phase 0*, as this represents the process of evidence accumulation where the cortical input and striatal activity of both channels ramp until one of the thalamic populations' firing rates reaches the threshold of $30\, Hz$. To be largely consistent with commonly used experimental paradigms, the maximal duration of this phase is considered to be $\Delta_{max} = 1000\, ms$ such that, if the agent makes no decision within 1000 ms, the trial times out and the decision is marked as "none". These trials can be conveniently removed from the recorded data before analysis. If a decision is made, then the simulation proceeds as though a reward is delivered at the end of *phase 1* – that is, at the end of the motor sensory response – such that *phase 1* represents the plasticity phase, where the choice selected in *phase 0* is reinforced with a dopaminergic signal. During this phase, the cortical population of the selected channel receives 70% of the maximum cortical stimulus applied during the ramping phase, although the user can change this percentage. This induces sustained activity that promotes dopamine- and activity-dependent plasticity as described in S2. The activity-dependent plasticity rule strengthens (weakens) the corticostriatal weight to dSPNs (iSPNs) of the selected channel when dopamine rises above its baseline level. CBGTPy allows for the specification of other parameters such as learning rate, maximum weight values for the corticostriatal projections and dopamine-related parameters (see details with examples in Section 2.2.3).

At the beginning of the simulation, with baseline network parameters, the selection probabilities are at chance level (i.e., 50% for a 2-choice task). If the network experiences rewards, however, the dopamine-dependent plasticity strengthens the corticostriatal projection to the dSPN population of each rewarded choice, thereby increasing the likelihood that it will be selected in the future. CBGTPy allows for probabilistic reward delivery associated with each option, as well as switching of these probabilities between the two actions (Figure 2.5). When such a change point occurs, the previously learned action now elicits a negative reward prediction error, forcing the network to unlearn the previously learned choice and learn the new reward contingency.

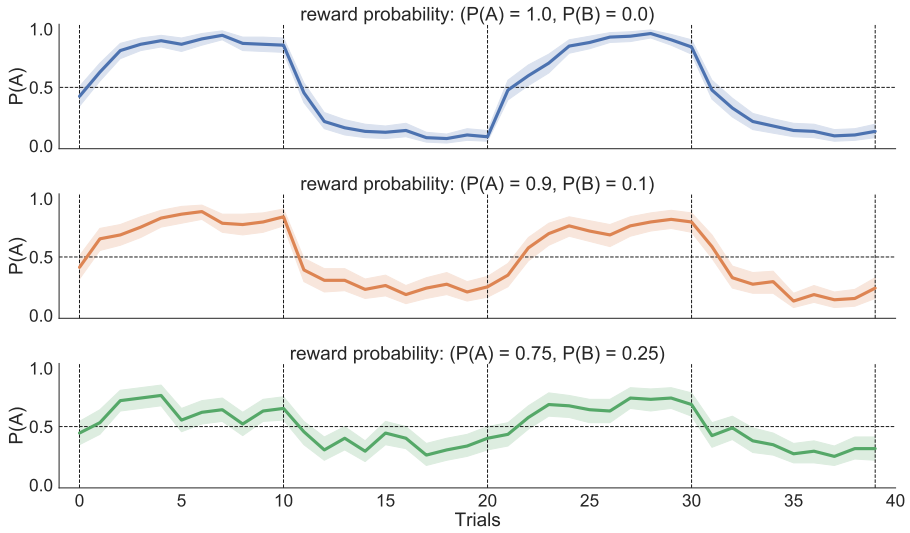


Fig 2.5. Probability of choosing the more rewarded action (e.g., A or B) for different levels of conflict in a 2-choice task. The reward contingencies flip between the two choices every 10 trials (marked by vertical dashed lines), at which point the probability of choosing the more rewarded option drops below chance. The probability of choosing A is high in the 1st and 3rd blocks; however, the probability to choose A drops in the 2nd and 4th blocks (where B is rewarded with a higher probability). Performance, measured in terms of probability of selecting option A, degrades in general as conflict increases, but sensitivity to change points drops. The performance was averaged over 50 random seeds for each conflict level.

2.3.2 A stop signal task

The stop signal task represents a common paradigm used in cognitive psychology and cognitive neuroscience for the study of reactive inhibition [59]. In this task, participants are trained to respond as fast as possible after the presentation of a "Go" cue. Sometimes the "Go" cue is followed by the presentation of a "Stop" cue, which instructs subjects to withhold their decision and hence, if successful, prevents any corresponding movement before it begins.

Imaging and electrophysiological studies in humans, rodents, and monkeys agree in reporting that STN neurons become activated in response to a stop signal [60], providing a fast, non-selective pause mechanism that contributes to action suppression through the activation of the cortical hyperdirect pathway [61, 62]. However, this mechanism, by itself, mostly fails to inhibit locomotion, appearing to be not selective and long-lasting enough to prevent a late resurgence of the evidence accumulation process as needed to guarantee a complete cancellation of the execution of the motor response [63]. A complementary slower but selective mechanism is thought to be provided by the activation of arkypallidal neurons in the GPe in response to an external stimulus that instructs the network to brake the ongoing motor planning process [62–64]. According to this idea, a long-lasting action inhibition results from the activation of pallidostriatal GABAergic projections. For more details on how this mechanism takes place see [42].

To reproduce these mechanisms and to simulate the interruption of the action selection process, we inject two independent, external, excitatory currents directly into STN and GPeA neurons during a typical CBGT simulation. This choice is based on the findings of Mallet et. al. (2016) [63]. The stop signal is excitatory and hence is

simulated by up regulating the baseline input frequency to the AMPA receptors

$$f_{ext,x}(t) = f_{ext,x}(t) + \texttt{stop\_amplitude},$$

where `stop_amplitude` defines the magnitude of the stop signal stimulation. The currents injected as a step function cause an increase in the firing rates of the target nuclei. Both of these external currents are defined using parameters that can be modified in an easy and user-friendly way, without requiring any familiarity or advanced knowledge of the details of the implementation (see Section 2.2.3 for further details and examples).

The following list of parameters characterizes each one of these currents:

(a) `amplitude`, specifying the magnitude of the stimulation applied;

(b) `population`, specifying the CBGT region or sub-population being stimulated.

(c) `onset`, specifying the onset time of the stimulation, with respect to the trial onset time (i.e., the beginning of *phase 0*);

(d) `duration`, defining the duration of the stimulation. Since the length of *phase 0* is not fixed and is dependent on how long it takes for the thalamic firing rates to reach the decision threshold (30Hz), this parameter should be set carefully;

(e) `probability`, determining the fraction of trials on which the stop signal should be introduced;

(f) `channel`, defining which action channels are stimulated.

A more detailed description of all of these parameters can be found in Section 2.2.3. The details of the stop parameters used to reproduce Figure 2.7 are included in S16 Table.

The characterization of the different network phases described in Section 2.2 slightly changes when performing the stop signal task (see Figure 2.6). During the *decision* phase (*phase 0*, which in this task lasts for a maximum of 300 ms) the stop signals are directly presented to the target populations by injecting independent external currents. The user can choose the moment of the injection by manipulating the variable `stop_signal_onset_time`. These signals are kept active for a period equal to `stop_signal_duration`, with a magnitude equal to the `stop_signal_amplitude`. These values do not need to be the same for all of the stop signals used. At this stage, two possible outcomes follow: (a) despite the presentation of the stop signals, the network still manages to choose an action; or, (b) the network is not able to make an action after the presentation of the stop signals, and *phase 0* ends with no action triggered, which represents a `stop` outcome. The former option could arise for various reasons; the strength of the stop signal may not be sufficient to prevent the network from triggering an action or the evaluation process may still have enough time to recover, after the stop signal ends, to allow the thalamic firing rates to reach the decision threshold (e.g., $30\,Hz$) within the permitted decision window.

In Figure 2.7 we show an example of stop signal stimulation applied to STN and GPeA populations, independently, in a 1-choice task. The onset of the stimulation applied to STN occurred at $30\,ms$ while that for the stimulation of GPeA was set to $60\,ms$; both signals were applied for a duration of $145\,ms$. Both stimulations were applied in both of the trials shown. Note that trial number 1 corresponds to a correct stop trial (no decision was made within *phase 0*), whereas the following trial corresponds to a failed stop trial. These outcomes can be inferred from the activity of various populations at the end of the decision making windows: the thalamic firing rate reaches a higher level there and the firing traces in GPi decrease more for the failed stop trial than for correct stop, while cortical activity is sustained beyond this time specifically when stop fails.

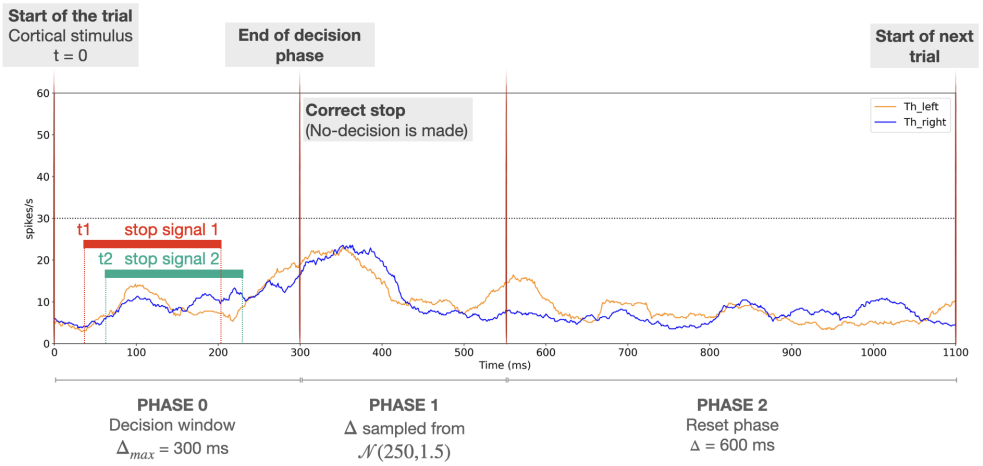


Fig 2.6. Representation of the phases of the stop signal task with two action channels. This sketch represents one trial of the simulation. The first red vertical line indicates the presentation of the cortical stimulus (causing ramping of cortical activity), which represents the start of the trial. Red and green horizontal bars depict the presentation of two stop currents, according to the onset time and duration values chosen by the user. In this example, the two stop signals are considered to be applied with different onset times t_1 and t_2, respectively. The second red vertical line depicts either the moment when an action has occurred (end of *phase 0*) or that $300\,ms$ has expired and no action has been triggered, so a successful stop has occurred. The third red vertical line depicts the end of the motor sensory response phase (end of *phase 1*), if an action is triggered (failed stop). Here, the stop was successful (no decision threshold crossing within the decision window), so no motor sensory response is visible. Finally, the *reset phase* (*phase 2*) occurs, after which a new trial begins. Blue and orange traces represent the mean thalamic firing rates Th_x for $x \in \{\text{right,left}\}$, respectively, and the horizontal black dotted line marks the decision threshold.

2.3.3 Optogenetic stimulation

CBGTPy also allows for simulation of optogenetic stimulation of CBGT nuclei while an agent performs the available tasks (stop signal or n-choice). Optogenetic stimulation is implemented by setting a conductance value for one of two opsins dependant upon the mode of stimulation, channelrhodopsin-2 for excitation and halorhodopsin for inhibition. The excitatory or inhibitory optogenetic input is applied as a current I_{opto} added to the inward current I_{ext} of all neurons in a nucleus or subpopulation during a typical CBGT simulation such that

$$I_{opto}(t) = \begin{cases} g_{opto}(V(t) - V_{ChR2}) & g_{opto} \geq 0 \\ -g_{opto}(V(t) - V_{NpHR}) & g_{opto} < 0 \end{cases}$$

where the conductance g_{opto} is a signed value entered via the configuration variable in the notebooks. The reversal potential of channelrhodopsin (V_{ChR2}) was considered to be $0\,mV$ and halorhodopsin (V_{NpHR}) was considered to be $-400\,mV$ [65].

The stimulation paradigm includes the following parameters:

(a) `amplitude`, the sign of which specifies the nature (positive→excitatory / negative→inhibitory) and the absolute value of which specifies the magnitude of the conductance applied;

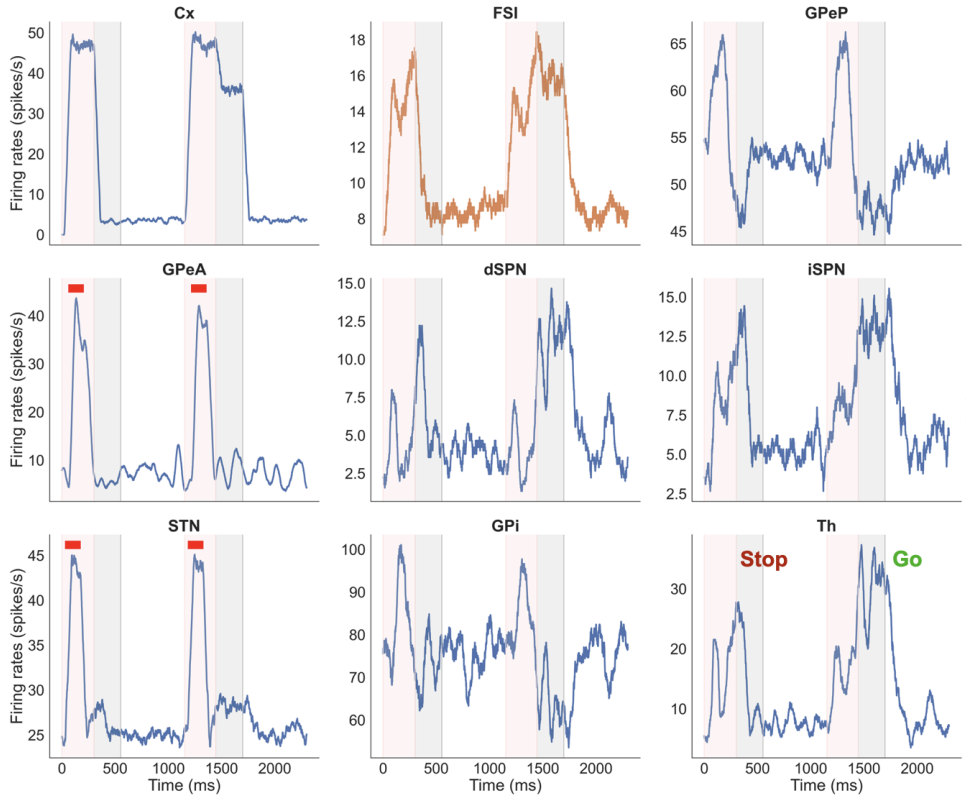


Fig 2.7. Example figure showing firing rates for all nuclei for two consecutive stop trials. Note that the simulation has been run in a 1-channel regime and two stop currents have been applied to STN and GPeA, respectively (see thick red bars). Segments of the simulation are color-coded to distinguish times associated with decision making (pink, phase 0) and subsequent times of motor response (grey, phase 1, showing sustained activity in the selected channel when a decision is made) in each trial. The unshaded regions after the trials are the inter-trial-intervals (phase 2).

(b) `population`, specifying the CBGT region or sub-population being stimulated;

(c) `onset`, specifying the onset time of the stimulation;

(d) `duration`, defining the duration of the stimulation;

(e) `probability`, indicating the fraction of trials or a list of trial numbers to include stimulation;

(f) `channel`, specifying which action channels are stimulated.

The parameter `population` should be entered as a list of the subpopulations to be stimulated. The parameter `onset` is calculated from the beginning of *phase 0*; for example, if this parameter is 10, then the optogenetic stimulation starts 10 ms after *phase 0* starts. The parameter `duration` controls the duration of the optogenetic stimulation. This parameter either accepts a numeric value in ms or a string specifying which *phase* should be stimulated. The numeric value stipulates that the list of selected populations will be stimulated from the specified onset time for the specified time duration. The string (e.g., "phase 0") stipulates that the stimulation should be applied throughout the specified phase, thereby allowing the user to specifically target the

decision, *consolidation* or *inter-trial interval* phase. If an optogenetic configuration results in extending the duration of a phase (e.g., strongly inhibiting dSPN may extend *phase 0*), a time out is specified for every phase to prevent a failure to terminate the phase. The default timeouts for *phase 0*, *1* and *2* are $1000\,ms$, $300\,ms$ and $600\,ms$ respectively unless specified by the user.

The parameter `probability` offers the flexibility of either assigning a number that determines the fraction of trials (randomly sampled from the full collection of trials) on which the stimulation is to be delivered or else entering a list of specific trial numbers. Lastly, the parameter `channel` specifies the name of the action channel, such as "left", onto which the stimulation should be applied. This parameter also accepts two additional options, "all" or "any", the former of which leads to the application of a global stimulation to the same population in all channels and the latter of which randomly selects a channel for stimulation on each trial. The details of the optogenetic input is included in S17 Table.

We show an example of optogenetic stimulation applied to a list of iSPN and dSPN populations in a 3-choice task (Figure 2.8). The excitatory stimulation (shown as thick blue bar), with an amplitude of 0.1, was applied to the iSPN populations of all the channels (namely A, B, and C) in the first trial, for the duration of *phase 0*. An increased activity in the iSPN population (activation of the indirect pathway) caused a choice time out on this trial. In the subsequent second trial, an inhibitory stimulation with an amplitude of -0.5 was applied to the dSPNs (shown as yellow bar) for $400\,ms$. This resulted in brief but strong inhibition of dSPNs (direct pathway), thereby delaying the action selection. This can be observed by comparing the durations of the *decision phase between the second and third trials*, where no such manipulation was imposed.

2.4 Discussion

Here we introduce CBGTPy, an extensible generative modeling package that simulates responses in a variety of experimental testing environments with agent behavior driven by dynamics of the CBGT pathways of the mammalian brain. A primary strength of this package is the separation of the agent and environment components, such that modifications in the environmental paradigm can be made independent of the modifications in the CBGT network. This allows the user to derive predictions about network function and behavior in a variety of experimental contexts, which can be vetted against empirical observations. Moreover, various changes in the parameters of the network, as well as the experimental paradigm, can be made through the higher-level `configuration` variable that is sent as an argument in running the simulation, thereby avoiding a considerable coding effort on the part of the user. CBGTPy also returns behavioral outcomes (e.g., choices made and decision times) and "recordings" of neuronal outputs (instantaneous firing rates) for all of the CBGT nuclei in the form of easily usable and readable data frames. Overall, CBGTPy allows for theorists and experimentalists alike to develop and test novel theories of the biological function of these critical pathways.

The individual components of CBGTPy are all designed to enable maximum flexibility. The basal ganglia model is constructed in an organized series of steps, beginning with high-level descriptions of the model and gradually providing more fine-grained details. Developing a modification to the network becomes a matter of inserting or modifying the appropriate components or steps, allowing high-level redesigns to be implemented as easily as more precise low-level modifications. CBGTPy's high degree of extensibility can, in large part, be attributed to its use of a data-flow programming paradigm. Neural pathways between major populations, for example, can be specified at a very high level, requiring only a single entry in the

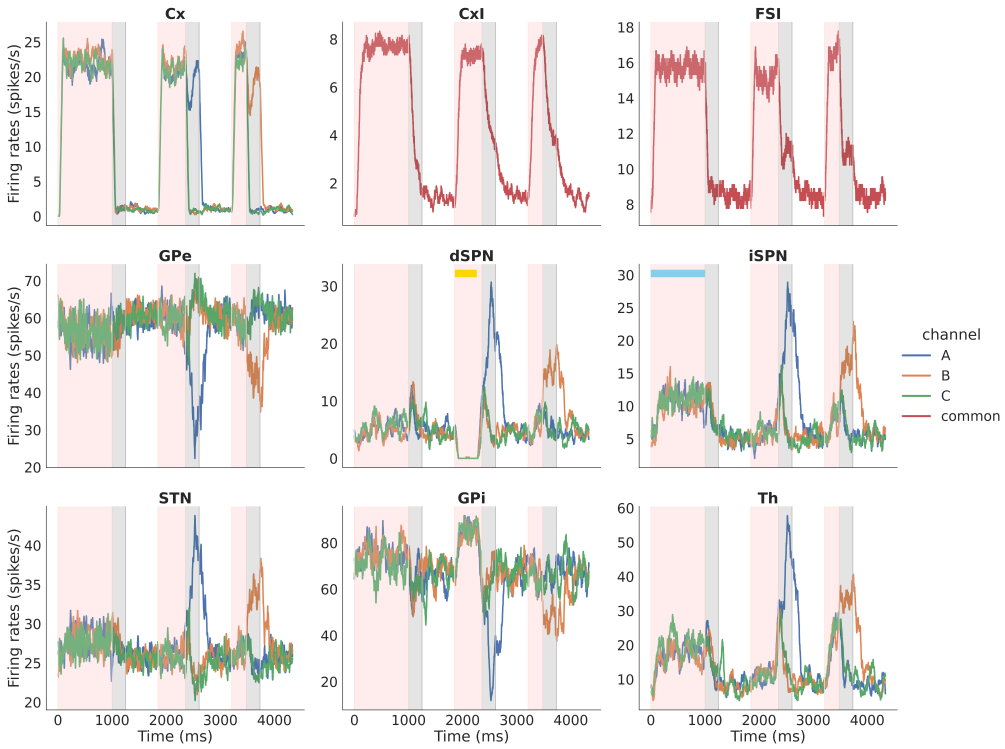


Fig 2.8. Example figure showing optogenetic stimulation for the nuclei 'iSPN' and 'dSPN'. The configuration specified was: `amplitude: [0.5, -0.5], duration: ['phase 0', 400], trial numbers: [[0], [1]], channels: ['all', 'all']`. The excitatory optogenetic stimulation given to iSPN (shown as blue bar) and lasts all through *phase 0*, whereas inhibitory stimulation to dSPN (shown as yellow bar) lasted for 400 ms. In both cases, stimulation were applied to all channels (namely A, B and C) of the nuclei.

pathway table to describe how each subpopulation is connected. If the connectivity of a particular subpopulation, or even a particular neuron, needs adjustment, then the later steps in network construction can be adjusted to implement those changes. CBGTPy was designed with this degree of flexibility to ensure that in the future, more complex biological models of the CBGT network can be developed and implemented in an efficient manner.

Of course, CBGTPy is not the only neural network model of these cortical-subcortical networks. Many other models exist that describe the circuit-level dynamics of CBGT pathways as either a spiking [53, 66–71] or a rate-based [38, 39, 72–78] system. CBGTPy has some limitations worth noting, such as not being as computationally efficient as rate-based models in generating macroscale dynamics, including those observed using fMRI or EEG, and associated predictions. Also, the properties of the cortical systems modeled in CBGTPy are quite simple and do not capture the nuanced connectivity and representational structure of real cortical systems. For these sorts of questions, there are many other modeling packages that would be better suited for generating hypotheses (e.g., [79]). Where CBGTPy excels is in its a) biologically realistic description of subcortical pathways, b) scalability of adding in new pathways or network properties as they are discovered, c) flexibility at emulating a variety of typical neuroscientific testing environments, and d) ease of use for individuals with relatively limited programming experience. These benefits should make

CBGTPy an ideal tool for many researchers interested in basal ganglia and thalamic pathways during behavior.

One issue that has been left unresolved in our toolbox is the problem of parameter fitting [80, 81]. Spiking network models like those used in CBGTPy have an immense number of free parameters. The nature of both the scale and variety of parameters in spiking neural networks makes the fitting problem substantially more complex than that faced by more abstracted neural network models, such as those used in deep learning and modern artificial intelligence [82, 83]. This is particularly true when the goal is to constrain both the neural and behavioral properties of the network. Models like CBGTPy can be tuned to prioritize matching cellular level properties observed empirically (for example see [44]) or to emphasize matching task performances to humans or non-human participants (see [45]). We view this as a weighted cost function between network dynamics and behavioral performance whose balance depends largely on the goals of the study. To the best of our knowledge, there is no established solution to simultaneously fitting both constraints together in these sorts of networks. Therefore, CBGTPy is designed to be flexible to a wide variety of tuning approaches depending on the goal of the user, rather than constrain to a single fitting method.

Because our focus is on matching neural and behavioral constraints based on experimental observations, CBGTPy's environment was designed to emulate the sorts of task paradigms used in systems and cognitive neuroscience research. We purposefully constructed the environment interface to accommodate a wide variety of traditional and current experimental behavioral tasks. These tasks are often simpler in design than the more complex and naturalistic paradigms used in artificial intelligence and, to an increasing degree, cognitive science. Nonetheless, a long-term goal of CBGTPy development is to interface with environments like OpenAI's Gym [84] in order to provide not only a mechanistic link towards more naturalistic behavior, but also a framework to test hypotheses about the underlying mechanisms of more dynamic and naturalistic behaviors.

In summary, CBGTPy offers a simple way to start generating predictions about CBGT pathways in hypothesis-driven research. This tool enables researchers to run virtual experiments in parallel with *in vivo* experiments in both humans and non-human animals. The extensible nature of the tool makes it easy to introduce updates or expansions in complexity as new observations come to light, positioning it as a potentially important and highly useful tool for understanding these pathways.

Chapter 3

Modeling CBGT-hippocampal cooperation via successor representation

3.1 Introduction

Consider the process of dining out at a restaurant. To have a rewarding experience and to avoid negative outcomes, one must perform a sequence of actions in this environment, such as getting seated, ordering food, ordering dessert, and paying the bill. There are many possible courses of action, each of which may incur a different degree of reward, and the general problem of learning the best behavioral pattern that maximizes future rewards is known as reinforcement learning (RL) [85]. Within the mammalian brain, the process of action selection [13], particularly in RL tasks, is believed to be heavily driven by the cortico-basal ganglia-thalamic (CBGT) pathways [2] [14]. In response to dopaminergic input from the substantia nigra pars compacta (SNc), the CBGT circuit adjusts its internal competitive dynamics to favor previously rewarded actions (e.g., [1]), thus fulfilling the core computational goal of RL. Simple action policies, however, in which rewarding actions are broadly favored, are far from sufficient to solve real-world tasks such as dining out. First, the value of an action is heavily contingent upon the context in which it was performed, so the agent's action selection process must consider the current environmental state. In other words, the importance of paying a bill depends heavily on whether food has previously been ordered. Second, for the learning process to be efficient, the agent must identify and leverage the patterns, or structure, present in the environment. For example, one might select food based on the expected bill without needing to relearn the casual relationship between food and bills for each new restaurant visited. Both of these points require the agent to learn the structure of the environmental state space and use state information throughout the RL process. Thus, it is natural to ask how the dynamics of the basal ganglia network may integrate structural information to guide the formation of complex action policies.

Structure learning, the process of inferring the latent structure of the environmental state space from experience [10], is quite a distinct problem from RL. It is unclear how the basal ganglia alone would be equipped to perform all the computations necessary to both determine the shape of state space of a task and perform RL over that space. There is, however, substantial evidence that structured RL tasks engage multiple learning systems within the brain, and that it is the coordination of these systems that enables the production of complex volitional behavior [18]. Notably, activity within the

hippocampal-entorhinal cortical (HC-EC) system, famous for its role in spatial navigation [27], appears to encode the latent structure of structured RL tasks [5]. Conceptually, behavior in structured RL tasks can be viewed as navigation over a mental map [7] in which each environmental state is a location on the map. This conceptual parallel suggests that the neural representations present in the hippocampus could encode the distinct metaphorical places over which the RL action policy should be learned.

There is a wealth of evidence supporting the existence of a role for the HC-EC system in RL tasks. Neural recordings in primates during a dynamic bandit task indicate that, as the values of the bandits change over time, some hippocampal cells respond to specific locations on the latent manifold of bandit values [7]. These cells have been deemed "place value cells" and their presence indicates that the hippocampus could encode the state of RL tasks in a manner similar to the spatial states of navigation tasks [7]. The hippocampus also appears to aid RL by forming a conjunctive representation over which value learning can occur [4]. These conjunctive representations allow values to be assigned to combinations of features, such that the value of a feature depends on the context in which it appears. Human neuroimaging data indicates that this representation, provided by the hippocampus, influences reward prediction errors in the basal ganglia [4]. Furthermore, other human imaging studies indicate that, when presented with a set of bandit tasks with varying correlative structures, the structures of the tasks are encoded in the activity of the entorhinal cortex in a manner that generalizes across tasks of similar structure [5]. Taken together, it is evident that the HC-EC system is engaged during RL problems and that the system may encode the structure of the RL task state space. The hippocampus may influence the RL process in the basal ganglia both during the process of action selection and during reward-induced dopaminergic plasticity. As the cortex is the primary driver of striatal activity, it is likely that hippocampal projections to the cortex are key to this interaction. CA1 neurons, which include hippocampal place cells, project to prefrontal cortex [27], and electrical recordings in primates have established that primary sensorimotor and premotor cortical neurons can exhibit spatially-selective activity [29]. These cortical areas, in turn, project to the basal ganglia [28], completing the avenue by which place cells could influence action selection and decision-making.

Perhaps the key to understanding the computational role of hippocampal representations in RL lies in the apparent predictive nature of the representations and how those representations appear to be influenced by environmental transitions [6]. It has been hypothesized that the hippocampus embodies a particular form of predictive coding known as successor representation (SR), in which each state is encoded in terms of the expected future occupancy of other states [6] [23] [8]. SR is particularly useful for RL, as the one of the direct outputs of SR is a transformation matrix that describes the extent to which each potential future state transition contributes to the value of the present state [26]. Notably, in SR, each individual feature of the representation is tuned both towards a particular state and to the states which often precede it [6]. This matches observations that place cells in rodents appear to be influenced by an animal's movement patterns, with individual place cells expanding their receptive fields backward relative to the direction of the animals [23]. Furthermore, an imaging study in humans has shown that hippocampal activity appears to be influenced by transition probabilities in non-spatial tasks [23]. The similarity between place cell fields and SR is also supported by modeling results. A model of place cells by de Cothi and Barry has shown how, given a set of boundary vector cells to serve as the state basis vector, SR gives rise to simulated place cells that are quite similar to their biological equivalents [8]. SR is thus a unifying idea that could explain how the hippocampal network performs the same fundamental computations in both spatial and non-spatial contexts.

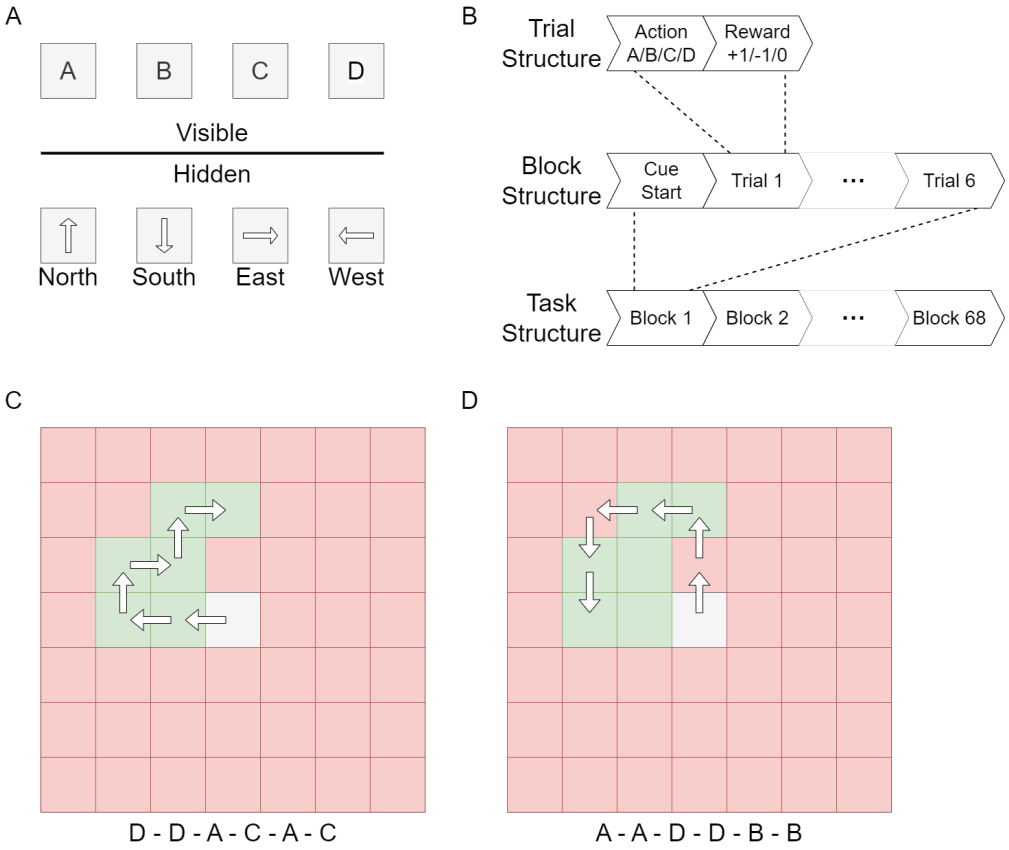


Fig 3.1. A four-armed bandit with latent structure (A) The agent sees only four buttons representing the four arms of the bandit, with no spatial information provided. Each button, however, secretly corresponds to movement in a cardinal direction. (B) The task contains 68 blocks, each block consisting of a start cue followed by six trials. Each trial consists of an action along with the corresponding positive, negative, or neutral reward feedback. (C) The sequence of actions within a block determine the agent's trajectory on the latent grid, with the optimal sequence of actions corresponding to following an optimal path on the grid. (D) Deviations from the optimal path result in less net reward. The value of the squares do not depend on the order in which they are visited, though each square only provides a non-zero reward once per block when first visited.

Here, we explore how place-cell-like representations, similar to those present in the HC-EC system, could aid the CBGT circuit in forming a complex action policy within a structured RL task. We consider a four-armed bandit task, illustrated in Figure 3.1, with a grid-like latent structure that closely resembles the type of pseudo-spatial structure the HC-EC system is expected to be best equipped to learn. First, we show how a model of place cells, derived from SR, gives rise to a neural representation of environmental state in which the place cells are tuned to the distinct latent states of the RL task. The SR-derived representation is contrasted with two handcrafted representations which encode distinct aspects of the task structure. Secondly, we incorporate these place cell representations into a model of the basal ganglia, using the CBGTPy python toolbox [24], to investigate how the place-cell-like representations could complement the structure and dynamics of the CBGT network and produce a simulated behavioral agent capable of learning a complex multi-step optimal policy.

Furthermore, we investigate how properties of the neural representations, such as sparsity, and structural properties of corticostriatal connectivity influence the network's performance. Ultimately, this work aims to highlight how structure learning, as performed via hippocampal circuits, can potentially coordinate with the basal ganglia to regulate RL in complex environments.

3.2 Methods

3.2.1 Defining a Structured RL Task

Within the scope of this chapter, RL tasks are composed of a discrete sequence of interactions between an environment and a behavioral agent. The environment possesses a hidden state s_t, and each state transition is accompanied by a visible sensory observation x_t. Each sensory observation is a symbol selected from a predefined symbol set, which includes any cue presentations, movements, or reward presentations that might occur throughout the task. As shown in Figure 3.2, certain state transitions are influenced solely by the environmental state, while others involve the selection of a transition by the agent from a list of available options. The environment is defined as being memoryless, meaning that an agent with knowledge of the true present state s_t does not gain any additional predictive power by knowing the values of past states or observations.

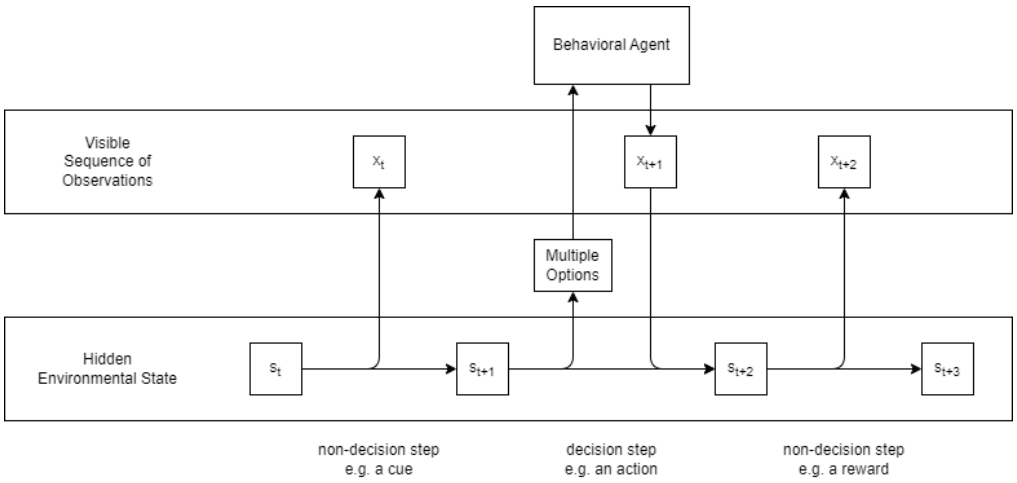


Fig 3.2. Relationship between the environmental state s and sensory observation x. Sensory observations are generated by environmental transitions. Some transitions, termed decision steps, are selected by the agent in response to options presented by the environment, while other transitions are influenced only by the internal dynamics of the environment. The behavioral agent only has access to the sequence of observations and not the true underlying state.

Each possible observation symbol is assigned a reward quantity $R(x)$, and the goal of an RL agent is to maximize the expected time-discounted sum of future rewards. In other words, the agent aims to influence the environmental state through its actions to achieve the states with the highest possible value

$$V(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(x_t) | s_0 = s\right] \tag{3.1}$$

where γ is a time-discounting factor that determines the trade-off between the quantity of a reward and the delay of the reward. The agent, however, only has access to the symbolic sequence of observations and thus must form its own internal model of the environment and an estimate $\hat{s}_t$ of the true state. The agent's estimate of state value, given imperfect information about the environmental structure and transitions, is denoted $\hat{V}$.

To serve as a prototypical example of a structured RL task, this work adapts the latent grid bandit task introduced by Bond [9]. In this four-arm bandit task, the behavioral agent is asked to complete a sequence of six decisions, each of which may result in a reward, penalty, or neither. Unknown to the agent, the rewards presented in the task are dictated by the agent's position on a latent two-dimensional grid, as shown in Figure 3.1, with each of the four options corresponding to a movement in one of the four cardinal directions. Thus, the value of a particular bandit arm is contingent on the sequence of arms previously selected, but the structure of this contingency is relatively simple once the structure of the latent space is understood. Through extensive repetition of the 6-choice blocks, the agent should eventually discover the optimal six-step path that maximizes reward. The pseudo-spatial structure of the latent grid is precisely the sort of structure that hippocampal circuits would excel at learning. So, it serves as an ideal test-bed for how hippocampal representation might enable the basal ganglia to select the optimal action along each point on the imaginary path.

To adapt the task from Bond [9], we must precisely describe the sensory observations that occur throughout the performance of the task. As outlined in panel B of Figure 3.1, each of the 68 blocks of the task contain a start cue followed by six trials, and each trial consists of an action followed by a reward signal. The events of the task can therefore be represented by symbolic sequences constructed from the following alphabet:

$$\{START, A, B, C, D, +1, -1, 0\} \tag{3.2}$$

Here, $START$ is the block start cue, while A, B, C, and D indicate the selection of each of the four bandit arms. The three possible reward values, positive, negative, and neutral, are represented by $+1$, -1, and 0 respectively. $R(x)$ is set to zero for all symbols except $+1$ and -1. Each trial is two symbols long, and each block is $1 + 2 \times 6 = 13$ symbols long. A complete run of the entire task is $13 \times 68 = 884$ symbols long and contains $6 \times 68 = 408$ separate decisions. The state of the task is determined by the current row and columnar position of the agent within the latent grid, along with the list of cells that have already been visited during the current block. As each block of 6 trials is independent, the environment resets to the same initial state s_0 after the end of each block.

To successfully perform this task, the agent must accomplish two objectives each time it is presented with the opportunity to make a decision. First, it must process the previously-observed symbols $x_0, x_1, \cdots, x_{t-1}$ to form an internal estimate $\hat{s}_t$ of the environmental state, and then it must select x_t from $\{A, B, C, D\}$ to maximize $\mathbb{E}\left[\sum_{t'=t}^{\infty} \gamma^{t-t'} R(x_{t'}) | \hat{s}_t\right]$. To accomplish these goals, the agent must perform both structure learning, to understand the relationships between past and future observations, and RL, in which the state value estimates and observed rewards guide the formation of an action policy.

3.2.2 Successor Representation

Within the SR paradigm, the agent forms an estimate of the time-discounted future occupancy matrix M, whose entries are defined as

$$M(s, s') = \sum_{t=0}^{\infty} \gamma^t \mathrm{P}\left(s_t = s' | s_0 = s\right) \tag{3.3}$$

or equivalently

$$M(s, s') = \mathrm{E}\left[\sum_{t=0}^{\infty} \begin{cases} \gamma^t & s_t = s' \\ 0 & \text{otherwise} \end{cases} | s_0 = s\right] \tag{3.4}$$

where s is the present state, s' is a future state, and γ is the time discounting factor[†]. The key property of M is that the agent's value estimate $\hat{V}(s)$ of any particular present state s can be calculated through the simple dot-product operation

$$\hat{V}(s) = \sum_{s'} M(s, s') R(s') = \sum_{t=0}^{\infty} \sum_{s'} \gamma^t R(s') \mathrm{P}\left(s_t = s' | s_0 = s\right) \tag{3.5}$$

where $R(s')$ is the amount of immediate reward the agent expects to receive directly after visiting state s'. Here one of the major strengths of SR becomes evident, as the agent can learn M, which encodes the environmental structure, and R, which encodes the reward of each state, independently. Furthermore, changes in either M or R can be immediately reflected in the value estimate without needing to perform value iteration.

While the definition of SR above relies on the existence of discrete states, de Cothi and Barry (2020) describe a generalization to distributed representations [8]. Given any vector-valued function f of the state estimate $\hat{s}$, the successor matrix $\tilde{M}$ represents a linear mapping from $f(\hat{s})$ to the expected time-discounted sum of future values of $f(\hat{s})$, denoted $\tilde{\psi}(\hat{s})$.

$$\tilde{\psi}(\hat{s}) = \tilde{M} f(\hat{s}) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t f(\hat{s}_t) | \hat{s}_0 = \hat{s}\right] \tag{3.6}$$

The entries of the vector $f(\hat{s})$ are referred to as the basis features, while the entries of the vector $\tilde{\psi}(\hat{s})$ are referred to as the successor features. Importantly, when the basis features are a one-hot encoding of states, the resultant successor matrix $\tilde{M}$ is identical to the previously-defined M. The matrix $\tilde{M}$ can be learned through temporal difference learning via the update equation

$$\tilde{M}_{t+1} \leftarrow \tilde{M}_t + \alpha_{\tilde{M}} \left[f(\hat{s}_t) + \gamma \tilde{M}_t f(\hat{s}_{t+1}) - \tilde{M}_t f(\hat{s}_t)\right] \otimes f(\hat{s}_t)^\top \tag{3.7}$$

where $\alpha_{\tilde{M}}$ is the learning rate. This update equation leverages the important fact that

$$\mathbb{E}\left[f(\hat{s}_t) + \gamma \tilde{\psi}(\hat{s}_{t+1}) - \tilde{\psi}(\hat{s}_t)\right] = 0 \tag{3.8}$$

which provides a notion of successor prediction error. Each entry of $\tilde{M}_t$ is adjusted by a value proportional to the successor error, times the learning rate, times the partial derivative $\partial \tilde{\psi}(\hat{s}_t) / \partial \tilde{M}_t = \vec{1} \otimes f(\hat{s}_t)^\top$ of the successor features with respect to the entries of $\tilde{M}_t$.

The model of de Cothi and Barry also outlines the relationship between the successor features $\tilde{\psi}(\hat{s}_t)$ and the place cell activity factor $\vec{p}(\hat{s}_t)$. Each place cell corresponds to a single component of the successor vector, with the ith place cell firing rate calculated using

$$\vec{p}(\hat{s}_t)_i \propto \mathrm{ReLU}\left[-T_i + \tilde{\psi}(\hat{s}_t)_i\right] \tag{3.9}$$

where T_i is a threshold, set to 80% of the maximum value of $\tilde{\psi}(\hat{s}_t)_i$ across all environmental states, and ReLU is the rectified linear activation function [8]. Finally, the firing rates of the place cells are re-scaled so that each cell achieves the same maximum activation.

[†]It is important to emphasize that the SR matrix $M(s, s')$ is distinct from the state transition function, as the definition of $M(s, s')$ does not require for a direct transition to exist between s and s'. Rather, the time-discounted future occupancy of state s' given the current state s depends on both how far into the future s' is expected to be visited and how often those visits are expected to occur.

3.2.3 Training the Place Cell Representation

Key to the mechanics of SR, as described above, is the existence of an encoding of state as a vector of basis features over which the SR matrix can be learned. This encoding must be learned by the agent, and the quality of this encoding influences the quality of the successor features and the accuracy of the agent's value estimates. Although it is not known how the brain might process sensory data to form an estimate of environmental state, here we take a pragmatic approach, illustrated in detail in Figure 3.3, to developing the basis vector f. First, a predictive recurrent neural network (RNN) was trained on sequences of observations, which are purely symbolic sequences constructed from the previously-described 8-symbol alphabet (Eq. 3.2). Second, assuming that the hidden state h_t of the RNN encodes the agent's internal state estimate $\hat{s}_t$, a transformation between h_t and $f(\hat{s}_t)$ was learned such that the final derived place cell vector $\vec{p}$ is highly predictive of the environment's true state. Once both components were trained, the simulated place cells obey the properties of SR, encode the task structure, and are computed given only a sequence of sensory observations as input.

One very efficient form of RNN is known as the gated recurrent unit (GRU) [31]. The GRU takes in a sequence of inputs x_t, along with a learnable initial state h_0, and outputs a sequence of hidden states h_t. Internally, the GRU first computes a reset gate r_t that enables the network to partially forget its previous state. Then, a candidate activation $\hat{h}_t$ is computed using the input symbol and the partially-reset state. Finally, the next hidden state activation h_t is calculated as a linear interpolation between the previous state h_{t-1} and the candidate activation, as controlled by an update gate z_t. The computations of a single iteration of the GRU are as follows:

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \tag{3.10}$$

$$\hat{h}_t = \phi(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \tag{3.11}$$

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \tag{3.12}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t \tag{3.13}$$

Here, W, U, and b are parameters learned via training, while σ and ϕ are the logistic and hyperbolic tangent function respectively.

While training the GRU, the input x_t is a one-hot encoding of the sequence of observed sensory events, as sampled from random behavior in the task environment. These sensory events include the block start cues, action selections, and reward signals. The output sequence h_t from the GRU is converted into a cell firing rate vector $v_t = \sigma(h_t)$, bounding the activity of each cell between 0 and 1. Then, a linear classifier with a softmax activation function is applied to v_t to predict x_{t+1}. Through gradient descent, the entire GRU-classifier network is trained to minimize this predictive loss, leading the GRU to learn an embedding of the environmental state in the firing rates v_t which is predictive of future sensory events in x_t.

Once the GRU has been trained, the model is extended with a transformation from the GRU output vector v_t to the basis cell vector f_t for each time t. For simplicity, a linear layer with logistic activation $f_t = \sigma(W_f v_t + b_f)$ is used, where W_f and b_f are the trainable weights and biases. Once the sequence of basis feature vectors is calculated, the corresponding successor matrix $\tilde{M}$ is found through temporal difference (Eq. 3.7), after which the successor features $\tilde{\psi}_t$ (Eq. 3.6) and place cell vectors $\vec{p}_t$ (Eq. 3.9) follow naturally. This place cell representation was evaluated by three distinct metrics to compute a total loss: (1) the ability to predict the next observation, (1) the ability to encode important aspects of the true environmental state, namely the agent's current location on the latent grid, and (3) the consistency of total place cell activation over

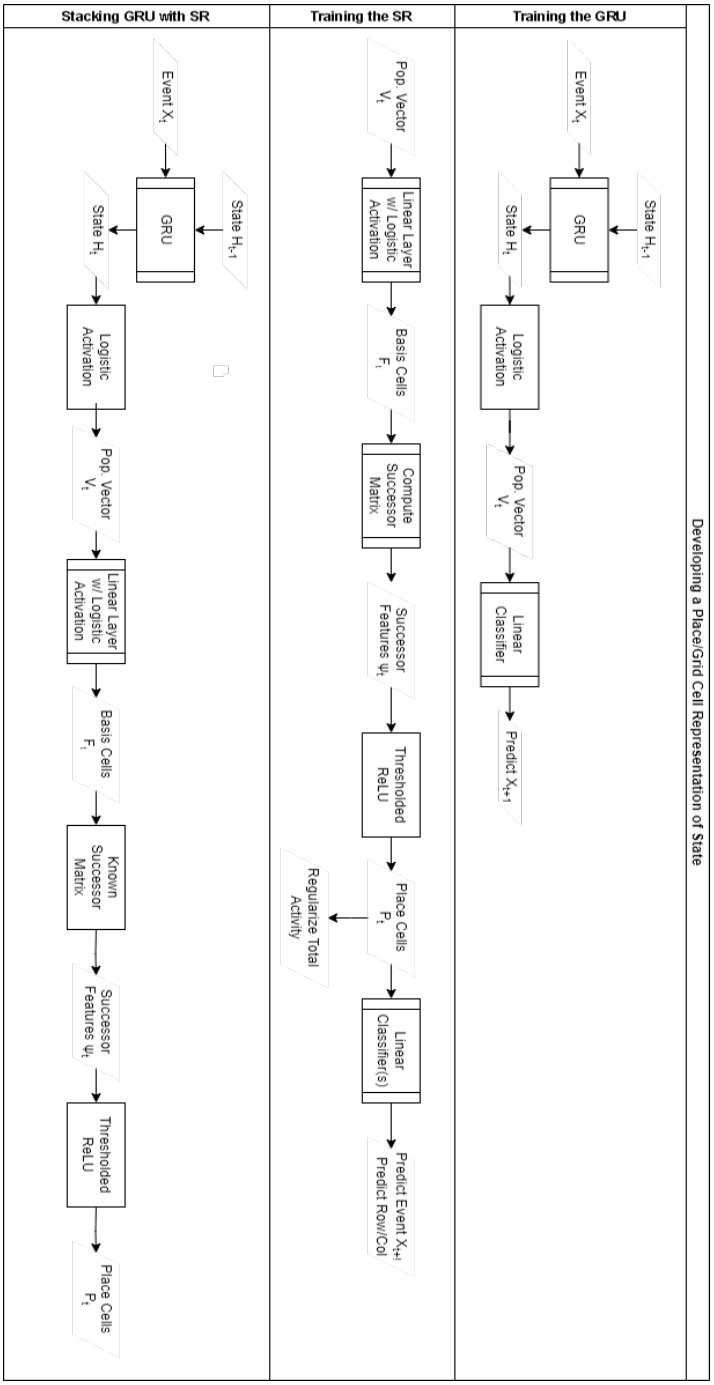


Fig 3.3. Overview of the process of generating simulated place cell activity from sequences of observed sensory events. First, a gated recurrent unit (GRU) [31] is trained to process sequences and compute a state representation which is predictive of future events. Second, basis features are extracted from this state representation, optimized such that the resultant SR representation is highly informative about the environment's true state. Finally, the previously-trained GRU, basis features, and successor matrix are used to produce place cell population vectors which change in response to sensory events.

time. Through back-propagation across the entire SR calculation, the GRU-SR model is trained to optimize the final place cell representation.

3.2.4 Extending the CBGTPy Framework

To investigate how place-cell-like neural representations could influence the dynamics and behavior of the basal ganglia, we extend the functionality of CBGTPy, a fully-spiking model of the CBGT loop, in three major ways. First, the interface between the CBGT network and the task environment is expanded to include the GRU-SR model, which processes the key events of the simulation, namely the actions generated by the network and rewards generated by the environment, to produce an up-to-date place cell activity vector. Secondly, the CBGT network is adjusted so that each cortical neuron can receive a different amount of external input, allowing the cortical activity pattern within each channel to be driven by the place cell activity vector. Finally, the calculation of reward prediction errors is replaced by a mixture-of-experts model, allowing the place cell representations to inform the agent's value estimates and therefore the magnitude of dopaminergic feedback. The interactions between the task environment, the CBGT network model, the place cell model, and the mixture-of-experts model are illustrated in Figure 3.4.

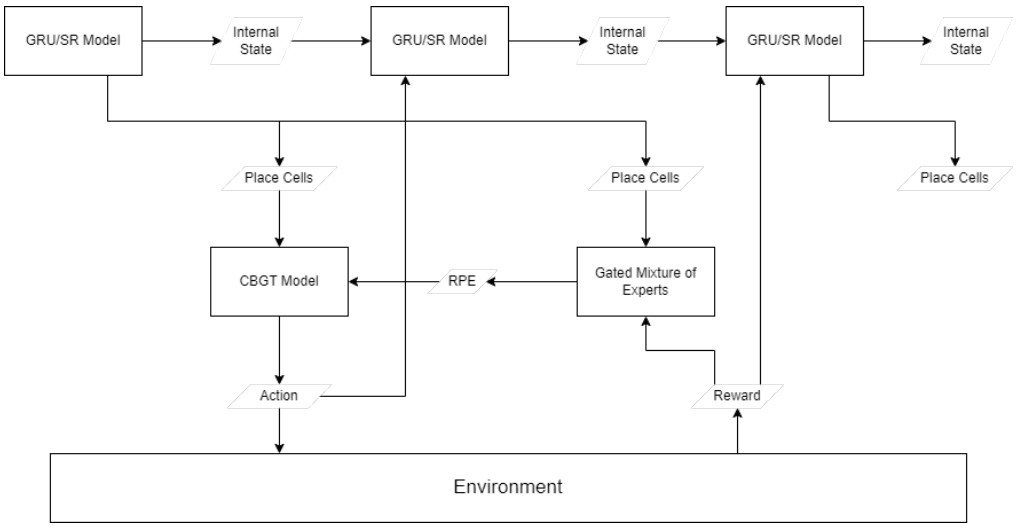


Fig 3.4. Overview of interactions between the GRU-SR model, the CBGT network, and the environment. The GRU-SR model produces place cell activations which bias the cortical input of the CBGT loop, which selects an action. This action triggers a possible reward from the environment. This reward, in conjunction with the place cells, teaches a gated-mixture-of-experts to estimate the value of each place and provide a dopaminergic RPE signal to the basal ganglia. All events (cues, actions, and rewards) are fed back into the GRU-SR model to update the place cells for future decisions.

Driving Cortex with Place Cells

Within the CBGTPy framework, most neural populations are subdivided into sub-populations referred to as action channels, each of which is tuned to a different action. Many network connections occur within action channels, such that neurons that are tuned to a particular action drive other neurons tuned to the same action. As the network receives dopaminergic feedback, certain action channels are reinforced, leading

to the formation of an action policy. While, in the standard CBGTPy model, every cortical neuron within the same action channel receives the same level of external input, here we tune each cortical neuron to a place cell. During the decision phase and sustained activation phase of the network, the amount by which the neuron is driven above baseline is proportional to the activation of the place cell to which it is tuned. As a result, the amount of input received by downstream striatal SPNs during a particular environmental state is determined by the weights of the corticostriatal connections from the neurons that are tuned to that state. It is this difference that could allow the CBGT network to learn an action policy which is contingent upon the current place cell representation of state. The relationship between place cells, cortical neurons, action channels, and striatal subpopulations is shown in Figure 3.5.

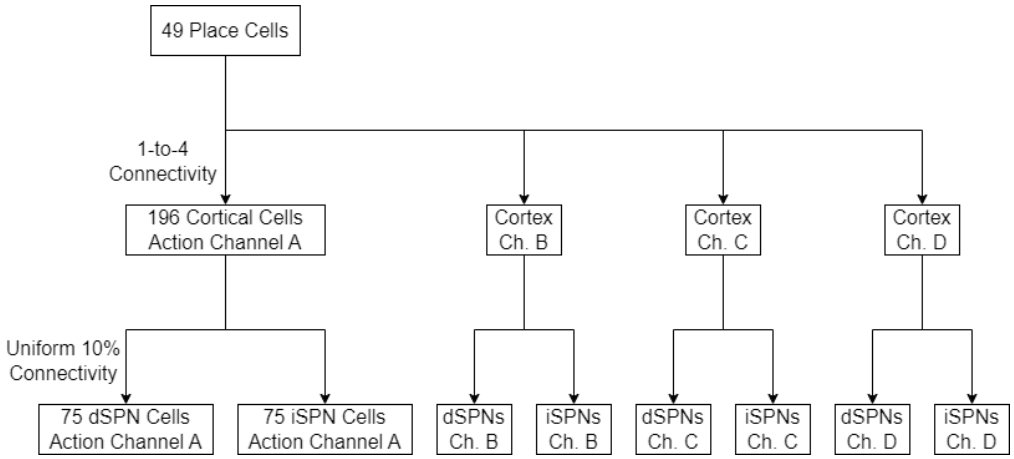


Fig 3.5. Relationship between the place cell vector, cortical cells, striatal cells, and action channels. Each channel contains one cortical population and two striatal subpopulations, which are all spiking components of the CBGTPy model. The place cells, which are simply a vector of values produced by the GRU-SR model, drive the external inputs of the cortical cells during the decision phase of the network. Each cortical cell is tuned to a single place cell, though multiple cortical cells are tuned to each place cell.

To ensure that the addition of cortical tuning would be reflected in downstream neural activity, two main changes were made to the connectivity of the CBGTPy model. First, within-channel recurrent cortical connections were removed, as these connections would weaken the specificity of the cortical neurons by allowing neurons tuned to different places to drive each other. Second, the corticostriatal connections were made sparse, with approximately 10% of all neuron pairs connected, so that each striatal neuron would receive input corresponding to a different combination of cortical neurons. The corticostriatal connections were not random, but rather uniformly distributed so that each place cell would influence the same number of striatal neurons, as shown in Figure 3.6. Within each channel, each of the cortical neurons assigned to the same place cell projected to the same set of striatal cells. Furthermore, the matrix was carefully constructed with highly orthogonal rows to prevent the formation of any spurious relationships between place cells.

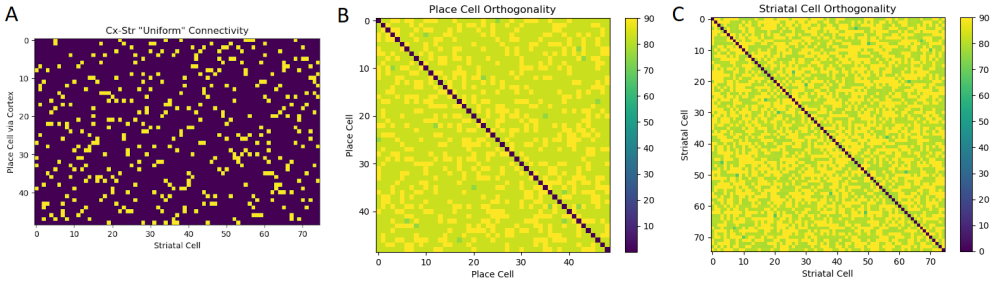


Fig 3.6. Orthogonal uniform connectivity matrix. Each cortical neuron, which is tuned to a single place cell, projects to a subset of striatal neurons. These projections were configured so that the projections tuned to different place cells are maximally orthogonal.

3.2.5 Generating RPEs with Place Cells

To provide dopaminergic feedback to the CBGT network, the agent must be equipped with a method of forming reward prediction errors, which in turn require the formation of a value estimate over environmental states. For this, we use a gated-mixture-of-experts model [86] in which the place cells serve as the gates and each expert learns the value of the corresponding place cell. The formula for the value estimate is thus

$$\hat{V}_t = \frac{\vec{p}_t}{|\vec{p}_t|} \cdot \vec{\epsilon}_t \tag{3.14}$$

where $\vec{p}/|\vec{p}|$ is the normalized place cell vector and $\vec{\epsilon}$ is the vector of experts. This equation is conceptually similar to how value estimates are calculated in SR using $\hat{V}(s) = \sum_{s'} M(s, s')R(s')$.

To calculate a reward prediction error (RPE), the model measures the change in value estimate between two successive decisions and compares this to the amount of reward received between the two decisions.

$$\text{RPE} = \left(\gamma^{b-a}\hat{V}(\vec{p}_b, \vec{\epsilon}) + \sum_{t=a}^{b-1} \gamma^{t-a} R_t\right) - \hat{V}(\vec{p}_a, \vec{\epsilon}) \tag{3.15}$$

Here, a and b are the start times of the decisions, and $\vec{p}_a$ and $\vec{p}_b$ are the place cell vectors used by CBGTPy during the decision processes. This RPE value is then used to supply the CBGT network with dopaminergic feedback as well as update the experts of the mixture-of-experts model.

$$\vec{\epsilon} \leftarrow \vec{\epsilon} + \alpha_\epsilon \cdot \text{RPE} \cdot \vec{\epsilon} \tag{3.16}$$

Here, α_ϵ is a tunable parameter representing the learning rate of the experts.

3.2.6 Investigating Alternative Representations

Within the joint GRU-SR-CBGTPy model, the role of the CBGT network is to learn a complex action policy over the representation provided by the SR model. To investigate how properties of the representation may facilitate or hinder the formation of a complex action policy, two alternative artificial place cell representations were considered, each reflecting different aspects of the task structure. The first alternative was one in which the place cells encode the current trial number within a block, which was expected to lead the agent to form a policy that was purely a function of trial number. The place cells were divided into six groups, each corresponding to one of the

trials within the blocks. This representation was reflective of the fact that the optimal action policy can be described as a sequence of six actions which repeats each block, though it does not capture any information about the latent grid structure and therefore does not capture any dependencies between the outcomes of separate trials within the same block. The second alternative representation was one in which a single place cell was assigned to each of the 49 square locations on the latent grid, expected to lead the agent to form a policy that was purely a function of latent position. Each place cell was only active when the agent's latent position matched the assigned position of the cell. This representation was reflective of the fact that the optimal policy can also be described a function of position, though it does not quite completely capture the path-dependency of rewards. By simply replacing the GRU-SR place cell model with these alternative representations, the CBGTPy network and gated mixture-of-experts model can be used without modification.

As an additional experiment, we investigated the potential importance of structure within the organization of corticostriatal connectivity. If the HC-EC circuit is responsible for learning the structure of the task and conveying this information in the form of place-cell-like representations, then it is unlikely that much of this learned structure would be reflected in the organization of the corticostriatal synapses. Any relationship between place cell representations and corticostriatal connectivity would require a more complex interplay between CBGT and HC-EC circuit dynamics during the formation of the place cell receptive fields. For this reason, the baseline model assumes that corticostriatal synapses are segregated by action channel, but not by place. While each cortical neuron is tuned to a specific place cell, the striatal neurons receive input from a mixture of cortical neurons and is not tuned by place. The ability of the CBGT network to learn, however, is contingent upon its ability to assign credit to the correct synapses. For any given cortical and striatal neuron pair, the striatal neuron may be highly active in a wider variety of states than the cortical neuron, which could reduce the efficacy of the local plasticity rule. To determine whether the segregation of corticostriatal synapses by place is crucial for the performance of the network, we investigated how the ability of the network to learn an action policy over trials was influenced by the degree of synaptic organization. The trial-based network, described in the preceding paragraph, was trained using a range of corticostriatal connectivity matrices in a parameter sweep. These matrices, shown in Figure 3.7, ranged from fully segregated into six groups to uniformly mixed. At one extreme, the model with segregated synapses has six groups of striatal cells, each receiving inputs from only one of the six groups of cortical cells, so each striatal cell is essentially tuned to a single place. At the opposite extreme, the model with uniformly distributed synapses would have striatal cells that were equally driven by each group of place cells, meaning the postsynaptic activity of each synapse is independent of the current place. By measuring the change in performance as corticostriatal connections move from segregated to diffuse, it is possible to determine the extent to which effective credit assignment is reliant on the organization of these synapses.

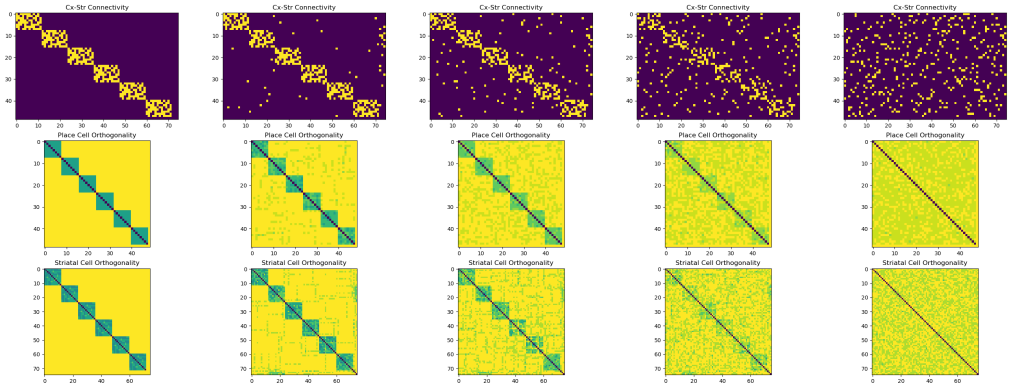


Fig 3.7. Varying segregation of corticostriatal synapses. On the left, the corticostriatal connections form 6 distinct sub-action channels, one for each state in the agent's environmental model. On the right, the corticostriatal connections are uniform with no relationship to place.

3.3 Results

3.3.1 GRU-SR Model Learns the Task Structure

We found that, after training the GRU-SR model on random sequences from the latent grid task, the place cells were able to effectively encode the structure of the task and the population vector was highly predictive of reward value. Figure 3.8 shows that the model produces a GRU cell, a basis cell, a raw SR cell, and ultimately a place cell representation vector for each step of the task. When comparing the internal state cells of the GRU to the place cells, it is evident that while GRU cells often abruptly change in activity from step to step, the place cells tend to change in activity more smoothly over time. This smoothness is measurable as an increase in the auto-correlation of cell activity, as visualized in Figure 3.9, and this is a clear consequence of the predictive-coding nature of SR. The SR matrix $\tilde{M}$ is shown in Figure 3.10.

The place cell vector after event x_t was highly predictive of the event x_{t+1}, except of course in the case of actions, which were random and thus impossible to predict. This indicates that the GRU successfully learned to encode an estimate of environmental state within its hidden state vector and that this state estimate was reflected by the place cells. Figure 3.11 shows a partial timecourse of a single experiment, depicting the GRU-SR model's sequence of predicted events, colored to indicate the accuracy of each prediction. Each column represents one step of the simulation, and the rows represent the potential events which can occur. On steps where the agent is prompted to select an action, the GRU-SR model tends to predict each of the four actions equally, reflective of the random behavior upon which it was trained. For steps on which the agent is due for a reward or penalty, the linear classifier forms much stronger predictions, reflective of the deterministic nature of the environment in this task. When considering only predictions for the outcomes of trials, the GRU-SR model achieved an accuracy of 97.9% across a test data set of 100 runs of 408 trials each. This indicates that the GRU-SR model has learned the vast majority of the structure of the task and that this structural information was present in the produced place cells.

To determine whether these place cells demonstrate selectivity towards specific squares in the latent space, the receptive field of each cell was plotted by computing the average activation of the cell across all visits to each latent square. The receptive fields are shown in Figure 3.12, along with the receptive fields of the other cell types of the GRU-SR model from which the place cells are derived. These fields indicate that the

majority of place cells are selective towards particular regions of the grid, with roughly one-third of modeled place cells showing a tuning towards a small contiguous region akin to the fields of biological place cells, while other place cells are tuned to larger areas like groups of rows or columns. To better quantify the size of these receptive fields, each field was normalized to sum to 1, and the Kullback–Leibler (KL) divergence was calculated for each field relative to a uniform distribution across the 49 squares of the latent grid. The result is a number, measured in bits, which expressed how selective each field is relative to an uninformative field. As shown in Figure 3.13, most place cells had receptive fields which were smaller than half the size of the latent grid.

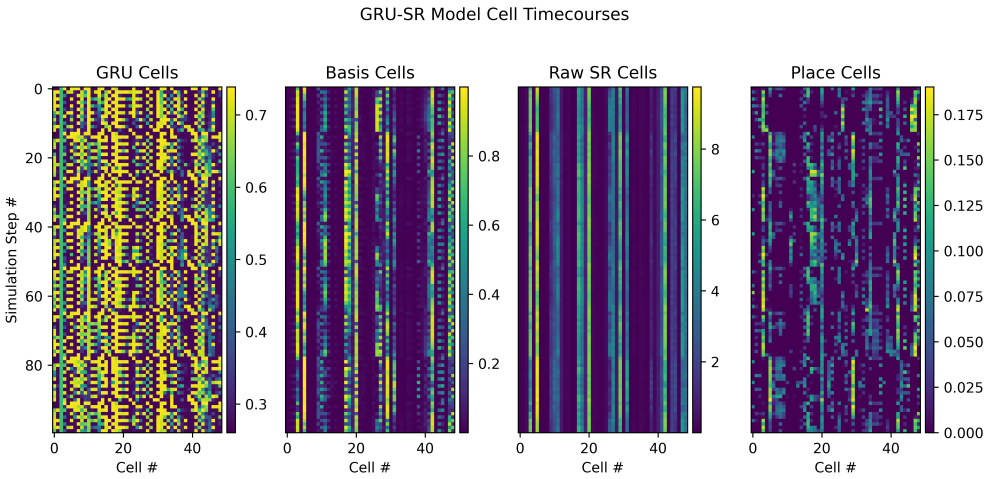


Fig 3.8. Partial timecourses of simulated cell activity in the GRU-SR model. From left to right: the state of the GRU, the basis features extracted from the GRU, the successor features derived from the basis features, and finally the place cells created by re-scaling and thresholding the successor features.

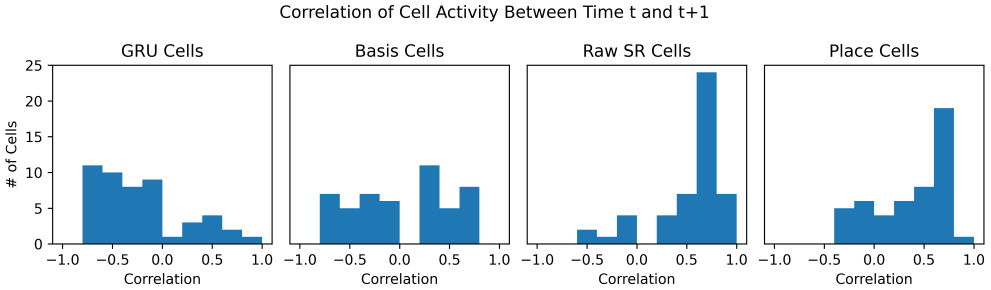


Fig 3.9. Histograms of cell auto-correlations between times t and $t+1$. For each cell in each population, a correlation was computed to measure the extent to which the cell's activity at time t predicted its activity at time $t+1$. Higher correlation values indicate cells whose activities change gradually over time.

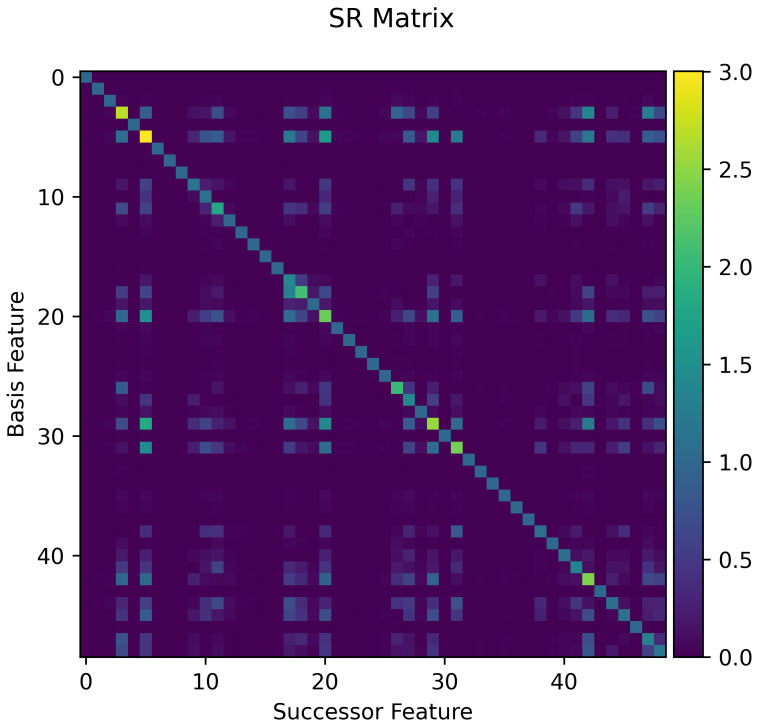


Fig 3.10. Computed successor matrix $\tilde{M}$. This matrix indicates the relationship between the 49 basis features and the 49 successor features.

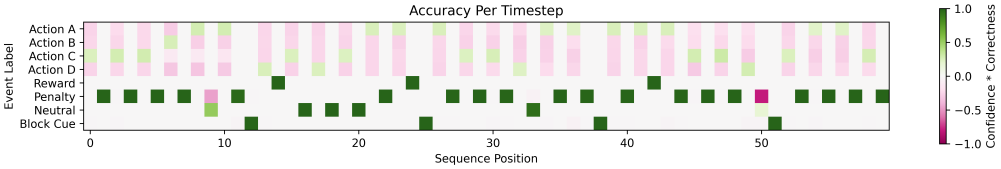


Fig 3.11. Sensory predictions of logistic classifier trained on modeled place cells. Shown are the first 60 events of a single run of the task. The plotted intensities indicate the classifier's prediction at each step, and the colors indicate the accuracy of that prediction.

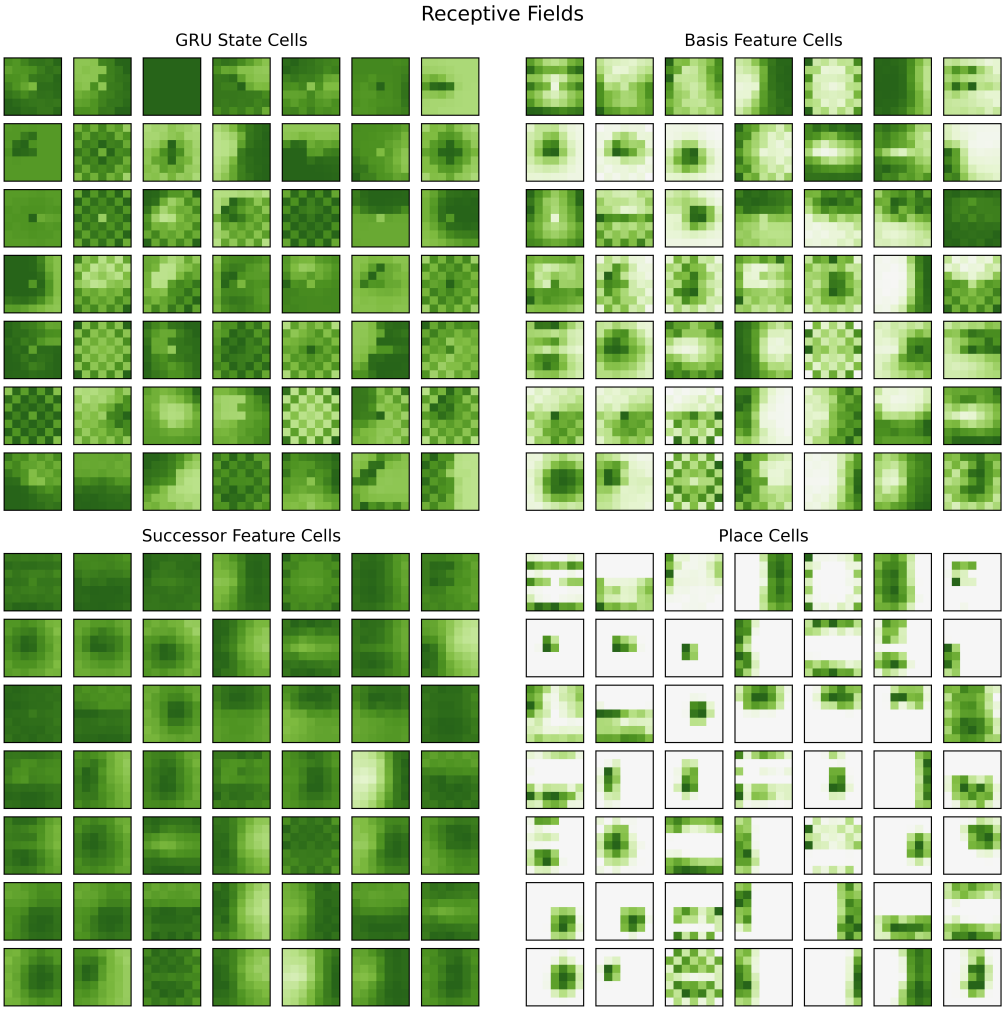


Fig 3.12. Receptive fields (green) of modeled place cells. For each cell population, the activity of each cell was averaged across all visits to each latent grid square.

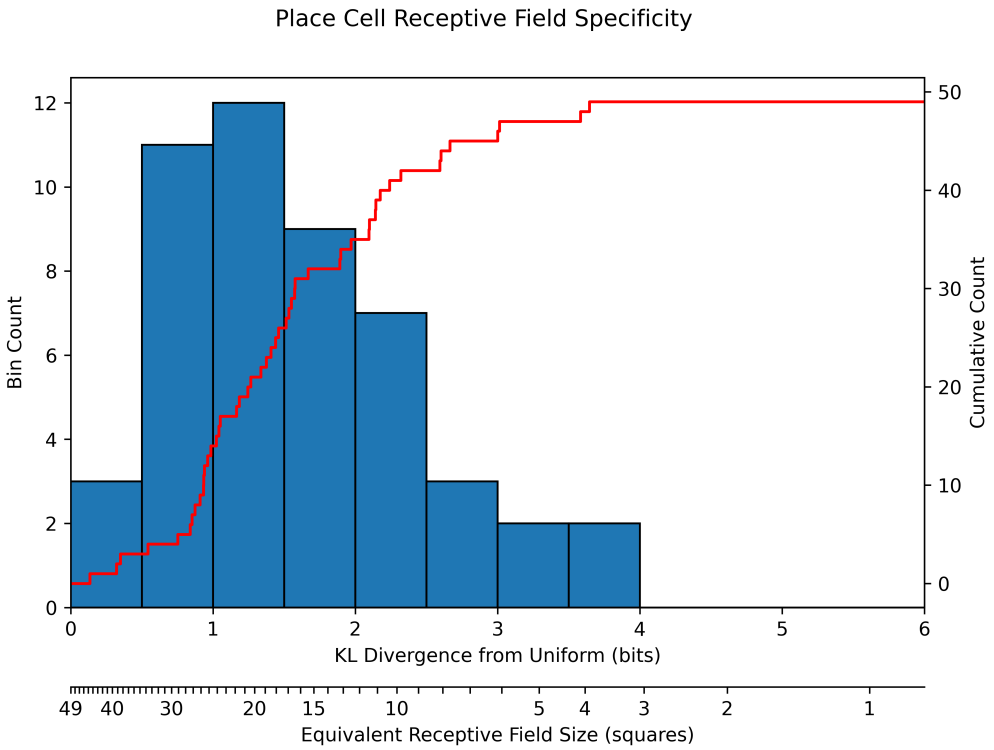


Fig 3.13. Size distribution of pseudo-spatial receptive fields. For each place cell, the Kullback–Leibler (KL) divergence was calculated between the cell's receptive field (see Figure 3.12) and a uniform receptive field. Higher KL divergence corresponds to a more focused receptive field.

3.3.2 Joint GRU-SR-CBGTPy Model Performance

Even though the GRU-SR model can effectively learn how latent position determines the trialwise rewards, when the GRU-SR place cells were supplied as cortical inputs to the CBGTPy network, the model was unable to learn the optimal decision path. On average, the model performed only around 2 of the 6 steps of the path correctly, as shown in Figure 3.14, and it was extremely rare for the network to perform the entire path correctly. The average total reward received by the network hovered between +1 and +2 per block.

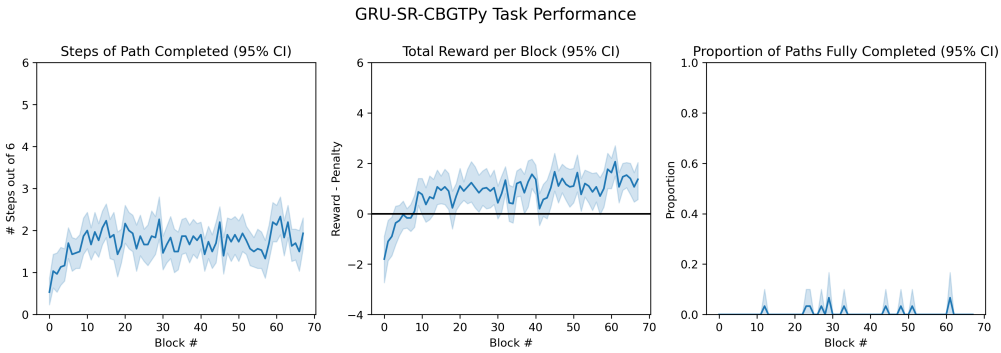


Fig 3.14. Performance of the complete GRU-SR-CBGT model. Left: The average number of correct steps followed by the network along the optimal path. Center: The average total reward received in each block. Right: The proportion of blocks in which the agent performed the entire optimal path.

Although the network failed to learn the full action policy, much of the structure contained within the place cell representations was reflected in the weights learned by the network. Figure 3.15 shows the final average learned weights of the network, calculated by averaging together the corticostriatal synapses of all cortical neurons that are receptive to the same place cell, separated by channel, and subtracting the indirect weights from the direct weights. The resulting plot shows, for each place cell, which action channels undergo net excitation or net inhibition by the activity of that place cell. In essence, this shows how each place cell contributed to the decision dynamics and therefore overall action policy of the network.

Although Figure 3.15 shows the existence of structure in the learned corticostriatal weights, it is also important to determine whether this structure in weights is reflective of the latent state space structure encoded in the place cell representation. For each pair of place cell and action, the correlation between the cell's activity and the true expected reward value of that action was measured. Since the values of the actions depend on the agent's location, these correlations are expected to exist, and the strength of a cell's correlations indicates its predictive utility. The strength of the place cell-action value correlation was then plotted against the average learned corticostriatal synapse weight for that place cell and action channel. These plots, shown in Figure 3.16, demonstrate the existence of a clearly visible and approximately linear relationship between the predictive utility of the place cells and the magnitude of the network's learned weights. This potentially indicates that, despite the behavior of the network, all the structural information of the place cells, and thus all the information necessary to solve the task, has been encoded into the weights of the network.

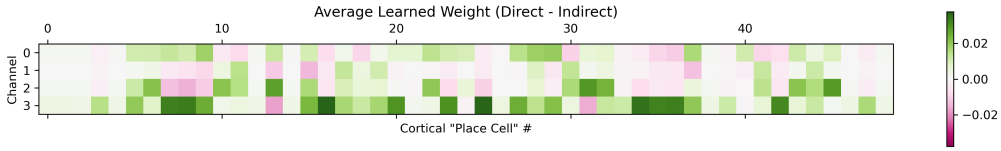


Fig 3.15. Average learned weights by action channel. The final learned weight values were averaged across all corticostriatal synapses whose cortical neurons received input from the same place cell. This shows the activation of a single place cell influences the balance of the network towards or against each action, essentially showing the action policy the network learned over the place cell representations.

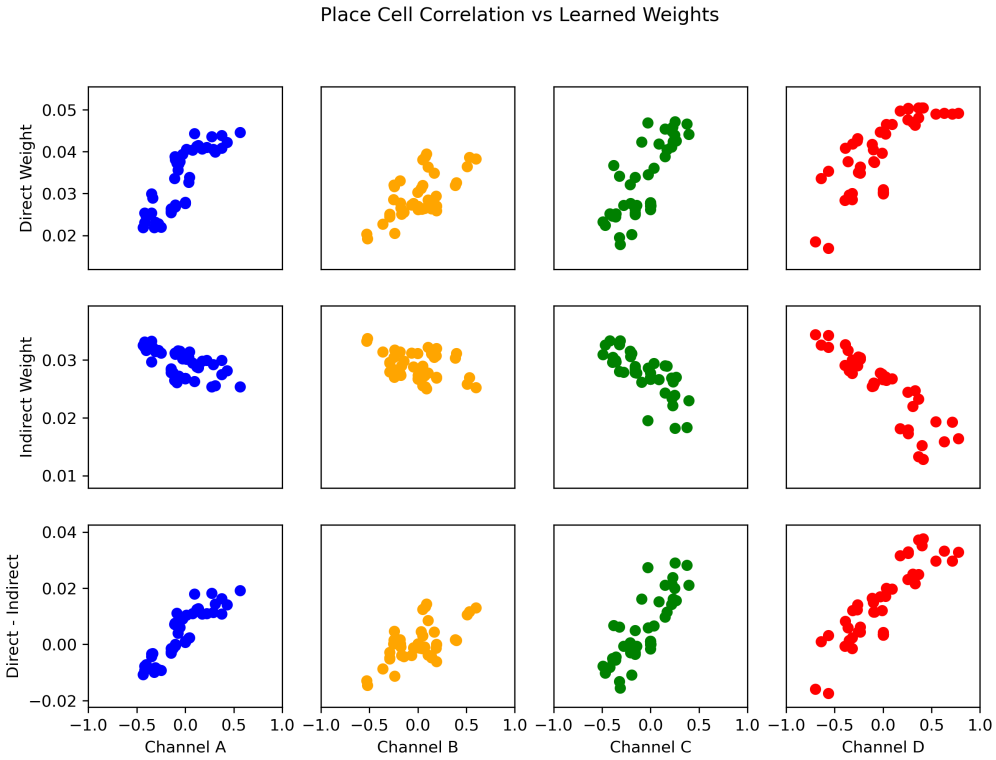


Fig 3.16. The relationship between learned weights and the correlation of place cell activity to action values. For each place cell, and each action, a correlation was computed between the activation of the place cell and the reward received after selecting that particular action. This correlation was then plotted against the weights that were learned for that place cell.

3.3.3 Performance of Alternative Place Cell Regimes

To qualitatively compare the GRU-SR place cells with the trial-based and latent-square-based place cells, the average activity of each place cell, across random behavioral trajectories, was computed both on a per-square and per-trial basis. The plots, in Figure 3.17, depict the pseudo-spatial and temporal receptive fields of each place cell under each representation scheme. The GRU-SR place cells formed receptive fields that were receptive to specific areas of the latent grid and also demonstrated some amount of temporal structure. The place cells that were tuned to trial number, as expected, had the lowest degree of spatial selectivity, as the cells could not distinguish between squares that were the same distance from the origin. When the place cells were tuned to individual squares, the cells had the maximum degree of spatial selectivity but had very diffuse temporal selectivity. This final representation was very sparse, with only one cell active on each trial.

When the CBGTPy network was provided place cells that were tuned to trial number, the performance of the network greatly increased relative to the original GRU-SR place cells, with the network learning to reliably perform between 4 and 5 steps of the optimal path, over double the performance of the GRU-SR network. The average performance by block is shown in Figure 3.18. This network was also capable of occasionally performing the full optimal path, though still only on a minority of blocks. Furthermore, the introduction of additional structure to the corticostriatal weights, reflecting the six groups within each action channel, enhanced the performance of the

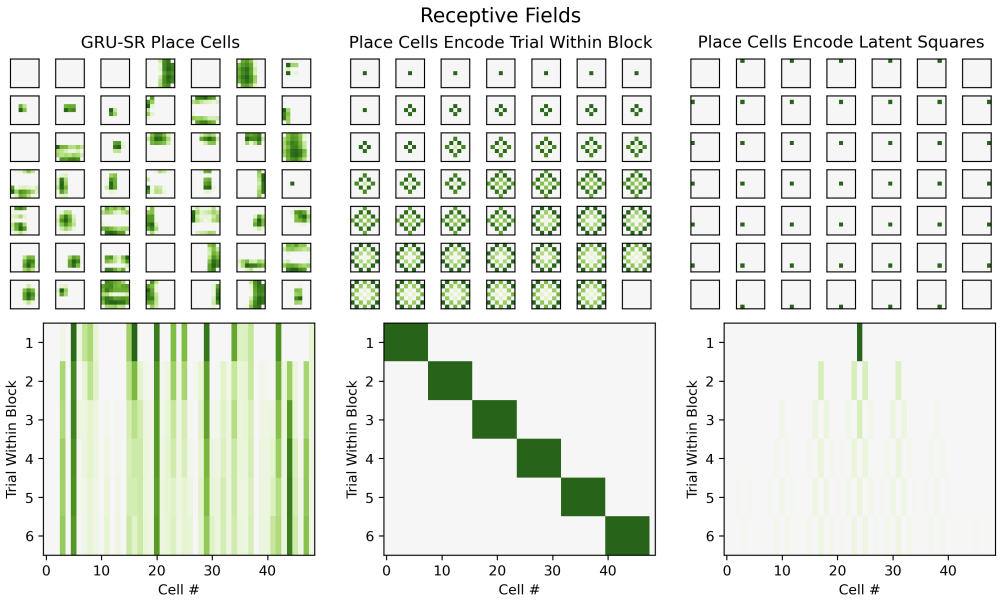


Fig 3.17. Comparison of the three modeled place cell regimes. Left: The pseudo-spatial and temporal receptive fields of place cells produced by the GRU-SR model. Center: The place cells are divided into 6 equal groups, each of which is active during a single trial within a block. Right: Each of the 49 place cells is selective towards exactly 1 square of the latent space. These receptive fields were computed while only considering the population vectors that occur directly prior to an action selection, i.e. the vectors over which the network learns an action policy, not any intermediate place cell representation.

network. For each connectivity matrix in the sweep outlined in Figure 3.7, the average performance of the simulated agent was calculated and plotted in Figure 3.19. The network achieved the highest performance when the synapses were completely segregated, and the performance dropped slightly as the synapses become more uniformly distributed.

While Figure 3.18 shows good performance when the place cell groups were each tuned to precisely one of the six trials of a block, increasing the temporal discounting γ of this representation had a strong negative effect on performance. As the γ increased, each place cell was increasingly activated during the trials preceding the trial to which it was primarily tuned, creating overlap in the activation patterns of the six place cell groups. As γ approached 1, the modeled place cells became equally active during each trial, gradually losing all tuning selectivity. As shown in Figure 3.20, the gradual increase of γ resulted in a steady decline in performance, as the network lost the ability to form an independent action policy for each trial. At the far extreme, when $\gamma = 1$, the network received no structural information from the place cells, which were all equally active at all times. When the average behavior of the network was plotted by trial (Figure 3.21), low gamma values enabled the network to form a distinct policy for each trial, while higher values resulted in a blurring of action policy across trials.

When provided with place cells that were precisely tuned to the individual latent squares, the network performed poorly (Figure 3.22), with results that were slightly worse than the original GRU-SR model and greatly worse than the network with the trial-based cells. As only 1 of 49 place cells was active each trial, only 4 of the 196 cortical neurons per channel drove the network's decision dynamics. To determine whether the low number of active cortical neurons was at fault for the poor

performance, an expanded version of the network was tested with double the number of cortical neurons, a total of 392 neurons per action channel. The number of place cells was likewise doubled so that 2 of 98 were active each trial. With the increased input size, the model dramatically recovered its performance (Figure 3.23). These results are comparable to the scores achieved by network with the trial-based representation (Figure 3.18). Among the tested configurations, the expanded latent square network had the highest path completion rate, executing the optimal sequence of six actions in over 20% of all blocks. Taken together, the results of this section illustrate that the properties of the supplied input representations can greatly influence the success of the network.

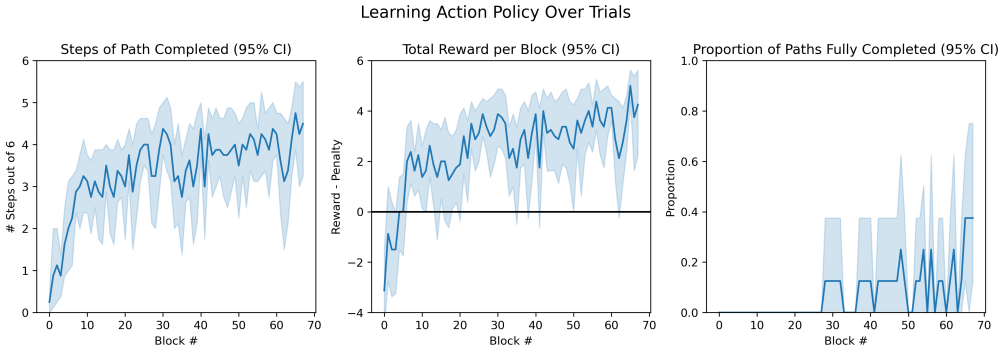


Fig 3.18. Behavior with a trial-based action policy. The network received a place cell representation which encoded the trial number, from 1 to 6, within the block.

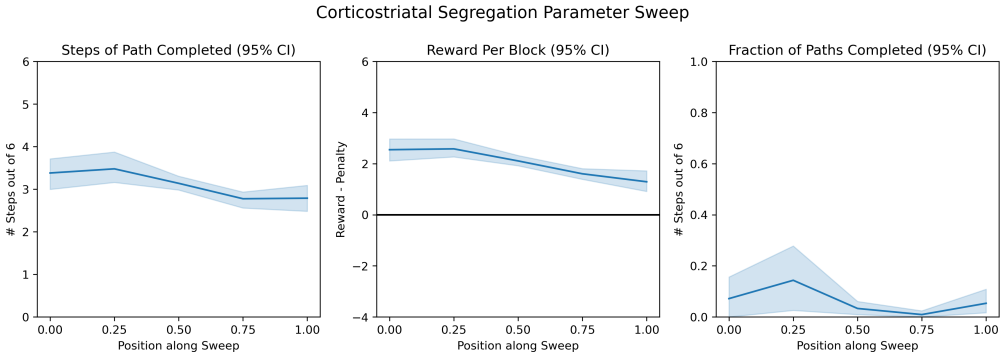


Fig 3.19. Effect of corticostriatal synapse segregation on performance. As the corticostriatal synapses become more uniformly distributed, network performance drops slightly.

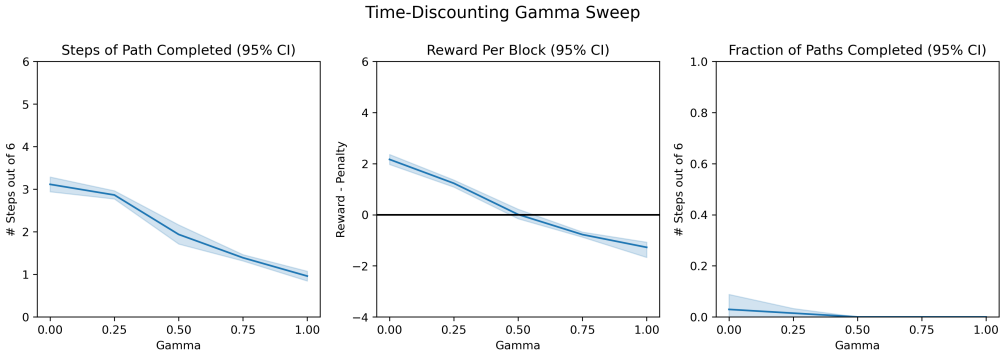


Fig 3.20. Effect of SR temporal discount γ on performance. The network receive a place cell representation consisting of 6 groups, each tuned to a single trial, as well as to the preceding trials according to the selected value of γ. When γ was 0, each group was only active during 1 trial, and when γ was 1, each group was equally active during all 6 trials. As γ increased, the performance of the network degraded significantly.

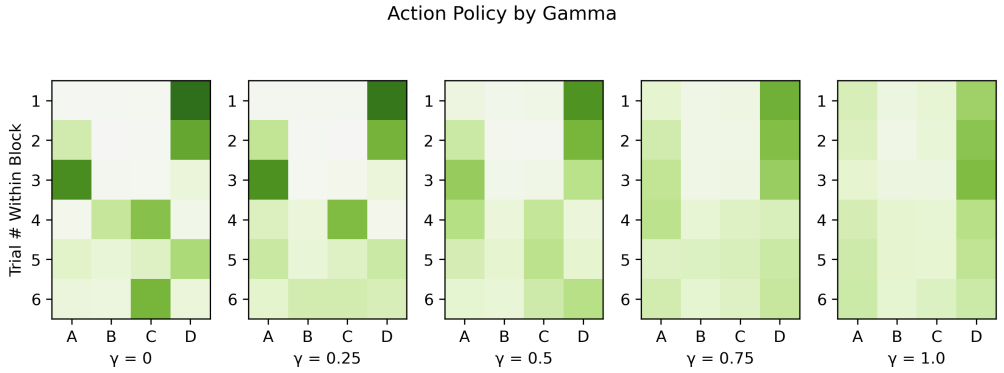


Fig 3.21. Effect of SR temporal discount γ on the learned action policy. For each condition from Figure 3.20, the behavior of the network was averaged across all simulations and all blocks. This produces a heatmap (in green) representing the average policy for each of the six trials.

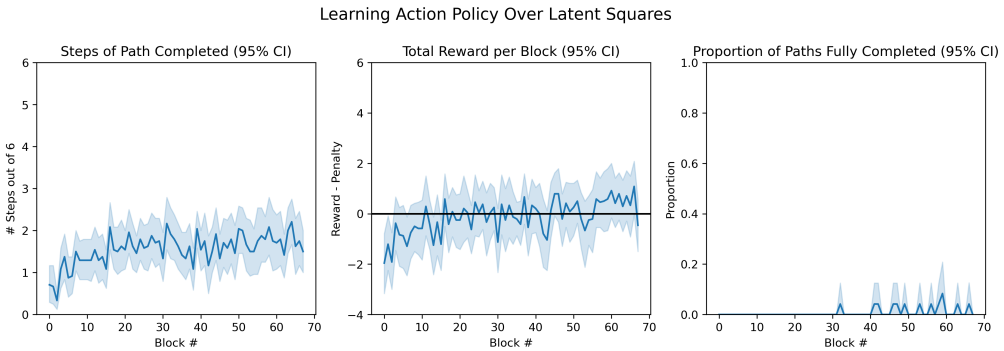


Fig 3.22. Behavior with policy over the latent squares. The network received a place cell representation which directly encoded the agent's latent square coordinate, with one place cell active at each square.

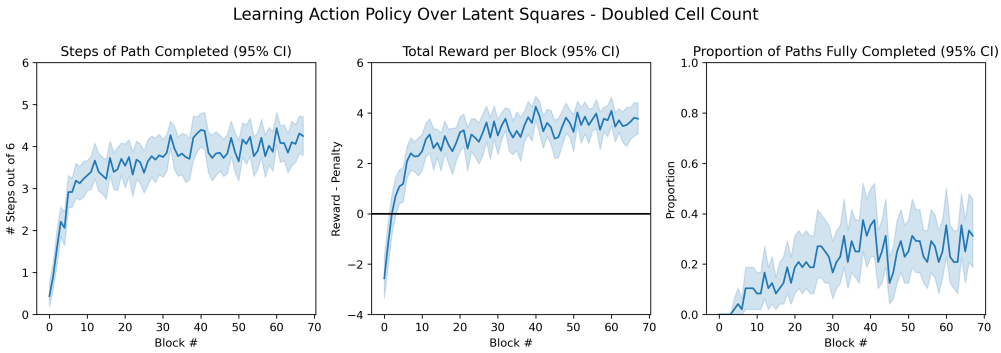


Fig 3.23. Behavior with policy over the latent squares with doubled cortical population size. Similar to Figure 3.22, the network received a representation encoding latent squares, but with two place cells per square. The number of cortical neurons was likewise doubled from 196 to 392 per channel.

3.4 Discussion

In this chapter, we investigated how the CBGT circuit and HC-EC system could potentially cooperate [18] to solve complex reinforcement learning tasks in structured environments. To accomplish this, we placed the CBGTPy model [24] within the latent grid environment from Bond [9], and integrated it with a variety of place cell representations, each encoding different aspects of environmental state. Leveraging the SR place cell hypothesis [8] [6], we developed a model of place cells that exhibits biologically-realistic properties in non-spatial contexts. For comparison, we also handcrafted two place cell models that encoded solely the temporal or pseudo-spatial component of the state, in which the network would either learn an action policy over trials or over grid squares respectively. We found that the performance of the network varied greatly by the features of the supplied place cell representation and was also influenced by structure embedded in the corticostriatal synaptic connectivity. Notably, the combined network was unable to consistently learn the complete task. These results point towards an important relationship between the structure of cortical input representations and the learning capabilities of the basal ganglia, and furthermore gives important clues about the nature of the basal ganglia-hippocampal interactions necessary for solving structured RL tasks.

Starting from de Cothi and Barry's model of SR place cells [8] in spatial navigation, we were able to successfully develop an equivalent model for our non-spatial RL task by replacing the spatially-tuned boundary vector cells with alternative basis features extracted from a gated recurrent network [31]. By training this GRU-SR model on symbolic sequences of events, such as rewards and actions, the resultant place cells obeyed the predictive-coding nature of real place cells. Simultaneously, these place cells encoded an estimate of environmental state, as indicated by the model's ability to predict future events. As a linear classifier was capable of decoding the upcoming reward value from the place cell representation with 97.9% accuracy, it is clear the artificial representations reflected essentially the entire structure of the structured RL task. The majority of these artificial cells, despite being trained on wholly non-spatial data, formed receptive fields that were spatially-selective with respect to the latent grid space of the task. These results support the notion that, if the HC-EC circuit is indeed a center for learning SRs, it is possible the same computational learning mechanisms which give rise to the classical place cell receptive fields in spatial environments could also produce receptive fields selective to agent's position in non-spatial environmental

state spaces.

Although the majority of the simulated GRU-SR cells were selective to small, connected regions of the latent grid, some cells were found to be selective to the majority of the grid or to multiple regions of the grid. The failure of some GRU-SR cells to form place-cell-like fields could be attributed to a few factors. First, the true latent structure is not purely grid-shaped, as the task is divided into blocks of 6 trials each, within each trial there is an alternation between action and reward, and each square only dispenses a reward upon the first visit within a block. All of this additional structural information is necessary to predict the occurrence of rewards and thus must also be encoded within the place cell activation patterns. Secondly, the mathematical definition of SR for distributed representations [8] does not require each basis feature to be tuned to a single latent location. The GRU-SR model can achieve a low predictive loss as long as there is still sufficient information within the basis feature vector as a whole. As biological place cell representations are sparse, likely resulting from a form of competitive learning [87], this indicates that the SR model is not a complete description of the behavior of place cells and that additional computational mechanisms are at work to produce cells selective to singular latent locations.

When joined with the CBGT network model, the combined GRU-SR-CBGTPy model failed to achieve strong performance in the latent grid task, with an average of two out of six steps completed and an average net reward that was barely positive. Although the structure of the artificial place cell representation was reflected in the learned weights of the network, the network was unable to learn the complex action policy necessary to solve the task. Simulations performed with the alternative, hand-crafted representations resulted in varying degrees of success, with the highest performance achieved by a representation in which the cells encoded only the repetitive structure of the trials and not any latent spatial information. When the cells encoded the coordinates of the latent grid, however, network performance was greatly reduced. As the optimal action policy, a sequence of six actions forming a path on the latent grid, can be equally described as either a policy over trials or a policy over latent squares, both of these alternative representations contain sufficient structural information to solve the task. Therefore, the differences in performance are likely attributable to differences in other properties of these representations and effects of these properties on the dynamics of the basal ganglia.

The first major influential property of the representations appears to be sparsity. The latent square representation was very sparse, with only around 2% of place cells active in each state, and this sparsity likely explains the failure of the network to learn with that configuration. With the sparse representation, very few cortical neurons were driven by external input on each trial, leading to very low levels of activity in both the direct and indirect pathways, as well as few synapses eligible for dopaminergic plasticity. Previous modeling work, in which the dynamics of the CBGT circuit were compared to the drift-diffusion model of decision-making, has shown that activity levels in several nuclei are strongly correlated with the behavioral properties of the network [44]. In particular, low activity of indirect SPNs was associated with a low standard of evidence when selecting actions. Furthermore, limited contrast in activity between channels is associated with a lower level of commitment towards any particular action choice [44]. Thus, highly sparse place cell representations may lead to an inability of the network to form a decisive action policy. When the number of active corticostriatal synapses is small, the information encoded in those synaptic weights does not sufficiently influence the downstream evidence accumulation dynamics of the network, regardless of whether the learned weights reflect the correct policy. Importantly, results from an expanded network show that increasing the number of cortical neurons counteracts this effect and produces successful learning. This can be viewed as increasing the signal-to-noise ratio

of the network, which is fully-spiking. Overall, it appears critical that the cortical input representations possess a certain minimum level of total activity to ensure effective learning and decision-making by the CBGT network.

A second important property of the cortical inputs appears to be the orthogonality of the representations of distinct states, as the network appears to form incorrect generalizations when the representations are not sufficiently orthogonal. Generalization, or the ability of the agent to extrapolate learned behavior to related observations, is both a key feature and also a core challenge of RL [88]. Fundamentally, generalization relies on the transfer of value information from the experienced states and actions to related states and related actions [89]. Within the context of biological RL, this transfer relies on the underlying neural representations [89], and within the CBGT circuit specifically, this transfer is mediated both by the corticostriatal synapses and the dopaminergic feedback signal as part of the actor-critic framework [90]. In the joint model, similarity in place cell representations affects both the state value estimates learned by the mixture-of-experts model and the weights learned by the corticostriatal synapses. The highest-performing representation was one in which the cells were divided into six groups completely selective to the six individual trials of a block. This representation works to minimize generalization across the trials, thereby preventing conflation of the learned action values, which is critical for the modeled task, in which the correct action varies between trials.

When the input representations are not completely orthogonal, some proportion of place cells, and likewise corticostriatal synapses, are active in multiple distinct states. As a direct result, dopaminergic feedback which occurs while the agent is in one state affects the action policies in related states, as determined by the degree of representational overlap. While this mechanism allows for the generalization between similar states, it can clearly be detrimental when the similarity of representations does not reflect the ideal degree of generalization. When the six-trial representation was adjusted by increasing its temporal discounting, γ, over a range from 0 to 1, the six groups became less selective and the performance of the network was greatly impeded. These higher values of γ meant that the cells would become increasingly active in the trials preceding the ones to which they were tuned, in accordance with the predictive coding nature of SR, an effect that greatly increased the representational similarity of adjacent trials. This increased similarity, and therefore decreased orthogonality, presumably causes over-generalization, or conflation, by the network during the learning process. This conflation at high γ was reflected as a high degree of similarity between the learned action policies for each trial. This result demonstrates an important obstacle for the use of SR-like representations by the basal ganglia, as temporal discounting is a core component of SR [25]. Without a high discounting factor, an SR-based agent would be unable to consider the long-range consequences of its actions, but a high discounting factor leads to representations whose similarities are primarily determined by their temporal proximity. This dramatically reduces the ability of the network to learn an action policy that varies from step to step, as is the case with the optimal policy of the example task.

Overall, it appears that, in order for the CBGT network to effectively learn over a particular representation, the representation must have at least these two key properties. Each representation must contain enough active neurons to facilitate effective decision-making dynamics [44], and the overlaps between representations must be limited to prevent over-generalization. These two requirements appear to be at odds with place-cell-like and SR-like representations, which tend to be sparse [87] and smoothed across adjacent states [8]. Additional computational mechanisms, beyond those modeled here, are likely required to leverage the structural information present in place-cell-like representations. These mechanisms would in turn enable the agent to

solve complex structured RL tasks such as the latent grid problem.

First, it is possible that previously-formed structure within the organization of the corticostriatal synapses could aid in learning, as the CBGT circuit is known to be topographically organized [91]. When the corticostriatal synapses were segregated, matching the structure encoded in the cortical inputs, performance moderately improved (Figure 3.19). As a result of this segregation, corticostriatal synapses whose pre-synaptic cortical neurons were tuned to a particular state experienced less post-synaptic activity in other states, likely leading to less spurious eligibility and more accurate credit assignment. Theoretically, the formation of the place cell receptive fields, during structure learning in the HC-EC system, could be guided by the connectivity of the CBGT loops as to maximally leverage pre-existing corticostriatal structure. There is likely a limit, however, to the extent to which this pre-existing structure can match the shape of a particular task's state space. This process might also require a biological mechanism for extracting information about this corticostriatal structure from the striatum and passing it to the hippocampus during structure learning, and it is currently unclear how this process would be supported by the physiology [27]. Ultimately, our results suggest that a guiding influence from the basal ganglia upon HC-EC computations might be useful but not critical for performance. We should rather ask how the CBGT circuit could better leverage the information provided by place cells.

As the CA1 neurons, which are the cells argued to encode the SR matrix [6], form the primary output of the hippocampus and are known to project to the prefrontal cortex [27], is it clear that the cortical inputs of the CBGT circuit have direct access to place cell information. There is room, however, for additional layers of processing, both within the cortex and by the CBGT circuit. As the CBGT circuit is hierarchically organized, information from prefrontal cortex may go through multiple iterations of processing by the basal ganglia before ultimately influencing action selection in motor cortex [17]. As a result of this hierarchical organization, the basal ganglia are able to form complex conditional action policies, which alone are theoretically sufficient for solving structured tasks [17]. Quite relevantly, one optogenetic study in mice found evidence for a hierarchical organization in the learned action policy of a sequence-based task [92], indicating that the basal ganglia may apply its own hierarchy-based version of structure learning even while performing non-hierarchical tasks. Thus it makes sense to consider the process of structure learning within RL tasks to be shared between the hippocampus and the basal ganglia, rather than entirely governed by one or the other. The basal ganglia may learn a hierarchical structure, while the place cells may provide an efficient set of conditional rules upon which the hierarchy can be built. The lack of a hierarchical component may be the key factor preventing our CBGTPy model from fully solving structured tasks, even when the modeled place cells contain enough information to accurately predict the outcomes of actions.

Lastly, biological CA1 neurons likely exhibit more heterogeneity than is reflected by the SR model presented here, as certain CA1 subpopulations appear to explicitly encode temporal information [93] [94]. Recent calcium imaging in mice has indicated that temporal information in dorsal CA1 neurons can influence decision-making [95]. Our artificial SR-derived place cells appear to encode much more spatial information than temporal information, as shown in the receptive fields of Figure 3.17. The inclusion of a stronger temporally-coded signal would likely be highly beneficial for tasks in which a precise sequence of actions is needed to achieve the maximal reward. An enhanced space-time encoding, which combines the properties of the two handcrafted representations in Figure 3.17, could potentially facilitate very effective learning. Another avenue for exploration could be the introduction of a heterogeneous discounting factor in the SR model, producing cells with differing time horizons, thereby introducing more temporal information. In general, larger changes in the place cell

activation pattern from trial to trial should assist the basal ganglia in learning an action policy that is heavily contingent upon the precise trial number.

In conclusion, the HC-EC system, as a center of predictive coding, appears to be an ideal center for structure learning. A model of place cells, derived from the predictive code of SR, was able to process non-spatial observations during a simulated RL task. This place cell model produced neural representations reflective of an environmental state estimate that was sufficient to predict action outcomes with over 97% accuracy. When joined with the CBGTPy network, however, the joint model was not able to fully learn the optimal policy for the structured task. Through an investigation of alternative place cell representations, it was found that the tendency of place cell representations to overlap led to harmful over-generalizations, and attempts to reduce this over-generalization required either increasing the representational sparsity, which harmed the fundamental accumulative dynamics of the network, or sacrificing the amount of structural information contained within. Although the place cells of the HC-EC system can provide a very efficient and natural representation of the structure of an RL task, it appears that the CBGT circuit cannot efficiently learn an action policy directly over these representations. Rather, the hierarchical organization of the basal ganglia may be the key to leveraging place-cell-like information in cortex, with the basal ganglia simultaneously performing its own hierarchy-based form of structure learning. Future work should focus on understanding the computational interplay of these two forms of structure learning, both on a theoretical basis, and through the potential extension of the CBGTPy model with a hierarchical component.

Chapter 4

Conclusion

4.1 Summary of Results

The objective of this dissertation was to investigate two core problems surrounding the dynamics of the cortico-basal ganglia-thalamic (CBGT) circuit in reinforcement learning. First, to study the system dynamics during the decision process, I developed CBGTPy, a computational model incorporating known circuit anatomy and featuring realistic corticostriatal plasticity. This model demonstrates accurate single-channel and multi-channel dynamics, enabling researchers to explore hypotheses about these dynamics across various manipulated experimental conditions. Second, to understand how the computations of the CBGT circuit could coordinate with hippocampal structure learning, I developed a model of place cells and integrated these cells as inputs to the CBGTPy model during a structured reinforcement learning task. Although the joint model achieved limited success, the simulation results revealed how particular properties of the place cell representations could affect the learning processes in the basal ganglia. Together, these accomplishments build a bridge between the intrinsically model-free plasticity of the CBGT circuit and the model-based reinforcement learning computations necessary for humans and other mammals to efficiently solve the complex and structured tasks typical of real-world environments.

4.1.1 The CBGTPy Framework

In Chapter 2, I address my first aim: the creation of CBGTPy. This Python package provides a flexible and extensible platform for simulating goal-directed agents with internal CBGT circuit dynamics. By including spiking neural populations, relevant neural pathways, and dopamine-induced synaptic plasticity, the network produces realistic single-channel and multi-channel dynamics, such as go/no-go behavior and competitive action selection. Environmental feedback generates reward prediction errors and dopaminergic feedback, altering the corticostriatal pathway balance and guiding the formation of the network's action policy. By generating complete simulated experimental timecourses, the CBGTPy framework enables researchers to investigate the relationships between circuit physiology, dynamics, and behavior.

A core feature of CBGTPy is its flexibility, embodied in its data-flow programming and agent-environment paradigms. The data-flow paradigm distinguishes between model specification and model execution, with smaller code blocks forming steps in the larger simulation pipeline. This organization allows for efficient modification of the simulation. CBGTPy's integration with multiprocessing libraries enables the final assembled pipeline to be applied to many alternative parameter inputs in parallel. The

agent-environment organization separates the network agent properties, which contain the CBGT neural circuitry, from its experimental environment, which presents stimuli and provides rewards in response to the agent's behavior. This separation allows for independent modification of each component and the application of the same agent to various experimental conditions and manipulations.

The capabilities of CBGTPy were demonstrated with three example experiments, each showcasing important features of the platform. The first example was the n-choice task, where the agent had to select a single action from multiple alternatives, with rewards that were probabilistic and could change over time. During the evidence-accumulation phase, competition between multiple action channels resulted in a single winner crossing the thalamic decision threshold first. Later in the trial, sustained activation of the selected pathway created high synaptic eligibility, and dopaminergic feedback altered the eligible synapses, affecting the future dynamics and behavior of the network. Over several trials, the agent learned to select the most rewarding action and could flexibly adapt its decision policy when the optimal action changed, demonstrating the network's ability to use dopaminergic learning for reinforcement learning in simple, relatively unstructured environments. Next, the agent performed a stop signal task, where the subthalamic nucleus received stimulation representing a stop cue during the decision phase. This stop cue simulation affected downstream activity in the internal globus pallidus and thalamus, inhibiting the selection of an action. This demonstrated the realistic go/no-go dynamics of the action channels, consistent with the CBGT network's known role in action facilitation and inhibition [1]. Finally, the n-choice task was revisited with the addition of optogenetic stimulation, which could either excite or inhibit specific nuclei at designated times in the decision process, leading to the facilitation or suppression of action selection. These results further showcased the realistic channel dynamics of the model and demonstrated the utility of the CBGTPy framework for predicting the effects of experimental interventions on CBGT network behavior. In summary, CBGTPy's core strength lies in its ability to test predictions at multiple scales, from synapses to behavior. This framework opens the door to studying circuit dynamics and dopaminergic learning within a broad range of simulated experimental contexts, a necessary prerequisite for the remainder of the project.

4.1.2 Basal Ganglia and Hippocampal Interaction

In Chapter 3, I address my second aim: investigating the computational coordination between reinforcement learning in the basal ganglia and structure learning in the hippocampus. The CBGTPy agent was placed in a complex structured environment featuring navigation over a latent grid [9], and the network was augmented with artificial place cell representations. These artificial place cells, whose activation patterns formed a predictive code derived from successor representation, were trained to process sequences of sensory observations and estimate the current environmental state. These place cell representations were supplied as cortical inputs to the network and were also used to calculate reward prediction errors via a gated mixture-of-experts model. This enabled the network to form a complex action policy over the latent state space. Additionally, these successor representation place cells were compared to various handcrafted place cell representations to understand how the properties of the representations affected network performance.

To generate realistic place cell representations, I adapted the successor representation model from de Cothi and Barry [8], swapping the basis features from spatially selective boundary vector cells to an alternative set of features extracted from the state of a recurrent artificial neural network. This recurrent network, which was provided only non-spatial symbolic sequences of observations, was trained to produce an

internal state representation highly predictive of future events, including reward values. Once transformed by the successor matrix, these basis features created a set of place cells whose activation patterns formed a predictive code emulating the properties of biological place cells. When the cell activities were plotted against the latent squares of the structured task, most place cells formed receptive fields selective to small portions of the state space, qualitatively similar to the receptive fields of biological place cells during spatial navigation. This success supports the notion that the same computational processes could enable the hippocampal-entorhinal cortical system to learn the structure of both spatial and non-spatial environments.

When these successor representation cells were supplied to the CBGTPy network, the agent was unfortunately unable to learn the complex multi-step action policy required to receive the maximal possible reward. An investigation into the corticostriatal weights revealed that the predictive power of each place cell was accurately reflected by the values of the learned weights, indicating that the network's credit assignment mechanism was functional and not responsible for the behavioral deficiencies. When provided with alternative handcrafted place cell representations, the network's performance changed. A place cell representation containing six groups, each selective to an individual trial within a block, led to improved performance. Conversely, using a representation containing forty-nine groups, each selective to an individual latent square, decreased the network's performance. Despite the large quantity of state space information encoded in this second handcrafted representation, the sparsity of the representation led to an inability to leverage the information learned by the synaptic weights to shape downstream action-selection dynamics. Doubling the number of cortical neurons, however, restored strong performance to the network. This indicates that a certain minimum quantity of cortical input is required for the CBGT network to form a robust action policy and reliably select previously rewarded actions.

Revisiting the trial-based representation with six groups, simulations that increased the temporal discounting γ demonstrated an extremely deleterious effect on performance. As γ increased from 0 to 1, the place cells selective for each trial became increasingly responsive during preceding trials, mirroring the predictive coding property of successor representation. This led to increased overlap and decreased orthogonality of the neural representations. At higher values of γ, the network increasingly failed to form an independent action policy for each trial, resulting in incorrect over-generalization across environmental states. Since a high value of γ, representing the contribution of future state transitions to the current state value, is fundamental to the utility of successor representation, a high degree of representational overlap between temporally proximal states is fundamentally unavoidable. In conclusion, it appears that the degree of representational overlap between any particular pair of states must match the appropriate degree of generalization between those states. Biologically realistic place cell representations lead to harmful over-generalizations in tasks where the optimal action varies sharply from step to step. This problem of over-generalization explains the original failure of the joint model to learn a complex action policy and points to the necessity of some additional mechanisms that would enable the basal ganglia to leverage the extremely rich level of structural information present in biological place cell representations. This mechanism could potentially take the form of additional network structure, such as the introduction of hierarchical loops or additional credit-assignment dynamics.

4.2 General Discussion

The natural world is filled with complex and structured environments. To efficiently learn to maximize rewarding outcomes, humans and other mammals must perform

reinforcement learning while leveraging the known structure of the environment. It is not yet fully understood, however, how the brain performs the requisite computations. Successful performance in structured tasks likely requires the cooperation of multiple learning centers across the brain [18], including the basal ganglia and the hippocampus. The cortico-basal ganglia-thalamic (CBGT) circuit, with its evidence accumulation and competitive action-selection dynamics [1] [2], appears well-suited for reinforcement learning guided by dopaminergic reward prediction errors [3]. In contrast, the hippocampal-entorhinal cortical system, in addition to its well-known role in spatial navigation, may be capable of learning the structure of the environment and forming the "mental map" over which the action policy should be learned [4] [5] [6] [7].

The development of the CBGTPy framework represents a significant step forward in modeling the dynamics, plasticity, and behavior of the CBGT circuit, particularly within the context of reinforcement learning tasks. This flexible platform enables researchers to easily subject the network to various experimental manipulations, allowing them to explore how the network dynamics at both micro and macro scales guide the system's computations during the decision process. When applied to the n-choice task, the model demonstrated accurate channel dynamics and effective credit assignment, leading to successful learning and policy-switching in a simple, volatile environment. As the network learns to select the most rewarding action without being supplied any additional information about the environment, this experiment exemplifies "model-free" reinforcement learning [85]. These results demonstrate how the physiology of the CBGT circuit supports the computations required for model-free learning.

Model-free learning, however, is inefficient in structured environments, especially when the structure is known. Instead, model-based learning leverages knowledge of the environmental state space to transfer value information across contexts, enabling the agent to learn how the favorability of an action depends on the current state and relates to the values of other states [85]. While it appears possible that the basal ganglia can learn hierarchical task structures [17] [92], it is unknown how the circuit could learn non-hierarchical structures or process other sources of structural information to perform model-based reinforcement learning. Emerging evidence indicates that the hippocampal-entorhinal cortical system is engaged during reinforcement learning tasks. Therefore, I hypothesized that structural information provided by place cell-like representations could be the missing key enabling the CBGT network to solve more complex tasks. Simulation results, however, indicate that while the hippocampus could serve as an ideal center for structure learning, the direct integration of place cell representations as inputs to the CBGT network is insufficient to produce effective reinforcement learning. The current CBGTPy model performs better when supplied with inputs reflective of a simpler state space, containing a small number of highly orthogonal representations. Additional research is needed to understand how the dynamics of the basal ganglia can process the rich structural information present in biologically realistic place cell representations to produce a complex and highly varied action policy.

Future work on the CBGTPy network could focus on the creation of a hierarchical component, where the competitive dynamics at higher levels of the hierarchy influence the dynamics at lower levels. This would involve the introduction of additional cortical areas and the corresponding pathways, organized into a chain-like structure rather than the current single-loop structure. Conceptual effort would be needed to address the credit assignment problem in such a model, as the existing sustained activation mechanism does not intuitively extend to cortical regions not clearly belonging to a particular action channel. If a hierarchical component were successfully introduced, the network could learn to behave according to conditional rules, introducing a form of hierarchical structure learning directly to the basal ganglia network. It would be

interesting to see if the presence of a hierarchy and the ability to learn conditional rules could enable the network to become more sensitive to smaller changes in inputs, potentially limiting harmful over-generalizations and overcoming the challenges posed by the representational overlaps of place cells. Given the complexity of the CBGTPy model, it might be beneficial to first focus on simpler, more theoretical hierarchical circuit models and use those models to guide development and build predictions for the dynamics of the full network. Ultimately, despite the experimental shortcomings of the current model, I believe that the cooperation of reinforcement learning and structure learning computations across multiple brain regions is the core mechanism by which the brain solves real-world structured tasks. The work presented here represents an important step away from studying the basal ganglia in isolation and towards understanding the nature of these complex multi-region interactions.

Bibliography

1. Dunovan K, Verstynen T. Believer-Skeptic meets actor-critic: Rethinking the role of basal ganglia pathways during decision-making and reinforcement learning. Frontiers in Neuroscience. 2016;10(MAR):1–15. doi:10.3389/fnins.2016.00106.

2. Mulcahy G, Atwood B, Kuznetsov A. Basal ganglia role in learning rewarded actions and executing previously learned choices: Healthy and diseased states. PLoS One. 2020;15(2):e0228081.

3. Watabe-Uchida M, Eshel N, Uchida N. Neural circuitry of reward prediction error. Annu Rev Neurosci. 2017;40(1):373–394.

4. Ballard IC, Wagner AD, McClure SM. Hippocampal pattern separation supports reinforcement learning. Nature Communications. 2019;10(1):1073. doi:10.1038/s41467-019-08998-1.

5. Baram AB, Muller TH, Nili H, Garvert MM, Behrens TEJ. Entorhinal and ventromedial prefrontal cortices abstract and generalize the structure of reinforcement learning problems. Neuron. 2021;109(4):713–723.e7.

6. Lee H. Toward the biological model of the hippocampus as the successor representation agent. Biosystems. 2022;213(104612):104612.

7. Knudsen EB, Wallis JD. Taking stock of value in the orbitofrontal cortex. Nature Reviews Neuroscience. 2022;23(7):428–438. doi:10.1038/s41583-022-00589-2.

8. de Cothi W, Barry C. Neurobiological successor features for spatial navigation. Hippocampus. 2020;30(12):1347–1355.

9. Bond K. Adaptive decision policy dynamics. Carnegie Mellon Univerisity. 2022;.

10. Gershman SJ, Niv Y. Learning latent structure: carving nature at its joints. Curr Opin Neurobiol. 2010;20(2):251–256.

11. Acuna D, Schrater PR. Structure Learning in Human Sequential Decision-Making. In: Koller D, Schuurmans D, Bengio Y, Bottou L, editors. Advances in Neural Information Processing Systems. vol. 21. Curran Associates, Inc.; 2008.Available from: `https://proceedings.neurips.cc/paper_files/paper/2008/file/bf62768ca46b6c3b5bea9515d1a1fc45-Paper.pdf`.

12. Gershman SJ, Pesaran B, Daw ND. Human reinforcement learning subdivides structured action spaces by learning effector-specific values. J Neurosci. 2009;29(43):13524–13531.

13. Mink JW. Basal ganglia mechanisms in action selection, plasticity, and dystonia. Eur J Paediatr Neurol. 2018;22(2):225–229.

14. Foerde K, Shohamy D. The role of the basal ganglia in learning and memory: insight from Parkinson's disease. Neurobiol Learn Mem. 2011;96(4):624–636.

15. Calabresi P, Picconi B, Tozzi A, Ghiglieri V, Di Filippo M. Direct and indirect pathways of basal ganglia: a critical reappraisal. Nat Neurosci. 2014;17(8):1022–1030.

16. Gerfen CR. Segregation of D1 and D2 dopamine receptors in the striatal direct and indirect pathways: An historical perspective. Front Synaptic Neurosci. 2022;14:1002960.

17. Frank MJ, Badre D. Mechanisms of hierarchical reinforcement learning in corticostriatal circuits 1: computational analysis. Cereb Cortex. 2012;22(3):509–526.

18. Caligiore D, Arbib MA, Miall RC, Baldassarre G. The super-learning hypothesis: Integrating learning processes across cortex, cerebellum and basal ganglia. Neurosci Biobehav Rev. 2019;100:19–34.

19. Bariselli S, Fobbs W, Creed M, Kravitz A. A competitive model for striatal action selection. Brain research. 2019;1713:70–79.

20. Bahuguna J, Aertsen A, Kumar A. Existence and control of Go/No-Go decision transition threshold in the striatum. PLoS Comput Biol. 2015;11(4):e1004233.

21. Rubin JE, Vich C, Clapp M, Noneman K, Verstynen T. The credit assignment problem in cortico-basal ganglia-thalamic networks: A review, a problem and a possible solution. European Journal of Neuroscience. 2021;53(7):2234–2253.

22. Schroll H, Hamker FH. Computational models of basal-ganglia pathway functions: focus on functional neuroanatomy. Front Syst Neurosci. 2013;7:122.

23. Stachenfeld KL, Botvinick MM, Gershman SJ. The hippocampus as a predictive map. Nature Neuroscience. 2017;20(11):1643–1653. doi:10.1038/nn.4650.

24. Clapp M, Bahuguna J, Giossi C, Rubin JE, Verstynen T, Vich C. CBGTPy: An extensible cortico-basal ganglia-thalamic framework for modeling biological decision making. Biorxiv. 2024;10.1101 / 2023.09.05.556301v1.

25. Gershman SJ. The successor representation: Its computational logic and neural substrates. J Neurosci. 2018;38(33):7193–7200.

26. Momennejad I, Russek EM, Cheong JH, Botvinick MM, Daw ND, Gershman SJ. The successor representation in human reinforcement learning. Nature Human Behaviour. 2017;1(9):680–692. doi:10.1038/s41562-017-0180-8.

27. Basu J, Siegelbaum SA. The corticohippocampal circuit, synaptic plasticity, and memory. Cold Spring Harb Perspect Biol. 2015;7(11):a021733.

28. Leisman G, Braun-Benjamin O, Melillo R. Cognitive-motor interactions of the basal ganglia in development. Front Syst Neurosci. 2014;8:16.

29. Yin A, Tseng PH, Rajangam S, Lebedev MA, Nicolelis MAL. Place cell-like activity in the primary sensorimotor and premotor cortex during monkey whole-body navigation. Sci Rep. 2018;8(1).

30. Emmi A, Antonini A, Macchi V, Porzionato A, De Caro R. Anatomy and connectivity of the subthalamic nucleus in humans and non-human primates. Front Neuroanat. 2020;14.

31. Chung J, Gulcehre C, Cho K, Bengio Y. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling; 2014. Available from: `https://arxiv.org/abs/1412.3555`.

32. Kriegeskorte N, Douglas PK. Cognitive computational neuroscience. Nature neuroscience. 2018;21(9):1148–1160.

33. Ma WJ, Peters B. A neural network walks into a lab: towards using deep nets as models for human behavior. arXiv preprint arXiv:200502181. 2020;.

34. Guest O, Martin AE. On logical inference over brains, behaviour, and artificial neural networks. Computational Brain & Behavior. 2023; p. 1–15.

35. Bowers JS, Malhotra G, Dujmović M, Montero ML, Tsvetkov C, Biscione V, et al. Deep problems with neural network models of human vision. Behavioral and Brain Sciences. 2022; p. 1–74.

36. Yamins DL, DiCarlo JJ. Using goal-driven deep learning models to understand sensory cortex. Nature neuroscience. 2016;19(3):356–365.

37. Nelson AB, Kreitzer AC. Reassessing models of Basal Ganglia function and dysfunction. Annual review of neuroscience. 2014;37:117–35. doi:10.1146/annurev-neuro-071013-013916.

38. Girard B, Lienard J, Gutierrez CE, Delord B, Doya K. A biologically constrained spiking neural network model of the primate basal ganglia with overlapping pathways exhibits action selection. European Journal of Neuroscience. 2021;53(7):2254–2277.

39. Gurney K, Prescott TJ, Redgrave P. A computational model of action selection in the basal ganglia. I. A new functional anatomy. Biological Cybernetics. 2001;84(6):401–410. doi:10.1007/PL00007984.

40. Vich C, Dunovan K, Verstynen T, Rubin J. Corticostriatal synaptic weight evolution in a two-alternative forced choice task: a computational study. Communications in Nonlinear Science and Numerical Simulation. 2020;82:105048.

41. Nambu A, Chiken S. External segment of the globus pallidus in health and disease: Its interactions with the striatum and subthalamic nucleus. Neurobiology of Disease. 2024;190:106362.

42. Giossi C, Rubin J, Gittis A, Verstynen T, Vich C. Rethinking the external globus pallidus and information flow in cortico-basal ganglia-thalamic circuits. Eur J Neurosci. 2024;doi:https://doi.org/10.1111/ejn.16348.

43. Dunovan K, Vich C, Clapp M, Verstynen T, Rubin J. Reward-driven changes in striatal pathway competition shape evidence evaluation in decision-making. PLOS Computational Biology. 2019;15(5):1–32. doi:10.1371/journal.pcbi.1006998.

44. Vich C, Clapp M, Rubin JE, Verstynen T. Identifying control ensembles for information processing within the cortico-basal ganglia-thalamic circuit. PLOS Computational Biology. 2022;18(6):e1010255. doi:10.1371/journal.pcbi.1010255.

45. Bond K, Rasero J, Madan R, Bahuguna J, Rubin J, Verstynen T. Competing neural representations of choice shape evidence accumulation in humans. Elife. 2023;12:e85223.

46. Smith GD, Cox CL, Sherman SM, Rinzel J. Fourier analysis of sinusoidally driven thalamocortical relay neurons and a minimal integrate-and-fire-or-burst model. J Neurophysiol. 2000;83(1):588–610.

47. Carnevale NT, Hines ML. The NEURON Book. Cambridge University Press; 2006.

48. Goodman DF, Brette R. Brian: a simulator for spiking neural networks in Python. Frontiers in Neuroinformatics. 2008;2. doi:10.3389/neuro.11.005.2008.

49. Dura-Bernal S, Suter BA, Gleeson P, Cantarelli M, Quintana A, Rodriguez F, et al. NetPyNE, a tool for data-driven multiscale modeling of brain circuits. eLife. 2019;8:e44494. doi:10.7554/eLife.44494.

50. Sousa TB. Dataflow programming concept, languages and applications. In: Doctoral Symposium on Informatics Engineering. vol. 130; 2012.

51. The Ray Team. Ray 1.x Architecture; 2020. Available from: `https://docs.ray.io/`.

52. McKinney W. Data Structures for Statistical Computing in Python. In: Stéfan van der Walt, Jarrod Millman, editors. Proceedings of the 9th Python in Science Conference; 2010. p. 56–61.

53. Goenner L, Maith O, Koulouri I, Baladron J, Hamker FH. A spiking model of basal ganglia dynamics in stopping behavior supported by arkypallidal neurons. European Journal of Neuroscience. 2021;53(7):2296–2321. doi:10.1111/ejn.15082.

54. Rothwell P, Hayton S, Sun G, Fuccillo M, Lim B, Malenka R. Input- and Output-Specific Regulation of Serial Order Performance by Corticostriatal Circuits. Neuron. 2015;88(2):345–356. doi:10.1016/j.neuron.2015.09.035.

55. Frank MJ, Samanta J, Moustafa Aa, Sherman SJ. Hold your horses: impulsivity, deep brain stimulation, and medication in parkinsonism. Science (New York, NY). 2007;318(5854):1309–12. doi:10.1126/science.1146157.

56. Zaghloul Ka, Weidemann CT, Lega BC, Jaggi JL, Baltuch GH, Kahana MJ. Neuronal activity in the human subthalamic nucleus encodes decision conflict during action selection. The Journal of neuroscience : the official journal of the Society for Neuroscience. 2012;32(7):2453–60. doi:10.1523/JNEUROSCI.5815-11.2012.

57. Cisek P, Kalaska JF. Neural Correlates of Reaching Decisions in Dorsal Premotor Cortex: Specification of Multiple Direction Choices and Final Selection of Action. Neuron. 2005;45(5):801–814. doi:10.1016/j.neuron.2005.01.027.

58. Nonomura S, Nishizawa K, Sakai Y, Kawaguchi Y, Kato S, Uchigashima M, et al. Monitoring and updating of action selection for goal-directed behavior through the striatal direct and indirect pathways. Neuron. 2018;99(6):1302–1314.

59. Verbruggen F, Aron AR, Band GP, Beste C, Bissett PG, Brockett AT, et al. A consensus guide to capturing the ability to inhibit actions and impulsive behaviors in the stop-signal task. elife. 2019;8:e46323.

60. Nambu A, Tokuno H, Takada M. Functional significance of the cortico–subthalamo–pallidal 'hyperdirect'pathway. Neuroscience research. 2002;43(2):111–117.

61. Schmidt R, Leventhal DK, Mallet N, Chen F, Berke JD. Canceling actions involves a race between basal ganglia pathways. Nature neuroscience. 2013;16(8):1118–1124.

62. Schmidt R, Berke JD. A Pause-then-Cancel model of stopping: evidence from basal ganglia neurophysiology. Philosophical Transactions of the Royal Society B: Biological Sciences. 2017;372(1718):20160202.

63. Mallet N, Schmidt R, Leventhal D, Chen F, Amer N, Boraud T, et al. Arkypallidal cells send a stop signal to striatum. Neuron. 2016;89(2):308–316.

64. Aristieta A, Barresi M, Lindi SA, Barriere G, Courtand G, de la Crompe B, et al. A disynaptic circuit in the globus pallidus controls locomotion inhibition. Current Biology. 2021;31(4):707–721.

65. Chow BY, Han X, Boyden ES. Genetically encoded molecular tools for light-driven silencing of targeted neurons. Progress in Brain Research. 2012;196(type I):49–61. doi:10.1016/B978-0-444-59426-6.00003-3.

66. Humphries MD, Stewart RD, Gurney KN. A physiologically plausible model of action selection and oscillatory activity in the basal ganglia. The Journal of neuroscience : the official journal of the Society for Neuroscience. 2006;26(50):12921–42. doi:10.1523/JNEUROSCI.3486-06.2006.

67. Mandali A, Rengaswamy M, Srinivasa Chakravarthy V, Moustafa AA. A spiking Basal Ganglia model of synchrony, exploration and decision making. Frontiers in Neuroscience. 2015;9(MAY):1–21. doi:10.3389/fnins.2015.00191.

68. Santaniello S, McCarthy MM, Montgomery Jr EB, Gale JT, Kopell N, Sarma SV. Therapeutic mechanisms of high-frequency stimulation in Parkinson's disease and neural restoration via loop-based reinforcement. Proceedings of the National Academy of Sciences. 2015;112(6):E586–E595.

69. Lindahl M, Hellgren Kotaleski J. Untangling Basal Ganglia Network Dynamics and Function: Role of Dopamine Depletion and Inhibition Investigated in a Spiking Network Model. eNeuro. 2016;3(6). doi:10.1523/ENEURO.0156-16.2016.

70. Maith O, Villagrasa Escudero F, Dinkelbach HÜ, Baladron J, Horn A, Irmen F, et al. A computational model-based analysis of basal ganglia pathway changes in Parkinson's disease inferred from resting-state fMRI. European Journal of Neuroscience. 2021;53(7):2278–2295. doi:10.1111/ejn.14868.

71. Chakravarty K, Roy S, Sinha A, Nambu A, Chiken S, Kotaleski JH, et al. Transient Response of Basal Ganglia Network in Healthy and Low-Dopamine State. eNeuro. 2022;9(2). doi:10.1523/ENEURO.0376-21.2022.

72. Frank MJ. Hold your horses: A dynamic computational role for the subthalamic nucleus in decision making. Neural Networks. 2006;19(8):1120–1136. doi:10.1016/j.neunet.2006.03.006.

73. Leblois A, Boraud T, Meissner W, Bergman H, Hansel D. Competition between feedback loops underlies normal and pathological dynamics in the basal ganglia. Journal of Neuroscience. 2006;26(13):3567–3583. doi:10.1523/JNEUROSCI.5050-05.2006.

74. van Albada SJ, Robinson Pa. Mean-field modeling of the basal ganglia-thalamocortical system. I Firing rates in healthy and parkinsonian states. Journal of theoretical biology. 2009;257(4):642–63. doi:10.1016/j.jtbi.2008.12.018.

75. Bogacz R, Larsen T. Integration of reinforcement learning and optimal decision-making theories of the basal ganglia. Neural Computation. 2011;23(4):817–851. doi:10.1162/NECO_a_00103.

76. Guthrie M, Leblois A, Garenne A, Boraud T. Interaction between cognitive and motor cortico-basal ganglia loops during decision making: A computational study. Journal of Neurophysiology. 2013;109(12):3025–3040. doi:10.1152/jn.00026.2013.

77. Nevado-Holgado AJ, Mallet N, Magill PJ, Bogacz R. Effective connectivity of the subthalamic nucleus - globus pallidus network during Parkinsonian oscillations. The Journal of physiology. 2014; p. 1–12. doi:10.1113/jphysiol.2013.259721.

78. Gurney KN, Humphries MD, Redgrave P. A New Framework for Cortico-Striatal Plasticity: Behavioural Theory Meets In Vitro Data at the Reinforcement-Action Interface. PLoS Biology. 2015;13(1):e1002034. doi:10.1371/journal.pbio.1002034.

79. Bekolay T, Bergstra J, Hunsberger E, DeWolf T, Stewart TC, Rasmussen D, et al. Nengo: a Python tool for building large-scale functional brain models. Frontiers in neuroinformatics. 2014;7:48. doi:10.3389/fninf.2013.00048.

80. Oyedotun OK, Olaniyi EO, Khashman A. A simple and practical review of over-fitting in neural network learning. International Journal of Applied Pattern Recognition. 2017;4(4):307–328.

81. Abdolrasol MG, Hussain SS, Ustun TS, Sarker MR, Hannan MA, Mohamed R, et al. Artificial neural networks based optimization techniques: A review. Electronics. 2021;10(21):2689.

82. Carlson KD, Nageswaran JM, Dutt N, Krichmar JL. An efficient automated parameter tuning framework for spiking neural networks. Frontiers in neuroscience. 2014;8:10.

83. Rossant C, Goodman DF, Fontaine B, Platkiewicz J, Magnusson AK, Brette R. Fitting neuron models to spike trains. Frontiers in neuroscience. 2011;5:9.

84. Brockman G, Cheung V, Pettersson L, Schneider J, Schulman J, Tang J, et al. Openai gym. arXiv preprint arXiv:160601540. 2016;.

85. Sutton RS, Barto AG. Reinforcement Learning: An Introduction. 2nd ed. The MIT Press; 2018. Available from: `http://incompleteideas.net/book/the-book-2nd.html`.

86. Masoudnia S, Ebrahimpour R. Mixture of experts: a literature survey. Artificial Intelligence Review. 2014;42(2):275–293. doi:10.1007/s10462-012-9338-y.

87. Kim S, Jung D, Royer S. Place cell maps slowly develop via competitive learning and conjunctive coding in the dentate gyrus. Nature Communications. 2020;11(1):4550. doi:10.1038/s41467-020-18351-6.

88. Ghosh D, Rahme J, Kumar A, Zhang A, Adams RP, Levine S. Why Generalization in RL is Difficult: Epistemic POMDPs and Implicit Partial Observability. CoRR. 2021;abs/2107.06277.

89. Colas JT, Dundon NM, Gerraty RT, Saragosa-Harris NM, Szymula KP, Tanwisuth K, et al. Reinforcement learning with associative or discriminative generalization across states and actions: fMRI at 3 T and 7 T. Human Brain Mapping. 2022;43(15):4750–4790. doi:https://doi.org/10.1002/hbm.25988.

90. Joel D, Niv Y, Ruppin E. Actor-critic models of the basal ganglia: new anatomical and computational perspectives. Neural Netw. 2002;15(4-6):535–547.

91. Simonyan K. Recent advances in understanding the role of the basal ganglia. F1000Res. 2019;8:122.

92. Geddes CE, Li H, Jin X. Optogenetic Editing Reveals the Hierarchical Organization of Learned Action Sequences. Cell. 2018;174(1):32–43.e15. doi:10.1016/j.cell.2018.06.012.

93. Umbach G, Kantak P, Jacobs J, Kahana M, Pfeiffer BE, Sperling M, et al. Time cells in the human hippocampus and entorhinal cortex support episodic memory. Proceedings of the National Academy of Sciences. 2020;117(45):28463–28474. doi:10.1073/pnas.2013250117.

94. Eichenbaum H. Time cells in the hippocampus: a new dimension for mapping memories. Nature Reviews Neuroscience. 2014;15(11):732–744. doi:10.1038/nrn3827.

95. Ma M, Simoes de Souza F, Futia GL, Anderson SR, Riguero J, Tollin D, et al. Sequential activity of CA1 hippocampal cells constitutes a temporal memory map for associative learning in mice. Current Biology. 2024;34(4):841–854.e4. doi:10.1016/j.cub.2024.01.021.

96. Chakravarthy VS, Joseph D, Bapi RS. What do the basal ganglia do? A modeling perspective. Biological cybernetics. 2010;103:237–253.

97. Alexander GE, DeLong MR, Strick PL. Parallel organization of functionally segregated circuits linking basal ganglia and cortex. Annual review of neuroscience. 1986;9(1):357–381.

98. Albin RL, Young AB, Penney JB. The functional anatomy of basal ganglia disorders. Trends in neurosciences. 1989;12(10):366–375.

99. Mink JW. The basal ganglia: Focused selection and inhibition of competing motor programs. Prog Neurobiol. 1996;50(4):381–425.

100. Kravitz AV, Tye LD, Kreitzer AC. Distinct roles for direct and indirect pathway striatal neurons in reinforcement. Nature neuroscience. 2012;15(6):816–818.

101. Mallet N, Micklem BR, Henny P, Brown MT, Williams C, Bolam JP, et al. Dichotomous organization of the external globus pallidus. Neuron. 2012;74(6):1075–1086.

102. Maass W. On the computational power of winner-take-all. Neural computation. 2000;12(11):2519–2535.

103. Hedreen JC, Delong MR. Organization of striatopallidal, striatonigral, and nigrostriatal projections in the macaque. Journal of Comparative Neurology. 1991;304(4):569–595.

104. Gerfen CR, Engber TM, Mahan LC, Susel Z, Chase TN, Monsma Jr FJ, et al. D1 and D2 dopamine receptor-regulated gene expression of striatonigral and striatopallidal neurons. Science. 1990;250(4986):1429–1432.

105. Schultz W. Predictive reward signal of dopamine neurons. Journal of Neurophysiology. 1998;80(1):1–27. doi:10.1152/jn.1998.80.1.1.

106. Frank MJ, Seeberger LC, O'reilly RC. By carrot or by stick: cognitive reinforcement learning in parkinsonism. Science (New York, NY). 2004;306(5703):1940–3. doi:10.1126/science.1102941.

107. Morita K, Kato A. Striatal dopamine ramping may indicate flexible reinforcement learning with forgetting in the cortico-basal ganglia circuits. Frontiers in neural circuits. 2014;8(April):36. doi:10.3389/fncir.2014.00036.

108. Cui G, Jun SB, Jin X, Pham MD, Vogel SS, Lovinger DM, et al. Concurrent activation of striatal direct and indirect pathways during action initiation. Nature. 2013;494(7436):238.

109. Tecuapetla F, Matias S, Dugue GP, Mainen ZF, Costa RM. Balanced activity in basal ganglia projection pathways is critical for contraversive movements. Nature communications. 2014;5:4315.

110. Shin JH, Kim D, Jung MW. Differential coding of reward and movement information in the dorsomedial striatal direct and indirect pathways. Nature communications. 2018;9(1):1–14.

111. Parker JG, Marshall JD, Ahanonu B, Wu YW, Kim TH, Grewe BF, et al. Diametric neural ensemble dynamics in parkinsonian and dyskinetic states. Nature. 2018;557(7704):177–182.

112. Dudman JT, Krakauer JW. The basal ganglia: from motor commands to the control of vigor. Current opinion in neurobiology. 2016;37:158–166.

113. Turner RS, Desmurget M. Basal ganglia contributions to motor control: a vigorous tutor. Current opinion in neurobiology. 2010;20(6):704–716.

114. Rueda-Orozco PE, Robbe D. The striatum multiplexes contextual and kinematic information to constrain motor habits execution. Nature neuroscience. 2015;18(3):453–460.

115. Thura D, Cisek P. The basal ganglia do not select reach targets but control the urgency of commitment. Neuron. 2017;95(5):1160–1170.

116. Yttri EA, Dudman JT. Opponent and bidirectional control of movement velocity in the basal ganglia. Nature. 2016;533(7603):402–406.

117. Hashimoto T, Elder CM, Okun MS, Patrick SK, Vitek JL. Stimulation of the subthalamic nucleus changes the firing pattern of pallidal neurons. Journal of neuroscience. 2003;23(5):1916–1923.

118. Wei W, Rubin JE, Wang XJ. Role of the indirect pathway of the basal ganglia in perceptual decision making. Journal of Neuroscience. 2015;35(9):4052–4064.

119. Gittis AH, Nelson AB, Thwin MT, Palop JJ, Kreitzer AC. Distinct roles of GABAergic interneurons in the regulation of striatal output pathways. Journal of Neuroscience. 2010;30(6):2223–2234.

120. Bevan MD, Booth PA, Eaton SA, Bolam JP. Selective innervation of neostriatal interneurons by a subclass of neuron in the globus pallidus of the rat. Journal of Neuroscience. 1998;18(22):9438–9452.

121. Corbit VL, Whalen TC, Zitelli KT, Crilly SY, Rubin JE, Gittis AH. Pallidostriatal projections promote β oscillations in a dopamine-depleted biophysical network model. Journal of Neuroscience. 2016;36(20):5556–5571.

122. Ketzef M, Silberberg G. Differential synaptic input to external globus pallidus neuronal subpopulations in vivo. Neuron. 2021;109(3):516–529.

123. Fox C, Rafols J. The striatal efferents in the globus pallidus and in the substantia nigra. Research Publications-Association for Research in Nervous and Mental Disease. 1976;55:37–55.

124. Smith Y, Bevan M, Shink E, Bolam JP. Microcircuitry of the direct and indirect pathways of the basal ganglia. Neuroscience. 1998;86(2):353–387.

125. Abdi A, Mallet N, Mohamed FY, Sharott A, Dodson PD, Nakamura KC, et al. Prototypic and arkypallidal neurons in the dopamine-intact external globus pallidus. Journal of Neuroscience. 2015;35(17):6667–6688.

126. Saunders A, Huang KW, Sabatini BL. Globus pallidus externus neurons expressing parvalbumin interconnect the subthalamic nucleus and striatal interneurons. PloS one. 2016;11(2):e0149798.

127. Hernández VM, Hegeman DJ, Cui Q, Kelver DA, Fiske MP, Glajch KE, et al. Parvalbumin+ neurons and Npas1+ neurons are distinct neuron classes in the mouse external globus pallidus. Journal of Neuroscience. 2015;35(34):11830–11847.

128. Dodson PD, Larvin JT, Duffell JM, Garas FN, Doig NM, Kessaris N, et al. Distinct developmental origins manifest in the specialized encoding of movement by adult neurons of the external globus pallidus. Neuron. 2015;86(2):501–513.

129. Fujiyama F, Nakano T, Matsuda W, Furuta T, Udagawa J, Kaneko T. A single-neuron tracing study of arkypallidal and prototypic neurons in healthy rats. Brain Structure and Function. 2016;221:4733–4740.

130. Mastro KJ, Bouchard RS, Holt HA, Gittis AH. Transgenic mouse lines subdivide external segment of the globus pallidus (GPe) neurons and reveal distinct GPe output pathways. Journal of Neuroscience. 2014;34(6):2087–2099.

131. Glajch KE, Kelver DA, Hegeman DJ, Cui Q, Xenias HS, Augustine EC, et al. Npas1+ pallidal neurons target striatal projection neurons. Journal of Neuroscience. 2016;36(20):5472–5488.

132. Steiner LA, Tomás FJB, Planert H, Alle H, Vida I, Geiger JR. Connectivity and dynamics underlying synaptic control of the subthalamic nucleus. Journal of Neuroscience. 2019;39(13):2470–2481.

133. Pamukcu A, Cui Q, Xenias HS, Berceau BL, Augustine EC, Fan I, et al. Parvalbumin+ and Npas1+ pallidal neurons have distinct circuit topology and function. Journal of Neuroscience. 2020;40(41):7855–7876.

134. Lo CC, Wang XJ. Cortico–basal ganglia circuit mechanism for a decision threshold in reaction time tasks. Nature neuroscience. 2006;9(7):956–963.

135. Kumar A, Cardanobile S, Rotter S, Aertsen A. The role of inhibition in generating and controlling Parkinson's disease oscillations in the basal ganglia. Frontiers in systems neuroscience. 2011;5:86.

136. Kimura M. Behavioral modulation of sensory responses of primate putamen neurons. Brain research. 1992;578(1-2):204–214.

137. Aosaki T, Graybiel AM, Kimura M. Effect of the nigrostriatal dopamine system on acquired neural responses in the striatum of behaving monkeys. Science. 1994;265(5170):412–415.

138. Barnes TD, Kubota Y, Hu D, Jin DZ, Graybiel AM. Activity of striatal neurons reflects dynamic encoding and recoding of procedural memories. Nature. 2005;437(7062):1158–1161.

139. Panigrahi B, Martin KA, Li Y, Graves AR, Vollmer A, Olson L, et al. Dopamine is required for the neural representation and control of movement vigor. Cell. 2015;162(6):1418–1430.

140. Pavlides A, Hogan SJ, Bogacz R. Computational models describing possible mechanisms for generation of excessive beta oscillations in Parkinson's disease. PLoS computational biology. 2015;11(12):e1004609.

141. Tachibana Y, Iwamuro H, Kita H, Takada M, Nambu A. Subthalamo-pallidal interactions underlying parkinsonian neuronal oscillations in the primate basal ganglia. European Journal of Neuroscience. 2011;34(9):1470–1484.

142. Nambu A, Tachibana Y. Mechanism of parkinsonian neuronal oscillations in the primate basal ganglia: some considerations based on our recent work. Frontiers in systems neuroscience. 2014;8:74.

143. Pessiglione M, Guehl D, Rolland AS, François C, Hirsch EC, Féger J, et al. Thalamic neuronal activity in dopamine-depleted primates: evidence for a loss of functional segregation within basal ganglia circuits. Journal of Neuroscience. 2005;25(6):1523–1531.

144. de Lafuente V, Jazayeri M, Shadlen MN. Representation of accumulating evidence for a decision in two parietal areas. Journal of Neuroscience. 2015;35(10):4306–4318.

Appendices

This supplemental information accompanies Chapter 2 and is likewise reproduced from the paper "CBGTPy: An extensible cortico-basal ganglia-thalamic framework for modeling biological decision making."

S1 CBGT network

S1.1 Overview of CBGT pathways

While many of the circuit-level details of the basal ganglia (BG) pathways are complex, with new cell types and connections being discovered with increased frequency as our biological tools improve, the consensus conceptualization of the canonical BG circuits has long remained stable [1, 19, 96]. Much of the theoretical work on this topic relates to a profoundly influential framework for representing the passage of top-down signals through the BG, which describes the network as a collection of feed-forward pathways, starting from a cortical input source and flowing to output units that project to certain thalamic nuclei and other subcortical targets [60, 97–100] (Fig 1 in the manuscript). In the *direct pathway*, cortical inputs drive a subpopulation of spiny projection neurons (dSPNs) in the striatum. These dSPNs send inhibitory projections directly to basal ganglia output units, which we refer to as the globus pallidus internal segment (GPi), but which can be tuned to represent other outputs such as the substantia nigra pars reticulata (SNr) to suit a user's interests. As these output units are comprised of GABAergic neurons that are suppressed by inhibition from the dSPNs, the traditional view posits that the direct pathway may act to facilitate action selection by disinhibiting populations downstream from the BG.

The traditional *indirect pathway* starts with cortical inputs to a second striatal subpopulation of spiny projection neurons (iSPNs). Like dSPNs, iSPNs send inhibition to the globus pallidus, specifically its external segment (GPe). Unlike the GPi, however, the GPe is not itself an output unit. GABAergic GPe efferents project to the GPi/SNr, to another region called the subthalamic nucleus (STN), and back to the striatum itself. The GPe feedback signals to the striatum, part of what is called the pallidostriatal circuit, rely both on prototypical (GPeP) and arkypallidal (GPeA) GPe cells [101] and are not considered part of the indirect pathway [64]. The STN relays signals to the GPi/SNr, but via glutamatergic rather than GABAergic synapses. Although complex in its structural organization, the "indirect" pathway framework produces a simple prediction: the net effect of cortical activation of iSPNs will be to inhibit GPe neurons. As a result, GPi/SNr neurons are directly disinhibited, and STN neurons are also disinhibited, which yields a surge in excitation to GPi/SNr that further enhances their firing and hence the suppression of downstream targets.

A key piece for adapting this framework to action selection is the concept of action channels. These channels represent putative parallel pathways, each responsible for whether a specific action (perhaps a specific muscle contraction or limb movement,

perhaps the performance of a more complete action comprising multiple movements) is implemented or suppressed [99]. Thus, the framework conceptualizes the basal ganglia as a collection of independent action channels [99], with evidence accumulation or other processing occurring in parallel across channels, until this competition results in victory for one action, while the others are suppressed (i.e., winner-take-all selection [102]). Outside of these canonical pathways, however, a third *hyperdirect pathway* stands ready to relay excitation from cortex directly to the STN, providing a proposed mechanism for reactive stopping that can abruptly interrupt or block all actions [60].

Beyond action selection, the CBGT pathways are also notable for their critical role in learning. Nigrostriatal pathways send dopaminergic projections directly to the striatal SPNs [103], where the dSPNs and iSPNs predominantly express D1 and D2 dopamine receptors, respectively [104]. Phasic dopamine signals differentially modulate D1 and D2 pathways in response to post-action feedback [78], sculpting corticostriatal synapses over time to effectively promote or inhibit actions in order to maximize future returns. The critical learning signal arises from a discrepancy between a received reward and the expected reward, known as the reward prediction error (RPE), which appears to be encoded in the firing of dopaminergic neurons in the substantia nigra pars compacta (SNc) [105]. Corticostriatal neural plasticity depends on the levels and timing of dopamine signals, which leads to computational models of reinforcement learning that integrate the RPE concept [40, 75, 78, 106]; such a model has even been used to suggest the computational underpinning for the temporal profile of the dopamine signal [107].

Of course, the reality of the CBGT pathways is more complicated than would expected from the simple canonical model. For example, from the onset of a decision process to the execution of an action, dSPN and iSPN activity has been observed to co-vary, contrasting with the idea that the two subpopulations activate at different times as simple "go" or "no-go" switches, respectively [108–111]. In addition, activity of these pathways has been found to impact the kinematics of a movement that is made, rather than or in addition to which action is chosen [112–116]. Continuous stimulation of the STN with deep brain stimulation in individuals with parkinsonism has been shown to fundamentally change BG output [117], impacting impulsiveness without compromising the selection of learned actions. Finally, the idea that hyperdirect pathway activation of the STN acts as a brake that can prevent planned actions via direct activation of the GPi has come into doubt as new cell types (e.g., arkypallidal cells in the GPe) and new connections (e.g., GPe arkypallidal outputs and thalamic projections to the striatum), have been recognized as playing critical roles in stopping [53, 62–64].

S1.2 CBGT model details

The total number of neurons per population is provided in Table S1 Table.

Population	CxI	Cx	$dSPN$	$iSPN$	FSI	GPe_A	GPe_P	GPi	STN	Th
Number of neurons	186	204	75	75	75	190	560	75	750	75

S1 Table. Number of neurons considered in each population. When no distinction between GPe_A and GPe_P is considered, the total number of neurons at GPe is the sum of arkypallidals and prototypicals (750).

As in previous works [43, 44, 118], the activity of each neuron is described by the integrate-and-fire-or-burst model [46], with equations given by

$$\begin{aligned} C\frac{dV}{dt} &= -g_L(V(t)-V_L) - g_T h(t) H(V(t)-V_h)(V(t)-V_T) - I_{syn}(t) + I_{ext}(t), \\ \frac{dh}{dt} &= \begin{cases} -h(t)/\tau_h^- & \text{when } V \geq V_h, \\ -(1-h(t))/\tau_h^+ & \text{when } V < V_h, \end{cases} \end{aligned} \tag{1}$$

where $V(t)$ denotes the activity of the membrane potential at time t. The equation describing the evolution of the membrane potential (dV/dt) contains the leak current, with constant conductance g_L and reversal potential V_L; the low-threshold Ca^{2+} current, with constant conductance g_T, gating variable $h(t)$, and reversal potential V_T; the synaptic current $I_{syn(t)}$; and, finally, the external current $I_{ext}(t)$. Parameter C stands for the capacitance of the membrane potential. The evolution (dh/dt) of the gating variable $h(t)$ changes according to a the relation of V to a constant voltage threshold for burst activation, V_h, where τ_h^+ and τ_h^- represent, respectively, the decay time constant when the membrane potential is below or above V_h . Finally, $H(\cdot)$ represents the Heaviside step function. As with all integrate-and-fire models, a reset condition is added to the model to control the spike generation such that if $V(t)$ crosses a certain threshold value V_{th}, then the membrane potential is reset to ta hyperpolarized membrane potential V_{re}, simulating the spike onset time. That is, if $V(t^-) > V_{th}$, then $V(t^+) = V_{re}$ and a spike has been made by the specific neuron. Parameters of the neuronal model are provided in Table S2 Table.

Population	$C\,(cm^2)$	$g_L\,(mS/cm^2)$	$g_T\,(mS/cm^2)$	$V_L\,(mV)$	$V_h\,(mV)$	$V_T\,(mV)$	$V_{th}\,(mV)$	$V_{re}\,(mV)$	$\tau_h^-\,(ms)$	$\tau_h^+\,(ms)$
CxI	1	1/10	0	−55	−60	120	−50	−55		
Cx	1	1/20	0	−55	−60	120	−50	−55	20	100
$dSPN$	1	1/20	0	−55	−60	120	−50	−55	20	100
$iSPN$	1	1/20	0	−55	−60	120	−50	−55	20	100
FSI	1	1/10	0	−55	−60	120	−50	−55	20	100
$^*GPe$	1	1/20	0.06	−55	−60	120	−50	−55	20	100
GPe_A	1	1/20	0.06	−55	−60	120	−50	−55	20	100
GPe_P	1	1/20	0.06	−55	−60	120	−50	−55	20	100
GPi	1	1/20	0	−55	−60	120	−50	−55	20	100
STN	1	1/20	0.06	−55	−60	120	−50	−55	20	100
Th	1	1/27.78	0	−55	−60	120	−50	−55	20	100

S2 Table. Neuronal parameters. Parameters used in the integrate-and-fire-or-burst model (see equation (1)) where C is the membrane capacitance and coincide with the inverse of the membrane time constant, g_L is the leak conductance, g_T is the low threshold Ca^{2+} maximal conductance, V_L is the leak reversal potential, V_h is the threshold potential for the burst activation, V_T is the low threshold Ca^{2+} reversal potential, τ_h^- is the burst duration, and τ_h^+ is the hyperpolarization duration. * Values in this row are the ones used when no intrinsic separation of neurons is considered.

All neurons in the network communicate through the simulated release of neurotransmitters across synapses. When action potentials arrive at postsynaptic neurons, the activity of each neuron's AMPA, GABA, and NMDA receptors increases according to the synaptic weight and neurotransmitter type. The activation of these receptors induces cellular currents which, in turn, drive future action potentials. The synaptic current $I_{syn}(t)$ is therefore modeled as

$$I_{syn}(t) = g_{AMPA}s_{AMPA}(t)(V(t) - V_E) + \frac{g_{NMDA}s_{NMDA}(t)}{1 + e^{-0.062V(t)/3.57}}(V(t) - V_E) + g_{GABA}s_{GABA}(t)(V(t) - V_I),$$

where g_x, for $x \in \{AMPA, NMDA, GABA\}$, stands for the maximal conductance in each channel; V_E and V_I are the excitatory and inhibitory reversal potentials, respectively; and finally, the variable $s_x(t)$, for $x \in \{AMPA, NMDA, GABA\}$, corresponds to the fraction of open channels of each type. The latter variables evolve according to the differential equations

$$\begin{aligned}
\frac{ds_{AMPA}}{dt} &= \textstyle\sum_j \delta(t - t_j) - \frac{s_{AMPA}}{\tau_{AMPA}}, \\
\frac{ds_{NMDA}}{dt} &= \alpha(1 - s_{NMDA}) \textstyle\sum_j \delta(t - t_j) - \frac{s_{NMDA}}{\tau_{NMDA}}, \\
\frac{ds_{GABA}}{dt} &= \textstyle\sum_j \delta(t - t_j) - \frac{s_{GABA}}{\tau_{GABA}}
\end{aligned}$$

where t_j stands for the $j-$th spike onset time; α is a constant rate; τ_x, for $x \in \{AMPA, NMDA, GABA\}$, is the decay time constant of the corresponding s_x. The term $(1 - s_{NMDA})$ has been designed in order to prevent s_{NMDA} from exceeding the value of 1. Finally, $\delta(\cdot)$ stands for the Dirac delta function.

Individual neurons within the same population are connected to each other with a population-specific probability (p_x) and connection strength (weights, w_x), such that the maximal conductance for a specific receptor x, for $x \in \{AMPA, NMDA, GABA\}$, is given by $g_x = p_x w_x$. Connections between populations similarly are characterized by probabilities and strengths. These connections are depicted with arrows in Fig 1 in the manuscript. The direct (green connections), indirect (blue connections), and pallidostriatal (yellow connections) pathways are present. Depending on the type of receptors, these connections can be inhibitory (arrows ending in a circle) or excitatory (arrows ending in a triangle). Parameters used for the connectivity are provided in Table S3 Table.

All populations have an external current I_{ext} to tune their baseline firing rate, given by

$$I_{ext}(t) = S_{ext,AMPA}(V(t) - V_E) + S_{ext,GABA}(V(t) - V_I)$$

where $S_{ext,x}$ for $x \in \{AMPA, GABA\}$ is a mean-reverting random walk derived from the stochastic differential equation

$$dS_{ext,x} = \frac{(\mu_{ext,x} - S_{ext,x})}{\tau_x} dt + \sigma_{ext,x} \sqrt{\frac{2}{\tau_x}} dW_t.$$

Here, W_t is a Wiener process, τ_x is the time decay of the external current, and $\mu_{ext,x}$ and $\sigma_{ext,x}$ are computed as

$$\mu_{ext,x} = 0.001 E_{ext,x} f_{ext,x} N_{ext,x} \tau_x,$$
$$\sigma_{ext,x} = E_{ext,x} \sqrt{0.0005 f_{ext,x} N_{ext,x} \tau_x}.$$

The parameter $f_{ext,x}$ is the external input frequency, $E_{ext,x}$ is the mean efficacy of the external connections, $N_{ext,x}$ is the number of connections, and τ_x the time decay constant. Values of all of these parameters are specified in Tables S3 Table and S5 Table.

all pathways network				direct/indirect pathways network			
Connected populations	Receptor type	Connection Probability	Connection strength	Connected populations	Receptor type	Connection Probability	Connection strength
$CxI - CxI$	GABA	1	1.075	$CxI - CxI$	GABA	1.0	1.075
$CxI - Cx$	GABA	0.5	1.05	$CxI - Cx$	GABA	0.5	1.05
$Cx - Cx$	AMPA	0.13	0.0127	$Cx - Cx$	AMPA	0.13	0.0127
	NMDA	0.13	0.1		NMDA	0.13	0.08
$Cx - CxI$	AMPA	0.0725	0.113	$Cx - CxI$	AMPA	0.0725	0.113
	NMDA	0.0725	0.525		NMDA	0.0725	0.525
$Cx - dSPN$	AMPA	1	0.022	$Cx - dSPN$	AMPA	1.0	0.015
	NMDA	1	0.03,		NMDA	1.0	0.02
$Cx - iSPN$	AMPA	1	0.022	$Cx - iSPN$	AMPA	1.0	0.015
	NMDA	1	0.028		NMDA	1.0	0.02
$Cx - FSI$	AMPA	1	0.085	$Cx - FSI$	AMPA	1.0	0.19
$Cx - Th$	AMPA	1	0.025	$Cx - Th$	AMPA	1.0	0.025
	NMDA	1	0.029		NMDA	1.0	0.029
$dSPN - dSPN$	GABA	0.45	0.28	$dSPN - dSPN$	GABA	0.45	0.28
$dSPN - iSPN$ [119]	GABA	0.45	0.28	$dSPN - iSPN$	GABA	0.45	0.28
$dSPN - GPi$	GABA	1	1.8	$dSPN - GPi$	GABA	1.0	2.09
$dSPN - GPe_A$ [120–122]	GABA	0.4	0.054				
$iSPN - iSPN$	GABA	0.45	0.28	$iSPN - iSPN$	GABA	0.45	0.28
$iSPN - dSPN$ [119]	GABA	0.5	0.28	$iSPN - dSPN$	GABA	0.5	0.28
$iSPN - GPe_A$ [64, 120–122]	GABA	0.4	0.61	$iSPN - GPe$ [123, 124]	GABA	1.0	4.07
$iSPN - GPe_P$ [64, 120, 122]	GABA	1	4.07				
$FSI - FSI$	GABA	1	2.7	$FSI - FSI$	GABA	1.0	3.25833
$FSI - dSPN$ [119]	GABA	1	1.25	$FSI - dSPN$	GABA	1.0	1.2
$FSI - iSPN$ [119]	GABA	1	1.15	$FSI - iSPN$	GABA	1.0	1.1
$GPe_A - GPe_A$	GABA	0.4	0.15				
$GPe_A - iSPN$ [101, 122, 125–127]	GABA	0.4	0.12				
$GPe_A - dSPN$ [101, 122, 125–127]	GABA	0.4	0.32				
$GPe_A - FSI$ [101, 120–122, 125–127]	GABA	0.4	0.01				
$GPe_P - GPe_P$	GABA	0.4	0.45	$GPe - GPe$	GABA	0.0667	1.75
$GPe_P - GPe_A$ [64, 77, 122, 128, 129]	GABA	0.5	0.3				
$GPe_P - STN$ [120, 125, 126, 130, 131]	GABA	0.1	0.37	$GPe - STN$ [132]	GABA	0.0667	0.35
$GPe_P - GPi$ [120, 125, 126, 130, 131]	GABA	1	0.058	$GPe - GPi$	GABA	1.0	0.058
$GPe_P - FSI$ [101, 121, 125, 126, 131]	GABA	0.4	0.1				
$STN - GPe_P$ [64, 122, 133]	AMPA	0.161666	0.10	$STN - GPe$ [132]	AMPA	0.161666	0.07
	NMDA	0.161666	1.51		NMDA	0.161666	1.51
$STN - GPe_A$ [64, 122, 133]	AMPA	0.161666	0.026				
	NMDA	0.161666	0.075				
$STN - GPi$	AMPA	1	0.0325	$STN - GPi$	AMPA	1.0	0.038
$GPi - Th$	GABA	1	0.3315	$GPi - Th$	GABA	1.0	0.3315
$Th - dSPN$	AMPA	1	0.3285	$Th - dSPN$	AMPA	1.0	0.3825
$Th - iSPN$	AMPA	1	0.3285	$Th - iSPN$	AMPA	1.0	0.3825
$Th - FSI$	AMPA	0.8334	0.1	$Th - FSI$	AMPA	0.8334	0.1
$Th - Cx$	NMDA	0.8334	0.03	$Th - Cx$	AMPA	0.8334	0.03
$Th - CxI$	NMDA	0.8334	0.015	$Th - CxI$	AMPA	0.8334	0.015

S3 Table. CBGT connectivity parameters. Two blocks of 4 columns each are depicted. The first block contains information regarding the network when the 4 different pathways are simulated (see Fig 1 in the manuscript), while the second block contains information regarding the network when only direct/indirect pathways are simulated. Columns in each block describe the parameters used to compute, in each population (1st columns), the maximal conductances g_x, for $x \in \{AMPA, NMDA, GABA\}$ (2nd column), which is the product of the probability of connectivity (3rd column) times the strength of connection (4th column). The rest of parameters are common such that $\tau_{AMPA} = 2\,ms$, $\tau_{NMDA} = 100\,ms$, $\tau_{GABA} = 5\,ms$, $V_E = 0\,mV$, $V_I = -70\,mV$, and $\alpha = 0.6332$. The references corresponding to the weights involved in the arkypallidal pathway are specified in each of the connections. The rest of connections were adjusted to reflect empirical knowledge about local and distal connectivity associated with different populations [43, 118, 134, 135], as well as resting and task-related firing patterns (see Table S4 Table and [44]).

Population	baseline FR range (Hz)	full FR range (Hz)	References
dSPN	$[0, 5]$	$[0, 35]$	[136–139]
iSPN	$[0, 5]$	$[0, 35]$	[136–139]
GPe	$[40, 90]$	$[40, 150]$	[140–142]
GPi	$[40, 90]$	$[40, 150]$	[142]
STN	$[10, 35]$	$[10, 55]$	[140–142]
Th	$[5, 20]$	$[5, 85]$	[143]
Cx		$[0, 100]$	[144]
FSI	$[5, 40]$	$[5, 70]$	[144]

S4 Table. Firing frequency ranges observed in different brain populations. The second column refers to the firing frequency ranges observed experimentally during baseline for each population set in the first column, whereas the third column refers to the ranges observed during decision tasks. In both cases, the ranges reflect experimental data from primates and rats (see references in the last column).

all pathways network					direct/indirect pathways network				
Population	Receptor	External input frequency	External connection efficacy	External connections number	Population	Receptor	External input frequency	External connection efficacy	External connections number
CxI	AMPA	3.7	1.2	640	CxI	AMPA	3.7	1.2	640
Cx	AMPA	2.5	2.0	800	Cx	AMPA	2.3	2.0	800
$dSPN$	AMPA	1.3	4.0	800	$dSPN$	AMPA	1.3	4.0	800
$iSPN$	AMPA	1.3	4.0	800	$iSPN$	AMPA	1.3	4.0	800
FSI	AMPA	4.8	1.55	800	FSI	AMPA	3.6	1.55	800
GPe_A	GABA	2.0	2.0	2000					
	AMPA	2.5	2.0	800					
GPe_P	GABA	2.0	2.0	2000	$^*GPe$	GABA	2.0	2.0	2000
	AMPA	4.0	2.0	800		AMPA	4.0	2.0	800
GPi	AMPA	0.84	5.9	800	GPi	AMPA	0.8	5.9	800
STN	AMPA	4.45	1.65	800	STN	AMPA	4.45	1.65	800
Th	AMPA	2.2	2.5	800	Th	AMPA	2.2	2.5	800

S5 Table. External current parameters. Parameters used to describe the external current (I_{ext}) arriving at the different populations of the CBGT network. From the third column to the last, we specify the different parameters used to describe the external current impinging in each population specified in column 1 and for the specific type of receptors. A non described receptor type means that the parameters are considered to be zero. The time decay constant τ is the same for all populations and only depends on the type of receptor being $\tau = 2\,ms$ if the receptor type is AMPA and $\tau = 5\,ms$ if it is GABA. * Values in this row are the ones used when no intrinsic separation of neurons is considered.

S2 Dopamine-dependent plasticity of corticostriatal weights

Synaptic plasticity in CBGTPy is implemented using a dopamine-dependent plasticity rule, in which the synaptic updates are governed solely by local factors, without requiring individual neurons to access information about the global system state. This rule is an adaptation of the plasticity mechanism presented in [40].

At each corticostriatal AMPA synapse, the model tracks three key values: eligibility $E(t)$, weight $w(t)$, and conductance $g_x(t)$. The conductance is associated with the synaptic current. How much the conductance grows with each pre-synaptic spike is determined by the weight. The weight is the plastic element in the system, which changes over time depending on the time courses of eligibility and dopamine release.

At a computational level, $E(t)$, which represents a synapse's eligibility to undergo weight modification, depends on the relative spike times of the pre- and post-synaptic neurons involved in the synapse. To compute this quantity, we first define the variables $A_{PRE}(t)$ and $A_{POST}(t)$, which serve as instantaneous estimates of the recent levels of pre- and post-synaptic spiking, respectively. Each time a spike occurs in the pre- or post-synaptic cell, these values are increased by a fixed amount (Δ_{PRE} and Δ_{POST}, respectively), and between spikes, they decay exponentially with a time decay constant τ_{PRE} and τ_{POST}, respectively. That is,

$$\frac{dA_{PRE}}{dt} = \frac{1}{\tau_{PRE}} \left(\Delta_{PRE} X_{PRE}\left(t\right) - A_{PRE}(t)\right),$$
$$\frac{dA_{POST}}{dt} = \frac{1}{\tau_{POST}} \left(\Delta_{POST} X_{POST}\left(t\right) - A_{POST}(t)\right)$$

where $X_{PRE}(t)$ and $X_{POST}(t)$ are sums of Dirac delta functions representing the spike trains of the two neurons. That is,

$$X_{PRE} = \sum_{t_s \in \mathcal{X}_{Cx}} \delta(t - t_s), \qquad X_{POST} = \sum_{t_s \in \mathcal{X}_{SPN}} \delta(t - t_s),$$

where t_s is the spike onset, $\mathcal{X}_{Cx}$ is the set of all cortical neurons projecting to the postsynaptic neuron of interest, and $\mathcal{X}_{SPN}$ refers to the identity of that postsynaptic neuron within the striatum.

Eligibility ($E(t)$) changes over time according to

$$\frac{dE}{dt} = \frac{1}{\tau_E} \left(X_{POST}\left(t\right) A_{PRE}(t) - X_{PRE}\left(t\right) A_{POST}(t) - E\right) \tag{2}$$

where τ_E is a time constant. Note that based on equation (2), $E(t)$ tends toward a level that is boosted whenever a post-synaptic spike occurs soon enough after a pre-synaptic spike and is reduced whenever a pre-synaptic spike occurs soon enough after a post-synaptic spike.

The corticostriatal synaptic conductance g_x takes the value of the synaptic weight, $w(t)$, at each pre-synaptic spike time and decays exponentially in-between these spikes:

$$\frac{dg_x}{dt} = \sum_j w(t_j)\delta(t - t_j) - \frac{g_x}{\tau_{AMPA}},$$

where x stands for the specific connection, t_j denotes the time of the $j-$th spike in the cortical presynaptic neuron, $\delta(t)$ is the Dirac delta function, τ_{AMPA} is the decay time constant associated with AMPA synapses, and w itself changes over time based on

dopamine release and the post-synaptic neuron's eligibility. The evolution of w is given by

$$\frac{dw}{dt} = [\alpha_w^j E(t) f(K_{DA})(w_{max}^j - w)]^+ + [\alpha_w^j E(t) f(K_{DA})(w - w_{min}^j)]^-, \tag{3}$$

where the nomenclature $[\cdot]^+$ ($[\cdot]^-$) represents a function whose output is the value inside the brackets if it is positive (negative) and 0 otherwise. The learning rate is denoted in equation (3) by α_w^j, for $j \in \{dSPN, iSPN\}$, depending on to which of the two populations the post-synaptic neuron belongs. This rate has a positive sign for dSPN neurons and a negative one for iSPN neurons to reproduce the observation that positive feedback signals lead to a strengthening of the eligible direct pathway connections and a weakening of the eligible indirect pathway connections. Furthermore, w_{max}^j and w_{min}^j are upper and lower bounds for the weight w, respectively, for $j \in \{dPSN, iSPN\}$.

In equation (3), the variable K_{DA} represents the level of available dopamine in the network, which is computed from the amount of dopamine released through the effect of the differential equation

$$\frac{dK_{DA}}{dt} = C_{scale} \sum_i (DA_{inc}(t_i) - K_{DA})\delta(t_i) - \frac{K_{DA}}{\tau_{DA}},$$

where $DA_{inc}(t_j)$ the increment of dopamine, relative to a baseline level, that is delivered at time t_j. That is, after a specific decision i is made at time t_j, a reward value $r_i(t_j)$ associated to action i is received, which induces a dopamine increment based on the reward prediction error

$$DA_{inc}(t_j) = r_i(t_j) - Q_i(t_j),$$

where $Q_i(t_j)$ is the expected reward for action i at time t_j. This expected reward obeys the update rule

$$Q(t_{j+1}) = Q_i(t_j) + \alpha_Q(r_i(t_j) - Q_i(t_j)),$$

where $\alpha_Q \in [0, 1]$ is the value learning rate. More precisely, note that to account for the motor sensory response, the reward is delivered to the network at the end of *phase 1*, $300\,ms$ after the decision is made (see Fig 3 in the manuscript); Q and DA_{inc} are updated together at this reward delivery time, and the update of DA_{inc} in turn impacts the evolution of K_{DA}. Finally, the function $f(K_{DA})$ in equation (3) represents the impact that the available dopamine K_{DA} has on plasticity, such that, if the target neuron lies in the dSPN population, then

$$f(K_{DA}) = \begin{cases} -\gamma, & \text{if } K_{DA} < -\mu, \\ \dfrac{\gamma}{\mu} K_{DA}, & \text{if } K_{DA} \geq -\mu, \end{cases}$$

while if the target neuron lies in the iSPN population, then

$$f(K_{DA}) = \begin{cases} \varepsilon \dfrac{\gamma}{\mu} K_{DA}, & \text{if } K_{DA} < \mu, \\ \varepsilon\gamma, & \text{if } K_{DA} \geq \mu. \end{cases}$$

for fixed, positive scaling parameters γ, μ. Parameters values used for the plasticity implementation can be found in Table S6 Table and Table S15 Table.

To achieve effective learning, it is critical to address the credit assignment problem of ensuring that the pathways promoting the choice of selected action are the ones that are reinforced by the reward following that action. To achieve this alignment, we introduce a sustained activation signal to the action channel associated with the selected action throughout *phase 1*, based on the patterns of sustained activity that

Parameter	Value
δ_{PRE}	0.8
δ_{POST}	0.04
τ_{PRE}	$15\,ms$
τ_{POST}	$6\,ms$
τ_E	$100\,ms$
α_w^{dSPN}	39.5
α_w^{iSPN}	-38.2
w_{max}^{dSPN}	0.055
w_{max}^{iSPN}	0.035
w_{min}^{dSPN}	0.001
w_{min}^{iSPN}	0.001
ε	0.3
γ	3.0
μ	0.5
C_{scale}	85
τ_{DA}	$2.0\,ms$
α_Q	0.6

S6 Table. Parameters used for the plasticity implementation.

have been observed in motor planning tasks [57]. Specifically, during this phase, the internal gain of cortical stimulation is altered so that the cortical population corresponding to the selected action maintains elevated activity (at 70% of its firing rate from the end of *phase 0*), while cortical populations corresponding to other actions return to baseline firing now that those actions are no longer under consideration. The localized, sustained cortical activation ensures that the downstream striatal neurons in the appropriate action channel have high eligibility [21].

Taken together, the alteration of the direct-indirect pathway balance increases the tendency of the network to select the rewarded action, giving rise to learning. By using a realistic plasticity rule to produce learning, CBGTPy will enable users to investigate the interplay between the dopaminergic system and basal ganglia dynamics in a way that would be impossible with a less physiologically-accurate learning rule.

S3 CGBTpy installation and dependencies

The CBGTPy codebase is written in Python 3.8. If the user is using Python version < 3, e.g. 2.7, some of the dependent libraries may not work. Further details about the installation procedure can be found on our github repository.

S4 List of files

Here we provide the list of the files that are found on our Github repository and that make up the network, including a short reference to what is implemented in each of them. We distinguish between different sets of files: some are common and used regardless of the type of experiment performed. The remainder have separate versions specific to each experiment type, either the n-choice experiment or the stop-signal task, enabling easier swapping between alternative configurations.

The common files are:

- agentmatrixinit.py: builds the CBGT network.
- backend.py: functions for handling pipeline modules, also connects to the Ray server.
- frontendhelpers.py: deals with the environment variable passed.
- generateepochs.py: where rewards and changepoints are defined; rewards are probabilistic and delivered according to which action has been chosen.
- pipeline_creation.py: creates all modules constituting the pipeline.
- plotting_functions.py: implementation of functions useful for data visualization.
- plotting_helper_functions.py: implementation of functions useful for extracting relevant data.
- postprocessing_helpers.py: contains code to extract the data frames for recorded variables.
- qvalues.py: sets up and updates the parameters for the Q-learning algorithm on every trial.
- setup.py: cythonizes the corresponding core simulator code in agent_timestep.pyx.
- tracetype.py: defines wrapper classes that can pair numeric values with metadata.
- generate_opt_dataframe.py: reads in all optogenetic signal-related parameters and generates a data frame.

The files that are used for the simulation of the plasticity experiments are:

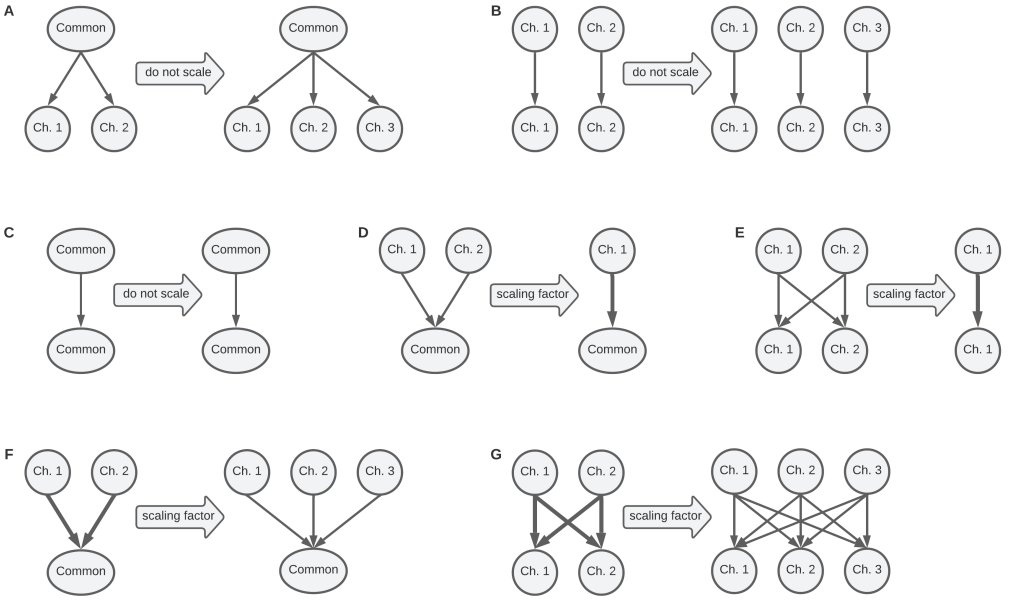
- agent_timestep_nchoice.pyx: contains code for simulating the timesteps of the spiking network.
- init_params_nchoice.py: sets neurons' parameters, receptors' parameters, populations' parameters, dopamine-related parameters for dSPNs and iSPNs, and action channels' parameters with either the defaults or values passed as arguments from the notebook.
- interface_nchoice.py: main simulation controller loop, interacts between environment and the CBGT network.
- popconstruct_nchoice.py: sets up connections between populations and corresponding parameters such as the probability of connection, the mean synaptic efficacy, and the parameters associated with synaptic plasticity (S2).

The files that belong to the stop-signal task experiment are:

- agent_timestep_stopsignal.pyx: contains code for simulating the timesteps of the spiking network.
- generate_stop_dataframe.py: reads in all stop signal-related parameters and generates a data frame.
- init_params_stopsignal.py: sets neurons' parameters, receptors' parameters, populations' parameters, dopamine-related parameters for dSPNs and iSPNs, and action channels' parameters with either the defaults or values passed as arguments from the notebook; this version differs from the one used to perform the plasticity experiment since different populations are considered for the simulation of the two experiments.
- interface_stopsignal.py: main simulation controller loop, interacts between environment and the CBGT network.
- popconstruct_stopsignal.py: sets connections between populations and corresponding parameters such as the probability of connection, the mean synaptic efficacy, and the parameters associated with synaptic plasticity.

S5 Network scaling

The CBGTPy model allows for the simulation of networks with an arbitrary number of action channels, with a default setting of 2 channels. As the number of channels is varied, certain pathways are automatically adjusted to ensure that the overall quantity of synaptic input to each subpopulation remains constant, allowing the neurons to maintain their proper baseline firing rates. To determine which pathways require scaling, the connectivity pattern of each pathway is compared to a set of cases, which are outlined in Figure S1 Fig. If, for a given pathway, each target subpopulation only receives input from a single channel or from a shared source, no scaling factor is applied. If, however, each target subpopulation receives input from all action channels, then that pathway requires a scaling factor. This factor is calculated as $2/n$, where n is the new number of action channels. When $n > 2$, the scaling factor is used to reduce the connection probability so that the expected number of afferent synapses per neuron remains constant. As a special case, when $n = 1$ and the scaling factor is 2, the weights of the synapses are increased rather than the connection probability, to avoid potentially setting the pathway's connection probability over 100%. For a detailed listing of connections to which a scaling factor is applied, see Table S7 Table.



S1 Fig. Overview of scaling rules. Pathways featuring solely divergent (**A**) or parallel (**B**,**C**) connectivity never have a scaling factor applied. As the number of incoming connections to each target subpopulation remains constant, no scaling of the pathway parameters is needed. When the number of action channels is reduced from 2 to 1, pathways defined by convergent (**D**) or all-to-all (**E**) connectivity have their synaptic weights scaled up by a factor of 2. When the number of action channels is increased above 2, convergent (**F**) and all-to-all (**G**) pathways have their synaptic connection probabilities decreased via the scaling factor.

all pathways network			direct/indirect pathways network		
Connected populations	Receptor type	Scaling rule applied?	Connected populations	Receptor type	Scaling rule applied?
$CxI - CxI$	GABA	no	$CxI - CxI$	GABA	no
$CxI - Cx$	GABA	no	$CxI - Cx$	GABA	no
$Cx - Cx$	AMPA	no	$Cx - Cx$	AMPA	no
	NMDA	no		NMDA	no
$Cx - CxI$	AMPA	yes	$Cx - CxI$	AMPA	yes
	NMDA	yes		NMDA	yes
$Cx - dSPN$	AMPA	no	$Cx - dSPN$	AMPA	no
	NMDA	no		NMDA	no
$Cx - iSPN$	AMPA	no	$Cx - iSPN$	AMPA	no
	NMDA	no		NMDA	no
$Cx - FSI$	AMPA	yes	$Cx - FSI$	AMPA	yes
$Cx - Th$	AMPA	no	$Cx - Th$	AMPA	no
	NMDA	no		NMDA	no
$dSPN - dSPN$	GABA	no	$dSPN - dSPN$	GABA	no
$dSPN - iSPN$	GABA	no	$dSPN - iSPN$	GABA	no
$dSPN - GPi$	GABA	no	$dSPN - GPi$	GABA	no
$dSPN - GPe_A$	GABA	no			
$iSPN - iSPN$	GABA	no	$iSPN - iSPN$	GABA	no
$iSPN - dSPN$	GABA	no	$iSPN - dSPN$	GABA	no
$iSPN - GPe_A$	GABA	no	$iSPN - GPe$	GABA	no
$iSPN - GPe_P$	GABA	no			
$FSI - FSI$	GABA	no	$FSI - FSI$	GABA	no
$FSI - dSPN$	GABA	no	$FSI - dSPN$	GABA	no
$FSI - iSPN$	GABA	no	$FSI - iSPN$	GABA	no
$GPe_A - GPe_A$	GABA	yes			
$GPe_A - iSPN$	GABA	no			
$GPe_A - dSPN$	GABA	no			
$GPe_A - FSI$	GABA	yes			
$GPe_P - GPe_P$	GABA	yes	$GPe - GPe$	GABA	yes
$GPe_P - GPe_A$	GABA	no			
$GPe_P - FSI$	GABA	yes			
$GPe_P - STN$	GABA	no	$GPe - STN$	GABA	no
$GPe_P - GPi$	GABA	no	$GPe - GPi$	GABA	no
$GPe_P - FSI$	GABA	no			
$STN - GPe_P$	AMPA	no	$STN - GPe$	AMPA	no
	NMDA	no		NMDA	no
$STN - GPe_A$	AMPA	no			
	NMDA	no			
$STN - GPi$	AMPA	yes	$STN - GPi$	AMPA	yes
$GPi - Th$	GABA	no	$GPi - Th$	GABA	no
$Th - dSPN$	AMPA	no	$Th - dSPN$	AMPA	no
$Th - iSPN$	AMPA	no	$Th - iSPN$	AMPA	no
$Th - FSI$	AMPA	yes	$Th - FSI$	AMPA	yes
$Th - Cx$	AMPA	yes	$Th - Cx$	AMPA	yes
$Th - CxI$	AMPA	yes	$Th - CxI$	AMPA	yes

S7 Table. Scaling rule application per connection. Two blocks of 2 columns each are depicted. The first block applies to the full network containing all pathways, while the second block applies to the reduced network containing only the direct/indirect pathways.

S6 Supporting tables

Feature	Table of reference parameters in the manuscript	Specifications	Possibility of modification by the user
Number of neurons considered in each population	S1 Appendix: Table S1_1		Yes
Neural parameters	S1 Appendix: Table S1_2 Suppl. Tables: S2 Table, S3 Table, S4 Table, S5 Table		Yes
CBGT connectivity parameters	S1 Appendix: Table S1_3	Receptors type Conn. probability Conn. strength Conn. presence	No Yes Yes Yes
External current parameters	S1 Appendix: Table S1_5	Receptor Frequency Conn. efficacy Conn. number	No Yes Yes No
Parameters used for the plasticity implementation	S2 Appendix: Table S2_1 Suppl. Tables: S6 Table, S7 Table, S8 Table		Yes
Parameters used for the stop signal implementations	Suppl. Tables: S9 Table		Yes
Parameters used for the optogenetic implementations	Suppl. Tables: S10 Table		Yes
Scaling rule application per connection.	S5 Appendix: Table S5_1		No

S8 Table. Relation of all parameters editable by the user. Here we list all those features that the user can modify and those that cannot. If so, we indicate in which table of the Supplementary information the specific parameters are described.

Parameter	Definition
N	Number of receptors of the neuron
C	Capacitance in nF
$Taum$	Membrane time constant in ms
$RestPot$	Neuron resting potential in mV
$ResetPot$	Neuron reset potential in mV
$Threshold$	Neuron reset potential in mV
$RestPot\_ca$	Resting potential for calcium ions
$Alpha\_ca$	Amount of increment of [Ca] with each spike discharge
Tau_ca	Time constant of Ca-related conductance
Eff_ca	Calcium efficacy
$tauhm$	Duration of the burst in ms
$tauhp$	Duration of hyperpolarization necessary to recruit a maximal post-inhibitory rebound response in ms
$V\_$	Threshold for bursts activation in mV
V_T	Low-threshold of Ca reversal potential in mV
g_T	Low-threshold of Ca maximal conductance in mS/cm^2
g_adr_max	Maximum value of the conductance
$Vadr\_h$	Potential for g_adr_max
$Vadr\_s$	Slop of g_adr at $Vadr\_h$, defining how sharp the shape of g_ard is
$ADRRevPot$	Reverse potential for ADR
g_k_max	Maximum outward rectifying current
Vk_h	potential for g_k_max
Vk_s	Defines how sharp the shape of g_k is
tau_k_max	Maximum time constant for outward rectifying K current
n_k	Gating variable for outward rectifying K channel
h	Gating variable for the low-threshold Ca current

S9 Table. Neuronal parameters editable by the user. These parameters can be modified through the data frame `params`.

Parameter	Definition
N	Population-specific number of neurons in the nuclei
C	Capacitance in nF
$Taum$	Membrane time constant in ms
g_T	Ca low-threshold maximal conductance in mS/cm^2

S10 Table. Population-specific neuron parameters changeable by the user. These parameters can be modifiable through the dictionary `pops`, addressing the population of interest.

Parameter	Definition
Tau_AMPA	AMPA time constant in ms
$RevPot\_AMPA$	AMPA reversal potential in mV
Tau_GABA	GABA time constant in ms
$RevPot\_GABA$	GABA reversal potential in mV
Tau_NMDA	NMDA time constant in ms
$RevPot\_NMDA$	NMDA reversal potential in mV
$RevPot\_ChR2$	Channelrhodopsin-2 reversal potential in mV
$RevPot\_NpHR$	Halorhodopsin reversal potential in mV

S11 Table. Synaptic and channel parameters changeable by the user. These parameters can be modified through the data frame `receps`.

Parameter	Definition
$FreqExt\_AMPA$	Baseline input firing rate to AMPA receptors
$MeanExtEff\_AMPA$	AMPA conductance
$MeanExtCon\_AMPA$	average of AMPA connections
$FreqExt\_GABA$	input firing to GABA receptors
$TMeanExtEff\_GABA$	GABA conductance
$MeanExtCon\_GABA$	average of GABA connections

S12 Table. Population-specific baseline parameters modifiable by the user. These parameters can be modified through the dictionary `base`, addressing the population of interest.

Parameter	Definition
dpmn_DOP	Time constant of the dopamine trace
dpmn_DAt	Tonic dopamine
dpmn_dPRE	Fixed increment for pre-synaptic spiking (Apre)
dpmn_dPOST	fixed increment for post-synaptic spiking (Apost)
dpmn_tauE	Eligibility trace decay time constant
dpmn_tauPRE	Decay time constant for the pre-synaptic spiking trace (Apre)
dpmn_tauPOST	Decay time constant for the post-synaptic spiking trace (Apost)
dpmn_m	Motivation, that modulates the strength of the dopamine level
dpmn_E	Eligibility trace
dpmn_DAp	Phasic dopamine
dpmn_APRE	Pre-synaptic spiking trace
dpmn_APOST	Post-synaptic spiking trace
dpmn_XPRE	Pre-synaptic spike time indicators
dpmn_XPOST	Post-synaptic spike time indicators
dpmn_fDA_D1	f(DA) value for D1-SPNs
dpmn_fDA_D2	f(DA) value for D2-SPNs
dpmn_x_FDA	threshold for f(DA) function
dpmn_y_FDA	threshold for f(DA) function
dpmn_d2_DA_eps	Scaling factor for dopamine levels of D2-SPNs as compared to D1-SPNs

S13 Table. Dopamine-related parameters editable by the user. These parameters can be modified through the data frame `dpmns`.

Parameter	Value
dpmn_type	1 for dopamine-related variables of dSPNs or 2 for iSPN neurons
dpmn_alphaw	weight increment proportional to the dopamine discharge
dpmn_wmax	upper bound for W

S14 Table. Dopamine-related parameters for corticostriatal projections to dSPN and iSPN neurons. Each of the striatal SPN population, dSPN and iSPN, maintain a copy of this data structure which can be independently modified through the data frames `dSPN_params` and `iSPN_params` defined in the `configuration` dictionary in the notebooks.

Parameter	Value
δ_{PRE}	0.8
δ_{POST}	0.04
τ_{PRE}	$15\,ms$
τ_{POST}	$6\,ms$
τ_E	$100\,ms$
α_w^{dSPN}	39.5
α_w^{iSPN}	-38.2
w_{max}^{dSPN}	0.055
w_{max}^{iSPN}	0.035
w_{min}^{dSPN}	0.001
w_{min}^{iSPN}	0.001
ε	0.3
δ	3.0
μ	0.5
C_{scale}	85
τ_{DA}	$2.0\,ms$
α_Q	0.6

S15 Table. Parameters used for plasticity implementation. For more details about the plasticity parameters, please refer to S2 Appendix. The parameters without a subscript can be modified using data frame `dpmns`, whereas the parameters with a subscript `dSPN` or `iSPN` can be modified through the data frames `dSPN_params` and `iSPN_params` respectively.

Parameter	Description	Example
Stop signal present	List of boolean variables	[True, True]
Stop signal probability	Proportional of trials to be randomly selected or list of trial numbers per nuclei	[1., 1.]
Stop signal amplitude	Excitatory conductance	[0.4, 0.4]
Stop signal onset	Onset time in ms	[70., 70.]
Stop signal duration	Duration time in ms or phase of the simulation	[145., 145.]
Stop signal channel	List of channels ("all" or channel name)	["all","all"]
Stop signal population	List of nuclei	["STN","GPeA"]

S16 Table. Parameters that can be set for stop signal stimulation. The example values included in the table describe the parameters used to generate Figure 7.

Parameter	Description	Example
Optogenetic signal present	List of boolean variables	[True, True]
Optogenetic signal probability	Proportional of trials to be randomly selected or list of trial numbers per nuclei	[[0],[1]]
Optogenetic signal amplitude	Excitatory or inhibitory conductance	[0.5, -0.5]
Optogenetic signal onset	Onset time in ms	[10., 10.]
Optogenetic signal duration	Duration time in ms or phase of the simulation	[“phase 0”, 400.]
Optogenetic signal channel	List of channels (“all” or channel name)	[“all”,“all”]
Optogenetic signal population	List of nuclei	[“iSPN”,“dSPN”]

S17 Table. Parameters that can be set for optogenetic stimulation. The example values included in the table describe the parameters used to generate Figure 8.

S7 Supporting figures

```
In [12]: # List all the agent variables accessible
         results[0].keys()
```

```
Out[12]: dict_keys(['experimentchoice', 'params', 'pops', 'receps', 'base', 'dpmns', 'd1', 'd2', 'channels', 'newpathways',
         'Q_support_params', 'Q_df_set', 'n_trials', 'volatility', 'conflict', 'reward_mu', 'reward_std', 'maxstim', 'recor
         d_variables', 'opt_signal_present', 'opt_signal_probability', 'opt_signal_amplitude', 'opt_signal_onset', 'opt_sig
         nal_duration', 'opt_signal_channel', 'opt_signal_population', 'sustainedfraction', 'actionchannels', 'volatile_pat
         tern', 'cp_idx', 'cp_indicator', 'noisy_pattern', 't_epochs', 'block', 'trial_num', 'chosen_action', 'celldefault
         s', 'popspecific', 'receptordefaults', 'basestim', 'dpmndefaults', 'd1defaults', 'd2defaults', 'popdata', 'pathway
         s', 'opt_df', 'opt_channels_df', 'opt_amplitude_df', 'opt_onset_df', 'opt_populations_df', 'opt_list_trials', 'con
         nectivity_AMPA', 'meaneff_AMPA', 'plastic_AMPA', 'connectivity_GABA', 'meaneff_GABA', 'plastic_GABA', 'connectivit
         y_NMDA', 'meaneff_NMDA', 'plastic_NMDA', 'Q_df', 'AMPA_con', 'AMPA_eff', 'GABA_con', 'GABA_eff', 'NMDA_con', 'NMDA
         _eff', 'agent', 'datatables', 'reward_val', 'popfreqs'])
```

S2 Fig. List of parameters and data frames that are returned from the simulation.

```
In [20]: results[0]['popfreqs']
```

Out[20]:

	GPi_left	GPi_right	STNE_left	STNE_right	GPeP_left	GPeP_right	D1STR_left	D1STR_right	D2STR_left	D2STR_right	Cx_left	Cx_right	Th_lef
0	64.000000	69.555556	25.511111	25.644444	61.755556	58.711111	5.333333	4.222222	4.666667	5.555556	0.000000	0.000000	7.77777
1	63.777778	68.444444	25.377778	25.711111	61.733333	58.911111	5.111111	4.444444	4.666667	5.333333	0.000000	0.000000	7.77777
2	64.000000	68.444444	25.511111	25.422222	61.488889	58.866667	5.111111	4.444444	4.666667	5.777778	0.000000	0.000000	7.77777
3	64.666667	67.555556	25.333333	25.488889	61.622222	58.266667	5.111111	4.444444	4.666667	5.777778	0.000000	0.000000	7.33333
4	65.111111	67.555556	25.200000	25.355556	61.800000	58.088889	5.111111	4.444444	4.666667	5.777778	0.000000	0.000000	7.33333
...	...	...	...	...	...	...	...	...	...	...	...	...	...
4026	64.222222	66.444444	27.133333	24.555556	56.711111	65.000000	5.555556	4.000000	5.555556	3.333333	0.735294	1.062092	10.00000
4027	63.777778	65.777778	27.400000	24.177778	56.555556	65.355556	5.777778	4.000000	5.333333	3.333333	0.735294	1.143791	9.77777
4028	64.222222	65.777778	27.355556	24.155556	56.977778	65.266667	5.777778	3.777778	4.888889	3.333333	0.735294	1.143791	9.11111
4029	65.111111	65.777778	27.377778	23.911111	57.422222	64.977778	5.777778	3.777778	4.888889	3.555556	0.735294	1.143791	9.11111
4030	64.444444	65.777778	27.600000	23.888889	57.977778	64.311111	5.777778	3.777778	4.888889	4.000000	0.735294	1.143791	9.33333

4031 rows × 17 columns

S3 Fig. Example of `results['popfreqs']` data frame.

```
In [28]: datatables[0]
```

Out[28]:

	decision	stimulusstarttime	decisiontime	decisionduration	decisiondurationplusdelay	rewardtime	correctdecision	reward
0	right	0	247	247	501	501	left	0.0
1	left	1102	1323	221	473	1575	right	0.0
2	none	2176	3177	1001	1253	3429	left	0.0

S4 Fig. Example of `datatables[0]` data frame.

```
In [29]: # Check the Q-values data frame
         results[0]['Q_df']
```

Out[29]:

	left	right
0	0.50	0.50
0	0.50	0.45
0	0.45	0.45
0	0.45	0.45

S5 Fig. Example of `Q_df` data frame.