DEPARTMENT OF COMPUTER SCIENCE

Self-Supervised Deep Learning and Neocortex

Aikaterini Vasilonikolidaki

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Bachelor of Science in the Faculty of Engineering.

Thursday 12th May, 2022

Abstract

The exploration exploitation dilemma is one of the most prominent open research problems in Reinforcement Learning (RL). Pathak et al [67] have proposed a solution that introduces balance between those two components. The solution involves an agent, that is the RL algorithm , producing its own intrinsic rewards, through a bolt-on module, namely Intrinsic Curiosity Module(ICM). Using intrinsic rewards the algorithm is guided towards more exploratory behaviour but still not loosing sight of its ultimate goal of maximising its learning progress, which is quantified by the extrinsic rewards it has accumulated by the end of the training. The first aim of this project is to implement this solution in environments with dense discrete extrinsic rewards supplied to the agent and importantly modify the implementation so that it works in Atari environments. The results of the implementation align with the one;s of the original creator of the ICM, as it is deduced that intrinsic curiosity boosts the learning progress of the agent. Moreover, as it has been suggested by prior research a form of intrinsic reward is being produced in the brain, which is in this research hypothesis it is proposed that it is processed in the same way as the ICM. The ICM is then paralleled with the functionality of the neocortex which is believed to be the main component of the brain that drives learning and is responsible for functionalities that distinguish humans from other mammals. The results of this implementation are in a promising direction as it is illustrated that the more biologically plausible model does yield even better results than the original ICM. However, this must be subject of further research as the reasons underlying the boost in learning performance of the more biologically plausible ICM compared to the original ICM remain in debate.

- I spent at least 240 hours collecting material about neuroscience, how the brain learns and the neocortical microcircuit.
- I developed a version of the algorithm proposed by Pathak and colleagues [67], see page $14-18$, by using Phil Tabor's implementation as a base which implemented the ICM without feature encoding using just a 4-D vector space as an input.
- Adjusted the version so that the agent can play and learn in the Atari Environment, receiving images as an input.
- I wrote around $100-150$ lines of code. It must be declared that the code for processing the frames was adapted by different online sources. The code for the encoders was 100% my creation, guided by the paper of the original implementation [67]
- I mapped the functionality of the neocortical microcircuit to the ICM, while illustrating it with schematics.
- Separated encoders in the original implementation to create a more biologically plausible ICM.

Dedication and Acknowledgements

I would like to thank my supervisor Rui Ponte Costa along with the the Bristol Neural Machine Learning Group for the important help they have provided me throughout my dissertation. I would not have been able to write this dissertation without the continuous support of my parents and friends which have been at my side throughout the whole time which I am grateful for.

Declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Taught Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, this work is my own work. Work done in collaboration with, or with the assistance of others, is indicated as such. I have identified all material in this dissertation which is not my own work through appropriate referencing and acknowledgement. Where I have quoted or otherwise incorporated material which is the work of others, I have included the source in the references. Any views expressed in the dissertation, other than referenced material, are those of the author.

Aikaterini Vasilonikolidaki, Thursday 12th May, 2022

Contents

List of Figures

List of Tables

Ethics Statement

This project did not require ethical review, as determined by my supervisor, Rui Ponte Costa.

Supporting Technologies

Software technologies used for this project:

- I used `Python` as a programming language to implement the ICM.
- I used `PyTorch` [2] Python's Machine Learning Framework to implement the ICM, this include using convolutional neural networks for encoding features of incoming images and neural networks for making predictions.
- I used a parts of the `OpenCV` [3] computer vision library to process incoming raw input images so that computations are more efficient.
- I used the `Open-AI Gym` [62] open source toolkit containing a range of environments where the agent can learn, but also allowing to directly modify the open-AI Gym environment by using Gym wrappers.

Chapter 1

Introduction

Combining reinforcement learning (RL) with artificial neural networks has resulted in creating effective algorithms that learn and act in uncertain environments by creating a model of that environment [85]. More specifically, RL mimics how a human would learn from interactions with their environment by exercising their direct sensorimotor connections and creating a cause-and-effect relationship with consequences of actions. Traditionally, this is achieved by giving a feedback signal to the RL algorithm in the form of extrinsic rewards, such as the score of an Atari game, when achieving target tasks, such as hitting a block in an Atari Breakout game. The RL algorithm will operate in order to maximise these rewards and its learning will be geared towards that purpose, by associating its actions with the feedback. However, while aiming to maximise the extrinsic rewards, often a loop will be created where the RL will not be able outclass a certain threshold of cumulative extrinsic rewards, as it will repeat actions that have previously achieved 'favourable' feedback and not trying new combinations that could possibly provide even higher reward. In this thesis, this problem is overcame by introducing an extra reward, intrinsic to the RL algorithm, therefore produced by the algorithm itself through a module namely Intrinsic Curiosity Module(ICM), as proposed by [67]. This reward will guide exploration by encouraging the algorithm to engage with novel parts of the environment it is learning in by trying to perform diverse actions. The ICM model incorporates Reinforcement Learning (RL) with Deep Neural Networks, Convolutional Neural Networks and self-supervised learning while the CartPole and Atari Breakout OpenAI Gym environments are used to examine the learning process of the agent.

This exploration approach was inspired by humans (agents) which from an early age appear to have an innate drive to explore and learn, which literature has attributed to intrinsic motivation. It was suggested that intrinsic curiosity is implemented in the human brain by the neocortex(a brain structure described in 2.2.3) and is used to drive learning, in the absence of dense extrinsic rewards. The notion that the sensory neocortex creates and constantly updates an internal model of its environment by using sensory epithelia is used as a foundation, supported by plethora of prior research. Following that, in the thesis it is supported that this mental model is updated by constantly comparing self-generating predictions about the upcoming sensory input and the actual sensory epithelia, as a considerable body of research has previously suggested. Prediction errors cause the mental models to adapt by strengthening and weakening connections between neurons so as to reduce discrepancy. Models created from sensory input are also linked with the human's actions through paths with basal ganglia, an assembly of (subcortical) neural structures that are responsible for motion (described in 2.2.1). In this thesis those two research areas of intrinsic curiosity in reinforcement learning and intrinsic neuroscience are interconnected by suggesting that the ICM works in a similar way to the neocortical microcircuit. Importantly, the ICM is adjusted so that it portrays a more biologically plausible model and results on performance are examined. Such project will provide the opportunity to understand better the functions of the canonical microcircuit through the ICM, while also open the path to further research on the implementation of intrinsic curiosity in RL, by creating more biologically plausible algorithms that could possibly improve performance.

However, it is important to mention that the pursuit of understanding the functions of the neocortex is ongoing and no definitive answer has been given up to this date. A major challenge included comparing a multitude of hypothesis existing about the computations of the neocortex. Additionally, a challenge was faced in RL experiments, as RL algorithms require high computational resources and time.

Chapter 2

Background

2.1 Neuroscience Basics

2.1.1 Neurons

To model and explain the computation of the canonical microcircuit we must first look into the structure and computations of a single neuron. **Neurons** are cells which comprise the main cellular component in the human brain's circuitry and nervous system. Neurons have highly specialised shapes which consists, typically, of a cell body, a single axon and dendrites, allowing them to process and transmit information, from the observed environmental conditions. Having multiple spatial compartments allows more complex computations to take rise, when needed. **Dendrites** are tree shaped structures with multiple branches, that branch from the cell body to receive inputs from other neurons. The inputs have the form of electrical and chemical signals and transmit information from the environment. The **soma** contains the cell nucleus -or DNA- and from a computational point of view is where all the chemical and electrical signals from An **axon** is a fiber carrying the output signal from the neuron to other target neurons. The output signal has the form of a sequence of electrical pulses, caused by the rapid change in current across the cellular membrane and is called the **action potential** or a **spike**. A neuron will fire if it generates a spike, which means that the current is sufficient to initiate a voltage response in the neuron's cell membrane. The axon can branch widely, therefore copies of the output action potential of a neuron with a wide branch structure will reach and influence many target neurons.

2.1.2 Synapses, Excitatory and Inhibitory Neurons

Neurons are not connected physically. The connections between neurons are referred to as synapses and they convert action potential from the pre-synaptic neuron's axon into the postsynaptic potential in the dendrite of another. The small physical space that separate the pre-synaptic and post-synaptic neuron is the synaptic cleft. A synapse, on the pre-synaptic side of the neuron, at the arrival of an action potential, will release a chemical neurotransmitter, into the synaptic cleft. The post-synaptic neuron has ion channel receptors, the neurotransmitter will bind into those receptors and open up the ion channels. When ions from the chemical neurotransmitter flow into those ion channels they will create a current that will excite or inhibit the post-synaptic neuron's spike-generating activity, according to the type of the neurotransmitter chemical. The type of chemical neurotransmitter released from the pre-synaptic neuron will either increase or decrease the probability of an action potential -or spike- occurring on the post-synaptic neuron. A pre-synaptic neuron will be excitatory if it releases neurotransmitters that will excite the post-synaptic neuron, therefore increase its probability to spike or inhibitory when it inhibits the post-synaptic neuron and decreases its probability to fire. If the post-synaptic neuron is excited enough the neuron will emit an action potential-spike- that will be sent from the neuron's axons to the dendrite of another neuron which will lead to post-synaptic potential again. The effect and strength of the action potentials will be determined by characteristics of excitatory or inhibitory operating neuron, for example the sensitivity of the synapse or the specificity of the neurotransmitters(change). This explanation of excitatory or inhibitory neurotransmitters is over-simplified but is sufficient for the purposes of this dissertation.

2.1.3 How is learning driven in the brain?

According to the theory of Donald O. Hebb [37] an increase in synaptic efficacy arises from a pre-synaptic cell's repeated and persistent stimulation of a post-synaptic cell. The adaptation of neural strengths during the learning process is known as synaptic plasticity and this theory is an attempt to explain it. When the axon of a cell is close enough to excite another cell and activates it repetitively or persistently, a metabolic change takes place in one or both cells, which increases the efficiency of the the pre-synaptic cell in the activation of the post-synaptic cell. Hebb, referred to this property as 'neurons that fire together wire together'.

2.1.4 Pyramidal Neurons and Interneurons

There are thousands of different types of neurons, with corresponding different electrical properties. Primal neurons are the most abundant cell type in the neocortex accounting for approximately 80% [33] of the neocortical neurons with 75% of them being pyramidal [45], named for their shape. Pyramidal cells are excitatory neurons using glutamate as a neurotransmitter [33]. Therefore most of the excitatory neurons within the neocortex are pyramidal neurons, with many apical and basal dendritic segments. Apical dendrites ascend vertically towards the cortical surface while basal dendrites emerge from around the base of the cell and branch horizontally. Due to their long axonal projections they have the ability to reach both local and distant targets [33].

The remaining cells in the neocortex are interneurons which are inhibitory releasing the GABA neurotransmitter and making mostly local connectioins [33].
Multiple sub-categories of interneurons and pyramidal neurons exist, whose classification is an active area of research [11].

2.1.5 Neural Circuit

A population of neurons that coalesce into a specific structure, form a neural circuit. The neurons in a neural circuit are interconnected by synapses and carry out a specific operation when activated. Neural circuits' functionality is mechanically compared to a "central processing units"(CPU), since they receive input from their surrounding environment and are responsible for its interpretation and of programming specific responses to those internal/external inputs, by relying on neural interconnections[53].

2.2 Neocortical Canonical Microcircuit

The mammalian brain is composed of six domains: the medulla, pons, cerebellum, midbrain, diencephalon and the cerebral hemisphares. One of the largest regions of the brain is the cerebral hemispheres where their surface is covered by a layer of neural tissue, namely the cerebral cortex. This layer is about 2mm to 4mm and is thrown into highly convoluted complex folds. One of the most striking evolutionary developments that have happened in the mammal brain is the great increase of the cerebral cortex area around the cerebral hemispheres. This has lead to the hypothesis that the cortex is associated with activities connected to the highest level of development in humans (affirmed by clinical and imagery evidence)[90].

The largest component of the cerebral cortex is of a type called neocortex, said to occupy 76% of the space in the human brain and 95% of the cortical area[90] in humans. It is a complex, highly organised structure that controls higher brain functions such as cognition and processing sensory, motor, language, emotional and associative information. Examples of these functions include sensory perception skills such as vision and touch, generation of motor commands, spatial navigation even conscious thought and language in humans. The neocortex, of which the literal translation from Latin is new 'bark' or shell, is the most recently evolved part of the cerebral cortex and is new with the evolution of animals. This means that it has be found to be present in brains of mammals but not other animals. Although the neocortex can also be sub-divided in different areas specialised for different functions such as vision or motion Mountcastle has proposed that all regions of the neocortex perform all the same basic operation to learn.

The neurons in the neocortex are formulated in a columnar structure by echeloned parallel and serial neural arrangements known as cortical (macro)columns [11]. In 1997 Mountcastle [61], remarked

that every region of the brain such as visual cortex,auditory cortex, prefrontal cortex- share a uniform appearance and basic structure of the neocortex, with minor discrepancies such as cell density or neocortical thickness. The uniformity and the similarity of structures in the neocortex has led neuroscientists to believe that across the neocortex in order to learn, the same basic computations are performed through all the regions. That is even if cognitive function such as hearing and vision are not the same and have different input, they are processed in the same manner. Studies have examined the effect of rerouting visual projections to the auditory pathway, at birth, in small ferrets, can affirm this hypothesis. It was uncovered that the auditory cortex developed functional features that are unique to the visual cortex such as cells being able to create an encoding of a two-dimensional map of visual space [84, 92]. Essentially the ferrets developed the ability to see through the auditory cortex, suggesting that information is processed in the same way in different parts of the neocortex even if they have different inputs and process different modalities.

Since then it is well documented in neuroscience that the cells of each macrocolumn are arranged in six distinct layers numbered from 1 to 6 [61, 36], where each layer consists of developmentally and functionally different sub-types of neurons. Layer 1 (L1) is the most superficial layer, while Layer 6 (L6) is the deepest neocortical layer, which both are largely populated by pyramidal neurons. The four in-between layers are populated by mostly small cells(pyramidal or non-pyramidal) or alternatively by mostly large pyramidal cells [90]. The layers are arranged depending on their primary input or output circuitry [60] and are specialised in distinct sensory, motor and associative tasks [40]. Much of the connectivity of the neurons within a given area of the sensory cortex is horizontally laminated within a 300–600 micron wide column, although spanning vertically across all six layers (Mountcastle, 1997–¿change). This cortical macrocoloumn of horizontally connected neural structures has been named a neocortical canonical microcircuit [88, 61]. An important property of neurons inside the sensory pathway of the canonical microcircuit is that their activity is driven by feed-forward sensory input so that their spiking patterns represent the presence of a feature or an object in the environment [5]. The canonical microcircuits across the neocortex are hierarchically interconnected by echeloned parallel and serial arrays of neurons that exhibiting a specific assembled organisation and together make up the cortical functions [40].

2.2.1 Basal Ganglia

Basal ganglia are a group of subcortical neural structures in the brain associated with motor control, as well as motor learning amongst others [47]. The basal ganglia consists of the striatum and globus pallidus in the cerebellum as well as substantia nigra in the mesencephalon and subthalamic nuclei in the diencephalon and pedunculopontine nucleus in pons [47]. The basal ganglia has multiple connections with the neocortex which will be analysed in 3.

2.2.2 Thalamus and Thalamo-cortical Pathways

The thalamus is a sub-cortical structure located in the diencephalon domain and its evolutionary expansion parallels the one of the cortex. It is the primal source of afferent to the neocortex and generally is ultimately involved in connections between cortical areas [30]. Therefore, in order to understand the computations of the canonical microcircuit and how raw sensory information is translated into knowledge, the thalamo-cortical input pathways are a critical part that must be considered. An important point to be made is that, it is well established that the brain processes information in a hierarchical way. In that sense in the thalamus, the 'first order thalamic relay' will pass directly sensory information(e.g. vision, sound, touch) to the 'first order neocortex', while a higher order relay nuclei(e.g second-order) will be responsible and allow for the transmission of information between different neocortical areas [11, 78]. The thalamus will also separate information in relation to modalities(e.g vision, audition).

Drivers and Modulators

Sherman and Guillery [79] proposed the idea that the glutamergic pathways -glutamate is a major excitatory neurotransmitter in the brain-, which include thalamo-cortical pathways can be divided into drivers and modulators. Driver input is associated with information bearing pathways therefore carrying the main information to their targets. Modulatory pathways modify this principal information streams by performing many of the classical modulator (e.g acetylcholine) operations [31]. A modulatory input usually is a synapse far away from the neuron's soma.

Excitatory and Inhibitory Cortical Targets

The effect of the action potentials projected from the thalamic excitatory (pre-synaptic) neurons to the neocortex will depend on the strength, position and timing of the connectivity to the (post-synaptic) neocortical neurons of the particular cortical region [31], as mentioned in section 2.1.2. Therefore, these attributes will determine how individual post-synaptic neurons will integrate inputs and issue action potentials.

Pathways from and to the thalamus related to neocortex

The thalamus is a collection of nuclei primarily composed of excitatory thalamo-cortical relay neurons that transmit sensory information to the cortex[78, 30, 11]. The relay of information starts from the activation of peripheral receptors such as retinal photoreceptors in the eye or skin mechanoreceptors and through the thalamus this information is projected to the neocortex. Multiple entry points exist into the neocortex layers receiving extracortical inputs but L4 has been perceived throughout time as the main entry [91, 78]. However the first order thalamus will also directly project excitatory input in L5B and L6 neurons [70], which have up till recently been assumed to be modulatory and not driving. Recent findings have demonstrated how L5/6 layer can receive sensory relay from the thalamus, so that sensory evoked post-synaptic potentials are exhibited with the same latencies as L4 [21, 70], which was up until recently though as the only principal target of thalamic afferent. Particularly Constantinople and Bruno [21] have provided evidence that the thalamic input activates in parallel two independent 'strata' of the cortex: one in L4; and one in L5/6. One of the inspirations for this study was that the same axons of the thalamo-cortical neurons that project input in L4, extend also and arborise -more sparsely- neurons in L5 and L6.

2.2.3 Inter-connectivity of Canonical Microcircuit (principal cells)

The layers of the canonical microcircuit contain various cell types that are densely interconnected in complex ways. [32, 11]. It is important to understand the different distinct connection between the neocortical layers, therefore in this section we will try to make a generalisation of the intrinsic connectivity between layers and then in following sections, a more specific picture will be provided. As mentioned in 2.2.2 sensory information from the thalamus will arrive in all cortical layers but L4 and L5/6 will receive dense driving sensory input [70, 21, 91, 79].

L4 principal cells are comprised by mostly stellate - excitatory GABAergic neurons- and some pyramidal neurons, whose intrinsic properties and coding properties appear to be similar. Layer 4 neural projections will target all layers but most strongly driving input will be provided to L2/3 neurons.

L2 and L3 are often considered together because their neurons exhibit similar patterns of connectivity and partly because of the difficulty of distinguishing them experimentally. L2/3 basal dendrites of pyramidal neurons extend through the entire macrocoloumn, therefore they arbour in all layers L1, L2/3, L4, L5. However, L2/3 main output will be projected strongly, locally to L5 and distantly to higher order cortices. The L2/3 to L5 feedforward connections are one of the strongest interlaminar connections in the circuit along with the L4 to L2/3 connection(described above).

L5 consists primarily of pyramidal neurons that are typically large with long axons branching widely. L5 pyramidal neurons can be divided into two main sub-categories that project in different local and distal areas. First the 'intratelencenphalic' neurons mainly found in upper L5, which project upwards, locally to L2/3 but also innervate the ipsilateral and contralateral cortex and striatum [89]. Layer 5 neurons will project driving input to higher order relays of the thalamus, for transmission from one cortical area to another [78]. Therefore they are responsible for projecting to other cortices, through the higher order relays. Second sub-category are the 'Subcerebral projection neurons' , which receive input from different cortical areas but in return they offer little reciprocal local output. This sub-category of neurons, in sensory cortices, mostly projects to sub-cerebral motor centres and can even directly drive movements [33, 54, 21]. It is important to emphasize that L5 neurons project to sub-cortical targets which are action related such as the striatum [21]. The striatum is the main component of the basal ganglia and is a critical component for motion related systems.

L6 is a quite heterogeneous layer of neurons and it blends into the white matter that forms the deep limit

of the cortex. L6 neurons will project back and provide modulatory input to the relay neurons that had provided input to L4 and L5/6 [74, 87].

L1 consists primarily of the dendrites of neurons in lower layers and contains few actual neurons. For the purpose of this dissertation L1 will not be considered.

The aforementioned structure of the neocortex is illustrated in figure 2.1 for better understanding. This is a wide simplification of the connectivity of the neocortical microcircuit, tailored for the purposes of this dissertation. The neocortex comprises immensely complicated intrinsic and extrinsic connections with multiple recurrent loops which are still a subject of research.

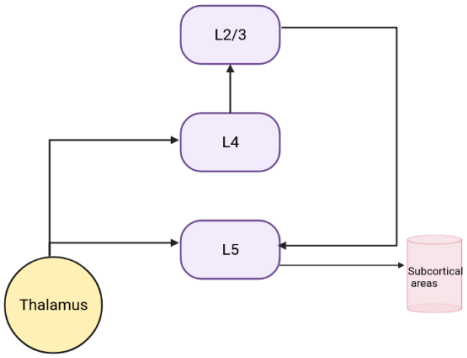


Figure 2.1: Structure of the neocortical canonical microcircuit

2.3 Related Work: Different Computational Models of the Canonical Microcircuit

There are four proposed leading computational frameworks that attempt to understand and model the canonical microcircuit functions: predictive coding, bayesian inference, hierarchical temporal memory(HTM) and Adaptive Resonance Theory(ARI) [27, 36, 46, 72, 81]. These models have in common that they postulate that the brain uses an internal representation of the environment to predict sensory input by using past sensory experience and movements, therefore commonalities with the ICM can be derived, as they both use prediction errors to drive learning. Moreover, they maintain the belief that the neocortex makes computation in a hierarchical manner so that the higher order regions receive input provided by the output of lower order regions [11]. Therefore, the shared concept behind those frameworks is that the purpose of the sensory cortex is to predict its sensory input, based on a hierarchical generative model of the environment which tries to minimise its prediction error [18].

The original predictive processing framework was proposed by Rao and Ballard in 1999 [72] , which implemented a computational model of the visual cortex derived by extracellular recordings from brain areas such as V1, V2 and V4 of a monkey who performed an experiment. The model was implemented by a hierarchical network and they used predictive coding to explain the end-stopping effect of receptive field neurons in V1. In the theory of predictive coding the top-down feedback connections from the higher order cortical area will try to predict the neural activations of that lower order cortical area. The prediction error between the top down prediction and the actual lower level activities will be conveyed by bottom-up feedforward connections. Therefore, the predictions from the feedback connections will serve to reduce prediction errors in lower levels.

Bastos et al [7] tried to relate the predictive coding framework with the neural computations intrinsically within layers of the cortical coloumn and partly extrinsically by including connections between columns in different cortical areas. In this theory the predictions of the generative model will be hierarchically broadcasted and adjusted in lower levels until the prediction error between the sensory inputs and the predictions is minimised. Mainly, the superficial layers of each canonical microcircuit (L2/3) will encode the error so that a certain pattern of neurons fire depending on the magnitude and direction of the prediction error and feedforward it to higher areas. On the other hand, mainly deep layers of the

canonical microcircuit will actually encode the predictions for lower areas and send feedback connections from higher areas to lower areas.

O'Reilly and colleagues [63] have also speculated how temporal difference prediction errors in the neocortex and pulvinar (part of the higher order thalamus, connecting cortical and sub-cortical domains [10]) could drive learning in the brain. In this model prior context will arrive at layer 6 of the visual cortex which will generate a prediction over the pulvinar. Bottom-up L5 intrinsic bursting neuron projections to the pulvinar will provide the actual input from the superficial L2/3, every 100ms. The L2/3 will simultaneously integrate information from other higher-level top-down feedback connections with bottom up sensory input to enhance the accuracy of its information before projecting to L5. This method presented the innovation of calculating the prediction error as a temporal difference and not as an error signal explicitly calculated by a distinct population of neurons. Because the pulvinar receives input sparsely every 100ms from L2/3 difference over time will be experienced by the pulvinar. The difference will be represented by changes in synaptic weights, with a prediction state from L6 causing an activation pattern, followed by the actual sensory input from bottom-up L5 creating a different activation state.
In a sense these frameworks model surprise in the brain In the interest of brevity hierarchical temporal memory models and adaptive resonance theory will not be mentioned.

2.4 Machine Learning(ML)

2.4.1 Reinforcement Learning

Reinforcement Learning(RL) is a branch of machine learning which concerns how to operate in a particular environment, by connecting situations to actions, in order to maximise a numerical cumulative reward. The environment is the world in which the learner lives and interacts, comprising everything outside the learner. The RL algorithm learns which actions will yield the highest reward through trial-and-error interactions with the dynamic environment. This is done by giving a higher reward to desired actions and lower reward to undesired ones. In RL in contrast with other ML domains such as supervised or unsupervised Learning, the algorithm learns to make appropriate decisions that lead to the optimisation objective [86], as well as learning hidden patterns of the data.

The key components of reinforcement learning are:

- Agent: The decision maker/ The learner
- Environment: The agent's world from which it learns by observing the effects of its interactions in it.
- Action: The agent interacts with the environment by performing an action.
- Current State : Current situation of the environment the agent is in.
- Next State : The situation environment after the agent has performed an action.

We formalise the idea of Reinforcement Learning by using the mathematical framework of Markov Decision Processes. Markov Decision Processes are an extension of Markov Chains, thus they poses the Markov property. amsthm

Definition 1 *(Markov Chain). Let $S_0, S_1, S_2, ...$ be a sequence of random states. This sequence is Markov if for every $t \in 1, 2, ...,$*

$$\mathbb{P}(S_t \in |S_0 = s_0, S_1 = s_1, ..., S_{t-1} = s_{t-1}) = \mathbb{P}(S_t \in |S_{t-1} = s_{t-1}), \text{ for any sequence } s_0, s_1, ..., s_{t-1}.$$

In other words, if t indicates time steps, then for all t, the probability of moving to the next state S_{t+1} is dependent only on the current state and not any of the previous ones. Therefore the agent does not have to look back at all the states.

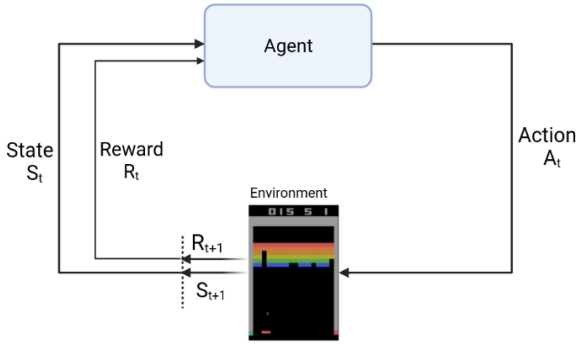


Figure 2.2: Interaction of the agent with its environment in a single time step t.

As figure 2.2 indicates interaction between the learning Agent, and the Environment occurs through a time sequenced loop. Atari Breakout is used as an example environment as it will be on of the main environments in this work's experimental setup. The actions, are of a finite number and are undertaken by the agent in discrete time steps $t = 1, 2, \ldots$. At each time step t the agent after receiving a representation of the environment's state s_t, it selects an action , a_t, from the probability distribution of actions, based on the observation. This will cause an observable change in the environment transitions from the current state of the environment s_t to the next state s_{t+1} along with a reward, $r_t \in \mathbb{R}$. The set of states, actions, rewards and resulting states, form a set called the Markov Decision Process(MDP).

Definition 2 *(Markov Decision Process.) A Markov Decision Process is a tuple* $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \gamma, \mathcal{R})$*, where:*

- $\mathcal{S}$ *is a finite set of observable states. / the finite state space set in the environment, where each state is finite and can be observed by the agent*
- $\mathcal{A}$ *denotes the finite action space, where in our case actions are discrete. The agent can take one action from the action space for each time step in the environment. The actions transition the environment to a different state and the agent gets a reward after this transition.*
- $\mathcal{P}$ *is the state transition probability function, which defines the conditional probabilities between all the states. The function* $\mathbb{P} : \mathcal{S} \times \mathcal{A} \to S$ *gives us the probability distribution over future states:*

  $$\mathcal{P}(s'|s, a) = \mathbb{P}[S_{t+1} = s'|S_t = s, A_t = a]$$

  Therefore being in state s *and taking at action* a *could take us to any state* s'*:* $\mathbb{P}(s'|s, a)$
- $\gamma \in [0, 1]$ *is called the discount factor*
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ *is a reward function, which gives the expected immediate reward the agent will receive when action* $a \in \mathcal{A}$ *is chosen in state* $s \in \mathcal{S}$ *:*

  $$R(s, a) = \mathbb{E}[R_{t+1}|S_t = s, A_t = a]$$

In Reinforcement Learning, the MDP is used to define and model the dynamics of the environment. Therefore, the probability of transitioning to a state s_{t+1} and get a reward r_t is dependent only on the present state s_t of the environment and action a_t undertaken by the agent. The implementation concerns an episodic MDP, in which the sequence of states will eventually reach a terminal state from which no future rewards will follow. Then the episode will reset and the game play will continue from there. An episode terminates after a predefined number of time steps or if the agent wins the game.

Episodes and Trajectory

The interaction of the agent and the environment from the initial state to the final state is referred to as an episode. In an episodic MDP, we have a terminal state and we assume that an episode will terminate(either because the agent died or completed the game) at a time step T. The episode information consists of the agent-environment interaction,which creates a sequential trajectory, denoted as τ. A trajectory involves a state, action, and reward, and a resulting state starting from the initial state until the final state:

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, ...(1)$$

Definition 3 *(Policy). At each time step the agent will select an action according to its (stochastic) policy* $\pi \in \Pi$. *Intuitively, a stochastic policy* $\pi(s)$ *is a function of the current state and a probability distribution that comprises the possible actions available to the agent for every possible state* s : $\pi(s) : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$. *If the agent is learning with respect to policy* π *at time* t, *then formally the policy will be defined as :* $\pi(a|s)$. *That is, the probability of choosing an action given a state, so that* $A_t = a$, *if* $S_t = s$. *Since the agent wants to maximise its rewards a good policy is one that assigns high probability to actions that have high potential future rewards and low probability to those with low future rewards.*

Definition 4 *(Discount Factor and Returns). In general, the goal of Reinforcement learning algorithm is to choose actions that will maximise the expected return denoted as* G_T. *The expected return is constructed by the sum of the total discounted rewards up to time step* T. *If the sequence of rewards is denoted as:*

$$R_{t+1} + R_{t+2} + R_{t+3} + ...R_T \text{ where } T \text{ is a final time step}$$

then the return would be :

$$G_T = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{t=0}^{\infty} \gamma^t R_{t+1}, \text{where } \gamma \rightarrow [0, 1]$$

. The time T *is a random variable and represents the termination of an episode. The discount rate determines the current value of future rewards. Thus as* $\gamma < 1$ *the agent will value rewards in later time steps in a decreasing trend. A reward expected to be received at a time step far in the future is worth less than if it were to be collected immediately. In that way the agent is still concerned to maximise the total expected return and has access/ takes into consideration/ to long-term rewards, but without neglecting the current rewards and being too farsighted. For a complete view of the reinforcement learning algorithm and theory it is important to pinpoint, for later reference, that rewards at successive times steps have a recursive relationship :*

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4}...) = R_{t+1} + \gamma G_{t+1}$$

+++ add things about infinite horizon states +++ Discount factor avoids infinite returns in cyclic Markov process (due to the property of geo- metric series); + Animal/human behavior shows preference for immediate reward.

Definition 5 *(State Value Function). The concept of maximising the total return over time with respect to the agent's policy, is formalised, for MDPs, using a mathematical construct known as the state value function :* $v_\pi : S \rightarrow \mathbb{R}$. *The following equation demonstrates the future returns the agent expects to receive given the current state* s, *following its policy* π, *which is the average reward observed after previous visits to that state :*

$$v_\pi(s) = E_\pi[G_t|S_t = s] = E_\pi[\sum_{t=0}^{\infty} \gamma^t R_{t+1}|S_t = s] \text{ for all } s \in \mathcal{S}$$

The value function is generated using the expected value of the returns, therefore a high value of expected returns will result in a high value function, indicating that the policy was good. As follows, the value function can be used to evaluate how effective is the policy. A better performing policy will correspond to a greater value of the value function.

Definition 6 *(Action Value Function). Similarly, we define a function for each state and action pair,given that the agent follows policy* π, *denoted as* $q_\pi(s, a)$. *The function calculates the expected total return given we are in some state* s *taking action* a, *following a policy* π

$$q_\pi(s, a) = E_\pi[G_t|S_t = s, A_t = a] = E_\pi[\sum_{t=0}^{\infty} \gamma^t R_{t+1}|S_t = s, A_t = a] \text{ for all } s \in S$$

Definition 7 *(Bellman Equation). The recursive relationship between the returns at each time step implies a recursive relationship between value functions as well, which is a fundamental property used throughout reinforcement learning. That is, there is a consistency condition, namely the Bellman Equation, between the value of state* s *and its possible successor states satisfied for any policy* π *and any state* s. *It is a relationship between the value function* $v_\pi(s)$ *for some state* s, *the agent's policy* π, *the state transition probabilities, the rewards* r *and the value function* $v_\pi(s')$ *for the resulting state s'. It looks at*

the possible state transitions and the probability the agent accessing those transitions given the agent's policy. It outputs all the possible outcome values with respect to the state s :

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \text{ for all } s \in \mathcal{S}.$$

Definition 8 *(Optimal Policy). Reinforcement learning algorithms improve their performance by optimising an objective function. Optimisation refers to the task of minimising or maximising that objective function, also called loss function when minimising. The optimal policy results from maximising the Bellman Equation. Finding a policy π that gives us the largest v for all states in our environment. An optimal policy $\pi*$ is defined as a policy better than or equal to all other policies π , if its expected return is greater or equal than each policy $\pi \in \Pi$ for all states. That is, $\pi* \geq \pi$ if and only if $v_\pi * (s) \geq v_{\pi'(s)}$ is satisfied for all $s \in \mathcal{S}$, and all $\pi \in \Pi$. There could exist a set of equally good policies, finding any of those policies would be sufficient for solving the RL problem.*

Definition 9 *(Optimal State Value Function). The optimal state value function $v_*(s)$ is the maximum value function over all policies:*

$$v_*(s) = \max_\pi v_\pi(s) \text{for all } s \in S.$$

Definition 10 *(Optimal Action value Function). The action value function plays a central role, as it tells us the value of selecting some action in a state and then following the policy after that. The optimal action value function tells us what is the best we can do for a given state.*

An important point to be made is that the optimal value functions and optimal policies can only be approximated. To actually solve this optimisation problem, is of extreme computational cost. A critical aspect of the RL problem facing the agent is the amount of computations it can perform in one time step and memory available to it, so the preciseness of an approximation is dependent on those factors.

Model Free Learning

The agent can learn the optimal policy using different methods. In this implementation, we will restrict our focus on model-free learning algorithms, as we are not supplying a model of the environment. In model-free learning, the agent discovers how to operate in the environment by interacting with it. The RL algorithm by receiving rewards and observing how its actions affect the state of the environment, it constructs the environment dynamics.

2.4.2 Deep Neural Networks

The implementation uses Deep Neural Networks as a function approximate for policies state- and action-value functions, so that we can maximise the value function for each state in the environment.

Neural networks are formulated according to the structure of the human brain, which is composed by billions of neurons arranged in layers, forming a network. A layer represents a specific set of neurons sharing a common property. A typical artificial neural network consists of the input layer, hidden layer(s) and output layer. Each layer will be connected in such way that information will be passed from one to another, mapping the input set to an output set. Therefore, the output layer is the one that is going to make the prediction.
Each neuron receives several inputs from the previous layer which are denoted as x_i.
Each input of a neuron in a layer will be assigned with a weight w_i. Deep Neural networks for each function will be characterised by some parameters, or weights, which will be updated so as to prioritise the inputs of each layer that are most important and help the algorithm achieve the best results.
Generally the output y of each neuron is calculated by the following linear equation:

$$y = \phi(\sum_{n=1}^{N}(w_i \times x_i) + b),$$

where ϕ is the activation function which is used to introduce a non-linearity transformation in neural networks, so that they are able to process and learn more complex relationships of the input. Without the activation function a neural computation would resemble to linear regression, which is not suitable

to represent complex underlying patterns.
In the context of reinforcement learning the input set will be a state and a reward with the output set being the approximations of the policy, the state and action value functions while the hidden layers are the inner-block where all the computations occur. The goal of the Reinforcement Learning algorithm is to find the parameters of that neural network so as to maximise the expected value of the sum of rewards over a trajectory. amssymb

Optimisation and Policy Gradients

The policy will be represented by a deep neural networks determined by a learnable parameter θ, which will represent the weights of the DNN. The agent will now choose the action a_t by computing the parameterised policy $\pi(a|s;\theta)$ for each action and then sampling the action from this probability distribution. We use policy gradients for optimisation, thus to improve the estimation of the state- or action-value function, while following the policy π. Generally, our goal is to achieve maximising the expected return of a trajectory G_τ by optimising the parameters θ. This will be denoted as:

$$\mathbb{E}_\tau[G_\tau|\pi;\theta]$$

At each iteration the parameters of the neural network will be adjusted by performing gradient ascent with respect to the parameter θ according to the learning of the agent:

$$\theta \mapsto \theta + a \cdot \nabla_\theta \mathbb{E}_\tau[G_\tau|\pi;\theta],$$

where $\nabla_\theta \mathbb{E}_\tau[G_\tau|\pi;\theta]$ can be computed as the expected value of:

$$\hat{g}(\tau) = G_t \nabla_\theta \sum_{t=0}^{T-1} log\pi(a_t|s_t;\theta)$$

To sum up, generally a random parameter θ will be initialised by the DNN which will be improved iteratively by sampling a trajectory τ from the environment according to the policy $\pi(a|s;\theta)$, computing $\hat{g}(\tau)$ and updating θ by:

$$\theta \mapsto \theta + a\hat{g}(\tau)$$

It is important to elucidate that gradient descent with respect to the weights of the NN can be performed too during training, when the goal is to minimise a loss function.

Finally, after adjusting the weights of the NN to the optimal direction at each iteration, by gradient descent or ascend, the weights of each layer of the NN will be updated by using **backpropagation**. Backpropagation is an algorithm that flows backwards the information calculated by gradient descend/ascend, through every layer from the last/output layer all the way to the input layer, using the chain rule. Backpropagation makes use of the chain rule to calculate the derivative from the last layer to the input layer.

2.4.3 Asynchronous Advantage Actor-Critic(A3C)

The Asynchronous Advantage Actor-Critic (A3C) is a state-of-the-art reinforcement learning algorithm based on the policy gradient method 2.4.2 approximated by neural networks developed by Google DeepMind [57]. The core idea introduced is the distribution of the training process into multiple CPU(central processing unit) threads, in order to reduce computational resources used by the agent geared towards data efficiency.

Actor-Critic

Reinforcement Learning can be broken down to value based and policy based methods. In value based methods we use neural networks to approximate the action value function $q_\pi(s,a)$, while in policy based neural networks are used to approximate the policy function $\pi(a|s)$ [73].

The actor-critic approach operates by approximating both policy and action value functions. The term 'Actor' refers to the learned policy, therefore the role of the actor network is to find an optimal policy, by using a policy gradient method. While the term 'critic' is a reference to the learned value function,

thus the critic network will evaluate the policy produced by the actor network through approximating the optimal state value. The two networks cooperate so that the policy is updated -in an online fashion- at every step of the episode based on the feedback by the updated value function, consequently the policy will select the most valuable actions for each state [85].

On policy method

The asynchronous Advantage Actor-Critic is an on policy Reinforcement Learning method. In on policy methods, the agent is updating the same policy it is using to generate actions [85].

Asynchronous and Multiprocessing Computational Method

This algorithm instantiates a global network with multiple independent agents, each interacting with their respective instance of the environment. Each agent has their own set of network parameters, a policy function and a value function, thus a different unique learning experience. Each of these self-contained environments will be processed on a separate CPU thread in parallel, a broad computational technique defined as multiprocessing, characterised by high computational efficiency [57].

The A3C algorithm is also classified in asynchronous computational methods. Agents are interacting with their instance of the environment at the same time, by inferring actions from an independent, local copy of the global actor-critic network. However, they are updating the parameters of the central neural network asynchronously, meaning at various times (without coordination with the other agents), using a stochastic gradient descent based optimisation method.

The whole process is demonstrated in Figure 2.3: Each local actor critic, or worker, is totally independent and learns on its own by updating the parameters of its own copy of the actor-critic network instance. When each worker terminates an episode locally, the neural network parameters of the global actor-critic agent are updated according to this local agent's learned parameters.
This algorithm is designed to imitate the circumstances of a real-life human environment, as humans will also learn by experiences of other humans, which accelerates their learning process, allowing the whole 'global network' to improve.

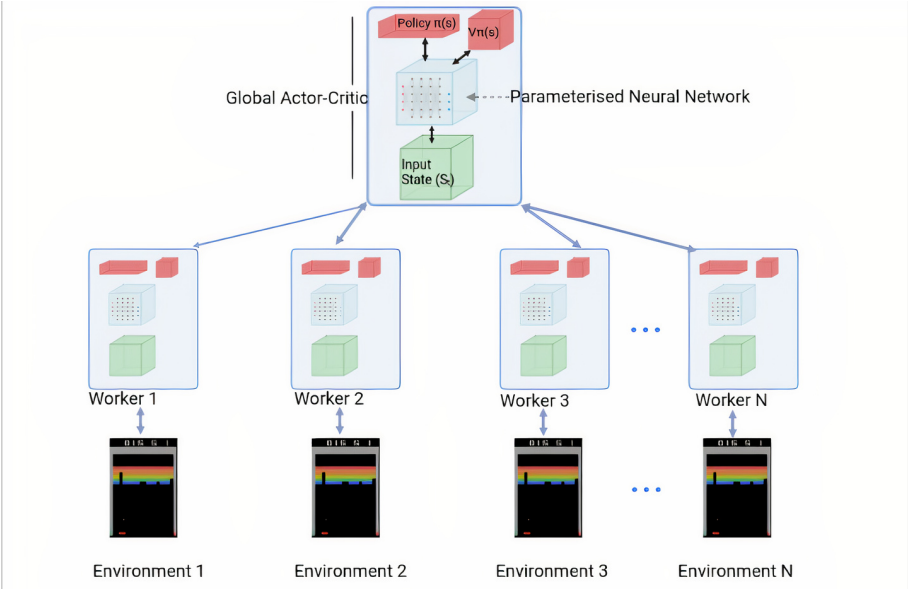


Figure 2.3: Illustration of the A3C high level structure. Adapted from [51]

.

Actor-Critic and Advantage

Reviewing the concepts of section 2.4.1 , general reinforcement learning solution methods are based on the policy and the state and action value functions. We use policy gradients for optimisation, thus to

improve the estimation of the value function. In the policy gradient method, we parameterise the policy, which represents the actor,so now we can denote it as $\pi(a|s,\theta)$, where θ indicates the parameters of our neural network used for the function approximation. This method aims to return the best possible approximation of the probability distribution over the action space -or the policy- by computing the optimal parameters θ of the neural network. As reiterated above, a good policy is one that generates high returns.

On the contrary, we can approximate the value function using another neural network, the critic network , with parameters denoted as θ_ν. To train the network we minimise the mean squared error, or difference, between the predicted value for a state and the true sampled rewards the agent receives from playing the game. This process is computed by the critic network by estimating the advantage function $A(s,a;\theta)$, defined as:

$$A(s_t,a_t;\theta)=\sum_{i=0}^{k-1}\gamma^i r_{t+1}+\gamma^k V(S_{t+k};\theta_\nu)-V(s_t;\theta_\nu), \tag{2.1}$$

where k can vary from state to state and is upper-bounded by T-max and $V(s_t;\theta_\nu)$ is the approximation of the state value function by the neural network with parameters θ. Using the estimated returns defined by (equation of G_t, say in equation 2.4,), the advantage function can be formulated as:

$$A(s_t,a_t;\theta)=G_t(s_t,a_t)-V(s_t;\theta) \tag{2.2}$$

Optimisation

In A3C, to learn the optimal parameter θ we must compute the gradient of the optimisation function, with respect to the policy parameter θ. The objective function for the policy is defined by Mnih et al as:

$$L(\theta,\theta_\nu)=-\log\pi(a_t|s_t;\theta)(G_t-V(s_t;\theta_\nu)-\beta\eta(\pi(a_t|s_t;\theta))=L_{actor}(\theta)+L_\nu(\theta_\nu)-\beta\eta(\pi(a_t|s_t;\theta)) \tag{2.3}$$

, where $\beta\eta(\pi(a_t|s_t;\theta))$ is the cross entropy(measure of randomness) of the policy. As this objective function is used as a loss function, our objective is to find the optimal weights that minimise this function, which is achieved by gradient descent.

Meanwhile, the optimal parameter θ_ν can be calculated by minimising the critic loss according to the advantage function:

$$L_\nu(\theta_\nu)=(G_t-V(s_t;\theta_\nu))^2 \tag{2.4}$$

While the actor loss is given by the following equation:

$$L_{actor}(\theta)=-A(s_t,a_t;\theta)\ln\pi(a_t|s_t;\theta) \tag{2.5}$$

In a successful training, during the course of training the values of the loss functions will converge to zero.

2.4.4 Convolutional Neural Networks (CNN)

Convolutional neural networks are a specialized type of neural networks that perform a convolution operation in at least one of their layers. The input layer of the CNN receives an input, which in our case will be an image. In this context, the image will be represented by a three-dimensional(3-D) tensor containing the scalar intensities of pixel data with dimensions of width, height and depth.

Neurons of each hidden layer of the CNN makes use of this mathematical convolution operation performed between the convolutional kernel with the layer's input matrix. A kernel, or filter, is a 2-D tensor, with width and height dimensions, that contains adaptable weight parameters which represent part of the image.

Definition 11 *(Convolution). A convolutional operation, or convolution, amounts to applying the kernel tensor to the receptive pixel input tensor, through calculating the inner dot product operation between the two tensors and produce a feature map.*
Therefore a convolution is a linear mathematical operation which for an image $I(i,j)$ with respect to a kernel ω can be defined as:

$$S(i,j)=\sum_m\sum_n I(i+m,j+n)\omega(m,n)$$

The procedure is repeated systematically for each overlapping part of the input image, left to right, top to bottom, as the kernel slides through the entire image in equidistant steps named strides. Therefore each layer learns multiple kernels in parallel, which leads to a significant reduction of parameters when compared to the dense weights of DNNs.

A feature map will be formulated by the values from the dot products, where each element will correspond to a scalar representation of the input encoding the spatial relations between the image pixels in the original image. In that way, convolutional layers can extract patterns from the pixel data, such as the ball in Atari-Breakout and encode them into feature representations. A convolutional layer will have several feature maps, so that multiple features can be extracted at each location. When a feature has been discovered, its exact location less essential if its position relative to other features is sustained. As the data pass through each additional convolutional layer the resolution of the feature map is being reduced allowing the detection to be less sensitive to shifts and distortions of a feature [4].

2.4.5 Self-Supervised Learning

Self-supervised learning is a technique of machine learning that can been though as the middle ground between supervised and unsupervised learning. In supervised learning the algorithm will learn to perform tasks or to classify data by looking at already labeled input data. In unsupervised learning the algorithm will try to extract features or patterns from unlabeled data. Therefore, self-supervised learning is an unsupervised learning method allowing to learn useful representations from unlabelled data, where the supervisory signals are created out of the unlabeled input data itself by allowing the algorithm to fine-tune the representations with few labels autonomously created by looking the input data. Consequently, the algorithm will withhold some part of the data and task the neural network to predict it, using the observed or unhidden part of the input. For example we can predict past or future frames(unobserved or hidden data) of a video by using the current ones(observed data).

Humans rely on knowledge attained from both supervised and unsupervised tasks in their every day life. Children will pick up a few important concepts from teachers and parents, but they will also fill up the gaps by exploring autonomously[16]. This autonomous exploration is guided by an intrinsic need to understand how things work.

2.5 Intrinsic Curiosity Module(ICM)

2.5.1 Intrinsic Curiosity

Introduction

Revisiting the Markov Decision Processes and the Reinforcement Learning framework, the goal of RL is to learn the transition model of the environment so as to maximise the expected return. However, an RL agent in complex environment is faced with the exploration exploitation trade-off [82]. Without sufficient exploration the agent will not be able to learn and discover effective control strategies, causing him not to achieve high rewards in complex environments, as it will not be able to collect sufficient learning experience to solve the proposed tasks. Contrastingly, the agent must not get lost in just meaningless exploration and overlook the ultimate goal of maximising the rewards.

Intrinsic and Extrinsic Motivation

An extrinsic reward is defined as an activity undertaken because it serves as a step to a separable consequence, which could be getting a high grade, satisfying survival needs such as food or materialistic wants such as money [43]. In psychology, intrinsic motivation, or curiosity, is defined as doing an activity when there is no apparent or separable reward except the inherently satisfactory activity itself and the enjoyment that the actor is deriving from it [75]. Various human behaviours could be categorised under this definition of intrinsic motivation such as a child engaging in playful, exploratory activities such as try to grasp, throw, bite new objects or adults engaging in creative activities, such as painting, singing or free-problem solving such as puzzles and cross-words.

Throughout the years there has been an attempt by psychologists to explain curiosity. In general, most

psychological approaches for curiosity seem to agree upon it being a particular relationship of uncertainty or unpredictability between an internal predictive model and the actual structure of the stimulus [13, 24, 42, 39, 25]. From early on as 1890, James et al[] defined it as "an inconsistency or a gap" of knowledge, followed by other psychologists developing similar definitions. Festinger in 1957 [25], suggested humans are motivated to reduce dissonance, that is inconsistency among beliefs or behaviours [43]. In 1957, Dember and Earl [24], suggested that intrinsic motivation is the discrepancy between the expected and the observed stimulus. Hunt in 1965 [39] suggested that interesting stimuli is one that gives rise to a difference between the perceived and the actual stimuli.

Importantly, intrinsic motivation has been recognised as a paramount for collecting sensorimotor and cognitive skills. Psychology studies have emphasised and explained its significance of exploration in acquiring or enhancing cognitive skills [13, 75, 22] useful for human development.

Related Work : Intrinsic Curiosity in Reinforcement Learning

Humans do not only seek to maximize extrinsic rewards like money, high grades, prizes, but also demonstrate an intrinsic curiosity, a motivation to engage in exploratory activities just for the sake of understanding how things work [43]. A growing number of researches in RL or developmental robotics have demonstrated how curiosity helps the agent, or robot, to learn the model environment dynamics with more efficient and internally incentivised exploration by developing several computer models [69, 67, 17, 6, 66]. Curiosity was introduce in reinforcement learning as an attempt to solve the exploration exploitation dilemma. It is often insufficient for an agent to just maximise its extrinsic rewards as there is often the danger of falling to a local optimum and remain at a stagnant reward level constantly visiting the same states, which the agent can overcome with exploration. Curiosity will be an incentive for the agent to explore the environment seeking new knowledge(or states-action pairs) [49] and so will also act as an effective apparatus for gaining new skills through the exploration. Therefore implementing curiosity in the agent will not only induce him to discover new states during exploration [8] but also exploration will contribute in decreasing the prediction error, since the agent would amass more experience and will be able to predict a wider range of state and actions [38].

The design of motivation systems to drive learning, based on curiosity has been a subject of research since the 90's by Schmidhuber , which suggested that the curiosity reward should be measured through the predictability of the task [76] and later on based on the learning progress [77]. Later on, the concept of the agent self-generating its rewards in reinforcement learning was introduced [85] and formalised as intrinsic motivation in reinforcement learning [6]. From a theoretical perspective curiosity has been successfully understood and demonstrated in different frameworks (e.g [65]). However, practical reinforcement learning algorithms remain sparse. At the present time, curiosity driven exploration has been used by agents to maximise information gain [48], as a medium to pursue less visited states by pseudo-counting states [64, 8] or my maximising the number of states an agent can reach [29]. Another broad category that has gained attention by many researchers is the generation of an intrinsic reward by the agent proportional to the difficulty of the agent to predict the consequence of its actions on environment dynamics [80, 55], in which the implementation of this dissertation belongs [67]. However, there are no previous work effectuating the innovation of this implementation which is modeling the features of the environment that are only important to the agent's learning.

Overview

An intrinsic curiosity module was introduced by Pathak et al [67], as a model which trains reinforcement learning agents to learn in a sparse reward environment, by modeling curiosity as an intrinsic reward signal, denoted as r_t^i, to guide exploratory behaviour, inspired by human intelligence. Therefore, the agent will produce its own intrinsic curiosity and will use it as a reward signal. Curiosity drives the development of substantial amount of knowledge and basic skills, as the agent –or the human in a real-life scenario- uses it as an incentive to explore the environment. By exploring the environment it is suggested that the agent will perform different actions in different states in order to observe and register the outcome. The agent will comprise two sub-systems: (a) a reward generator that produces an intrinsic reward signal based on curiosity -the intrinsic curiosity module- and (b) a policy that maximises the reward signal by selecting a sequence of optimal actions. The functionality of the ICM it is only to generates a new intrinsic reward signal based on information from the environment, therefore it is independent of how the agent model is implemented. A range of policy methods can be used with the ICM, but in this model the

agent is implemented using the Asynchronous Advantage Actor Critic(A3C) method mentioned in 2.4.3. The policy will be represented by a neural network with parameters θ_P denoted as $\pi(a_t|s_t;\theta_P)$ and the action that the agent a_t executes will be sampled by this parameterised policy.

In the implementation curiosity is formulated as the prediction error in the environment's dynamics. The idea behind this implementation is that in addition to trying to maximize extrinsic rewards, denoted as r_t^e, the agent would also just try to create a model of the environment dynamics and predict the next state s_{t+1} of the environment given the action taken a_t and the current state s_t while trying to lower this prediction error. In that sense, the agent will try to predict the consequence of its own actions on the environment by observing the rewards acquired and the resulting state. By predicting the consequence of, it is meant that the agent will predict the next state $\hat{s_{t+1}}$ according to the undertaken action a_t determined by the policy $\pi(a_t|s_t;\theta_P)$. Therefore the prediction error can be defined as the difference between the actual next state, s_{t+1} , and the predicted next state, $\hat{s_{t+1}}$. In states that have been revisited many times the agent will get familiarised, causing a decrease in uncertainty, as these states become highly predictable, therefore causing a reduction of the prediction error too.

This is analogous to the human brain, which will experience a high prediction error if something unexpected happens inducing its -theoretically called- parameters to update accordingly, in an attempt to resolve uncertainty in the future, by expecting to decrease prediction error[43]. Moreover, by supplying the intrinsic curiosity-or prediction error- as an additional intrinsic reward signal, the agent will be encouraged to visit unexplored or novel state-action pairs, taking into consideration the large intrinsic reward that follows those states. In other words, the higher the prediction error of a state, the higher the intrinsic reward will be. The aforementioned are accomplished by combining the extrinsic and intrinsic reward into a total reward signal demonstrated by the following reward function (corresponding to the expected reward R_t in the reinforcement learning section):

$$r_t = r_t^i + r_t^e$$

(2.6)

Therefore, following the RL section, the parameters θ_P will be optimised to maximise the expected sum of rewards:

$$\max_{\theta_P} \mathbb{E}_{\pi(s_t;\theta_P)}[\sum tr_t] \tag{2.7}$$

In a simplified explanation, the agent at each time step will choose the state-action pair from the policy that will offer a balance between having large uncertainty in prediction, because of the ICM exploration strategy, and maximising rewards. These actions will be taken, in hope that discovering new states will expand his knowledge about the environment allowing to make a more informative decision about rewards and maximise reduction in error -or minimise uncertainty- about the environment in the long-run.

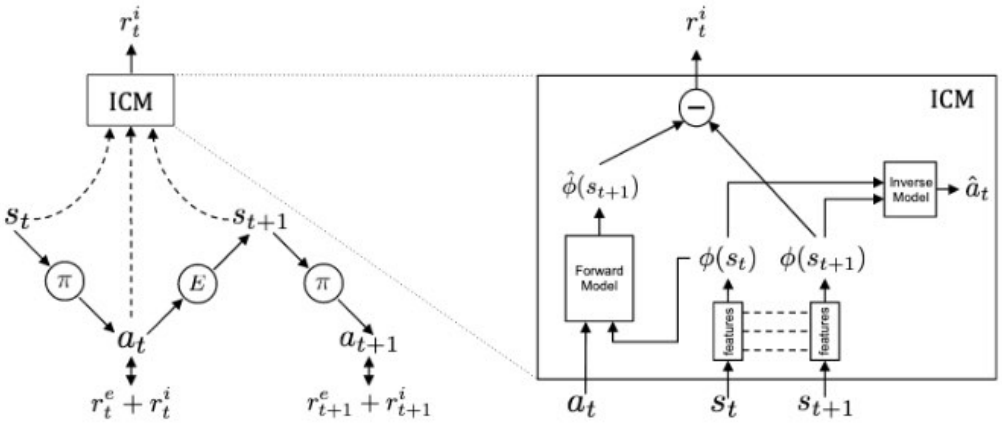


Figure 2.4: Illustration of the ICM high level architecture. Reproduced from [67]

.

2.5.2 Environment Representation and Prediction Error as Curiosity Reward

In order to get an accurate prediction of the next state, the agent needs to represent the dynamics of the environment in an effective way. The correct representation of the environment is a crucial factor affecting the performance of the algorithm. Predicting the future state in a high dimensional state space such as images is a challenging task which requires intensive computational resources, is time consuming and often does not reproduce the results that we require. Another problems that arises by using directly the pixel space as input, is varying stochasticity, which will affect the performance of the algorithm if not solved. In almost all games the dynamics within Open AI (Atari, CartPole, Acrobot, Pong) itself are deterministic. The agent will always start at the same initial state and a given sequence of actions will always lead to the same outcome. However, most non-trivial, environments have some degree of stochasticity, such as noise that will cause the same sequence of actions to output a slightly different outcome each time [50]. This indicates that often is almost impossible to reproduce the same state-action pair, which will cause the agent to have high prediction error, so high intrinsic curiosity as it will be curious about meaningless stochastic events that are not directly affected by the actions undertaken. Pathak et al [67], give the example of an agent continuously getting a high intrinsic reward for meaningless observing tree leaves moving with the breeze, which is a fundamentally stochastic process and inherently hard to model. Other examples include the presence of another entity in the environment whose motion is hard to predict and irrelevant to our agent goals, or shadow changes in the environment due to movement of other entities. Therefore, the agent needs to model only: (1) changes in the environment that arise from the actions undertaken by the agent; (2) changes in the environment that are not caused by the agent, but can affect the agent.

This implementation overcomes those limitations by embedding the environment state observations into feature representations, through encoders. The raw sensory input (an RGB image with heigh, width and channel dimensions in our case) is transformed into a feature space, by using convolutional neural networks which act as feature encoders denoted by ϕ. If s_t is the raw state observation then $\phi(s_t)$ is the feature representation of the encoded state, where the dimensionality is significantly lower than the raw state. At each time-step the feature space is extracted from the input image frame using a convolutional neural network. For example, in Breakout Atari a video frame is of height 210 pixels, width 160 pixels and has 3 colour channels(following the RGB convention). Those dimensions will produce a pixel array of $210 \times 160 \times 3 = 100,800$ elements in each time-step, from which many will be irrelevant and not useful to the agent. Therefore, in the model it is chosen to encode a state into a 288 element feature vector, with high level features that can still convey all the important and necessary information to the agent. Those features encode only information that is relevant to the agent by introducing an inverse dynamics model.

The inverse dynamics module, illustrated in 2.4, aims to generate a feature set that encompasses only the relevant information from the input image pixels in the reduced dimensional feature space. This module works by training a neural network in a self-supervised way to predict the action a_t undertaken by the agent given the current encoded state $\phi(s_t)$ and the next encoded state $\hat{\phi}(s_{t+1})$. Since the module is required to predict only the action then the agent will have no incentive to embed in its feature space, representations in the environment that do not directly affect him.

This feature representation of the current state $\phi(s_t)$ along with the action a_t will be used to predict the feature representation at the next time step $\hat{\phi}(s_{t+1})$. This will be accomplished by training a neural network on the forward dynamics model, illustrated in 2.4. The prediction error between the actual feature representation and the predicted encoded state will be produced and provided to the agent as a form of intrinsic reward. By combining the inverse dynamics model with the encoders and the forward model the ICM is created exploring the environment by using self-supervised prediction. Consequently, the self-supervised auxiliary tasks use the existing information as signals, which is the loss functions of the forward and inverse dynamics.

2.5.3 Inverse Dynamics Model(I)

This sub-module will take as an input the feature representation of the current state $\phi(s_t)$ and the succeeding state $\phi(s_{t+1})$ in order to output a prediction of the action a_t that was undertaken to end up from s_t to the succeeding statess_{t+1}. A neural network g with parameters θ_I will be initialised to train

the model and it will be represented by the learning function:

$$\hat{a}_t = g(s_t, s_{t+}1; \theta_I) \tag{2.8}$$

where $\hat{a}_t$ is the prediction of the action a_t. The neural network parameters θ_I will be optimised by minimising the following function:

$$\min_{\theta_I} L_i(\hat{a}_t, a_t), \tag{2.9}$$

where, L_I(or inverse loss) calculates the difference between the actual action and the predicted action by the inverse dynamics module. So as to achieve minimising this prediction error the model will back-propagate this error through the encoders as well as itself. The closer the predicted action $\hat{a}_t$ is to the actual action a_t the better will be the selected feature space , as the feature space will be more predictable and more related to the agent's action. As follows, the encoder will encode information about the environment that is directly useful to the task of the inverse model.

2.5.4 Forward Dynamics Model(f)

We train a neural network for the forward model ,f, at time step t, which will predict an estimate of the feature encoding of the next state, denoted as $\hat{\phi}(s_{t+1})$ given the feature vector of the current state ϕ_t and the action a_t:

$$\hat{\phi}(s_{t+1}) = f(\phi(s_t), a_t; \theta_F) \tag{2.10}$$

where, θ_F are the neural network parameters which are optimised by minimising the following regression loss function:

$$\min_{\theta_F} L_F\left(\hat{\phi}(s_{t+1}), \phi(s_{t+1})\right) \tag{2.11}$$

$$L_F(\phi(s_t), \hat{\phi}(s_{t+1})) = \frac{1}{2}||\hat{\phi}(s_{t+1}) - \phi(s_{t+1})|| \tag{2.12}$$

the intrinsic reward signal r_t^i is the scaled loss function of the forward model computed as:

$$r_t^i = \frac{\eta}{2}\left|\left|\hat{\phi}(s_{t+1}) - \phi(s_{t+1})\right|\right| \tag{2.13}$$

where, $\eta > 0$ is the scaling factor.

2.5.5 Composition of the Overall Optimisation Problem

The intrinsic reward is generated by a collaboration of the inverse dynamics module and the forward dynamics module, where g is trained to extract a useful, for the agent, representation of the environment and encode it in a feature space based on its actions only while f makes predictions for the environment's next state using this feature space only.The solution of the overall optimisation problem arises by combining equation 2.1, 2.3 and 2.6 :

$$\min_{\theta_P} {}_{\theta_I \theta_F} = \left[-\lambda \mathbb{E}_{\pi(s_t;\theta_P)}\left[\sum tr_t\right] + (1-\beta)L_I + \beta L_F\right] \tag{2.14}$$

where, $\lambda > 0$ is a scalar value that determines the importance of the policy gradient loss of the A3C against the intrinsic reward signal and $0 < \beta < 1$ is a scalar value that weights the inverse model loss against the forward model loss.

2.6 Open-AI Gym and Atari Arcade Learning Environment

2.6.1 Gym

Open-AI Gym [62] is a toolkit that provides a wide collection of simulated environments for developing reinforcement learning algorithms. It is an open source Python library and allows for communication between the agent, or the learning algorithm, and the environments via a standard intermediary API. This implementation will use the built-in simulated environments of Atari Breakout and CartPole for the agent to learn.

2.6.2 Atari

Atari 2600, was initially a Video Computer System console comprising of a diverse group of games ranging from arcade games to sports games such as Atari-Breakout or Atari-Pong. Different games are comprised by varying levels of difficulty, for instance Montezuma's Revenge is considered a a hard problem to solve for even modern reinforcement learning algorithms. Due to the diversity of the games and the unique, distinct characteristics of each they constitute a more than suitable environment to use as a set of learning problems for RL algorithms. Ideally, a good RL algorithm would be able to generalise in various games without changing the implementation of the model.

2.7 Statistical Measures

The results were the outcome of training an agent seven times and averaging the obtained values, in order to obtain more robust results. This is done because every time the algorithm is trained it will produce different results. This is a consequence of various factors, such as that weights of neural networks that the agent uses as function approximators are initialised randomly at each run, or the randomness(stochasticity) of observations, making the results vary. In order to evaluate how much they obtained results are based on randomness and stochasticity and how robust they are, the metrics below were used.

2.7.1 Standard Deviation

The variation or dispersion in a set of values is quantified by a statistical metric, namely the standard deviation [14]. It is calculated by the following equation:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{n}(x_i - \bar{x})^2}, \tag{2.15}$$

where N will be equal to the number of episodes, $\bar{x}$ is the mean of the values of each episode across the 10 samples and x_i will be the value of a particular episode i in a specific sample. The standard deviation of each of the seven samples will be calculated.

2.7.2 Standard Error

Standard error is a statistical measure that indicates the approximate standard deviation of a sample population, in our case each distinct run will constitute a sample population. Therefore, we will calculate how much the values of each run deviate from each other in order to measure robustness. Standard error is calculated as follows :

$$SE = \frac{\sigma}{\sqrt{n}}, \tag{2.16}$$

where n is the number of samples, in our case 10 as we have 10 sample runs of the agent and σ is the standard deviation of each sample. The smaller the standard error the more robust will be the obtained results.

Chapter 3

Connection: ICM and the Neocortical Canonical Microcircuit

Prediction errors are the central concept of both the implementation of the ICM and to most of the models of computations happening in the neocortex and the canonical microcircuits. One of the focuses of this dissertation is to try more biologically plausible variants of the ICM. In order to achieve this we need to map the module to a functionality of the brain. In our perspective both mechanism function by trying to predict their input, be it sensory information for the canonical microcircuit or pixel data for the agent. This statement is derived by considering the fundamentals of neuronal dynamics [7, 27]. Frinston [28], reviewed key brain theories, in biological and physical sciences and suggested that the basic elements governing actions, perception and learning are the optimisation of rewards or optimisation of its compliment surprise(prediction error, or cost). His argument was based on that biological system can be reduce to their homeostasis, such that the entropy between its internal and external milieu is minimised. Entropy is defined as the average of prediction error over time [7] therefore the biological organism will aim to minimise, over time, the surprise associated with sensory inputs in order to achieve flexible adaptation in the environment. This optimisation process can be argued that drives learning the brain.

This section will now investigate the hypotheses about how the canonical microcircuit could perform a similar function to the ICM in the brain. That is, speculating the hypothesis that this decrease in expected error could be signaled and be measured in the canonical microcircuit in a similar way to the intrinsic curiosity module. These computations in the circuits will take place so that regional learning progress is determined [43].

Both the intrinsic curiosity module and the most prevalent frameworks representing the neocortex have in common that they try to formulate an internal model of the environment by combining information from sensory experiences and movements. This mental model of the environment will be constantly updated by the agent or-the neocortex- trying to predict it sensory input. In a biological sense, the model is being inferred by the neocortex observing how its own inner states(e.g spike trains) flow and alter and updated by trying to minimise the error between its predictions about the inner states and the actual observations [18]. In that way, the input from the senses could be viewed as a training signal for the neocortex, whilst the model is trained to predict sensations.

3.1 Evolutionary Perspective

Firstly, it is essential to investigate how the cortical microcircuit's capabilities, in the sense of computational power and evolutionary advantage, could enable it to produce predictions and evaluate its own learning progress. Between literature review there seems to be a broader consensus that the canonical microcircuit is an unsupervised prediction machine as mentioned in the previous related work section 2.1.7 (predictive coding, hierarchical temporal memory [27, 36, 46, 72, 81]). There are lots of disagreements in the special features between them but there seems to be a broad agreement that somehow it is predicting its sensory input. These models have demonstrated the ability of the neocortex to be used as a prediction mechanism by exploring evidence and frameworks that could affirm this hypothesis. However, currently

concrete experimental evidence that the neocortex could be used for computing learning progress do not exist. The neocortex is a highly complex dynamic structure and defining what computations are performed within the cortical columns is a challenging task.

From an evolutionary perspective the neocortex could employ a mechanism of intrinsic motivation that drives learning. As the term neo- in neocortex discloses, the neocortex appeared fairly late in evolution. It is well established that the mammalian brain has increased in size enormously and that this increase is largely attributed to the evolution of the cortex due to the invention and expansion of cortical columns by evolution. [41, 90]. It can be argued that, the fairly recent invention and expansion of the cortical column circuits in mammals is connected with the fact that intrinsic motivation seems to be exhibited only in mammals as a behaviour [43]. Prior research has also demonstrated that the volume of cortical neurons in the human cortex is the largest amongst all animals with about 1.2×10^{10} neurons. Moreover, a sensible assumption to make is that the intrinsic motivation is a mechanism that manifested after the mammals were able to satisfy their extrinsic needs in a consistent basis. An animal would be only extrinsically motivated if it has to cover its survival needs. Such animal would be motivated by the pursue of extrinsic rewards such as homeostatic physiological needs like food and physical integrity and will evolve their behavioural strategies and focus on the acquisition of those experiences. This will lead to exploration of activities and methods to satisfy those needs fueling their development. Once mammals discovered efficient strategies to cover those simple extrinsic needs were satisfied, they had no motivation to develop further. The cortical columns and their expansion in the neocortex could be viewed as a product of evolution so that mammals could keep developing [43]. The apparition of a cortical circuit that could drive learning and minimise surprise further in the long run, by acting as a predictor could constitute the evolutionary mechanism behind intrinsic motivation. The neocortex will produce its own reward which will be the prediction error signal just like the ICM, so this mechanism will guide exploration, motivate development and learning and shape the intelligence of species [90].

3.2 Neuroscience Perspective

The hypothesis that the neocortical microcircuit and the ICM process input in an analogous way is based on the recent reestablishment of the thalamocortical pathways. In light of the fact the canonical microcircuit receives input in both L4 and L5 we can create a biomap from the ICM to the canonical microcircuit. Constantinople and Bruno [21] conducted an experiment on rats where they observed the effect of a sensory invoked input on the neurons of their neocortical canonical microcircuits. The input was in the form of sound in the auditory cortex while they used post-synaptic potentials of the neurons in the layers as a measure of activity responses to that input. As mentioned before thalamic afferent is the primal input in the canonical microcircuit which this experiment demonstrated that thalamo-cortical neurons synapse strongly not only on L4 neurons but also on L5/6 neurons thereby activating both of them. As a result, thalamo-cortical activity is simultaneously distributed, therefore L5/6 will receive distinct input from at least two pathways:

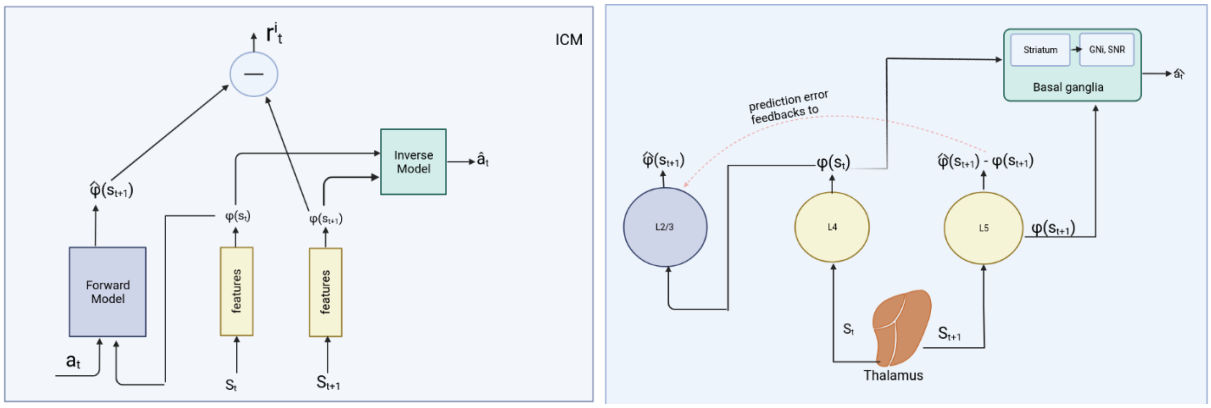
- Indirect pathway : Thalamus $\rightarrow$ L4 $\rightarrow$ L2/3 $\rightarrow$ $L5$
- Direct pathway: Thalamus $\rightarrow$ L5

Accordingly, these experimental revelation [21] can be interpreted as the sensory input arriving via a direct pathway to L4 and to L5, while L2/3 receiving delayed input since there is no direct thalamo-cortical pathway to provide input. The analysis of the post-synaptic potentials of the neurons in each layer demonstrated that sensory input evoked post-synaptic potentials to all layers but with different spike response latencies. Spike latencies are defined as the time interval between the first spike response of the post synaptic neuron after the injection of some stimuli [52]. Some of the deep cortical layers(L5/6) neurons exhibited similar and even rivaled the response latency of L4, previously though to be the main pathway. One of the longest spike latencies were observed on L2/3 neurons providing evidence that there are no thalamo-cortical connections to this layer.

The fact that for sensory input to arrive in L2/3 is obligatory to pass through L4, while the L4 and L5/6 layers receive direct input, could suggest that at a time step t, relatively to input information in L4 and L5/6, information arriving in L2/3 is constitutes sensory information about the past. By way of explanation at a time step t information will arrive from the thalamus and cause post-synaptic potentials distinctly to both L4 and L5/6. From L4 sensory information will follow the indirect pathway to L5/6,

therefore projecting firstly to L2/3 neurons which will then synapse directly to the L5/6 neurons inducing post-synaptic potentials at a time step $t+1$. During the time interval in which the information travels through the indirect path, new information will have been projected to L5/6 from the thalamus through the direct path. This is an important finding, as this insinuates that at a time step $t+1$ L5/6 will receive input from both L2/3, which processed information from a previous point in time t but also directly from the thalamus conveying information about the present $t+1$.

Analysing the progression of the information route within the microcircuit, perceptually the flow of the input could be mapped directly to the one of the ICM. In this dissertation, it is argued that not only that the flow of information follows an analogous route to the one of the ICM, but also the assembly of neurons in each layer perform similar operation to the modules of the ICM. Consequently, a mapping could be created from each layer of the canonical microcircuit to each module of the ICM, illustrated in 3.1, for which reasoning and evidence will be provided in subsequent sections.



(a) Intrinsic Curiosity Module

(b) Fitted canonical microcircuit into the ICM framework

Figure 3.1

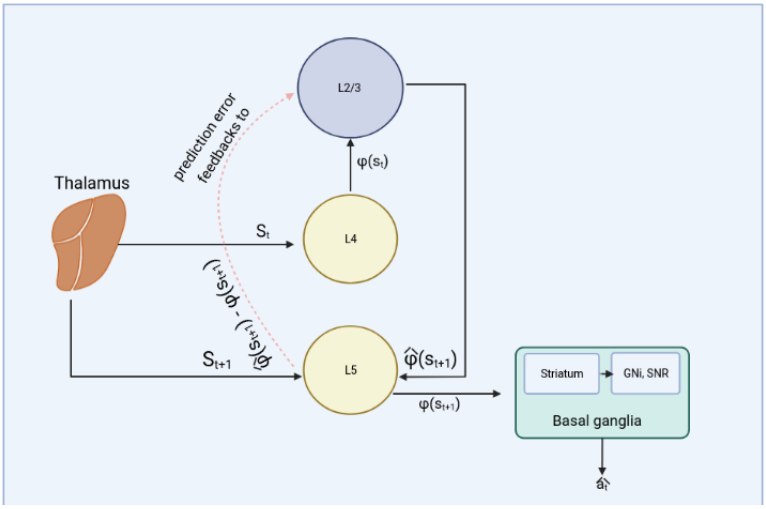


Figure 3.2: ICM framework mapped to the neocortical microcircuit

3.2.1 L4 neurons extract features from the thalamic sensory input

An important step in processing sensory input is extracting perceptually significant features. L4 receives very strong input from higher order thalamo-cortical projections and especially from core-type relay neurons specialised in transmitting sensory or motor information, but very few input from other thalamic or cortical regions. Evidently, this has led to the belief that L4 neurons are specialised in sensory processing [34] structured appropriately in the distinct cortical areas for each of the senses. A number of studies have demonstrated that upon the stimuli arrival L4 in visual cortex performs (feature) detection

of edges across which the luminance change. For example, visual cortex receives input from the lateral geniculate nucleus(key component in the thalamo-cortical visual pathway) in which cells have circularly symmetrical receptive fields that respond to 'on-centre' and 'off-centre' units [35]. In prior research, it was demonstrated that the visual receptive field of a cat is spatially structured according to the arrangement of thalamo-cortical inputs onto L4 neurons [19, 34]. The input neurons of the cat's primary visual cortex, which at the time of publication where though to be only L4 neurons, respond to stimuli of a particular location in their receptive field [10, 34, 19] thus extracting features from their input. Therefore, in our ICM model L4 could be paralleled with an encoder as their functionality is to extract features.

3.2.2 Linking sensory and motor areas

A biological organism can make a distinction between the environment states that are direct consequences of their actions using sensory epithelia. For example, humans can distinguish whether a ball, or external object, is moving towards them , or they are moving towards the ball, as a consequence of their own actions. Prior research have suggested the notion of action oriented predictive processing which links an action with its potential sensory representation [18]. There are 3 direct systems passing information between the cortex and the basal ganglia (subthalamic nucleus(STN), internal globus pallidus(GPI), substantia nigra pars reticulata (SNr)) using the striatum and the thalamus as a medium [26, 56]. As mentioned in 2.2.3, L5 projects to sub-cortical areas and specifically it provides input to the striatum, the main component of the basal ganglia, which regulates motor output. This implementation proposes that the input in the striatum will be the encoded version of the state coming from the direct pathway in L5 at time step $t+1$. Moreover, recent research demonstrated a pathway from STN to the granule cells of the cortex using pontine nucleus as a medium [56]. Granule cells can be found in layer 2 and layer 4 of the neocortex. In our model the forward model, which is mapped to layer 2/3, receives an action a_t at a time step t therefore in bio-mapping the STN could send information about the action selected in the cortical layer 2. Finally, evidence shows that there is a bilateral connection between the cerebral cortex and basal ganglia via the dentate nucleus, which is connected directly with GPi and SNr, consequently influencing the process of action selection driven by basal ganglia. Therefore, it is proposed that the inverse model could be implemented by the basal ganglia as there are reciprocal connections from and to the cerebral cortex that can be paralleled with the ICM. However, this assumption is missing a lot of details of how the basal ganglia would output the predicted action, were and how the loss function would be calculated and the details of self-supervision which are not mentioned here. Those are subjects that should be investigated in future work.

3.2.3 L2/3 performs predictions on the inputs(L4) received

It can be argued that, L2/3 will receive the encoded sensory input from L4 along with an action and will perform an encoded prediction of the future input. This prediction will be projected to L5. Therefore, the functionality of L2/3 can be paralleled to the forward model 2.10 in the ICM framework. This can be supported by evidence in the observed connectivity of this layer. Firstly, it has been established that axons from L4 will synapse strongly on L2/3 neurons providing its main input. The output provided by L4 will constitute the encoded features of the sensory thalamic input 3.2.1 . While L2/3 will also send massive projections in L5 neurons.

In general, after the updated view of the canonical microcircuit has been proposed the actual function of L2/3 has been a prominent subject of debate, where there is no definitive answer.

3.2.4 L5 encodes the input received and calculates a prediction error

L5 in our model would encode the input provided by the thalamus and compute a prediction error between the predicted input received from L2/3 and the actual sensory input directly from the thalamus 3.1, then it would backpropagate this error to L2/3 driving the learning process by changing the synaptic strengths in between the neurons of those layers. This could be represented by L5 being the encoder of state s_{t+1} and receiving the prediction $\hat{\phi}(s_{t+1})$ from the forward model or L2/3. The two encoded states $\phi(s_{t+1})$ and $\hat{\phi}(s_{t+1})$ would be compared and a prediction error will be calculated. Experimental studies have shown evidence that cortical neurons can detect prediction error signals using fMRI and EEG [7, 83]. L5 pyramidal cells receive extensive local input from neurons in multiple layers but in return they provide little in local responses [33]. They also receive additional direct driving input from thalamocortical connections as mentioned in previous sections 3.2,2.2.2. These neurons show a relatively high firing rate and encode information densely [23]. Such dense coding combined with its long-range

axons may provide a advantageous mechanism for efficient broadcasting of the cortical output to local and distant areas. As L5 represents the key cortical output, the cortex might use this dense coding as an efficient technique, as a small number of long-range axons do not require excessive physical volume [33]. Moreover, research so far suggests that the key role of L5 is to integrate the results of the local computations, which are received through L2/3, with the direct thalamic input [33]. This integration could be in the form of a prediction error calculated with a subtracting mechanism [72] in which the neuronal weights of the predicted input and the actual input would be balanced and opposing between excitation and inhibition in a given prediction error. For instance, negative prediction error neurons will respond if excitation of predicted stimuli is larger than inhibition of the actual sensory signals, while a positive error neurons will be activated when the opposite behaviour is exhibited. This prediction error will then be sent as a feedback to L2/3, following that L5a neurons send back a strong projections to both pyramidal and inhibitory interneurons in L2/3 [44, 11]. However, how these predictive errors are computed is still an open research, therefore a definitive answer cannot exist. Finally, in order to be able to compare the direct sensory input with the encoded predicted input, L5 must also encode the thalamic input and extract features. Therefore, L5 not only will calculate the prediction error but will also work as a feature encoder just like L4.

Chapter 4

Project Execution

4.1 Methods

To recapitulate the whole implementation, in principle, the environment that the agent interacts with, at each time-step, will output a current state s_t, an action a_t a reward r_t, which the agent will observe. The implementation is concerned with online reinforcement learning, therefore at each time-step the agent receives a tuple (s_t, a_t, s_{t+1}, r_t), which it uses to update its policy $\pi(a|s)$ -a conditional distribution over the set of actions given the set of states. The agent will then perform the next action a_{t+1} which is sampled from the agent's current policy $\pi(a|s)$ and will receive the reward r_{t+1} at the succeeding time-step. The agent will be embedded with curiosity which will be represented by the ICM. The function of the ICM will be to produce a signal based intrinsic reward r_t^i at each time step t based on the information received by its environment. Therefore, the implementation of the agent is separate from the implementation of the ICM, as the ICM is just a bolt-on module on the agent. Consequently, for the agent any model implementation could be chosen, which in this implementation A3C, introduced in 2.4.3 has been selected.

The base of the project was taken by Phil Tabor's [71] implementation of an A3C with ICM without feature encoding, taking a 4-D vector space as an input and not an image as needed. This implementation was modified and adjusted to accept images as an input, to use encoders for feature extraction and accept images, to work in a multitude of Atari Environments using custom wrappers and finally to be more biologically plausible inspired by the neocortical microcircuit.

4.2 Environments

There are interminable constructable learning problems that could be investigated in a reinforcement learning setting. The number of application of reinforcement learning are infinite and could include any learning problem in an environment involving a viable reward and a choice from an action set. Being able to solve this infinite set of problems is an infeasible task, therefore this dissertation narrowed down the problem space into a few but broad enough illustrative experiments that are possible to be used to investigate further in future research. This problem set consists of two gaming environments in which the agent learns to play autonomously and achieve the best performance: 1) the simple classic control CartPole environment from Open-AI Gym and the more complex Atari Breakout from the Arcade Learning Environment [9]. However, the implementation is scalable and experiments can be performed to both all classic control games of the open-AI gym and all the Atari games, without any algorithmic modification.

4.2.1 CartPole Environment

In the CartPole environment a cart moves along a floor, in both directions, that has no friction. A pole is attached to the cart and the agent is challenged to prevent the pole from falling over as the cart moves along the trail. The agent in order to balance the pole is allowed to apply a fore of +1 or -1 to the cart. At each time-step that the agent manages to balance the pole upright, a reward of +1 is provided. An episode will end when the pole is more than 15 degrees vertical, or the cart moves more than 2.4 units from the centre [1]. An agent will win the game if he achieves score of at least 190 points in 100 consecutive episodes. The game's dynamics are described as follows:

- **Observation space, s_t:** : The observation is a $4-D$ vector, so a valid observation is of shape (4,) with boundaries [0.05,0.05], where the values correspond to cart position and velocity and pole angle and angular velocity. The starting position of the system is a randomly initialised $4-D$ vector within the boundaries.
- **Discrete Finite Action Space, A:** Two valid actions are available to the agent: 1)apply force to the left; 2)apply force to the right
- **Reward Function, r:** $R(s_t, a_t)$. Here the reward can only be r=1 in every state of a trajectory.

4.2.2 Atari Breakout Environment

In Breakout Atari the agent moves a paddle and tries to catch and strike the ball with the goal of breaking the blocks on the upper half of the screen. When hitting a block with the ball the agent accumulates points depending on the colour of the block. Atari Breakout environment dynamics can be described as follows:

- **Observation space, s_t:** The raw observation space in the Atari library will be a box, which is a 2-D image, stored in a 210×160 RGB array. Each pixel is a tuple of three elements(RGB channels) each one representing red, green and blue pixels respectively taking values in the range of 0 up to 255. Therefore the observation will be of shape (210, 160, 3) representing the height, the width and the RGB channels of the frame, such that $s_t \in \mathbb{R} \times 210 \times 160 \times 3$.
- **Discrete Finite Action Space, A:** The actions that the agent can take are four: 1) Left; 2)Right; 3)No-Op (Do nothing); 4) Fire(Resets the game when the agent has missed the ball with the paddle and has lost a life. It is important for the agent to take the fire action or else the ball will not reappear when lost)
- **Reward Function, r:** $R(s_t, a_t)$ Depending on the colour of the brick that the ball will hit, the agent can amass 1, 3, 5 or 7 points. In each episode the agent wants to hit as many bricks as possible.

4.3 Modifying implementation to accept images

While in Atari environments the input observation space is already raw image pixels, generally classic control environments use a vector as an observation space as mentioned in 4.2.1. Therefore, the first step of the implementation was to render the environment and store the image pixels in an RBG array of shape (400,600,3).

4.3.1 Downsampling the image frame

However, the problem with a raw pixel space is its high dimensionality is that it affects heavily the speed of computation, making the algorithm slow and being demanding of expensive computational resources. To put this into context, taking the Breakout environment for instance, the raw pixel input would be of size $210 \times 160 \times 3 = 100800$, while the Atari environment makes updates of roughly 50 to 60 frames per second [15]. This issue has been addressed in a multitude of RL algorithms such as A3C or Deep Q-Networks, by downscaling and greyscaling the emulated image. Specifically, the screen in our implementation is cropped into a 84x84 pixel space representing height and width and converted into greyscale, by performing luminance mapping so that it has only one colour channel, as proposed by both A3C and ICM papers [67, 57] such that $s_t \in \mathbb{R} \times 84 \times 84$. Experimental researches have successfully established that preprocessing the frame before it goes into the model in such way does not have a significant impact on performance and is actually beneficial in terms of computational power as well as removing redundant information from the input [59, 58].

4.3.2 Frame Stacking

Moreover, the action that the agent will undertaken is a consequence of both current and previous frames as the agent important temporal information is stored in the sequence of previous frames. In Breakout, the agent needs to know information about the velocity of the ball or the direction of the paddle, while in CartPole, again the agent will need to extract information about the velocity of the cart or the direction

of the pole. This is information which can only be inferred by taking into consideration a sequence of previous images along with the current frame. A common technique proven useful to solve this problem in such RL settings is performing frame stacking. That is stacking together a pre-fixed number of previous consecutive images along with the current one and use this deque of stacked frames as an input observation s_t. Specifically, this implementation will stack four greyscaled and downscaled images, as proposed by both Deepmind's A3C and Pathak et al implementations [67, 57]. As it follows the dimension of the observation is $s_t \in \mathbb{R} \times 84 \times 84 \times 4$.

4.3.3 Frame Skipping

Finally, several studies has emphasised how frame skipping , or else action repeat, is a technique that when applied allows the agent to achieve state of the art performance [15]. Following closely again Deepmind's A3C and ICM implementation, the frame skip here is set equal to 4. Frame skipping equal to four in practice means that only one every four frames will be considered as an observation, so that the agent repeats an action four times before a new action is selected. Therefore, the collection of 4 stacked frames will actually add to the stack 1 in every 4 updated frames, meaning that out of 16 frames the four non-skipped frames will be stacked together to convey the required temporal information to the model. Another reason to use action repeat is that action selection is computationally intensive too so an appropriate number of action repeat, not too high or too low, can speed up computations without the cost of reducing performance.

It is important to mention that that Atari environments referenced in the Open-AI Gym documentation, such as 'Breakout-v0' or 'Pong-v0 use action repeating but on a random basis, so the action repeat at each game frame could be set to either 2, 3 or 4 randomly in attempt to overcome the deterministic nature of the environment. Therefore, for reasons of robustness and stability it is important to disable frame skipping in those environments, by using 'BreakoutNoFrameSkip-v4' for example, in order to remove this stochastic factor and obtain results with a reduced effect of randomness. Another source of stochasticity in Atari environments can be found in ALE interface, where it contains an action repeat probability. When the agent chooses an action from its policy, there is a 25% probability for it to be ignored so that a repeat of the previous action will be performed.

4.3.4 Using Gym wrappers to modify the Open-AI Gym environment

In order to achieve frame skipping, preprocessing the image and frame stacking Atari wrappers were used, which also allows the game to be replicated within the whole Atari library without any modification in the final implementation. For these tasks a python script was written. The functions will derive from two custom classes:

- A class of type gym.Wrapper : The wrapper class inherits the environment class so as it is implied by the name it is a class that wraps the environment class and modifies some of its attributes and functions. Custom functions of the environment include the reset, return and step functions .
- A class of type gym.ObservationWrapper : This wrapper allows for changes on the observations received from the environment by using the observation method of the wrapper class.

For both classes we will override the custom initialisation function by calling the class's super constructor in order to modify both the environment and the observation. For action repeat the Step function will need to be modified, which is defined in gym.Wrapper class. The step function returns four values: the observation , the reward , done(a Boolean signaling when an episode is done) and an info (optional information useful for debugging). For the fixed number of repeats we will call the step function and receive the observation, reward, done and info and then resetting the environment and receiving the initial observation of the next episode. Therefore, repeating a frame a fixed number of times.

For preprocessing the frame and stacking the four frames together the function derives from the gym.ObservationWrapper. The observation function takes a raw observation as an input which is converted to greyscale and resized to the new shape 84×84 using Python's image processing library OpenCV. For frame stacking we initialise an empty frame stack equal to the number of frames we want to collect together. We iterate through the size of the stack(four times) and append to that frame stack the preprocessed observation. The reset function will clear the stack each time, reset the environment and receive the observation.

4.4 Architecture of A3C Agent

A3C is used to evaluate the performance of ICM as a base line policy algorithm. The network architecture in the original A3C utilizes two successive convolutional layers, but in this project four convolutional layers will be used to be consistent with the ICM implementation. Therefore, the input processed frame, or state s_t will be fed into a convolutional neural network 2.4.4 containing four consecutive layers each with 32 filters, kernel size 3x3, stride of 2 and padding 1, consistent with the suggestion in the ICM paper [67]. Each convolutional layer is activated via an exponential linear unit (ELU [20]).

The convoluted output will be fed in a recurrent layer, in this case it will be the gated recurrent layer GRU with 256 hidden units, which allows to learn from long term temporal patterns. The output of the recurrent layer will be then fed into two separate linear layers :

- Policy, $\pi(a|s)$, layer explained in definition 3, which will be parameterised by the weights of its neural network θ denoted as $\pi(a|s;\theta)$ and will represent the function approximation of the actor. Since, the policy is the probability distribution over actions this linear layer will have outputs equal to the number of available actions in each game(four in Breakout and two in CartPole). A soft max activation function is used to normalise the output of the layer into a probability distribution, over the actions in $\mathcal{A}$. However, the soft max function considers the arg max as one hot representation of the output (so it outputs for example 0,0,1 ,0 and not a categorical). Thereby, the output of the soft max function will be passed through a categorical distribution, which is then possible to use by the agent to sample from the actions.
- Value Function, $v_\pi(s)$, layer explained in definitions 5, 6: This layer will have a single output, which will be the expected returns the agent is expecting to receive given the current state, according to its policy π. This linear layer will be parameterised by its neural network parameters θ_ν and will represent the critic.

During training the agent will update its neural network parameters in order to minimise the total actor-critic loss 2.3. The advantage function 2.2 which is part of the total loss 2.3 is calculated as a function of the total return which will include both the extrinsic and intrinsic reward as indicated in 2.5.1, so that the agent combines both exploration and exploitation. Finally for entropy reguralisation the γ hyperparameter was set to 0.99 and β to 0.01, as Deepmind's A3C paper has demonstrated that those yield the best performance. Therefore the total actor-critic loss will be calculated by:

4.5 ICM Architecture

For the ICM three distinct neural networks are initialised for function approximations. One for each of the forward, inverse model and the encoder, optimising the parameters θ_F, θ_I.
The neural network operations all together will constitute the feed-forward function of the ICM, which will return the new state s_{t+1}, the predicted action $\hat{a}_t$ and the predicted new state $\hat{\phi}(s_{t+1})$.

4.5.1 Encoders

The encoder neural network uses the exact same architecture as the one of the A3C agent. A single state s_t represented by an array of four stacked grayscaled and resized to 84×84 frames will be the input to the encoder, from which an array of 288 features will be extracted, namely the encoded state, $\phi(s_t)$ or $\phi(s_{t+1})$. The neural network will be composed of four convolutional layers, which will have 32 filters, kernel size of 3x3, stride of 2 and padding of 1. Each layer will be activated by the ELU activation function. The last convolutional layer output will be flattened to get a vector of dimensionality of 288 units, so that 288 features are extracted from the frame.

4.5.2 Forward Dynamics Model f

The forward dynamics neural network will take as input the encoded current state, s_t and an action, a_t and will output the predicted next state $\hat{\phi(s_{t+1})}$. Geared towards that goal the encoded state tensor and the action taken will be concatenated into one tensor of $288 + 1 = 289 -$ dimensions. This tensor will be fed into a sequence of two linear layers, with output of 256 units and 288 units respectively. The second and last linear layer will be the one making the prediction for the encoded next state, hence why

its output is a 288 dimensional vector/tensor. None of the layers will pass through an activation function in the forward model.

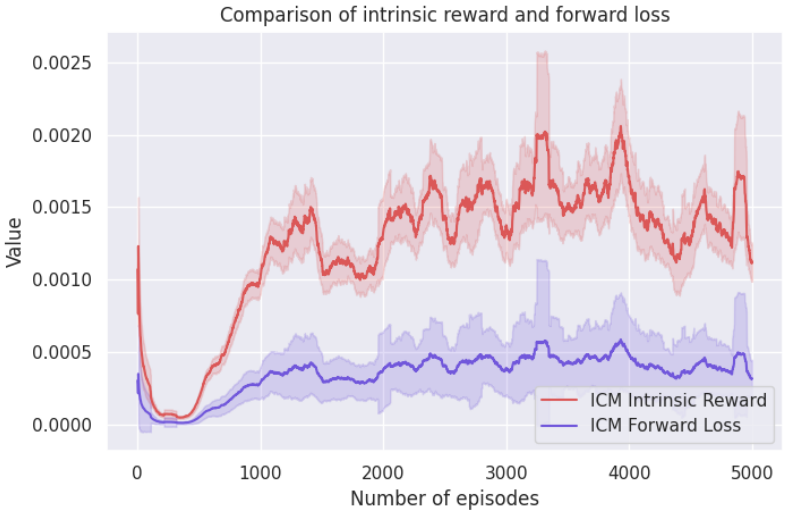


Figure 4.1: Demonstration of the relationship between intrinsic reward and forward loss (The agent was learning in the Breakout environment).

To train the forward model the mean squared error will be calculated, between the vector of the predicted encoded future state $\hat{\phi}(s_{t+1})$ and the vector of the actual future state s_{t+1} as demonstrated in equation 2.12. This error will be backpropagated to the forward model and the encoders but also used as an intrinsic reward therefore it is incorporated in the loss function of the A3C agent through the advantage function 2.2. Backpropagating the forward loss to the encoders is in contrast with the original implementation as the purpose of the encoder model was to encode features that are only relevant to the actions of the agent and not to give a low-dimensional input for improved performance, the effect of this will be explored in the results section 5. To be used as an intrinsic reward it will be scaled by the scaling factor η, as demonstrated in 2.13 and used as the intrinsic reward fed to the agent. In our case the scaling factor, or hyperparameter η is equal to 0.1 by following the execution of Pathak et al [67]. Figure 4.1 illustrates the relationship between forward loss and intrinsic reward and discloses that the intrinsic reward is just the scaled forward loss, showing that they follow the exact same pattern. The scaling factor that is applied to the forward loss in order to create the intrinsic reward is impacts significantly the agent's performance. If the scaling factor is small and the intrinsic reward is too low then it will have to impact on the agent's learning. While a large scaling factor causing a large intrinsic reward will distort training, as the agent will be incentivised to only explore and not exploit thus loosing sight of its initial objective to maximise rewards.

4.5.3 Inverse Dynamics Model I

The inverse dynamics neural network takes as an input the encoded current state s_t and future state s_{t+1} and will try to predict the action a_t undertaken by the agent. The inverse model will be also composed by two linear layers. First step is to concatenate the feature vectors of the current state, s_t and the feature state, s_{t+1} which both are $288-$dimensional, into one $288+288=576-$dimensional feature vector. This vector will then go through a linear layer of 256 units and then will be fed into another linear layer with output units equal to the number of actions available, depending on the specifications of each game (for example in breakout the last linear layer will have output of 6 units). The output of the last layer is not passed through a soft max activation, as in the training process it will be passed through a cross entropy loss function. The last linear layer will make the prediction of one undertaken action out of the possible actions $\hat{a}_t$ by the agent given the succession of states, which explains why its output is equal to 6.

To train the inverse model we calculate the loss function using the cross entropy error function. This will calculate the difference between the probability distribution over the predicted action and the vector of the actual action taken (which is the desired output of the predicted action). The loss is calculated by penalising the probability based on how far its from the expected value of the action. Therefore, if

the actual action undertaken is 2 (the third action out of the possible four) and the predicted probability output from the inverse model for taking that action is the lowest amongst others then the inverse loss will have a high value, indicating that the prediction made is far from the truth. This loss will be backpropagated to both the inverse model itself and the encoders so that both neural networks update their parameters accordingly. As follows, if the loss is large the encoder will have to change the features it encodes, so that it takes features that represent the state or observation better.

4.6 Training Details

4.6.1 Training

The training process of each agent will take place in a worker function. This function will instantiate the local agent, its respective local ICM and the local environment.

4.6.2 Shared Memory

The tuple $\{s_0, a_0, R, s_{t+1}\}$ along with the value function and the probability distribution over the actions for each transition and agent will be appended to a data structure representing the memory of the particular agent. During the training process at each time step t the agent will receive the samples and will append them to that data structure. Therefore each individual agent has its own internal memory, which at the termination of each episode is cleared.

4.6.3 Parallel Environment

The parallel local actor-critic agents, along with their local ICM simultaneously working on multiple instances of the environment are instantiated through a python class. Using the PyTorch multiprocessing package this class will instantiate twelve instances of the worker function, each agent(with or without the ICM depending on the specification) utilising a thread to make computations, therefore employing twelve CPU threads. The learning of each agent will be uploaded to a shared memory which will be used to update the global actor-critic agent and the global ICM. Each local instance will update the global parameters either every 20 steps or when an episode finishes(if it finishes before 20 steps).

4.6.4 Optimisation

For optimisation a shared Adam optimiser will be used, optimising the parameters that the global agent learns at a learning rate of $lr = 1e - 4$.

4.7 Step towards more biologically plausible ICM

To create a more biologically plausible model requires many steps. By observing 3.1 it can be distinguished that in the neocortical microcircuit at time step t, the sensory input s_t will go through the direct path and be encoded by L4, while at time step $t + 1$ sensory input s_{t+1} will arrive through the indirect path to be encoded by L5. Therefore, it can be inferred that biologically, two different encoders which are not shared are used to encode the two different inputs from the direct and indirect path at those time steps.

This is in contrast with the current ICM implementation which encodes both t and $t + 1$ input state sharing the same encoder. Accordingly, to create a more biologically plausible model the encoder in the current implementation will be separated as demonstrated in figure 4.2. This will be achieved by initialising a distinct convolutional neural network for each encoder with different parameters/weights. The left encoder, namely L4 will encode only the current state s_t and will create a feature map $\phi(s_{t+1})$ specifically for only that state, while the right encoder L5 will create a feature map $\phi(s_{t+1})$ particularly for the inputs of the next state. As illustrated in figure 4.2 everything else will remain the same in the implementation of the ICM and the agent. Briefly, the forward model will receive as an input an action a_t along with the encoded current state s_t, predict the next current state $\hat{\phi}(s_{t+1})$ and compute the forward loss from the discrepancy of the actual and predicted next state. The forward loss will be backpropagated back to itself and to the L4 and L5 encoder, in order to improve the neural network parameters θ_f in order to make better predictions but also to both the encoders L4 and L5 so that the parameters of each of the two CNN will be updated to improve feature extraction with respect to the purpose of making

better predictions. The inverse model will receive both current and next encoded states to predict the next action and will improve the modeling of features in the environment by backpropagating the inverse loss to both the L4 and L5 encoder, so that they update their neural network weights accordingly.

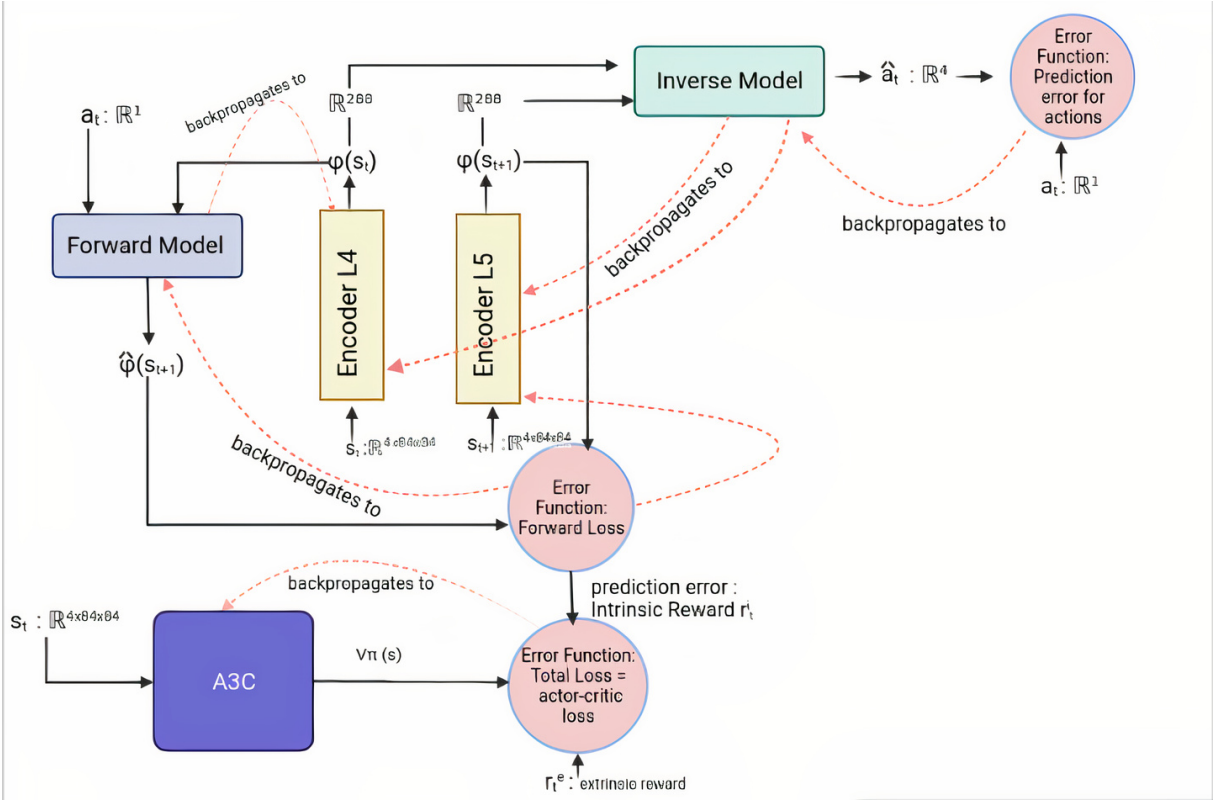


Figure 4.2: Training process of a local curious agent with separate encoders within a time step

Chapter 5

Critical Evaluation

5.1 Introduction

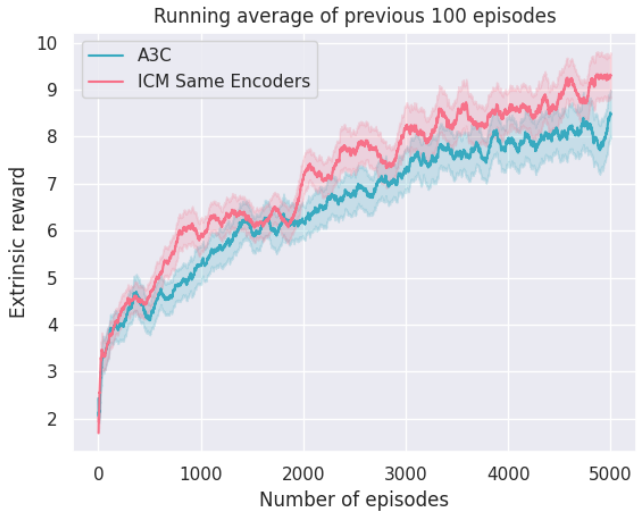
As discussed in the previous sections experiments were conducted by using three configurations:

- Just the A3C agent, where the agent operates exclusively with extrinsic rewards following its policy π.
- A curious agent, meaning A3C combined with the bolt on module of the ICM (A3C + ICM), so that it operates according to both extrinsic and intrinsic reward.
- A variant of a curious agent which is closer to a biologically plausible model, as it has separate encoders inspired by the neocortical microcircuit. (A3C + ICM (separate encoders).

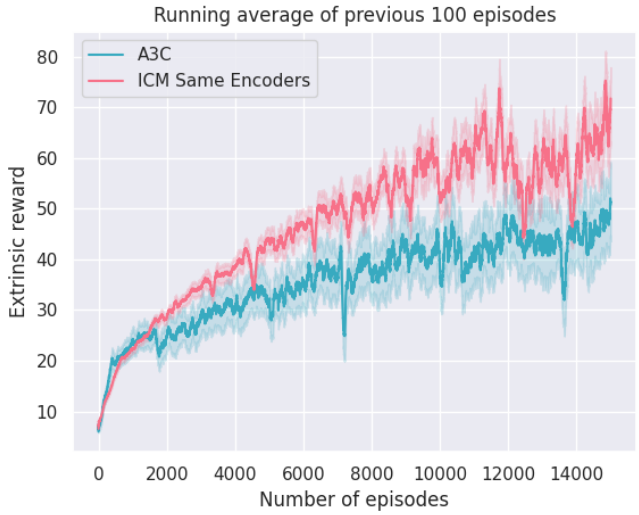
These variants were tested in two different environments that provide discrete extrinsic rewards to the agent which are Atari Breakout and CartPole. This implementation is not limited to those environments, it can be used along with any Atari or Open-AI Gym classic control environment. Breakout and CartPole where chosen as a representation to test the results of our agents. In this section, the performance of each of the three agents will be demonstrated in those two different environments and the findings will be analysed. The agents were trained using 12 parallel workers trained for 5000 time steps in Breakout and 15000 time steps in CartPole. For each different case the model was run 10 different times, to ensure robustness and the plots are the result of the average across those 10 runs, therefore the darker line in all plots represent the mean. All the illustrations include the standard error(SE) 2.7.2 across the seven runs illustrated as a shaded area bounding the bold line in each graph.

This section is separated into two subsections. One that displays the results of the ICM implementation with shared feature encoders and confirms that the results are aligned with the original ICM paper [67]. To compare the learning process of the curious agent, the performance of the A3C agent without embedded curiosity is demonstrated along with the curious agent performance. The second subsection will investigate and analyse the effect of separating the encoders of the ICM to two distinct neural networks which does not share parameters and have different distributions, inspired by the neuroscience of the neocortical canonical microcircuit.

5.2 Simple agent(A3C) and curious agent(A3C +ICM)



(a) Illustration of Breakout Environment



(b) Illustration of CartPole Environment

Figure 5.1: Comparison of the learning curves(bold curves), with shaded SE, of the baseline agent A3C(in blue) with performance of the curious agent (ICM+A3C) on two different environments(Breakout and CartPole). The learning curve depicts the running average extrinsic reward of the previous 100 episodes plotted against the corresponding episodes.

Figure 5.1 demonstrates that indeed curiosity will give an advantage to the agent in terms of performance, in accordance with [67]. An effective way to measure the performance of the agent is to track the extrinsic rewards i.e score of the game , it is able to acquire. The top graph is the accumulated score(with SE) achieved by each of the two agents during 5000 episodes in the Breakout environment, while the bottom one is the accumulated score(with SE) in the CartPole environment over 15000 episodes that each of the two agents has achieved.

It can be distinguished that curiosity improves the performance of the agent in both the Breakout and CartPole environment as it is able to achieve higher average scores, during almost the whole course of training. In Breakout the agent embedded with curiosity for the first 1000 to 2000 steps exhibited similar performance with the baseline A3C agent scoring similar rewards, but following that it continuously achieved higher rewards. Similar After 5000 training episodes in Breakout the curious agent(ICM+A3C)

was able to achieve an average score of approximately 9.5 while the vanilla agent (A3C) obtained an average score of roughly 8.5. Unfortunately, these results cannot be compared to the ones of the original ICM paper [67] as they have trained their agent for 9 million steps while our agent is trained only for 5000 or 15000 respectively, depending on the environment.

It is notable that in the original implementation extrinsic rewards were not supplied to the agent at all, as the purpose was to show how intrinsic motivation would guide an agent in a sparse environment. In this work, dense discrete extrinsic rewards are provided to the agent. Therefore, it can be deduced that intrinsic curiosity will boost performance and give an advantage to the agent even if it is already guided by extrinsic rewards. This can be explained by the fact that, the agent will have a more exploration oriented behaviour, which will help to surpass local minima that would have been created with a purely goal oriented behaviour.

The results can be affirmed by observing the tables 5.1, 5.2 which demonstrate the average of the maximum reward over the samples obtained by each implementation throughout the whole training and the average reward throughout the whole training calculated as the mean of the averages of all samples. Precisely, the maximum episodic reward is the maximum score value that the agent managed to achieve out of all episodes in a sample. To calculate the table values the mean across the seven samples was obtained and the ± standard error was given in order to show the dispersion and variation of the data across the samples.

Agent	**Max Episodic Reward Breakout**	**Average Reward Breakout**
A3C	16.6 ± 3.22	6.85 ± 0.649
A3C + ICM	18.84 ± 3.07	7.23 ± 0.652

Table 5.1: Comparison of the A3C model and the A3C+ICM in Breakout

Agent	**Max Episodic Reward Breakout**	**Average Reward CartPole**
A3C	178 ± 6.85	35.748 ± 1.81
A3C + ICM	243.14 ± 5.74	47.17 ± 1.62

Table 5.2: Comparison of the A3C model and the A3C+ICM in CartPole

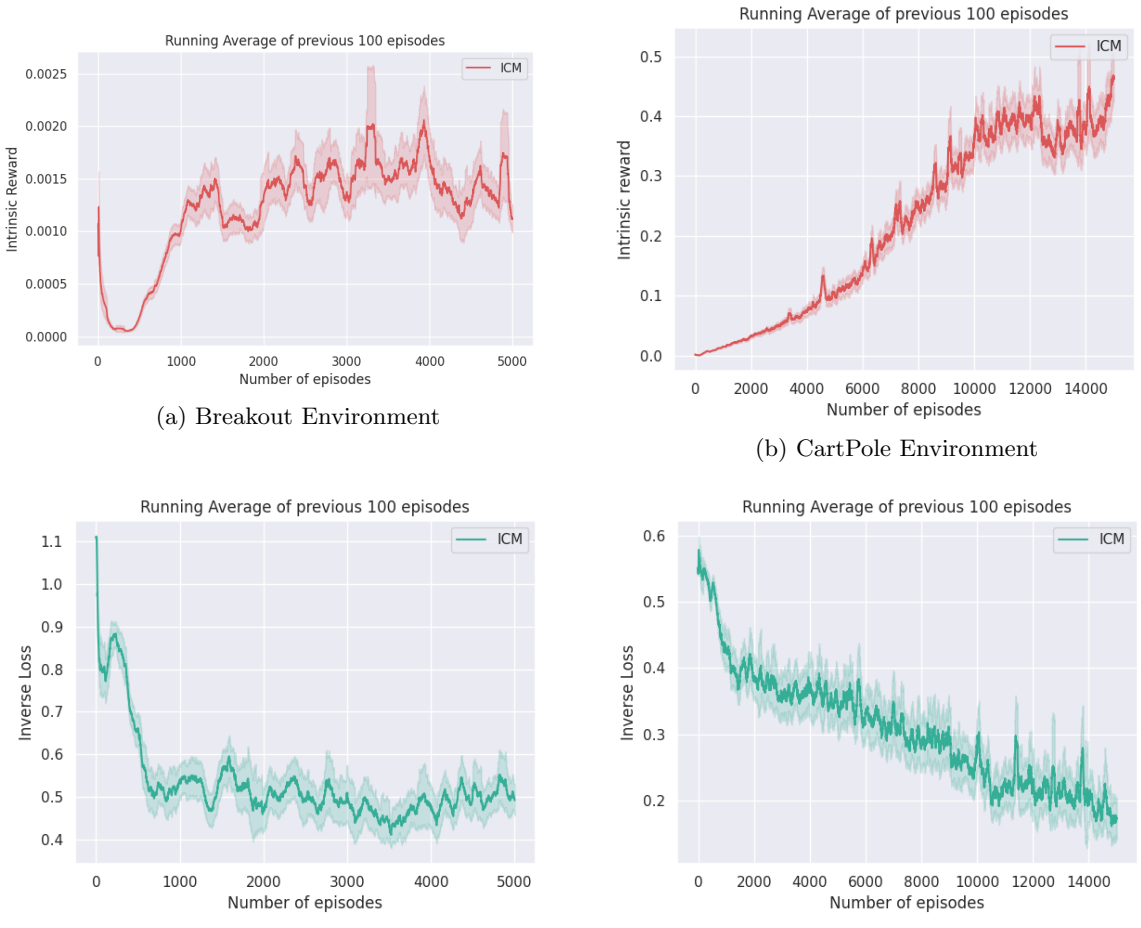
For both environments the curious agent managed to achieve a greater maximum episodic reward but also the average reward over all episodes throughout the training process is higher. In CartPole the maximum value that the agent can achieve in each episode is capped at 250 and the curious agent was almost able to reach that value, however the A3C with poor exploration strategy was far from reaching this value, in the same number of episodes. The improved results of the curious agent can be understood by an example: In Atari Breakout, the agent will gain extrinsic reward only when hitting a brick on the upper screen and depending on the brick the agent will be granted different rewards. The A3C agent with poor exploration skills will try to learn and hit the bricks which it knows that are going to earn a reward and only by chance it will discover that bricks in the upper layers will award more points. A curious agent with developed exploration skills will be interested in the pattern that is created when more and more bricks are struck, because the blocks disappear, inducing it to explore the patterns that are created by hitting different bricks, which will make it more likely to discover bricks with higher points. By discovering the new brick patterns the agent will receive an intrinsic reward and this is what will give him incentive to explore. Moreover, it has a higher incentive to stay alive, because if it looses the game will be set back to the initial configuration which the curious agent has already learnt, being a highly predictable state and offering small intrinsic reward. An increased episode length corresponds to higher chances of further increasing the episodic reward.

Consequently, by observing both the illustration of the learning curves 5.1 and the data on the tables 5.1,5.2 it can be deduced that curiosity used as an intrinsic reward delivers clearly an advantage to the agent when learning in an environment, as the intrinsic rewards accelerates the agent's process of learning, by offering an incentive to explore more and thus acquire more skills.

Moreover, an important observation that helps in the understanding of how curiosity driven learning with the ICM works is the relationship between the learning curve of the curious agent in CartPole 5.1

and the plot of the CartPole curious agent's intrinsic reward 5.2. After 12000 steps the learning curve of the curious agent initially experiences a steep fall and then it starts to increase but in a slower rate which is correlated with the fact that the intrinsic reward in 5.2 experiences the exact same pattern of sharp decrease and then slower increase than before. This is attributed to the fact that CartPole is a simple environment, so after 12000 steps the agent starts to become familiar with the environment's dynamics because it has learnt moderately how it works which reduces its prediction error therefore its intrinsic reward, as there are not a lot of remaining novel states to be discovered and as a consequence its learning curve will increase in a slower rate.

Finally, a remarkable observation when comparing the two environments in 4.1 is that the learning curve of both the curious agent and the baseline A3C exhibit a considerably higher variance in the CartPole environment, continuously fluctuating up and down, in contrast with the agents in the Breakout environment.



(a) Breakout Environment

(b) CartPole Environment

(c) Breakout environment

(d) CartPole environment

Figure 5.2: Top row illustrates the intrinsic reward, or scaled forward loss, of the two environments while bottom graphs demonstrate the inverse loss of the curious agents

An interesting analysis is comparing the different effect of curiosity on the two distinct environments of CartPole and Breakout demonstrated in both figures 5.1, 5.2. Notably, both settings regularly give extrinsic rewards; however, adopting curiosity in the CartPole environment appears to have a greater influence on performance than in the Breakout scenario. CartPole is an environment in which the agent's actions have an immediate consequence on the score as the agent applies force directly to the pole and the score is directly related to the motion of the pole. However, in Breakout the main control action the agent needs to perform is to catch and strike the ball using the paddle, which does not grant directly any points/ extrinsic reward. Points will be gained only when the ball hits a brick. Therefore each action of the agent will not have an immediate effect on the extrinsic reward causing a temporal difference between undertaking an action and receiving the reward caused by that particular action [12]. The agent

has a hard time attributing a particular action to the corresponding reward because of that temporal discrepancy. Moreover, the state-action relationship could take the form of both one to many and many to one, meaning that a change in a state could be attributed to many actions while also multiple changes in states could be attributed to only one action. Even when the agent receives four consecutive frames it is not enough to convey those temporal dynamics. Consequently, the curious agent will receive a lower intrinsic reward in Breakout than CartPole.

Technically this can be explained by interpreting the dynamics of the ICM. During training, based on the action undertaken by the agent, the total reward will be computed according to the extrinsic reward, the environment's dynamic behaviour and the agent's prediction. The agent is given a continuous extrinsic reward but also is rewarded proportionally to the discrepancy of its prediction model and the observation of the environment, that is the intrinsic reward. However, the intrinsic reward, produced by the forward model, is a function of the environment dynamics with respect to the performed action [68], as the agent will be induced to chose actions whose effects are poorly modelled because those are the most rewarding. Nevertheless, the actions in the Breakout environment will be modelled inadequately because of the discrepancies analysed above, in contrast to CartPole.

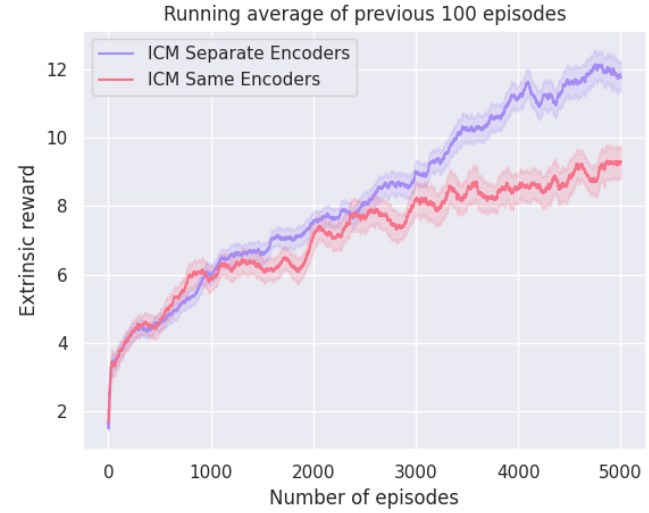
The forward model accepts the current action a_t chosen by the policy π and state s_t and tries to predict the next state $\hat{\phi}(s_{t+1})$ using those, but the next state could be a consequence of a previous action or a sequence of previous actions. It is really hard for the forward model to learn the temporal dynamics, therefore exploration will not be very efficient and will eventually lead to a local minima in its loss function. Therefore, even if the ICM solution introduces a way to balance exploration and exploitation and undeniably gives an advantage in performance, the agent's loss function will still reach a local minimum, as the forward loss is one of the components used to calculate the agent's policy gradient.

Comparing the learning curves in CartPole and Breakout of figure 5.1 and the forward losses in 5.2 can affirm the aforementioned analysis. In 5.1 although in Breakout the curiosity gives a definite advantage on the performance of the agent and induces him to learn quicker, in CartPole the learning curve of the curious agent is steeper implying that the curious agent is able to use his exploration technique in a more efficient way.

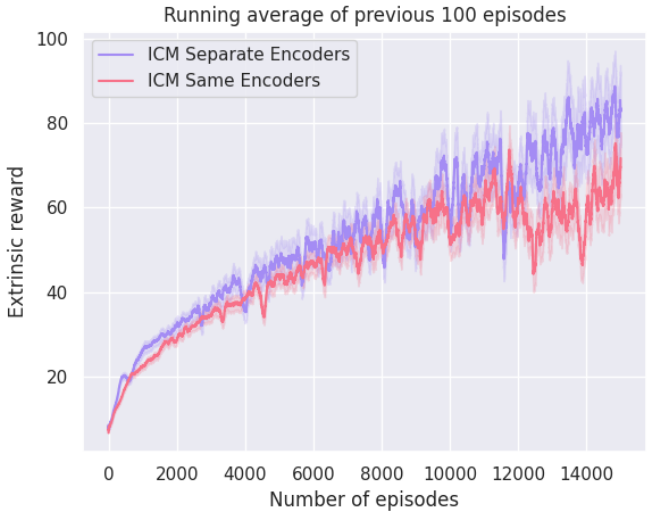
In 5.2a although the intrinsic reward is present and does help in exploration, it is ranging around the same levels, without any notable increase or decrease, which indicates that the agent does not engage with a highly exploratory behaviour. If the agent was fully exploiting his exploratory potential the intrinsic reward would be increasing as it would be induced to discover more and more novel states. However, the agent can never predict the next state accurately as it is not just a consequence of the current action therefore the prediction error always stays around the same levels. Meanwhile, the forward loss of the curious agent in the CartPole environment 5.2b is increasing consistently indicating that the agent explores the environment discovering novel states which cause the intrinsic reward to increase, due to the discrepancy between the predicted and actual state.

Similarly, the inverse model will take the current encoded frame and the next encoded frame and will try to predict the action that caused this change in the environment, in order to improve the features it selects for modeling the environment. Because of the temporal discrepancies and the dynamics between the changing states and the actions, the inverse model frequently will not be able to actually predict the action that caused this change in state as it could be correlated to a sequence of prior actions. This is depicted in 5.2c, where the inverse loss of the curious agent operating in the Breakout environment experiences a steep decrease at the first 1000 episodes but then is stagnating around the same levels for the rest of the training with maybe experiencing a slight decrease. Meanwhile in the CartPole environment 5.4b the constant decrease of the inverse loss is very apparent. A stagnating inverse loss actually means that the agent does not learn to map the actions to the corresponding states which aligns with the aforementioned argument about the temporal dynamics of Breakout Atari. A decreasing inverse loss indicates a successful learning progress in comprehending the relationship between the actions and the changing states.

5.3 Comparison of ICM with separate encoders and ICM with shared encoders



(a) Illustration of Breakout Environment



(b) Illustration of CartPole Environment

Figure 5.3: This figure demonstrates the average episodic reward of the previous 100 runs of the curious agent (ICM+A3C) with shared encoders compared to the curious agent (ICM+ A3C) with separated encoders.

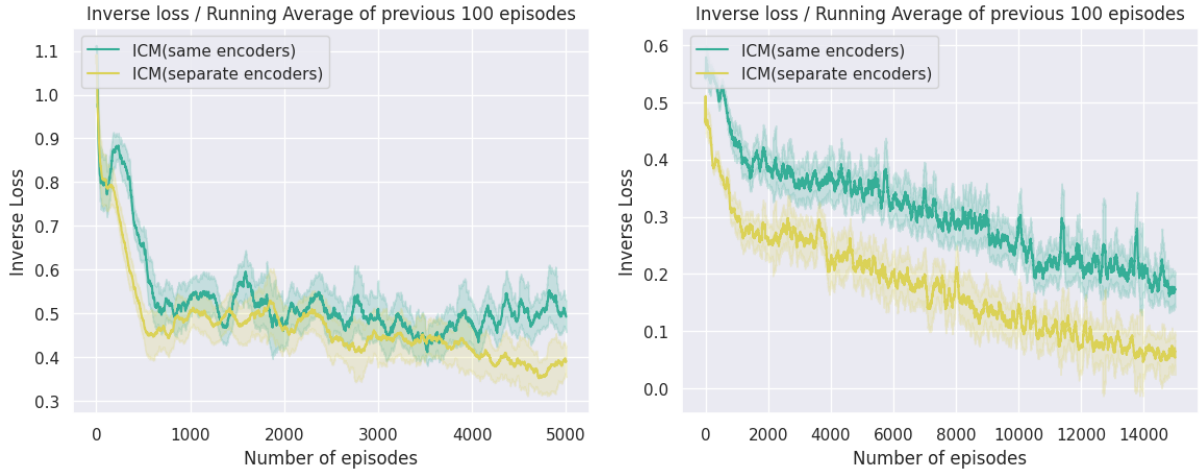
In figure 5.3 the increased performance of the agent with separate encoders can be observed in both CartPole and Breakout environment. The top graph illustrates a curious agent as per the original implementation of the ICM and a curious agent which is more biologically plausible learning in the environment of Breakout, while the bottom one in the environment of CartPole.

In the Breakout environment 5.3a the agent with the incorporated ICM with shared encoders(ICM) seems to overtake the ICM with separated encoders, which for the purposes of brevity will be named ICMS. After 1000 steps the curious ICMS agent demonstrates a steeper learning curve than the curious ICM agent achieving continuously a higher average reward. Importantly, after 3000 episodes the ICMS agent displays a substantially better performance as a significant divergence between the scores achieved by the curious ICM and the more biologically plausible ICMS is illustrated in the graph.

In the CartPole environment all though the ICM with separate encoders achieves higher scores on average, the increase in performance is not as prominent as it is in the Breakout environment. Interestingly, sometimes the ICMS will even achieve a lower score than the original ICM up until 12000 training step. An observation to be made is that before ICMS reaches the benchmark of 12000 episodes its performance experience a large fall similar to the one of ICM after 12000steps. However, the ICMS recovers almost instantaneously while the ICM demonstrates a lower learning progress afterwards.

The increase in performance and the acquisition of higher accumulated reward can be explained technically. Generally, going back to figure 4.2 the neural network for the L4 encoder and the neural network for the L5 encoder receive the exact same backpropagation signals in order to adapt their weights, which are from the loss of the inverse model and the loss of the forward model. Consequently, the weights of the two networks will be adjusted towards the same goal which is improving the feature representation so that the loss of the forward and inverse model are minimised.

It is important to elucidate that backpropagating the weights from the forward model might have not disturbed the learning process in the CartPole and Breakout environment as they are mostly deterministic and contain little to no noise, but if the implementation needs to be scaled for various different and more complex environments it is important that the forward loss is not backpropagated to the encoders so that the agent does not remain curious, thus gain an intrinsic reward, for exploring parts of the environment that are irrelevant to its final goal which is maximising extrinsic rewards, as mentioned in 2.5. To achieve that PyTorch method detach must be used, so as to detach the forward model from the encoders and stop it from backpropagating the forward loss.



(a) Illustration of Breakout Environment

(b) Illustration of CartPole Environment

Figure 5.4: This figure demonstrates the inverse loss of the ICM with the shared encoders and the ICM with the separate encoders in the two different environments. The inverse loss is given as an average of the previous 100 running episodes.

However, the two encoder model are diverse in the sense that each model's parameters are initialised differently while also the input they are trained on is different. The L4 encoder at time step t will receive the s_t observation as an input while the L5 encoder will receive the s_{t+1} observation, while also having received at the previous time step the observation s_t. Neural networks are non-linear function estimators, therefore when they are trained on different data they will output a different estimation for the function. To give an example that will aid the explanation: the creator of the ICM implementation has also published an implementation where the agent is driven to explore via disagreement [68]. This implementation introduces an ensemble of forward prediction models $\{f_{\theta 1}, f_{\theta 2}, ...f_{\theta k}\}$ where each forward model has the same functionality as the one in ICM. The intrinsic reward that will guide the agent's policy is introduced as the discrepancy between the different forward models.Each model is trained using maximum likelihood estimation that maximises the prediction error :

$$||f(x_t, a_t; \theta) - x_{t+1}|| \tag{5.1}$$

which equivalent to the ICM forward dynamics equation that minimises prediction 2.10, 2.12. Therefore,

to maximise exploration the discrepancy between the individual forward model predictions must be maximised. This is achieved as a large difference will occur when the models trained on the state-action tuple are not learnt, but small when there is noise from the environment, therefore accounting for complex scenarios. The author to ensure diversity between the distinct models by initialising different weights for the neural networks of each forward model and by training the models on different samples of data chosen randomly with replacement. This ensures that each model have different prior distributions therefore will output a different result, or posterior given the same input.

This implementation can be paralleled with the ICMS implementation and explain why two separate networks for the encoders has increased the exploration of the agent and yielded better results in terms of performance. For each individual encoder a distinct neural network is initialised and each encoders at a time step t will receive different inputs and have 'different prior distributions' they will encode different features from each other. Due to their discrepancy between the features that the two encoders encode the agent could be induced to explore further the environment and be more efficient. For example, a hypothesis could be the inverse model accepts the encoded current state and encoded feature state, as the neural networks will contain different parameters they will encode different features for the current and next state, this could yield a more concrete prediction by the inverse model for the action that caused the change in those two states. This hypothesis could be pronounced by the fact that the inverse loss is constantly lower for the ICMS compared to the one of ICM observed in 5.4 in both environments. Interestingly, in the Breakout environment even though the loss for the implementation with the same encoders is stagnating, ranging between the same values, the agent with the separate encoders has demonstrated a decrease of the inverse loss throughout training.

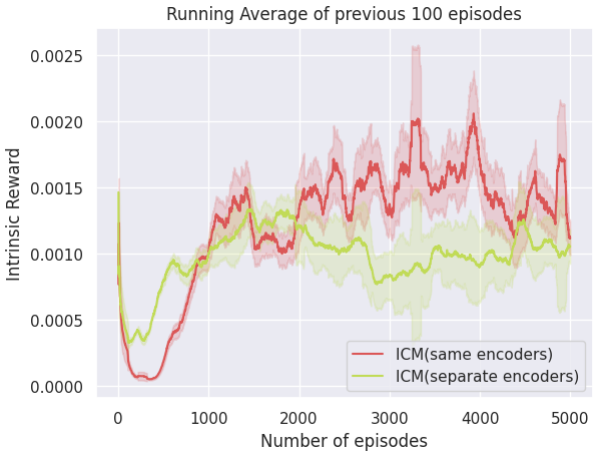
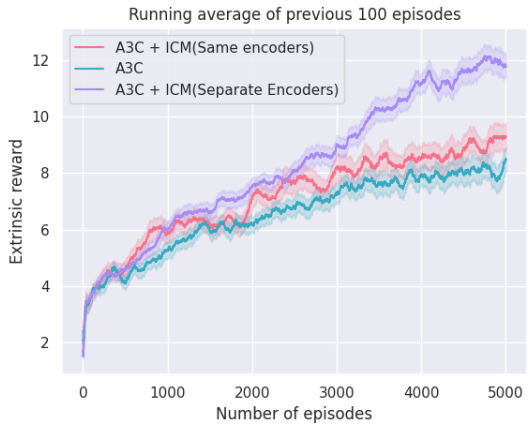


Figure 5.5: Intrinsic curiosity same vs separate

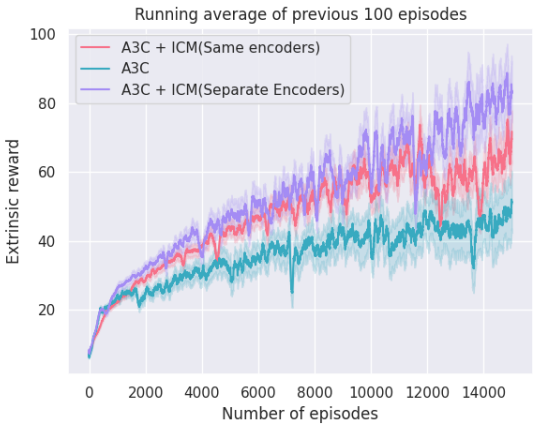
However, a stumbling block on this hypothesis is that the forward loss or intrinsic reward of the agent with the implementation of ICM with separate encoders is lower and stagnating even more than the implementation with the same encoders as demonstrated in 5.5. Intrinsic reward is the signal that guides exploration in the environment and is the reason why an agent would be able to achieve higher scores. If it is lower and stagnating, it is suggested that the agent explores less than the normal ICM. The aforementioned hypothesis would only be reinforced if the intrinsic reward of the ICMS was higher than the original ICM agent suggesting that the discrepancy between the way the two distinct encoders encode the features is introduces a more efficient way of exploration.

Figure 5.6 illustrates the three different agent implementation in one graph, once again demonstrating that the higher scores are achieved by the agent inspired by the neocortex , followed by the original curious agent implemented according to the ICM by Pathak et al [67] and lastly the A3C agent which achieves the lower scores as it learns slower. The fact that the ICMS agent is able to acquire the highest cumulative episodic reward but its high performance is unmatched by the values and pattern of the intrinsic reward is a subject that should be investigated further. In future work, the first step would be to detach the forward loss from backpropagating to the encoders so that the encoder model is not exposed to the conflicting goal of the inverse and forward model and observe performance of the ICMS relatively to the ICM and A3C + ICM. On this implementation the encoder model gradient is calculated with

respect to both inverse and forward loss, similar to training a network with multi-objectives. However, in a noisy environment backpropagating the forward loss to the encoder model will harm the exploratory behaviour of the agent as it will not model only features associated to its actions.



(a) Illustration of Breakout Environment



(b) Illustration of CartPole Environment

Figure 5.6: This figure compares the all three different policy methods: Baseline A3C, ICM +A3C , ICMS + A3C in the two different environments. It is clear that the ICM with different encoders overtakes the other two implementations.

Chapter 6

Conclusion & Future Work

In this work it has been demonstrated how intrinsic curiosity can help a reinforcement learning agent to achieve a better performance when operating in a certain environment. Intrinsic curiosity was used as an exploration approach encouraging the agent to engage with novel parts of the environment and was implemented as an Intrinsic Curiosity Module which is bolt-on part of the agent as proposed by [67]. The ICM was implemented in the project operating in a simple control environment (CartPole) following the paper of the original inventors [67] and the experimental results were analysed completing the first aim of the project. Moreover, the implementation was developed further to accommodate learning using the ICM in multiple Atari environments, that require discrete actions from the agent, from which Atari Breakout is demonstrated, thus completing another objective of the project. The results aligned with the ones of the original author with the curious agent exhibiting a better performance than an agent without curiosity in both environments.

In this thesis the two research areas of intrinsic curiosity in reinforcement learning and neuroscience are interconnected by suggesting that the ICM works in a similar way to the neocortical microcircuit, inspired by recent findings about thalamo-cortical connectivity [21]. This is a promising direction which is amplified by the results which suggests that a further investigation in this implementation could constitute a beneficial pathway for modelling intrinsic curiosity in agents to encourage exploratory behaviour. However, for the ICM to be biologically plausible some components have to be adjusted. Therefore, the implementation has been tweaked into resembling more to the structure of the neocortical microcircuit, thus completing the objective of making the implementation more biologically plausible, even though it is far from being completely biologically plausible. The experimental results of this adjustment are promising but need this implementation needs to be investigated further in future work.

Other aspects of the projects worth investigating in the future is implementing a model that is hierarchical. This implementation and bio-mapping is completely non-hierarchical while it is widely accepted that the brain processes information in a hierarchical way. Therefore, a further step would be mapping the ICM to the neocortex by taking into consideration top-down and bottom up signals, like other predictive coding models that model surprise in the neocortex employ, referred in previous section 2.3. Moreover, future work could investigate further the role of basal ganglia in modelling motion related sensory information and examine the commonalities with the inverse model, in hope to improve it, as it is deficient in modelling environment with complex temporal dynamics.

Bibliography

[1] The cartpole-v0 environment.

[2] Pytorch.

[3] Home, May 2022.

[4] Shunichi Amari et al. *The handbook of brain theory and neural networks*. MIT press, 2003.

[5] Richard Barlow and Larry Hunter. Optimum preventive maintenance policies. *Operations research*, 8(1):90–100, 1960.

[6] Andrew G Barto, Satinder Singh, Nuttapong Chentanez, et al. Intrinsically motivated learning of hierarchical collections of skills. In *Proceedings of the 3rd International Conference on Development and Learning*, pages 112–19. Piscataway, NJ, 2004.

[7] Andre M Bastos, W Martin Usrey, Rick A Adams, George R Mangun, Pascal Fries, and Karl J Friston. Canonical microcircuits for predictive coding. *Neuron*, 76(4):695–711, 2012.

[8] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

[9] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[10] Corbett Bennett, Samuel D Gale, Marina E Garrett, Melissa L Newton, Edward M Callaway, Gabe J Murphy, and Shawn R Olsen. Higher-order thalamic circuits channel parallel streams of visual information in mice. *Neuron*, 102(2):477–492, 2019.

[11] Max Bennett. An attempt at a unified theory of the neocortical microcircuit in sensory cortex. *Frontiers in neural circuits*, page 40, 2020.

[12] Vincent-Pierre Berges, Priyanka Rao, and Reid Pryzant. Reinforcement learning for atari breakout.

[13] Daniel E Berlyne. Conflict, arousal, and curiosity. 1960.

[14] J Martin Bland and Douglas G Altman. Statistics notes: measurement error. *Bmj*, 312(7047):1654, 1996.

[15] Alex Braylan, Mark Hollenbeck, Elliot Meyerson, and Risto Miikkulainen. Frame skip is a powerful parameter for learning to play atari. In *Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[16] Silvia Bucci, Antonio D'Innocente, Yujun Liao, Fabio Maria Carlucci, Barbara Caputo, and Tatiana Tommasi. Self-supervised learning across domains. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[17] Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros. Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*, 2018.

[18] Andy Clark. Whatever next? predictive brains, situated agents, and the future of cognitive science. *Behavioral and brain sciences*, 36(3):181–204, 2013.

[19] R Clay Reid and Jose-Manuel Alonso. Specificity of monosynaptic connections from thalamus to visual cortex. *Nature*, 378(6554):281–284, 1995.

[20] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[21] Christine M Constantinople and Randy M Bruno. Deep cortical layers are activated directly by thalamus. *Science*, 340(6140):1591–1594, 2013.

[22] Mihaly Csikszentmihalyi and Mihaly Csikzentmihaly. *Flow: The psychology of optimal experience*, volume 1990. Harper & Row New York, 1990.

[23] CPJ De Kock, Randy M Bruno, Hartwig Spors, and Bert Sakmann. Layer-and cell-type-specific suprathreshold stimulus representation in rat primary somatosensory cortex. *The Journal of physiology*, 581(1):139–154, 2007.

[24] William N Dember and Robert W Earl. Analysis of exploratory, manipulatory, and curiosity behaviors. *Psychological review*, 64(2):91, 1957.

[25] Leon Festinger. A theory of cognitive dissonance (row, peterson, evanston, il). *FestingerA Theory of Cognitive Dissonance1957*, 1957.

[26] Danielle M Friend and Alexxai V Kravitz. Working together: basal ganglia pathways in action selection. *Trends in neurosciences*, 37(6):301–303, 2014.

[27] Karl Friston. A theory of cortical responses. *Philosophical transactions of the Royal Society B: Biological sciences*, 360(1456):815–836, 2005.

[28] Karl Friston. The free-energy principle: a unified brain theory? *Nature reviews neuroscience*, 11(2):127–138, 2010.

[29] Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.

[30] Michael M Halassa and Sabine Kastner. Thalamic functions in distributed cognitive control. *Nature neuroscience*, 20(12):1669–1679, 2017.

[31] Michael M Halassa and S Murray Sherman. Thalamocortical circuit motifs: a general framework. *Neuron*, 103(5):762–770, 2019.

[32] Julie A Harris, Stefan Mihalas, Karla E Hirokawa, Jennifer D Whitesell, Hannah Choi, Amy Bernard, Phillip Bohn, Shiella Caldejon, Linzy Casal, Andrew Cho, et al. Hierarchical organization of cortical and thalamic connectivity. *Nature*, 575(7781):195–202, 2019.

[33] Kenneth D Harris and Thomas D Mrsic-Flogel. Cortical connectivity and sensory coding. *Nature*, 503(7474):51–58, 2013.

[34] Kenneth D Harris and Gordon MG Shepherd. The neocortical circuit: themes and variations. *Nature neuroscience*, 18(2):170–181, 2015.

[35] RJ Harvey. The extraction of features and disparities from images by a model based on the neurological organisation of the visual system. *Vision research*, 48(11):1297–1306, 2008.

[36] Jeff Hawkins and Sandra Blakeslee. *On intelligence*. Macmillan, 2004.

[37] Donald Olding Hebb. *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005.

[38] Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. *Advances in neural information processing systems*, 29, 2016.

[39] JMcVLevine Hunt. Intrinsic motivation and its role in psychological development. In *Nebraska symposium on motivation*, volume 13, pages 189–282. University of Nebraska Press, 1965.

[40] Denis Jabaudon. Fate and freedom in developing neocortical circuits. *Nature communications*, 8(1):1–9, 2017.

[41] Jon H Kaas. Evolution of columns, modules, and domains in the neocortex of primates. *Proceedings of the National Academy of Sciences*, 109(Supplement 1):10655–10660, 2012.

[42] Jerome Kagan. Motives and development. *Journal of personality and social psychology*, 22(1):51, 1972.

[43] Frederic Kaplan and Pierre-Yves Oudeyer. In search of the neural circuits of intrinsic motivation. *Frontiers in neuroscience*, 1:17, 2007.

[44] Georg B Keller and Thomas D Mrsic-Flogel. Predictive processing: a canonical cortical computation. *Neuron*, 100(2):424–435, 2018.

[45] Bruce M Koeppen and Bruce A Stanton. *Berne and levy physiology e-book*. Elsevier Health Sciences, 2017.

[46] Konrad P Körding and Daniel M Wolpert. Bayesian integration in sensorimotor learning. *Nature*, 427(6971):244–247, 2004.

[47] José L Lanciego, Natasha Luquin, and José A Obeso. Functional neuroanatomy of the basal ganglia. *Cold Spring Harbor perspectives in medicine*, 2(12):a009621, 2012.

[48] Daniel Ying-Jeh Little and Friedrich Tobias Sommer. Learning and exploration in action-perception loops. *Frontiers in neural circuits*, 7:37, 2013.

[49] Manuel Lopes, Tobias Lang, Marc Toussaint, and Pierre-Yves Oudeyer. Exploration in model-based reinforcement learning by empirically estimating learning progress. *Advances in neural information processing systems*, 25, 2012.

[50] Marlos C Machado, Marc G Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018.

[51] Shagun Maheshwari. Implementing the a3c algorithm to train an agent to play breakout!, Oct 2018.

[52] Eugenio Martinelli, Davide Polese, Francesca Dini, Roberto Paolesse, Daniel Filippini, Ingemar Lundström, and Corrado Di Natale. An investigation on the role of spike latency in an artificial olfactory system. *Frontiers in neuroengineering*, 4:16, 2011.

[53] Pedro Martinez and Simon G Sprecher. Of circuits and brains: the origin and diversification of neural architectures. *Frontiers in Ecology and Evolution*, 8:82, 2020.

[54] Ferenc Matyas, Varun Sreenivasan, Fred Marbach, Catherine Wacongne, Boglarka Barsy, Celine Mateo, Rachel Aronoff, and Carl CH Petersen. Motor control by sensory cortex. *Science*, 330(6008):1240–1243, 2010.

[55] Kathryn Merrick and Mary Lou Maher. Modelling motivation as an intrinsic reward signal for reinforcement learning agents. *Machine Learning. not yet accepted*, 2006.

[56] Demetrio Milardi, Angelo Quartarone, Alessia Bramanti, Giuseppe Anastasi, Salvatore Bertino, Gianpaolo Antonio Basile, Piero Buonasera, Giorgia Pilone, Giuseppe Celeste, Giuseppina Rizzo, et al. The cortico-basal ganglia-cerebellar network: past, present and future perspectives. *Frontiers in systems neuroscience*, page 61, 2019.

[57] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[58] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[59] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[60] LR Morcom, TJ Edwards, and LJ Richards. Chapter 14-cortical architecture, midline guidance, and tractography of 3d white matter tracts. *Axons and Brain Architecture*, pages 289–313, 2016.

[61] Vernon B Mountcastle. The columnar organization of the neocortex. *Brain: a journal of neurology*, 120(4):701–722, 1997.

[62] OpenAI. A toolkit for developing and comparing reinforcement learning algorithms.

[63] Randall C O'Reilly, Jacob L Russin, Maryam Zolfaghar, and John Rohrlich. Deep predictive learning in neocortex and pulvinar. *Journal of Cognitive Neuroscience*, 33(6):1158–1196, 2021.

[64] Georg Ostrovski, Marc G Bellemare, Aäron Oord, and Rémi Munos. Count-based exploration with neural density models. In *International conference on machine learning*, pages 2721–2730. PMLR, 2017.

[65] Pierre-Yves Oudeyer and Frederic Kaplan. What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurorobotics*, 1:6, 2009.

[66] Pierre-Yves Oudeyer, Frédéric Kaplan, Verena V Hafner, and Andrew Whyte. The playground experiment: Task-independent development of a curious robot. In *Proceedings of the AAAI spring symposium on developmental robotics*, pages 42–47. Stanford, California, 2005.

[67] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR, 2017.

[68] Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-supervised exploration via disagreement. In *International conference on machine learning*, pages 5062–5071. PMLR, 2019.

[69] Deepak Pathak, Parsa Mahmoudieh, Guanghao Luo, Pulkit Agrawal, Dian Chen, Yide Shentu, Evan Shelhamer, Jitendra Malik, Alexei A Efros, and Trevor Darrell. Zero-shot visual imitation. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 2050–2053, 2018.

[70] Leopoldo Petreanu, Tianyi Mao, Scott M Sternson, and Karel Svoboda. The subcellular organization of neocortical excitatory connections. *Nature*, 457(7233):1142–1145, 2009.

[71] Philtabor. Youtube-code-repository/reinforcementlearning/icm at master · philtabor/youtube-code-repository.

[72] Rajesh PN Rao and Dana H Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2(1):79–87, 1999.

[73] Sudharsan Ravichandiran. *Deep Reinforcement Learning with Python*. Packt Publishing, 2020.

[74] Iva Reichova and S Murray Sherman. Somatosensory corticothalamic projections: distinguishing drivers from modulators. *Journal of neurophysiology*, 92(4):2185–2197, 2004.

[75] Richard M Ryan and Edward L Deci. Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary educational psychology*, 25(1):54–67, 2000.

[76] Jürgen Schmidhuber. Curious model-building control systems. In *Proc. international joint conference on neural networks*, pages 1458–1463, 1991.

[77] Jürgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pages 222–227, 1991.

[78] S Murray Sherman and Rainer W Guillery. *Exploring the thalamus and its role in cortical function*. MIT press, 2006.

[79] S Murray Sherman and RW Guillery. On the actions that one nerve cell can have on another: distinguishing "drivers" from "modulators". *Proceedings of the National Academy of Sciences*, 95(12):7121–7126, 1998.

[80] Satinder Singh, Richard L Lewis, Andrew G Barto, and Jonathan Sorg. Intrinsically motivated reinforcement learning: An evolutionary perspective. *IEEE Transactions on Autonomous Mental Development*, 2(2):70–82, 2010.

[81] Michael W Spratling. Predictive coding as a model of response properties in cortical area v1. *Journal of neuroscience*, 30(9):3531–3543, 2010.

[82] Bradly C Stadie, Sergey Levine, and Pieter Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*, 2015.

[83] Christopher Summerfield and Floris P De Lange. Expectation in perceptual decision making: neural and computational mechanisms. *Nature Reviews Neuroscience*, 15(11):745–756, 2014.

[84] Mriganka Sur, Alessandra Angelucci, and Jitendra Sharma. Rewiring cortex: The role of patterned activity in development and plasticity of neocortical circuits. *Journal of neurobiology*, 41(1):33–43, 1999.

[85] Richard S Sutton, Andrew G Barto, et al. Introduction to reinforcement learning. 1998.

[86] Csaba Szepesvári. Algorithms for reinforcement learning. *Synthesis lectures on artificial intelligence and machine learning*, 4(1):1–103, 2010.

[87] Alex M Thomson. Neocortical layer 6, a review. *Frontiers in neuroanatomy*, 4:13, 2010.

[88] K Tsunoda. Yamane y, nishizaki m, tanifuji m. *Complex objects are represented in macaque inferotemporal cortex by the combination of feature columns. Nat Neurosci*, 4:832–838, 2001.

[89] Yoshifumi Ueta, Jaerin Sohn, Fransiscus Adrian Agahari, Sanghun Im, Yasuharu Hirai, Mariko Miyata, and Yasuo Kawaguchi. Ipsi-and contralateral corticocortical projection-dependent subcircuits in layer 2 of the rat frontal cortex. *Journal of Neurophysiology*, 122(4):1461–1472, 2019.

[90] Todd Vanderah and Douglas J Gould. *Nolte's The Human Brain E-Book: An Introduction to its Functional Anatomy*. Elsevier Health Sciences, 2020.

[91] Ilaria Vitali and Denis Jabaudon. Synaptic biology of barrel cortex circuit assembly. In *Seminars in cell & developmental biology*, volume 35, pages 156–164. Elsevier, 2014.

[92] Laurie Von Melchner, Sarah L Pallas, and Mriganka Sur. Visual behaviour mediated by retinal projections directed to the auditory pathway. *Nature*, 404(6780):871–876, 2000.

Appendix A

An Example Appendix

Content which is not central to, but may enhance the dissertation can be included in one or more appendices; examples include, but are not limited to

- lengthy mathematical proofs, numerical or graphical results which are summarised in the main body,
- sample or example calculations, and
- results of user studies or questionnaires.

Note that in line with most research conferences, the marking panel is not obliged to read such appendices. The point of including them is to serve as an additional reference if and only if the marker needs it in order to check something in the main text. For example, the marker might check a program listing in an appendix if they think the description in the main dissertation is ambiguous.