Neuro-Inspired Deep Learning

Harnessing Cognitive and Neural Principles for Advancing AI Model Architectures and World Modeling

Roy Levy

Under the Guidance of Dr. Maya Herman

December 2024

Table of Contents

1. Introduction

How can machines learn as efficiently as humans and animals?

How can we build machine architectures capable of complex reasoning, planning and problem solving, common-sense, understanding of the physical world, and capable of autonomous intelligence?

What does nature's hundred-million-year optimization algorithm have to teach us about intelligence? More fundamentally, what is intelligence?

This seminar aims to explore the nature of human intelligence and implement it in intelligent machine architectures, through knowledge of the complex operation of the human and animal brain, then combined with deep learning research.
We will first explore foundations in neuroscience, cognitive science and brain evolution that will guide us for the rest of the seminar, understanding the major mechanisms behind the human and animal brain, as well as their operation.

Then, we will research reward mechanisms and reinforcement learning in machine learning, a crucial foundation for building the higher, subsequent forms of machine intelligence mechanisms.

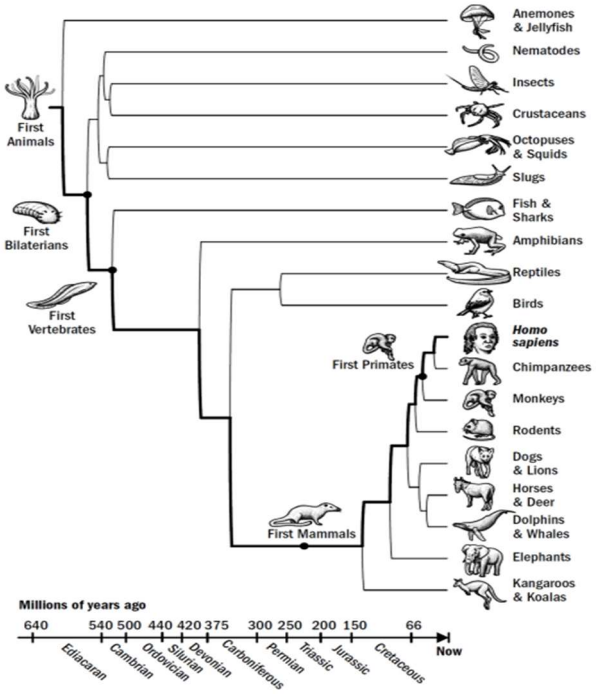
Next, this seminar will research implementations of internal simulation and world modeling in deep learning architectures, leading to a comprehensive hierarchical non-generative architecture capable of representing multiple futures in a non-deterministic manner, accounting for uncertainty and internal chaos in the world. We will discuss the limitations of current generative models (i.e., LLMs), how to solve them, as well as the bigger-picture place of language in AI models.

Lastly, we will explore a new form of intelligence in machine architectures, capable of theory-of-mind as well as thinking about one's own thinking, understanding the notion of goal and the whys, as well as learning skills from observation. This mechanism is known as mentalization.

2. Foundations: Principles and Insights from Neuroscience, Brain Evolution, and Cognitive Science

Breakthroughs in Evolution of the Human Brain:

Retracing the evolutionary steps of human brain evolution offers valuable insights to the operating mechanisms behind the human brain function and capabilities.
A model consisting of five major milestones (herein referred to as breakthroughs) in the phylogenetic origins of the human brain has been proposed [1], aiming to approximate the numerous evolved physical modifications from the first bilaterians through to the first human brains. Each breakthrough, encapsulating interconnected physical changes in the human brain structure, enabled various new adaptive behavioral applications benefiting survival and/or reproduction.



The Human Evolutionary Lineage [2]

2.1 The First Breakthrough – Steering in Early Bilaterians:

Before brains, there were simpler nerve nets, seen in the first multicellular organisms, enabling sensing of external stimuli such as food. A significant milestone with early bilaterians roughly 540 million years ago is steering – the capability of dividing external stimuli into two groups - "positive valence" (approach) and "negative valence" (avoid). Thus, the organism can seek directions where the positive valence increases and negative valence decreases, and steering away from directions of decreasing valence, climbing up or down sensory gradients. This process is

known as "taxis navigation". Such capabilities enabled adaptive applications including local area restricted search, food seeking and escaping predation, as well as maintaining homeostasis of the organism.

The model [1] identifies four physical modifications in early bilaterians that are hypothesized to have together achieved this breakthrough: i) a bilateral body structure, ii) sensory valence neurons connecting to a global neural integration center (the first "brain"), iii) associative neuroplasticity mechanisms, and iv) neuromodulatory mechanisms generating persistent behavior – utilizing neurotransmitters such as dopamine and norepinephrine (used for valence signaling). Dopamine was released in the presence of food as well as after the food cue disappeared, enabling local area restricted search even without specific food findings. The mechanisms of associative neuroplasticity (both presynaptic and postsynaptic) enabled changing of the weights and hence the valence of various stimuli.

2.2 The Second Breakthrough – Reinforcing in Early Vertebrates:

Various physical brain structure adaptations in early vertebrates established the ability of reinforcing, enabling applications such as map-based navigation, interval timing, and omission learning.

This reinforcement was a type of model-free reinforcement learning, as opposed to model-based reinforcement learning. Model-based reinforcement learning involves a constructed model of the environment and planning, which consists of the agent "playing out actions" before acting, evaluating different future outcomes. Model-free learning, in contrast, consists only of learning the direct relation of the current state and possible actions. The model-based approach is hypothesized to have evolved only later with the emergence of mammals.

Model-free reinforcement learning requires multiple features: the recognition of states, reward magnitude prediction, a temporal difference error signal (between prediction and reward), and the use of that signal to revise reward predictions.

The four main new brain structures emerging in the first vertebrates were the basal ganglia (BG), the pallium, the tectum and the cerebellum [3]. This new brain network implemented model-free reinforcement learning as follows (see Figure 1): Specific regions in the pallium developed the ability to represent an allocentric representation of the space ("spatial map"), while other to recognize patterns in external stimulus cues (early "perception"). The valence neurons from the inherited hypothalamus acted as dopamine controllers (for stimulating or inhibiting dopamine according to valence). The BG then stored stimulus sequences that tended to activate dopamine, allowing it to create a prediction for its own dopamine activation. This prediction was then subtracted from the real dopamine activation to filter out the reward prediction error signal, thereby used to reinforce learning (known as "temporal difference learning signal", see later discussion of TD learning). The BG then passed the signal to the tectum where allocentric representations from the pallium were converted to egocentric movements, creating a homing vector toward goal locations, driving movement up the dopamine gradient. Timing signals in the cerebellum (and possibly also the pallium) enabled a representation of time, allowing the agent to decide not only based on *where* it is relative to an outcome, but also *when* it is relative to an outcome, allowing it to learn when to act.

This model-free reinforcement learning therefore enabled adaptive behavioral abilities such as map-based navigation towards or away from remembered locations in the environment (and not merely taxis-navigation), as well as interval timing, and omission learning – enabling vertebrates to perform avoidance tasks based on omission of stimuli. Such learning could not have been possible without the prior inherited valence neurons and neuromodulatory signals.

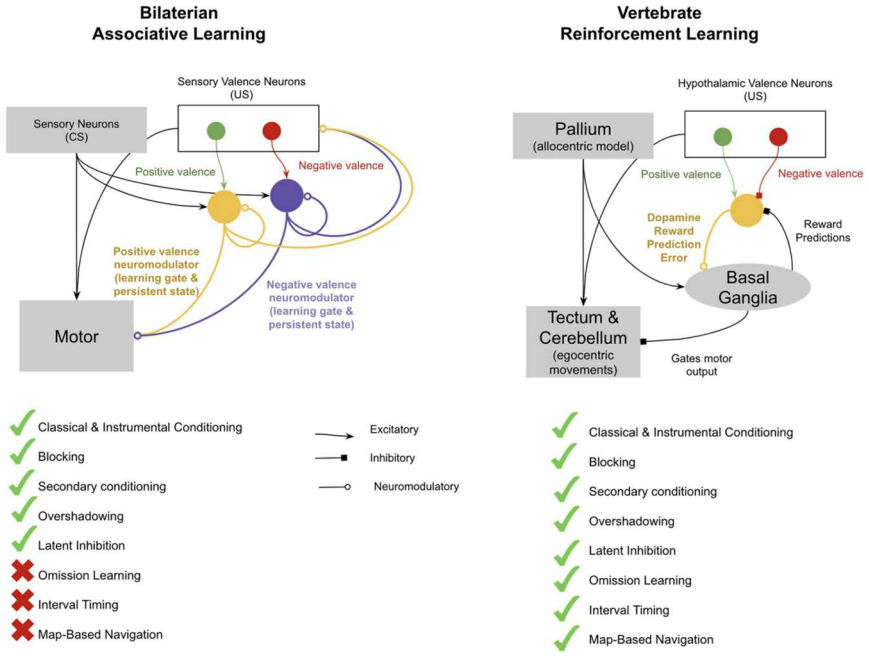


Figure 1: Difference between associative learning of early bilaterians and reinforcement learning of vertebrates [1]

2.3 The Third Breakthrough – Simulation in Early Mammals:

It has long been observed that mice navigating a maze will occasionally pause at choice points and move their head back and forth – this behavior has been called vicarious trial and error (VTE) and it has been interpreted as the mice simulating going through each option. VTE shows several interesting features - mammals perform VTE selectively when decisions are hard, such as in the early stages of solving a maze, when the difference between the outcomes is small, or when there is conflict between evidence [4].

"Simulation" simply refers to the ability of performing model-based reinforcement learning, where an animal can pause and simulate different action sequences before acting. The dorsal pallium has evolved in early mammals into the neocortex. It is proposed that the neocortex has offered the ability of internally simulating actions and stimuli, whereby an animal pauses, simulates reality, manipulates it, evaluates it and then acts accordingly. This enables new applications such as VTE (simulating paths), episodic memories (simulating past events), and counterfactual learning (simulating alternative choices).

The neocortex is composed of many repeating column-like microcircuits, purposed to act as a self-supervised "model", attempting to predict its input [5]. Its operation is akin to that of the generative class of machine learning models (e.g., Variational Autoencoders) [6][7], in which a

"latent representation" of its bottom-up input is learned. This generative model then has two modes – an "inference mode" in which it creates a latent representation that best represents its input, and a "generative mode" which given a latent representation (world model) generates its own input data. Learning occurs by minimizing the error between the generated data and the real data. Therefore, the model is "self-supervised" in the manner of being trained only by the degree of success in which its own internal model of the world predicts its own input.

The neocortex of late mammals consists of the sensory cortex and the anterior cingulate cortex (ACC) (see Figure 2). The former consists of homologous subregions that implement for each modality – visual, somatosensory, auditory – a generative model of sensory data aiming to explain its own input. The ACC performs an identical computation although for a different function. Instead of receiving input from external sensory, it receives input from the hippocampus and the amygdala, and projects onto the sensory cortex. The hypothesis is that the ACC builds a generative model of "paths" from the hippocampus, given a latent representation of "goals" from the amygdala (which consist of the actual amygdala valence results). Namely, the ACC tries to explain the sequence of places to be taken given "goals" from the amygdala. The result is therefore a latent representation model of "intent".

The function of such an "intent" model, is the ability to invoke internal simulation, enabling model-based learning. When the animal encounters a choice point where the right answer is uncertain, physically represented as several conflicting predictions in different columns of the ACC, it pauses (through top-down neural inhibition by the ACC connection to the subthalamic nucleus (STN)). The ACC then triggers simulated paths through its loop with the hippocampus and internally invokes a corresponding sensory representation ("imagines"), either through its connection with the sensory cortex, or the indirect connection through the hippocampus. During the pause, the sensory cortex switches from being externally driven ("inference mode") to being internally driven ("generative mode"), with the ACC exploring different actions consistent with its generative model of intent. Such internal representations of the world in the sensory neocortex are then evaluated in the BG (as discussed with the inherited reinforcing mechanism), and when the imagined path triggers enough dopamine by the BG, there is a "GO" response. The imagined path of the ACC is then sensitized and projected in the hippocampus, biasing subsequent actions (the latent representations in the sensory neocortex) to be consistent with the imagined path's latent representation. This may perhaps be the first version of cognitive control and "attention", through biasing certain sensory representations [8].

This "pause and simulate" mechanism applies not only to imagined action paths, but also to simulating past events (episodic memory), simulating alternative choices to choices of the past (counterfactual learning), and even working memory ("holding things in mind").

The motor cortex, which emerged in later mammals, can also be seen as using a model of intent to predict movement and trigger simulation in face of uncertainty. This is required for movements that require preplanning and fine motor skills. The motor cortex simulates actions through projection to the somatosensory cortex, just as the ACC simulates path through the general sensory cortex. However, the difference between the two is that the motor cortex gets its model of "intent" from the ACC, and simulates body movements in the somatosensory cortex, while the ACC gets it "intent" input from the amygdala and simulates general paths in the

hippocampus. This created the first motor hierarchy, with goals flowing from the ACC to the motor cortex.

Simulation enables learning not only from real world actions, but also imagined ones, and could not have occurred without two inherited features from early vertebrates, spatial mapping – needed for vicarious exploration of the environment and imagined actions with their consequences, as well as the foundation of the older basal ganglia, which could predict dopamine response with no differentiation between internally and externally evoked sensory data.

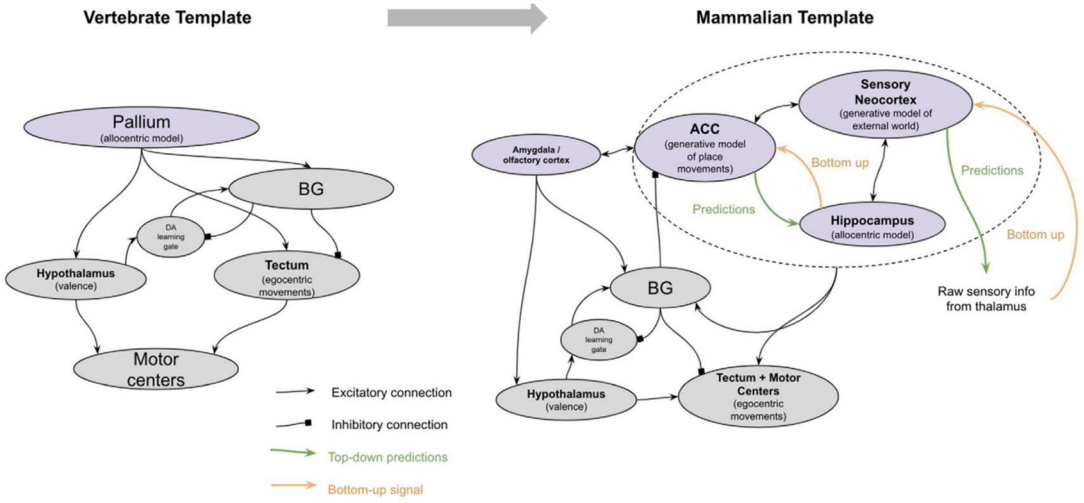


Figure 2: The major brain modifications in early mammals [1]

2.4 The Fourth Breakthrough – Mentalizing in Early Primates:

Unique brain regions that evolved in early primates established the singular breakthrough of "mentalizing", referring to the ability to construct a model of the mind, consisting of an individual's intent and knowledge. Such a model can be applied in numerous ways, three of which are anticipating future needs, theory of mind ("thinking about the thinking of others" and "thinking about one's own thinking") or metacognition, and learning by observation. For example mentalizing can be used to simulate a state of mind you do not have yet (imagine being hungry if I don't collect food right now, even though I am not currently hungry), to simulate the mind state of another creature (such as imagining how they must feel given their situation), and enables you to simulate the actions and intentions of others while watching them, which allows you to learn by observation (and not merely from one's own experience).

The two main new brain structures are the granular prefrontal cortex (gPFC) and the polysensory cortex (PSC; includes temporoparietal-junction (TPJ) and superior temporal cortex (STC)) (see Figure 3). The hypothesis is that the gPFC and PSC together implement a generative model of the ACC-sensory generative model itself, thus "explaining" the "intentions" from the ACC given "knowledge" from the PSC, effectively making it a generative model of one's own mind (the use of which is mentalizing). While the ACC-sensory network is self-supervised to predict hypothalamic and amygdala activations and therefore predicts paths that accomplish the current experienced needs, the gPFC-PSC network is self-supervised to predict the latent

representations within the ACC-sensory network, thus simulating situations and predicting what intentions *would* be selected in the ACC-sensory network *given* such situations.
Because the gPFC-PSC network is a model of what behaviors are generated from what intentions and knowledge, it can then attempt to predict what intentions and knowledge in others are consistent with their observed behaviors, thus enabling also mentalization about others.

Interestingly, given the uniform structure of the neocortical columns throughout the neocortex [4], any new function should be merely a matter of unique input and outputs (for the generative prediction model), without changing the underlying computations of the neocortical column. Consistent with this is the fact that the microcircuitry of the gPFC and the ACC-sensory system is the same with the same generative model, differing merely in where the input is received from. The ACC receives input from the hippocampus, constructing a latent model of a current intent (explanation of an observed path), while the gPFC receives input from the ACC, constructing a latent representation of a mind state (explanation of intent).
Of importance to note is that such ability as mentalizing could not have occurred without the inherited ability of earlier mammals to simulate world states.

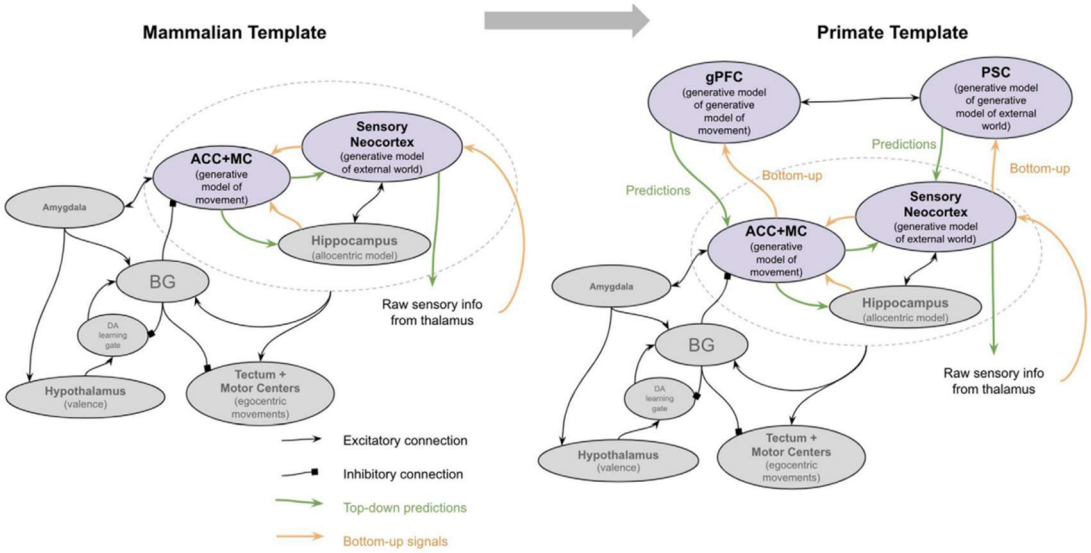


Figure 3: The major brain modifications in early primates [1]

2.5 The Fifth Breakthrough - Language in Early Humans:

Humans convey a unique ability to communicate our imaginations with symbols we create and together with the development of language was the development of writing. These symbols can therefore be chained together in sequences of any length to represent thoughts of any kind or level of complexity. And when reading or listening to language we can simulate someone else's thoughts as if we had them as well.
With language comes not only vocal communication, but more broadly rhythmic semantic communication, such as speaking, writing and music.

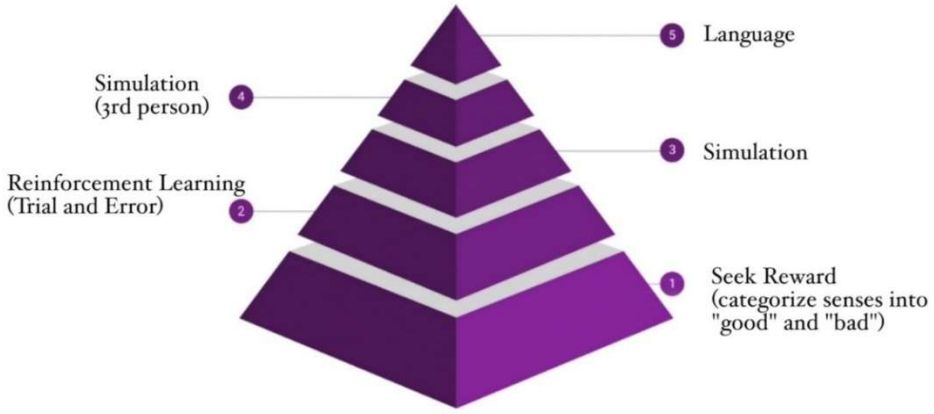
The physical modification (and a fundamental difference between primate brains and human brains) enabling such abilities is the emergent modification of the arcuate fasciculus and its connectivity with the basal ganglia (AF-BG network). It is proposed that both language and

music are two sides of the same coin and emerge from the same neural abilities. Consistently, both contain rhythmic properties, as well as a hierarchical structure (beats, within bars, within phrases, as well as phonemes within words within sentences). Both are highly "predictive" given a beginning of a familiar sequence (in the sense you cannot help but finish it).
The model theorizes that the reason for the order of these breakthroughs is that rhythmic semantic processing for communication was only possible after the ability of mentalizing, allowing the ability to infer and understand the knowledge and intent of oneself and others, as well as synchronized communication. Consistently, the AF-BG network much overlaps the "mentalizing" regions that evolved before.

Overall, the major breakthroughs in the evolution and operation of the human brain consisted first of the ability to learn from one's own actions and experience (reinforcing), then the ability to learn from one's own imagined actions (simulation), later the ability to learn from other people's actions, and finally the ability to learn from other people's imagined actions.

	REINFORCING IN EARLY BILATERIANS	**SIMULATING IN EARLY VERTEBRATES**	**MENTALIZING IN EARLY PRIMATES**	**SPEAKING IN EARLY HUMANS**
SOURCE OF LEARNING	Learning from your own actual actions	Learning from your own imagined actions	Learning from others' actual actions	Learning from others' imagined actions
WHO LEARNING FROM?	Yourself	Yourself	Others	Others
ACTION LEARNING FROM?	Actual actions	Imagined actions	Actual actions	Imagined actions

(a)



(b)

Figure 4: The Evolution of Progressively More Complex Forms of Learning (a)-(b) [2]

2.6 Applying Neuroscientific Principles in AI Model Design

Much of the success of machine learning has been based upon human brain operation – from the Perceptron model [9] and activation thresholds (for non-linear activation functions such as ReLU), through to Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), attention mechanisms (including the Transformer) and many more. It is very much sensible to learn from a model that has undergone optimization for hundreds of millions of years for the purpose of the very abilities we are trying to achieve and model – that is, the human brain.
Let us explore how some of these discussed principles of brain operation and neuroscience, such as reward mechanisms, internal simulation (using world model representations), and mentalization, may be applied to create more efficient, capable deep learning models and architectures.

Additionally, Recent breakthroughs in deep learning model design and machine intelligence have been achieved by Large Language Models (LLMs), which have learned to model language through massive corpora of textual data. These systems take the exact opposite approach to that of evolution – "shortcutting" and beginning with language. This is the big debate in the AI field – can machine learning short-circuit the steps that human intelligence took, or must it learn a more foundational experience and understanding of the world, and if so, what architecture should such models have?

3. Reward Mechanisms

Reward mechanisms are fundamental to both artificial intelligence and neuroscience, serving as the driving force behind learning and decision-making processes. The importance of reward mechanisms lies in their ability to encapsulate goals and motivations within a single signal. This aligns with the hypothesis that all attributes of intelligence (e.g., learning, perception, planning, social intelligence, generalization) can be understood as subserving the maximization of a single reward signal by an agent in its environment [10].

When intelligence-related abilities emerge as solutions to a singular goal of reward maximization, it offers a clearer explanation of why these abilities develop (e.g., classification of crocodiles is important to avoid being eaten). In contrast, understanding each ability as the solution to its own specialized goal, avoids the *why* question and instead focuses on *what* that ability does (e.g., recognizing crocodiles versus logs). Implementing abilities in service of a singular goal, rather than for their own specialized goals, also resolves the issue of how to integrate various abilities, a challenge that otherwise remains unresolved. By focusing on reward maximization, agents can develop complex behaviors that are adaptive and goal-oriented, without the need for explicit programming of each possible action or outcome. Notably, error minimization can be seen as a form of reward maximization.

Connecting to the previous neuroscience model, the concept of reward mechanisms in reinforcement learning (RL) parallels the role of reward signaling in the human brain. Neuromodulators like dopamine play a crucial role in signaling reward prediction errors - the difference between expected and received rewards – showing parallels to temporal difference RL algorithms. Brain structures such as the basal ganglia (BG) are involved in action selection and reward prediction, mirroring the function of policy evaluation and optimization in RL algorithms, as we will soon see. The hypothalamus, with its valuation of stimuli based on survival and homeostatic needs, provides a biological basis for the reward functions in RL that encapsulate the goals of an agent. By integrating these neurological mechanisms into artificial intelligence models, we can develop more sophisticated, objective driven learning that benefits from the efficiency and adaptability observed in natural intelligence.

In essence, reward mechanisms serve as a bridge between perception and action, both in artificial agents and biological organisms, utilizing hierarchical sensory processing systems. They enable the continuous improvement of behavior through feedback, supporting the development of complex skills and adaptive strategies that are essential for navigating dynamic environments.

Let us first examine basic concepts in reinforcement learning (RL), and subsequently we will explore the integration of the mechanisms for model-free reinforcing discussed in the previous (neuroscience) section. The contribution will be two-fold. First, utilization of model-free RL and reward mechanisms for various machine learning systems. Second, this will form the basis and allow the integration of the more advanced breakthrough mechanisms in the human brain, as previously discussed.

3.1 Model-free reinforcing

In classic reinforcement learning, the problem formulation involves an agent (the machine), which is then provided perception of the world through sensors. This perception of the world is called a state (s). At time t, the agent observes a state s_t, selects an action a_t and may receive a reward r_t. The goal is to learn what actions to take given a current state, also called an optimal policy $\pi(a|s)$, to maximize the cumulative (discounted) future reward $R = \Sigma_{t=0}^{\infty}\gamma^t r_t$ where $\gamma \in [0,1]$ is the discount factor (how important future rewards are relative to immediate ones).

To predict future outcomes, the evaluation function V(s), first proposed by Shannon [11], measures the expected return from a state s:

$$V(s) = E_\pi\left[\sum_{t=0}^{\infty}\gamma^t r_t \mid s_0 = s\right]$$

The Q-value function adds action-dependence:

$$Q(s,a) = E_\pi\left[\sum_{t=0}^{\infty}\gamma^t r_t \mid s_0 = s, a_0 = a\right]$$

However, in order to provide an ongoing learning signal at each step, due to it requiring many actions to receive an outcome or reward signal, a method called Temporal Difference (TD) Learning is used [12]:

$$V(s) \leftarrow V(s) + \alpha(r + \gamma V(s') - V(s))$$

Where α is the learning rate and V(s) is adjusted based on the temporal difference error $\delta = r + \gamma V(s') - V(s))$, which measures how far our current estimate V(s) is from the target value $r + \gamma V(s')$. The updating based on the estimated target value is also known as bootstrapping.

The Q-learning algorithm then extends TD learning by learning action-values instead of state-values [13]:

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

Where $Q(s,a)$, the action-value, estimates the expected cumulative reward for taking action a in state s. $\max_{a'} Q(s',a')$ indicates the best future action (for next state, s'). This equation works similarly to the TD learning equation but uses the Bellman Optimality Equation to estimate the best possible future action. Q-learning is an off-policy method because it assumes the agent follows a greedy policy.

Q-learning becomes computationally impractical when the state space is high-dimensional, such as in games like Atari or environments involving visual data. To solve this, Deep Q-Networks (DQN) [14][15] are used, in which a deep neural network is used to approximate $Q(s,a)$ (instead of the Q-table used in simpler Q-learning). DQN also introduces two key techniques: i) experience replay, in which experiences $e_t = (s_t, a_t, r_t, s_{t+1})$ are stored in a replay buffer and sampled in random minibatches for training, breaking temporal correlations of experiences and

increasing efficiency and convergence. And ii) using an additional target network $Q(s', a'; \theta^-)$ that is updates less frequently, stabilizing the learning process.

Two additional improvements to DQN are Double DQN which solves an overestimation bias of the best future action by decoupling the action selection and evaluation, and Dueling DQN which adds an advantage function A(s, a) for improved policy evaluation in environments with many similar-valued actions [16][17]. Rainbow DQN combines several improvements of DQN such as these into a unified architecture [18].

The main problem with the methods previously discussed is that they are suitable for discrete action spaces, not large, continuous action spaces. A solution to this problem is the policy gradient approach, in which a state would be provided as input, but the output would be the probability across all actions, known as an action distribution [19]. This is achieved by, instead of learning value functions, directly optimizing the policy, represented using neural networks by function approximation (and updated using gradient ascent). This enables scalability for complex environments. However, because learning is done across a distribution of all actions, this generally requires much more experience to learn effectively. Because of this need, most of the initial progress was in simulated robotics, due to the ability for time speed up in computer simulations, gaining years of experience in hours. Unfortunately, models trained in simulation failed in transferring to the real world in complex tasks, this being due to the unpredictability and complexity of the real world, and even then, training being expensive and slow. Initially, solutions have been proposed such as using a second value function (critic) to speed up learning (inspired by TD learning but with two networks) – known as actor-critic methods [20]. The more recent Soft Actor-Critic incorporates a maximum entropy framework in addition to reward maximization in order to balance exploration and exploitation, analogous to uncertainty handling in the brain [21][22].

A critical realization was, instead of using more accurate simulations for learning, using *less* accurate simulations, thus training a model in purposefully messy environments by randomizing all key aspects of the environment. This was demonstrated by OpenAI when training a physically complex hand to learn dexterous manipulation of a cube [23]. Interestingly, the researchers also noted several naturally emergent behaviors ("physical intelligence") also found in human manipulation. Similar findings were identified in Google DeepMind's humanoid robots, trained with direct policy methods, in simulation with domain randomization to play soccer, using "scoring" as a reward [24]. Results showed complex emergent behavior that humans learn, such as anticipating ball movements and blocking opponent shots before they happen. A key next step in their work is including in-life learning allowing for continuous improvement, thus proposing the key question of how much must be learned in simulation, versus the wild?

3.2 Implementing reinforcing in deep learning architectures

Now that we are equipped with the tools of reinforcement learning, let us integrate the mechanisms of reinforcing in the human brain, originating in the brains of early vertebrae. Neurological mechanisms inherited from bilaterians, as discussed, include neuromodulatory signals (i.e., dopamine) and stimuli valence neurons in the hypothalamus, providing a *reward signal*

learned to represent various goals of the agent (this will later come useful for the amygdala's role in simulation and later mentalization).

As mentioned previously, model-free reinforcement learning involves multiple features - the recognition of states (akin to the pallium in vertebrae, originating in inherited sensory valence neurons, and in mammals this will evolve into parts of the sensory neocortex), reward magnitude prediction (akin to the BG's role), a temporal difference error signal (between prediction and reward), and the use of that signal to revise reward predictions (train the BG, or optimize the policy).

The pallium, which creates an allocentric representation of space as well as recognition of patterns of stimulus cues, is the perception of the model, and such a perception module in a deep learning architecture could be implemented using analogous deep learning layers such as CNNs (as demonstrated by [15], handling high dimensional input), or Vision Transformers and attention layers (which also play the important role of human attention, filtering unimportant inputs and focusing on processing salient ones, aiding efficiency) [25][26]. However, for the sake of conciseness, this will remain out of this seminar's scope.

The BG are responsible for predicting the reward for recognized patterns and action sequences (or rather state and action evaluation), and they are the ones being trained. Therefore, the BG is a *policy,* as well as *value* function (or critic). The temporal difference error signal used to train the BG is therefore like that of TD learning and in the other learning signals discussed for policy optimization. This ability of policy learning using reinforcing can be implemented using the methods discussed, such as Rainbow DQN and training in domain randomized simulated environments for efficiency (an advantage of computers over animal brains; however, of importance to note is that this is not internal simulation, but rather model-free reinforcing merely using a computer-generated outer environment). After the action is generated using the policy implementation, disinhibiting neurons in a separate egocentric action module (analogous to the tectum) allows for converting the allocentric representation into egocentric actions or movements, thus climbing the reward gradient (see previous chapter for exact details). Proposed architectures such as [27] offer methods for parallelizing deep reinforcement learning components such as in DQN, allowing increased efficiency and performance.

Additionally, adding temporal signals in the state and environment perception, akin to the timing signals in the cerebellum, will allow for interval timing as discussed previously.

However, an unanswered question remains. The RL reward signals often used are straightforward, such as scoring in a game or finding food in animals, therefore designed for very specific problem solving. For deep learning models with specifically defined tasks (e.g. winning a game, minimizing specific target prediction error), manual definition of the reward could still be viable and straightforward. What then will be the reward signal when implementing the ability for reinforcing in more general-purpose deep learning architectures? Well, for task or query targeted models, a reward module could be added (akin to the hypothalamus whose valence weights of stimuli represented and determined the reward), using a neural network (NN) for representing the reward signal, given a state (state value function estimation). An encoder could be used to encode the query or task into a latent representation of reward signal, then used to drive reinforcing [7]. Such latent reward representation could be then computed into a reward

scalar given a state, for example, by using a dot product (or cosine similarity) of encoded reward signal and encoded state or action, similar to query-value multiplication in an attention layer given a hidden state [26]. However, there may be value to a non-scalar, multi-dimensional or distributional, reward signal, and there is evidence to support that this may be the case in the brain, as proposed by [28].

As an example, cosine similarity of embeddings (dot product of vectors divided by norms) has been successfully used as a reward for the task of image captioning [29]. This consists of the input sentence embedding - generated using the last hidden state of an RNN - and input image embedding – extracted using a CNN and linear mapping function to embedding space:

$$r = \frac{f_e(v) \cdot h'_T(S)}{||f_e(v)||\,||h'_T||}$$

Where r is the reward, $f_e(v)$ is the embedding-mapped CNN-extracted feature vector of the image, and $h'_T(S)$ is the last hidden state of sentence from RNN [29].

A different approach to implementing model-free reinforcing in deep learning architectures may draw upon the simplicity and scalability of the transformer architecture and associated advances in language modeling such as GPT and BERT [26]. The Decision Transformer as well as the Trajectory Transformer architectures, frame RL as a sequence modeling problem using transformer architectures [30][31]. Instead of relying on traditional methods like value functions or policy gradients, a distribution of trajectories is modeled by treating the sequence of returns, states, and actions similarly to language modeling tasks. This approach is consistent with cognitive processes in the brain, particularly the role of the prefrontal cortex in planning and attention, with the Transformer's self-attention mechanism enabling focused predictions on relevant input sequences.
This idea is consistent with the observation that, even in physical intelligence, an action is simply a sequence of numbers sent into a motor. Therefore, an "action transformer" may be constructed as a word-for-action problem, with next action prediction. However, as we will soon see, using a reward mechanism for reinforcing may be an essential foundation for many additional abilities such as simulation.

3.3 Laying ground for internal simulation

As in the neurological analogy, the reward mechanisms are essential in laying ground for simulating and vicarious learning (as well as the identified emergent behaviors). Without a spatial map (or perception space) it would be impossible to simulate various movements and actions. Additionally, the policy representation (akin to BG) is needed for it to be trained vicariously. Lastly, in internal simulation, the amygdala played the role of providing the "goals" to the ACC generative model of intent. The goals were not a complex representation of the objects or sensory stimulus, but rather the actual valence weights and results based in the amygdala and hypothalamus (i.e., reward module) [1].

I therefore hypothesize that, as with the brain, in order to create a similar representation of the model's goals (for simulation) in deep learning, the representation of reward signal used herein

for reinforcing could suffice, as the reward valence results themselves encapsulate the very goals of the model. As previously explained, for task-specific models, in which the reward signal can be directly represented, the goal representation is straightforward. Otherwise, latent representation of reward used (generated by the encoder based on the task) may suffice.

3.4 Further ideas

An additional aspect of the brain that builds upon reward mechanisms, which will remain out of the scope of this seminar but is nonetheless worthy of mentioning, is associative plasticity mechanisms (presynaptic and postsynaptic), in the form of reward-modulated Hebbian learning. In this mechanism, synaptic changes are influenced by both pre- and post-synaptic activity, modulated by a global reward signal [32]:

$$\Delta w_{ij} = \eta \cdot \delta \cdot x_i \cdot y_j$$

Where: Δw_{ij} - change in synaptic weight, η – learning rate, δ – reward prediction error (mismatch between expected and actual outcomes), and x_{ij}, y_{ij} are the activities of pre- and post-synaptic neurons.

An advanced extension that builds upon Hebbian learning is *differentiable plasticity*. This approach allows both synaptic weights and their plasticity rules to be learned and optimized through gradient descent (i.e., the plasticity itself is trainable), enhancing the brain's ability for adaptive and continuous learning [33].

4. Internal Simulation and World Models

How exactly can planning be defined? How can machine intelligence architectures perform complex reasoning and planning, exhibiting "common sense", understanding how the world works?
Planning may be seen as a form of prediction – estimating missing information that cannot be inferred from perception, predicting future states as a result of current state and an action sequence. This may be achieved using a world model for internal simulation. A world model can be defined as a system that, given an observation $x(t)$, a previous estimate of the world state $s(t)$, an action proposal $a(t)$, and possibly a latent variable $z(t)$, computes:
A representation $h(t) = Enc(x(t))$ and a prediction $s(t+1) = Pred(h(t), s(t), z(t), a(t))$, where $Enc()$ is a trainable deterministic encoder mapping sensory input into a compact latent representation, $Pred()$ is a trainable predictor, and the latent variable $z(t)$ sampled from a distribution represents unknown information, parameterizing the set of plausible predictions. Such a world model allows, as we have seen, internal simulation, vicarious planning and learning (reducing the need for real world expensive and potentially dangerous trial and search), counterfactual learning as well as a form of episodic memory, thereby forming an understanding of how the world works, desirably in multiple levels of abstraction. World models and internal simulation may be necessary for enabling complex reasoning and planning in machine intelligence architectures, the type of fundamental "common sense" that current, arguably "brittle", architectures (e.g., LLMs) do not exhibit.

4.1 Existing approaches in machine learning

One of the foundational architectures incorporating internal simulation and world modeling in machine systems (model-based reinforcement learning), which formed the basis for many later architectures, is the Dyna architecture [34]. Dyna integrates learning, planning and acting by learning not only from real-world interactions with the environment but also from simulated experiences generated by a learned model of the world. The architecture consists of two main components – a model-free learning algorithm (e.g., DQN), as discussed in the previous chapter, and a learned model of the environment's dynamics ("action model"), a model that takes as input a state and action and outputs a prediction of the next state. The agent updates its value function based on real experiences and simultaneously learns a model $P(s', r \mid s, a)$ that predicts the next state s' and reward r' given the current state s and action a. It then performs **planning** – finding the optimal policy given the action model – by generating simulated trajectories using the learned model, effectively "imagining" future states to train its value function further. Such an approach reduces reliance on real-world learning (a major problem in standard reinforcement learning), thus improving sample efficiency [35] (akin to experience replay in DQN), and additionally enables the agent to anticipate and plan future events.

The idea of internal simulation is further exemplified in AlphaGo, developed by DeepMind, which combines deep neural networks with Monte Carlo Tree Search (MCTS) to master the

game of Go [36]. AlphaGo uses two neural networks – a policy network that predicts the probability distribution over possible moves and a value network that estimates state value. During play, MCTS constructs a search tree by simulating possible future move sequences, guided by the policy network to prioritize promising moves and the value network to evaluate leaf nodes. The policy network was first trained on a library of human games using supervised learning, which then guided MCTS simulations to train the value network via self-play.

This work was then followed by AlphaGo Zero, which wasn't trained on human games, instead learning entirely from self-play [37]. Surprisingly, the performance beat its predecessor – showing that the human play supervised learning decreased performance, since it limited the exploration of the space of possible game play styles.

Further generalizing this approach, AlphaZero extends this architecture to additional games such as chess and shogi. The key idea was that AlphaZero could be provided as input a world model of a game to learn (i.e., the game rules), and it achieved superhuman performance through self-play [38].

The previous methods, however, do not involve learning a world model, but instead receive as input a fixed world model. A key advancement was made in **Ha and Schmidhuber's World Models** framework, which involves learning a compact latent representation of the environment and then predicting future latent states using a constructed model of the world [39]. The architecture consists of three components – the vision, memory, and controller models.
The vision (V) model, typically a variational autoencoder (VAE), which compresses high dimensional sensory observations into a latent space, thus learning an abstract simple representation of input frames. VAEs are generative encoder-decoder models that encode input data into a latent probability distribution and then decode samples from the distribution to reconstruct the original data, optimized by reconstruction loss and Kullback-Leibler divergence (a type of statistical distance) [7].
The memory (M) model is an RNN (Recurrent Neural Network), typically an LSTM [40], which predicts the next future latent state – the latent vector z_{t+1} produced by V - based on the current latent state and action. Since many complex environments are stochastic, the LSTM is trained to produce a probability density function p(z) instead of a deterministic prediction z, with p(z) approximated as a mixture of Gaussian distributions (an approach known as a Mixture Density Network with RNN, or MDN-RNN). The RNN therefore models $P(z_{t+1} | z_t, a_t, h_t)$ where a_t is the action at time t and h_t is the hidden state of the RNN at time t. Additionally, a temperature parameter τ can be adjusted to control uncertainty, which is useful in order to prevent "cheating the world model" – a problem in which the controller undesirably exploits inaccuracies of the model (caused by the model being a latent "simplification" of the environment) instead of learning (which may then transfer poorly to real world testing).

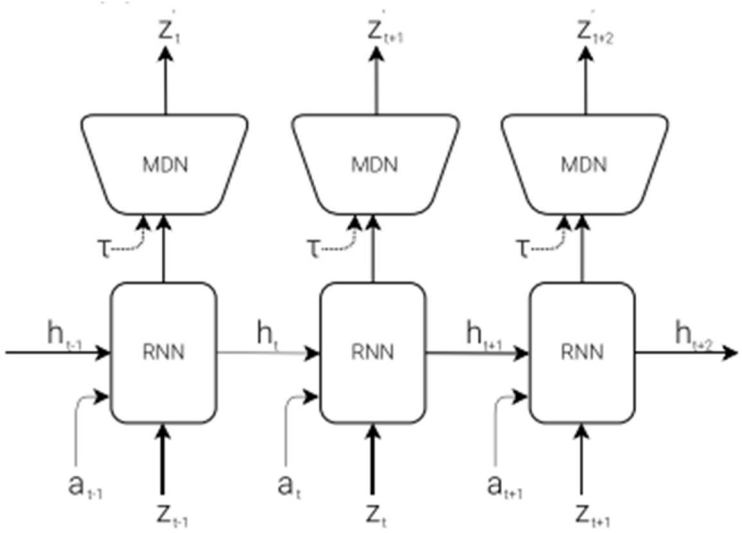


Figure 5: RNN with a Mixture Density Network output layer. The MDN outputs the parameters of a mixture of Gaussian distribution used to sample a prediction of the next latent vector z. [39]

Lastly is the controller (C), which determines which actions to take to maximize expected cumulative reward (the policy), composed of a single linear network in order to retain simplicity and allow quick learning, thus allowing most of the complexity to reside in the world model. It is optimized using evolutionary strategies (i.e., CMA-ES).

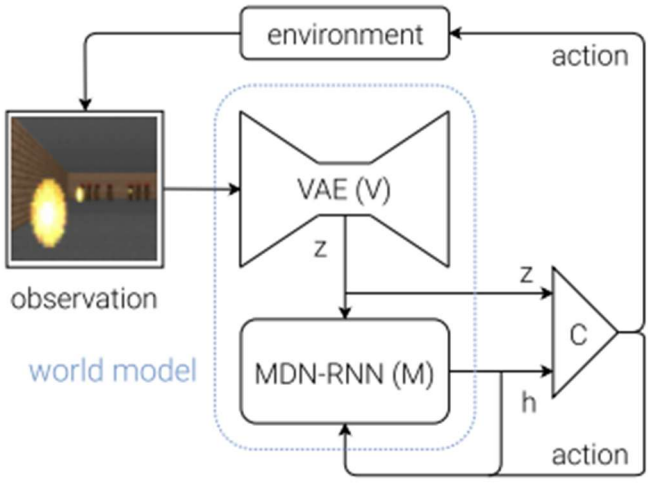


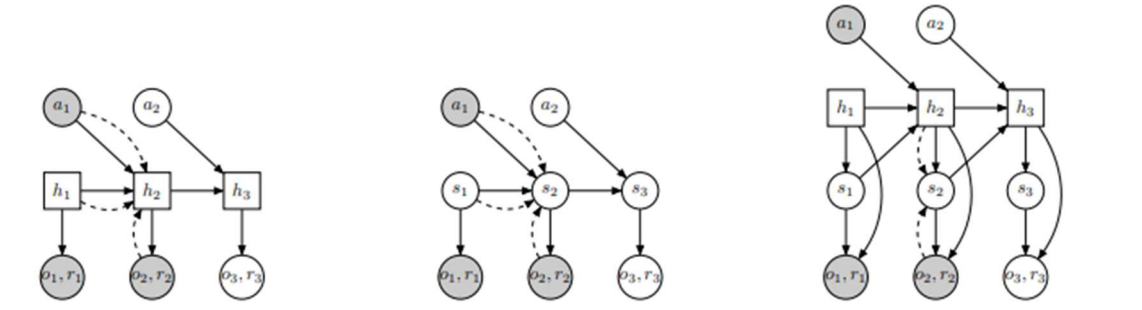
Figure 6: Flow diagram of the agent architecture [39]

Notably, Ha and Schmidhuber have demonstrated that this architecture can be trained entirely within its own "imagined" version of the game generated by its world model, and then transfer to real world policy [39] (therefore enabling much faster learning through simulated experiences). The shortcoming of this framework, however, is that the world model is trained independently of the action selection network and therefore does not focus on task-relevant features, for example it may reproduce irrelevant features such as brick patterns on a wall. Thus, such

separation between world modeling and action selection means the model is not optimized for solving the task at hand.

MuZero (developed by DeepMind) addresses such limitations and expands on AlphaZero by combining model-based and model-free RL to learn not the full state of the environment but only task relevant information necessary for the policy and valuation functions [41]. The system consists of three components:
A representation function that encodes the observation into a hidden state (abstract representation of the current state). As in previous models, a second prediction function that, given the hidden state, predicts the policy (next action) and value (position evaluation). Lastly, a third world model ("dynamics function"), given an action (from the prediction function) and hidden state, predicted the next hidden state and immediate reward (short term consequences). This next predicted state can then be fed back into the prediction function to produce a value estimation, thus providing a learning signal (long term consequences). Instead of attempting to reconstruct the entirety of the original observation, MuZero focuses on learning a latent state representation sufficient for planning and decision-making using MCTS. This allowed it to 'dream' down branches of any MCTS using a learned world model, outperforming previous methods in various domains, even shown by researchers to compress videos efficiently [42].

Another method building upon Ha and Schmidhuber's work is **PlaNet** (Deep Planning Network), a model-based architecture for continuous control tasks that learns world dynamics from raw pixels and utilizes efficient online model-predictive control (MPC) in a compact latent space [43]. The planning thus involves optimizing an action sequence in an online method, in contrast to the offline learning in previous methods (which is a parametric policy), thus allowing dynamic adaptation to new, previously unseen scenarios.
PlaNet utilizes a stochastic recurrent state-space model (RSSM; see Figure 7), which combines a deterministic recurrent model (h_t) for long-term pattern capturing together with a stochastic latent variable (s_t) to model uncertainty and partial observability. This approach was shown to predict multi-step trajectories more robustly than purely deterministic (which fail to capture multiple futures and make it easier for the planner to exploit inaccuracies) or purely stochastic models (which fail to remember long-term pattern). The RSSM consists of an encoder that turns image observations into latent states s_t, a transition model $P(h_{t+1} | h_t, s_t, a_t)$ to predict next latent state (a_t is an action), and a decoder used during training for a meaningful latent space. PlaNet also introduces a new latent overshooting objective for both one-step and multi-step prediction without requiring observation reconstruction during planning.



(a) Deterministic model (RNN) (b) Stochastic model (SSM) (c) Recurrent state-space model (RSSM)

Figure 7: Latent dynamics model designs. In this example, the model observes the first two timesteps and predicts the third. Circles represent stochastic variables and squares deterministic variables. Solid lines denote the generative process and dashed lines the inference model. (a) Transitions in a recurrent neural network are purely deterministic. This prevents the model from capturing multiple futures and makes it easy for the planner to exploit inaccuracies. (b) Transitions in a state-space model are purely stochastic. This makes it difficult to remember information over multiple time steps. (c) PlaNet splits the state into stochastic and deterministic parts, allowing the model to robustly learn to predict multiple futures. [43]

Previous model-based agents typically select actions either by planning through many model predictions or by using the world model as a simulator to reuse existing model-free techniques. Both designs are computationally demanding and do not fully leverage the learned world model. Moreover, world models can be shortsighted if they use a finite imagination horizon.
Therefore to overcome these limitations, and building upon PlaNet, the **Dreamer** architecture learns a world model and then leverages it to learn long-horizon behaviors by **latent imagination**, which involves using gradient-based policy optimization through imagined trajectories in latent space [44]
Dreamer consists of three processes, which can be executed in parallel (see Figure 7) – learning the world model (which is based on the RSSM used in PlaNet), learning behaviors based on predictions from the world model, and implementing the learned behaviors in the environment to collect new experiences. To learn behaviors, it uses an actor-critic approach, optimizing the policy (actor network) to maximize expected reward and its value network (critic) to estimate long-term rewards beyond the planning horizon (thus addressing the limitations of finite planning horizons).

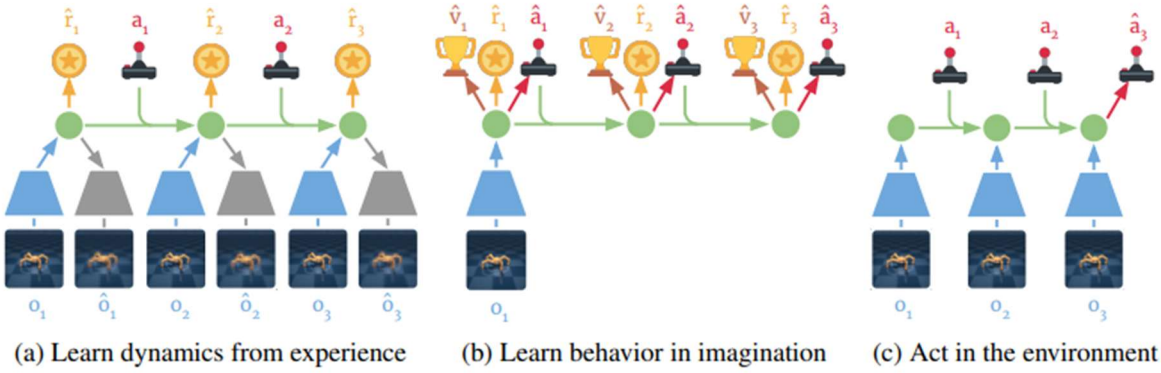


Figure 8: Components of Dreamer. (a) From the dataset of past experiences, the agent learns to encode observations and actions into compact latent states (green circles), for example via reconstruction, and predicts environment rewards $\hat{r}$. (b) In the compact latent space, Dreamer predicts state values $\hat{v}$ and actions a that maximize future value predictions by propagating gradients back through imagined trajectories. (c) The agent encodes the history of the episode to compute the current model state and predict the next action to execute in the environment. [44]

Imagined trajectories allow Dreamer to **directly compute gradients** of future returns with respect to the actions, value functions, and rewards predicted by the world model, thus removing the need for derivative-free optimization (as in 'World Models') or extensive online planning (as in PlaNet). This tells Dreamer how small changes to its actions affect what rewards are predicted in the future, allowing it to refine the actor network in the direction that increases the rewards the most (we will later see this idea in JEPA as well). Such an architecture allows not only greater

computational efficiency but also better generalization to unseen scenarios, with the ability to predict outcomes directly in latent space allowing parallel simulation of thousands of trajectories on a single GPU. As such, Dreamer outperforms previously best model-based and model-free methods in terms of data-efficiency, computation time and final performance (see Figure 9).

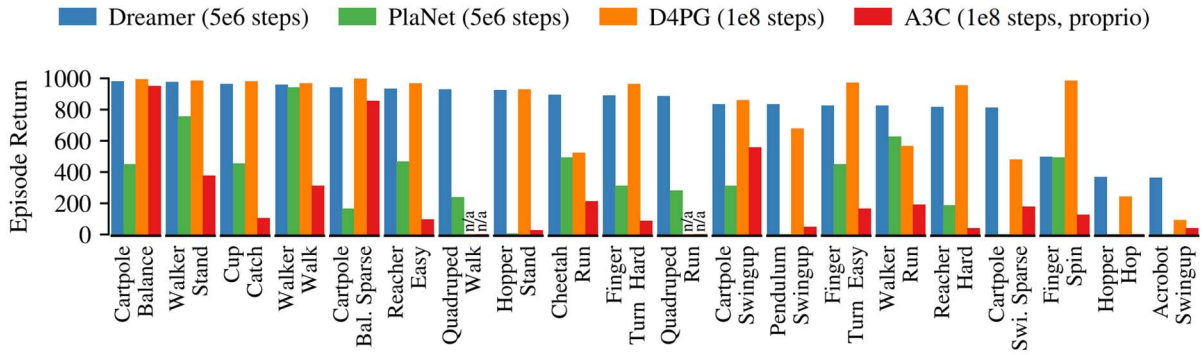


Figure 9: Performance comparison to existing methods. Dreamer inherits the data-efficiency of PlaNet while exceeding the asymptotic performance of the best model-free agents. After 5 × 106 environment steps, Dreamer reaches an average performance of 823 across tasks, compared to PlaNet at 332 and the top model-free D4PG agent at 786 after 108 steps. Results are averages over 5 seeds. [44]

There have been several improvements made to the Dreamer architecture, including:

- DreamerV2 extends the framework to handle discrete latent variables, enabling it to achieve human-level performance on tasks like Atari, significantly more efficiently than MuZero [45].
- Director builds upon Dreamer and DreamerV2 by introducing hierarchical planning, in which a high-level policy creates latent goals (abstract subgoals; using goal autoencoders) and a low-level policy learns to achieve them using primitive actions. This enables complex, temporally extended plans as well as effectiveness in long-horizon tasks with very sparse rewards, learning successfully in a wide range of domains [46]. We will later see a similar idea in JEPA as well.
- DayDreamer applies the Dreamer framework to physical robot learning, enabling online learning and adaptation to new tasks without needing any pre-training. It demonstrates capabilities such as learning to roll and walk in quadrupeds within an hour (while withstanding pushes and perturbations), object manipulation tasks in robotic arms and wheeled robot navigation tasks, all using the same hyperparameters [47].
- DreamerV3 further generalizes the Dreamer architecture to an even wider range of domains (such as collecting diamonds in Minecraft from scratch, a task posed as a significant challenge in AI), with improvements in stability and scalability. The architectural improvements include optimizing the training to handle both discrete and continuous actions, refining the loss function and regularization to improve stability and gradient propagation [48].
- Lastly, DynaLang interestingly builds upon DreamerV3 by integrating natural language understanding into the architecture and world model, enabling processing and acting on textual instructions, descriptions and questions in addition to visual data [49]. This is achieved by a language-conditioned multimodal world model (predicting future latent

states, which include both visual and linguistic cues). The representation module encodes both visual observations and language tokens into a unified stochastic latent representation (allowing joint reasoning across modalities) using a variational autoencoding objective (ensures meaningful and efficient representation) [7]. The representation learning loss used is therefore the sum of (whereby CCEL means categorical cross-entropy loss):
An Image loss: $L_x = \|\hat{x}_t - x_t\|$
A Language loss: $L_l = CCEL(\hat{l}_t - l_t)$
As well as reward loss, continuation loss (whether episode continues), and a regularization term.
Additionally, DynaLang shows such a model of language and vision enables capabilities such as text generation and pretraining on text-only and video-only datasets without actions or rewards.

- It should be mentioned that there have also been world models implemented using Transformers as sequence models instead of the RNN in the RSSM [50][51].

4.2 Further abilities

Models with internal simulation mechanisms and world models may enable further capabilities and applications beyond those discussed in this seminar – scientific simulations and exploration, as well as building creativity into a model – which is in essence a combinatorial ability, as well as complex motor task planning – all of which deserve further exploration and whose nature is closely related to the mechanisms discussed above.

Another architecture implementing internal simulation and world modeling is the JEPA architecture, whose operation closely aligns with the neuroscientific and cognitive principles discussed in the neuroscience chapter as well as the ideas in the current chapter.

5. Joint Embedding Predictive Architecture (JEPA) and the Role of Language in AI Models

How can machines learn as efficiently as humans and animals? How can machines learn perceptual representations at multiple levels of abstraction, enabling planning and predicting in multiple time scales?

In Yann LeCun's words: "We humans give way too much importance to language and symbols as the substrate of intelligence. Primates, dogs, cats, crows, parrots, octopi, and many other animals don't have human-like languages, yet exhibit intelligent behavior beyond that of our best AI systems. What they do have is an ability to learn powerful "world models" that allow them to predict the consequences of their actions and to search for and plan actions to achieve a goal. The ability to learn such world models is what's missing from AI systems today."

These models operate at multiple levels of abstraction, enabling reasoning, planning, and action over diverse time horizons, learning new skills with very few trials. Common sense knowledge allows not only future outcome prediction, but filling in missing information, whether temporally or spatially.

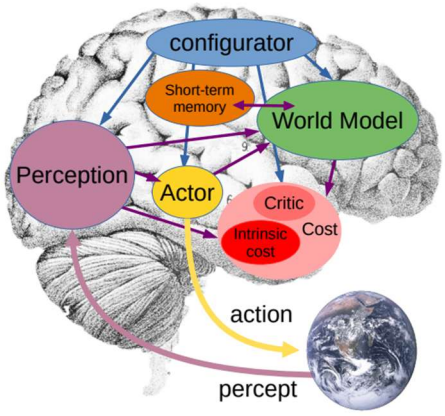


Figure 10: The system architecture for autonomous machine intelligence [52]

5.1 Overall architecture

The proposed architecture consists of the following modules (see Figure 10), all of which are differentiable [52]:
- The perception module produces a latent representation of the current state of the world from sensory observations (e.g. visual, auditory), which may be of hierarchical fashion with multiple levels of abstraction.
- The world model module predicts future states of the world as a result of a proposed sequence of actions, as well as natural evolutions of the world and estimation of missing information not provided by perception. It can predict multiple possible future world states parameterized by latent variables representing uncertainty about the world state, accounting for the fact the world is not entirely predictable, due to intrinsic stochasticity (aleatoric uncertainty) or partial

observability (epistemic uncertainty), especially more so with adversarial agents.
- The cost module computes a scalar "energy" value that represents the level of discomfort of the current or predicted state and is ultimately to be minimized. It consists of two sub-components – the intrinsic cost, which is immutable (not trainable) and computes the immediate energy of the current state (pain, pleasure, hunger, etc.; akin to the hypothalamus and amygdala), and the trainable critic which predicts future internal cost values (akin to the basal ganglia). In a robot, the intrinsic cost may include measurements such as external force overloads, low energy reserves, as well as basic drives such as standing up, finding human interactions and praise rewarding, or curiosity to maximize diversity of experience for the world model to train on.
- The actor module computes action sequence proposals optimized to minimize future costs. It consists of two components - a policy module for direct action based on the world state estimate of the perception and short-term memory, and an action optimizer. The latter consists of the actor proposing an action sequence, the world model predicting the future states later fed into the cost (which defines the goals) which estimates future energy associated with the action sequence. This action sequence can then be optimized to minimize the energy using gradient based methods, or alternatively dynamic programming or sparse search techniques (such as tree search with pruning) if the action space is discrete. These modes are akin to Kahneman's "System 1" and "System 2", as well as Dyna's reactive behavior and planning – Mode 1 involves direct action and no complex reasoning while Mode 2 involves reasoning and planning through the world model and the cost [53]. Reasoning here refers to constraint satisfaction through energy minimization.
- The short-term memory module stores past, current and future world states as well as their corresponding intrinsic cost, accessed by querying (implemented with a Key-Value associative memory search) updated and used by the world model and critic both for computation and for training (as in Dreamer and variants). It plays a role similar to the hippocampus in mammals.
- The configurator module receives input from all modules and configures them for the task at hand by modulating their parameters and attention circuits, particularly the perception, world model and cost modules.

5.2 Perception-Action Loop: Mode-1 and Mode-2 Operation

As described, Mode-1 reactive behavior (see Figure 11) consists of the perception extracting a latent representation of the state of the world through an encoder, $s[0] = Enc(x)$. The actor then directly computes an action through the learned policy module, $a[0] = A(s)$. This process does not make use of the world model nor the cost. The cost module computes the energy of the state and stores the pair in short-term memory. The next state and cost may be computed in order to train the world model once the next observation becomes available.

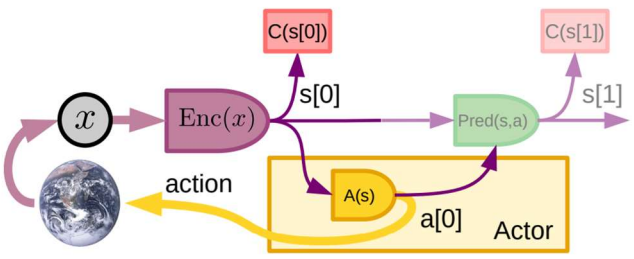


Figure 11: Mode-1 Perception-Action episode [52]

With the use of the world model, the agent can imagine courses of action and predict their effects, reducing the need for expensive and potentially dangerous search for good actions and policies through attempting multiple actions in the external world.
Mode-2 perception-action operation (see Figure 12) involves **perception** producing the current world state representation, and then an iteration of an **action sequence proposal** $(a[0], \dots, a[T])$ to be evaluated, **simulation** in which the world model predicts one or several likely sequences of resulting world states, **evaluation** by the cost module of the total cost from the predicted state sequence, which may be a discounted sum of estimated energy over time steps, **planning** in which the actor proposes a new action sequence with lower cost by gradient propagation of through the cost to the action variables. After converging on a low-cost sequence, the first actions are performed, and lastly the action state and cost are stored in short-term memory. This process is essentially model-predictive control, but with a learned world model and cost function.

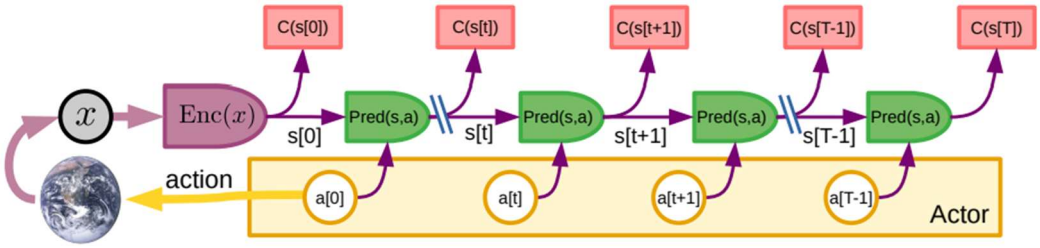


Figure 12: Mode-2 Perception-Action episode [52]

The new skills learned using mode-2 can be distilled into the mode-1 reactive policy by training it to minimize divergence from the mode-2 optimal sequence found, a process referred to as amortized inference, therefore allowing the agent to effectively learn the new skill, no longer requiring careful planning (utilizing the speed and efficiency of mode-1 policy). Once the policy is sufficiently trained it can be used before mode-2 inference to initialize on "good" action sequences.
This process can be seen as a form of reasoning based on *simulation* utilizing the world model and on *optimization* of the energy with respect to action sequences. Such type of planning through simulation and optimization may comprise the most frequent form of intelligence in natural intelligence and, as we have seen, involves a process similar to that which enables internal simulation in mammalian brains, along with the associated intelligent abilities discussed.

5.3 World model designing and training

The world model is arguably the most complex component in the architecture. It must enable multiple plausible outcomes to handle uncertainty, and enable long term prediction and planning through high-level abstract representations of the world and complex goals then decomposed into subgoals and lower-level actions. The world model will be built using the Joint-Embedding Predictive Architecture (JEPA).

Self-Supervised Learning and Energy-Based Models:

Self-Supervised Learning (SSL) exploits mutual dependencies between observed (x) and unobserved (y) portions of input through learning whether various parts of the input are consistent with each other. For example, whether a second video clip is a logical continuation of

a first clip. Unlike generative models, JEPA does not attempt to fully reconstruct y from x. This is because there may be an infinite number of y compatible with a given x, and furthermore it is inefficient and may be intractable to predict *all* the (possibly irrelevant) details of y. A different approach instead uses Energy-Based Models (EBM), in which a scalar energy function $F(x, y)$ assigns low energy to compatible pairs (x, y) and high energy otherwise (see Figure 13), thus enabling the system to represent multimodal dependencies in which the set of y compatible with x can be a single point, several discrete points, a manifold, or a combination.

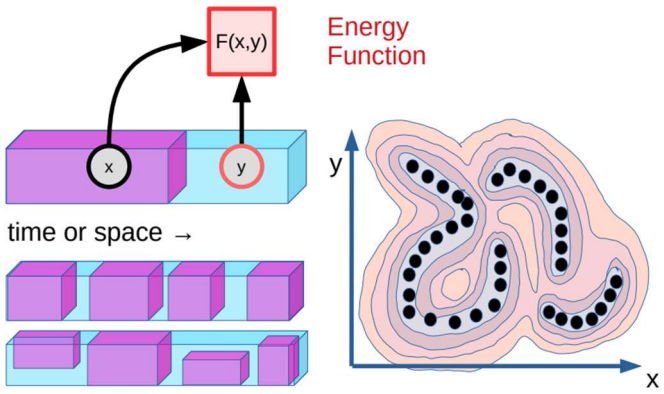


Figure 13: Self-Supervised Learning (SSL) and Energy-Based Models (EBM). Black dots represent data points, closed lines represent energy function contours. [52]

To enable Mode-2 planning, JEPA predicts *representations* of y from *representations* of x, through feature (perceptual) encoders $s_x = g_x(x), s_y = g_y(y)$ such that i) s_x and s_y are maximally informative about x and y, ii) s_y is easily predictable from s_x, a tradeoff between predictability and extracting informative representations of the world. Learning to predict a small image region from neighboring regions in space and time would enable edge extraction and contour movement detection. These would in turn form the basis for more complex concepts such as depth maps, then the notion of a 3D object, object permeance, then intuitive physics at the object level such as gravity, and so on forming increasingly abstract representations and timescales, therefore enabling hierarchical understanding of how the world works.
It should be mentioned that energy and loss functions are sometimes confused, however one important distinction is that energy functions can be minimized at inference time, while loss is minimized at training time only. Furthermore, as we will now see, an energy framework enables even more.

Uncertainty handling with latent variables:

Latent variables z parameterize the set of possible relationships between x and y, encoding information necessary to link x and y but that cannot be extracted from x, whose value is not observed but inferred. In a video prediction task, z might represent the car's trajectory at a fork, encompassing both possible outcomes – left or right. It contains information that would be useful for prediction but is not knowable due to partial observability or stochasticity. One may not know whether the driver in front of me will turn left or right, accelerate or brake.
A latent-variable EBM *infers* a z that minimizes the energy (i.e., "best explains" the relationship between x and y):

$$F(x, y) = \min_{z \in Z} E(s_y, Pred(s_x, z))$$

A major challenge, however, when training EBM architectures is **collapse** (see Figure 14), where the energy landscape becomes "flat", giving the same value to all values of y.

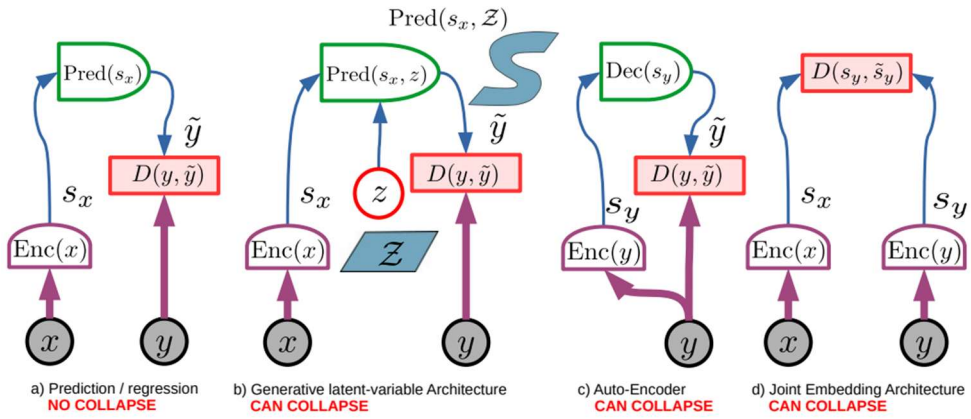


Figure 14: A few standard architectures and their capacity for collapse.
(a) Deterministic generative architecture: cannot collapse because it can only produce a single output. For a given x, only one value of y may have zero energy.
(b) Non-deterministic generative architecture: can collapse when the latent variable has excessive information capacity. If for a given x and for all y there exists a z that produces zero prediction energy (e.g. if z has the same or higher dimension as y), the entire y space will have low energy. The information capacity of z should be just enough so that varying z over its set will produce all the plausible $\tilde{y}$ for a given x.
(c) Auto-encoder: can collapse if the system learns the identity function or if it can correctly reconstruct a region of y space that is much larger than the region of high data density, thereby giving low energy to an overly large region.
(d) Simple joint embedding architecture: can collapse if the encoders ignore the inputs and produce representations that remain constant and equal (i.e. carry insufficient information). [52]

There are two main methods for collapse prevention, contrastive methods and regularized methods. Contrastive methods consist of finding negative samples on which to push energy function higher. Methods such as SimCLR, GANs, and Denoising Auto-Encoders (of whom Masked Auto-Encoders are a special case) can be seen as a form of contrastive methods. However, contrastive methods scale poorly in high-dimensional spaces, due to the need for devising a scheme for generating or finding contrastive samples, as well as the curse of dimensionality – a very large number of negative samples may be needed to ensure the energy function is high in all regions unoccupied by the data distribution, growing exponentially with the dimension of the representation space.
Regularized methods, on the other hand, operate by constraining and minimizing the volume low-energy regions can take, therefore "shrink-wrapping" the high data density regions. Such regularizing constraints include sparsity, noise (as in VAEs) or quantization.

Joint-Embedding Predictive Architecture (JEPA):

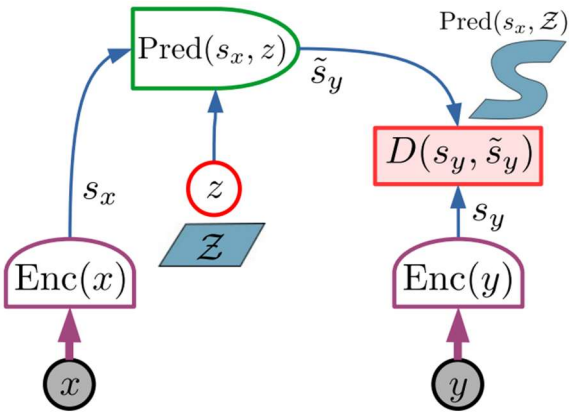


Figure 15: The Joint-Embedding Predictive Architecture (JEPA) [52]

JEPA's main advantage is replacing generative objectives and the need to predict every detail of y, instead *performing predictions in abstract representation space*. Two encoders (see Figure 15) map x, y into latent representations s_x, s_y. A predictor then estimates s_y from s_x, optionally utilizing a latent variable z. The energy is the prediction error in representation space $D(s_y, Pred(s_x, z))$. To prevent collapse during training using regularizers as mentioned, four criteria are used: maximize the information of s_x about x, maximize the information of s_y about y, make s_y easily predictable from s_x, minimize the information content of the latent variable z used in the prediction (by making z discrete, low-dimensional, sparse, noisy, etc.). The former two can be implemented using VICReg, which adds variance and covariance loss terms over components of s_x, s_y as well as over batches, therefore ensuring they are independent and are not constant [54]. JEPA therefore finds a trade-off between completeness and predictability of representations. Additionally, a prediction head can be added to train for representations useful for specific tasks.

Hierarchical JEPA (H-JEPA):

A promising aspect of JEPA is the capacity for stacking higher level JEPAs on top of lower-level ones, forming an architecture able to predict at multiple timescales and multiple levels of abstraction. The lower-level models predict detailed, short-term states, while the higher-level models abstract away details for longer-term predictions. Once the architecture is trained, it can be used for **hierarchical planning** (see Figure 16), in which a sequence of high-level abstract actions $(a2[2], a2[4])$ is inferred to minimize a high-level objective $C(s2[4])$. The inferred abstract actions are fed to lower-level cost modules $C(s[2]), C(s[4])$ which define subgoals for the lower layer. The lower layer then infers an action sequence that minimizes the subgoal costs. As such, a complex planning task such as traveling to Paris can be decomposed into booking a flight, getting to the airport, etc. Getting to the airport can in turn be decomposed into calling a taxi, exiting the house and so on, with exiting the house further decomposed into standing up, walking to the door, opening the door. This decomposition can descend all the way down to fine-grained, millisecond-by-millisecond motor controls.

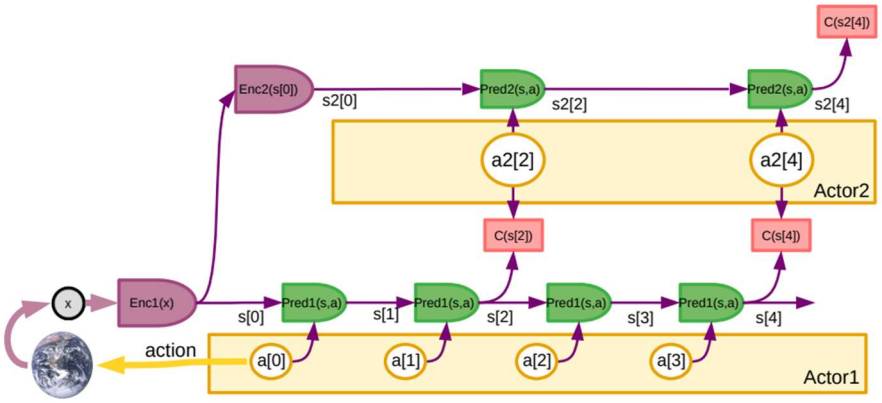


Figure 16: Hierarchical JEPA for Mode-2 hierarchical planning [52]

5.4 Implementations – I-JEPA, V-JEPA

The JEPA architecture has been implemented in two notable self-supervised implementations – I-JEPA and V-JEPA, the former is image-based, and the latter is video-based. The main idea is performing prediction in *representation* space (rather than reconstructing pixel-level details), therefore avoiding irrelevant information and allowing for multiple plausible predictions, in contrast to deterministic, generative architectures. Notably, this is done without the use of pretrained image encoders, text, negative examples, reconstruction, data augmentation or other sources of supervision.

Image-based JEPA learns by predicting the representations of specific image regions based on surrounding context [55]. The architecture consists of a target encoder and a context encoder for producing a representation of the target region and surrounding context, respectively, and of a predictor for predicting the representation of the target region from the representation of its surroundings. All three are implemented using a Vision Transformer (ViT) [24].

V-JEPA extends this idea to learning visual representations from video data, with masking and prediction performed along both the spatial and the temporal dimensions of the video frames [56]. The architecture is shown in Figure 17. Evaluated on downstream image and video tasks, it performs well on motion and appearance-based tasks, demonstrating higher sample efficiency and performance than state-of-the-art pixel-based approaches, as shown in Figure 18 (a)-(b).

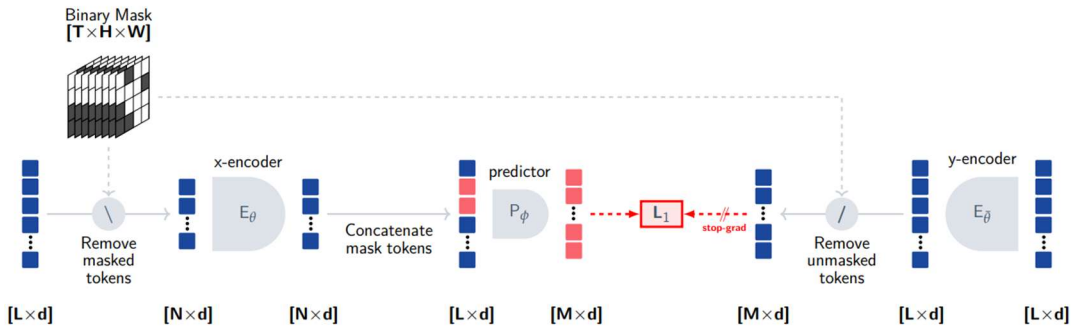
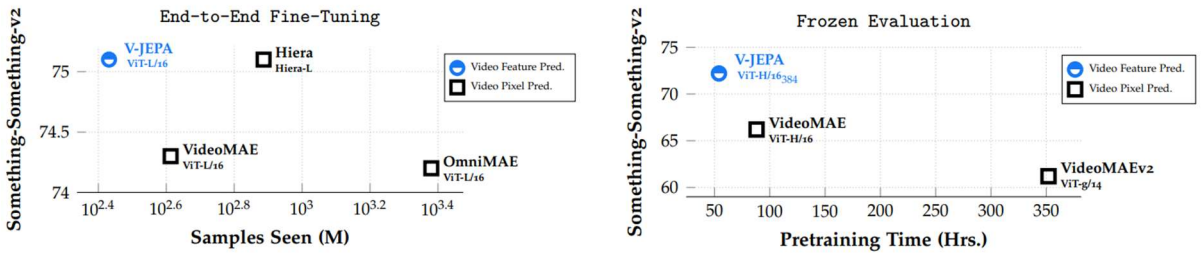


Figure 17: V-JEPA. Training operates on a video clip of T frames with spatial resolution H × W, flattened into a sequence of L tokens. The x-encoder processes the masked video sequence, and outputs an embedding vector for each input token. Next, the outputs of the x-encoder are concatenated with a set of learnable mask tokens containing positional embeddings of the masked spatio-temporal patches. The predictor network processes the combined token sequence, and outputs an embedding vector for each

mask token. The error is computed from predictor outputs and prediction targets using an L1 loss. The prediction targets correspond to the output of the y-encoder. [56]

Method	Arch.	Params.	Data	Video Tasks K400 (16×8×3)	Video Tasks SSv2 (16×2×3)	Video Tasks AVA	Image Tasks IN1K	Image Tasks Places205	Image Tasks iNat21
Methods pretrained on Images									
I-JEPA	ViT-H/16_{512}	630M	IN22K	79.7	50.0	19.8	84.4	66.5	85.7
OpenCLIP	ViT-G/14	1800M	LAION	81.8	34.8	23.2	85.3	**70.2**	83.6
DINOv2	ViT-g/14	1100M	LVD-142M	**83.4**	50.6	24.3	**86.2**	68.4	**88.8**
Methods pretrained on Videos									
MVD	ViT-L/16	200M	IN1K+K400	79.4	66.5	19.7	73.3	59.4	65.7
OmniMAE	ViT-H/16	630M	IN1K+SSv2	71.4	65.4	16.0	76.3	60.6	72.4
VideoMAE	ViT-H/16	630M	K400	79.8	66.2	20.7	72.3	59.1	65.5
VideoMAEv2	ViT-g/14	1100M	Un.Hybrid	71.2	61.2	12.9	71.4	60.6	68.3
Hiera	Hiera-H	670M	K400	77.0	64.7	17.5	71.4	59.5	61.7
V-JEPA	ViT-L/16	200M	VideoMix2M	80.8	69.5	25.6	74.8	60.3	67.8
	ViT-H/16	630M		**82.0**	71.4	**25.8**	75.9	61.7	67.9
	ViT-H/16_{384}	630M		81.9	**72.2**	25.0	**77.4**	**62.8**	**72.6**

(a)



(b)

Figure 18: Comparison with State-of-the-Art Models (a)-(b) [56]

5.5 An integrated architecture

The architecture presented in this chapter is consistent with many of the previously discussed ideas – from a neuroscientific standpoint, the cost module presented closely resembles the function and form of the hypothalamus and amygdala, as well as certain parts of the BG (predicting future reward from a state). These are essentially a form of reward mechanism. The neural substrate of immediate emotions in animals and humans (e.g., pain, pleasure, hunger) play a role similar to the intrinsic cost, while emotions such as fear may be the result of anticipation of outcome whose function is akin to the trainable critic.
The perceptual module is akin to the various sensory and perceptual areas of the brain, also potentially forming a hierarchical representation. The ACC is deeply involved in simulation and world modeling, with MPC inference allowing for inferring optimal action sequences (differentiable optimization) and latent variables allowing for, in addition to accounting for uncertainty, discretizing different *options* which may be then searched through (for example as seen in AlphaGo MCTS and its variants). The configurator, although vaguely presented, plays the role of certain areas of the prefrontal cortex, performing executive control and attention modulation. Additionally, JEPA may resemble the latent imagination mechanism from PlaNet and Dreamer architectures, as well as their planning in representation space for greater efficiency. The idea of hierarchical planning has also been presented in the Director model.
Supporting the framework of SSL and EBM, the neocortical columns, as we have seen, all perform the same computation, trained to explain (predict) their own input.

This architecture does not, however, model intent as in the brain - the ACC forms a generative model of intent, explaining a sequence of states (with their transitions representative of actions)

given goals from the amygdala. As also stated by LeCun in [52], there has yet to be found a good way of learning "good" goals for hierarchical planning, without being predefined by humans. Perhaps utilizing the direct valence results of the reward mechanism to predict optimal action sequences will not only reduce the expensive action sequence optimization procedure but will also form the notion of intent modeling. Crucially, this may allow for a higher layer module – a self-supervised predictive model of intent, as well as a generative model of perception encoding, or rather – *mentalization*, opening up a new set of intelligent abilities that may be lacking with the proposed architecture. We will briefly explore this idea in the next chapter.

5.6 Language and symbols in machine intelligence - Discussion

Language and symbolic reasoning, as demonstrated by large language models (LLMs), have proven useful in capturing a large amount of knowledge from text. This has led to the question whether human-level intelligence can emerge by scaling such architectures.
However, these systems remain somewhat brittle, fundamentally disconnected from the physical and sensory reality underpinning human common sense, consisting of the ability to use models of the world to fill in information or predict the future. Current LLMs rely on tokenized, latent-variable free, generative paradigms, trained using contrastive methods (in particular, masked autoencoders, a type of denoised autoencoder), and are subject to the mentioned limitations of these methods [52]. This approach may be suitable for text, which is a discrete tokenized sequence, but struggles with continuous, high-dimensional sensory data such as video or physical interaction, whose representation also necessitates eliminating irrelevant information. LLMs simplify uncertainty representation by producing a probability vector over words in the dictionary. As such, their reasoning is limited to language alone, lacking "grounded intelligence" – common sense through world models from low-levels of abstraction all the way up to knowledge acquired by language. Further, the absence of latent variables prevents exploration of multiple interpretations of a percept and search for optimal courses of action to achieve a goal. In fact, dynamic goal specification in such models is essentially impossible.
The main adaptive cognitive ability enabled by language is transfer of thought. But what is the point if you cannot fully understand the real world meaning of that thought, both immediate and hidden?
Furthermore, current LLMs, and large-scale generative transformer-based models in general, are largely inefficient, requiring massive amounts of training, computation, energy and data.
Language must perhaps be built on top of a more foundational form of intelligence as paved by evolution, integrated into the world model, possibly through encoders into a unified latent space as seen in the DynaLang architecture, allowing for knowledge to be shared in a single, general world model.

6. Mentalization

6.1 STaR: An approximation of Mentalization in Deep Learning Language Models

STaR (Self-Taught Reasoner) is a method for bootstrapping step-by-step reasoning capabilities of a large language model (LLM) [57]. It starts with a small set of examples containing step-by-step reasoning called "rationales" and prompts the LLM to generate for a larger, unannotated question dataset. The model then filters out rationales that led to incorrect answers and fine-tunes (trains) itself on the remaining successful rationales. For questions answered incorrectly, a process called "rationalization" is employed, where the correct answer is provided as a hint and the model is prompted to generate a rationale that justifies it. This loop is then iteratively repeated, letting the model improve itself by learning from its own generated reasoning.
This method has proved effective in improving complex reasoning capabilities in such LLMs.

Although initially developed for language models, this rationalization process may be seen as a special case of one aspect of mentalization in the brain which, as we have seen, infers intent given observed actions and knowledge. Rationales in this case may seem as a "flattened out", reduced form of intent as modeled by the ACC in the brain, which aims to explain an action sequence given goals – whereby the observed action sequence is the chain of thought (CoT) represented as a discrete dictionary word sequence. One might then ask why would a more general form of mentalization be needed in machine intelligence architectures? What is missing from the autonomous intelligence architecture presented in the previous chapter?

6.2 The Importance of Mentalization in Machine Intelligence

I hypothesize mentalization may be necessary for building machine intelligence architectures capable of understanding, predicting, and interacting within complex environments. It enables systems to model and infer the intent, goals and knowledge of others, as well as to plan for their own future goals – needs they may not feel now (i.e., to simulate future states of self). Such theory of mind (ToM) may be crucial for imitation learning and learning skills through observation, discerning between intentional and accidental actions, as has been demonstrated by [58].
Furthermore, mentalization allows reasoning about the whys behind actions, enabling interpolating observed actions and learning from these interpolations not merely as a sensory stimulus but as a coherent model of the notion of other and self (enabling meta-cognition – "thinking about thinking") [59]. This allows social and emotional intelligence through a notion of intent (of self and others) behind observed behavior, allowing social interactions and collaborations.
For example, understanding the nuanced objective of a user's query extends beyond its literal content to inferring his underlying goal as learned from knowledge, observation, and actions. This supports long-term planning, with applications ranging from robotics where learning by

observation from human demonstration to conversational models capable of inferring user intent and knowledge gaps for optimal, tailored outputs and interaction.

6.3 Implementing Mentalization in Deep Learning - Discussion

As we have seen, the gPFC-PSC network plays a major role in mentalization in the brain, building on the evolutionary older simulation and reinforcing mechanisms. Perhaps one way of extending the machine architecture discussed in the previous chapter for mentalization and forming a self-supervised generative (or predictive) model of the ACC-sensory network itself, may be utilizing joint-embedding predictive architectures to form two modules – one to predict the latent activations of the sensory-perceptual system, given an allocentric representation of space (as formed in the HPC), observed information, and perhaps mirror neurons as exist in the premotor cortex as well as short-term memory. Given the knowledge and observed behavior, a second predictor could be used to predict analogue of ACC latent activation (intent), which could be the highest-level objective in the H-JEPA architecture ($C(s2[4])$ in Figure 16; akin to intent), thus forming a model of what intentions and knowledge in others are consistent with their observed behaviors. During "inference mode" it could be trained on the real-time intent representations in the highest objective function of H-JEPA, then allowing pause-and-simulate behavior for modeling latent intent and knowledge in others or self most consistent with observed behavior. As discussed in the neuroscience chapter, such mechanism allows for theory-of-mind to infer the knowledge and intentions of others, to learn skills through observation, and to plan for future needs and goals which aren't currently experienced. This implementation therefore forms a second-order model of the world model itself.

This suggested implementation is merely a hypothesis, as no current machine intelligence architecture performs a neurologically consistent form of mentalization. Worth mentioning is the fact that, as this proposal utilizes the same underlying self-supervised predictive architecture as in internal simulation and world modeling, the neocortical columns perform the same similar generative-predictive computation in both the ACC-sensory and the gPFC-PSC networks, as explained in the first chapter. This may be indicative that indeed no different computational architecture is needed for the two functions, merely dependent on the unique input and output connectivity, as well as further adaptive plasticity and training. It is also worth mentioning that this model is a simplification and approximation of the countless intricate circuits and structures in the human brain involved in this mechanism, which are yet to be fully understood, including the posterior cingulate cortex (PCC), the several different structures in the gPFC such as the dlPFC and BA10 (Broadman area 10), the vmPFC and many more.

7. Conclusion

We have now researched neuroscientific principles behind the brain, understanding part of what may make human intelligence unique from other animals, as well as deep learning research regarding reward mechanisms and reinforcing, simulation and world models, the Joint-Embedding Predictive Architecture and H-JEPA, and lastly proposed ideas for implementing mentalization in deep learning architectures and extending the cognitive architecture built throughout the seminar. In addition to each chapter being foundationally based on its predecessor, as in the brain, these mechanisms may also be complementary in machine intelligence (from an operational standpoint).
These ideas can be utilized for autonomous general machine intelligence architectures, application specific deep learning models, or physical intelligence and robots.

There are many further directions of research that remained out of this seminar's scope, and are nonetheless important and worthy of mentioning, including: spike-timing dependent neural networks, local plasticity, memory modules (long term semantic and episodic memory, working memory), attention – both external and internal, topographical and network theory analysis of NNs, Bayesian inference models and probabilistic modeling for decision-making and problem solving, combinatorial creativity in ML models, as well as more in-depth inspection of the neocortical columns and various other brain structures and circuits. Additionally, there are many different types of neurons, with various dendritic integration properties, different functions, inhibitory neurons and recurrent feedback connections, and many chemical and physical properties to understand regarding the brain operation and computation.

7.1 Self-Criticism:

The breakthrough neuroscientific model is merely an approximation of the brain evolution and various structures and mechanisms. There are countless other brain structures, circuits which have not been mentioned, or were generalized under larger structures and regions.
Additionally, many aspects of intent modeling in simulation and particularly also mentalization have been vaguely described, with no currently existing fully plausible deep learning architectures. These architecture idea proposals are merely theoretical and currently have not an implementation or have not been experimented with, tested, evaluated and compared.
Lastly, there is a larger question with regard to the researched mechanisms and models – are inductive biases in architectures optimal? Examples are CNNs and LSTMs, which assume and contain structural inductive priors with respect to the data. With the increase of compute and data, the Transformer architecture – a *generalization* of MLPs (with weights computed "on the fly") – has demonstrated superior performance in many cases. Are specific architectures even better than forming more general models and allowing the model to learn its own optimal structure? A difference is, however, that the discussed architecture does not assume priors on the data but is simply intelligently designed.
Lastly, it should be remembered that the brain should inform but not constraint – there may be different optimal models for in-silico ML architectures with possibly different functions and environments. This seminar attempted to follow this principle with regard to machine learning.

References

[1] Bennet MS (2021), "Five Breakthroughs: A First Approximation of Brain Evolution From Early Bilaterians to Humans", Frontiers in Neuroanatomy

[2] Bennet M (2023), "A Brief History of Intelligence", Mariner Books

[3] Sugahara F et al. (2017), "Reconstructing the Ancestral Vertebrate Brain"

[4] Bennet MS (2021), "What Behavioral Abilities Emerged at Key Milestones in Human Brain Evolution? 13 Hypotheses on the 600-Million-Year Phylogenetic History of Human Intelligence"

[5] Bennet MS (2020), "An Attempt at a Unified Theory of the Neocortical Microcircuit in Sensory Cortex", Frontiers Neural Circuits, Volume 14

[6] Harshvardhan GM et al. (2020), "A comprehensive survey and analysis of generative models in machine learning", Computer Science Review, Volume 38

[7] Kingma DP, Welling M (2022), "Auto-Encoding Variational Bayes"

[8] White MG et al. (2018), "Cingulate Cortex Input to the Claustrum Is Required for Top-Down Action Control", Cell Report, Volume 22

[9] Rosenblatt F (1958), "The perceptron: a probabilistic model for information storage and organization in the brain", Psychology Review

[10] Silver D et al. (2021), "Reward is enough"

[11] Shannon CE (1949), "Programming a Computer for Playing Chess"

[12] Sutton RS (1988), "Learning to predict by the methods of temporal differences"

[13] Watkins CJCH et al. (1992), "Q-learning"

[14] Mnih V et al. (2013), "Playing Atari with Deep Reinforcement Learning"

[15] Mnih V et al. (2015), "Human-level control through deep reinforcement learning", Nature 518, 529–533

[16] van Hasselt H et al. (2015), "Deep Reinforcement Learning with Double Q-learning"

[17] Wang Z et al. (2015), "Dueling Network Architectures for Deep Reinforcement Learning"

[18] Hessel M (2017), "Rainbow: Combining Improvements in Deep Reinforcement Learning"

[19] Sutton RS et al. (1999), "Policy Gradient Methods for Reinforcement Learning with Function Approximation"

[20] Konda VR et al. (1999), "Actor-Critic Algorithms"

[21] Haarnoja T et al. (2018), "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor"

[22] Friston K (2010), "The free-energy principle: a unified brain theory?"

[23] Andrychowicz M et al. (2018), “Learning Dexterous In-Hand Manipulation”

[24] Haarnoja T et al. (2024), “Learning Agile Soccer Skills for a Bipedal Robot with Deep Reinforcement Learning”

[25] Dosovitskiy A et al. (2020), “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”

[26] Vaswani A et al. (2017), “Attention Is All You Need”

[27] Nair A et al. (2015), “Massively Parallel Methods for Deep Reinforcement Learning”

[28] Dabney W et al. (2020), “A distributional code for value in dopamine-based reinforcement learning”

[29] Ren Z et al. (2017), “Deep Reinforcement Learning-based Image Captioning with Embedding Reward”, CVPR

[30] Chen L et al. (2021), “Decision Transformer: Reinforcement Learning via Sequence Modeling”

[31] Janner M et al. (2021), “Offline Reinforcement Learning as One Big Sequence Modeling Problem”, NeurIPS

[32] Frémaux, N. et al. (2016), "Neuromodulated Spike-Timing-Dependent Plasticity, and Theory of Three-Factor Learning Rules", Frontiers in Neural Circuits

[33] Miconi T et al. (2018), “Differentiable plasticity: training plastic neural networks with backpropagation”, ICML

[34] Sutton RS (1991), “Dyna, an Integrated Architecture for Learning, Planning, and Reacting”

[35] Andrychowicz M et al. (2017), “Hindsight Experience Replay”

[36] Silver D et al. (2016), “Mastering the game of Go with deep neural networks and tree search”

[37] Silver D et al. (2017), “Mastering the game of Go without human knowledge”

[38] Silver D et al. (2017), “Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm”

[39] Ha D, Schmidhuber J (2018), “World Models”

[40] Hochreiter S et al. (1997), “Long Short-Term Memory”

[41] Schrittwieser J et al. (2020), “Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model”

[42] Mandhane A et al. (2022), “MuZero with Self-competition for Rate Control in VP9 Video Compression”

[43] Hafner D et al. (2019), “Learning Latent Dynamics for Planning from Pixels”, ICML

[44] Hafner D et al. (2020), "Dream to Control: Learning Behaviors by Latent Imagination", ICLR

[45] Hafner D et al. (2020), "Mastering Atari with Discrete World Models", ICLR

[46] Hafner D et al. (2022), "Deep Hierarchical Planning from Pixels"

[47] Wu P et al. (2022), "DayDreamer: World Models for Physical Robot Learning"

[48] Hafner D et al. (2023), "Mastering Diverse Domains through World Models"

[49] Lin J et al. (2023), "Learning to Model the World with Language", ICML

[50] Chen C et al. (2022), "TransDreamer: Reinforcement Learning with Transformer World Models"

[51] Robine J et al. (2023), "Transformer-based World Models Are Happy With 100k Interactions", ICLR

[52] LeCun Y (2022), "A Path Towards Autonomous Machine Intelligence"

[53] Kahneman D (2011), "Thinking, Fast and Slow"

[54] Bardes A et al. (2021), "VICReg: Variance-Invariance-Covariance Regularization for Self-Supervised Learning", ICLR

[55] Assran M et al. (2023), "Self-Supervised Learning from Images with a Joint-Embedding Predictive Architecture"

[56] Bardes A et al. (2024), "Revisiting Feature Prediction for Learning Visual Representations from Video"

[57] Zelikman E et al. (2022), "STaR: Bootstrapping Reasoning With Reasoning"

[58] Abbeel P et al. (2010), "Autonomous Helicopter Aerobatics through Apprenticeship Learning"

[59] Zhu Y et al. (2020), "Dark, Beyond Deep: A Paradigm Shift to Cognitive AI with Humanlike Common Sense"