Deep learning, reinforcement learning, and world models

Romain Wang, Jean Tozza

February 28, 2025

1 Introduction

The article *Deep Learning, Reinforcement Learning, and World Models* provides a comprehensive summary of the *International Symposium on Artificial Intelligence and Brain Science (AIBS2020)*. The symposium aimed to advance artificial intelligence (AI) toward a level of intelligence comparable to the human brain in the future.

Currently, AI surpasses human capabilities in specific tasks such as *image recognition* and *speech generation*. However, in other domains—such as *robotics in industrial production*—AI struggles to adapt to *unpredictable situations*, whereas humans can easily adjust.

To address these limitations, researchers at the symposium introduced new approaches, including the *sim-to-real method* and *Bayesian inference*, which enhance AI's ability to learn and adapt.

This review presents these methods in a structured format, covering:

- Deep Learning
- Reinforcement Learning
- Learning Methods and Models
- The Small-Sample Learning Problem
- Future Directions

Each section highlights key insights from the symposium, exploring the latest advancements in AI and its connection to neuroscience.

2 Deep Learning

2.1 World model for perception, control and language

Humans don't rely on just one sense to interact with the world. To make decisions, they use multiple perceptual inputs like vision, hearing, smell, and touch. To achieve human-like performance, AI should also be able to integrate multiple sensory inputs.

To get closer to how the human brain processes information, researchers have developed multimodal deep generative models that combine different input types, such as text and images, to make better decisions. However, this approach comes with a challenge: the missing modality problem.

The missing modality problem occurs because these models often assume that all input sources are always available. When one of the modalities is missing, the model might fail to generate a response, even if the other available inputs still contain useful information.

To solve this issue, researchers introduced the Joint Multimodal Variational Autoencoder (JMVAE). This model includes separate encoders for each modality. Each encoder learns to reconstruct data that resembles the original dataset using Variational Autoencoder (VAE) models:

- Encoder: Compresses input data into a lower-dimensional latent space.
- Decoder: Reconstructs the original data from this latent space.

After training, JMVAE can generate missing data from the available modalities. This means that even if one modality is missing, the model can still function correctly by using the learned representations from the remaining inputs.

We can define the JMVAE algorithm as follows:

Let x_1 and x_2 be two different modalities. The goal is to model the joint distribution $p(x_1, x_2)$ while allowing flexible inference over missing modalities.

We define a shared latent variable z that captures the dependencies between modalities x_1 and x_2.

In a standard single-modality VAE, we assume that observed data x is generated from some latent variable z. Our goal is to learn the distribution $p(z|x)$, but computing this exactly is intractable. Instead, we approximate it with a learned variational distribution $q(z|x)$:

$$q(z|x) \approx p(z|x)$$

This approximation allows us to perform inference efficiently.

In our case, the true posterior we want to approximate is:

$$p(z|x_1, x_2)$$

Since exact inference is again intractable, we introduce a variational approximation:

$$q(z|x_1, x_2) \approx p(z|x_1, x_2)$$

To allow for flexible inference even when some modalities are missing, we introduce modality-specific variational posteriors:

$$q(z|x_1) \quad \text{and} \quad q(z|x_2)$$

This means:

- When both x_1 and x_2 are available, we use $q(z|x_1, x_2)$.
- When only x_1 is available, we use $q(z|x_1)$.
- When only x_2 is available, we use $q(z|x_2)$.

For a single-modality VAE, the goal is to approximate the true posterior $p(z|x)$ using a variational distribution $q(z|x)$. The ELBO is derived from the log marginal likelihood $\log p(x)$, which is intractable to compute directly:

$$\log p(x) \geq \mathbb{E}_{q(z|x)} \left[\log p(x|z)\right] - D_{KL} \left[q(z|x) \| p(z)\right]$$

where:

- $\mathbb{E}_{q(z|x)}[\log p(x|z)]$ is the reconstruction loss, ensuring that the latent representation z can generate x.
- $D_{KL}[q(z|x)\|p(z)]$ is the KL-divergence term, which regularizes $q(z|x)$ to be close to a prior $p(z)$ (typically a standard normal distribution).

This balance between reconstruction loss and latent regularization is what allows a VAE to learn meaningful representations.

Now, JMVAE extends this to two modalities x_1 and x_2. Instead of maximizing $\log p(x)$, we now maximize:

$$\log p(x_1, x_2) \geq \mathbb{E}_{q(z|x_1,x_2)} \left[\log p(x_1, x_2|z)\right] - D_{KL} \left[q(z|x_1, x_2)\|p(z)\right]$$

This means:

- We encode both modalities jointly into a shared latent space z.
- We ensure that the decoder can reconstruct both x_1 and x_2 from z.
- We regularize the latent space to be close to a prior distribution.

If both modalities are available, we use the full posterior $q(z|x_1, x_2)$ for inference.

JMVAE is widely used in multimodal neural machine translation (NMT), where both text and images are used together for improved translation accuracy. It significantly outperforms traditional neural machine translation models that rely only on text.

For scientists, human intelligence consists of two main components: the Animal OS and the Language App. To replicate human intelligence, they aim to imitate these two components in AI systems. However, before exploring this concept, it is important to define some key terms.

A world model refers to the mental representation of how the world works. Humans rely on their world model to predict future events and make decisions based on past experiences. In AI, a world model is an internal simulation of the environment, learned from data. These models are widely used in AI applications, including the BREMEN method.

Behavior Regularized Offline Reinforcement Learning (BREMEN) is a method designed to minimize the need for real-world data collection in reinforcement learning (RL) applications, making AI training less costly and less risky. One of its key advantages is data efficiency—BREMEN requires only 5 to 10% of the training data needed for earlier RL methods.

BREMEN builds world models and uses a Dyna-style RL approach, allowing AI to learn both from real-world experiences and simulated experiences using its learned world model. To improve reliability, BREMEN trains multiple world models and conducts simulations on them, increasing the robustness of the learning process.

While BREMEN reduces the need for real-world deployments, making RL safer and more efficient, it also has limitations. If the agent is trained using data from a system that is significantly different from its own (e.g., a robot learning from human demonstrations), the learned world model may struggle to generalize effectively to the agent's specific characteristics and capabilities.

Now that we have defined all these terms, we can return to the discussion of human intelligence.

Firstly, we have the Animal OS, which represents the sensory-motor system that includes world models and a controller. Its role is to perceive information from the environment and respond appropriately. In AI, perception occurs through sensory inputs, and the world model is learned from multiple sources using techniques like JMVAE, as mentioned earlier. The system then interacts efficiently with the environment using methods such as BREMEN, which we previously explained.

The other component of human intelligence is the Language App, which is responsible for higher-level cognitive functions such as language, reasoning, and abstract thinking. It operates on top of the Animal OS and calls upon its world model as a module. For example, when imagining a pink elephant, the brain retrieves previous images of elephants from memory and modifies them based on the new linguistic input (pink).

In AI, deep generative models serve a similar purpose by generating new images, texts, and creative outputs based on learned data.

This creative function of the brain is referred to as the "Mental Canvas." It signifies that the world model, built from real-world experiences, can be conditioned by language, enabling imagination to become an essential aspect of intelligence.

2.2 Self-supervised learning

The next major area of improvement in AI is self-supervised learning. In humans, self-supervised learning is how babies learn about the world around them—whether it be language, object interactions, or cause-and-effect relationships. Even though scientists do not yet fully understand this process, they know that knowledge is built through the accumulation of observations over time.

In AI, self-supervised learning is currently best illustrated by predicting missing words in a sentence or forecasting future events. The key idea is for AI to learn patterns without relying on explicit human-provided labels, allowing it to train with minimal human supervision.

There are two major applications of self-supervised learning in AI. The first is learning hierarchical representations of the world, where the algorithm automatically discovers patterns and relationships in data. For example, AI models like BERT and GPT learn language structure by predicting missing words. The second is learning predictive (forward) models of the world, where the algorithm learns to predict future states of the world based on past data. A practical example of this is self-driving cars, which anticipate the next movement of other vehicles.

We want AI to make accurate predictions, but one major challenge is uncertainty and multimodality. Traditional AI algorithms usually produce only one output, but in reality, the world is highly uncertain. AI must be able to predict multiple possible futures, not just one. To address this, Energy-Based Models (EBMs) were proposed. EBMs assign an "energy score" to different possible outcomes and learn to distinguish likely from unlikely states, helping AI handle uncertainty more effectively.

Two methods are derived from Energy-Based Models. The first is contrastive methods, which work by comparing "good" and "bad" examples and adjusting the model so that correct predictions receive low energy scores while incorrect predictions receive high energy scores.

The algorithm works as follows:

1. Define the Key Components - Energy Function $E_\theta(x)$: A function that assigns an energy score to an input x. The model is trained to lower this energy for real data and increase it for fake (contrastive) data. - Dataset D: A set of real data samples $\{x_i\}_{i=1}^N$.

2. Sampling Positive and Negative Examples - Positive Examples x^+: A batch of real data samples drawn from the dataset:

$$x^+ \sim D$$

The model should learn to assign them low energy. - Negative Examples x^-: Fake or generated data samples that the model should assign high energy.

3. Generating Negative Samples Using Langevin Dynamics (MCMC Sampling) One method to generate contrastive (negative) samples is Markov Chain Monte Carlo (MCMC) Sampling using Langevin Dynamics: - Start with random noise and modify it using the gradient of the energy

function:

$$x^- \leftarrow x^- - \alpha \nabla_x E_\theta(x^-) + \sqrt{2\alpha}\epsilon, \quad \epsilon \sim \mathcal{N}(0, I)$$

- This updates the negative sample x^- to be more realistic but still distinguishable from real data, improving the model's ability to learn a meaningful energy landscape.

4. Compute Energy Scores for Both Real and Fake Samples For both real and contrastive (negative) samples, compute their energy values:

$$E^+ = E_\theta(x^+), \quad E^- = E_\theta(x^-)$$

- The model should reduce E^+ (energy of real samples). - The model should increase E^- (energy of negative samples).

5. Compute the Contrastive Loss Function The loss function ensures that real data has lower energy than fake data. A common choice is:

$$L(\theta) = \mathbb{E}_{x^+}[E_\theta(x^+)] - \mathbb{E}_{x^-}[E_\theta(x^-)]$$

This encourages the model to push down the energy of real data while pushing up the energy of contrastive data.

6. Optimize the Model Parameters - Compute the gradient of the loss function:

$$\nabla_\theta L(\theta)$$

- Update the model parameters using gradient descent or an Adam optimizer:

$$\theta \leftarrow \theta - \eta \nabla_\theta L(\theta)$$

This step helps the model improve its energy function, making it better at distinguishing real from fake samples.

This method is widely used in Natural Language Processing (NLP), such as in BERT, RoBERTa, and GPT for predicting missing words. It is also used in computer vision for learning representations of images. However, a significant disadvantage is that it requires a lot of computational power, making it difficult to scale.

Energy-Based Models (EBMs) can be improved using regularized or architectural methods, which constrain a latent variable model to enforce sparsity. This constraint forces the model to focus on the most important aspects of the input data, improving both efficiency and generalization.

Several models incorporate this principle, such as:

- Sparse Autoencoders (Sparse AE) – Introduce sparsity constraints to extract meaningful features.
- Learned ISTA (LISTA) – A neural network-based optimization method that efficiently approximates sparse coding solutions.

One practical application of these methods is predicting how cars around a particular vehicle will move on a highway. In this case:

- The latent variable represents different possible future scenarios.
- Randomly removing parts of the information in the latent space during training prevents the model from relying too heavily on specific features, enhancing its ability to generalize across different situations.

A Forward Model is a system that estimates the next state of the world based on:

- Action A – The decision or movement taken by an agent.
- Latent Variable Z – Represents unknown or uncertain aspects of the world.

Mathematically, the Forward Model is expressed as:

$$S_{t+1} = f(S_t, A_t, Z_t) \tag{1}$$

where:

- S_t is the current state.
- A_t is the action taken.
- Z_t is the latent variable representing uncertainty.
- f is the function modeling the transition to the next state S_{t+1}.

The system is trained using backpropagation, minimizing a cost function that evaluates the quality of the predicted outcomes. A technique called the adjoint state method—commonly used in optimal control problems—is used for optimization.

To make this process more efficient:

- Instead of computing optimal actions from scratch using gradient descent at every step, a policy network (a neural network) is trained.
- The policy network learns a direct mapping from state to optimal action:

$$A_t = \pi(S_t) \tag{2}$$

where π is the policy function, learned through backpropagation.

By training a policy network, the AI can predict optimal actions instantly, avoiding the computational cost of running full optimizations at each step.

2.3 How do neural systems learn to infer?

The brain refines its internal model by predicting what should happen next and comparing those predictions with reality. If the model primarily captures the fundamental interactions or relationships among elements—rather than merely memorizing object statistics—it can adapt more quickly to new situations. This suggests that the system has learned a kind of "skeleton of causal interactions."

From this perspective, we can represent the system using a probabilistic graphical model (PGM)—a mathematical framework that describes dependencies and independencies among variables. There are two main types of PGMs:

- Directed models often align with causal interpretations, where one variable influences changes in another.
- Undirected models capture constraints or symmetrical relationships, such as mutual consistency among variables.

A central question is how a neuron or a group of neurons can represent a probability distribution $P(z)$. This involves a set of encoding functions $\Psi_i(z)$, each representing a small transformation of z. For example, a neuron's response to z might take the form:

$$\Psi_i(z) = g(w_i \cdot z + b_i)$$

where g is a nonlinear activation function. The system then stores the expected response by computing its average:

$$\mu_i = E_{P(z)}[\Psi_i(z)]$$

The most "neutral" or maximum-entropy distribution typically takes the form:

$$p(z) \propto \exp\left(\sum_i \eta_i \Psi_i(z)\right)$$

In theory, deep density estimation (DDE) can approximate distributions well. However, implementing it in a purely feedforward neural network is challenging. Real brains, in contrast, rely on recurrent connectivity, where neurons not only communicate forward but also loop back and receive signals from other areas. Due to this, the authors suggest switching to a recurrent model, which better reflects the structure of biological neural systems. As we mentioned, neural columns in the cortex are densely interconnected and reciprocally linked with other columns. This means that a change in one column affects others, creating a dynamic system that continuously updates its beliefs based on new data. Unlike traditional deep learning models, inference in the brain is a dynamic process rather than a one-shot calculation. This dynamic nature makes the brain better suited for probabilistic graphical models, particularly undirected models like Markov Random Fields, where relationships between variables are mutual and interdependent.

Short-term adaptation happens quickly, allowing the brain to adjust to new environments, while long-term learning involves structural changes and occurs over a longer period. Effective learning algorithms should separate variables into short-term priors and long-term structures to enable efficient adaptation.

Today's deep learning models are still far from replicating the brain's learning and inference mechanisms due to several key limitations. They struggle with handling sequential actions and long-term dependencies. Their network structures lack sufficient loops and feedback mechanisms, which are crucial for dynamic inference.

To better model brain-like intelligence, future deep learning architectures should rely more on recurrent, graph-based structures that incorporate feedback loops and dynamic adaptation mechanisms.

3 Reinforcement Learning

Reinforcement learning (RL) can be simply described as the idea of an automated agent learning through experience to make good decisions under uncertainty. Generally, RL involves four main fields: optimization, delayed consequences, exploitation, and generalization.

Firstly, optimization is focusing on finding the best ways to make decisions. This could involve tasks such as determining the shortest route between two points or maximizing rewards in a given environment. So, RL aims to enable agents to determine the best action in any given situation.

Then, delayed consequences involve understanding that current decisions can have long-term impacts. For instance, actions taken now might not yield immediate rewards but can lead to better

outcomes in the future. This poses two key challenges: planning and learning. Planning involves determining the best course of action despite knowing the environment, while learning refers to the difficulty of determining which specific actions, taken at an earlier time, are responsible for a positive or negative outcome that occurs later. This makes it difficult to accurately attribute credit or blame to decisions.

In exploration, agents learn through direct experience with their environment. This process involves trying, failing, and learning from the outcomes. Although exploration enables agents to improve decision-making over time, it is limited by the inability to know the results of untested actions. This balance between exploring new options and exploiting known information is a fundamental challenge in RL.

Finally, generalization involves wanting to solve challenging problems. This means applying learned strategies to unseen situations, which is essential for handling complex tasks effectively.

Thus, RL is inherently linked to decision processes, as a learning agent gets a reward signal depending on its actions, which also alter that environment's state. Indeed, RL focuses on sequential decision-making under uncertainty, where an agent interacts with the world by taking actions and receiving observations and reward signals.

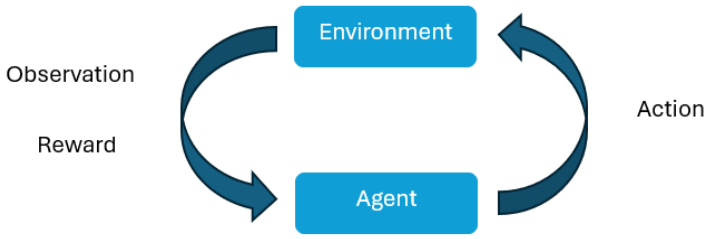

Figure 1: Agent-environment interaction

As we can see in the figure above, the decision-making process in RL is typically modeled as a series of time steps. At each step, the agent takes an action, the environment is updated, and the agent receives an observation and a reward. This creates a feedback cycle that allows the agent to learn and improve its decision-making over time. So, the goal of the agent is to maximize the total expected future reward. This is often achieved through a reward system, such as penalizing the agent for being outside a desired range or rewarding it for reaching specific goals. In RL, we usually use a state to summarize the history of actions, rewards, and observations. This approach relies on the Markov assumption, which simplifies learning by stating that the future depends only on the present state. The Markov assumption balances simplicity and effectiveness, reducing computational demands and training data requirements.

So, by defining problems as Markov Decision Processes (MDPs), RL provides a powerful framework for solving complex sequential decision-making tasks under uncertainty. This approach has found applications in various fields, including robotics, finance, and healthcare, where agents must learn to make optimal decisions in dynamic and uncertain environments.

Reinforcement Learning has emerged from the intersection of AI, neuroscience, and cognitive science, leading to advancements in understanding decision-making processes. Indeed, cognitive science views cognition as mental computations, aligning with RL's modeling of decision-making as computations on learned representations. And neuroscience supports RL because RL models correspond to neural processes, particularly in the reward-based learning system.

There is also the fact that behaviorist ideas have been converted into computational RL algo-

rithms because of their shared focus on learning through interaction with the environment.

So, thanks to RL's combination of AI, neuroscience, and cognitive science, each area progresses. Indeed, cognitive science and neuroscience provide insights for developing more human-like AI algorithms, while AI offers tools to understand cognitive and neural processes.

Reinforcement learning serves as a framework for decision-making in scenarios where an agent chooses actions to maximize rewards, such as in robot control, where agents optimize a robot's movements to complete a task, or in educational systems designed to enhance student learning. However, a main challenge in RL is reward hacking, where agents exploit reward functions to maximize scores without achieving the desired outcomes. For example, an agent trained to reward students for mastering addition and subtraction might be fooled by students who focus only on addition. This highlights the importance of reward function design, which must align with the true objective.

In the following sections, we introduce previous achievements and key ideas and explain that RL is a strong framework for AI to achieve high-level intelligence.

3.1 Fast reinforcement learning with generalized policy updates (by Doina Precup)

Humans tackle big challenges by breaking them into smaller tasks. However, AI often struggles with this approach, requiring a lot of data and time to learn. Barreto et al. (2020) present a smarter solution in their paper "Fast Reinforcement Learning with Generalized Policy Updates," which enables AI to reuse prior knowledge, mirroring human learning strategies.

Their framework improves reinforcement learning efficiency by decomposing complex decision-making problems into smaller, interconnected tasks. The key innovation is generalized policy updates, which extend traditional RL policy evaluation and policy improvement to leverage solutions from previous tasks to accelerate learning in new ones. This divide-and-conquer approach significantly reduces the data and computational requirements for solving large-scale problems that standard RL struggles to handle. By splitting problems into manageable parts and combining existing solutions, AI can learn faster and tackle more complex challenges, making it more efficient and adaptable—closer to human-like problem-solving.

In the paper, the authors considered a Markov decision process (MDP) consisting of:

$$M \equiv (\mathcal{S}, \mathcal{A}, p, r, \gamma) \tag{3}$$

where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, p is the transition probability, r is the reward function, and γ is the discount factor.

A policy in a MDP specifies what action to take in each state. Given a policy π, we want to know how good it is (policy evaluation) and we want to find the best policy (policy improvement).

So, policy evaluation and policy improvement can be thought of as two operators that map between policies and value functions, with:

- **Policy evaluation**: Given a policy π on a single task r, compute the action-value function Q_r^π.
- **Policy improvement**: Given an action-value function Q_r^π, compute an improved policy: $\pi'(s) = \arg\max_{a' \in \mathcal{A}} Q_r^\pi(s, a')$

The process of policy improvement provides a guarantee that the new policy π' obtained is better or at least as good as the old policy π at all states:

$$Q_r^{\pi'}(s, a) \geq Q_r^\pi(s, a) \quad \forall s \in S, \forall a \in \mathcal{A} \tag{4}$$

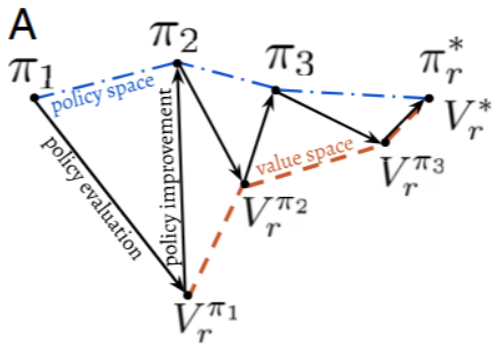

Figure 2: Sequence of policy updates as a trajectory that alternates between the policy and value spaces and eventually converges to an optimal solution.

As shown in the figure above, the process of policy evaluation and policy improvement can be visualized as a series of steps where an agent starts with a policy, evaluates its value function, and then improves the policy, with these steps being interleaved until an optimal policy and value function are reached.

So, the goal of this paper is to generalize this process to multiple reward functions and multiple policies, thereby generalizing the processes of policy evaluation and policy improvement. This approach is useful for multi-task learning and large environments where the agent may encounter new states and new reward functions.

Firstly, generalized policy updates (GPU) enable agents to learn novel policies for multiple tasks by reusing existing policies. On the other hand, generalized policy improvement (GPI) takes a set of policies Π and a reward function (a task r), to computes a new policy π' that is better than the previous ones for that specific reward:

$$Q_r^{\pi'}(s,a) \geq \sup_{\pi \in \Pi} Q_r^{\pi}(s,a) \quad \text{for all } (s,a) \in S \times \mathcal{A}. \tag{5}$$

The goal of generalized policy improvement is to find a policy that is better than all the previous ones, where the upper bound in the equation guarantee that the new policy will have a better value than any of the policies in the given set.

While GPE and GPI offer promise for multi-task RL, they face challenges. Their computational cost can be high, potentially negating the benefits over traditional methods. GPE and GPI effectiveness depends on efficient implementations, as standard policy updates might outperform GPE and GPI if policies are not competent for the task. Overcoming these limitations is vital for their practical application.

To overcome these challenges, the paper suggests using successor features. This involves breaking down rewards into two parts: a feature function and weight parameters. Successor features are similar to Q-functions but predict future features instead of rewards. This allows agents to reuse what they've learned for new tasks by simply adjusting the weight parameters, avoiding the need for costly re-evaluations. Combining this with Generalized Policy Evaluation (GPE) makes reinforcement learning faster and more efficient, requiring less data and computation.

Furthermore, the successor representation (SR) is important in understanding how the brain works. Stachenfeld, Botvinick, and Gershman (2017) studies showed that the hippocampus works like a predictive map. It uses the SR to predict future states based on current ones. Their research showed how the brain combines information about space and time to help us navigate and remember

things better. Momennejad et al. (2017) used the SR model to explain how people learn and make decisions. The study showed that people can sometimes adjust to new situations without practice, suggesting they use a mix of SR-based and more flexible thinking strategies. This research suggests that the brain might use SR as a starting point for estimating values, then use more detailed thinking to refine these estimates when needed, showing how human learning balances efficiency and flexibility in decision-making.

The successor representation does a better job explaining how people learn in experiments compared to traditional reinforcement learning methods. This means that using SR in reinforcement learning, like in generalized policy updates, is not just faster and more efficient with data. It might also help us understand how our brains learn throughout our lives. So we might be able to create AI that learns more like humans do, solving problems in a way that is similar to how our brains work on lifelong learning.

3.2 Deep reinforcement Learning

We know that intelligence is not a collection of separate abilities but rather a unified system that works together in the pursuit of rewards. This means that, for a single goal—such as a kitchen robot maximizing cleanliness—all aspects of intelligence (perception, planning, memory, motor control, etc.) must be engaged simultaneously.

On the other hand, deep learning operates on the principle that any function can be learned by a neural network through gradient descent. This suggests that deep learning can approximate any decision-making function required by Reinforcement Learning (RL) to achieve intelligence.

Deep Reinforcement Learning (DRL) combines these two approaches:

- Reinforcement Learning provides a goal-driven learning process through reward maximization.
- Deep Learning enables function approximation, allowing the system to generalize and handle complex, high-dimensional environments.

General DRL Algorithm

1. Define the environment E, the state space S, and the action space A.
2. Initialize a policy $\pi(s, a; \theta)$, where θ are the parameters (weights) of a deep neural network.

Training Loop (For Each Episode):

1. Observe the initial state s_0 from the environment.
2. Repeat Until the Episode Ends (Terminal State Reached):
   - Select an action a_t based on the current policy:
     - **Exploration**: Try new actions to gather more knowledge.
     - **Exploitation**: Select the best-known action.
   - Execute the action a_t, observe the reward r_t, and the next state s_{t+1}.
   - Compute target values for updating the policy.
   - If using Policy Gradient (PG) methods, update policy parameters using the gradient of expected rewards:

$$J(\theta) = \mathbb{E}\left[R(\tau)\nabla_\theta \log \pi(a_t|s_t;\theta)\right]$$

- Optimize the neural network:
  - Compute the loss function and perform gradient descent to update the parameters θ.
  - Update the target network periodically (for stability in DQN-like methods).

4 Learning methods and models (by Masashi Sugiyama)

Scientists are working on making AI smarter and more helpful for people. They are using different ways to teach AI, each with its own special approach. Some methods use data that humans have already labeled (supervised learning), while others let AI figure things out on its own (unsupervised learning). There are also methods that mix labeled and unlabeled data (semi-supervised learning) or teach AI by letting it interact with an environment and learn from the results (reinforcement learning). Some newer approaches use imprecise or noisy labels to train models (weakly-supervised learning) and aim to improve model performance on datasets with noisy labels(noise-robust learning). The goal is to make AI that can learn more like humans do, adapt to new situations, and be really useful in our everyday lives and society.

Neuroscience has inspired two key AI learning algorithms: **Hebbian learning** and **backpropagation**. Hebbian learning, modeled after long-term potentiation in neurons, follows the principle "neurons that fire together, wire together." It strengthens connections between simultaneously active input and output neurons, mimicking how the brain reinforces frequently used neural pathways. Backpropagation adjusts neural network weights by revisiting the previous layers so that the calculated output is closer to the actual expected output. Both methods are drawn from biological processes but serve different roles in AI learning.

AI has also advanced significantly in learning models, especially deep neural networks. These networks, inspired by neuroscience, include techniques like convolutional neural networks, ReLU, LSTM, and attention mechanisms. These techniques have significantly advanced the capabilities of neural networks, enabling them to tackle complex tasks in various domains such as computer vision, natural language processing, and speech recognition. Furthermore, this progress has also enabled AI to surpass humans in games like chess, go, and shogi. However, AI still struggles to replicate human-like agility in real-world environments, largely due to its need for more learning examples. To address this limitation, researchers employ approaches such as learning from humans, where AI systems observe and imitate human movements, and human-in-the-loop methods, which incorporate real-time human feedback during the learning process.

5 Discussion

AlphaGo's success highlighted a major challenge in deep reinforcement learning: the need for vast amounts of training data to achieve superhuman performance. This sample inefficiency is a significant drawback that limits the application of deep reinforcement learning to many real-world problems. The following sections will explore strategies to address this issue.

5.1 World models and planning methods

The use of "world models" in AI learning has evolved significantly, with the **sim-to-real strategy** being widely adopted to pre-train policies in simulated environments before applying them to real-world scenarios. Notable studies include Akkaya et al. (2019), who used automatic domain randomization to train a robotic hand to solve a Rubik's cube, and Morimoto and Doya (2001),

who applied hierarchical reinforcement learning to teach a robot to stand up. In contrast, Ha and Schmidhuber (2018) introduced a novel approach where AI systems acquire environmental models from image sequences rather than proprioceptive inputs. These methods highlight how simulated environments can accelerate learning and improve performance in complex robotic tasks.

Recent AI advancements have enabled the learning of low-dimensional latent state representations from complex environments, such as Atari games. **MuZero**, for instance, learns game models from images and actions, using Monte-Carlo Tree Search to develop successful strategies without prior knowledge of game rules. Similarly, **DreamerV2**, a world-model-based method, achieves human-level performance on Atari games by learning behaviors in a compact latent space with limited resources. These approaches, along with learning latent locally linear models for robot control, demonstrate significant progress in efficient learning and planning for complex tasks.

If we know how the environment works, we can use a planning method called **model predictive control (MPC)** for robot control. MPC has been used with many types of robots, like wheeled robots, flying drones, and human-like robots. But when controlling real robots, MPC needs a lot of computing power because it has to quickly solve complex problems over and over to plan the robot's actions in real-time.

5.2 Generating/reusing data

Self-play and **experience replay** are valuable techniques in reinforcement learning that improve sample efficiency. Self-play allows agents to generate training data by competing against themselves, while experience replay stores and reuses past experiences. Off-policy algorithms like Deep Q-Networks and Soft Actor-Critic benefit from this approach.

And **importance-sampling-based methods** help efficiently reuse past data, and simulation models can generate additional training data, updating them with real-world data to improve real-world performance. These methods enhance learning in various environments, including robot control, but their applicability depends on the specific context.

5.3 Entropy regularization

Entropy regularization is a key technique in modern deep reinforcement learning, enhancing exploration by maximizing policy entropy alongside rewards, and improving sample efficiency. Algorithms like Soft Actor-Critic (SAC) have achieved state-of-the-art performance in complex tasks, such as quadrupedal locomotion and robotic manipulation. Moreover, incorporating Kullback-Leibler divergence ensures stable learning. Several modern algorithms, including Soft Q-learning, SAC, Trust Region Policy Optimization, and Maximum a Posteriori policy Optimization, use entropy regularization.

Then, **imitation learning**, which is often more sample efficient than traditional reinforcement learning, benefits from entropy regularization, particularly in inverse reinforcement learning, where it helps mitigate ambiguity issues. Recent approaches combining generative adversarial networks with imitation learning further enhance exploration capabilities.

5.4 Hierarchical architecture/composite control

Deep Reinforcement Learning has successfully generated human-like character motions in simulated environments, but it requires large amounts of data and many more learning trials compared to humans. To address this, researchers are exploring two promising approaches: **Hierarchical Reinforcement Learning (HRL)** and **policy composition**. HRL mimics the hierarchical structure of the human brain, potentially leading to more data-efficient learning by implementing

a divide-and-conquer strategy. And policy composition involves combining previously learned policies to create new ones for novel tasks, leveraging frameworks like linearly solvable Markov Decision Processes and compositionality theory.

This approach has been applied to tasks such as character animation and robot walking, allowing for faster learning of composite policies compared to training from scratch. Recent work has further generalized compositionality theory using entropy-regularized reinforcement learning, exploring the relationship between composite value functions and those trained with composite reward functions.

6 Future directions

6.1 Learning to learn

Recent advancements in AI have enabled agents to automatically extract relevant features for given tasks, eliminating the need for hand-designed features. The focus has now shifted to **lifelong learning problems**, where AI agents must efficiently master multiple policies for diverse tasks. Furthermore, this shift has also led to the development of **meta-learning methods**, enabling policy adaptation across different tasks with limited learning trials.

A key area of innovation is **the autonomous adjustment of meta-parameters**, such as learning rates and discount factors, which were previously manually tuned. **Meta-gradient learning methods** have emerged to address this, allowing AI systems to update these parameters, handle multiple tasks, and even discover surrogate objectives. Some cutting-edge studies are exploring AI agents that can find learning rules by themselves without relying on traditional value functions.

New meta-reinforcement learning studies that follow the learning-to-learn principle could lead to big advancements in AI, having a major impact on society.

6.2 Human-inclusive AI

AI can benefit society in creative fields like art and design by offering new perspectives and ideas, especially when clear solutions do not exist. A good example is a collaboration between the fashion designer Ema Rie and AI researchers from the University of Tokyo and RIKEN. Ema provided her dress designs to an AI system, which then suggested new ideas based on those designs. So, this highlights how a human-AI collaboration can impact positively our society.

bibliographystyleplain

References

[1] utaka Matsuo a, Yann LeCun b,c, Maneesh Sahani d, Doina Precup e,f, David Silver e, Masashi Sugiyama g,a, Eiji Uchibe h, Jun Morimoto h, Deep learning, reinforcement learning, and world models In *ASpecial Issue on AI and Brain Science: Perspective*, 2021.

[2] Masahiro Suzuki, Kotaro Nakayama, Yutaka Matsuo JOINT MULTIMODAL LEARNING WITH DEEP GENERA- TIVE MODELS In *AICLR* , 2017.

[3] Samina Amin. ://medium.com/@samina.amin/deep-q-learning-dqn-71c109586bae.

[4] André Barreto. https://www.pnas.org/doi/10.1073/pnas.1907370117.