A Path Towards Autonomous Machine Intelligence

Yann LeCun

https://openreview.net/pdf?id=BZ5a1r-kVsf

Summary by Michael Wollowski

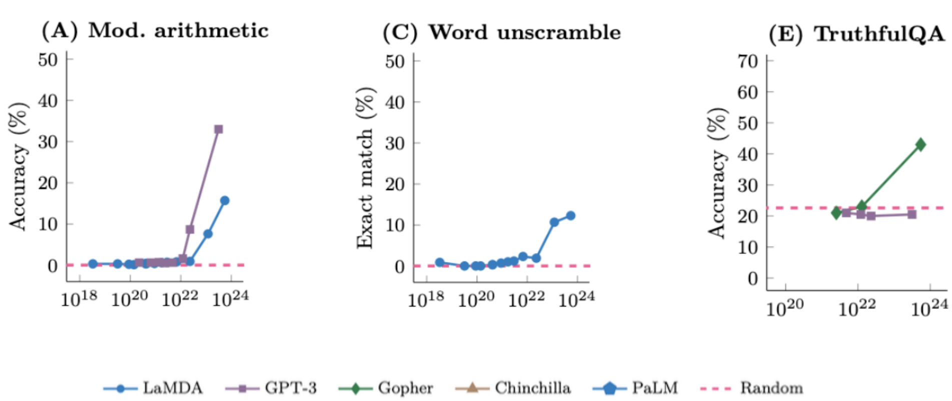
About this Presentation

- It summarizes, without much comment a position paper by Yann LeCun.
- He states that it is "… a position paper expressing my vision for a path towards intelligent machines that learn more like animals and humans, that can reason and plan, and whose behavior is driven by intrinsic objectives, rather than by hard-wired programs, external supervision, or external rewards."

Scaling Laws

- The performance of large language models has shown to be mainly determined by 3 factors:
  - model size (the number of parameters),
  - dataset size (the amount of training data), and
  - the number of iterations used for training.
- We can improve a model by adding parameters (adding more layers or having wider contexts or both), by training on more data, or by training for more iterations.
- The relationships between these factors and performance are known as *scaling laws*.
- LeCun believes there is a limit to what can be achieved by scaling.

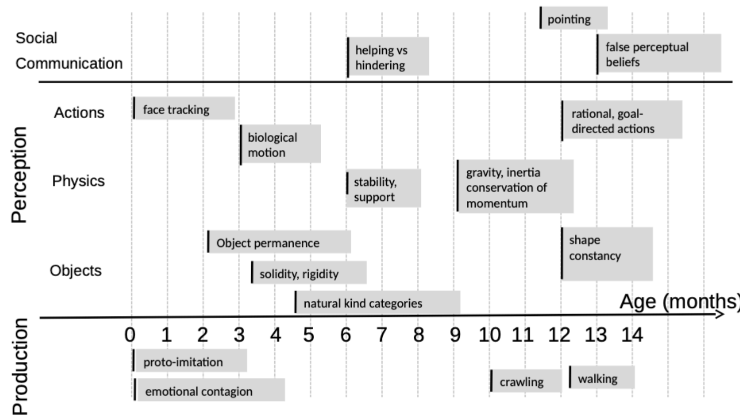
Sudden Emergence of Capabilities



Source: Anderljung et al. Frontier AI Regulation: Managing Emerging Risks to Public Safety

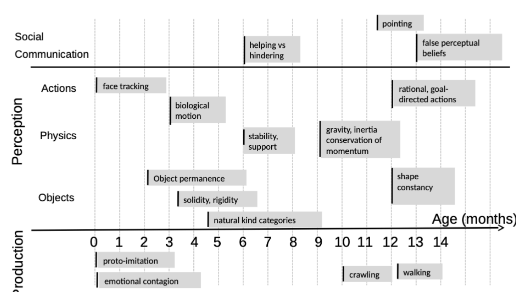
Acquisitions of Concepts about the World

- Learning from the bottom-up
- Turing (and others) argued for that.



Acquisitions of Concepts about the World

- Abstract concepts, such as gravity and inertia, are acquired on top of less abstract concepts, such as object permanence.
- Much of this knowledge is acquired mostly by observation.
- Very little direct intervention, particularly in the first few weeks and months.



System Architecture

- The components and structure of the proposed approach.
- It is an agent architecture.

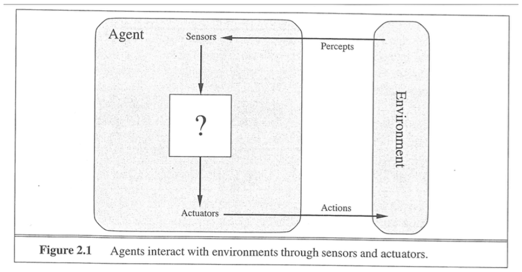


Figure 2.1 Agents interact with environments through sensors and actuators.

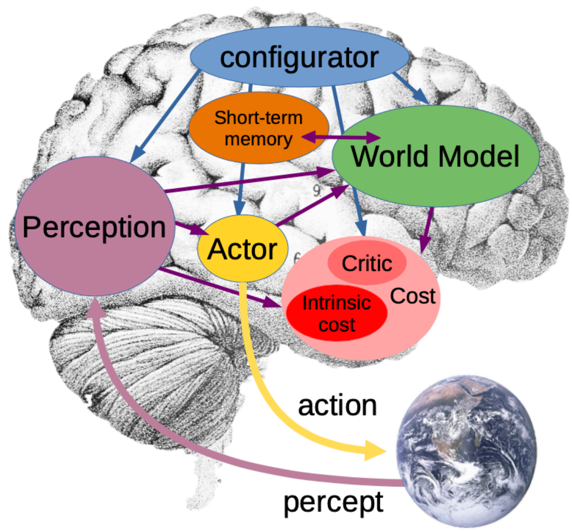
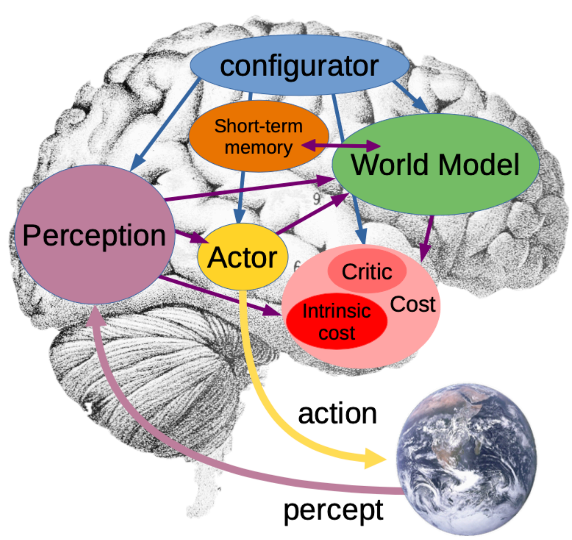


Figure 2.1 source: Russell and Norvig. AIMA.

System Architecture

Configurator module takes inputs (not represented for clarity) from all other modules and configures them to perform the task at hand. **Perception module** estimates the current state of the world. **World model** module predicts possible future world states as a function of imagined actions sequences proposed by the actor.

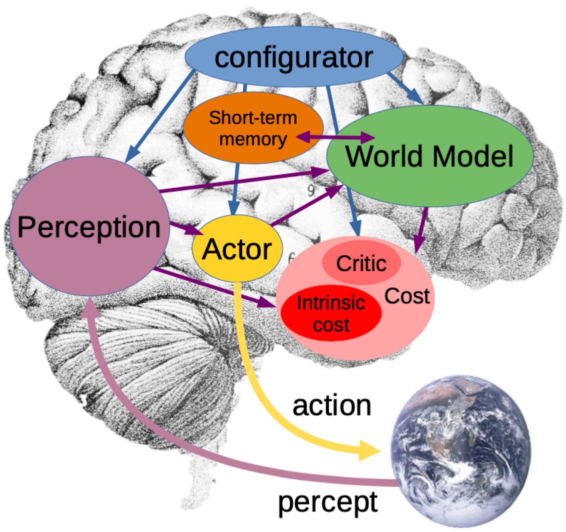


System Architecture

Cost module computes a single scalar output called "energy" that measures the level of discomfort of the agent.
It is composed of two sub-modules, the

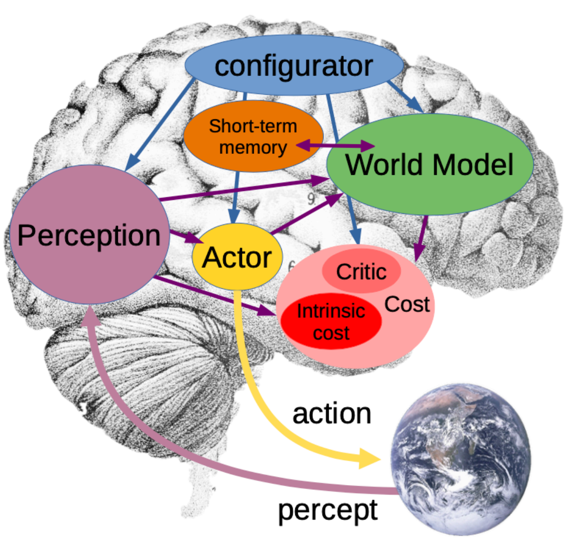
- *intrinsic cost*, which is immutable and
- the *critic*, a trainable module that predicts future values of the intrinsic cost.

Short-term memory module keeps track of the current and predicted world states and associated intrinsic costs.



System Architecture

Actor module computes proposals for action sequences.
World model and the critic compute the possible resulting outcomes.
The actor can find an optimal action sequence that minimizes the estimated future cost, and output the first action in the optimal sequence.



System-1 and System-2 Thinking

- Two possible modes that the model can employ for a perception-action episode.

1. **No complex reasoning**. Produces an action directly from the output of the perception and a possible short-term memory access.

   We call it "Mode-1", by analogy with Kahneman's "System 1".

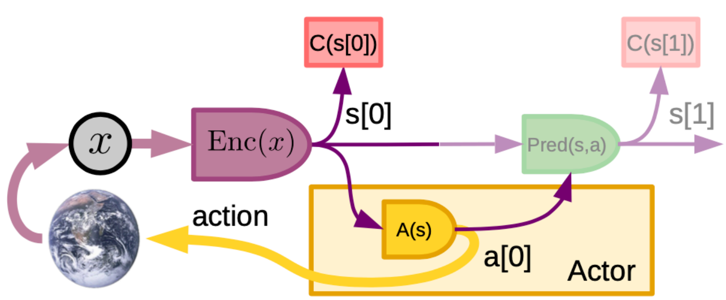
2. **Reasoning and planning** through the world model and the cost.

   We call it "Mode-2" by analogy to Kahneman's "System 2".

- We use the term "reasoning" in a broad sense here to mean constraint satisfaction (or energy minimization).

Mode-1 Perception-Action Episode

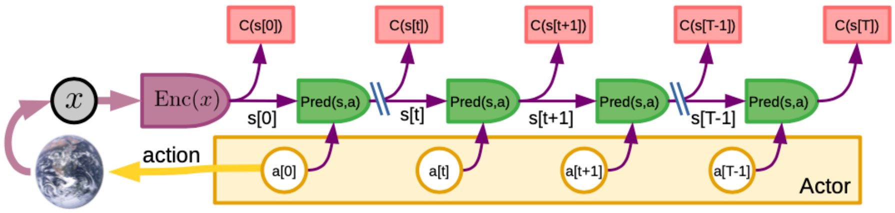
- The perception module estimates the state of the world s[0] = Enc*(*x*).
- The actor directly computes an action, or a short sequence of actions, through a policy module a[0] = A(s[0]).



- This reactive process does not make use of the world model nor of the cost.
- The cost module computes the energy of the initial state f[0] = C(s[0]) and stores the pairs (s[0], f[0]) in the short-term memory.

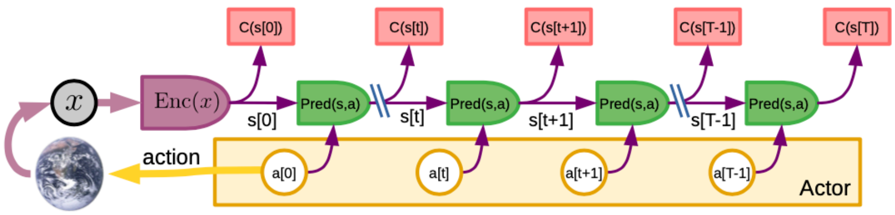
*) Enc stands for Encoder

Mode-2 Perception-Action Episode



- The perception module estimates the state of the world s[0].
- The actor proposes a sequence of actions a[0], a[1], . . . , a[t], a[t + 1], . . . , a[T].
- The world model recursively predicts an estimate of the world state sequence using s[t + 1] = Pred(s[t], a[t]).
- The cost C(s[t]) computes an energy for each predicted state in the sequence, the total energy being the sum of them.

Mode-2 Perception-Action Episode

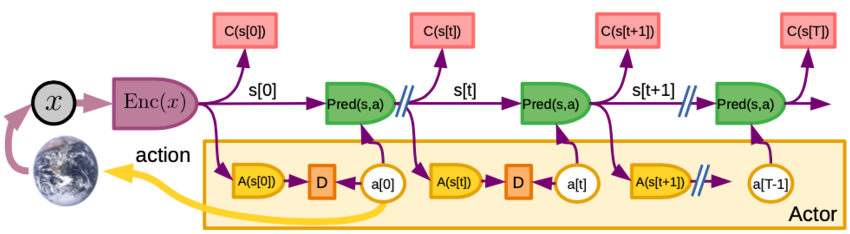


- Through an optimization or search procedure, the actor infers a sequence of actions that minimizes the total energy.
- It then sends the first action in the sequence (or the first few actions) to the effectors.
- Since the cost and the model are differentiable, gradient-based methods can be used to search for optimal action sequences.
- Pairs of states (computed by the encoder or predicted by the predictor) and corresponding energies from the intrinsic cost and the trainable critic are stored in the short-term memory for subsequent training of the critic.

Training as a result of Mode-2 reasoning

This diagram depicts how to train a policy module A(s[t]) to approximate the action that results from Mode-2 optimization.
The system first operates in Mode-2 and produces an optimal sequence of actions (ǎ[0], . . . , ǎ[T]).

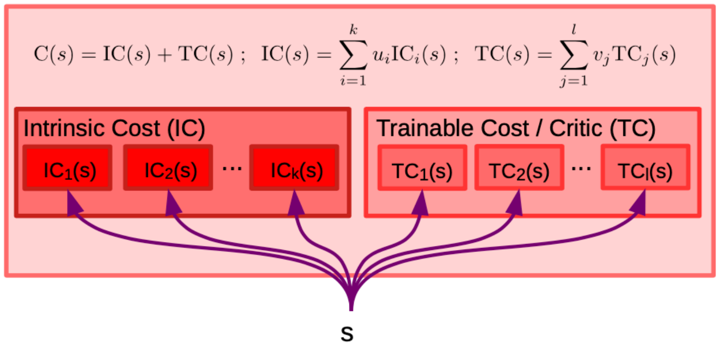
Training as a result of Mode-2 reasoning



Then the parameters of the policy module are adjusted to minimize a divergence D(ǎ[t]), A(s[t])) between the optimal action and the output of the policy module.

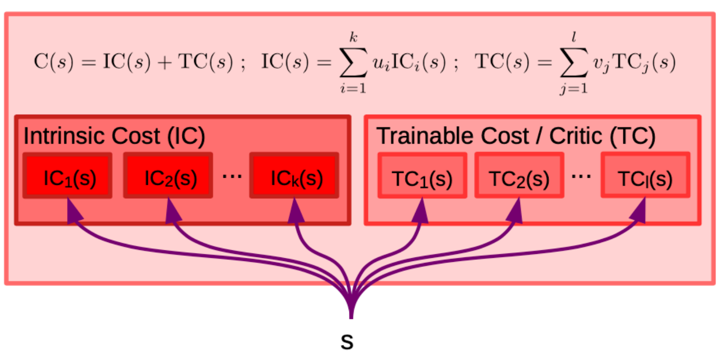
The policy module can then be used to produce actions reactively in Mode-1, or to initialize the action sequence prior to Mode-2 inference and thereby accelerate the optimization.

Architecture of Cost Module



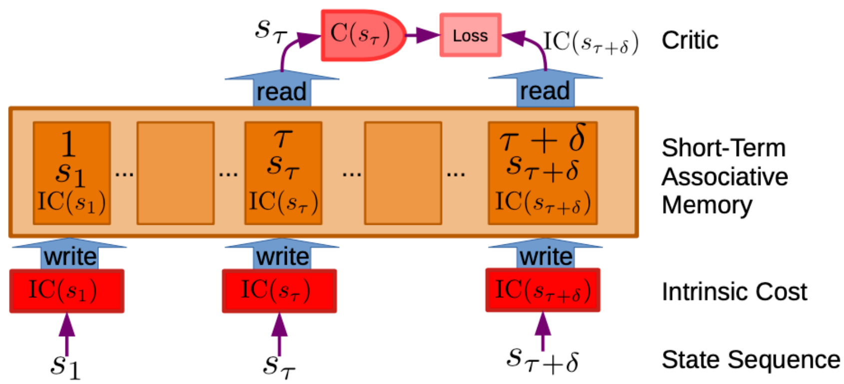
- The cost module comprises the:
  - intrinsic cost module which is immutable $\mathrm{IC}_i(s)$ (left) and
  - the critic or Trainable Cost $\mathrm{TC}_j(s)$ (right), which is trainable.
- Both IC and TC are composed of multiple submodules whose output energies are linearly combined.

Architecture of Cost Module



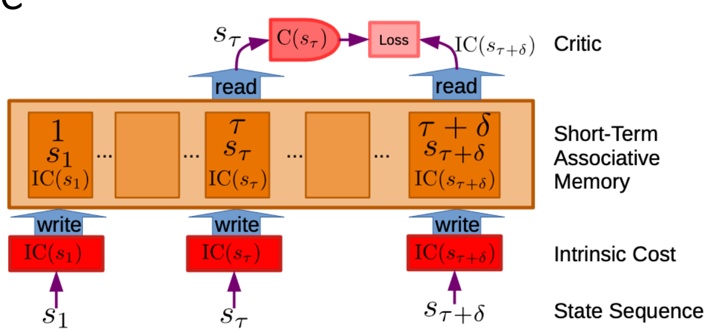
- Each submodule imparts a particular behavioral drive in the agent.
- The weights in the linear combination, u_i and v_j, are determined by the configurator module and allow the agent to focus on different subgoals at different times.

Training the critic



- During planning episodes, the intrinsic cost module stores triplets (time, state, intrinsic energy): (τ, S_τ , $IC(S_\tau)$) into the associative short-term memory.
- During critic training episodes, the critic retrieves a past state vector S_τ , together with an intrinsic energy at a later time $IC(S_\tau+\delta)$.

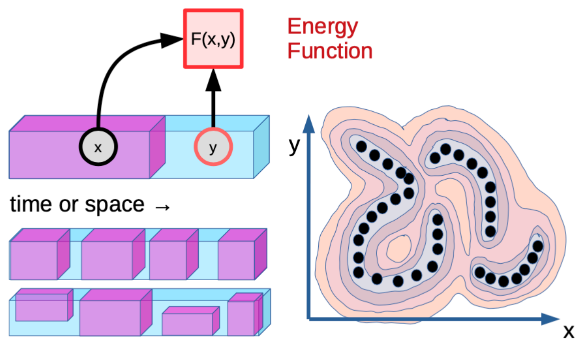
Training the critic



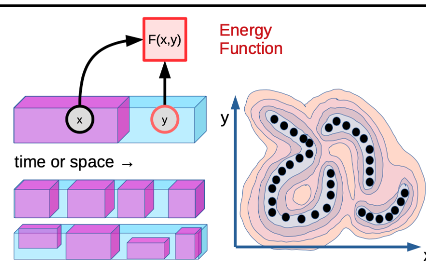
- In the simplest scenario, the critic adjusts its parameters to minimize a divergence measure between the target $IC(S_\tau + \delta)$ and the predicted energy $C(S_\tau)$.
- In more complex schemes, it may use combinations of future intrinsic energies as targets.
- Note that the state sequence may contain information about the actions planned or taken by the agent.

Self-Supervised Learning (SSL) and Energy-Based Models (EBM)

- SSL is a learning paradigm in which a learning system is trained to “fill in the blanks”.
- It captures the dependencies between observed parts of the input and possibly unobserved parts of the input.
- Part of the input signal is observed and denoted x (in pink), and part of the input signal is either observed or unobserved and denoted y (in blue).

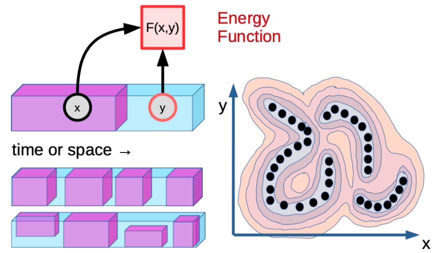


SSL and EBM



- In a temporal prediction scenario, x represents past and present observations, and y represent future observations.
- In a general pattern completion scenario, various parts of the input may be observed or unobserved at various times.
- The learning system is trained to capture the dependencies between x and y through a scalar-valued energy function F (x, y) that takes low values when x and y are consistent or compatible, and higher values if x and y are inconsistent or incompatible.
- In a video prediction scenario, the system would produce a low energy value if a video clip y is a plausible continuation of the video clip x.

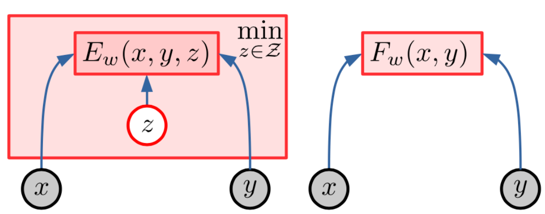
SSL and EBM



- This energy-based model (EBM) formulation enables the system to represent multi-modal dependencies in which multiple values of y (perhaps an infinite set) may be compatible with a given x.
- In the right panel, an energy landscape is represented in which dark discs represent data points, and closed lines represents contours (level sets) of the energy function.

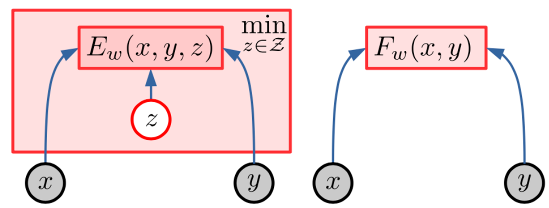
Latent-Variable Energy-Based Model (LVEBM)

- To evaluate the degree of compatibility between x and y, an EBM may need the help of a latent variable z.
- The latent variable can be seen as parameterizing the set of possible relationships between an x and a set of compatible y.
- Latent variables represent information about y that cannot be extracted from x.



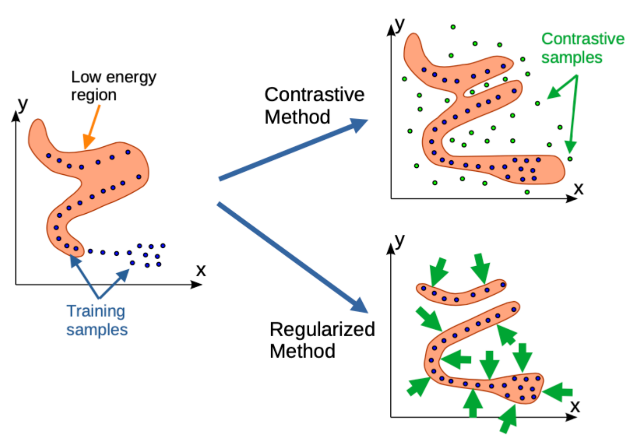
Latent-Variable Energy-Based Model (LVEBM)

- For example, if x is a view of an object, and y another view of the same object, z may parameterize the camera displacement between the two views.
- Inference consists in finding the latent that minimizes the energy ž = $\text{argmin}_z \in Z\ E_w(x, y, z)$.
- The resulting energy $F_w(x, y) = E_w(x, y, ž)$ only depends on x and y.
- In the dual view example, inference finds the camera motion that best explains how x could be transformed into y.



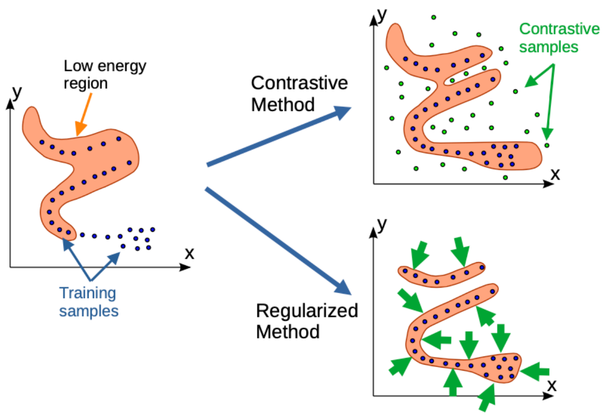
Contrastive and *regularized* methods for EBM training

- A conceptual diagram of an energy landscape is shown on the left.
- Training samples are blue dots.
- The region of low energy is shown in orange (a level set of the energy function).
- **Contrastive** methods (top right) push down on the energy of training samples (blue dots) and pulls up on the energies of suitably-placed contrastive samples (green dots).



Contrastive and regularized methods for EBM training

- **Regularized** methods (bottom right) push down on the energy of training samples and use a regularizer term that minimizes the volume of low-energy regions.
- This regularization has the effect of “shrink-wrapping” the regions of high data density within the low-energy regions, to the extent that the flexibility of the energy function permits it.

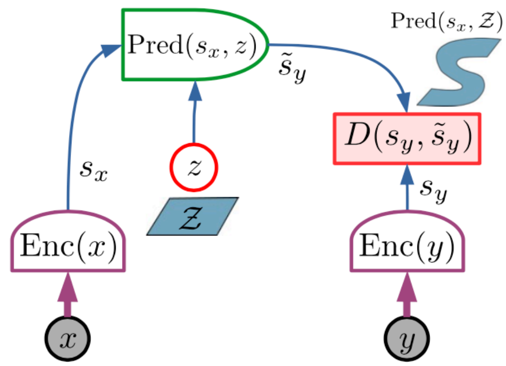


Joint-Embedding Predictive Architecture (JEPA)

- The Joint Embedding Predictive Architectures (JEPA) is an architecture for SSL.
- It can seen as a combination of the Joint Embedding Architecture and the Latent-Variable Generative Architecture.
- Centerpiece of the paper

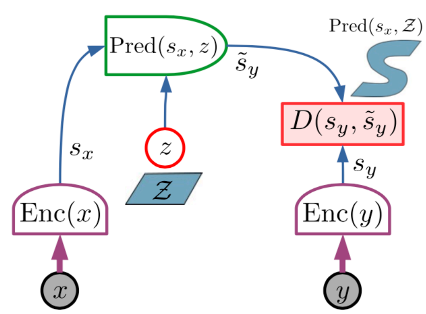
The Joint-Embedding Predictive Architecture (JEPA) consists of two encoding branches

- The first branch computes s_x, a representation of x and the second branch s_y a representation of y.
- The encoders do not need to be identical.
- A predictor module predicts s_y from s_x with the possible help of a latent variable z.
- The energy is the prediction error.



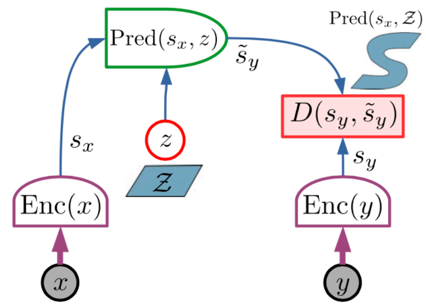
The Joint-Embedding Predictive Architecture (JEPA) consists of two encoding branches

- The main advantage of this architecture for representing multi-modal dependencies is twofold:
  1. The encoder function $s_y = \mathrm{Enc}(y)$ may possess invariance properties that will make it produce the same s_y for a set of different y.
     This makes the energy constant over this set and allows the model to capture complex multi-modal dependencies.
  2. The latent variable z, when varied over a set Z, can produce a set of plausible predictions $\mathrm{Pred}(s_x, Z) = \{\tilde{s}_y = \mathrm{Pred}(s_x, z)\ \forall z \in Z\}$



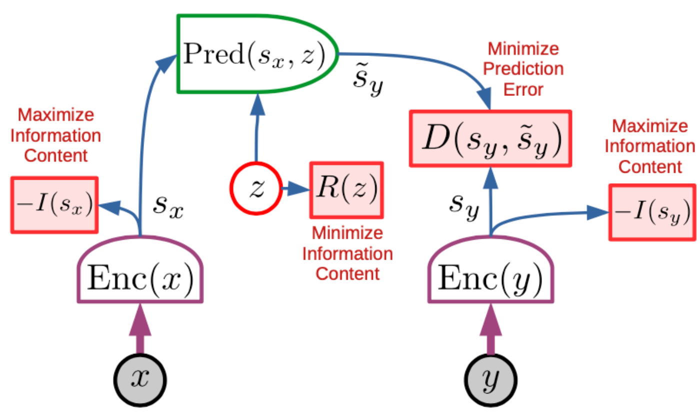
The Joint-Embedding Predictive Architecture (JEPA) consists of two encoding branches

- If x is a video clip of a car approaching a fork in the road, s_x and s_y may represent the position, orientation, velocity and other characteristics of the car before and after the fork, z may represent whether the car takes the left branch or the right branch of the road.



Non-contrastive training of JEPA

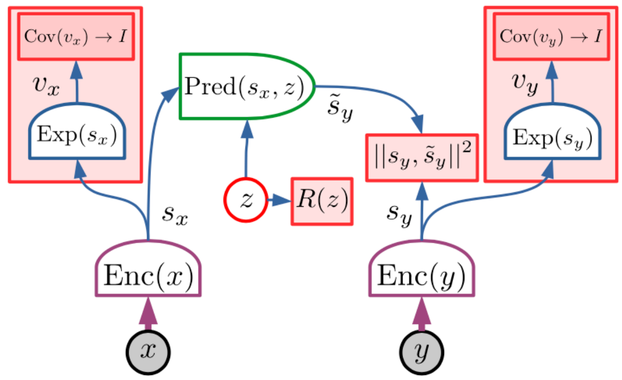
- The main attraction of JEPAs is that they can be trained with non-contrastive* methods.
- The basic principle of such training is that
  1. s_x should be maximally informative about x;
  2. s_y should be maximally informative about y;
  3. s_y should be easily predictable from sx;
  4. z should have minimal information content.



*) Non-contrastive learning is a self-supervised learning technique that only uses positive sample pairs.

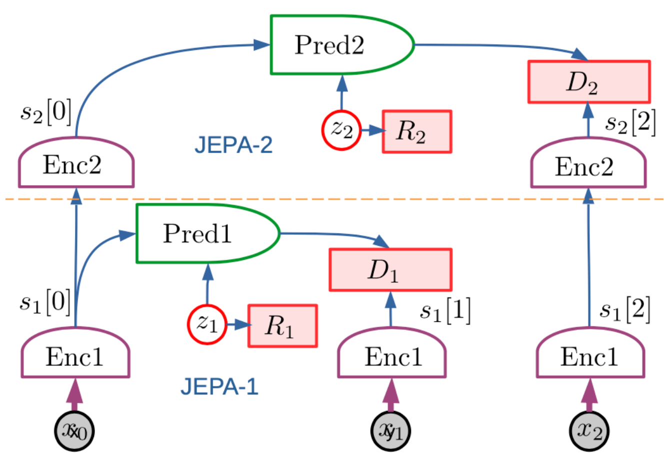
Training a JEPA with VICReg

- VICReg is a non sample-contrastive method for training embeddings.
- The information content of the representations s_x and s_y is maximized by first mapping them to higher-dimensional embeddings v_x and v_y through an expander (e.g. a trainable neural net with a few layers).



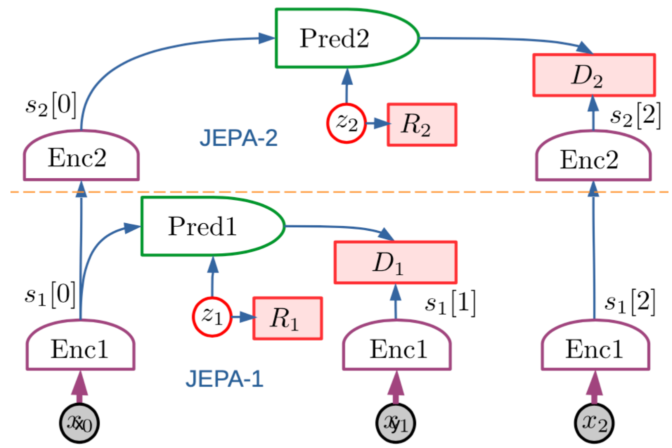
Hierarchical JEPA (H-JEPA)

- The ability of the JEPA to learn abstract representations in which accurate prediction can be performed allows hierarchical stacking.
- JEPA-1 extracts low-level representations and performs short-term predictions.

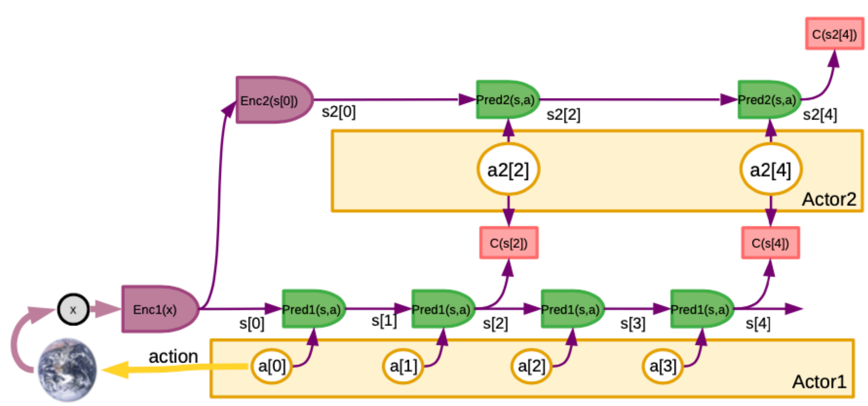


Hierarchical JEPA (H-JEPA)

- JEPA-2 takes the representations extracted by JEPA-1 as inputs and extracts higher-level representations.
- More abstract representations ignore details of the inputs that are difficult to predict in the long term, enabling them to perform longer-term predictions with coarser descriptions of the world state.



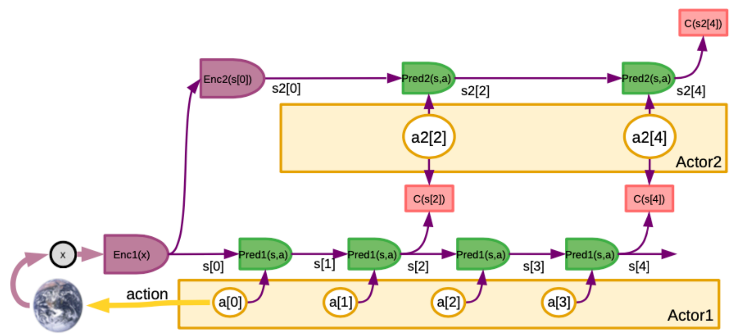
Hierarchical JEPA for Mode-2 hierarchical planning



- A complex task is defined by a high-level cost computed from a high-level world-state representation C(s2[4]).
- A sequence of high-level abstract actions (a2[2],a2[4]) is inferred that minimizes C(s2[4]).

Hierarchical JEPA for Mode-2 hierarchical planning

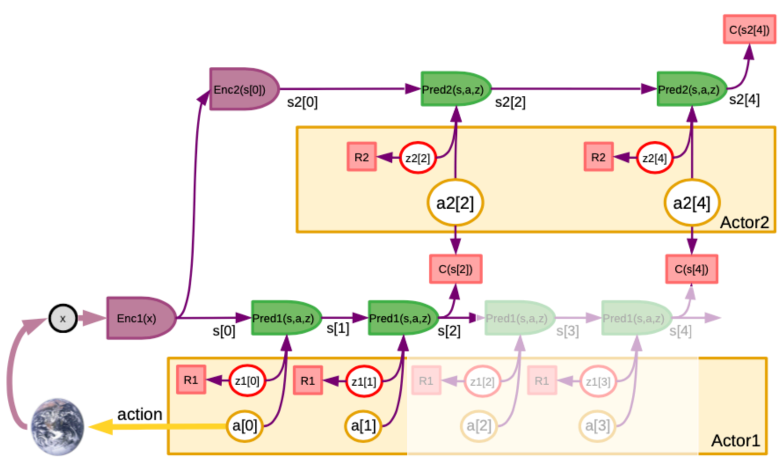
- The inferred abstract actions are fed to lower-level cost modules C(s[2]), C(s[4]) which define subgoals for the lower layer.



- The lower layer then infers an action sequence that minimizes the subgoal costs.
- Although only a 2-layer hierarchy is shown here, it is straightforward to extend the concept to multiple levels.
- The process described here is sequential top-down, but a better approach would be to perform a joint optimization of the actions in all the layers.

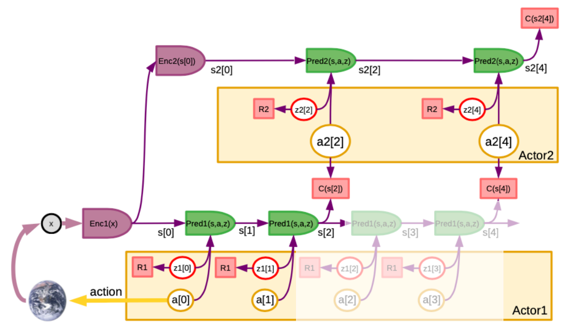
Hierarchical JEPA for Mode-2 hierarchical planning in an uncertain environment

- Realistic environments are not entirely predictable, even when using highly-abstract representations.
- Uncertainty about predictions can be handled by predictors with latent variables.



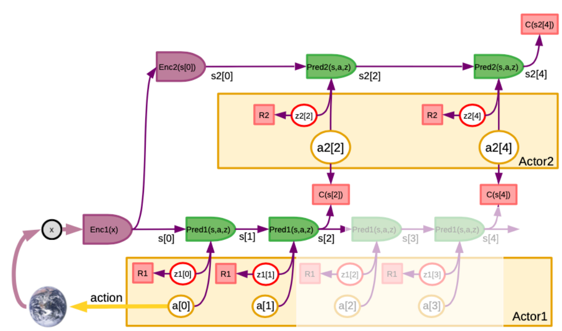
Hierarchical JEPA for Mode-2 hierarchical planning in an uncertain environment

- The latent variables (red circles) contain information about the prediction that cannot be derived from the prior observation.
- To produce consistent latent sequences, the parameters of the regularizer can be functions of previous states and retrieved memories.



Hierarchical JEPA for Mode-2 hierarchical planning in an uncertain environment

- Each sample leads to a different prediction.
- As the prediction progresses, the number of generated state trajectories may grow exponentially.
- If each latent variable has k possible discrete values, the number of possible trajectories will grow as k_t, where t is the number of time steps.



Hierarchical JEPA for Mode-2 hierarchical planning in an uncertain environment

- Directed search and pruning strategies must be employed.
- With multiple predicted trajectories, optimal action sequences can be computed that minimize the average cost, or a combination of average and variance of the cost so as to minimize risk.