Yann Lecun’s Position Paper

A Path Towards Autonomous Machine Intelligence

This position paper proposes an architecture and training paradigms with which to construct autonomous intelligent agents. It combines concepts such as configurable predictive world model, behavior driven through intrinsic motivation, and hierarchical joint embedding architectures trained with self-supervised learning.

A Model Architecture for Autonomous Intelligence

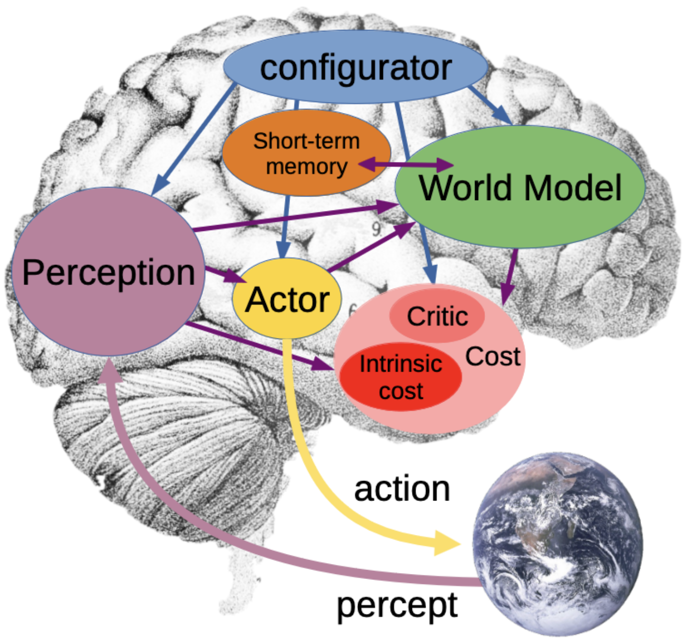

Figure 2: *A system architecture for autonomous intelligence. All modules in this model are assumed to be "differentiable", in that a module feeding into another one (through an arrow connecting them) can get gradient estimates of the cost's scalar output with respect to its own output.*
The configurator module *takes inputs (not represented for clarity) from all other modules and configures them to perform the task at hand.*
The perception module *estimates the current state of the world.*
The world model module *predicts possible future world states as a function of imagined actions sequences proposed by the actor.*
The 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 (not trainable) and computes the immediate energy of the current state (pain, pleasure, hunger, etc), and the critic, a trainable module that predicts future values of the intrinsic cost.*
The short-term memory module *keeps track of the current and predicted world states and associated intrinsic costs.*
The actor module *computes proposals for action sequences. The 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.*
See Section 3 for details.

Typical Perception-Action Loops

- Mode-1: Reactive behavior

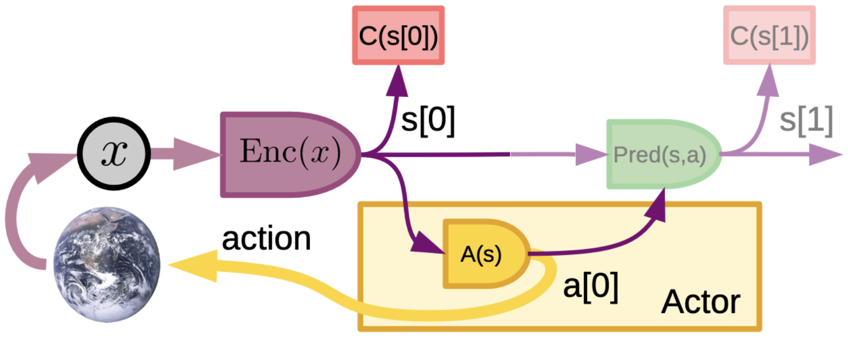

Figure 3: *Mode-1 perception-action episode. The perception module estimates the state of the world $s[0] = \mathrm{Enc}(x)$. The actor directly computes an action, or a short sequence of actions, through a policy module $a[0] = A(s[0])$.*
This reactive process does not make use of the world model nor of the cost. The cost module computes the energy of the initial state $f[0] = \mathrm{C}(s[0])$ and stores the pairs $(s[0], f[0])$ in the short-term memory. Optionally, it may also predict the next state using the world model $s[1] = \mathrm{Pred}(s[0], a[0])$, and the associated energy $f[0] = \mathrm{C}(s[0])$ so that the world model can be adjusted once the next observation resulting from the action taken becomes available.

- Mode-2: reasoning and planning using the world model

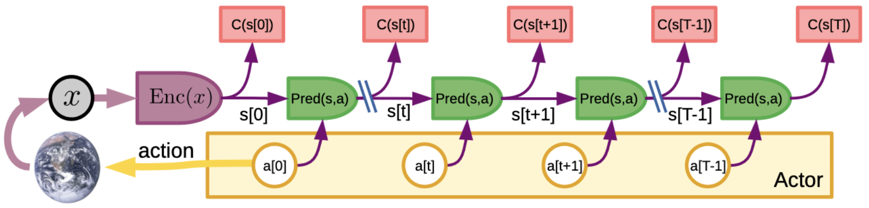

Figure 4: *Mode-2 perception-action episode. The perception module estimates the state of the world $s[0]$. The actor proposes a sequence of actions $a[0], a[1], \ldots, a[t], a[t+1], \ldots, a[T]$. The world model recursively predicts an estimate of the world state sequence using $s[t+1] = \mathrm{Pred}(s[t], a[t])$. The cost $C(s[t])$ computes an energy for each predicted state in the sequence, the total energy being the sum of them. 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. This is, in effect, an instance of classical model-predictive control with receding-horizon planning. Since the cost and the model are differentiable, gradient-based methods can be used to search for optimal action sequences as in classical optimal control. Since the total energy is additive over time, dynamic programming can also be used, particularly when the action space is small and discretized. 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.*

- From Mode-2 to Mode-1: Learning New Skills

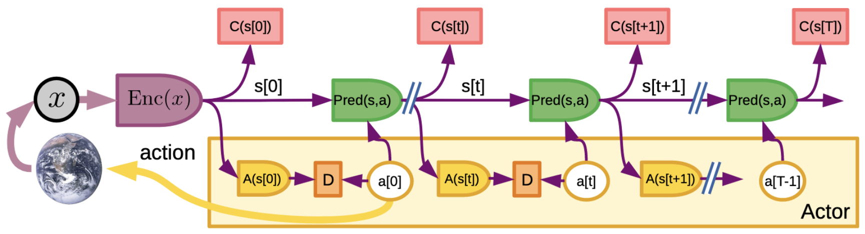

Figure 5: *Training a reactive policy module from the result of Mode-2 reasoning. Using Mode-2 is onerous, because it mobilizes all the resources of the agent for the task at hand. It involves running the world model for multiple time steps repeatedly. This diagram depicts how to train a policy module* $A(s[t])$ *to approximate the action that results from Mode-2 optimization. The system first operates in Mode-2 and produces an optimal sequence of actions* $(\check{a}[0], \ldots, \check{a}[T])$*. Then the parameters of the policy module are adjusted to minimize a divergence* $D(\check{a}[t]), A(s[t]))$ *between the optimal action and the output of the policy module. This results in a policy module that performs* amortized inference, *and produces an approximation for a good action sequence. 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.*

The Cost Module as the Driver of Behavior

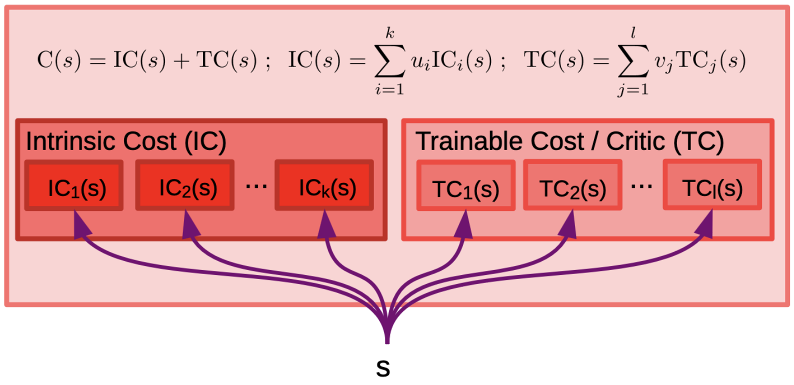
$\mathrm{C}(s) = \mathrm{IC}(s) + \mathrm{TC}(s)\ ;\ \ \mathrm{IC}(s) = \sum_{i=1}^{k} u_i \mathrm{IC}_i(s)\ ;\ \ \mathrm{TC}(s) = \sum_{j=1}^{l} v_j \mathrm{TC}_j(s)$
Intrinsic Cost (IC)
IC_1(s)
IC_2(s)
···
IC_k(s)
Trainable Cost / Critic (TC)
TC_1(s)
TC_2(s)
···
TC_l(s)
s

Figure 6: *Architecture of the 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. 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

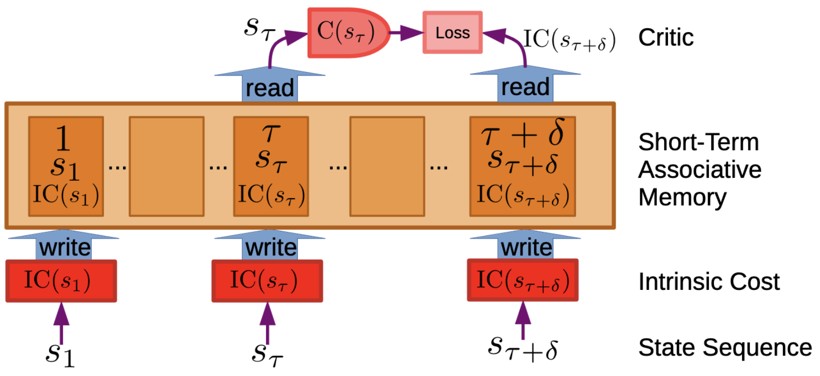

Figure 7: *Training the critic. During planning episodes, the intrinsic cost module stores triplets (time, state, intrinsic energy): $(\tau, s_\tau, IC(s_\tau))$ into the associative short-term memory. During critic training episodes, the critic retrieves a past state vector s_τ, together with an intrinsic energy at a later time* $\mathrm{IC}(s_{\tau+\delta})$. *In the simplest scenario, the critic adjusts its parameters to minimize a divergence measure between the target* $\mathrm{IC}(s_{tau+\delta})$ *and the predicted energy* $\mathrm{C}(s_\tau)$. *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.*

Designing and Training the World Model

Energy-Based Models

Joint Embedding Predictive Architecture (JEPA)

- JEPA and Hierarchical JEPA: a non-generative architecture for predictive world models that learn a hierarchy of representations

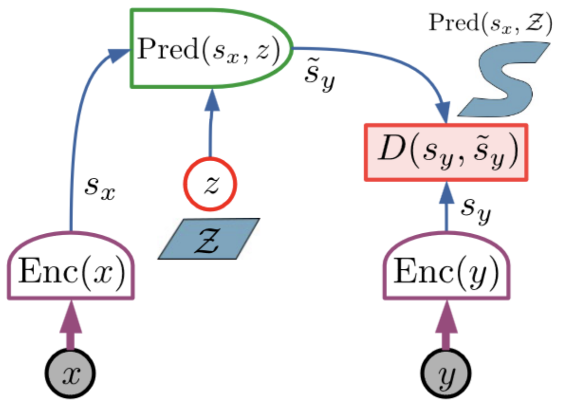

Figure 12: *The Joint-Embedding Predictive Architecture (JEPA) consists of two encoding branches. The first branch computes s_x, a representation of x and the second branch s_y a representation of y. The encoders do not need to be identical. A predictor module predicts s_y from s_x with the possible help of a latent variable z. The energy is the prediction error. Simple variations of the JEPA may use no predictor, forcing the two representations to be equal, or may use a fixed predictor with no latent, or may use simple latents such as discrete variables.*
The main advantage of JEPA is that it performs predictions in representation space, eschewing the need to predict every detail of y, and enabling the elimination of irrelevant details by the encoders. More precisely, the main advantage of this architecture for representing multi-modal dependencies is twofold: (1) the encoder function $s_y = \text{Enc}(y)$ 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 $\mathcal{Z}$, can produce a set of plausible predictions $\text{Pred}(s_x, \mathcal{Z}) = \{\tilde{s}_y = \text{Pred}(s_x, z)\ \forall z \in \mathcal{Z}\}$
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, respectively, ignoring irrelevant details such as the trees bordering the road or the texture of the sidewalk. z may represent whether the car takes the left branch or the right branch of the road.

Training a JEPA

- a non-contrastive self-supervised learning paradigm that produces representations that are simultaneously informative and predictable

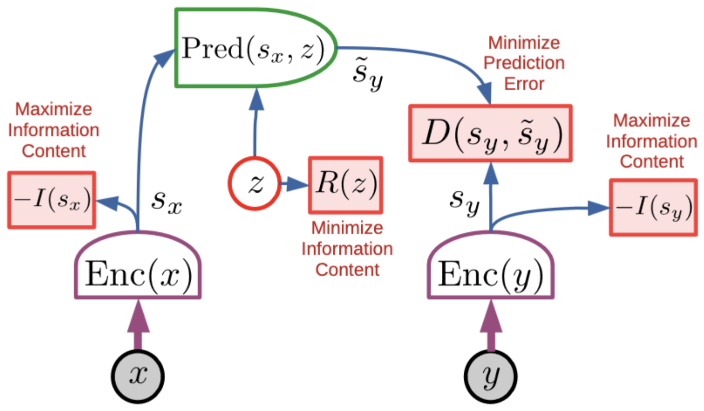

Figure 13: *Non-contrastive training of JEPA.*
The main attraction of JEPAs is that they can be trained with non-contrastive methods. The basic principle of such training is that (1) s_x should be maximally informative about x; (2) s_y should be maximally informative about y; (3) s_y should be easily predictable from s_x; and (4) z should have minimal information content. Criteria 1, 2, and 4 collectively prevent a collapse of the energy function.
Examples of such non-contrastive criteria for JEPA training include VICReg and Barlow Twins.
As with every EBM, JEPAs can also be trained with contrastive methods. But doing so runs into the curse of dimensionality and limits the practical dimension of s_y.

Hierarchical JEPA (H-JEPA)

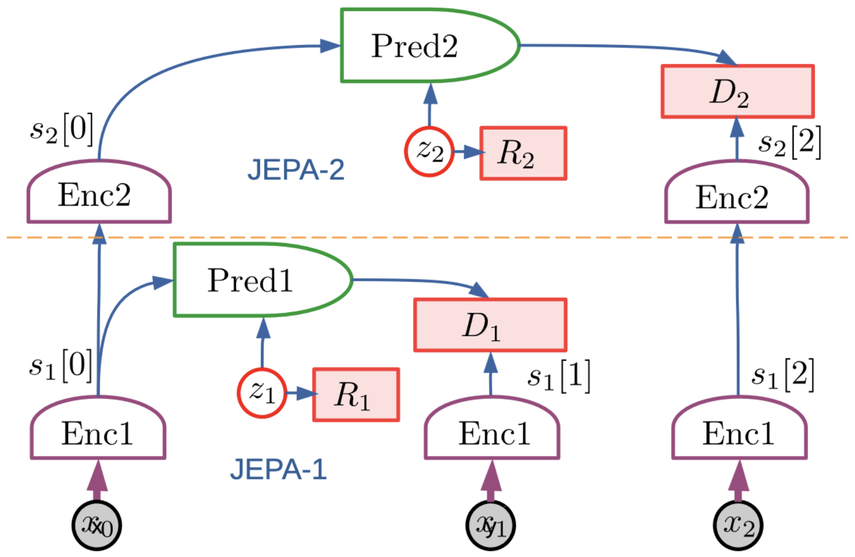

Figure 15: *Hierarchical JEPA (H-JEPA)*
The ability of the JEPA to learn abstract representations in which accurate prediction can be performed allows hierarchical stacking. In this diagram JEPA-1 extracts low-level representations and performs short-term predictions. JEPA-2 takes the representations extracted by JEPA-1 as inputs and extracts higher-level representations with which longer-term predictions can be performed. 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 Planning

- a way to use H-JEPA as the basis of predictive world models for hierarchical planning under uncertainty

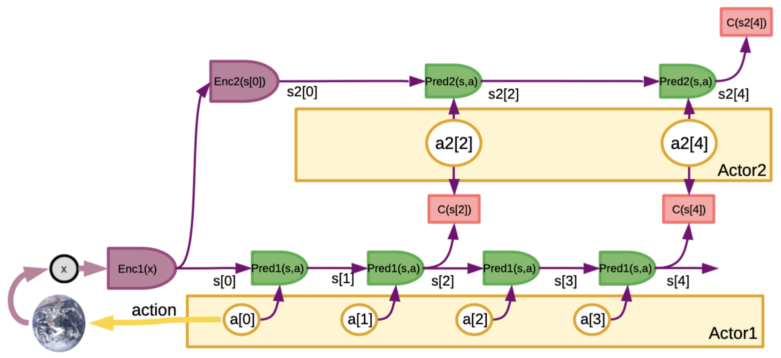

Figure 16: *Hierarchical JEPA for Mode-2 hierarchical planning.*
A complex task is defined by a high-level cost computed from a high-level world-state representation $C(s2[4])$. A sequence of high-level abstract actions $(a2[2], a2[4])$ is inferred that minimizes $C(s2[4])$. The inferred abstract actions 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.

Handling uncertainty

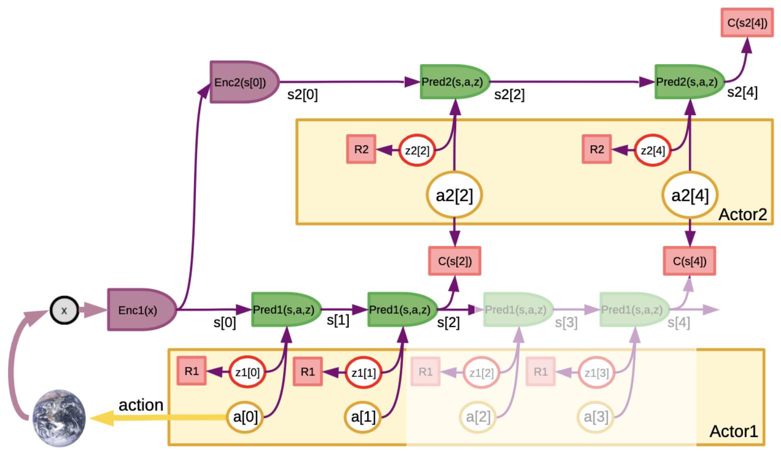

Figure 17: *Hierarchical JEPA for Mode-2 hierarchical planning in an uncertain environment. Realistic environments are not entirely predictable, even when using highly-abstract representations. Uncertainty about predictions can be handled by predictors with latent variables. The latent variables (red circles) contain information about the prediction that cannot be derived from the prior observation. The latent variables must be regularized to prevent an energy collapse and to force the system to predict as much as possible without the help of it.*
At planning time, latent variables are sampled from distributions obtained by applying a Gibbs distribution to the regularizers. Each sample leads to a different prediction. To produce consistent latent sequences, the parameters of the regularizer can be functions of previous states and retrieved memories.
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. 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.