In the latent-variable generative architecture, restricting the information capacity of z will restrict the volume of y space that can take low energy. If z is discrete with possible k values, at most k points in y space will have zero energy. if $\mathcal{Z}$ is a manifold of dimension d then the region of y space with zero energy will have at most d dimensions.

Similarly, in the auto-encoder architecture, restricting the information capacity of s_y will restrict the volume of y space that can be reconstructed with low energy.

Lastly, in the Joint Embedding Architecture, Maximizing the information that s_x contains about x and s_y contains about y will minimize the volume of y space that can take low energy.

In the following, we will focus on an architecture for SSL the Joint Embedding Predictive Architectures (JEPA) which can seen as a combination of the Joint Embedding Architecture and the Latent-Variable Generative Architecture. The JEPA is non-generative in that it does not actually predict y, but predicts the representation of y, s_y from that of x, s_x.

4.4 Joint Embedding Predictive Architecture (JEPA)

The centerpiece of this paper is the *Joint Embedding Predictive Architecture (JEPA)*. JEPA is *not generative* in the sense that it cannot easily be used to predict y from x. It merely capture the dependencies between x and y without explicitly generating predictions of y.

A generic JEPA is shown in Figure 12. The two variables x and y are fed to two encoders producing two presentations s_x and s_y. These two encoders may be different. They are *not* required to possess the same architecture nor are they required to share their parameters. This allows x and y to be different in nature (e.g. video and audio). A predictor module predicts the representation of y from the representation of x. The predictor may depend on a latent variable z. The energy is simply the prediction error in representation space:

$$E_w(x, y, z) = D(s_y, \text{Pred}(s_x, z)) \tag{10}$$

The overall energy is obtained by minimizing over z:

$$\check{z} = \operatorname*{argmin}_{z \in \mathcal{Z}} E_w(x, y, z) = \operatorname*{argmin}_{z \in \mathcal{Z}} D(s_y, \text{Pred}(s_x, z)) \tag{11}$$

$$F_w(x, y) = \min_{z \in \mathcal{Z}} E_w(x, y, z) = D(s_y, \text{Pred}(s_x, \check{z})) \tag{12}$$

$$\tag{13}$$

The main advantage of JEPA is that *it performs predictions in representation space*, eschewing the need to predict every detail of y. This is enabled by the fact that the encoder of y may choose to produce an abstract representation from which irrelevant details have been eliminated.

But there are two ways a JEPA may represent the multiplicity of values of y compatible with x. The first one is invariance properties of the y encoder, the second one is the latent variable z, as explained below.

multi-modality through encoder invariance: The encoder function $s_y = \text{Enc}(y)$ may have invariance properties. If all the y's in a set map to the same value of s_y, all those y's will have identical energies. With JEPA, we lose the ability to generate outputs, but we gain a powerful way to represent multi-modal dependencies between inputs and outputs.

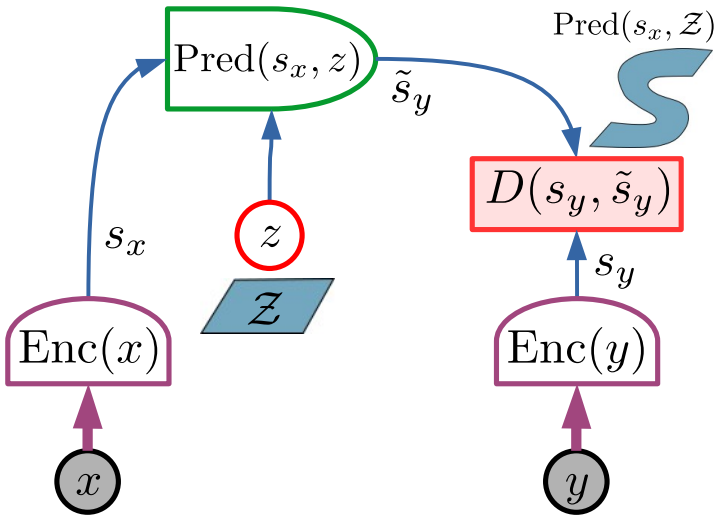

Figure 12: *The Joint-Embedding Predictive Architecture (JEPA) consists of two encoding branches. The first branch computes s_x, a representation of x and the second branch s_y a representation of y. The encoders do not need to be identical. A predictor module predicts s_y from s_x with the possible 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.

multi-modality through latent variable predictor: The predictor may use a latent variable z to capture the information necessary to predict s_y that is not present in s_x. When z is varied over a set $\mathcal{Z}$, the predictor produces a set of plausible predictions $\text{Pred}(s_x, \mathcal{Z}) = \{\tilde{s}_y = \text{Pred}(s_x, z)\ \forall z \in \mathcal{Z}\}$. For example, if x is a video clip of a car approaching a fork in the road, s_x and s_y may represent the past and future positions, orientations, velocities and other characteristics of the car, ignoring irrelevant details such as the trees bordering the road or the texture of the sidewalk. The latent z may be a binary variable indicating whether the car takes the left branch ($z = 0$) or the right branch ($z = 1$ if the road. If the car takes the left branch, the value $z = 0$ will produce a lower energy $D(s_y, \tilde{s}_y)$ than $z = 1$.

4.5 Training a JEPA

Like any EBM, a JEPA can be trained with contrastive methods. But, as pointed out above, contrastive methods tend to become very inefficient in high dimension. The relevant

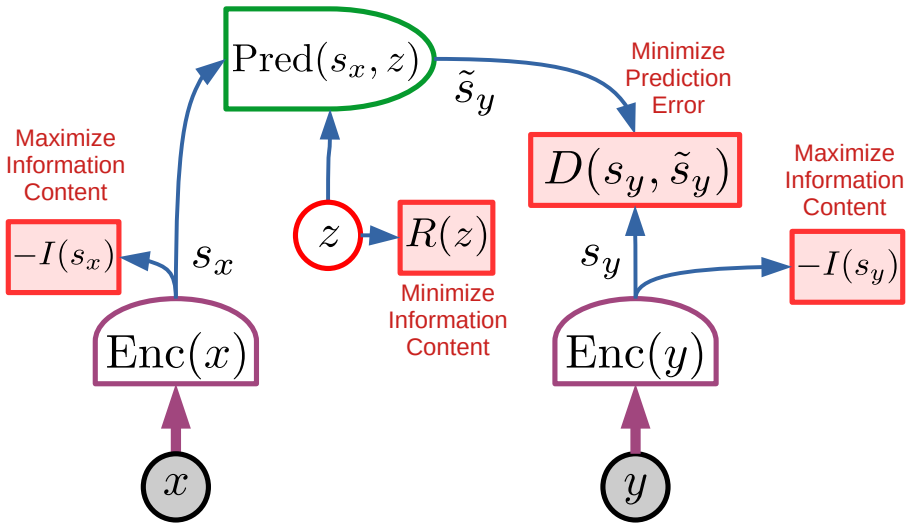

Figure 13: *Non-contrastive training of JEPA.*
The main attraction of JEPAs is that they can be trained with non-contrastive methods. The basic principle of such training is that (1) s_x should be maximally informative about x; (2) s_y should be maximally informative about y; (3) s_y should be easily 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.

dimension here is that of s_y, which may be considerably smaller than y, but still too high for efficient training.

What makes JEPAs particularly interesting is that we can devise *non-contrastive methods* to train them. As explained in section 4.3, non-contrastive methods use regularizers that measure the volume of space that can take low energy values. In the case of the JEPA, this can be done through four criteria, as depicted in Figure 13:

1. maximize the information content of s_x about x
2. maximize the information content of s_y about y
3. make s_y easily predictable from s_x
4. minimize the information content of the latent variable z used in the prediction.

Criteria 1 and 2 prevent the energy surface from becoming flat by *informational collapse*. They ensure that s_x and s_y carry as much information as possible about their inputs. Without these criteria the system could choose to make s_x and s_y constant, or weakly informative, which would make the energy constant over large swaths of the input space.
Criterion 3 is enforced by the energy term $D(s_y, \tilde{s}_y)$ and ensures that y is predictable from x *in representation space*.
Criterion 4 prevents the system from falling victim to another type of informational collapse by forcing the model to predict s_y with as little help from the latent as possible. This

type of collapse can be understood with the following thought experiment. Imagine that z has the same dimension as s_y. Assume that the predictor is a parameterized function (e.g. a neural net) that can choose to ignore s_x and to simply copy z on its output $\tilde{s}_y = z$. For any s_y it is possible to set $\check{z} = s_y$, which would make the energy $D(s_y, \tilde{s}_y)$ zero. This corresponds to a totally flat and collapsed energy surface.
How do we prevent this collapse from happening?
By limiting or minimizing the information content of the latent variable.
How can this be done?
By making z discrete, low-dimensional, sparse, or noisy, among other methods.

A few concrete examples may help build an intuitive understanding of the phenomenon. Suppose that $D(s_y, \tilde{s}_y) = ||s_y - \tilde{s}_y||^2$ and that z is discrete with K possible integer values $[0, K-1]$. For a given x, there can be only K possible values of $\tilde{s}_y$:

$$\text{Pred}(s_x, 0), \text{Pred}(s_x, 1), \ldots, \text{Pred}(s_x, K-1).$$

Hence, these can be the only values of s_y with zero energy, and there are only K of them. Consider a point s_y that starts from $\text{Pred}(s_x, 0)$ and moves towards $\text{Pred}(s_x, 1)$. Its energy will start from zero, increase quadratically as s_y moves away from $\text{Pred}(s_x, 0)$, until s_y. When s_y becomes closer to $\text{Pred}(s_x, 1)$ than to $\text{Pred}(s_x, 0)$, the energy will decrease, reaching zero when s_y reaches $\text{Pred}(s_x, 1)$. In representation space, the energy will be the minimum of K quadratic energy wells.

Similarly, imagine that z is a vector whose dimension d is lower than that of $\tilde{s}_y$. Then, assuming that $\text{Pred}(s_x, z)$ is a smooth function of z, the set of possible predictions will be at most a d-dimensional manifold in the space of s_y.

More to the point, imagine that the energy function is augmented by a regularization term on z of the form $R(z) = \alpha \sum_{i=1}^{d} |z_i|$, i.e. the L_1 norm of z. This will drive $\check{z}$ to be sparse. As with classical sparse coding, this will cause the region of low energy to be approximated by a union of low-dimensional manifolds (a union of low-dimensional linear subspaces if $\text{Pred}(s_x, z)$ is linear in z), whose dimension will be minimized by the L_1 regularizer.

Making z a stochastic sample from a distribution whose entropy is maximized will also have a proper regularization effect. This is the basis of Variational Auto-Encoders and similar models.

A more complete discussion of regularizers that can minimize the information content of latent variables is beyond the scope of this paper. For now, we can mention four classes of methods: discretization/quantification (e.g. as in VQ-VAE (Walker et al., 2021), dimensionality/rank minimization (e.g. as in Implicit Rank-Minimizing AE (Jing et al., 2020), sparsification (as in linear sparse modeling (Olshausen and Field, 1996), LISTA (Gregor and LeCun, 2010b), and non-linear sparse modeling (Evtimova and LeCun, 2022)), and fuzzyfication (as in noisy AE (Doi et al., 2007), VAE (Kingma and Welling, 2013), and variants used in control problems (Henaff et al., 2019)).

The ability of the JEPA to predict in representation space makes it considerably preferable to generative models that directly produce a prediction of y. In a video prediction scenario, it is essentially impossible to predict every pixel value of every future frame. The details of the texture on a carpet, the leaves of a tree moving in the wind, or the ripples on a pond, cannot be predicted accurately, at least not over long time periods and not without consuming enormous resources. A considerable advantage of JEPA is that *it can choose to*

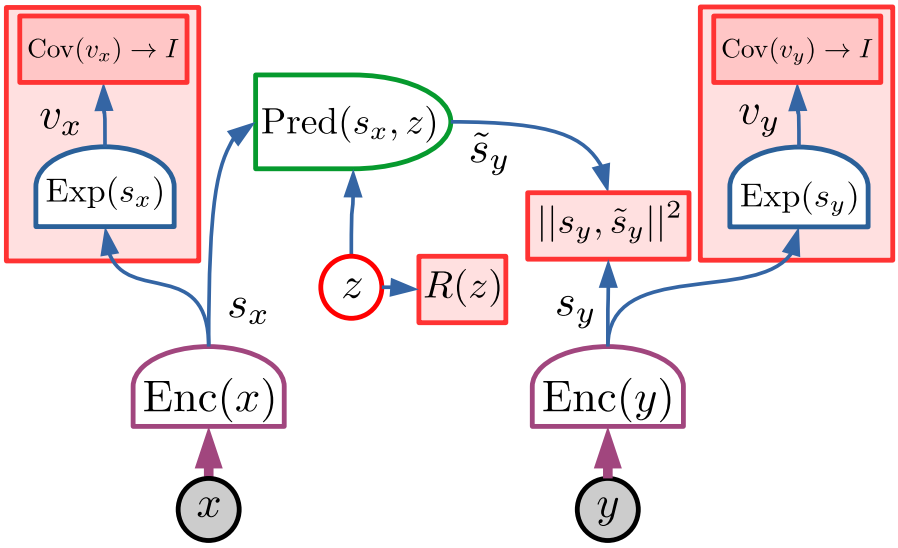

Figure 14: *Training a JEPA with VICReg.*
VICReg is a non sample-contrastive method for training embeddings. The information content of the representations s_x and s_y is maximized by first mapping them to higher-dimensional embeddings v_x and v_y through an expander (e.g. a trainable neural net with a few layers). The loss function drives the covariance matrix of the embeddings towards the identity (e.g. computed over a batch).
VICReg can be seen as a dimension-contrastive method *as opposed to sample-contrastive methods.*

ignore details of the inputs that are not easily predictable. Yet, Criteria 1 and 2 will ensure that the information content of the ignored details are kept to a minimum.

How can we implement Criteria 1 and 2?
In other words, given a parameterized deterministic encoding function $s_y = \text{Enc}_w(y)$, how do we maximize the information content of s_y?
If $\text{Enc}_w(y)$ is invertible, s_y contains all the information about y, but that may be sub-optimal for Criterion 3, as s_y will contain many irrelevant or hard-to-predict details about y. More precisely, s_y is maximally informative about y if the function $\text{Enc}_w(y)$ is minimally surjective, i.e. if the volume of sets of y that map to the same s_y is minimal. The same reasoning applies to the x encoder. To turn this criterion into a differentiable loss, we need to make some assumptions.

4.5.1 VICReg

The VICReg method (Bardes et al., 2021) makes a few assumptions about the distributions of s_x and s_y. A graphical representation is shown in Figure 14. To maximize the information content of s_x, VICReg uses the following two sub-criteria: (1) the components of s_x must not be constant, (2) the components of s_x must be as independent of each other as possible. This is approximated by first non-linearly mapping s_x and s_y to higher-dimensional embeddings v_x and v_y through a trainable expander module (e.g. a neural net with a few layers), and using a loss function with two differentiable loss terms computed over a batch of samples:

1. **Variance**: a hinge loss that maintains the standard deviation of each component of s_y and v_y above a threshold over a batch.

2. **Covariance**: a covariance loss in which the covariance between pairs of different components of v_y are pushed towards zero. This has the effect of decorrelating the components of v_y, which will in turn make the components of s_y somewhat independent.

The same criteria are applied to s_x and v_x separately.

The third criterion of VICReg is the representation prediction error $D(s_y, \tilde{s}_y)$. In the simplest implementations of VICReg, the predictor is constant (equal to the identity function), making the representations **invariant** to the transformation that turns x into y. In more sophisticated versions, the predictor may have no latent variable, or may depend on a latent variable that is either discrete, low dimensional, or stochastic.

The fourth criterion is necessary when the predictor uses a latent variable whose information content must be minimized, for example a vector whose dimension approaches or surpasses that of $\tilde{s}_y$.

A simple instantiation of VICReg to learn invariant representations consists in making x and y be different views (or distorted versions) of the same content, setting the predictor to the identity function, and defining $D(s_y, \tilde{s}_y) = D(s_y, s_x) = ||s_y - s_x||^2$.

Inferring the latent variable through gradient-based methods may be onerous. But the computational cost can be greatly reduced by using amortized inference, as explained in Appendix 8.3.3.

While contrastive methods ensure that representations of different inputs in a batch are different, VICReg ensures that different components of representations over a batch are different. VICReg is contrastive over components, while traditional contrastive methods are contrastive over vectors, which requires a large number of contrastive samples.

But the most promising aspect of JEPA trained with VICReg and similar non-contrastive methods is for *learning hierarchical predictive world models*, as we examine in the next section.

4.5.2 Biasing a JEPA towards learning "useful" representations

With the training criteria listed above, the JEPA finds a trade-off between the completeness and the predictability of the representations. What is predictable and what does not get represented is determined implicitly by the architectures of the encoders and predictor. They determine a inductive bias that defines what information is predictable or not.

But it would be useful to have a way to bias the system towards representations that contain information relevant to a class of tasks. This can be done by adding prediction heads that take $\tilde{s}_y$ as input and are trained to predict variables that are easily derived from the data and known to be relevant to the task.

4.6 Hierarchical JEPA (H-JEPA)

JEPA models trained non-contrastively may constitute our best tool for learning world models that are able to learn relevant abstractions. When trained with VICReg and similar

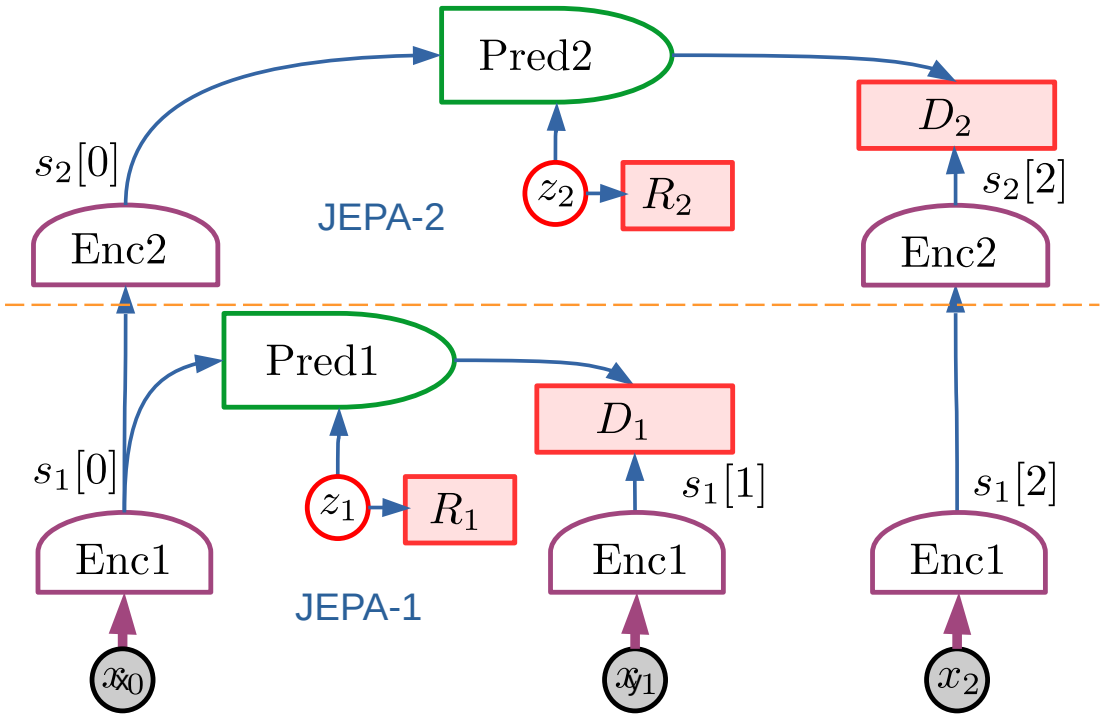

Figure 15: *Hierarchical JEPA (H-JEPA)*
The ability of the JEPA to learn abstract representations in which accurate prediction can be performed allows hierarchical stacking. In this diagram JEPA-1 extracts low-level representations and performs short-term predictions. JEPA-2 takes the representations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

criteria, a JEPA can choose to train its encoders to eliminate irrelevant details of the inputs so as to make the representations more predictable. In other words, a JEPA will learn abstract representations that make the world predictable. Unpredictable details will be eliminated by the invariance properties of the encoder, or will be pushed into the predictor's latent variable. The amount of information thereby ignored will be minimized by the training criteria and by the latent variable regularizer.

It is important to note that *generative latent-variable models are not capable of eliminating irrelevant details*, other than by pushing them into a latent variable. This is because they do not produce abstract (and invariant) representations of y. This is why we advocate *against* the use of generative architectures.

The capacity of JEPA to learn abstractions suggests an extension of the architecture to handle prediction at multiple time scales and multiple levels of abstraction. Intuitively, low-level representations contain a lot of details about the input, and can be used to predict in the short term. But it may be difficult to produce accurate long-term predictions with the same level of details. Conversely high-level, abstract representation may enable long-term predictions, but at the cost of eliminating a lot of details.

Let's take a concrete example. When driving a car, given a proposed sequence of actions on the steering wheel and pedals over the next several seconds, drivers can accurately predict the trajectory of their car over the same period. The details of the trajectory over longer periods are harder to predict because they may depend on other cars, traffic lights, pedestrians, and other external events that are somewhat unpredictable. But the driver can still make accurate predictions at a higher level of abstraction: ignoring the details of trajectories, other cars, traffic signals, etc, the car will probably arrive at its destination within a predictable time frame. The detailed trajectory will be absent from this level of description. But the approximate trajectory, as drawn on a map, is represented. A discrete latent variable may be used to represent multiple alternative routes.

Figure 15 shows a possible architecture for multilevel, multi-scale world state prediction. Variables x_0, x_1, x_2 represent a sequence of observations. The first-level network, denoted JEPA-1 performs short-term predictions using low-level representations. The second-level network JEPA-2 performs longer-term predictions using higher-level representations. One can envision architectures of this type with many levels, possibly using convolutional and other modules, and using temporal pooling between levels to coarse-grain the representation and perform longer-term predictions. Training can be performed level-wise or globally, using any non-contrastive method for JEPA.

I submit that the ability to represent sequences of world states at several levels of abstraction is essential to intelligent behavior. With multi-level representations of world states and actions, a complex task can be decomposed into successively more detailed sub-tasks, instantiated into actions sequences when informed by local conditions. For example, planning a complex task, like commuting to work, can be decomposed into driving to the train station, catching a train, etc. Driving to the train station can be decomposed into walking out of the house, starting the car, and driving. Getting out of the house requires standing up, walking to the door, opening the door, etc. This decomposition descends all the way down to millisecond-by-millisecond muscle controls, which can only be instantiated when the relevant environmental conditions are perceived (obstacles, traffic lights, moving objects, etc).

4.7 Hierarchical Planning

If our world model can perform predictions hierarchically, can it be used to perform Mode-2 reasoning and planning hierarchically?

Hierarchical planning is a difficult topic with few solutions, most of which require that the intermediate vocabulary of actions be predefined. But if one abides by the deep learning philosophy, those *intermediate representations of action plans should also be learned.*

Figure 16 shows a possible architecture for hierarchical Mode-2 planning that can exploit the hierarchical nature of a multi-scale world model.

A percept is encoded into representations at multiple levels of abstractions by a cascade of encoders:

$$s[0] = \text{Enc1}(x); \;\; s2[0] = \text{Enc2}(s[0]); \; \ldots \tag{14}$$

Prediction takes place at all levels. Higher levels perform longer-term prediction, while lower levels perform shorter-term predictions. The overall task is defined by a high-level objective, depicted as $C(s_2[4])$ in the diagram. The top level infers a sequence of high-level actions

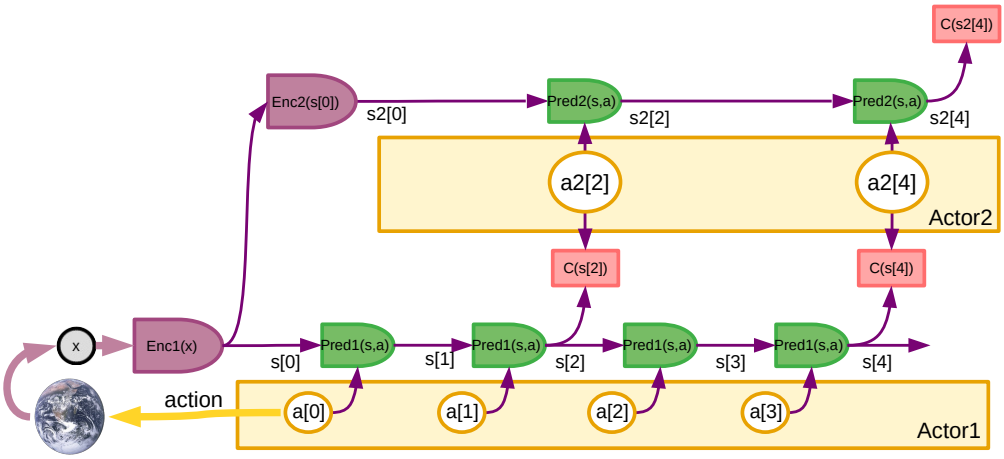

Figure 16: *Hierarchical JEPA for Mode-2 hierarchical planning.*
A complex task is defined by a high-level cost computed from a high-level world-state representation $C(s2[4])$*. A sequence of high-level abstract actions* $(a2[2], a2[4])$ *is inferred that minimizes* $C(s2[4])$*. The inferred abstract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

$(a2[2], a2[4])$ to optimize this objective. These high-level "actions" are not real actions but targets for the lower level predicted states. One can think of them as conditions that the lower-level state must satisfy in order for the high-level predictions to be accurate. Whether these conditions are satisfied can be computed by cost modules $C(s[2])$ and $C(s[4])$. They take a lower-level state $s[2]$ and a high-level condition $a2[2]$ and measure to what extent the state satisfies the condition. With these subgoals defined, the lower level can perform inference and find a low-level action sequence that minimizes the mid-level subgoals $C(s[2])$ and $C(s[4])$.

The process just described is top down and greedy. But one may advantageously iterate the optimization so that high level and low-level action sequences are optimized jointly. The cost modules may be configured by the configurator for the situation at hand.

The idea that an action is merely a condition to be satisfied by the level below is actually an old one in control theory. For a example, a classical proportional servomechanism can be seen as being given a target state. A quadratic cost measures the squared distance between the target and the current state, and the control is simply proportional to the negative gradient of the cost with respect to the action variables.

4.8 Handling uncertainty

The real world is not entirely predictable. Uncertainty in predictions of future world states may be due to a number of reasons:

- the world is intrinsically stochastic (aleatoric uncertainty, type 1)