A Path Towards Autonomous Machine Intelligence

by Jacob Posz

Overview

JEPA

In this paper, Yann LeCun proposes how he thinks we should reach machine intelligence.

LeCun's proposal is centered around JEPA. As stated by LeCun, "JEPA models learn high-level representations that capture the dependencies between two data points, such as two segments of video that follow each other. JEPA replaces contrastive learning with “regularized” techniques that can extract high-level latent features from the input and discard irrelevant information."

Problem

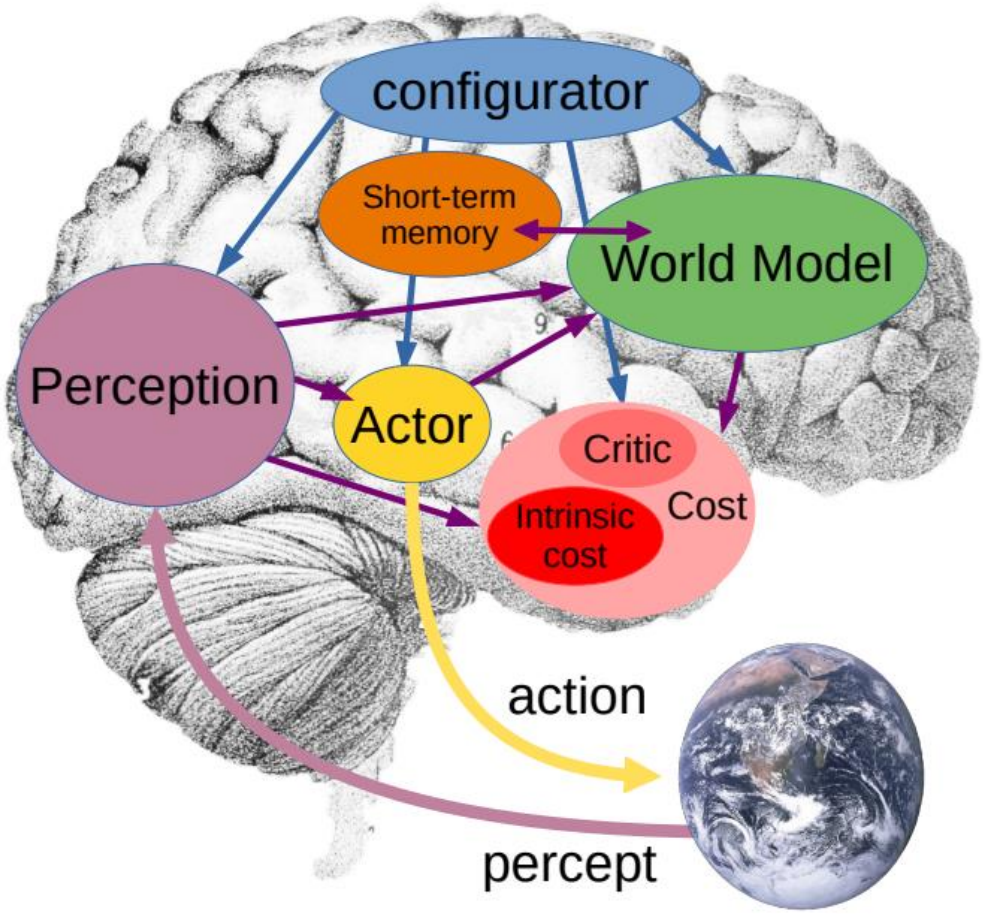
The main issue that LeCun states in this paper is that "ML models severely lack the ability to adapt to a new situation they haven’t seen before." For example, he discusses how an adolescent can learn to drive a car in hours while ML models require a very large iteration of training so they know how to respond in the rarest of situations.

Main contributions

As stated in the paper, the main contributions of this paper include:

1. An overall cognitive architecture in which all modules are differntiable and many of them are trainable
2. JEPA and Hierarchical JEPA: a non-generative architecture for predictive worl models that learn a hierarchy of representations
3. A non cntrastive self-supervised learning paradigm that produces represenations that are simultaneously informative and predictable
4. A way to use H-JEPA as the basis of predictive world models for hierarchical planning under uncertainty

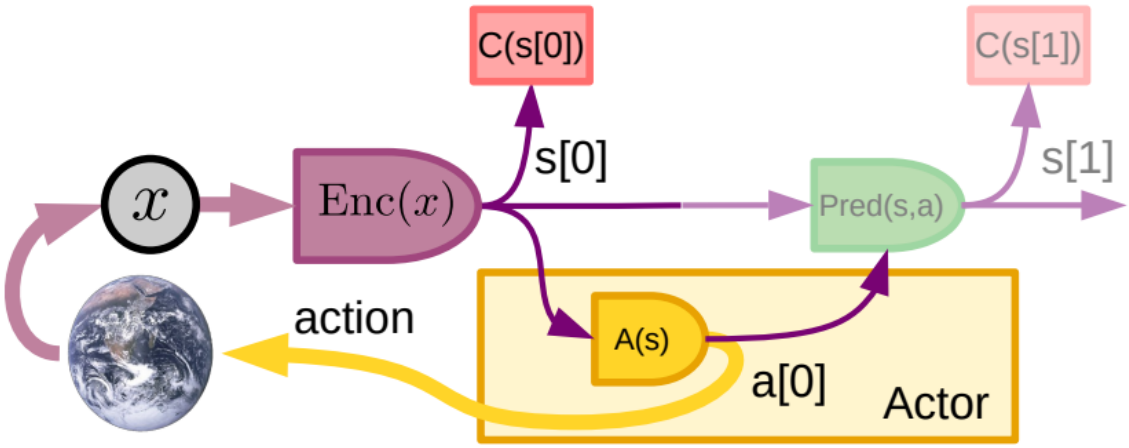
Entire proposed architecture



- The world module is the center piece of the model that predicts the state of the world forward in time
- The actor module interacts with the world model and is what performs the action; the actor can also act inside of the world module in essentially a simulated reality to plan forward or it could interact with the world model to find the best action
- the short-term memory is going to be used to train the world model and the the "critic" (things that happen in the world that will be stored in the short-term memory)
- The perception module takes whatever the world gives it and makes it available as a representation or as a perception
- The configurator is the "master module" that configures all the other modules depending on what situation they're in

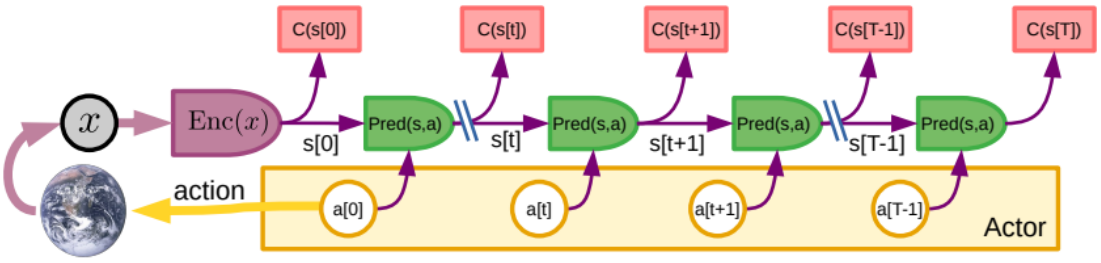
World Module

Mode-1 perception-action episode



- Mode 1 is reactive; you simply go from perception of the world to action without much thought
- We start with the world and get an observation x then put it through the encoder Enc (which is also the perception module) which will give us a latent representation
- Different paths emerge, although only one is important: the path that goes to the actor A -> the actor then sends back an action to the world
- The other path leads to the cost module, C, which tells us the cost of something -> we can compute it, however, in this basic loop, the actor has already been trained to act on a perception
- At inference time, the actor doesn't need to look at the cost anymore in order to act

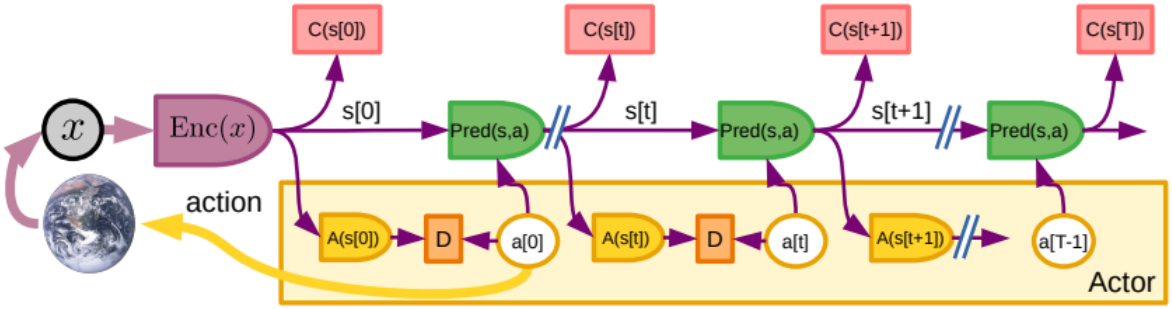
Mode-2 perception-action episode



- Again, we have an input x and put it through the encoder Enc -> however, now, we are going to roll out the world module across different time steps
- The actor a is going to take the state it recieves from the encoder and propse an action (this is the same actor as before) -> we can then use this and put it back into the world module along with the latent prediction
- The predictor Pred takes whatever comes out of the encoder; in other words, it takes a latent state of the world and it predicts the next latent state of the world (hence, this is why LeCun calls this "non-generative")
- We can now give the actor the representation; if it proposes an action, we can use the world module to predict the next state
- From the next state, we can ask the actor for an action -> the actor gives us an action and we can predict the next state
- We can optimize all of these actions at inference time using gradient descent -> that is, we can improve the actions, a, using gradient descent, through all of the modules until we have completely optimized the action sequence -> which means, the very first action, a[0], is hopefully a much better action than the action first proposed by the naive actor -> we can then take this action and feed it back to the world as an action

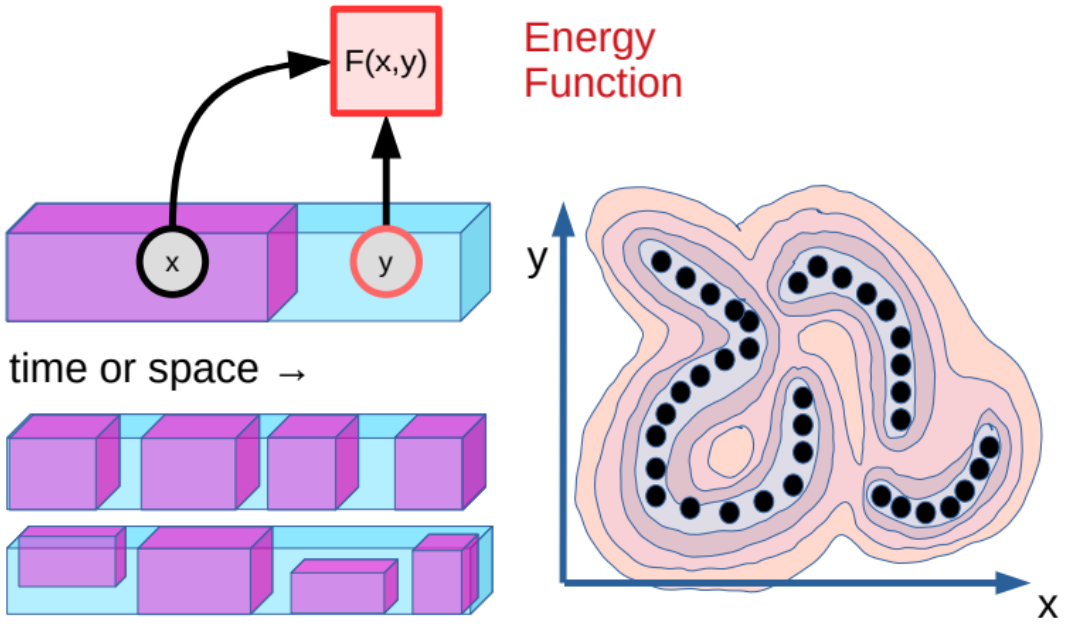
you can include the costs, C, which you can have after every step... thees don't have to be optimization; they can also be search, evolutionary search, tree search, etc. -> essentially, anything that actually tries to improve the action sequence at inference time

Training a reactive module from the result of Mode-2 reasoning



- Actions, a, are what we have come up with through this optimization process
- Everything's differentiable, so you can train the actor to essentially match the best actions because if you have a good world model, you can improve the low level actor, A(s[0]), and at some point, the initial action sequence will already be close to optimal

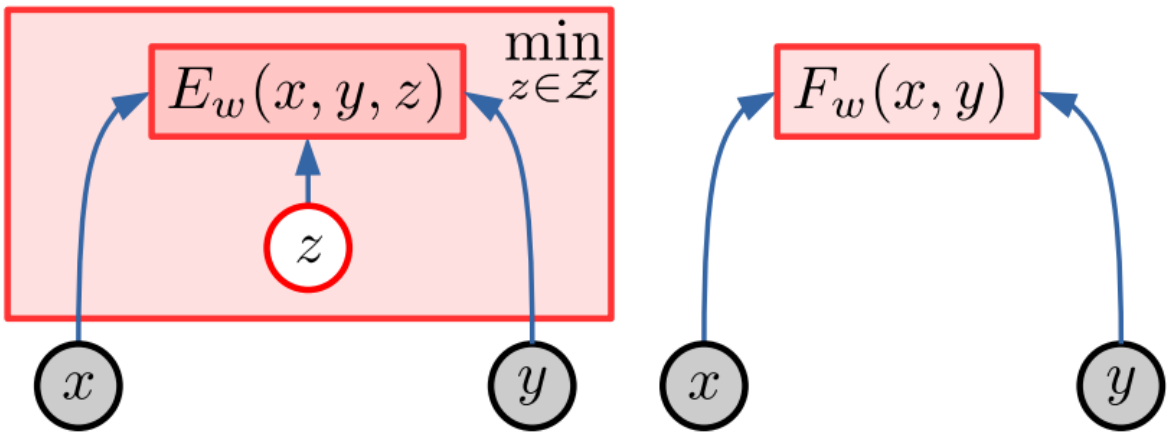
Self-Supervised Learning (SSL)



- You have a piece of data (which is x + y) and you mask out the right hand side (y) and then you use what you do know (x) and you try to predict the thing you don't know

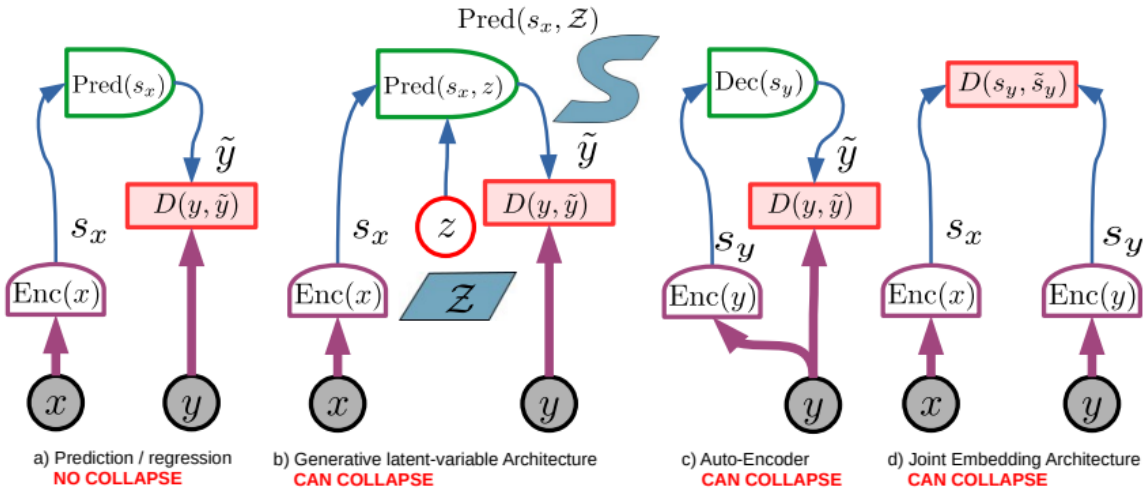
- You don't want to predict the thing you don't know but instead, create an energy function -> an energy function tells you how well x and y fit together
- You want to train a system that sees the data space in this format, which is going to be an energy landscape
- we don't need to predict y from x directly but instead train the energy function -> the energy function could assign a low value to any possible continuation as long as it assigns a high value everywhere else

Latent-Variable Energy-Based Model (LVEBM)



- Same formula as before; we have an x and a y and an energy function -> Ew that tells us how well x and y are compatible with each other (which is also Fw)
- However, there could be many y's that are possible for a given x
- Just by looking at x, we can't tell whcih of the y's is compatible -> thats why we introduce a latent variable, z -> z captures all of the information about y that isnt directly in x... for example, if we have a car that has the option of turning left or right, this decision would be represented by z
- if we have an x and a y, in order to compute that energy that tells us how well the two are compatible, we need to minimize over (capital) Z
- So, we could infer z or if we have this model trained then if we have an x, we could actually sample some z values in order to produce different possibilities of y -> this gives us a lot of freedom to handle uncertainty in the world or unobserved structure in the world

Architectures and their capacity for collapse



a) Prediction / regression - No Collapse

D represents the energy/compatibility function

- We have an encoder for x -> this gives us a latent representation, sx
- Then, we use a predictor module in order to predict y
- We'll predict y~ directly then compare it with the true y -> there'll be a difference between them
- This cannot collapse becasue we need to predict the actual y

b) Generative latent-variable Architecture - Can Collapse

- We have an encoder for x -> this gives us a latent representation, sx -> but now we introduce z which can vary over a certain domain -> this gives us a domain that we can control for the output of the predictor
- If we now try to predict y from z and x, we can set z equal to y and we'd be good -> otherwise, this can collapse

c) Auto-Encoder - Can Collapse

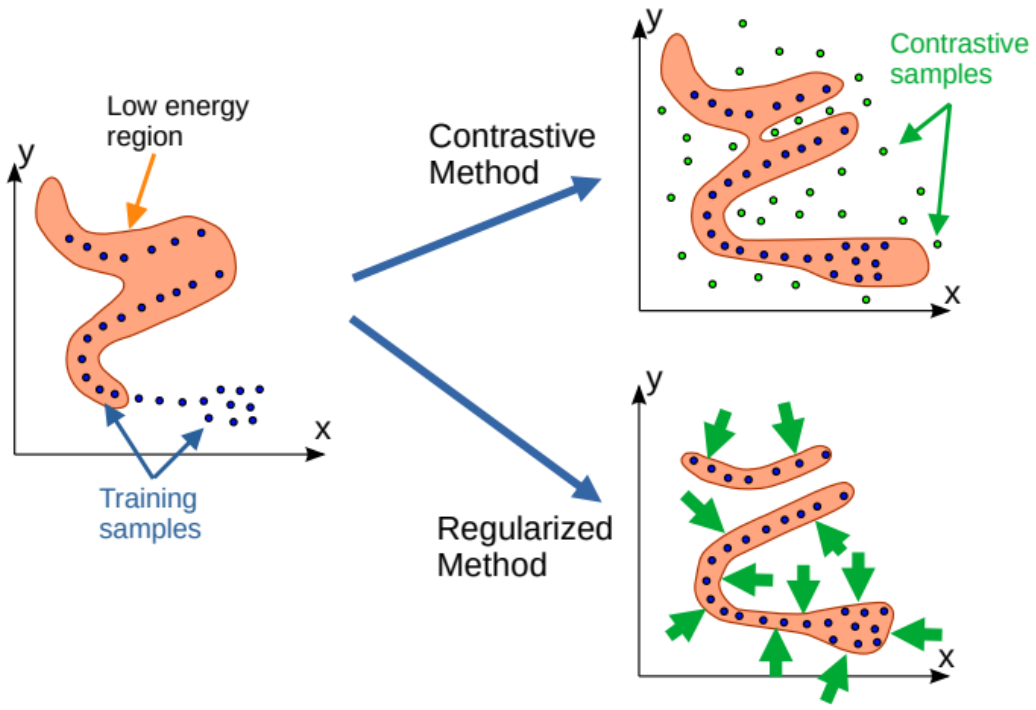
- This is the same as the first architecture, except only y goes into the encoder, instead of both x and y
- After going through the encoder, y gets a latent representation, sy, then goes through a decoder that gives back an estimation of oneself -> so, this can collapse easily

d) Joint Embedding Architecture - Can Collapse

- We have an encoder for x and an encoder for y (these could be the same but don't have to be) -> this will give us 2 latent representations, sx and sy -> then, we use an energy function to compute how well these 2 latent representations fit together, possibly with the help of a latent variable
- If the encoders always output constant vectors for both x and y, we'll always be good -> however, this can collapse if they are different

so, how do we design the loss to prevent collapse? -> there are two approaches: constrastive methods and regularized methods

Constrative Methods Versus Regularized Methods



Contrastive Methods

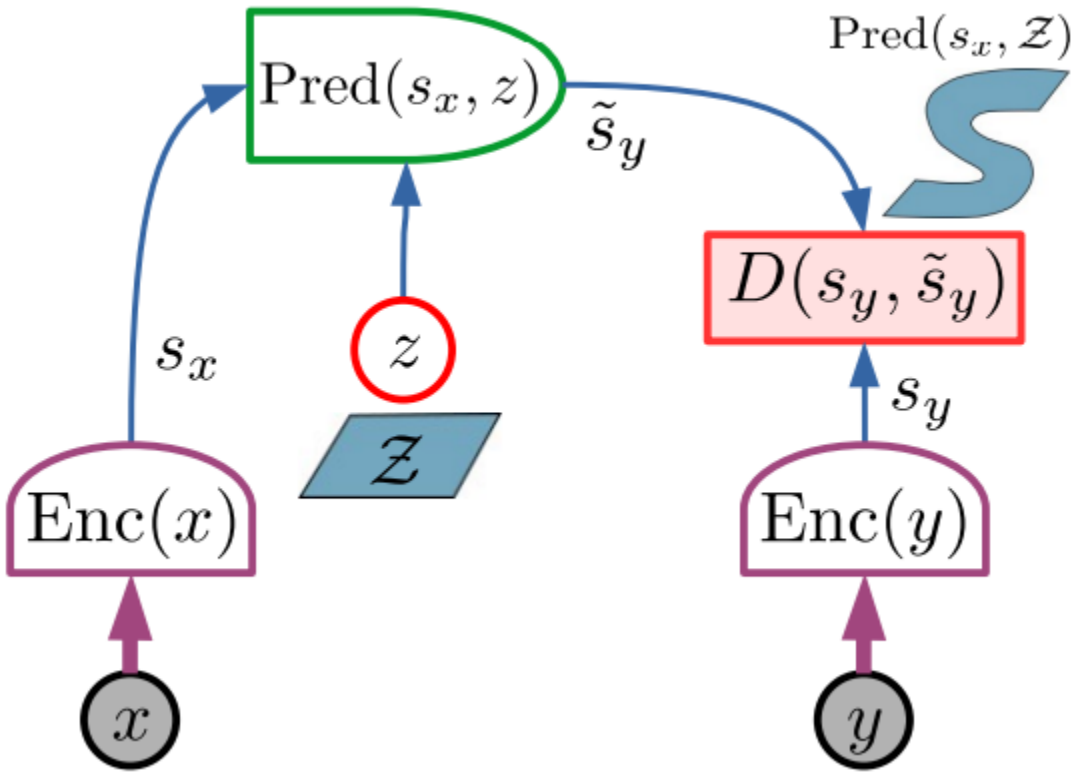
- Many self supervised image training procedures are contrastive -> they'll have an image -> then they'll make 2 variations of that image -> then, they'll take a third image from the database and will make a variation of that -> then, they use embedding models to embed all the variations -> this will give a data point somewhere in high dimensional space

- You then try to pull the 2 variations from the same image together and push the variations from different images apart
- Contrastive training relies on you coming up with these training/negative samples
- this quickly runs into problems (curse of dimensionality),therefore LeCun advertises for something different: regularized methods

Regularized Methods

- have other means of restricting space that is a low energy region
- you enforce/encourage the system to keep the region where the energy is low very small -> this is done through regularization

The JEPA Architecture



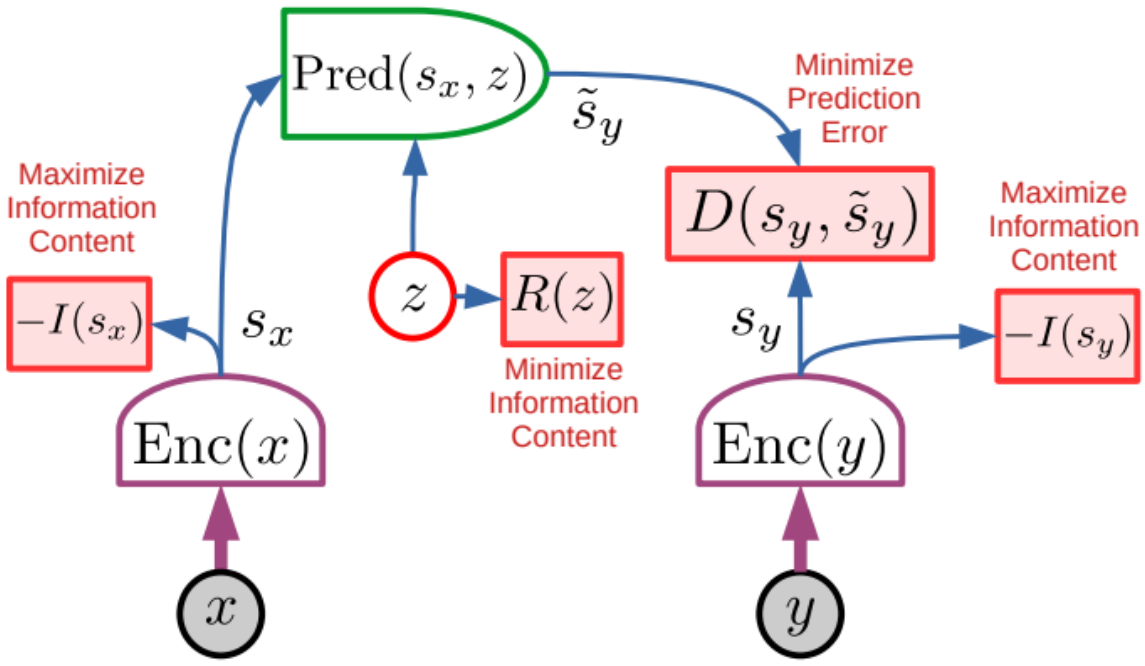
- This is the basic module
- it looks similar to the Generative latent-variable Architecture model but is slightly different

- we have our x and our y and we want to check if they're compatible with each other
- we'll embed both of them using deterministic encoders which gives us latent representations, sx & sy
- we then use the predictor to predict the latent representation of y, ~sy from the representation of x
- z in this case controls which latent representation we're getting -> so, z can vary over the domain (captal) Z which then leads the s~y to vary over the domain, (capital) S which then goes into the energy function, D regardless of whether the representation of sy is compatible with the predicted representation of y, ~sy
- As stated by LeCun, "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" -> obviously, this means this will be subject to collapse
- these encoders could just throw away everything that's not relevant about x and y because we never need to predict y directly

So how do we train a model like this?

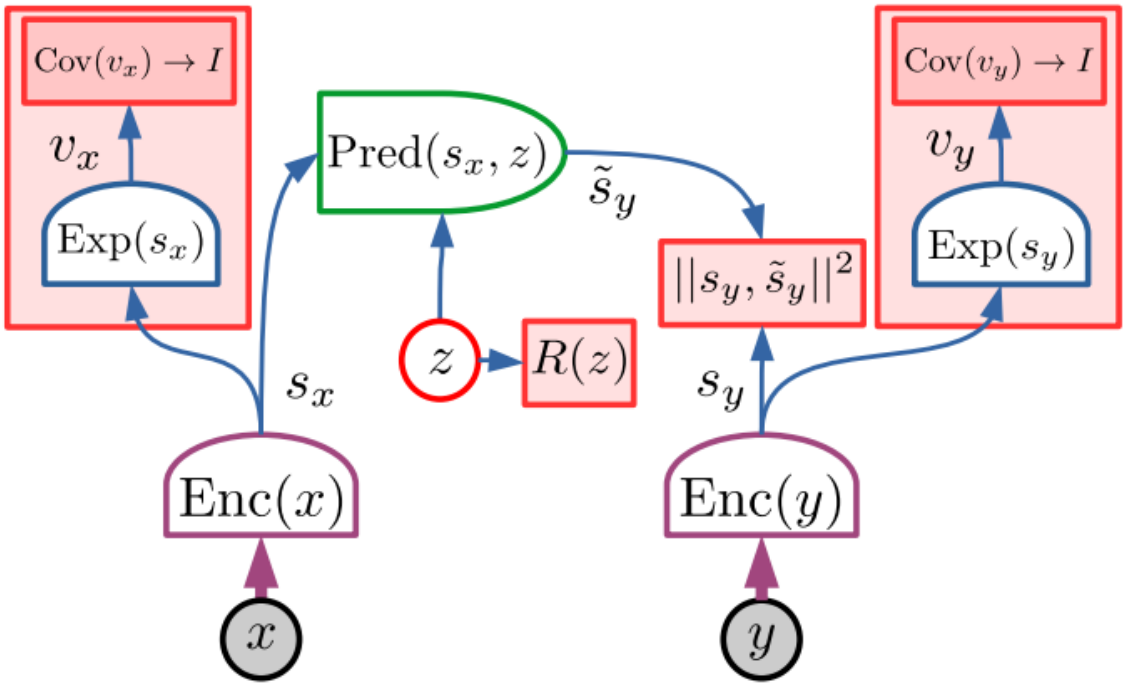
Here is where regularization comes in

Non-contrastive training of JEPA



- We obviously train it by minimizing the predictive error, D
- we actually want to predict the latent representation of y, sy from the latent representation of x, sx to predict ~sy
- In this case, we have a couple of regularizers to prevent collapse:
- **first of all, we regularize z by minimizing its information content**
- **next, are the regularizors (-I) on the information content of the latent representations, sy & sx** -> we want to maximize the information content of -I
- essentially, this just means that if sx or sy always has the same value, it doesn't have much information inside of it -> so for example, we could use a mini batch approach

Training a JEPA with VICReg



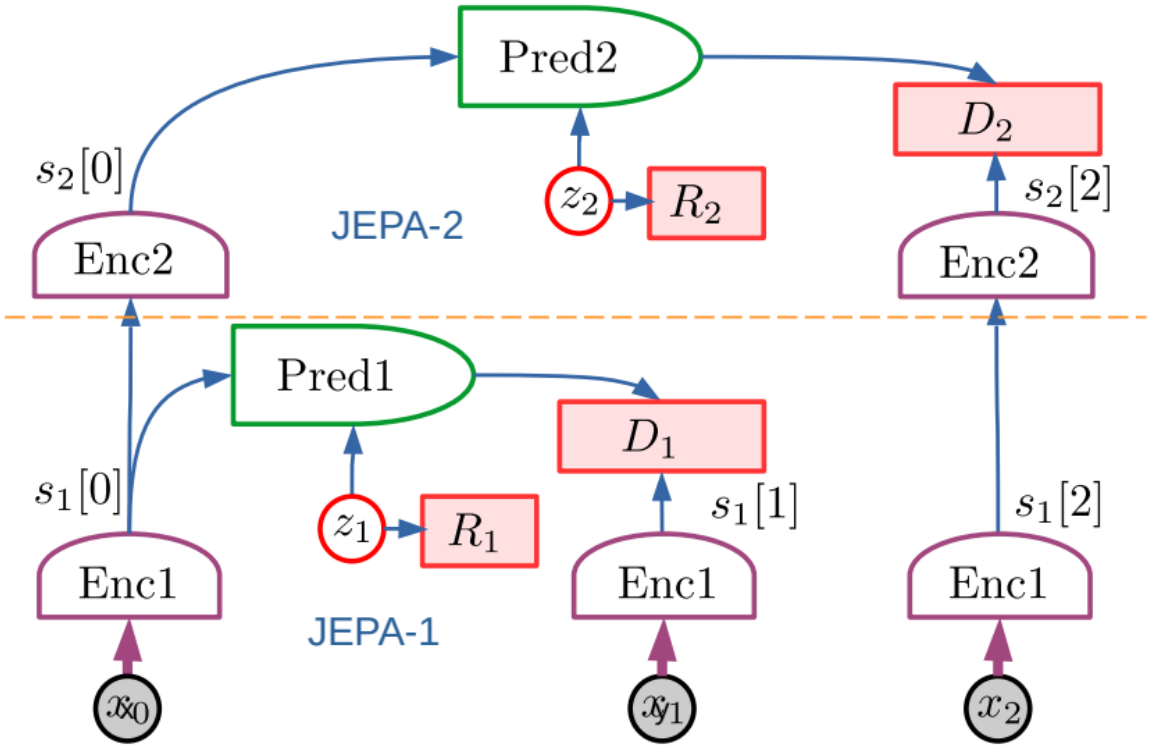
- the maximizing of the information content of the latent representation, sy, is done via regularizing the covariance matrix, vx

As stated by LeCun, he suggests that we could also bias a JEPA towards learning "useful" representations, stating, "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 ˜sy as input and are trained to predict variables that are easily derived from the data and known to be relevant to the task."

He also states that in addition, you could also attach some kind of prediction head to ~sy

So, what can we do with this? -> we can arrange it in a hierarchical fashion

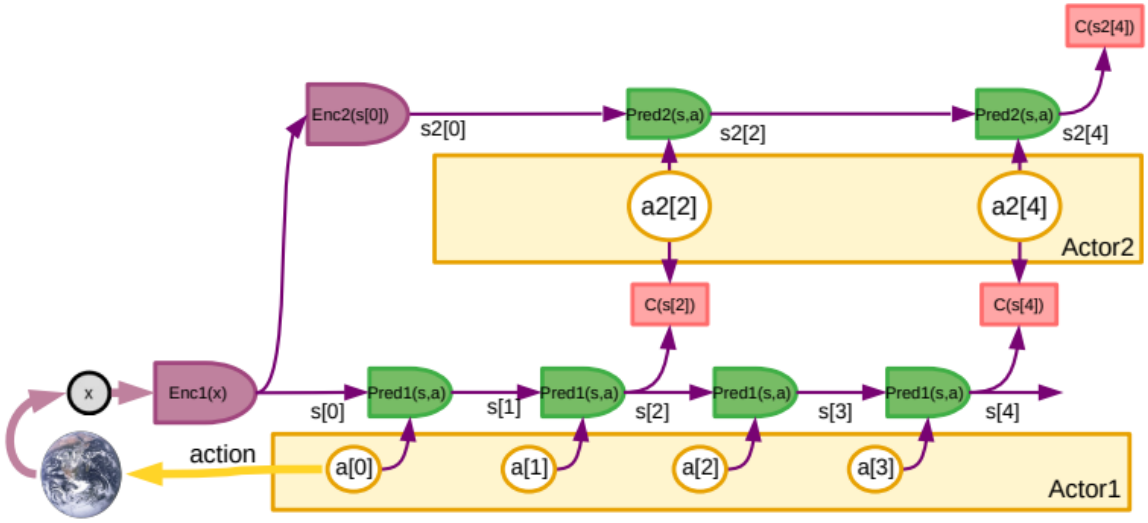
Hierarchical JEPA (H-JEPA)



- The lower level predicts over short time frames while the higher level predicts over longer time frames
- the latent representation, s2 is obtained from the latent representation, s1, by a second encoder -> it then makes predictions over a longer period of time

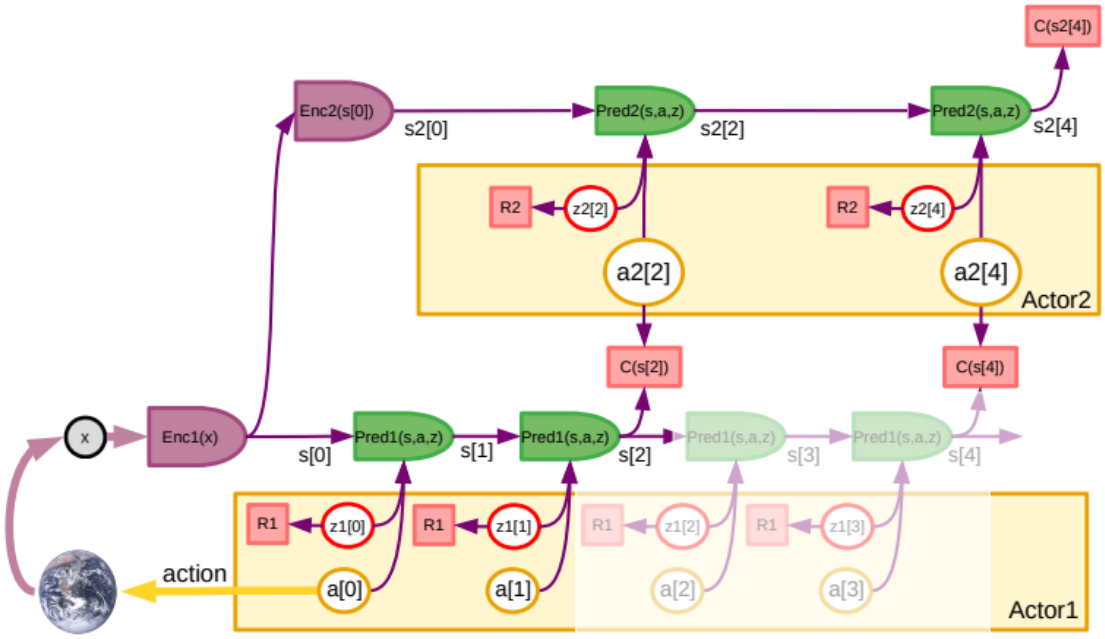
We can use the H-JEPA model to perform hierarchical planning:

Hierarchical JEPA for Mode-2 hierarchical planning



- x is the state of the world and we know that at some point, we have a desired outcome (cost function/reward), C
- if we've trained such a multi-layered predictive model in latent space, we're going to ask this high level actor (Actor 2) to give us high level actions, a2 -> we can then roll out the world model with it until we reach C(s2) -> we can use an optimization teqnique (like back propogation) in order refine the high level actions, a2, as best we can -> now, we have targets, a2, for these low levels actions, a1
- the rewards on the lower level, C(s) are how well we match the targets that are given by the higher level
- we could also optimize all of the different levels together until we have the optimal sequence of lower level and higher level actions in order to reach our goal, C(s2) -> at this point, we can be very confident than our first action, a[0], will serve us well -> we can send this to the world
- we then get the next state and repeat the same process

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



- if you are in an uncertain environment, you could introduce latent variables, z2 & R2, which you can infer so if you want to reach a certain goal, C(s2), you can infer the latent variables through an optimization procedure or you can sample the latent variables in order to give you different continuations of your world model
- training this would be done with non-constrative training of JEPA

Conclusion

When answering the question of "Could this Architecture be the Basis of a Model of Animal Intelligence," LeCun states, "The presence of a cost module that drives the behavior of the agent by searching for optimal actions suggests that autonomous intelligent agents of the type proposed here will inevitably possess the equivalent of emotions. In an analogous way to animal and humans, machine emotions will be the product of an intrinsic cost, or the anticipation of outcomes from a trainable critic."

Moreover, when asnwering the question of "Could this be a path towards machine common sense," LeCun states, "I speculate that common sense may emerge from learning world models that capture the self-consistency and mutual dependencies of observations in the world, allowing an agent to fill in missing information and detect violations of its world model."

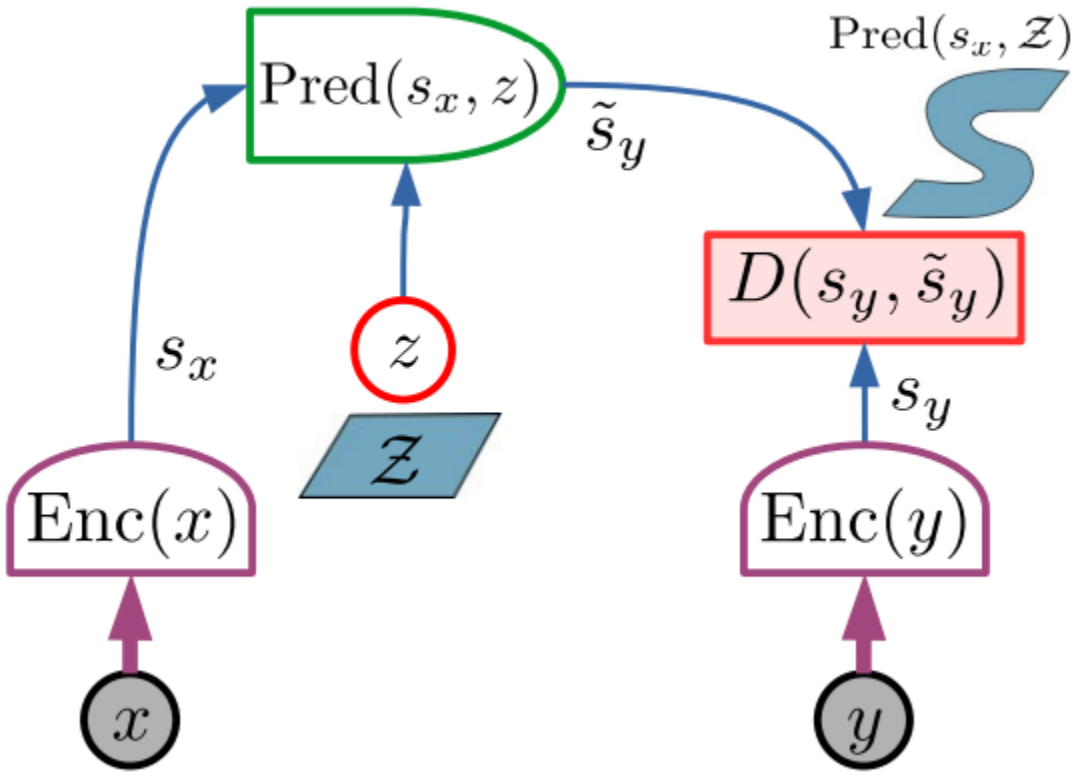
Furthermore, LeCun ends the paper by stating, "A remain question is whether the type of reasoning proposed here can encompass all forms of reasoning that humans and animals are capable of."

Question 1

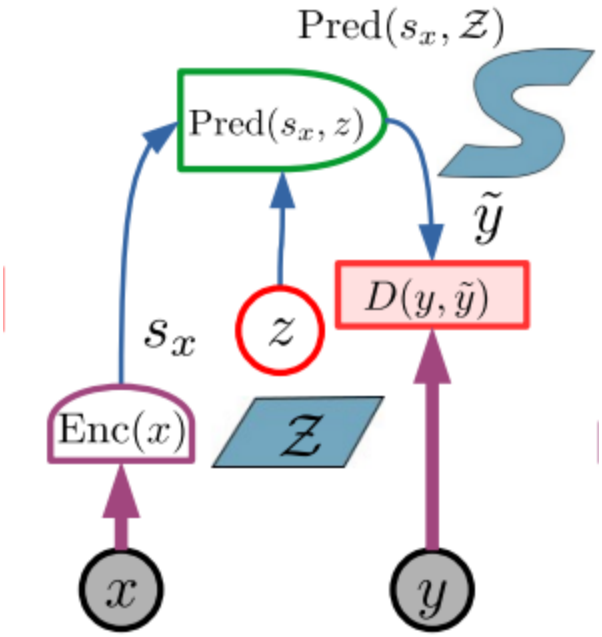
What is the difference between the JEPA Achitecture and the Generative latent-variable Architecture model?

Hint:

JEPA Architecture:



Generative latent-variable Architecture model:



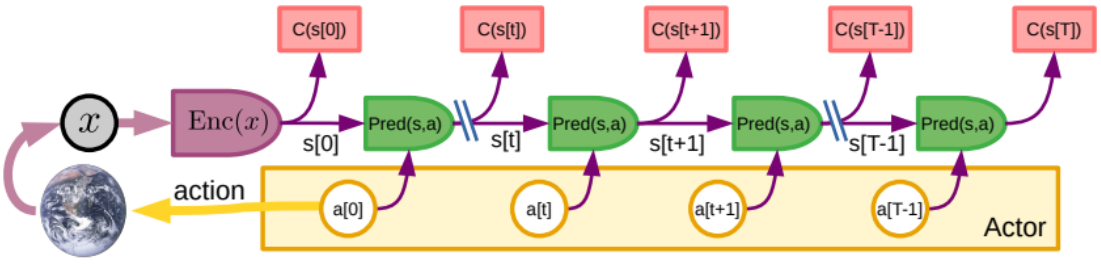
b) Generative latent-variable Architecture
CAN COLLAPSE

Question 2

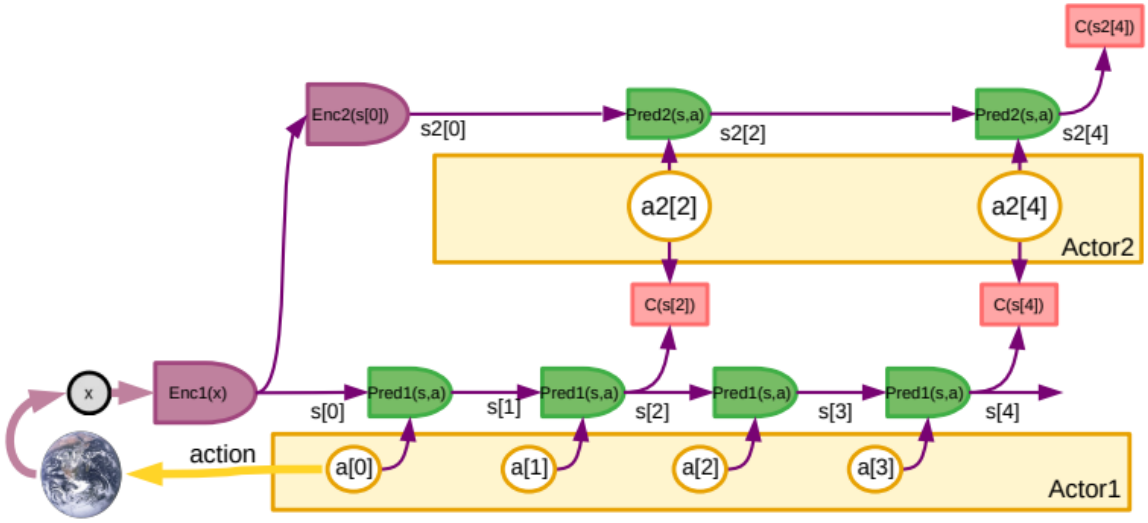
Explain how Hierarchical JEPA for Mode-2 hierarchical planning modifies/changes Mode-2 perception-action. What are the similarities? What are the differences?

Hint:

Mode-2 perception-action



Hierarchical JEPA for Mode-2 hierarchical planning



Architecture Overview (pseudocode)

As stated in the paper, LeCun states, "This document is not a technical nor scholarly paper in the traditional sense, but 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. Many ideas described in this paper (almost all of them) have been formulated by many authors in various contexts in various form. The present piece does not claim priority for any of them but presents a proposal for how to assemble them into a consistent whole. In particular, the piece pinpoints the challenges ahead. It also lists a number of avenues that are likely or unlikely to succeed."

Therefore, since this paper is not technical, there is no code involved. So, I have no pseudocode to show. However, at the end of the paper, LeCun provides **A list of contrastive methods and loss functions to train energy-based models** which I will provide below:

	Method	Energy	$\hat{y}$ Generation	Loss
1	Max Likelihood	discrete y	exhaustive	$F_w(x,y) + \log \sum_{y' \in \mathcal{Y}} \exp(-F_w(x,y'))$
2	Max Likelihood	tractable	exhaustive	$F_w(x,y) + \log \int_{y' \in \mathcal{Y}} \exp(-F_w(x,y'))$
3	Max likelihood	any	MC or MCMC	$F_w(x,y) - F_w(x,\hat{y})$
4	Contr. Divergence	any	trunc'd MCMC	$F_w(x,y) - F_w(x,\hat{y})$
5	Pairwise Hinge	any	most offending	$[F_w(x,y) - F_w(x,\hat{y}) + m(y,\hat{y})]^+$
6	Min-Hinge	positive	most offending	$F_w(x,y) + [m(y,\hat{y}) - F_w(x,\hat{y})]^+$
6	Square-Hinge	divergence	most offending	$F_w(x,y)^2 + \left([m(y,\hat{y}) - F_w(x,\hat{y})]^+\right)^2$
7	Square-Exp	any	most offending	$F_w(x,y)^2 + \exp(-\beta F_w(x,\hat{y}))$
8	Logistic	any	most offending	$\log(1 + \exp(F_w(x,y) - F_w(x,\hat{y}))$
9	GAN	any	$\hat{y} = g_u(z)$	$H(F_w(x,y), F_w(x,\hat{y}), m(y,\hat{y}))$
10	Denoising AE	$D(y, g_w(y))$	$\hat{y} = N(y)$	$D(y, g_w(\hat{y})$

Table 1: *A of list of contrastive methods and loss functions to train energy-based models. They are all use loss functions with two terms, one that pushes down on the energy of a training sample, and one the pulls up the energies of one or several contrastive samples.*
They differ by the strategy they employ to generate contrastive samples, and by the precise form of the loss function.
Exact or approximate Maximum Likelihood methods (rows 1-4) are used whenever the model needs to produce probability estimates. When the second term is intractable, its gradient may be approximated through Monte-Carlo methods, which can be seen as particular ways to produce $\hat{y}$. Many contrastive self-supervised methods for joint embedding architectures (Siamese nets) use Row 1 (InfoNCE).
A number of contrastive methods (Rows 5-8) are based on finding a $\hat{y}$ that is "highly offending", meaning different from the desired y, yet given a low energy by the model. Pairs of energies for y and $\hat{y}$ are fed to a loss function that pushes the former to low values and the latter to higher values. This can be done with a variety of losses including hinge loss.
GANs (row 9) are contrastive methods in which the contrastive samples are produced by a generator network whose input is a random vector. The generator is trained to produce samples to which the model currently attributes a low energy, but should attribute a high energy.
Denoising Auto-Encoders (row 10) apply a corruption process to training samples to produce contrastive samples $\hat{y} = N(y)$. The energy function is the reconstruction error $F_w(y) = D(y, g_w(y))$ where $D()$ is a symmetric divergence measure and $g_w(y)$ a parameterized function. By training $g_w()$ to map $\hat{y}$ to y, the energy for $\hat{y}$ is trained to be equal to $D(\hat{y}, y)$, while the energy of y is trained to be zero.

Critical Analysis

First of all, I want to make it clear that I did enjoy this paper and like LeCun's proposal. However, there were a few things that I noticed throughout the paper that could be improved. I will discuss them below:

Unorganized

First and foremost, throughout the paper, I noticed numerous spelling errors and gramatical mistakes. For example, on page 62, LeCun states, "A **of** list **of** contrastive

methods." In case you don't see it immediately, he uses the word "of" twice; one of which is out of place.

Moreover, the paper isn't very long (which doesn't mean that it is poorly written); However, it isn't very sophisticated either (ex: there is no code and no math involved). Knowing that LeCun is the Chief AI specialist at Meta, this really suprised me. To give him credit though, he states at the beginning of the paper that, "The text is written with as little jargon as possible, and using as little mathematical prior knowledge as possible, so as to appeal to readers with a wide variety of backgrounds including neuroscience, cognitive science, and philosophy, in addition to machine learning, robotics, and other fields of engineering"

Furthermore, compared to "The Soar Cognitive Architecture" by John E. Laird (a book about developing the fixed computational building blocks necessary for general intelligent agents), this book has much fewer components and is not very organized. I want to note that I did not read "The Soar Cognitive Architecture," but I did briefly skim it, and even from that, I could still see a huge difference in organization and complexity.

Lastly, there are multiple versions of the book as well which makes it confusing. For example, the version that I read was "Version 0.9.2, 2022-06-27," although when researching, I found multiple other versions that were completely different.

Missing Key Points

Again, I did NOT read "The Soar Cognitive Architecture," but I did do some research on it and noticed afterwards that LeCun's model (specifically, the system architecture for autonomous intelligence model) was missing key components such as an agent model, cognitive control, and long-term memory.

Moreover, there was not nearly enough emphasis on intrinsic motivation and minimizing risk which are obviously essential in any AI model (specifically, minimizing risk). Intrinsic motivation wasn't mentioned at all and minimizing risk was only mentioned once in the entire paper.

No code or Real World Examples

As I mentioned before, LeCun didn't intend for this paper to be super complex or mathematical. I'm fine with him providing no code in the paper. However, I would've liked him to have included at least one example of an actual real world experiment where his model has been/could be used. In my opinion, this is the biggest and most

important thing missing in this paper; real world examples always help the reader to better understand what you (the author) is trying to convey in his text.

Final Thoughts

Other than that, I thought the paper was well written and did truely enjoy learning about JEPA and LeCun's approach on how he thinks we should reach machine intelligence.

Resource Links

- Yann LeCun discussing "A Path Towards Autonomous AI" at NYU https://www.youtube.com/watch?v=DokLw1tILlw
- The link to Yann LeCun's github: https://github.com/ylecun
- Yann LeCun's personal website: http://yann.lecun.com/
- Lenet-5 (one of the earliest pre-trained models proposed by Yann LeCun and others in the year 1998) Huggingface page: https://huggingface.co/mindspore-ai/LeNet
- the link to Lesswrong blog about Yann LeCun and the paper: https://www.lesswrong.com/posts/BGiehNuRttGeH47W7/yann-lecun-a-path-towards-autonomous-machine-intelligence
- an article from shaped.ai about Yann LeCun and his paper: https://www.shaped.ai/blog/yann-lecun-a-path-towards-autonomous-machine-intelligence
- a YouTube video by David Shapiro discussing the paper: https://www.youtube.com/watch?v=RNwgfppbPQ4

Video Recordings

- Overview: https://www.youtube.com/watch?v=Se8IjGBodBU
- Detailed Explanation: https://www.youtube.com/watch?v=AC87In3cnjM&t=13s
- Code Demonstration: **LeCun provides no code in this paper**

Answers to Questions

Answer to Question 1

y is put into an encoder to give us a latent representation, sy, which is then put into the energy/compatibility function for the JEPA Achitecture

Answer to Question 2

For Hierarchical JEPA for Mode-2 hierarchical planning, there is a higher level action sequence with a higher level actor (Actor 2) to give us high level actions, a2... there is also a second encoder with higher level predictors.

References

[1] LeCun, Yann. "A Path Towards Autonomous Machine Intelligence Version 0.9.2, 2022-06-27." Openreview.net, New York University, 27 June 2022, https://openreview.net/pdf?id=BZ5a1r-kVsf.

[2] Preetham, Freedom. "A Quick Take on Yann Lecun's Path towards Autonomous Machine Intelligence." Medium, Autonomous Agents, 5 Oct. 2022, https://medium.com/autonomous-agents/a-quick-take-on-yann-lecuns-path-towards-autonomous-machine-intelligence-973d44fa35ce?source=rss----add2a54a5aee---4.

[3] Spencer, Michael. "A Path towards Autonomous Machine Intelligence." A Path Towards Autonomous Machine Intelligence, AI Supremacy, 28 June 2022, https://aisupremacy.substack.com/p/a-path-towards-autonomous-machine.

[4] Spencer, Michael. "A Path towards Autonomous Machine Intelligence." A Path Towards Autonomous Machine Intelligence, AI Supremacy, 28 June 2022, https://aisupremacy.substack.com/p/a-path-towards-autonomous-machine.

[5] "State of the Industry: Yann Lecun: A Path towards Autonomous Machine Intelligence." YouTube, YouTube, 14 Sept. 2022, https://www.youtube.com/watch?v=RNwgfppbPQ4.

[6] "Yann Lecun, a Path towards Autonomous Machine Intelligence [Link]." LessWrong, https://www.lesswrong.com/posts/BGiehNuRttGeH47W7/yann-lecun-a-path-towards-autonomous-machine-intelligence.

[7] “Yann Lecun: A Path towards Autonomous Machine Intelligence.” Shaped, https://www.shaped.ai/blog/yann-lecun-a-path-towards-autonomous-machine-intelligence.

[8] “Yann Lecun: From Machine Learning to Autonomous Intelligence.” YouTube, YouTube, 28 Sept. 2022, https://www.youtube.com/watch?v=VRzvpV9DZ8Y.

[9] “Yann Lecun: ‘A Path towards Autonomous AI’, Baidu 2022-02-22.” YouTube, YouTube, 25 Feb. 2022, https://www.youtube.com/watch?v=DokLw1tILlw.