Introduction to Latent Variable Energy-Based Models:
A Path Towards Autonomous Machine Intelligence

Anna Dawid[1,2] and Yann LeCun[3,4*]

1 ICFO - Institut de Ciències Fotòniques, The Barcelona Institute of Science and Technology,
Av. Carl Friedrich Gauss 3, 08860 Castelldefels (Barcelona), Spain
2 Faculty of Physics, University of Warsaw, Pasteura 5, 02-093 Warsaw, Poland
3 Courant Institute of Mathematical Sciences, New York University
4 Meta - Fundamental AI Research
* yann@cs.nyu.edu

June 6, 2023

Abstract

Current automated systems have crucial limitations that need to be addressed before artificial intelligence can reach human-like levels and bring new technological revolutions. Among others, our societies still lack Level 5 self-driving cars, domestic robots, and virtual assistants that learn reliable world models, reason, and plan complex action sequences. In these notes, we summarize the main ideas behind the architecture of autonomous intelligence of the future proposed by Yann LeCun. In particular, we introduce energy-based and latent variable models and combine their advantages in the building block of LeCun's proposal, that is, in the hierarchical joint embedding predictive architecture (H-JEPA).

Contents

1   Introduction

Over the past decade, machine learning (ML) methods have exploded in popularity and played a crucial role in many substantial technological advancements. We have witnessed ML models that achieve expert performance in strategic games like Go, Chess, and Shogi [1,2], that can solve challenging physical simulation problems like protein folding with groundbreaking accuracy [3], and that translate text between over 200 languages [4]. These new technologies all have one technique in common: deep learning (DL) or the use of deep neural networks (NNs) that may have hundreds or thousands of layers. Perhaps surprisingly, training deep NNs on extensive datasets has seemingly become a straightforward recipe to achieve human-like performance on generic computational tasks, even demonstrating what may seem to be intelligent and creative problem-solving!

In exchange for their groundbreaking performance, creating DL models can require training on massive datasets, which is an extreme computational expense. Human learning, in contrast, is often highly efficient. With only a few examples, we can quickly find an intuitive way to complete a task and easily generalize our learning to other tasks. For instance, babies quickly obtain an intuitive understanding of physics, allowing them to predict their motions' outcomes and change them accordingly. Conversely, robots trained through reinforcement learning (RL) struggle with this task, requiring years of simulated interactions with the world to achieve intuitive motion [1,2].

In these lecture notes, we explore the concept of *autonomous intelligence,* which can learn efficiently and automatically to predict the state of the world, much like human learners. Eventually, we hope to achieve a *fully autonomous AI,* which can perform well on generic tasks by transferring its knowledge and automatically adapting to new situations without trying out many solutions first. The content of this paper follows a series of lectures given by Yann LeCun in July 2022 as a part of the Summer School on Statistical Physics and Machine Learning [5] in École de Physique des Houches, organized by Florent Krzakala and Lenka Zdeborová. We aim here to explain the limitations of the current ML approaches and introduce central concepts needed for understanding a possibly autonomous artificial intelligence (AI) of the future proposed by Yann LeCun in his paper "A Path Towards Autonomous Machine Intelligence" in 2022 [6] as well as the main idea behind the design.

2   Towards autonomous machine intelligence

Over the last decade, ML has made tremendous progress in various domains, such as image recognition, machine translation, or more recently text-to-image generation. This progress has been enabled by the use of graphics processing unit (GPU) hardware designed for fast matrix multiplications which vastly accelerate the computational speed of convolutional neural network (CNN) as well as fully connected and transformer-based NN architectures. In turn, large-scale models can be trained on massive amounts of data, leading to impressive performance [7–9]. Those technological developments have allowed various real-world applications.

2.1   Applications of machine learning today

Most large-scale applications of ML today rely on supervised learning (SL) or on deep RL, both of which require extreme computational expense to learn patterns from training data. More recently, these models have been increasingly making use of *self-supervised pretraining*, a type of learning that allows the model to re-use and adapt its learned weights to new prediction problems, much like how humans learn efficiently.

Transportation   Deep learning (DL) revolutionizes the advanced driver-assistance systems (ADAS's) that include automatic emergency braking, adaptive cruise control, lane keeping assist, traffic jam assist, and forward collision warning. In Europe, an additional boost to ADAS development is provided by a regulation from July 2022 that makes some of ADAS's mandatory in new vehicles [10]. ADAS's rely largely on DL and CNNs in particular [11]. The market is dominated by the Israeli company Mobileye (which became a subsidiary of Intel in 2017) with its flagship device called EyeQ. Regarding self-driving cars, driverless taxi service limited to some city areas is already offered for customers in San Francisco by Cruise, Phoenix by Waymo, and Beijing by Baidu.

Translation, transcription, accessibility   Models that can learn adaptable representations are especially useful for translation and transcription tasks, where the goal is to transform content from one representation to another. For example, the *No Language Left Behind* (NLLB) model [4] is able to translate text between 202 languages by learning to convert input text to a language-agnostic representation that can later be decoded in any other language, avoiding the need for supervised training examples of translation between every possible pair of languages. Similarly, models such as DALL-E [12] or Make-A-Scene [13] use learned internal representations to convert input text or sketch data to photorealistic images. In speech recognition, models like Wav2Vec [14] and XLSR [15] allow to generate representations of audio that can be used for audio-to-text transcription. Translation models have a huge variety of applications, such as video subtitle generation, controllable image generation, or even expanding the accessible content for people who may have impaired vision, or hearing, or who may not be able to read.

Online safety and security   Next to recommendation systems, possibly the largest application of ML today is filtering harmful/hateful content and dangerous misinformation. While companies may not have the legitimacy to decide what content is acceptable or questionable, the lack of effective legal regulations addressing this question is not helping in creating a safe digital environment. A concrete example is the moderation of hate speech on Facebook which is dominated by DL techniques and assisted by human moderators. In 2018, only around 40% of hateful content was taken down before it was seen by users, while between January-March 2022 the reported rate was 95.6% [16]. This increase is solely thanks to the progress in

natural language processing (NLP) techniques and especially in self-supervised learning (SSL) and transformers. Interestingly, a major technique is also a nearest-neighbor search against the blacklisted content.

Computer vision   ML algorithms can detect objects in images and videos, segment them, handle up to tens of thousand categories, and identify human poses. Major breakthroughs are the semantic and instance segmentation [17] which consist in labeling each pixel with the category that the consisting object belongs to and detecting that two overlapping shapes belong to two different objects, respectively. An example of such a detection and segmentation system is Detectron 2 [18].

Medical analysis   Very promising are DL applications in medical analysis [19]. An example is acceleration (by a factor of four) of data acquisition from magnetic resonance imaging (MRI) (so-called fastMRI) [20]. The acceleration comes from the physics-informed convolutional-deconvolutional NN (sometimes called a U-Net) which recovers an image (which is then analyzed by physicians) taking advantage of known redundancies in the momentum space where the measurements take place and allowing for its undersampling without any degradation of the final image. The fastMRI is now being integrated into new MRI machines by Siemens, Phillips, and General Electrics. The next step that is under development is to skip entirely the generation of two-dimensional image slices that are needed for the human analysis and to go directly from the momentum space to the diagnosis [21].

Biomedical sciences   DL models are also vastly accelerating scientific progress in biomedical sciences. In neuroscience, NNs are used to model and interpret cognitive processes such as neural activity [22] and sensory input representation [23]. In genomics, DL models have been used to identify networks of gene regulation and in understanding genetic diseases [24]. In biology, the recent models such as AlphaFold [3] and RoseTTA Fold [25] can accurately predict the three-dimensional structure of proteins from their amino acid sequence, an extremely important and challenging problem whose solution opens new pathways to drug design.

Physics   DL is used to analyze particle physics experiments and to accelerate the solutions of partial differential equations (PDEs), allowing for more complex simulations of fluids, aerodynamics, atmosphere, oceans [26]. In astrophysics, it enables universe-wide simulations [27], galaxy classification, and discovers exoplanets. In quantum sciences [28], NNs boost phase classification, design of experiments as well as serve as a representation of quantum states.

Chemistry and material science   Chemists use DL to find new compounds and boost expensive quantum-chemical methods [28, 29]. ML designs also new materials and catalysts that are promising, e.g., for large-scale energy storage [30].

2.2   Limitations of the current machine learning

Despite all their successes, ML models still suffer from important limitations. Self-driving cars are a good example of this claim. Indeed, current models of self-driving cars work well in restricted scenarios: fully-mapped environments, the presence of many sensors, good weather conditions, and wide roads. Their abilities are far from Level 5 autonomous cars [31] which are reliable in any environment as well as require no human attention, therefore have no driving wheels or acceleration/braking pedals. More importantly, these limitations stem from the fact that current models required thousands of footage to learn, while in contrast humans can learn to drive in twenty hours or so.

Next to self-driving cars, there are virtual assistants that could help us in our daily lives. Such assistants would need to manage the information deluge, understand real-time speech, translate on-the-fly, or overlay useful information on our augmented reality glasses. For this, they also need to understand human intents. Domestic robots additionally require a high-level understanding of the environment. All in all, the future autonomous systems that would take human civilization to the next level and also help in understanding the human brain, will need a near-human level of intelligence.

> Current ML systems still miss crucial requirements for the future AI systems, such as a basic understanding of the world and humans that may be called *"common sense"* which we understand here as the ability to use models of the world to fill in information about the world that is unavailable from perception or memory (e.g., to predict the future) [6, 32].

So far, however, our ML systems largely rely on SL, which requires large numbers of labeled samples, and RL, which requires an enormous number of trials, which makes them impractical for the real-world applications in the current form. RL works great with games because they can be played over and over in parallel. In the real world, however, each action takes time and has some cost (in particular, can kill you if you drive a car). Moreover, the modern ML systems are very specialized (trained for one task) and brittle (make stupid mistakes). Most of them have a constant number of computational steps between input and output. They cannot reason or plan. While the modern large language models sometimes provide an illusion of reasoning, they have no understanding of reality. One of the reasons is that everything they know comes solely from written sources which are unlikely to describe all possible experiments humans and animals can do in the physical world. Moreover, they miss the visual input, the primary knowledge source for young humans and animals.

In contrast, humans and animals rely mostly on active observations of their environment and build a world model from that. In particular, babies learn almost entirely by observation and between the second and sixth month, they already understand concepts of object permanence, solidity, and rigidity, while around of tenth month, they start understanding gravity, inertia, conservation of momentum, and reach rational, goal-directed actions around the twelfth month of life [6]. Their learning process is the most similar to SSL, with only a little bit of SL (e.g., through interaction with parents) or RL (by trying out various solutions in practice). In reality, humans *imagine* most of the outcomes instead of trying them all. Humans and animals have a good dynamical model of their own bodies but also of physics and social interactions. They can also predict the consequences of their actions, perform chains of reasoning with an unlimited number of steps, and plan complex tasks by decomposing them into sequences of subtasks. They percept the world in various ways, i.e., through visual, interactions, text, and other senses. As a result, when humans and animals reach a basic understanding of how the world works, they can learn new tasks very quickly.

Therefore, the path to fully autonomous human-like intelligence has three main challenges:

1. **Learning representations and predictive models of the world** that would allow AI systems to predict the future and in particular consequences of their actions which would give them the ability to plan and control the outcome. Moreover, this representation should be task-agnostic. The most likely approach is *self-supervised learning (SSL)* as SL and RL require too many samples or trials.

2. **Learning to reason** in a way that is compatible with DL.[1] Reasoning (in an analogy

---

[1]There are proposals to include logical reasoning in AI systems [33], but logical reasoning is not differentiable and as a result is challenging to combine with DL.
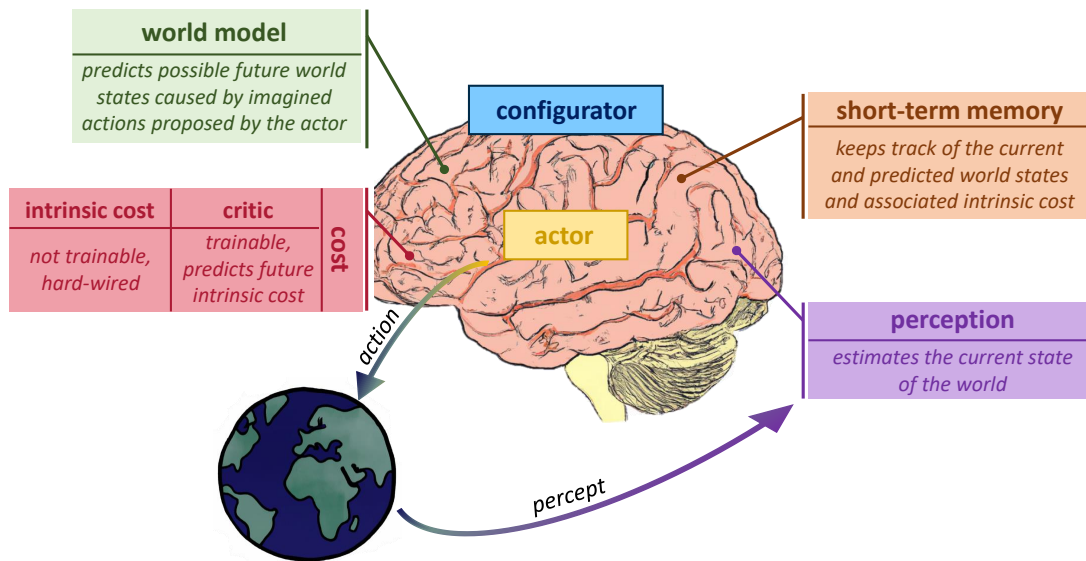
Figure 1: The modular structure of an autonomous AI proposed by LeCun in Ref. [6]. Drawings generated by DALL-E 2 [12].

of Daniel Kahneman's "System 2"[2]) takes into account an intent in contrary to feed-forward subconscious computation (analogously to "System 1"). The possible approach is designing reasoning and planning as *energy minimization*.

3. **Learning to plan complex action sequences** that require *hierarchical* representations of action plans.

2.3   New paradigm to autonomous intelligence

In Ref. [6], LeCun proposed that such an autonomous AI may have a modular structure presented in Fig. 1. We refer there for the detailed description. Here, we start by giving an overview of the main ideas behind the proposed architecture. Then we focus on its building blocks which are self-supervised learning (SSL), energy-based models (EBMs), and latent variables, and we introduce them in detail in the following sections.

   The proposed AI architecture consists of multiple interconnected modules. The *perception* module estimates the current state of the world which then may be used by the *actor* proposing optimal action sequences guided by the *world model* which predicts (or "imagines") future possible world states given actions of the actor. We call those connections a "perception-planning-action cycle". While imagining possible consequences of the actor's actions, the world model uses the *cost* module for inference. Interestingly, this cost module is proposed to be divided into two sub-modules: the immediately calculable *intrinsic* cost which is hard-wired into the architecture (i.e., not trainable) and models basic needs like pain, pleasure, hunger, and the *critic*, a trainable module that predicts future values of the intrinsic cost and is influenced by the perception module. The changes of the critic module model the external influence that is provided by culture, norms, and other people. Moreover, there is a *short-term memory* module which stores state-cost episodes that can be used by the world model when predicting the

---

[2]Daniel Kahneman is a psychologist awarded with the Nobel Prize in Economic Sciences in 2002 for his contributions to understanding the psychology of judgment and decision-making (in particular, he challenged the assumption of human rationality prevailing in modern economic theory). In his book "Thinking, Fast and Slow", he describes a dichotomy between two modes of thought: "System 1" which is fast, instinctive, and emotional, and "System 2" which is slower, more deliberate, and logical [34].
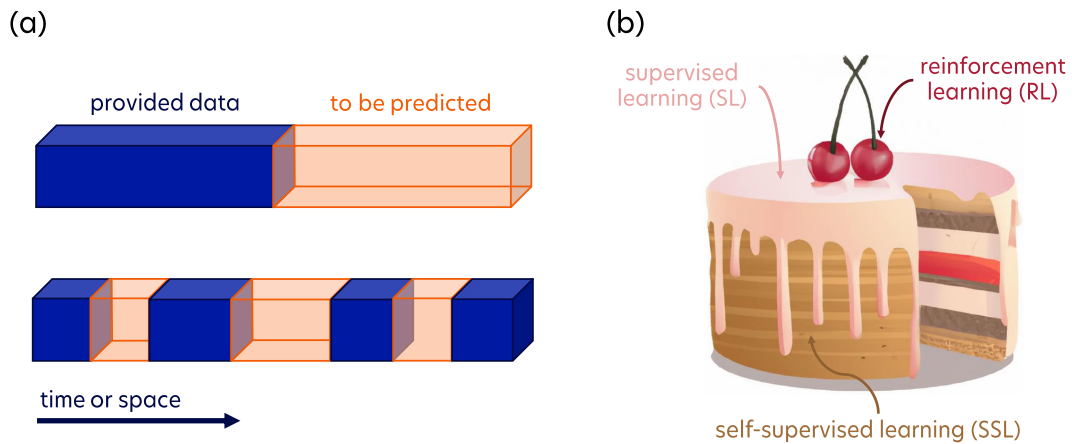
Figure 2: **Self-supervised learning (SSL).** (a) In SSL, the system is trained to predict hidden parts of the input (in orange) from visible parts of the input (in blue). (b) SSL will play a central role in the future AI systems. SSL is the cake (provides millions of information bits per sample), SL is the icing (10-10,000 bits per sample), RL is the cherry on top (a few bits of information for some samples). A cake image was generated by DALL-E 2 [12].

future world states. The global task of this architecture is to take action that minimizes the cost or, even better, minimizes the expected cost in the future. Finally, there is a *configurator* module which enables switching between tasks by configuring all other modules. An advantageous property of this architecture in the lens of modern tools is that all modules are fully differentiable.

The *perception-planning-action cycle* in the proposed model is akin to model-predictive control (MPC) [35] known in optimal control. The key difference is that the world model predicting the future here is learned. It is different also from RL because here the cost function is known and all modules are fully differentiable, so no action needs to be taken in reality to predict the future cost. The world model, in analogy to humans and animals, may be primarily trained with SSL which is proposed as the central element for the next AI revolution.

2.4   Self-supervised learning and representing uncertainty

The main aim of the SSL is reconstructing the input or predicting missing parts of the input as presented in Fig. 2(a) [36]. The input can be an image, video, or text. Within the training, the model learns hierarchical representations of the data, and thanks to that, nowadays, the SSL pretraining often precedes a SL or RL phase. It is also used for learning predictive (forward) models for MPC or learning policies for control or model-based RL. SSL learns and also predicts the largest amount of information bits per sample as compared to SL and RL and is, therefore, proposed as the main approach for future autonomous AI as schematically shown in Fig. 2(b).

SSL works very well for text, but for images, when models are trained to make a single prediction, training makes them predict the average of all possible predictions. As a result, SSL produces blurry predictions. In general, due to the stochasticity of reality and limited perception, there is no chance of predicting every detail of the world. However, making decisions usually does not require predicting all possible details of the world, just the relevant ones that are task-dependent.

> Therefore, a great challenge of SSL for the future AI systems is (1) to represent uncertainty in the prediction and (2) to allow for multiple predictions being equally probable for a single input, i.e., *multimodal* predictions.

The approaches that have been developed so far have limitations. In particular, probabilistic models are intractable in continuous domains, meaning we do not know how to properly represent normalized continuous distributions in high dimensions. For example, we can predict the next word in the sentence because we can produce a probability distribution over all words in a dictionary and pick the most probable one. It used to be computationally challenging, but that is how modern large language models write text. However, if we want to predict the next frame in the video or the next one hundred words in the sentence, building a distribution over this enormous set of all possibilities stops being feasible. In such high-dimensional and multi-modal real-world setups, we may need to *give up the idea of representing probability distributions over predictions*.

> The proposed solution is, therefore, to replace probabilistic models with energy-based models (EBMs) and combine them with latent variables for handling uncertainty and task-dependent information redundancy as with the hierarchical structure for complex planning within the architecture called hierarchical joint embedding predictive architecture (H-JEPA).

2.5   Structure of the manuscript

The following sections aim to describe every element of this fundamental building block of proposed future AI systems. In section 3, we introduce energy-based models (EBMs) and latent variable EBMs and compare them with the probabilistic models. In section 4, we describe how to train EBMs with contrastive and regularized methods, and we explain the disadvantages of the former. Section 5 presents examples of EBMs of historical and practical relevance, i.e., Hopfield networks, Boltzmann machines, and masking and denoising autoencoders (AEs). Finally, section 6 introduces the building block of the proposed AI of the future, that is joint embedding predictive architectures (JEPAs), describes their training and extension to hierarchical joint embedding predictive architectures (H-JEPAs) which should be able to provide multi-level and multi-timescale predictions needed for future AI systems.

3   Introduction to energy-based models

Probabilistic models require normalization and may therefore become intractable in the limit of high-dimensional data. However, in decision-making tasks such as driving a car, it is merely necessary that the system chooses the correct answer. The probabilities of other answers are irrelevant as long as they are smaller. Therefore, instead of predicting a single most probable event, we can let the model represent the dependency between variables (e.g., decisions y and conditions x) through the energy function, as schematically presented in Fig. 3(a). Such an energy-guided model only needs to assign the lowest energy to the correct answer and larger energies to the incorrect ones.
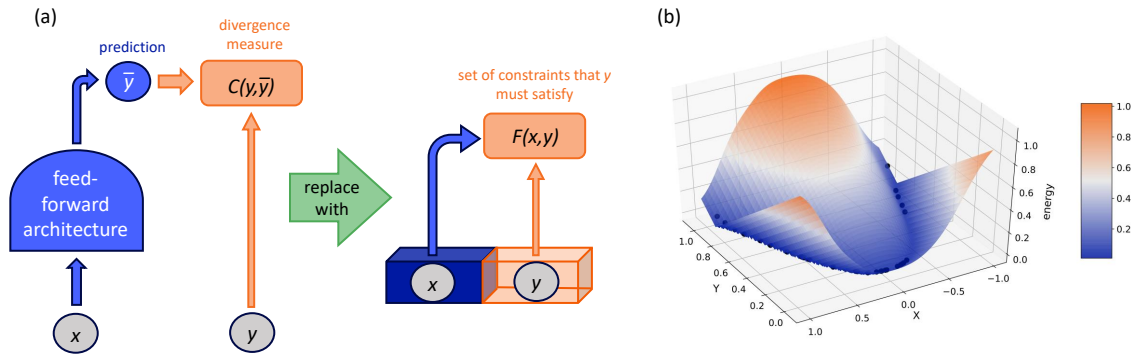
Figure 3: **Towards energy-based models (EBMs).** (a) To achieve multimodal predictions in high dimensions, we can replace probabilistic models with EBMs. Then, instead of minimizing the divergence measure between the prediction and the target, we look for y's that satisfy a set of constraints posed by x, expressed as the energy function, $F(x, y)$. A trained EBM should assign low energies to y's that are a good continuation of x (in case of video or text) or that are compatible or similar (in case of images of an object taken from different angles). (b) Exemplary energy function capturing the dependence of x and y (which is $y = x^2$) from the training set, represented as blue points. The applied architecture is presented in Fig. 5(a). Note that the energy function is not unique given only the training data set!

> Energy-based models (EBMs) capture dependencies between variables x and y by associating a scalar energy $F(x, y)$ to each configuration of the variables [37, 38]. Low energies should be assigned to compatible pairs of x and y (e.g., slow driving when snow is present) and high energies to incompatible pairs. The inference consists then in finding values of y that minimize $F_w(x, y)$: $\check{y} = \operatorname{argmin}_y F_w(x, y)$.

An exemplary energy function representing the quadratic dependency between x and y is presented in Fig. 3(b). The inference then involves finding the minimum energy value for a given x. An advantage of EBMs is that they can represent multimodal dependencies in which multiple values of y are compatible with a given x. In theory, they can also describe dependencies between data in various forms (text, visuals, etc.). Moreover, a useful property of EBMs is that if the y domain is continuous, the energy function F is smooth and differentiable so that we can use gradient-based inference algorithms.

An important thing to note is that the energy function discussed here is used only for inference, not learning. Training of an EBM (described in more detail in section 4) involves a loss function and consists in finding an energy function in which observed configurations of the variables are given lower energies than unobserved ones.

> Therefore, the energy function is not the objective function to minimize within the learning! The energy is used only for inference.

3.1   Energy-based models vs. probabilistic models

In the probabilistic setting, the training consists in finding such model parameters w that maximize the likelihood (or minimize the negative likelihood) of observing outputs given inputs:[3]

$$-\log P_w(y_1, \ldots, y_n \mid \mathbf{x}_1, \ldots, \mathbf{x}_n) = -\log \prod_i P_w(y_i \mid \mathbf{x}_i) = -\sum_i \log P_w(y_i \mid \mathbf{x}_i), \qquad (1)$$

where the first equality comes from the assumption that data points are independent, and the second transformation is done because summation is easier computationally than multiplication. With probabilistic models, the training is confined to loss functions generated from the negative log-likelihood like the cross entropy. When we use other losses, we lose even this weak connection to the probabilistic setting. EBMs, which do not have an immediate probabilistic interpretation, give much more flexibility in the choice of the training setup and, in particular, the objective function used for learning. They also give more flexibility in the choice of the similarity/scoring function.

While it may be surprising that we give up on a probabilistic setting, note that making decisions can be viewed as choosing the option with the highest score instead of the most probable one. A well-studied example is playing chess, where deciding on the next move by looking at all possibilities and their consequences is intractable. Instead, one can explore a part of the tree of possibilities [2], e.g., with Monte-Carlo tree search or dynamical programming to find the shortest path in some graph giving, in the end, the minimal energy (or minimal energy for you and maximal for your opponent). Therefore, there is no need to use a probabilistic framework, as you just need scores for each situation (word/move, etc.), which in particular do not need to be normalized. In fact, it is hurtful to normalize those scores as we enter the label bias problem [39], which happens when the transition probabilities of leaving a given state are normalized for only that state instead of globally.

While the advantages of EBMs over probabilistic models are obvious in high dimensions, we lose something when giving up on the probabilistic setting. Firstly, it is challenging to understand uncertainty in an energy-based setting. Secondly, we lose outputs that can be compared between models. In particular, energies are uncalibrated (i.e., measured in arbitrary units), so combining two separately trained EBMs is not straightforward. To avoid this drawback, we need to design an architecture that does not require transferring outputs between its components.

However, if needed, we can make a connection between EBMs and probabilistic models by considering energies as the unnormalized negative log probabilities. Therefore, we can turn the EBM into the probabilistic model as probabilistic models are a special (normalized) case of EBMs. The most common method for turning a collection of arbitrary energies into a collection of numbers between 0 and 1 whose sum (or integral) is one is through the *Gibbs-Boltzmann distribution*:

$$P(y \mid x) = \frac{\exp[-\beta F(x, y)]}{\int \mathrm{d}y' \exp[-\beta F(x, y')]}, \qquad (2)$$

where the denominator is called the *partition function*, which encodes how the probabilities are partitioned among the different possibilities based on their individual energies $F(x, y)$ (therefore, plays a role of a normalization constant), and β plays a role of the inverse temperature. Computing a partition function is very challenging, so probabilistic models in continuous domains in high dimensions are intractable.

---

[3]What is also worrying, if we take a probabilistic model and train it on data by maximizing likelihood, the only model maximizing it is the one that assigns an infinite probability to every data point, and zero exactly everywhere else. It is a terrible model, and we cannot use it for inference! Of course, the Bayesian setting that assumes some prior and maximizes posterior avoids this problem by regularization.

Prediction problem	Examples of latent variables
Face recognition	the gender of the person, the orientation of their face, the color of their eyes/hair
Object recognition or scene parsing	the pose parameters of the object (location, orientation, scale), lighting conditions, segmentation of the image into component objects, assignment of labels to objects
Speech tagging	sentence segmentation into syntactic units, the parse tree
Speech recognition	sentence segmentation into phonemes or phones
Handwriting recognition	the segmentation of the line into characters

Table 1: Examples of prediction problems and potentially relevant latent variables

3.2   Latent variable energy-based models

We can expand the possibilities of EBMs by using an additional energy function that depends on a set of hidden variables z whose correct value is never (or rarely) given to us. Those hidden variables are often called *latent variables*, and they aim at capturing the information in y that is not readily available in x. Examples of such latent variables may be gender, pose, or hair color in the face detection task. In the case of self-driving cars, latent variables can parametrize the possible behaviors of other drivers. Thus, they give us a way of handling real-world uncertainty. More examples of latent variables are in Tab. 1. Finally, latent variables are useful in so-called *structured prediction problems*.

> In a structured prediction problem, we assume that the data has some unknown structure that the learner must resolve to make an accurate prediction.

For example, in object detection, input data is structured as an image composed of multiple objects in a scene, and the goal is to recover both the structure of the image in the form of a *segmentation map* and multiple predictions for the contents of each substructure in the image. Another example of a structured prediction task is speech recognition (predicting the text transcript from an audio recording, e.g., to provide video captioning), where the missing structure is segmenting the audio into individual words. Handwriting recognition is also a structured prediction problem as to predict the text transcript from handwritten words, the algorithm needs to detect the segmentation of the handwritten word into individual letters. In the examples of structured prediction tasks listed above, the segmentation maps can naturally be viewed as latent variables for the prediction problem.

When working with latent variable EBMs, the inference process for a given set of variables x and y involves additionally minimizing or marginalizing[4] with respect to these unseen variables z, and the energy function becomes either $F_w(x, y) = \text{argmin}_z E_w(x, y, z)$ or $F_w(x, y) = -\frac{1}{\beta} \log \left[\int \mathrm{d}z' \exp \left[-\beta E_w(x, y, z') \right] \right]$ as presented schematically in Fig. 4(a). The inference then is simply $\check{y} = \text{argmin}_y F_w(x, y)$.

An educational example of inference with latent variable EBMs is an EBM parametrized by $w = \{r_1, r_2\}$ that learned an ellipse in the data manifold as presented in Fig. 4(b) and whose latent variable was used to encode the angle: $E_{r_1, r_2}(y, z) = (y_1 - r_1 \sin z)^2 + (y_2 - r_2 \cos z)^2$. The resulting energy function computes a squared distance between a point y to the learned manifold by minimizing the $F_{r_1, r_2}(y) = \text{argmin}_z E_{r_1, r_2}(y, z)$ with respect to z which amounts to finding the distance between the data point y and its closest point of the ellipse at angle

---

[4]Marginalization leads to a formula known from statistical physics, where F is the free energy of the ensemble of systems with energies E, see Eq. (A5) in Appendix A. However, it becomes expensive in high dimensions and is usually replaced with a cheaper minimization instead.
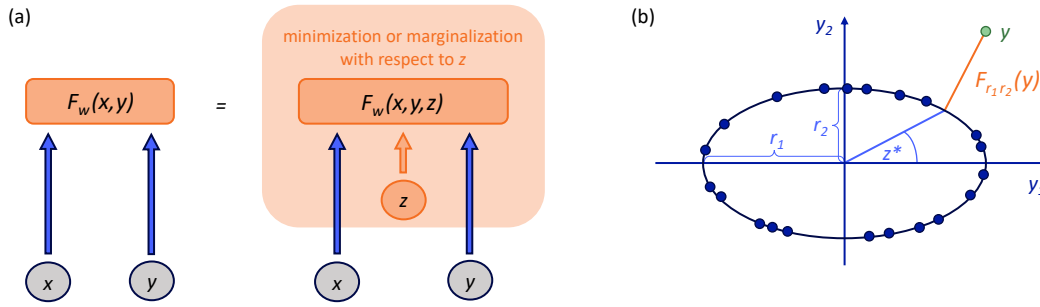
Figure 4: **Latent variable energy-based models (EBMs).** (a) Inference in latent variable EBMs additionally includes the minimization (or marginalization) with respect to the latent variable. (b) An example of the latent variable EBM in the problem of finding the distance of a green point y from an ellipse learned from the training points depicted as blue dots. The latent variable here encoded the angle at which the closest to y point on the ellipse lies.

$\check{z}$. We can use the same example by visualizing the difference between minimization and marginalization. If the inference is made by $F_{r_1,r_2}(y) = -\frac{1}{\beta} \log\left[\int \mathrm{d}z' \exp\left[-\beta E_{r_1,r_2}(y,z')\right]\right]$, it is analogous to computing contributions to the energy of y from all points of the ellipse (so for z from 0 to 2π) with contributions from the closest points being the largest.

> An important feature of latent variable models is that while they are composed of deterministic functions, they are non-deterministic as the latent variable they depend on is determined outside the model.

The latent variable can be drawn out of a particular probability distribution or chosen from a set of possibilities, and such sampling produces either a distribution over predictions (making the model probabilistic) or just a set of predictions (making the model only non-deterministic), respectively. Therefore, latent variables enable multiple predictions instead of just one. They can play a role of a switch or modulator, taking into account various situational factors. Ideally, the latent variable represents independent explanatory factors of prediction variation. The tricky part is that the information capacity of the latent variable must be minimized. Otherwise, the training may put all the information needed for the prediction into them.

4 Training an energy-based model

So far, we have discussed how to use EBMs, in particular, latent variable EBMs, for inference. In the case of basic EBMs, the inference consists in finding values of y that minimize the energy function $F(x, y)$: $\check{y} = \mathrm{argmin}_y F(x, y)$. When we deal with latent variable EBMs, we minimize the energy function also with respect to the latent variable z: $\check{y} = \mathrm{argmin}_{y,z} E(x, y, z)$. In this section, we describe how to train EBMs.

> The main principle of training an EBM is to sculpt the energy landscape such that data points that are probable or compatible have low energy, while unlikely data points have high energy. There are two main approaches: *contrastive methods* and *regularized and architectural methods*.

The choice of the training technique depends on the choice of the EBM architecture. Let
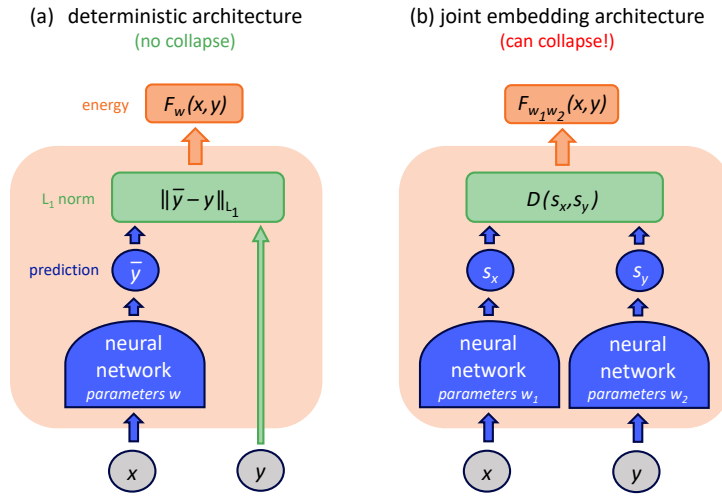
Figure 5: **Energy-based models (EBMs) can collapse.** (a) Standard deterministic architecture for prediction or regression, where energy function $F_w(x, y)$ is the distance between the NN prediction for x and the y itself, is immune to collapse. (b) An example of an EBM that can collapse.

us compare two EBM architectures in Fig. 5. In the first case, the energy function is simply the distance between the data point y and the output of an encoder (e.g., an NN) of the data point x. Such an architecture can be thought of as a regression model[5] and trained by simply minimizing the energy of training samples. However, for other architectures, such training may lead to the *collapse* of the energy function, i.e., given an x the energy landscape could become "flat", giving essentially the same energy to all values of y. Consider for example a joint-embedding architecture in Fig. 5(b). Such an architecture encodes the inputs x and y into $\text{Enc}_x(x)$, $\text{Enc}_y(y)$, respectively. The goal is to find such Enc_x, Enc_y so their representations of x and y are close. If we train our model only to minimize the distance between the outputs of encoders, then the two encoders are likely to ignore inputs entirely and just produce identical constant outputs.

There are more examples of EBMs that can collapse. The latent variable generative EBM architecture in Fig. 4(b), if trained to simply minimize the distance between y and latent variable-dependent prediction on x, can ignore x entirely and find such a latent variable that causes the prediction to be identical with y. Another example is autoencoder (AE) that encodes y, the decodes y into $\tilde{y}$, and if trained to minimize the distance between y and $\tilde{y}$, it can collapse because it can learn the identity function and have zero energy for all the inputs. Thus, the AE will reconstruct perfectly inputs on which it has not been trained, which we may want to avoid.

All in all, every model that can be multimodal, i.e., have multiple predictions for a single input, is susceptible to collapse.

> To prevent the energy collapse, i.e., flat energy landscape, we need to constrain our energy surface so that data points have low energies but also that points outside the regions of high data density have higher energies.

---

[5]Such an architecture is immune to collapse but has no way of expanding predictions to be multimodal or non-deterministic.
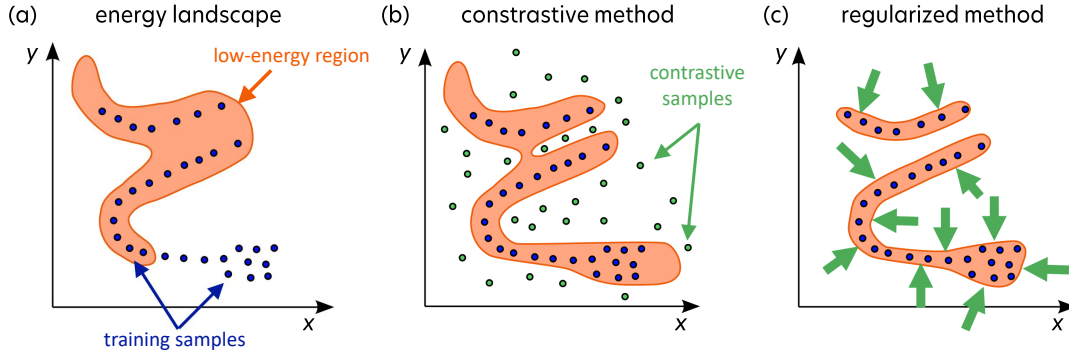
Figure 6: **Training of an energy-based model (EBM).** (a) Proper training minimizes training samples' energy while preventing energy collapse. (b) Contrastive methods minimize the energy of training samples while increasing the energy of samples outside the training set. (c) Regularized methods limit the volume of space that the low-energy region can take. Adapted from Ref. [6].

4.1 Contrastive methods

To prevent energy collapse, we can make use of contrastive methods.

> *Contrastive methods* aim to push down the energy of training samples and to pull up the energy of suitably-generated contrastive samples that by definition are out of the training set. They work for any architecture but scale badly with data dimension.

An example of the contrastive loss is the following pairwise hinge loss:

$$L_{\text{hinge}}(x, y, \hat{y}, w) = [F_w(x, y) - F_w(x, \hat{y}) + m(y, \hat{y})]^+ . \tag{3}$$

x, y are training data points whose energy we aim to lower and are represented as blue dots in panel (b) of Fig. 6. $\hat{y}$ is a contrastive point, represented as green dots in panel (b) of Fig. 6, whose energy we need to increase. When minimizing L_{hinge}, we ensure that the energy of training samples is smaller than the energy of a training sample and contrastive sample, at least by the margin m that depends on the distance between y and $\hat{y}$. Suitable contrastive loss functions need to ensure a non-zero margin to avoid energy collapse. The contrastive loss functions can be calculated pairwise for specific data sets as the hinge loss in Eq. (3), but lately, a growing interest is given to group losses, which depend on a group of points instead of pairs of points, like InfoNCE [40, 41].

4.1.1 How to generate contrastive points

The heart of contrastive methods lies in how to generate contrastive points. For example, if the space of possible contrastive points is discrete or tractable, the generation of $\hat{y}$ can be exhaustive. Otherwise, they can be generated by sampling with Monte Carlo methods or their approximations [42]. Moreover, we can look for the "most offending" predictions of the model, i.e., points to which the model incorrectly assigns low energy, and choose them as contrastive points. Another approach was developed along with Siamese NNs [43], which is an example of joint-embedding architecture. There, the training set consists of both compatible and incompatible pairs, and the latter plays the role of contrastive samples. The compatible pairs can be, e.g., different views on the same object, and the incompatible data would be views on different objects. Incompatible pairs can also be generated by distorting or corrupting the original data

set, e.g., with denoising AEs. Interestingly, the contrastive points can be also generated with generative adversarial networks (GANs) [44, 45]. There, the generator is trained to produce contrastive samples with minimal energies, while the discriminator is trained to assign high energies to these generated samples. For a more exhaustive list of possible approaches and contrastive losses, we refer to Refs. [37] (section 2.2) and [6] (appendix 8.3.3).

> The need for generating contrastive samples is the source of bad (even exponential!) scaling of contrastive methods with the data dimension. As real-world data can be extremely complex and high-dimensional, this unfavorable scaling makes contrastive methods unlikely to lead to autonomous intelligence of the future.

4.1.2 Maximum likelihood as a special case of a contrastive method

Finally, the maximum likelihood can be interpreted as a special case of a contrastive method. As we have explained in section 3.1, we can transform EBM into a probabilistic model (and vice versa) using the Gibbs-Boltzmann distribution from Eq. (2). We can use the same transformation (and knowledge that $\log[\exp(x)] = x$) to show that the connection between the maximum likelihood (or negative log-likelihood) contrastive EBMs:

$$L_{\mathrm{NLL}}(x, y, w) = -\frac{1}{\beta} \log P_w(y \mid x) = F_w(x, y) + \frac{1}{\beta} \log\left[\int \mathrm{d}y' \exp\left[-\beta F_w\left(x, y'\right)\right]\right], \quad (4)$$

where we choose to divide the loss by $\frac{1}{\beta}$. When minimizing this loss, the energy of training points $F_w(x, y)$ decreases. At the same time, the second term increases the energy of all the points in the available space (including the training points). To minimize the loss in (4), we need its gradient. The gradient of the first term is available through the backpropagation, while the gradient of the second term (assuming we know the gradient of the first term) is the following:

$$\frac{\partial\left[\frac{1}{\beta} \log\left[\int \mathrm{d}y' \exp\left[-\beta F_w\left(x, y'\right)\right]\right]\right]}{\partial w} = -\int \mathrm{d}y' P_w\left(y' \mid x\right) \frac{\partial F_w\left(x, y'\right)}{\partial w}. \quad (5)$$

Therefore, we can approximate it by computing a sum over Monte Carlo samples of $P_w\left(y' \mid x\right)$ without explicitly computing this distribution.

> It means that the minimization of L_{NLL} consists in minimizing the energy of training samples while pulling up the energy of Monte Carlo-generated samples. Negative log-likelihood is then a contrastive method!

The main disadvantage of maximum likelihood as a contrastive EBM is that it wants to make the difference between the energy on the data manifold and the energy just outside of it infinitely large, making the data manifold an infinitely deep and infinitely narrow canyon. Maximum likelihood needs to be then *regularized*, e.g., with Bayesian prior or by limiting values of model parameters. Both the need for regularization and the bad scaling with data dimension of contrastive methods suggests that probabilistic models also may fail on the path to autonomous intelligence.

4.2 Architectural and regularized methods

> Another option to prevent collapse is to use *regularized methods*. These methods aim to minimize the volume of space that the low-energy points can take. When the energy of input data is lowered, it is required to increase the energy of some parts of the space. Because regularized methods are less likely than contrastive methods to fall victim to the curse of dimensionality, they seem more promising to train EBM of the future.

The main challenge is choosing how to limit the volume of the low-energy space. One approach is to *build architectures where the low-energy space volume is bounded by construction*. Examples of such models are principal component analysis (PCA) (where reconstruction capacity is limited to the highest principal components), k-means (which has a discrete data representation), Gaussian mixture models, and normalizing flows (both with hard-coded normalization). Another approach is to *add a regularization term that minimizes a certain measure of the volume of the low-energy space*. In the case of the latent variable EBMs (Fig. 4(b)), such a regularization can be done by limiting the information capacity of the latent variable, e.g., by making it discrete, sparse (as in the case of sparse AEs), stochastic (variational AEs), or low-dimensional. Finally, *score matching* [46] is a regularization technique that minimizes the gradient and maximizes the curvature of the energy landscape around data points.

4.2.1 Regularized autoencoders

In this section, we consider the task of *autoencoding*, where the input and label are the same: $x = y$. The goal is to find a good encoding $\text{Enc}(y)$ of the input which is then decoded and outputted by $\text{Dec}(\text{Enc}(y))$. In this case, the regularized EBM training objective takes the form of

$$L_{\text{reg}}(y, \boldsymbol{w}) = D(y, \text{Dec}(\text{Enc}(y))) + R(\text{Enc}(y)),$$

where D represents some distance between the input and the output and R is a regularization of the encoder. Perhaps surprisingly, this representation encompasses many constrained fitting problems, for example:

1. Principal component analysis (PCA): $L_{\text{reg}}(y, \boldsymbol{w}) = \|y - \boldsymbol{w}^T \boldsymbol{w}\|^2$ where an implicit regularization comes from the low rank of $\boldsymbol{w}^T \boldsymbol{w}$.

2. An autoencoder (AE) with a bottleneck: $L_{\text{reg}}(y, \boldsymbol{w}) = \|y - \text{Dec}(\text{Enc}(y))\|^2$, implicitly regularized by the small size of the bottleneck.

3. k-means: $L_{\text{reg}}(y, \boldsymbol{w}) = \min_{\boldsymbol{w} \in \mathcal{Z}} \|y - \text{Dec}(\boldsymbol{w})\|^2$ where $\boldsymbol{w} \in \mathcal{Z}$ is a discrete set of k means values which is responsible for the regularization.

4. Gaussian mixtures whose regularization has an analogous source like k-means.

5. Sparse coding: $L_{\text{reg}}(y, \boldsymbol{w}) = \min_z \|y - \boldsymbol{w}z\|^2 + \lambda\|z\|_1$ with an explicit regularization of strength λ.

Interestingly, some regularized AEs learn features that are interpretable and mirror existing hand-crafted approaches to data compression. For example, convolutional sparse AEs learn features corresponding to known image filters used for feature extraction and compressed representation [47].

4.2.2   Regularization through the variational marginalization of a latent variable

Another way to control the information content in the case of latent variable models is to replace the latent variable z with a noisy random variable, $z \sim q(z \mid y)$. Then, instead of energy minimization, one minimizes the expected value of the energy:

$$\langle F(y) \rangle_z = \int \mathrm{d}z q(z \mid y) E(y, z) \, dz.$$

The information content of $q(z \mid y)$ can be measured using the Kullback-Leibler (KL) divergence $\mathrm{KL}(q(z \mid y) \parallel p(z))$, where $p(z)$ is the prior $\mathbb{E}_y[q(z \mid y)]$. The KL divergence measures the level of dependence of $q(z \mid y)$ on y. So, we can recast the regularized AE as jointly *minimizing the expected energy* while also *maximizing the entropy*.

5   Examples of energy-based models

In this section, we describe a few examples of EBMs of historical and practical relevance. We start with the Hopfield networks and Boltzmann machines and focus on denoising and masking AEs in the end.

5.1   Hopfield networks

Hopfield networks are fully-connected recurrent networks popularized by John Hopfield in 1982 [48]. They have symmetric connections, $w_{ij} = w_{ji}$, and binary activations, $y_i \in \{-1, 1\}$). The scheme of such a network is presented in Fig. 7(a). It has the following resulting energy function:

$$F_w(y) = -\sum_{ij} y_i w_{ij} y_j \tag{6}$$

They were first proposed as a spin glass or the Ising type of an NN that is able to store patterns (memories) [49]. In this model, the inference, which is finding the set of y_i that minimizes the energy given trained parameters $\breve{w}$, is done by repeatedly updating the neuron states depending on the values of all other neurons: $y_i \longleftarrow \mathrm{sign}(\sum_j \breve{w}_{ij} y_j)$. Within this procedure, $y = \{y_i\}$ converge to the local minimum of the energy function.

The energy function of Hopfield networks is also the loss function that is used for training: $L_{\mathrm{Hopfield}}(y^{\mathrm{train}}, w) = F_w(y^{\mathrm{train}})$. The training is done by updating the connections w to lower the energy of data samples:

$$w_{ij} \leftarrow w_{ij} - \frac{\partial L(y^{\mathrm{train}}, w)}{\partial w_{ij}} = w_{ij} + y_i^{\mathrm{train}} y_j^{\mathrm{train}}. \tag{7}$$

This training rule is called Hebbian learning by neuroscientists. It states: whenever two neurons are active simultaneously, strengthen the connection between them. The problem of such training is the lack of contrastive terms which contributes to the emergence of the so-called spurious minima (or memories), i.e., accidental energy minima not shaped by the training data which may be found during inference. This limitation renders the Hopfield networks less usable in practice.

5.2   Boltzmann machines

A year later, in 1983, the extension of the Hopfield network, called the Boltzmann machine [50] was proposed by Geoffrey Hinton and Terrence Sejnowski, introducing neurons named

(a) Hopfield network   (b) Boltzmann machine   (c) restricted Boltzmann machine
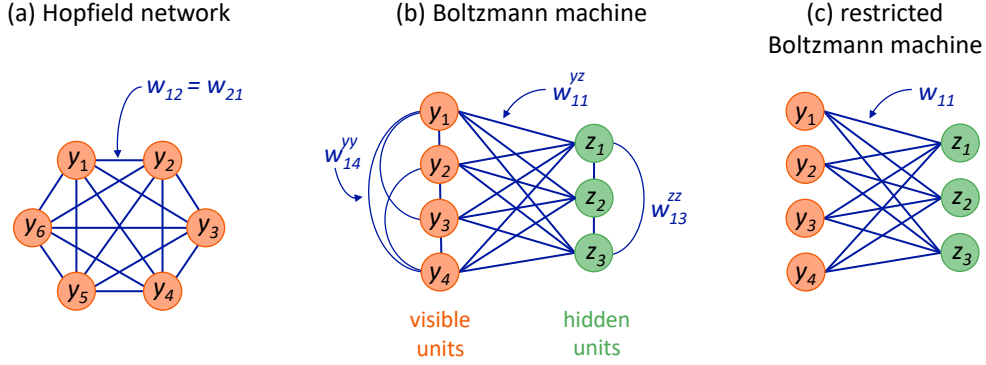

Figure 7: **Examples of energy-based models (EBMs).** (a) Hopfield network. (b) Boltzmann machine, which is the Hopfield network with hidden units. (c) Restricted Boltzmann machine has only connections between visible and hidden units.

hidden units, as presented in Fig. 7(b). This proposal is important to the whole ML community as it was the first introduction of hidden units, i.e., neurons whose inputs and outputs were unobserved. Those hidden units can also be understood as latent variables of the model. The energy function of the Boltzmann machine and its free energy (after marginalizing over hidden units) are the following:

$$E(\boldsymbol{y}, \boldsymbol{z}) = -\sum_{ij} y_i w_{ij}^{yy} y_j - \sum_{ij} z_i w_{ij}^{zz} z_j - \sum_{ij} y_i w_{ij}^{yz} z_j, \tag{8}$$

$$F(\boldsymbol{y}) = -\log \sum_{\boldsymbol{z}} \exp[-E(\boldsymbol{y}, \boldsymbol{z})] \tag{9}$$

If we allow only connections between hidden and visible units (so we set $\boldsymbol{w}^{yy} = \boldsymbol{w}^{zz} = 0$), the Boltzmann machine becomes the restricted Boltzmann machine, as presented in Fig. 7(c). Contrary to Hopfield networks, the training here involves minimizing a loss function with a contrastive term:

$$L_{\text{Boltzmann}}(y, \boldsymbol{w}) = F_{\boldsymbol{w}}(y) + \log \sum_{y'} \exp\left[F_{\boldsymbol{w}}(y')\right], \tag{10}$$

where summation goes over contrastive samples generated with Markov chain Monte Carlo sampling. The challenge here is that we need to sample both values of visible and hidden units to approximate the gradient of $L_{\text{Boltzmann}}$:

$$\frac{\partial L_{\text{Boltzmann}}(y, \boldsymbol{w})}{\partial w_{ij}} = \frac{\partial F_{\boldsymbol{w}}(y)}{\partial \boldsymbol{w}} - \sum_{y'} P(y') \frac{\partial F_{\boldsymbol{w}}(y')}{\partial \boldsymbol{w}} \quad \text{with} \quad P(y) = \frac{\exp[-F_{\boldsymbol{w}}(y)]}{\sum_{y'} \exp[-F_{\boldsymbol{w}}(y')]} \tag{11}$$

$$\frac{\partial L_{\text{Boltzmann}}(y, \boldsymbol{w})}{\partial w_{ij}} = \sum_{z'} P(z' \mid y) \frac{\partial E_{\boldsymbol{w}}(y, z')}{\partial \boldsymbol{w}} - \sum_{y', z'} P(y', z') \frac{\partial E_{\boldsymbol{w}}(y', z)}{\partial \boldsymbol{w}} \quad \text{where} \tag{12}$$

$$P(z \mid y) = \frac{\exp[-E_{\boldsymbol{w}}(y, z)]}{\sum_{z'} \exp[-E_{\boldsymbol{w}}(y, z')]} \quad \text{and} \quad P(y, z) = \frac{\exp[-E_{\boldsymbol{w}}(y, z)]}{\sum_{y', z'} \exp[-E_{\boldsymbol{w}}(y', z')]}.$$

The first term requires sampling on z' from the conditional distribution $P(z' \mid y)$ where y is fixed, while the second term on both z' and y' from the joint distribution $P(y', z')$. As an example, let us analyze sampling on z' from $P(z' \mid y)$. We can compute the difference between energies $E_{\boldsymbol{w}}(y, z)$ for $z_i = 1$ and $z_i = 0$ from Eq. (8): $\Delta E_{\boldsymbol{w}}(y, z) = \sum_j w_{ij}^{yz} y_j$. With this energy difference, sampling can be done with the use of the Fermi-Dirac distribution:
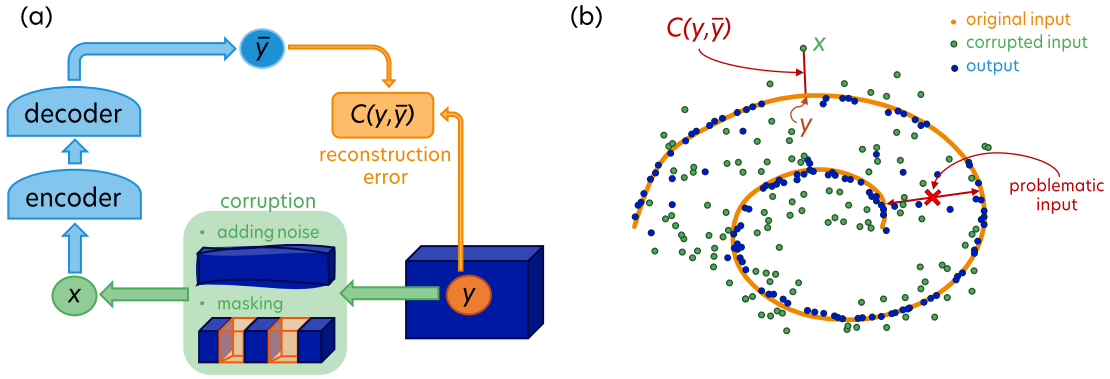
Figure 8: **Denoising and masked autoencoders (AEs).** (a) A scheme of a denoising (and masked) AE. (b) An exemplary denoising problem.

$P(z_s \mid y, z_{i \neq s}) = \frac{1}{1 + \exp\left(-\sum_j w_{ij}^{yz} y_j\right)}$. Once we have those samples, e.g., z' from $P(z' \mid y)$, and y'' and z'' from $P(y', z')$, each learning step gets simplified to $w_{ij} \leftarrow w_{ij} + (y_i z'_j - y''_i z''_j)$, where $y_i z'_j$ is called the positive phase, and $y''_i z''_j$ - the negative phase of the training.

The Boltzmann machines were fashionable for some time,[6] but the large cost of sampling makes them less useful for real-world applications.

5.3 Denoising autoencoders

Denoising autoencoder (AE) [51] is a type of a contrastive EBM. It is an AE that is trained to restore a clean version of a corrupted input. The scheme of its architecture is presented in Fig. 8(a). For example, the AE may be trained to shift data points to their original positions after adding a random noise like in Fig. 8(b). The original data points come from the orange spiral and are corrupted by adding some noise to their position. The green corrupted points are then input as x to the denoising AE in Fig. 8(a) along with their clean version y. The reconstruction error is the distance between the corrupted point and the original point, and when minimized, the denoising AE outputs blue data points that are back on the spiral. Note that in the same problem, there are also problematic points for which denoising AEs may not work. For example, an AE is unable to reconstruct a data point located between two branches of the spiral, equidistant to them.[7] Example of such a problematic input is visualized in Fig. 8(b). This pitfall is due to the folded structure of the data, which, however, rarely occurs in real-world data.

A particular type of a denoising AE is a masked AEs that is also trained on corrupted data. However, in these models, instead of using noisy data, part of the input is masked, and the goal is to predict the masked part. This method is especially successful in NLP [52–55] with applications in text understanding and translation systems, where part of the sentence is hidden, typically the last word. In particular, they are part of a huge open-source initiative called "No Language Left Behind" which aims at the automatic translation between any language [4]. The current model, NLLB-200, can translate between 202 languages (in any of the 40602 directions). Interestingly, the training set contained 18 billion pairs of sentences but only for 2440 language directions. Moreover, the model performance on individual languages improves as more languages are added to the training set.

However, masked AEs are less successful with images, so in continuous domains, in general, [56]. A promising approach is to combine masked AEs with transformers for images [57].

---

[6]They still are in some particular applications like representing quantum states [28, 29].

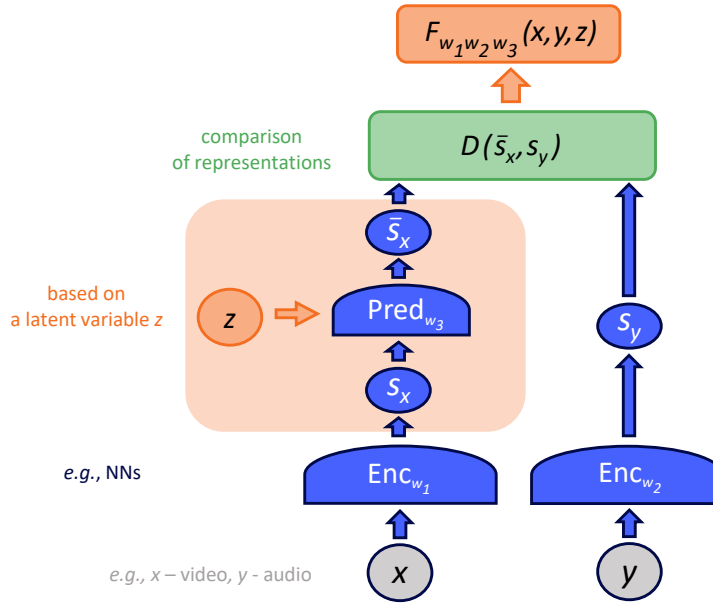[7]This problem can be solved, however, by introducing latent variables!

Figure 9: **A joint embedding predictive architecture (JEPA)**. It consists of two parallel encoders, which can be different in general, learning representations of x and y, i.e., s_x and s_y. Those encoders can learn to filter irrelevant information about x and y. The aim of JEPA is to predict s_y out of s_x with help of a latent variable z that can represent relevant information about y that may be missing from x. The energy is the distance between those representations.

The reason for the limited success of masked AEs in continuous domains like images or videos is probably the multimodality of predictions. There are many ways of filling in the missing parts of the image or video, but the model is allowed to output only one. The proposed solution to tackle multimodality is latent variable models with joint embeddings, and we discuss them in the next section.

6   Building block of proposed autonomous systems of the future

We have already shown how EBMs overcome limitations of probabilistic models and that in the case of high-dimensional data, we probably should train them with regularized instead of contrastive methods. We also discussed latent variable models and explained their usefulness in structured prediction problems or in incorporating uncertainty. Now, we combine those advantages in an architecture called a joint embedding predictive architecture (JEPA).

6.1   Joint embedding predictive architecture

A joint embedding predictive architecture (JEPA), presented in Fig. 9, is an EBM that combines embedding modules with latent variables. As an EBM, a JEPA learns a dependency between input data, x and y, but *compares them at the level of learned internal representations, s_x and s_y*, where $s_i = \text{Enc}(i)$. The two encoders, producing representations s_x and s_y, can be different, in particular, have different architectures, and do not share parameters. Thanks to that, the input data can have various formats (e.g., video and audio). Moreover, JEPAs naturally handle multimodality. Firstly, the encoders of x and y can have invariance properties and, e.g., map various y's onto the same s_y. Analogous invariance can be captured by the latent variable z that is used to infer the information necessary to predict s_y that is not present in s_x.
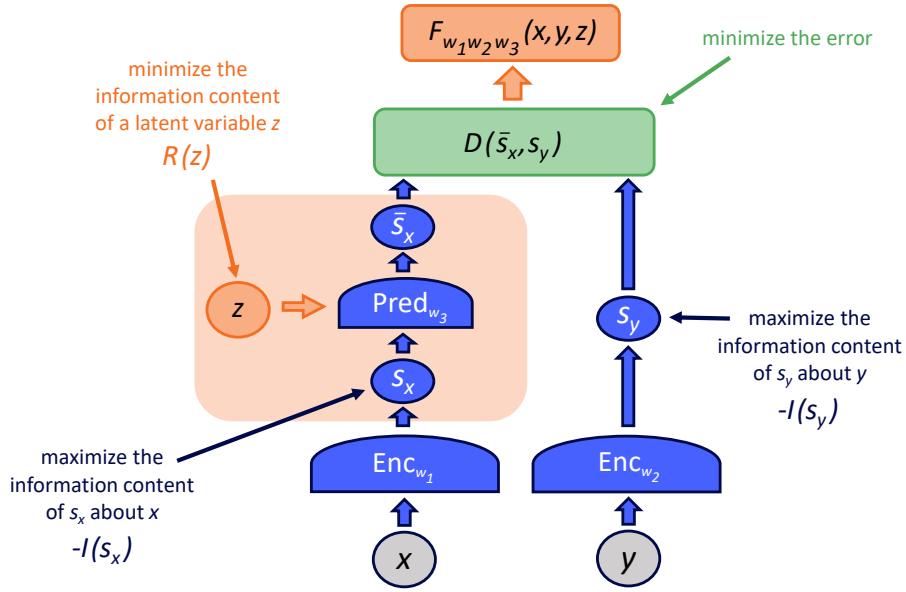
Figure 10: **Regularized loss function for training a joint embedding predictive architecture (JEPA).** Apart from the prediction error that needs to be minimized, there are also additional terms that maximize the information content of representations about their respective input data points and minimize the information content of a latent variable. Adapted from Ref. [6].

Ultimately, the goal when training the JEPA is to make the representations s_x and s_y predictable from each other. As we discussed in section 4, EBMs can be trained with both contrastive and regularized methods, but contrastive methods tend to become very inefficient in high dimensions. Therefore, the JEPA can be trained with a loss function that apart from the prediction error includes also regularization terms presented schematically in Fig. 10. In particular, to prevent the informational energy collapse, we need to make sure that s_x and s_y carry as much information as possible about x and y. Otherwise, the training could lead encoders to be, e.g., constant. Finally, we need to minimize or limit the information content of the latent variable to prevent the model from, e.g., relying solely on information there.

We have already discussed regularization methods, in particular for the latent variable models in section 4.2, which give an idea of how to tackle the regularizing term $R(z)$ limiting the information content of a latent variable. Here, let us give an example of how to maximize the information content of learned representations about respective input data. The method is called variance-invariance-covariance regularization (VICReg) [58]. Imagine that the encoder, out of an input data x, produces its representation s_x composed of vectors v_i. VICReg maximizes the information content of the representation *by minimizing covariance of its components*, $\text{Cov}(v_i, v_j)|_{i \neq j}$, which is calculated over a batch of samples. Covariance is related to correlation, and its zero value indicates that there is no linear dependence between variables. In other words, this term decorrelates the representation components, encouraging the learning of different input data aspects. Additionally, VICReg keeps the variance of each component above a given threshold via a hinge loss, e.g., $[1 - \sqrt{\text{Var}(v_i)}]^+$ which forces the representation vectors of samples within a batch to be different. Finally, to discourage also non-linear dependencies between v_i, at the stage of pretraining, the architecture uses an expander, e.g., an NN, whose task is to expand the dimensionality of the embedding in a non-linear fashion that reduces the dependencies (not just the correlations) between the variables of the representation vector.

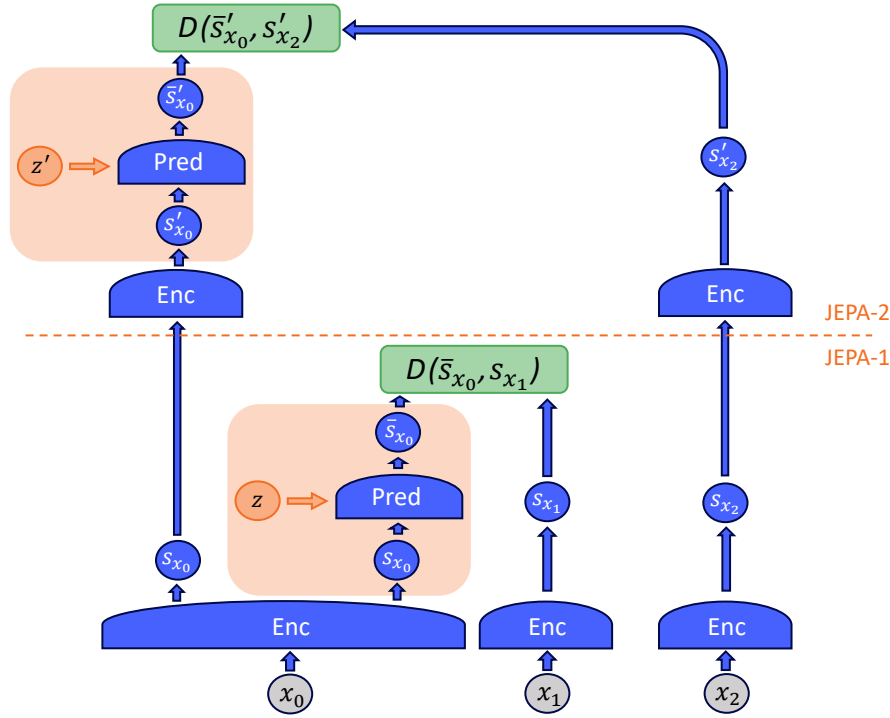Of course, there are various approaches to regularizing JEPAs and related architectures

Figure 11: Hierarchical joint embedding predictive architecture (H-JEPA) can be used for multiscale planning where a lower-level JEPA-1 is used for short-time predictions based on many details, and a higher-level JEPA-2 makes predictions for the longer term based on fewer details. Adapted from Ref. [6].

such as BYOL [59], SimSiam [60], or Barlow Twins [61]. Interestingly, related ideas were explored already in 1992 [62]!

6.2 Hierarchical joint embedding predictive architecture

When trained with regularized methods, a JEPA can train its encoders to eliminate irrelevant details of the inputs and produce representations that are more predictable from each other. However, which information is important may be scale-dependent. In particular, precise predictions of the state of the world, based on the limited percepts of the environment, are possible only in the short term and require a lot of details about the input. The same level of detail may not be needed or be even hurtful for long-term predictions, which may require different information on the environment.

An architecture that may be able of both short- and long-term predictions relying on representations on various levels of abstraction is hierarchical joint embedding predictive architecture (H-JEPA) and is presented in Fig. 11. It is composed of stacked JEPAs whose aim is to represent data on various levels of abstraction. When provided with a sequence of observations, $x_0, x_1, \ldots$, the first-level network, denoted as JEPA-1 performs short-term predictions using low-level representations. The second-level network JEPA-2 performs longer-term predictions using higher-level representations. H-JEPA can be constructed out of multiple JEPAs assisted by CNNs and pooling layers between levels to coarse-grain the representation and enable even longer-term predictions.

Such a hierarchical structure allows for multi-level representations of the world and, more importantly, for hierarchical planning, i.e., a decomposition of a complex task into more detailed subtasks regularly updated by new observations. This addresses the third and final challenge of the path to fully autonomous human-like intelligence.

7 Conclusion

Deep learning (DL) and, more broadly, artificial intelligence (AI) undoubtedly revolutionized the industry in the past years and started reshaping science as well. However, before AI gets a chance to take our civilization to the next level with Level 5 self-driving cars, virtual assistants, and domestic robots that we know from science fiction, it needs to be freed from its current limitations. For example, supervised learning (SL) and reinforcement learning (RL), which dominate modern real-world applications, are highly inefficient compared to human learning. They require an enormous number of either labeled samples or trials. More importantly, current automatic systems still miss crucial requirements for the future AI systems, such as a basic understanding of the world and humans that may be called "common sense" which we understand here as the ability to use models of the world to fill in information about the world that is unavailable from perception or memory (e.g., predict the future).

Here we summarized the main ideas of LeCun from Ref. [6] that address those limitations. In section 3, we explained that as real-world data (such as video or text) is usually high-dimensional, the energy-based models (EBMs) may be a more promising approach to the future human-like intelligence than probabilistic models that stop being tractable in continuous high-dimensional domains. In section 4, we introduced contrastive and regularized methods to train EBMs and explained that due to the large cost of generating contrastive samples in high dimensions, regularized methods seem more promising for training EBMs of the future. We gave examples of EBMs of historical and practical relevance in section 5.

Finally, section 6 focused on the fact that a human-like decision process is based on data of various formats and representations, with a structure that often needs to be decoded to make a prediction, also containing information that can be redundant depending on the task. Such multimodality can be addressed with a new architecture proposed by LeCun in Ref. [6], called joint embedding predictive architectures (JEPAs), on three levels. Firstly, JEPAs are trained to capture dependencies of two input objects and allow those objects to have a different format (e.g., video and audio). Secondly, JEPAs make predictions in the representation space, allowing encoders to remove irrelevant data features for the task at hand. Thirdly, latent variables of JEPAs can encode additional features not readily present in the input data, allowing for handling uncertainties in the perceived data.

The final challenge is to allow the autonomous AI of the future to handle predictions of the state of the world on various time scales and levels of abstraction. Such multi-level predictions may be achieved with a hierarchical joint embedding predictive architecture (H-JEPA). Its architecture is simply a series of stacked JEPAs. Lower-level JEPAs encode data and feed its representations to higher-level JEPAs, creating multi-level representations. This architecture, trained with regularized methods, may be a starting point for designing predictive world models able of hierarchical planning under uncertainty that would constitute a breakthrough in developing an autonomous AI of the future.

Acknowledgements

List of acronyms

ADAS advanced driver-assistance system

AE autoencoder

AI artificial intelligence

CNN convolutional neural network

DL deep learning

EBM energy-based model

GAN generative adversarial network

GPU graphics processing unit

H-JEPA hierarchical joint embedding predictive architecture

JEPA joint embedding predictive architecture

KL Kullback-Leibler

ML machine learning

MPC model-predictive control

MRI magnetic resonance imaging

NLP natural language processing

NN neural network

PCA principal component analysis

PDE partial differential equation

RL reinforcement learning

SL supervised learning

SSL self-supervised learning

VICReg variance-invariance-covariance regularization

A   How to turn energies into probabilities

- Discrete and continuous Gibbs distribution (or softmax, should be called softargmax):

$$P_w(y) = \frac{\exp[-\beta F_w(y)]}{\sum_{y'} \exp[-\beta F_w(y')]} \quad \text{and} \quad P_w(y) = \frac{\exp[-\beta F_w(y)]}{\int \mathrm{d}y' \exp[-\beta F_w(y')]} \tag{A1}$$

- Joint distribution:

$$P_w(y,z) = \frac{\exp[-\beta E_w(y,z)]}{\int \mathrm{d}y' \int \mathrm{d}z' \exp[-\beta E_w(y',z')]} \tag{A2}$$

- Conditional distribution:

$$P_w(y,z \mid x) = \frac{\exp[-\beta E_w(x,y,z)]}{\int \mathrm{d}y' \int \mathrm{d}z' \exp[-\beta E_w(x,y',z')]} \tag{A3}$$

- Marginal distribution:

$$P_w(y \mid x) = \int \mathrm{d}z' P_w(y,z' \mid x) = \frac{\int \mathrm{d}z' \exp[-\beta E_w(x,y,z')]}{\int \mathrm{d}y' \int \mathrm{d}z' \exp[-\beta E_w(x,y',z')]} \tag{A4}$$

Note that the marginal distribution is equivalent to Eq. (A1) in which the energy function has merely been redefined from $E_w(x,y,z)$ to $F_w(x,y) = -\frac{1}{\beta} \log \int_{z \in \mathcal{Z}} \exp[-\beta E_w(x,y,z)]$, which is the free energy of the ensemble $E_w(x,y,z), z \in \mathcal{Z}\}$:

$$
\begin{aligned}
P_w(y \mid x) &= \frac{\int \mathrm{d}z' \exp[-\beta E_w(x,y,z')]}{\int \mathrm{d}y' \int \mathrm{d}z' \exp[-\beta E_w(x,y',z')]} \\
&\overset{\exp[\log(a)]=a}{=} \frac{\exp\left[-\beta(-\frac{1}{\beta}) \log \int \mathrm{d}z' \exp[-\beta E_w(x,y,z')]\right]}{\int \mathrm{d}y' \exp\left[-\beta(-\frac{1}{\beta}) \log \int \mathrm{d}z' \exp[-\beta E_w(x,y',z')]\right]} \\
&= \frac{\exp\left[-\beta\left[-\frac{1}{\beta} \log \int \mathrm{d}z' \exp[-\beta E_w(x,y,z')]\right]\right]}{\int \mathrm{d}y' \exp\left[-\beta\left[-\frac{1}{\beta} \log \int \mathrm{d}z' \exp[-\beta E_w(x,y',z')]\right]\right]} \\
&= \frac{\exp[-\beta F_w(x,y)]}{\int \mathrm{d}y' \exp[-\beta F_w(x,y')]}
\end{aligned}
\tag{A5}
$$

References

[1] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. P. Lillicrap, K. Simonyan *et al.*, *Mastering chess and shogi by self-play with a general reinforcement learning algorithm*, CoRR **abs/1712.01815** (2017), https://arxiv.org/abs/1712.01815.

[2] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe *et al.*, *Mastering the game of Go with deep neural networks and tree search*, Nature **529**(7587), 484 (2016), doi:10.1038/nature16961.

[3] A. W. Senior, R. Evans, J. Jumper, J. Kirkpatrick, L. Sifre, T. Green, C. Qin, A. Žídek, A. W. R. Nelson, A. Bridgland, H. Penedones, S. Petersen *et al.*, *Improved protein structure prediction using potentials from deep learning*, Nature **577**(7792), 706 (2020), doi:10.1038/s41586-019-1923-7.

[4] NLLB Team, M. R. Costa-jussà, J. Cross, O. Çelebi, M. Elbayad, K. Heafield, K. Heffernan, E. Kalbassi, J. Lam, D. Licht, J. Maillard, A. Sun *et al.*, *No language left behind: Scaling human-centered machine translation* (2022), https://arxiv.org/abs/2207.04672.

[5] F. Krzakala and L. Zdeborová, *Summer school on statistical physics and machine learning*, https://leshouches2022.github.io/, École de Physique des Houches (2022).

[6] Y. LeCun, *A path towards autonomous machine intelligence*, Version 0.9.2, 2022-06-27 (2022), https://openreview.net/forum?id=BZ5a1r-kVsf.

[7] A. Krizhevsky, I. Sutskever and G. E. Hinton, *Imagenet classification with deep convolutional neural networks*, In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, p. 1097–1105. Curran Associates Inc., Red Hook, NY, USA, doi:10.1145/3065386 (2012).

[8] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus and Y. LeCun, *Overfeat: Integrated recognition, localization and detection using convolutional networks*, CoRR **abs/1312.6229** (2014), https://arxiv.org/abs/1312.6229.

[9] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, B. C. Van Esesn, A. A. S. Awwal and V. K. Asari, *The history began from alexnet: A comprehensive survey on deep learning approaches* (2018), https://arxiv.org/abs/1803.01164.

[10] E. Commission, *New rules to improve road safety and enable fully driverless vehicles in the eu* (2022), https://ec.europa.eu/commission/presscorner/detail/en/IP_22_4312.

[11] S. Kuutti, R. Bowden, Y. Jin, P. Barber and S. Fallah, *A survey of deep learning applications to autonomous vehicle control* (2021), https://arxiv.org/abs/1912.10773.

[12] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen and I. Sutskever, *Zero-shot text-to-image generation*, https://arxiv.org/abs/2102.12092.

[13] O. Gafni, A. Polyak, O. Ashual, S. Sheynin, D. Parikh and Y. Taigman, *Make-a-scene: Scene-based text-to-image generation with human priors*, https://arxiv.org/abs/2203.13131.

[14] S. Schneider, A. Baevski, R. Collobert and M. Auli, *wav2vec: Unsupervised pre-training for speech recognition* (2019), https://arxiv.org/abs/1904.05862.

[15] A. Conneau, A. Baevski, R. Collobert, A. Mohamed and M. Auli, *Unsupervised cross-lingual representation learning for speech recognition* (2020), https://arxiv.org/abs/2006.13979.

[16] Meta, *Community standards enforcement report*, Q1 2022 report, https://transparency.fb.com/data/community-standards-enforcement.

[17] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz and D. Terzopoulos, *Image segmentation using deep learning: A survey*, https://arxiv.org/abs/2001.05566.

[18] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo and R. Girshick, *Detectron2*, https://github.com/facebookresearch/detectron2 (2019).

[19] S. K. Zhou, H. Greenspan, C. Davatzikos, J. S. Duncan, B. Van Ginneken, A. Madabhushi, J. L. Prince, D. Rueckert and R. M. Summers, *A review of deep learning in medical imaging: Imaging traits, technology trends, case studies with progress highlights, and future promises*, Proceedings of the IEEE **109**(5), 820 (2021), doi:10.1109/JPROC.2021.3054390.

[20] J. Zbontar, F. Knoll, A. Sriram, T. Murrell, Z. Huang, M. J. Muckley, A. Defazio, R. Stern, P. Johnson, M. Bruno, M. Parente, K. J. Geras *et al.*, *fastMRI: An open dataset and benchmarks for accelerated MRI* (2018), https://arxiv.org/abs/1811.08839.

[21] R. Singhal, M. Sudarshan, L. Ginocchio, A. Tong, H. Chandarana, D. Sodickson, R. Ranganath and S. Chopra, *Accelerated MR screenings with direct k-space classification*, Joint Annual Meeting ISMRM-ESMRMB & ISMRT 31st Annual Meeting, 07-12 May 2022, London, UK.

[22] S. Linderman, A. Nichols, D. Blei, M. Zimmer and L. Paninski, *Hierarchical recurrent state space models reveal discrete and continuous dynamics of neural activity in C. elegans*, doi:10.1101/621540 (2019).

[23] J. Lindsey, S. A. Ocko, S. Ganguli and S. Deny, *The effects of neural resource constraints on early visual representations*, In *International Conference on Learning Representations* (2019), https://arxiv.org/abs/1901.00945.

[24] J. Zou, M. Huss, A. Abid, P. Mohammadi, A. Torkamani and A. Telenti, *A primer on deep learning in genomics*, Nat. Genet. **51**, 12–18 (2019), doi:10.1038/s41588-018-0295-5.

[25] M. Baek, F. DiMaio, I. Anishchenko, J. Dauparas, S. Ovchinnikov, G. R. Lee, J. Wang, Q. Cong, L. N. Kinch, R. D. Schaeffer, C. Millán, H. Park *et al.*, *Accurate prediction of protein structures and interactions using a three-track neural network*, Science **373**(6557), 871 (2021), doi:10.1126/science.abj8754, https://www.science.org/doi/pdf/10.1126/science.abj8754.

[26] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto and L. Zdeborová, *Machine learning and the physical sciences*, Rev. Mod. Phys. **91**, 045002 (2019), doi:10.1103/RevModPhys.91.045002.

[27] S. He, Y. Li, Y. Feng, S. Ho, S. Ravanbakhsh, W. Chen and B. Póczos, *Learning to predict the cosmological structure formation*, Proceedings of the National Academy of Sciences **116**(28), 13825 (2019), doi:10.1073/pnas.1821458116, https://www.pnas.org/doi/pdf/10.1073/pnas.1821458116.

[28] A. Dawid, J. Arnold, B. Requena, A. Gresch, M. Płodzień, K. Donatella, K. A. Nicoli, P. Stornati, R. Koch, M. Büttner, R. Okuła, G. Muñoz-Gil *et al.*, *Modern applications of machine learning in quantum sciences* (2022), https://arxiv.org/abs/2204.04198.

[29] J. Hermann, J. Spencer, K. Choo, A. Mezzacapo, W. M. C. Foulkes, D. Pfau, G. Carleo and F. Noé, *Ab-initio quantum chemistry with neural-network wavefunctions* (2022), https://arxiv.org/abs/2208.12590.

[30] C. L. Zitnick, L. Chanussot, A. Das, S. Goyal, J. Heras-Domingo, C. Ho, W. Hu, A. P. Thibaut Lavril and, M. Riviere, M. Shuaibi, A. Sriram, K. Tran *et al.*, *An introduction to electrocatalyst design using machine learning for renewable energy storage*, https://arxiv.org/abs/2010.09435.

[31] S. International, *Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles*, https://www.sae.org/standards/content/j3016_202104/.

[32] K. Craik, *The Nature of Explanation*, University Press, Macmillan, ISBN 978-0521094450 (1943).

[33] G. Marcus and E. Davis, *Rebooting AI: Building artificial intelligence we can trust*, Pantheon, New York, ISBN 1524748250 (2019).

[34] D. Kahneman, *Thinking, Fast and Slow*, Farrar, Straus and Giroux, New York, ISBN 978-0374275631 (2011).

[35] M. Schwenzer, M. Ay, T. Bergs and D. Abel, *Review on model predictive control: an engineering perspective*, Int. J. Adv. Manuf. Technol. **117**, 1327–1349 (2021), doi:10.1007/s00170-021-07682-3.

[36] Y. LeCun and I. Misra, *Self-supervised learning: The dark matter of intelligence*, https://ai.facebook.com/blog/self-supervised-learning-the-dark-matter-of-intelligence/.

[37] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato and F. J. Huang, *A tutorial on energy-based learning*, http://yann.lecun.com/exdb/publis/pdf/lecun-06.pdf (2006).

[38] P. Huembeli, J. M. Arrazola, N. Killoran, M. Mohseni and P. Wittek, *The physics of energy-based models*, Quantum Mach. Intell. **4**(1) (2022), doi:10.1007/s42484-021-00057-7.

[39] A. Hannun, *The label bias problem*, https://awni.github.io/label-bias/ (2019).

[40] A. van den Oord, Y. Li, and O. Vinyals, *Representation learning with contrastive predictive coding* (2018), https://arxiv.org/abs/1807.03748.

[41] D. T. Hoffmann, N. Behrmann, J. Gall, T. Brox and M. Noroozi, *Ranking info noise contrastive estimation: Boosting contrastive learning via ranked positives* (2022), https://arxiv.org/abs/2201.11736.

[42] G. E. Hinton, *Training products of experts by minimizing contrastive divergence*, Neural Comput. **14**(8), 1771–1800 (2002), doi:10.1162/089976602760128018.

[43] S. Chopra, R. Hadsell and Y. LeCun, *Learning a similarity metric discriminatively, with application to face verification*, In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 539–546, doi:10.1109/CVPR.2005.202 (2005).

[44] J. Zhao, M. Mathieu and Y. LeCun, *Energy-based generative adversarial network* (2016), https://arxiv.org/abs/1609.03126.

[45] M. Arjovsky, S. Chintala and L. Bottou, *Wasserstein GAN* (2017), https://arxiv.org/abs/1701.07875.

[46] A. Hyvärinen, *Estimation of non-normalized statistical models by score matching*, Journal of Machine Learning Research **6**(24), 695 (2005), http://jmlr.org/papers/v6/hyvarinen05a.html.

[47] Y. LeCun and A. Canziani, *Deep learning (ds-ga 1008)*, https://atcold.github.io/pytorch-Deep-Learning/, Spring 2020 course for New York University Center For Data Science (2020).

[48] J. J. Hopfield, *Neural networks and physical systems with emergent collective computational abilities*, Proc. Natl. Acad. Sci. U.S.A. **79**(8), 2554 (1982), doi:10.1073/pnas.79.8.2554.

[49] W. Little, *The existence of persistent states in the brain*, Mathematical Biosciences **19**(1), 101 (1974), doi:https://doi.org/10.1016/0025-5564(74)90031-5.

[50] G. E. Hinton and T. J. Sejnowski, *Optimal perceptual inference*, In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition* (1983).

[51] P. Vincent, H. Larochelle, Y. Bengio and P.-A. Manzagol, *Extracting and composing robust features with denoising autoencoders*, In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, p. 1096–1103. Association for Computing Machinery, New York, NY, USA, ISBN 9781605582054, doi:10.1145/1390156.1390294 (2008).

[52] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu and P. Kuksa, *Natural language processing (almost) from scratch*, J. Mach. Learn. Res. **12**, 2493–2537 (2011), https://arxiv.org/abs/1103.0398.

[53] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, *BERT: Pre-training of deep bidirectional transformers for language understanding*, In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota, doi:10.18653/v1/N19-1423 (2019).

[54] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer and V. Stoyanov, *RoBERTa: A Robustly Optimized BERT Pretraining Approach* (2019), https://arxiv.org/abs/1907.11692.

[55] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, T. Mihaylov, M. Ott *et al.*, *OPT: Open Pre-trained Transformer language models* (2022), https://arxiv.org/abs/2205.01068.

[56] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell and A. A. Efros, *Context encoders: Feature learning by inpainting* (2016), https://arxiv.org/abs/1604.07379.

[57] K. He, X. Chen, S. Xie, Y. Li, P. Dollár and R. Girshick, *Masked autoencoders are scalable vision learners* (2021), https://arxiv.org/abs/2111.06377.

[58] A. Bardes, J. Ponce and Y. LeCun, *VICReg: Variance-invariance-covariance regularization for self-supervised learning*, In *International Conference on Learning Representations* (2022), https://arxiv.org/abs/2105.04906.

[59] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu *et al.*, *Bootstrap your own latent: A new approach to self-supervised learning* (2020), https://arxiv.org/abs/2006.07733.

[60] X. Chen and K. He, *Exploring simple Siamese representation learning* (2020), https://arxiv.org/abs/2011.10566.

[61] J. Zbontar, L. Jing, I. Misra, Y. LeCun and S. Deny, *Barlow twins: Self-supervised learning via redundancy reduction*, In M. Meila and T. Zhang, eds., *Proceedings of the 38th International Conference on Machine Learning*, vol. 139 of *Proceedings of Machine Learning Research*, pp. 12310–12320. PMLR (2021), https://arxiv.org/abs/2103.03230.

[62] S. Becker and G. E. Hinton, *Self-organizing neural network that discovers surfaces in random-dot stereograms*, Nature **355**(6356), 161 (1992), doi:10.1038/355161a0.