Discovering physical concepts with neural networks

Raban Iten[1 2], Tony Metger[1 2], Henrik Wilming[1], Lídia del Rio[1], and Renato Renner[1]

[1]Institute for Theoretical Physics, ETH Zürich, 8093 Zürich, Switzerland
[2]These authors contributed equally to this work.

While neural networks have been remarkably successful for a variety of practical problems, they are often applied as a black box, which limits their utility for scientific discoveries. Here, we present a neural network architecture that can be used to discover physical concepts from experimental data without being provided with additional prior knowledge. For a variety of simple systems in classical and quantum mechanics, our network learns to compress experimental data to a simple representation and uses the representation to answer questions about the physical system. Physical concepts can be extracted from the learned representation, namely: **(1)** The representation stores the physically relevant parameters, like the frequency of a pendulum. **(2)** The network finds and exploits conservation laws: it stores the total angular momentum to predict the motion of two colliding particles. **(3)** Given measurement data of a simple quantum mechanical system, the network correctly recognizes the number of degrees of freedom describing the underlying quantum state. **(4)** Given a time series of the positions of the Sun and Mars as observed from Earth, the network discovers the heliocentric model of the solar system — that is, it encodes the data into the angles of the two planets as seen from the Sun. Our work provides a first step towards answering the question whether the traditional ways by which physicists model nature naturally arise from the experimental data without any mathematical and physical pre-knowledge, or if there are alternative elegant formalisms, which may solve some of the fundamental conceptual problems in modern physics, such as the measurement problem in quantum mechanics.

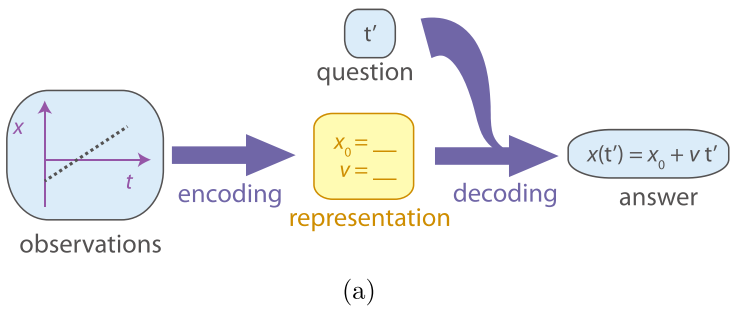
1 Introduction

Theoretical physics, like all fields of human activity, is influenced by the schools of thought prevalent at the time of development. As such, the physical theories we know may not necessarily be the simplest ones to explain experimental data, but rather the ones that most naturally followed from a previous theory at the time. The formalism of quantum theory, for instance, is built upon classical mechanics; it has been impressively successful, but leads to conceptually challenging consequences (see [1, 2] for a review and [3] for a recent example).

This raises an interesting question: are the laws of quantum physics, and other physical theories more generally, the most natural ones to explain data from experiments if we assume no prior knowledge of physics? While this question will likely not be answered fully in the near future, recent advances in artificial intelligence allow us to make a first step in this direction. Here, we investigate whether neural networks can be used to discover physical concepts in classical and quantum mechanics from experimental data.

While neural networks have been applied to a variety of problems in physics, most work to date has focused on the efficiency or quality of predictions of neural networks, without an understanding how they solve the problem [4–9] (see Section 4.1 and [10] for a review and further references). Other algorithms and neural network architectures have been developed to produce a physical model by imposing some structure on the space of solutions and on the input data [11–18]. For example, in [12], an algorithm recovers the

Raban Iten: itenr@itp.phys.ethz.ch
Tony Metger: tmetger@ethz.ch



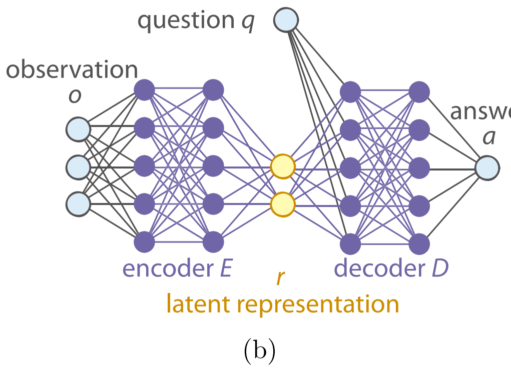


Figure 1: **Learning physical representations. (a) Human learning.** A physicist compresses experimental observations into a simple representation (*encoding*). When later asked any question about the physical setting, the physicist should be able to produce a correct answer using only the representation and not the original data. The process of producing the answer (by applying a physical model to the representation) is called *decoding*. For example, the observations may be the first few seconds of the trajectory of a particle moving with constant speed; the representation could be the parameters "speed v" and "initial position x_0" and the question could be "where will the particle be at a later time t'?" **(b) Neural network structure for *SciNet*.** Observations are encoded as real parameters fed to an encoder (a *feed-forward neural network*, see Appendix A), which compresses the data into a representation (*latent representation*). The question is also encoded in a number of real parameters, which, together with the representation, are fed to the decoder network to produce an answer. Note that the number of layers and neurons depicted is not representative.

laws of motion of simple mechanical systems, like a double pendulum, by searching over a space of mathematical expressions on the input variables. To apply this method, one must specify *a priori* which variables may enter the mathematical models, and how they can be combined; in other words, one must impose on the network our intuition of which parameters will be physically relevant. In contrast, here we are interested in questioning those same intuitions, and asking whether an unconstrained neural network would characterize physical settings through parameters and representations similar to those of standard physics textbooks.

1.1 Network structure: *SciNet*

To approach this task, we apply machine learning techniques and use ideas from representation learning [19–24]. Concretely, we introduce a neural network architecture, which we call *SciNet* for brevity, which mimics a physicist's modelling process (Figure 1a), and apply it to study various physical scenarios. The idea is that the physicist (or *SciNet*) is exposed to some experimental observations pertaining to a physical setting (e.g. a time series $(t_i, x(t_i))_{i\in\{1,\ldots,N\}}$ describing the movement of a particle at constant speed), finds a simpler representation (e.g. the two parameters initial position and speed, (x_0, v)), and will later be asked a question about this physical setting (e.g. "where is the particle at time t'?"). Ideally, the representation should be as compact as possible while still fully characterizing the physics of the situation, so that the physicist may forget the original data and answer the question using only the representation (Section 3).

For a purely input-output (black box) analysis, the modelling process of *SciNet* can be seen as a map $F : \mathcal{O} \times \mathcal{Q} \to \mathcal{A}$ from the sets of possible observations $\mathcal{O}$ and questions $\mathcal{Q}$ to the set of possible answers $\mathcal{A}$. We can split this map into an *encoder* $E : \mathcal{O} \to \mathcal{R}$ mapping the original observation to a compressed representation (called *latent representation* in machine learning), followed by a *decoder* $D : \mathcal{R} \times \mathcal{Q} \to \mathcal{A}$ that takes the representation and the question to produce an answer. The corresponding network structure is shown in Figure 1b. A similar network architecture was recently applied for scene representation and rendering [25]. It is this decomposition, $F(o, q) = D(E(o), q)$, that will allow us to implement the encoder and decoder as a neural network in such a way that we can interpret the network's learned representation, by analyzing how it changes as we tweak known parameters of the setting. In order to force *SciNet* to find and exploit the physical structure of a prob-

Box 1: Time evolution of a damped pendulum (Section 2.1)

Problem: Predict the position of a one-dimensional damped pendulum at different times.

Physical model: Equation of motion: $m\ddot{x} = -\kappa x - b\dot{x}$.
Solution: $x(t) = A_0 e^{-\frac{b}{2m}t}\cos(\omega t + \delta_0)$, with $\omega = \sqrt{\frac{\kappa}{m}}\sqrt{1 - \frac{b^2}{4m\kappa}}$.

Observation: Time series of positions: $o = \left[x(t_i)\right]_{i\in\{1,\dots,50\}} \in \mathbb{R}^{50}$, with equally spaced t_i. Mass $m = 1\text{kg}$, amplitude $A_0 = 1\text{m}$ and phase $\delta_0 = 0$ are fixed; spring constant $\kappa \in [5, 10]$ kg/s^2 and damping factor $b \in [0.5, 1]$ kg/s are varied between training samples.

Question: Prediction times: $q = t_{\text{pred}} \in \mathbb{R}$.

Correct answer: Position at time t_{pred}: $a_{\text{cor}} = x(t_{\text{pred}}) \in \mathbb{R}$.

Implementation: Network depicted in Figure 1b with 3 latent neurons.

Key findings:

- *SciNet* predicts the positions $x(t_{\text{pred}})$ with a root mean square error below 2% (with respect to the amplitude $A_0 = 1\text{m}$) (Figure 2a).
- *SciNet* stores κ and b in two of the latent neurons, and does not store any information in the third latent neuron (Figure 2b).

lem, we encourage it to minimize the number of neurons in the latent representation and their correlations (see Section 2 for examples and Section 3 for theoretical considerations). We use fully connected feed-forward neural networks to implement the encoder and the decoder of *SciNet* (see Appendix A for an introduction to neural networks).

1.2 Training and testing *SciNet*

We train *SciNet* with data samples of the form $(o, q, a_{\text{cor}}(o, q))$, where the observation o and question q are chosen from the sets $\mathcal{O}$ and $\mathcal{Q}$ of all possible observations and questions, respectively, and where $a_{\text{cor}}(o, q)$ denotes the correct reply to question q given observation o.

We consider families of encoders E_ϕ and decoders D_θ, whose parameters ϕ and θ include the weights and biases of the corresponding networks. During training, we encourage *SciNet* to adapt its free parameters ϕ and θ to improve its prediction accuracy and to learn *minimal uncorrelated representations* (see Section 3, Appendix B). For this, we use methods from representation learning, specifically disentangling variational autoencoders [19, 23, 26, 27]. However, finding minimal uncorrelated representations reliable (and efficiently) is still a problem of current research.

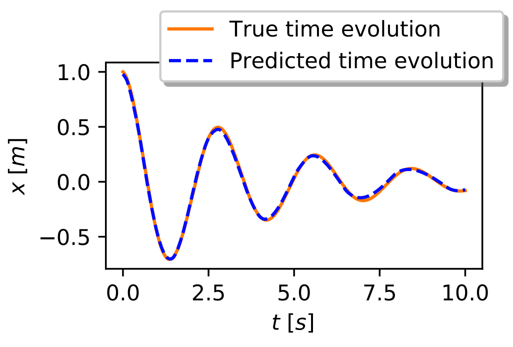
The structure of the training data and of the network does not fall into the standard categorisation of "unsupervised" versus "supervised". However, *SciNet* can be regarded as a generalisation of the idea of autoencoders for all examples presented here, since the answers to the questions correspond to subsets of collected measurement data and do not require human labelling. In this sense, the training is unsupervised.

To make good predictions, *SciNet* needs a sufficient number of latent neurons. Knowing nothing about the physical system under consideration, we might choose this number to be too small. In that case, *SciNet* will make predictions with low accuracy and we can set up a new network with more latent neurons. To determine the prediction accuracy, we test *SciNet* on new data samples (*test data*), which have not been seen during training.

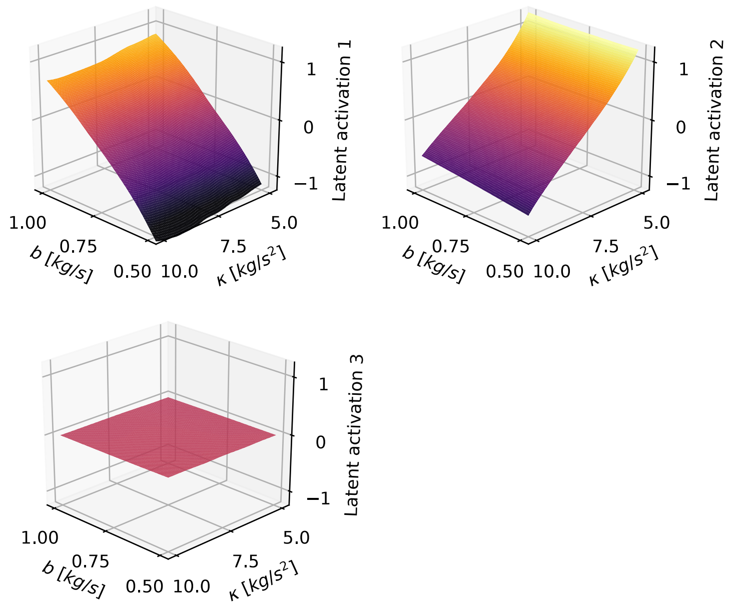
Once the network is trained to make accurate predictions, our aim is to read out conceptual information from the representation it found. For any given observation, we consider the activations of the neurons in the latent representation (which are real numbers) as the values of the (unknown) physical variables present in the observation.

2 Results

We demonstrate with four examples how *SciNet* is able to recover the relevant physical variables, both in quantum and in classical systems. Moreover, we will show that *SciNet* can be used to recover important concepts of physics, like conserved quantities or the heliocentric model of our solar system. A theoretical analysis follows in Sec-



(a)



(b)

Figure 2: **Damped pendulum**. *SciNet* is fed a time series of the trajectory of a damped pendulum. It learns to store the two relevant physical parameters, frequency and damping, in the representation, and makes correct predictions about the pendulum's future position. **(a) Trajectory prediction of *SciNet*.** Here, the spring constant is $\kappa = 5\text{kg/s}^2$ and the damping factor is $b = 0.5\text{kg/s}$. *SciNet's* prediction is in excellent agreement with the true time evolution. **(b) Representation learned by *SciNet*.** The plots show the activations of the three latent neurons of *SciNet* as a function of the spring constant κ and the damping factor b. The first two neurons store the damping factor and spring constant, respectively. The activation of the third neuron is close to zero, suggesting that only two physical variables are required. On an abstract level, learning that one activation can be set to a constant is encouraged by searching for uncorrelated latent variables, i.e., by minimizing the common information of the latent neurons during training (see Section 3).

tion 3.

2.1 Damped pendulum

We start with a simple example from classical physics, the damped pendulum, described in Box 1. The time evolution of the system is given by the differential equation $-\kappa x - b\dot{x} = m\ddot{x}$, where κ is the *spring constant*, which determines the frequency of the oscillation, and b is the *damping factor*. We keep the mass m constant (it is a scaling factor that could be absorbed by defining $\kappa' = \kappa/m$ and $b' = b/m$), such that κ and b are the only variable parameters. We consider the case of weak damping here, where the solution to the equation of motion is given in Box 1.

We choose a network structure for *SciNet* with 3 latent neurons. As an input, we provide a time series of positions of the pendulum and we ask *SciNet* to predict the position at a future time (see Box 1 for details). The accuracy of the predictions given by *SciNet* after training is illustrated in Figure 2a.

Without being given any physical concepts, *SciNet* learns to extract the two relevant physical parameters from (simulated) time series data for the x-coordinate of the pendulum and to store them in the latent representation. As shown in Figure 2b, the first latent neuron depends nearly linearly on b and is almost independent of κ, and the second latent neuron depends only on κ, again almost linearly. Hence, *SciNet* has recovered the same time-independent parameters b and κ that are used by physicists. The third latent neuron is nearly constant and does not provide any additional information — in other words, *SciNet* recognized that two parameters suffice to encode this situation.

2.2 Conservation of angular momentum

One of the most important concepts in physics is that of conservation laws, such as conservation of energy and angular momentum. While their re-

Box 2: Collision of two bodies under angular momentum conservation (Section 2.2)

Problem: Predict the position of a particle fixed on a rod of radius r (rotating about the origin) after a collision at the point $(0, r)$ with a free particle (in two dimensions, see Figure 3a).

Physical model: Given the total angular momentum before the collision and the velocity of the free particle after the collision, the position of the rotating particle at time t'_{pred} (after the collision) can be calculated from angular momentum conservation: $J = m_{\text{rot}} r^2 \omega - r m_{\text{free}} (\mathbf{v}_{\text{free}})_x = m_{\text{rot}} r^2 \omega' - r m_{\text{free}} (\mathbf{v}'_{\text{free}})_x = J'$.

Observation:
Time series of both particles before the collision: $o = [(t_i^{\text{rot}}, \mathbf{q}_{\text{rot}}(t_i^{\text{rot}})), (t_i^{\text{free}}, \mathbf{q}_{\text{free}}(t_i^{\text{free}}))]_{i \in \{1,\dots,5\}}$, with times t_i^{rot} and t_i^{free} randomly chosen for each training sample. Masses $m_{\text{rot}} = m_{\text{free}} = 1\text{kg}$ and the orbital radius $r = 1\text{m}$ are fixed; initial angular velocity ω, initial velocity $\mathbf{v}_{\text{free}}$ and final velocity $\mathbf{v}'_{\text{free}}$ are varied between training samples. Gaussian noise ($\mu = 0, \sigma = 0.01\text{m}$) is added to all position inputs.

Question: Prediction time and position of free particle after collision: $q = \left(t'_{\text{pred}}, [t'_i, \mathbf{q}'_{\text{free}}(t'_i)]_{i \in \{1,\dots,5\}}\right)$.

Correct answer: Position of rotating particle at time t'_{pred}: $a_{\text{cor}} = \mathbf{q}'_{\text{rot}}(t'_{\text{pred}})$.

Implementation: Network depicted in Figure 1b with one latent neuron.

Key findings:

- *SciNet* predicts the position of the rotating particle with root mean square prediction error below 4% (with respect to the radius $r = 1\text{m}$).
- *SciNet* is resistant to noise.
- *SciNet* stores the total angular momentum in the latent neuron.

lation to symmetries makes them interesting to physicists in their own right, conservation laws are also of practical importance. If two systems interact in a complex way, we can use conservation laws to predict the behaviour of one system from the behaviour of the other, without studying the details of their interaction. For certain types of questions, conserved quantities therefore act as a compressed representation of joint properties of several systems.

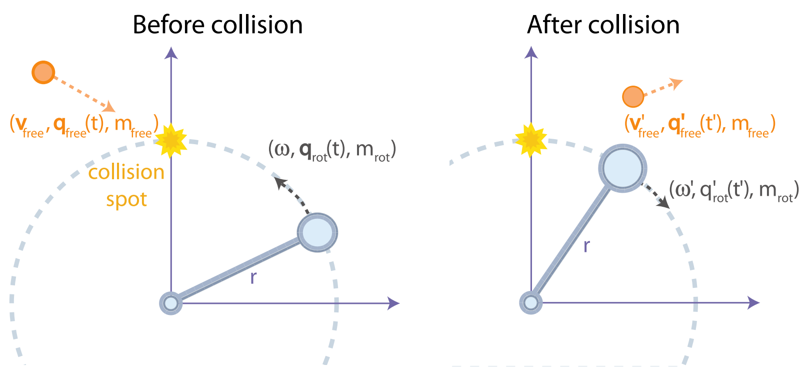
We consider the scattering experiment shown in Figure 3 and described in Box 2, where two point-like particles collide. Given the initial angular momentum of the two particles and the final trajectory of one of them, a physicist can predict the trajectory of the other using conservation of total angular momentum.

To see whether *SciNet* makes use of angular momentum conservation in the same way as a physicist would do, we train it with (simulated) experimental data as described in Box 2 with one latent neuron, and add Gaussian noise to show that the encoding and decoding are robust. Indeed, *SciNet* does exactly what a physicist would do and stores the total angular momentum in the latent representation (Figure 3b). This example shows that *SciNet* can recover conservation laws, and suggests that they emerge naturally from compressing data and asking questions about joint properties of several systems.

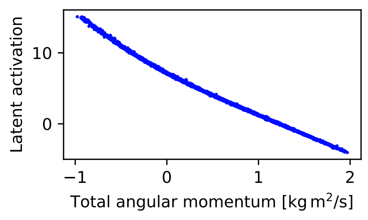
2.3 Representation of qubits

In quantum mechanics, it is not trivial to construct a simple representation of the state of a quantum system from measurement data. Indeed, *quantum state tomography* is an active area of research [28]. Ideally, we look for a *faithful representation* of the state of a quantum system, such as the wave function: a representation that stores all information necessary to predict the probabilities of the outcomes for arbitrary measurements on that system. However, to specify a faithful representation of a quantum system it is not necessary to perform all theoretically possible measurements on the system. If a set of measurements is sufficient to reconstruct the full quantum state, such a set is called *tomographically complete*.

Here we show that, based only on (simulated) experimental data and without being given any assumptions about quantum theory, *SciNet* recovers a faithful representation of the state of small quan-


Before collision
After collision
$(\mathbf{v}_{\text{free}}, \mathbf{q}_{\text{free}}(t), m_{\text{free}})$
collision spot
$(\omega, \mathbf{q}_{\text{rot}}(t), m_{\text{rot}})$
r
$(\mathbf{v}'_{\text{free}}, \mathbf{q}'_{\text{free}}(t'), m_{\text{free}})$
$(\omega', q'_{\text{rot}}(t'), m_{\text{rot}})$
r

(a)



(b)

Figure 3: **Collision under conservation of angular momentum.** In a classical mechanics scenario where the total angular momentum is conserved, the neural network learns to store this quantity in the latent representation. **(a) Physical setting.** A body of mass m_{rot} is fixed on a rod of length r (and of negligible mass) and rotates around the origin with angular velocity ω. A free particle with velocity $\mathbf{v}_{\text{free}}$ and mass m_{free} collides with the rotating body at position $\mathbf{q} = (0, r)$. After the collision, the angular velocity of the rotating particle is ω' and the free particle is deflected with velocity $\mathbf{v}'_{\text{free}}$. **(b) Representation learned by *SciNet*.** Activation of the latent neuron as a function of the total angular momentum. *SciNet* learns to store the total angular momentum, a conserved quantity of the system.

tum systems and can make accurate predictions. In particular, this allows us to infer the dimension of the system and distinguish tomographically complete from incomplete measurement sets. Box 3 summarizes the setting and the results.

A (pure) state on n qubits can be represented by a normalized complex vector $\psi \in \mathbb{C}^{2^n}$, where two states ψ and ψ' are identified if and only if they differ by a global phase factor, i.e., if there exists $\phi \in \mathbb{R}$ such that $\psi = e^{i\phi}\psi'$. The normalization condition and irrelevance of the global phase factor decrease the number of free parameters of a quantum state by two. Since a complex number has two real parameters, a single-qubit state is described by $2 \times 2^1 - 2 = 2$ real parameters, and a state of two qubits is described by $2 \times 2^2 - 2 = 6$ real parameters.

Here, we consider binary projective measurements on n qubits. Just like states, these measurements can be described by vectors $\omega \in \mathbb{C}^{2^n}$, with measurement outcomes labeled by 0 for the projection on ω and 1 otherwise. The probability to get outcome 0 when measuring ω on a quantum system in state ψ is then given by $p(\omega, \psi) = |\langle \omega, \psi \rangle|^2$, where $\langle \cdot, \cdot \rangle$ denotes the standard scalar product on $\mathbb{C}^{2^n}$.

To generate the training data for *SciNet*, we assume that we have one or two qubits in a lab that can be prepared in arbitrary states and we have the ability to perform binary projective measurements in a set $\mathcal{M}$. We choose n_1 measurements $\mathcal{M}_1 := \{\alpha_1, \dots, \alpha_{n_1}\} \subset \mathcal{M}$ randomly, which we would like to use to determine the state of the quantum system. We perform all measurements in $\mathcal{M}_1$ several times on the same quantum state ψ to estimate the probabilities $p(\alpha_i, \psi)$ of measuring 0 for the i-th measurement. These probabilities form the observation given to *SciNet*.

To parameterize the measurement ω, whose outcome probabilities should be predicted by *SciNet*, we choose another random set of measurements $\mathcal{M}_2 := \{\beta_1, \dots, \beta_{n_2}\} \subset \mathcal{M}$. The probabilities $p(\beta_i, \omega)$ are provided to *SciNet* as the question input. We always assume that we have chosen enough measurements in $\mathcal{M}_2$ such that they can distinguish all the possible measurements $\omega \in \mathcal{M}$, i.e., we assume that $\mathcal{M}_2$ is tomographically complete.[1] *SciNet* then has to predict the probability

[1]This parameterization of a measurement ω assumes that we know the equivalence between binary projective measurements and states. However, this is not a fundamental assumption, since we could parameterize the set of possible measurements by any parameterization that is natural for the experimental setup, for example the settings of the dials and buttons on an experimental apparatus. Such a natural parameterization is assumed to fully specify the measurement, in the sense that the same settings on the experimental apparatus will always result in the same measurement being performed. Because $\mathcal{M}_2$ only represents our choice for parameterizing the measurement setup, it is natural to assume that $\mathcal{M}_2$ is tomographically complete.

Box 3: Representation of pure one- and two-qubit states (Section 2.3)

Problem: Predict the measurement probabilities for any binary projective measurement $\omega \in \mathbb{C}^{2^n}$ on a pure n-qubit state $\psi \in \mathbb{C}^{2^n}$ for $n = 1, 2$.

Physical model: The probability $p(\omega, \psi)$ to measure 0 on the state $\psi \in \mathbb{C}^{2^n}$ performing the measurement $\omega \in \mathbb{C}^{2^n}$ is given by $p(\omega, \psi) = |\langle \omega, \psi \rangle|^2$.

Observation: Operational parameterization of a state ψ: $o = [p(\alpha_i, \psi)]_{i \in \{1,\dots,n_1\}}$ for a fixed set of random binary projective measurements $\mathcal{M}_1 := \{\alpha_1, \dots, \alpha_{n_1}\}$ ($n_1 = 10$ for one qubit, $n_1 = 30$ for two qubits).

Question: Operational[1] parameterization of a measurement ω: $q = [p(\beta_i, \omega)]_{i \in \{1,\dots,n_2\}}$ for a fixed set of random binary projective measurements $\mathcal{M}_2 := \{\beta_1, \dots, \beta_{n_2}\}$ ($n_2 = 10$ for one qubit, $n_2 = 30$ for two qubits).

Correct answer: $a_{\text{cor}}(\omega, \psi) = p(\omega, \psi) = |\langle \omega, \psi \rangle|^2$.

Implementation: Network depicted in Figure 1b with varying numbers of latent neurons.

Key findings:

- *SciNet* can be used to determine the minimal number of parameters necessary to describe the state ψ (see Figure 4) without being provided with any prior knowledge about quantum physics.
- *SciNet* distinguishes tomographically complete and incomplete sets of measurements (see Figure 4).

$p(\omega, \psi)$ for measuring the outcome 0 on the state ψ when performing the measurement ω.

We train *SciNet* with different pairs (ω, ψ) for one and two qubits, keeping the measurement sets $\mathcal{M}_1$ and $\mathcal{M}_2$ fixed. We choose $n_1 = n_2 = 10$ for the single-qubit case and $n_1 = n_2 = 30$ for the two-qubit case. The results are shown in Figure 4.

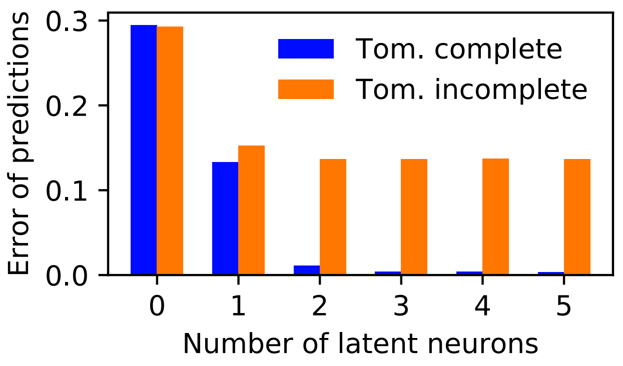
Varying the number of latent neurons, we can observe how the quality of the predictions improves as we allow for more parameters in the representation of ψ. To minimize statistical fluctuations due to the randomized initialization of the network, each network specification is trained three times and the run with the lowest mean square prediction error on the test data is used.

For the cases where $\mathcal{M}_1$ is tomographically complete, the plots in Figure 4 show a drop in prediction error when the number of latent neurons is increased up to two or six for the cases of one and two qubits, respectively.[2] This is in accordance with the number of parameters required to describe a one- or a two-qubit state. Thus, *SciNet* allows us to extract the dimension of the underlying quantum system from tomographically complete measurement data, without any prior information about quantum mechanics.

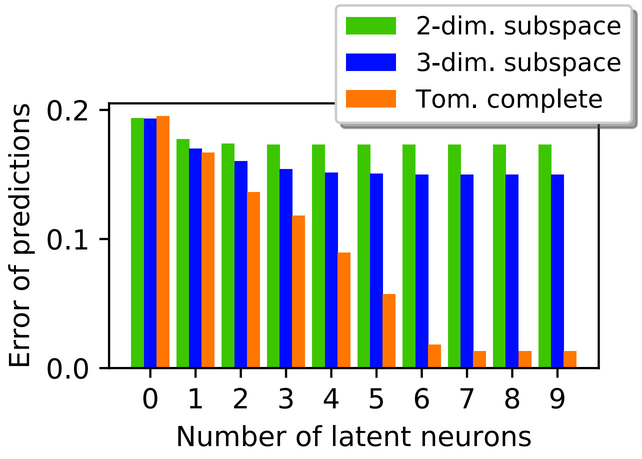
SciNet can also be used to determine whether the measurement set $\mathcal{M}_1$ is tomographically complete or not. To generate tomographically incomplete data, we choose the measurements in $\mathcal{M}_1$ randomly from a subset of all binary projective measurements. Specifically, the quantum states corresponding to measurements in $\mathcal{M}_1$ are restricted to random *real* linear superpositions of k orthogonal states, i.e., to a (real) k-dimensional subspace. For a single qubit, we use a two-dimensional subspace; for two quibts, we consider both two- and three-dimensional subspaces.

Given tomographically incomplete data about a state ψ, it is not possible for *SciNet* to predict the outcome of the final measurement perfectly regardless of the number of latent neurons, in contrast to the tomographically complete case (see Figure 4). Hence, we can deduce from *SciNet's* output that $\mathcal{M}_1$ is an incomplete set of measurements. Furthermore, this analysis provides a qualitative measure for the amount of information provided by the tomographically incomplete measurements: in the two-qubit case, increasing the subspace dimension from two to three leads to higher prediction accuracy and the required number of latent neurons increases.

[2]In the case of a single qubit, there is an additional small improvement in going from two to three latent neurons: this is a technical issue caused by the fact that any two-parameter representation of a single qubit, for example the Bloch sphere representation, includes a *cyclic parameter*, which cannot be exactly represented by a continuous encoder (Appendix D). The same likely applies in the case of two qubits, going from 6 to 7 latent neurons. This restriction also makes it difficult to interpret the details of the learned representation.



(a) One qubit.



(b) Two qubits.

Figure 4: **Quantum tomography.** *SciNet* is given tomographic data for one or two qubits and an operational description of a measurement as a question input and has to predict the probabilities of outcomes for this measurement. We train *SciNet* with both tomographically complete and incomplete sets of measurements, and find that, given tomographically complete data, *SciNet* can be used to find the minimal number of parameters needed to describe a quantum state (two parameters for one qubit and six parameters for two qubits). Tomographically incomplete data can be recognized, since *SciNet* cannot achieve perfect prediction accuracy in this case, and the prediction accuracy can serve as an estimate for the amount of information provided by the tomographically incomplete set. The plots show the root mean square error of *SciNet's* measurement predictions for test data as a function of the number of latent neurons.

2.4 Heliocentric model of the solar system

When observed from Earth, the orbits of the Sun and the other planets in our solar system take complicated shapes. In the 16th century, Copernicus measured the angles between a distant fixed star and several planets and celestial bodies (Figure 6a) and hypothesized that the Sun, and not the Earth, is in the centre of our solar system and that the planets move around the Sun on simple orbits. This explains the complicated orbits as seen from Earth.

Here, we show that *SciNet* similarly uses heliocentric angles when forced to find a representation for which the time evolution of the variables takes a very simple form, a typical requirement for time-dependent variables in physics.

In the first three examples, we saw that *SciNet* is able to learn representations of time-independent parameters of different physical systems. In order to study the time evolution of the variables stored in the network's representation, we extend the network structure slightly (Figure 5).

Ideally we want the representation to store variables that evolve under simple rules — in this example, if *SciNet* stored the angles as seen from the Sun, the evolution would take a very simple form, whereas evolving the angles as seen from Earth requires the implementation of a much more involved time evolution. In order to find variables with a simple time evolution, we add a very small feed-forward neural network after the representation, which is meant to evolve the variables by a time step Δt. The evolution over a longer period of time may then be simulated by concatenating a series of identical time evolution networks and representation layers $\{r(t_i)\}_i$. We thus model the time evolution using a *recurrent neural network*.

After each step of the time evolution, we may again ask a question about the representation $r(t_i)$ — this is done by attaching a decoder D to the representation, as before. In this example we always ask the same question: "what are the angles as seen from Earth at the time t_i?" Hence, we do not need to feed a question as a new input to the decoder explicitly (but using an explicit question is in principle possible). The decoder at each time step is identical, and it receives the representation $r(t_i)$ as an input.

In this example, we restrict the latent layer to

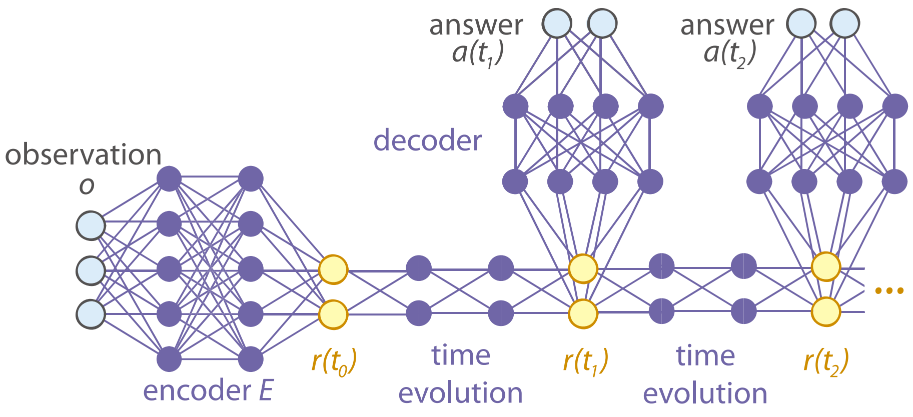


Figure 5: **Recurrent version of *SciNet* for time-dependent variables.** Observations are encoded into a simple representation $r(t_0)$ at time t_0. Then, the representation is evolved in time to $r(t_1)$ and a decoder is used to predict $a(t_1)$, and so on. In each (equally spaced) time step, the same time evolution network and decoder network are applied.

two neurons, and the time evolution network to maps of the simple form $r_j(t_i) \to r_j(t_i) + b_j$ on the j-th component $r_j(t_i)$ of the representation, where the biases b_j are the same for all time steps.[3] The setting is summarized in Box 4.

The activations of the two latent neurons of the trained network are plotted in Figure 6b. As can be seen from the plots, the activations of the latent representation are given by a linear combination of the angles ϕ_E and ϕ_M as seen from the Sun — *SciNet* has discovered the heliocentric model of the solar system. *SciNet* is not expected to store the angles ϕ_E and ϕ_M in separate neurons, since the time evolution update rule for any linear combination of the angles is still of the same simple form specified above.

3 Minimal representations

Here, we describe some of the theoretical considerations that went into designing *SciNet* and helping it to find useful representations that encode physical principles. Given a data set, it is generally a complex task to find a simple representation of the data that contains all the desired information. *SciNet* should recover such representations by itself; however, we encourage it to learn "simple" representations during training. To do so, we have to specify the desired properties of a representation. In this, our approach follows the spirit of several works on representation learning theory [19–24].

For the theoretical analysis, we introduce some additional structure on the data that is required to formulate the desired properties of a representation. We consider real-valued data, which we think of as being sampled from some unknown probability distribution. In other words, we assign random variables to the observations O, the questions Q, the latent representation R, and the answers A. We use the convention that a random variable $X = (X_1, \dots, X_{|X|})$ takes samples in $\mathcal{X} \subset \mathbb{R}^{|X|}$, where $|X|$ denotes the dimension of the ambient space of X. In particular, $|R|$ will correspond to the number of neurons in the latent representation.

We require the following properties for an *uncorrelated (sufficient) representation* R (defined through an encoder mapping $E : \mathcal{O} \to \mathcal{R}$) for the data described by the triple (O, Q, a_{cor}), where we recall that the function $a_{\text{cor}} : \mathcal{O} \times \mathcal{Q} \to \mathcal{A}$ sends an observation $o \in \mathcal{O}$ and a question $q \in \mathcal{Q}$ to the correct answer $a \in \mathcal{A}$.

1. **Sufficient (with smooth decoder)**: There exists a smooth map $D : \mathcal{R} \times \mathcal{Q} \mapsto \mathcal{A}$, such that $D(E(o), q) = a_{\text{cor}}(o, q)$ for all possible observations $o \in \mathcal{O}$ and questions $q \in \mathcal{Q}$.

[3]For a general system, there might not exist a representation that admits such a simple time evolution. In this case, one has to proceed as follows: first, the time evolution is restricted to the simplest possible case, i.e., addition of a constant. If the network is unable to achieve good prediction accuracy, more complexity is added to the time evolution network until the network is able to make good predictions. Following this prescription, one will end up with the representation of the system that admits the simplest possible time evolution. However, for time evolution steps that are more complex than an affine function, it remains to be investigated how to quantify the complexity of the time evolution network in a meaningful way.

Box 4: Heliocentric model of the solar system (Section 2.4)

Problem: Predict the angles $\theta_M(t)$ and $\theta_S(t)$ of Mars and the Sun as seen from Earth, given initial states $\theta_M(t_0)$ and $\theta_S(t_0)$.

Physical model: Earth and Mars orbit the Sun with constant angular velocity on (approximately) circular orbits.

Observation: Initial angles of Mars and the Sun as seen from Earth: $o = (\theta_M(t_0), \theta_S(t_0))$, randomly chosen from a set of weekly (simulated) observations within Copernicus' lifetime (3665 observations in total).

Question: Implicit.

Correct answer: Time series $\big[a(t_1), \ldots, a(t_n)\big] = \big[(\theta_M(t_1), \theta_S(t_1)), \ldots, (\theta_M(t_n), \theta_S(t_n))\big]$ of $n = 20$ (later in training: $n = 50$) observations, with time steps $t_{i+1} - t_i$ of one week.

Implementation: Network depicted in Figure 5 with two latent neurons.

Key findings:

- *SciNet* predicts the angles of Mars and the Sun with a root mean square error below 0.4% (with respect to 2π).
- *SciNet* stores the angles ϕ_E and ϕ_M of the Earth and Mars as seen from the Sun in the two latent neurons (see Figure 6b) — that is, it recovers the heliocentric model of the solar system.

2. **Uncorrelated**: The elements in the set $\{R_1, R_2, \ldots, R_{|R|}\}$ are mutually independent.

Property 1 asserts that the encoder map E encodes all information of the observation $o \in \mathcal{O}$ that is necessary to reply to all possible questions $q \in \mathcal{Q}$. We require the decoder to be smooth, since this allows us to give the number of parameters stored in the latent representation a well defined meaning in terms of a dimension (see Appendix C).

Property 2 means that knowing some variables in the latent representation does not provide any information about any other latent variables; note that this depends on the distribution of the observations.

We define a *minimal uncorrelated representation* R as an uncorrelated (sufficient) representation with a minimal number of parameters $|R|$. This formalizes what we consider to be a "simple" representation of physical data.

Without the assumption that the decoder is smooth, it would, in principle, always be sufficient to have a single latent variable, since a real number can store an infinite amount of information. Hence, methods from standard information theory, like the information bottleneck [29–31], are not the right tool to give the number of variables a formal meaning. In Appendix C, we use methods from differential geometry to show that the number of variables $|R|$ in a minimal (sufficient) representation corresponds to the number of relevant degrees of freedom in the observation data required to answer all possible questions.

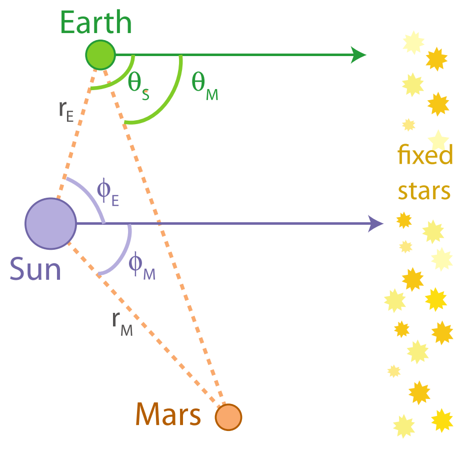
4 Discussion

4.1 Comparison with previous work

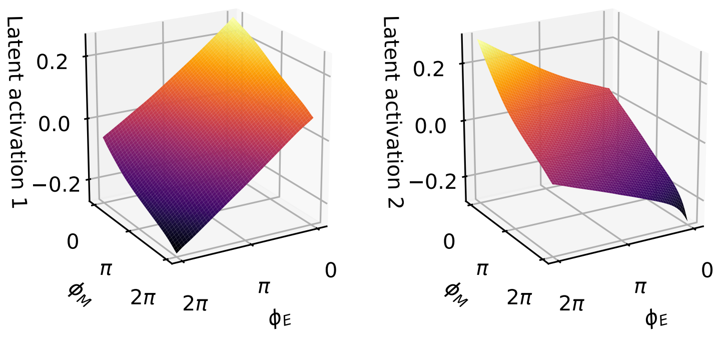
Neural networks have become a standard tool to tackle problems where we want to make predictions without following a particular algorithm or imposing structure on the available data (see for example [32–34]) and they have been applied to a wide variety of problems in physics. For example, in condensed matter physics and generally in many-body settings, neural networks have proven particularly useful to characterize phase transitions [4–9]. The aim of these works is to optimise the accuracy of the predictions, but not to extract information on what the network learned during training.

In quantum optics, automated search techniques and reinforcement-learning based schemes have been used to generate new experimental setups [35, 36]. Projective simulation [37] is used in [36] to autonomously discover experimental building blocks with maximum versatility.

Closer to our work, neural networks have also been used to efficiently represent wave functions of particular quantum systems [38–48]. In particular, in [39], variational autoencoders are used



(a)



(b)

Figure 6: **Heliocentric model of the solar system.** *SciNet* is given the angles of the Sun and Mars as seen from Earth at an initial time t_0 and has to predict these angles for later times. **(a) Physical setting.** The heliocentric angles ϕ_E and ϕ_M of the Earth and Mars are observed from the Sun; the angles θ_S and θ_M of the Sun and Mars are observed from Earth. All angles are measured relative to the fixed star background. **(b) Representation learned by *SciNet*.** The activation $r(t_0)$ of the two latent neurons at time t_0 (see Figure 5) is plotted as a function of the heliocentric angles ϕ_E and ϕ_M. The plots show that the network stores and evolves parameters that are linear combinations of the heliocentric angles, even though it was given the angles as observed from Earth. The slight non-linearity in the plots for extremal values of ϕ_M and ϕ_E is due to the sparsity of training data for these values.

to approximate the distribution of the measurement outcomes of a specific quantum state for a fixed measurement basis and the size of the neural network can provide an estimate for the complexity of the state. In contrast, our approach is not specifically designed for learning representations of quantum systems. Nevertheless, *SciNet* can be used to produce representations of arbitrary quantum states of simple systems without retraining. This allows us to extract information about the degrees of freedom required to represent any state of a (small) quantum system.

Another step towards extracting physical knowledge in an unsupervised way is presented in [18]. The authors show how the relevant degrees of freedom of a system in classical statistical mechanics can be extracted under the assumption that the input is drawn from a Boltzmann distribution. They make use of information theory to guide the unsupervised training of restricted Boltzmann machines, a class of probabilistic neural networks, to approximate probability distributions. While the focus in [18] is on systems for which the renormalization group procedure is applicable, here we presented a network architecture that is supposed to work, in principle, for arbitrary physical setups.

A different line of work has focused on using neural networks and other algorithmic techniques to better understand how humans are able to gain an intuitive understanding of physics [49–54]. For example, it has been shown that neural networks can build up some physical intuition: given a picture of a tower built with blocks, they can predict with good accuracy whether the tower is stable or will fall [55].

Very recently, physical variables were extracted in an unsupervised way from time series data of dynamical systems [56]. The network structure used in [56] is built on interaction networks [57–59] and it is well adapted to physical systems consisting of several objects interacting in a pair-wise manner. The prior knowledge included in the network structure allows the network to generalise to situations that differ substantially from those seen during training.

In the last few years, significant progress was made in extracting dynamical equations from experimental data [15, 16, 60–62], which is known to be an NP-hard problem [63]. In [15] (non-linear) dynamical models are inferred efficiently from experimental data. The possible models are ordered by mathematical simplicity, and the procedure consists in searching them according to this order, using methods from statistic inference. In [16], a different method is used, where the simplest formula describing the data is found by applying sparse regression techniques to some space of mathematical expressions. Instead of searching for dynamical models in the input data, in [60–62] neural networks are used to find a new set of variables such that the evolution of the new variables is approximately linear (motivated by Koopman operator theory). Our example given in Section 2.4 uses a similar network structure as the one used in [60–62], which corresponds to a special case of *SciNet* with a trivial question input and a latent representation that is evolved in time. The concept of evolving the system in the latent representation has also been used in machine learning to extract the relevant features from video data [64].

4.2 Future work

The interpretability of the latent variables remains challenging. In our examples, the interpretation of the latent representation was aided by comparing it to known representations in physics. However, in general we might be interested in settings without a hypothesized representation that the network's learned representation can be compared to. In that case, one could, for example, try to apply methods from symbolic regression to the trained encoder and decoder separately to obtain an analytic expression that might be more easily interpreted. Alternatively, one could try to develop more efficient techniques that explicitly use the structure of the neural network.

Another promising direction would be to consider methods from reinforcement learning for the investigation of physical data, e.g., along the lines proposed in [37].

4.3 Conclusion

In an overview of challenges for artificial intelligence in the near future [65], Lake *et al.* wrote:

> "For deep networks trained on physics-related data, it remains to be seen whether higher layers will encode objects, general physical properties, forces and approximately Newtonian dynamics."

In this work, we have shown that neural networks can be used to recover physical variables from experimental data. To do so, we have introduced a new network structure, *SciNet*, and employed techniques from unsupervised representation learning to encourage the network to find a minimal uncorrelated representation of experimental data.

The architecture of *SciNet* allows us to ask different questions about the physical system that the network has to answer using only its learned representation. Thus, the representation does not have to contain all information of the input data, but only the minimum amount of information that is necessary for the network to reply to all questions in some fixed set of questions.

In the case of our examples, the representations turned out to be the ones commonly used in physics textbooks. Our results therefore suggest that neural networks can indeed encode the relevant physical properties and provide a step towards solving the problem posed by Lake *et al.*

To summarize, the main aim of this work is to show that neural networks can be used to discover physical concepts without any prior knowledge. To achieve this goal, we introduced a neural network architecture that models the physical reasoning process. The examples illustrate that this architecture allows us to extract physically relevant data from experiments, without imposing further knowledge about physics or mathematics.

Source code and implementation details

The source code, as well as details of the network structure and training process (including pre-trained *SciNets*) are available at `https://github.com/eth-nn-physics/nn_physical_concepts`. The networks were implemented using the Tensorflow library [66]. For all examples, the training process only takes a few hours on a standard laptop.

Acknowledgements

We would like to thank Alessandro Achille, Serguei Beloussov, Thomas Frerix, Viktor Gal, Thomas Häner, Maciej Koch-Janusz, Aurelien Lucchi, Joseph M. Renes, Andrea Rocchetto, Ying Zhe Ernest Tan, Jinzhao Wang and Leonard Wossnig for helpful discussions. We acknowledge support from the Swiss National Science Foundation through SNSF project No. 200020_165843 and through the National Centre of Competence in Research *Quantum Science and Technology* (QSIT). LdR and RR furthermore acknowledge support from the FQXi grant *Physics of the observer*. TM acknowledges support from ETH Zürich and the ETH Foundation through the *Excellence Scholarship & Opportunity Programme*.

Appendix

A Neural networks

For a detailed introduction to artificial neural networks and deep learning, see for example [32]. Here we give a very short overview of the basics.

Single artificial neuron. The building blocks of neural networks are single neurons (Figure 7a). We can think of a neuron as a map that takes several real inputs $x_1, \dots, x_n$ and provides an output $\sigma(\sum_i w_i x_i + b)$, according to an *activation function* $\sigma : \mathbb{R} \to \mathbb{R}$, where the weights $w_i \in \mathbb{R}$ and the bias $b \in \mathbb{R}$ are tunable parameters. The output of the neuron is itself sometimes denoted by *activation*, and there are different possible choices for the activation function. For the implementation of the examples in this paper, we use the exponential linear unit (ELU) [67], depicted in Figure 7b. The ELU is defined for a parameter $\alpha > 0$ as

$$\sigma_{\mathrm{ELU}}(z) = \begin{cases} z & \text{for } z > 0\,, \\ \alpha\,(e^z - 1) & \text{for } z \leq 0\,. \end{cases}$$

Neural network. A (feed-forward) neural network is created by arranging neurons in layers and forwarding the outcomes of the neurons in the i-th layer to neurons in the $(i+1)$-th layer (see Figure 7c). The network as a whole can be viewed as a function $F : \mathbb{R}^n \to \mathbb{R}^m$ with $x_1, \dots, x_n$ corresponding to the activations of the neurons in the first layer (which is called *input layer*). The activations of the input layer form the input for the second layer, which is a *hidden layer* (since it is neither an input nor an output layer). In the case of a fully connected network, each neuron in the $(i+1)$-th layer receives the activations of all neurons in the i-th layer as input. The activations of the m neurons in the last layer, which is called *output layer*, are then interpreted as the output of the function F. It can be shown that neural networks are *universal*, in the sense that any continuous function can be approximated arbitrarily well by a feedforward network with just one hidden layer by using sufficiently many hidden neurons. For a mathematical statement of the result, see [68, 69]. A visualization is given in [32].

Training. The weights and the biases of the neural network are not tuned by hand; instead, they are optimized using training samples, i.e., known input-output-pairs $(x, F^\star(x))$ of the function $F^\star$ that we would like to approximate. We may think of a neural network as a class of functions $\{F_\theta\}_\theta$, parametrized by θ, which contains the weights and biases of all the neurons in the network. A cost function $C\,(x, \theta)$ measures how close the output $F_\theta(x)$ of the network is to the desired output $F^\star(x)$ for an input x. For example, a common choice for the cost function is $C\,(x, \theta) = \|F^\star(x) - F_\theta(x)\|_2^2$.

The weights and biases of a network are initialized at random [32]. To then update the parameters θ, the gradient $\vec{\nabla}_\theta C\,(x, \theta)$ is computed and averaged over all training samples x. Subsequently, θ is updated in the negative gradient direction — hence the name *gradient descent*. In practice, the average of the gradient over all training samples is often replaced by an average over a smaller subset of training samples called a *mini-batch*; then, the algorithm is called *s*tochastic gradient descent. The backpropagation algorithm is used to perform a gradient descent step efficiently (see [32] for details).

B Variational autoencoders

The implementation of *SciNet* uses a modified version of so-called variational autoencoders (VAEs) [19, 23]. The standard VAE architecture does not include the question input used by *SciNet* and tries to reconstruct the input from the representation instead of answering a question. VAEs are one particular architecture used in the field of representation learning [21]. Here, we give a short overview over the goals of representation learning and the details of VAEs.

Representation learning. The goal in representation learning is to map a high-dimensional input vector x to a lower-dimensional representation $z = (z_1, z_2, \dots, z_d)$, commonly called the *latent vector*.[4] The representation z should still contain all the relevant information about x. In the case of an autoencoder, z is used to reconstruct the input x. This is motivated by the idea that the better the (low-dimensional) representation is, the better the original data can be recovered from it. Specifically, an autoencoder uses a neural network (*encoder*) to map the input x to a small number

[4]The variables x and z correspond to the observation o and the representation r used in the main text.

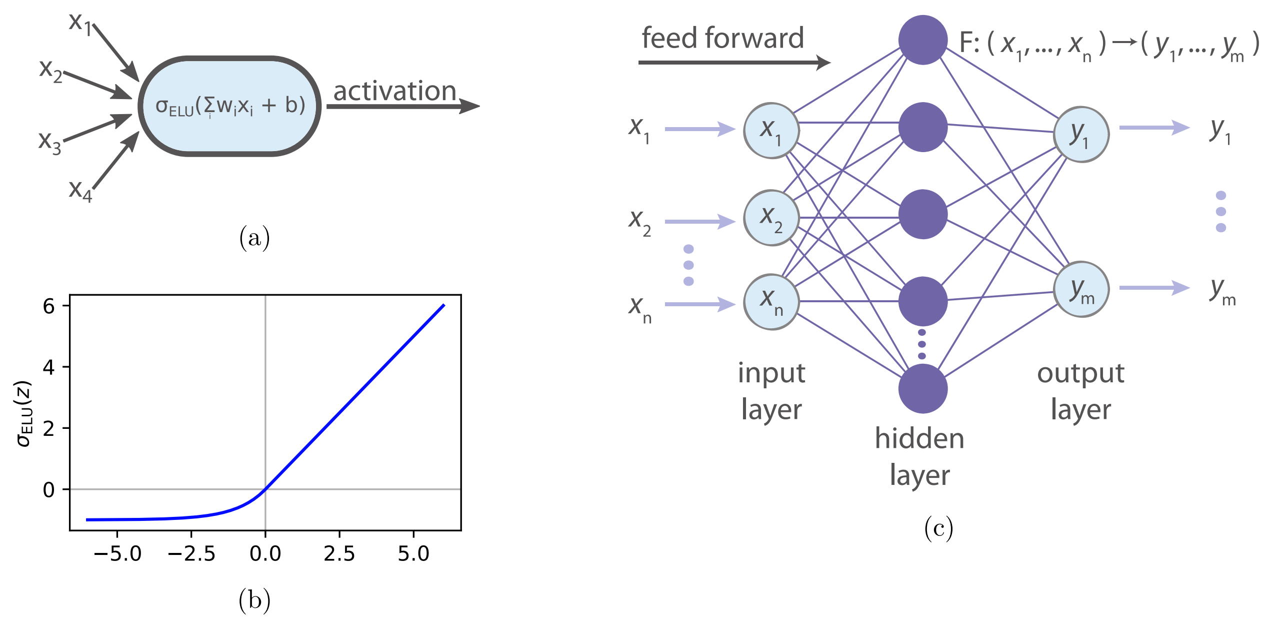


Figure 7: **Neural networks.** **(a)** Single artificial neuron with weights w_i, bias b and ELU activation function σ_{ELU}. The inputs to the neuron are denoted by $x_1, \ldots, x_4$. **(b)** ELU activation function for $\alpha = 1$. **(c)** Fully connected (feed-forward) neural network with 3 layers. The network as a whole can be thought of as a function mapping the inputs $(x_1, \ldots, x_n)$ to the output $(y_1, \ldots, y_m)$.

of latent neurons z. Then, another neural network (*decoder*) is used to reconstruct an estimate of the input, that is $z \mapsto \tilde{x}$. During training, the encoder and decoder are optimized to maximize the reconstruction accuracy and reach $\tilde{x} \approx x$.

Probabilistic encoder and decoder. Instead of considering deterministic maps $x \mapsto z$ and $z \mapsto \tilde{x}$, we generalize to conditional probability distributions $p(z|x)$ for the encoder and $p(\tilde{x}|z)$ for the decoder. This is motivated by the Bayesian view that the most informative statement the encoder can output a description of a probability distribution over all latent vectors, instead of outputting a single estimate. The same reasoning holds for the decoder. We use the notation $z \sim p(z)$ to indicate that z is picked at random according to the distribution p.

We cannot treat the general case analytically, so we make restricting assumptions to simplify the setting. First we assume that the input can be perfectly compressed and reconstructed by an encoder and decoder which are both neural networks, that is we assume that the ideal distributions $p(z|x)$ and $p(\tilde{x}|z)$ that reach $x = \tilde{x}$ are members of parametric families $\{p_\phi(z|x)\}_\phi$ and $\{p_\theta(\tilde{x}|z)\}_\theta$, respectively. We further assume that it is possible to achieve this with a latent representation where each neuron is independent of the others, $p_\phi(z|x) = \prod_i p_\phi(z_i|x)$. If these distributions turn out hard to find for a given dimension d of the latent representation, we can try to increase the number of neurons of the representation to disentangle them. Finally, we make one more simplifying assumption, which is justified *a posteriori* by good results: that we can reach a good approximation of $p(z|x)$ by using only independent normal distributions for each latent neuron, $p_\phi(z_i|x) = \mathcal{N}(\mu_i, \sigma_i)$, where μ_i is the mean and σ_i the variance. We can think of the encoder as mapping x to the vectors $\mu = (\mu_1, \ldots, \mu_d)$ and $\sigma = (\sigma_1, \ldots, \sigma_d)$.

The optimal settings for ϕ and θ are then learned as follows, see Figure 8:

1. The encoder with parameters (weights and biases) ϕ maps an input x to $p_\phi(z|x) = \mathcal{N}[(\mu_1, \ldots, \mu_d), (\sigma_1, \ldots, \sigma_d)]$.
2. A latent vector z is sampled from $p_\phi(z|x)$.
3. The decoder with parameters (weights and biases) θ maps the latent vector z to $p_\theta(\tilde{x}|z)$.
4. The parameters ϕ and θ are updated to maximize the likelihood of the original input x under the decoder distribution $p_\theta(\tilde{x}|z)$.

Reparameterization trick. The operation that samples a latent vector z from $p_\phi(z|x)$ is not

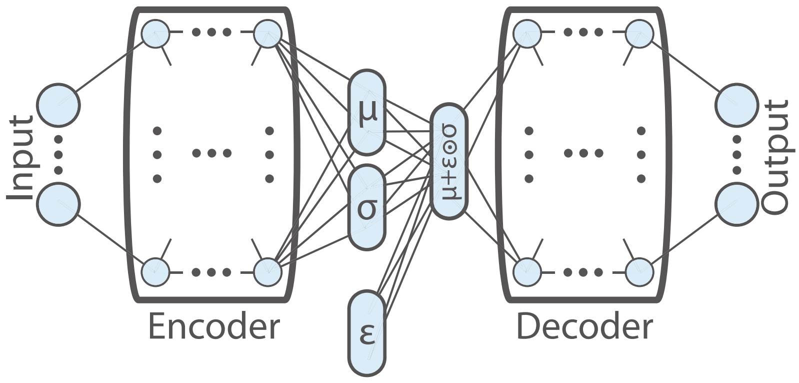


Figure 8: Network structure for a variational autoencoder. The encoder and decoder are described by conditional probability distributions $p(z|x)$ and $p(x|z)$ respectively. The output distribution of the encoder are the parameters μ_i and $\log(\sigma_i)$ for independent Gaussian distributions $z_i \sim \mathcal{N}(\mu_i, \sigma_i)$ of the latent variables. The reparameterization trick is used to sample from the latent distribution.

differentiable with respect to the parameters ϕ and θ of the network. However, differentiability is necessary to train the network using stochastic gradient descent. This issue is solved by the reparameterization trick introduced in [19]: if $p_\phi(z_i|x)$ is a Gaussian with mean μ_i and standard deviation σ_i, we can replace the sampling operation using an auxiliary random number $\varepsilon_i \sim \mathcal{N}(0,1)$. Then, a sample of the latent variable $z_i \sim \mathcal{N}(\mu_i, \sigma_i)$ can be generated by $z_i = \mu_i + \sigma_i \varepsilon_i$. Sampling ε_i does not interfere with the gradient descent because ε_i is independent of the trainable parameters ϕ and θ. Alternatively, one can view this way of sampling as injecting noise into the latent layer [24].

β-VAE cost function. A computationally tractable cost function for optimizing the parameters ϕ and θ was derived in [19]. This cost function was extended in [23] to encourage independency of the latent variables $z_1, \ldots, z_d$ (or to encourage "disentangled" representations, in the language of representation learning). The cost function in [23] is known as the β-VAE cost function,

$$C_\beta(x) = - \left[\mathbb{E}_{z\sim p_\phi(z|x)} \log p_\theta(x|z) \right] + \beta \, D_{\mathrm{KL}} \left[p_\phi(z|x) \| h(z) \right] ,$$

where the distribution $h(z)$ is a prior over the latent variables, typically chosen as the unit Gaussian[5], $\beta \geq 0$ is a constant, and D_{KL} is the Kullback-Leibler (KL) divergence, which is a quasi-distance[6] measure between probability distributions,

$$D_{\mathrm{KL}} \left[p(z) \| q(z) \right] = \sum_z p(z) \log \left(\frac{p(z)}{q(z)} \right) .$$

Let us give an intuition for the motivation behind the β-VAE cost function. The first term is a log-likelihood factor, which encourages the network to recover the input data with high accuracy. It asks "for each z , how likely are we to recover the original x after the decoding?" and takes the expectation of the logarithm of this likelihood $p_\theta(x|z)$ (other figures of merit could be used here in an alternative to the logarithm) over z sampled from $p_\phi(z|x)$, in order to simulate the encoding. In practice, this expectation is often estimated with a single sample, which works well enough if the mini-batches are chosen sufficiently large [19].

The second term encourages disentangled representations, and we can motivate it using standard properties of the KL divergence. Our goal is to minimize the amount of correlations between the latent variables z_i: we can do this by minimizing the distance $D_{\mathrm{KL}} [p(z) \| \prod_i p(z_i)]$ between $p(z)$ and the product of its marginals. For any other distribution with independent z_i, $h(z) = \prod_i h(z_i)$, the KL divergence satisfies

$$D_{\mathrm{KL}} \left[p(z) \| \prod_i p(z_i) \right] \leq D_{\mathrm{KL}} \left[p(z) \| h(z) \right] .$$

The KL divergence is furthermore jointly convex

[5]The interpretation of $h(z)$ as a prior is clear only when deriving VAEs as generative networks. For details, see [19].

[6]The KL divergence satisfies all axioms of a metric apart from symmetry.

in its arguments, which implies

$$D_{\mathrm{KL}}\left[\sum_x p(x)\ p_\theta(z|x)\|h(z)\right] \leq \sum_x p(x)\ D_{\mathrm{KL}}\left[p_\theta(z|x)\|h(z)\right].$$

Combining this with the previous inequality, we obtain

$$D_{\mathrm{KL}}\left[p(z)\ \|\prod_i p(z_i)\right] \leq \mathbb{E}_{x\sim p(x)}\ D_{\mathrm{KL}}\left[p(z|x)\|h(z)\right].$$

The term on the right hand side corresponds exactly to the second term in the cost function, since in the training we try to minimize $\mathbb{E}_{x\sim p(x)}\, C_\beta(x)$. Choosing a large parameter β also penalizes the size of latent representation z, motivating the network to learn an efficient representation. For an empirical test of the effect of large β see [23], and for another theoretical justification using the information bottleneck approach see [24].

To derive an explicit form of C_β for a simple case, we again assume that $p_\phi(z|x) = \mathcal{N}(\mu,\sigma)$. In addition, we assume that the decoder output $p_\theta(\tilde{x}|z)$ is a multivariate Gaussian with mean $\hat{x}$ and fixed covariance matrix $\hat{\sigma} = \frac{1}{\sqrt{2}}\mathbb{1}$. With these assumptions, the β-VAE cost function can be explicitly written as

$$C_\beta(x) = \|\hat{x}-x\|_2^2 - \frac{\beta}{2}\left(\sum_i \log(\sigma_i^2) - \mu_i^2 - \sigma_i^2\right) + C.$$

The constant terms C do not contribute to the gradients used for training and can therefore be ignored.

C Interpretation of the number of latent variables

In Section 3, we require that the latent representation should contain a minimal amount of latent variables; we now relate this number to the structure of the given data. Proposition 2 below asserts that the minimal number of latent neurons corresponds to the relevant degrees of freedom in the observed data required to answer all the questions that may be asked.

For simplicity, we describe the data with sets instead of random variables here. Note that the probabilistic structure was only used for Property 2 in Section 3, whereas here, we are only interested in the number of latent neurons and not in that they are mutually independent. We therefore consider the triple $(\mathcal{O}, \mathcal{Q}, a_{\mathrm{cor}})$, where $\mathcal{O}$ and $\mathcal{Q}$ are the sets containing the observation data and the questions respectively, and the function $a_{\mathrm{cor}} : (o,q) \mapsto a$ sends an observation $o \in \mathcal{O}$ and a question $q \in \mathcal{Q}$ to the correct reply $a \in \mathcal{A}$.

Intuitively, we say that the triple $(\mathcal{O}, \mathcal{Q}, a_{\mathrm{cor}})$ has dimension at least n if there exist questions in $\mathcal{Q}$ that are able to capture n degrees of freedom from the observation data $\mathcal{O}$. Smoothness of this "oracle" is a natural requirement, in the sense that we expect the dependence of the answers on the input to be robust under small perturbations. The formal definition follows.

Definition 1 (Dimension of a data set)**.** *Consider a data set described by the triple $(\mathcal{O}, \mathcal{Q}, a_{\mathrm{cor}})$, where $a_{\mathrm{cor}} : \mathcal{O}\times\mathcal{Q} \to \mathcal{A}$, and all sets are real, $\mathcal{O} \subseteq \mathbb{R}^r, \mathcal{Q} \subseteq \mathbb{R}^s, \mathcal{A} \subseteq \mathbb{R}^t$. We say that this triple has dimension at least n if there exists an n-dimensional submanifold $\mathcal{O}_n \subseteq \mathcal{O}$ and questions $q_1,\dots,q_k \in \mathcal{Q}$ and a function*

$$\begin{aligned} f:\quad &\mathcal{O}_n \to \mathcal{A}^k := \overbrace{A\times A\times\cdots\times A}^{k} \\ &o \mapsto [a_{\mathrm{cor}}(o,q_1),\dots,a_{\mathrm{cor}}(o,q_k)] \end{aligned}$$

such that $f : \mathcal{O}_n \to f(\mathcal{O}_n)$ is a diffeomorphism.

Proposition 2 (Minimal representation for *SciNet*)**.** *A (sufficient) latent representation for data described by a triple $(\mathcal{O}\subset\mathbb{R}^r, \mathcal{Q}\subset\mathbb{R}^s, a_{\mathrm{cor}} : \mathcal{O}\times\mathcal{Q}\to\mathcal{A}\subset\mathbb{R}^t)$ of dimension at least n requires at least n latent variables.*

Proof. By assumption, there is an n-dimensional submanifold $\mathcal{O}_n \subset \mathcal{O}$ and k questions $q_1,\dots,q_k$ such that $f : \mathcal{O}_n \to \mathcal{I}_n := f(\mathcal{O}_n)$ is a diffeomorphism. We prove the statement by contradiction: assume that there exists a (sufficient) representation described by an encoder $E : \mathcal{O} \to \mathcal{R}_m \subset \mathbb{R}^m$ with $m < n$ latent variables. By sufficiency of the representation, there exists a smooth decoder $D : \mathcal{R}_m\times\mathcal{Q} \to \mathcal{A}$ such that $D(E(o),q) = a_{\mathrm{cor}}(o,q)$ for all observations $o \in \mathcal{O}$ and questions $q \in \mathcal{Q}$. We define the smooth map

$$\begin{aligned} \tilde{D} : \mathcal{R}_m &\to \mathcal{A}^k \\ r &\mapsto [D(r,q_1),\dots,D(r,q_k)], \end{aligned}$$

and denote the pre-image of $\mathcal{I}_n$ by $\tilde{\mathcal{R}}_m := \tilde{D}^{-1}(\mathcal{I}_n)$.

By sufficiency of the representation, the restriction of the map $\tilde{D}$ to $\tilde{\mathcal{R}}_m$ denoted by $\tilde{D}|_{\tilde{\mathcal{R}}_m}$:

$\tilde{\mathcal{R}}_m \to \mathcal{I}_n$ is a smooth and surjective map. However, by Sard's theorem (see for example [70]), the image $\tilde{D}(\tilde{\mathcal{R}}_m)$ is of measure zero in $\mathcal{I}_n$, since the dimension of the domain $\tilde{\mathcal{R}}_m \subset \mathbb{R}^m$ is at most m, which is smaller than the dimension n of the image $\mathcal{I}_n$. This contradicts the surjectivity of $\tilde{D}|_{\tilde{\mathcal{R}}_m}$ and finishes the proof. □

We can consider an autoencoder as a special case of *SciNet*, where we ask always the same question and expect the network to reproduce the observation input. Hence, an autoencoder can be described by a triple $(\mathcal{O}, \mathcal{Q} = \{0\}, a_{\text{cor}} : (o, 0) \mapsto o)$. As a corollary of Proposition 2, we show that in the case of an autoencoder, the required number of latent variables corresponds to the "relevant" number of degrees of freedom that describe the observation input. The relevant degrees of freedom, which are called *(hidden) generative factors* in this context in representation learning (see for example [23]), may be described by the dimension of the domain of a smooth nondegenerate data generating function H, defined as follows.

Definition 3. *We say that a smooth function $H : \mathcal{G} \subset \mathbb{R}^d \to \mathbb{R}^r$ is nondegenerate if there exists an open subset $\mathcal{N}_d \subset \mathcal{G}$ such that the restriction $H|_{\mathcal{N}_d} : \mathcal{N}_d \to H(\mathcal{N}_d)$ of H on $\mathcal{N}_d$ is a diffeomorphism.*

One may think of H as sending a small dimensional representation of the data onto a manifold in a high dimensional space of observations.

Corollary 4 (Minimal representation for an autoencoder)**.** *Let $H : \mathcal{G} \subset \mathbb{R}^d \to \mathcal{O} \subset \mathbb{R}^r$ be a smooth, nondegenerate and surjective (data generating) function, and let us assume that $\mathcal{G}$ is bounded. Then the minimal sufficient representation for data described by a triple $(\mathcal{O}, \mathcal{Q} = \{0\}, a_{\text{cor}} : (o, 0) \mapsto o)$ contains d latent variables.*

Proof. First, we show the existence of a (sufficient) representation with d latent variables. We define the encoder mapping (and hence the representation) by $E : o \mapsto \text{argmin}[H^{-1}(\{o\})] \in \mathcal{G}$, where the minimum takes into account only the first vector entry.[7] We set the decoder equal to the smooth map H. By noting that $D(E(o), 0) = o$ for all $o \in \mathcal{O}$, this shows that d latent variables are sufficient.

[7]Note that any element in $H^{-1}(\{o\})$ could be chosen.

Let us now show that there cannot exist a representation with less than d variables. By definition of a nondegenerate function H, there exists an open subset $\mathcal{N}_d \subset \mathcal{G}$ in $\mathbb{R}^d$ such that $H|_{\mathcal{N}_d} : \mathcal{N}_d \to H(\mathcal{N}_d)$ is a diffeomorphism. We define the function $f : o \in H(\mathcal{N}_d) \mapsto a_{\text{cor}}(o, 0) \in \mathcal{I}$, where $\mathcal{I} = H(\mathcal{N}_d)$. Since f is the identity map and hence a diffeomorphism, the data described by the triple $(\mathcal{O}, \mathcal{Q} = \{0\}, a_{\text{cor}} : (o, 0) \mapsto o)$ has dimension at least d. By Proposition 2, we conclude that at least d latent variables are required. □

D Cyclic representations

Here we explain the difficulty of a neural network to learn representations of cyclic parameters, which was alluded to in the context of the qubit example (Section 2.3, see [71, 72] for a detailed discussion relevant to computer vision). In general, this problem occurs if the data $\mathcal{O}$ that we would like to represent forms a closed manifold (i.e., a compact manifold without boundary), such as a circle, a sphere or a Klein bottle. In that case, several coordinate charts are required to describe this manifold.

As an example, let us consider data points lying on the unit sphere $\mathcal{O} = \{(x, y, z) : x^2 + y^2 + z^2 = 1\}$, which we would like to encode into a simple representation. The data can be (globally) parameterized with spherical coordinates $\phi \in [0, 2\pi)$ and $\theta \in [0, \pi]$ where $(x, y, z) = f(\theta, \phi) := (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta)$.[8] We would like the encoder to perform the mapping f^{-1}, where we define $f^{-1}((0, 0, 1)) = (0, 0)$ and $f^{-1}((0, 0, -1)) = (\pi, 0)$ for convenience. This mapping is not continuous at points on the sphere with $\phi = 0$ for $\theta \in (0, \pi)$. Therefore, using a neural network as an encoder leads to problems, as neural networks, as introduced here, can only implement continuous functions. In practice, the network is forced to approximate the discontinuity in the encoder by a very steep continuous function, which leads to a high error for points close to the discontinuity.

In the qubit example, the same problem appears. To parameterize a qubit state ψ with two parameters, the Bloch sphere with parameters $\theta \in [0, \pi]$ and $\phi \in [0, 2\pi)$ is used: the state ψ can be written as $\psi(\theta, \phi) = (\cos(\theta/2), e^{i\phi}\sin(\theta/2))$ (see for example [73] for more details). Ideally, the

[8]The function f is not a chart, since it is not injective and its domain is not open.

encoder would perform the map $E : o(\psi(\theta, \phi)) := (|\langle \alpha_1, \psi(\theta, \phi)\rangle|^2, \dots, |\langle \alpha_{N_1}, \psi(\theta, \phi)\rangle|^2) \mapsto (\theta, \phi)$ for some fixed binary projective measurements $\alpha_i \in \mathbb{C}^2$. However, such an encoder is not continuous. Indeed, assuming that the encoder is continuous, leads to the following contradiction:

$$\begin{aligned}(\theta, 0) &= E(o(\psi(\theta, \phi = 0))) \\ &= E(o(\lim_{\phi\to 2\pi} \psi(\theta, \phi))) \\ &= \lim_{\phi\to 2\pi} E(o(\psi(\theta, \phi))) \\ &= \lim_{\phi\to 2\pi} (\theta, \phi) = (\theta, 2\pi)\,,\end{aligned}$$

where we have used the periodicity of ϕ in the second equality and the fact that the Bloch sphere representation and the scalar product (and hence $o(\psi(\theta, \phi))$) as well as the encoder (by assumption) are continuous in ϕ in the third equality.

References

[1] Y. Aharonov and D. Rohrlich, *Quantum Paradoxes*, (Wiley-VCH, Weinheim, Germany, 2008).

[2] G. Chiribella and R. W. Spekkens (editors), *Quantum Theory: Informational Foundations and Foils*, Fundamental Theories of Physics Vol. 181 (Springer, Dordrecht, 2016).

[3] D. Frauchiger and R. Renner, "Quantum theory cannot consistently describe the use of itself", Nature Commun. 9, 3711 (2018).

[4] J. Carrasquilla and R. G. Melko, "Machine learning phases of matter", Nature Phys. 13, 431 (2017).

[5] E. P. L. v. Nieuwenburg, Y.-H. Liu, and S. D. Huber, "Learning phase transitions by confusion", Nature Phys. 13, 435 (2017).

[6] E. P. L. v. Nieuwenburg, E. Bairey, and G. Refael, "Learning phase transitions from dynamics", *Preprint* (2017), arXiv: 1712.00450.

[7] P. Huembeli, A. Dauphin, P. Wittek, and C. Gogolin, "Automated discovery of characteristic features of phase transitions in many-body localization", *Preprint* (2018), arXiv: 1806.00419.

[8] G. Torlai and R. G. Melko, "Learning thermodynamics with Boltzmann machines", Phys. Rev. B 94, 165134 (2016).

[9] T. Ohtsuki and T. Ohtsuki, "Deep learning the quantum phase transitions in random electron systems: applications to three dimensions", J. Phys. Soc. Jpn. 86, 044708 (2017).

[10] V. Dunjko and H. J. Briegel, "Machine learning & artificial intelligence in the quantum domain: a review of recent progress", Reports on Prog. Phys. 81, 074001 (2018).

[11] J. P. Crutchfield and B. S. McNamara, "Equations of motion from a data series.", Complex Syst. 1, 417 (1987).

[12] M. Schmidt and H. Lipson, "Distilling free-form natural laws from experimental data", Science 324, 81 (2009).

[13] M. Schmidt *et al.*, "Automated refinement and inference of analytical models for metabolic networks", Phys. Biol. 8, 055011 (2011).

[14] C. Hillar and F. Sommer, "Comment on the article "Distilling free-form natural laws from experimental data"", *Preprint* (2012).

[15] B. C. Daniels and I. Nemenman, "Automated adaptive inference of phenomenological dynamical models", Nature Commun. 6, 8133 (2015).

[16] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems", Proc. Natl. Acad. Sciences 113, 3932 (2016).

[17] M. Guzdial, B. Li, and M. O. Riedl, "Game Engine Learning from Video", Proc. Twenty-Sixth Int. Jt. Conf. on Artif. Intell. (2017).

[18] M. Koch-Janusz and Z. Ringel, "Mutual information, neural networks and the renormalization group", Nature Phys. 14, 578 (2018).

[19] D. P. Kingma and M. Welling, "Auto-encoding variational bayes", *Preprint* (2013), arXiv: 1312.6114.

[20] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks", Science 313, 504 (2006).

[21] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: a review and new perspectives", *Preprint* (2012), arXiv: 1206.5538.

[22] Y. Bengio, "Deep learning of representations: looking forward", *Preprint* (2013), arXiv: 1305.0445.

[23] I. Higgins *et al.*, "beta-VAE: learning basic visual concepts with a constrained variational framework", ICLR (2017).

[24] A. Achille and S. Soatto, "Information dropout: learning optimal representations through noisy computation", IEEE Transactions on Pattern Analysis Mach. Intell. 8828, 1 (2018).

[25] S. M. A. Eslami *et al.*, "Neural scene representation and rendering", Science 360, 1204 (2018).

[26] H. Kim and A. Mnih, "Disentangling by factorising", *Preprint* (2018), arXiv: 1802.05983.

[27] C. P. Burgess *et al.*, "Understanding disentangling in beta-VAE", NIPS (2018).

[28] M. Paris and J. Reháček (editors), *Quantum State Estimation*, Lecture Notes in Physics (Springer, Berlin, Heidelberg, 2004).

[29] N. Tishby, F. C. Pereira, and W. Bialek, "The information bottleneck method", *Preprint* (2000), arXiv: 0004057.

[30] N. Tishby and N. Zaslavsky, "Deep learning and the information bottleneck principle", 2015 IEEE Inf. Theory Work. (ITW), 1 (2015).

[31] R. Shwartz-Ziv and N. Tishby, "Opening the black box of deep neural networks via information", *Preprint* (2017), arXiv: 1703.00810.

[32] M. A. Nielsen, Neural networks and deep learning, 2018, `http://neuralnetworksanddeeplearning.com/`.

[33] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning", Nature 521, 436 (2015).

[34] D. Silver *et al.*, "Mastering the game of Go with deep neural networks and tree search", Nature 529, 484 (2016).

[35] M. Krenn, M. Malik, R. Fickler, R. Lapkiewicz, and A. Zeilinger, "Automated search for new quantum experiments", Phys. Rev. Lett. 116, 090405 (2016).

[36] A. A. Melnikov *et al.*, "Active learning machine learns to create new quantum experiments", Proc. Natl. Acad. Sciences 115, 1221 (2018).

[37] H. J. Briegel and G. D. l. Cuevas, "Projective simulation for artificial intelligence", Sci. Reports 2, 400 (2012).

[38] G. Carleo and M. Troyer, "Solving the quantum many-body problem with artificial neural networks", Science 355, 602 (2017).

[39] A. Rocchetto, E. Grant, S. Strelchuk, G. Carleo, and S. Severini, "Learning hard quantum distributions with variational autoencoders", npj Quantum Inf. 4, 28 (2018).

[40] Z. Cai and J. Liu, "Approximating quantum many-body wave-functions using artificial neural networks", Phys. Rev. B 97 (2018).

[41] Y. Huang and J. E. Moore, "Neural network representation of tensor network and chiral states", *Preprint* (2017), arXiv: arXiv:1701.06246.

[42] D.-L. Deng, X. Li, and S. D. Sarma, "Machine learning topological states", Phys. Rev. B 96 (2017).

[43] M. Schmitt and M. Heyl, "Quantum dynamics in transverse-field Ising models from classical networks", SciPost Phys. 4 (2018).

[44] G. Torlai *et al.*, "Many-body quantum state tomography with neural networks", Nature Phys. 14, 447 (2018).

[45] Y. Nomura, A. S. Darmawan, Y. Yamaji, and M. Imada, "Restricted-Boltzmann-machine learning for solving strongly correlated quantum systems", Phys. Rev. B 96 (2017).

[46] D.-L. Deng, X. Li, and S. D. Sarma, "Quantum entanglement in neural network states", Phys. Rev. X 7 (2017).

[47] X. Gao and L.-M. Duan, "Efficient representation of quantum many-body states with deep neural networks", Nature Commun. 8 (2017).

[48] G. Torlai *et al.*, "Many-body quantum state tomography with neural networks", Nature Phys. 14, 447 (2018).

[49] J. Hamrick, P. Battaglia, and J. B. Tenenbaum, "Internal physics models guide probabilistic judgments about object dynamics", In Cogn. Science Soc., 1545 (2011).

[50] P. W. Battaglia, J. B. Hamrick, and J. B. Tenenbaum, "Simulation as an engine of physical scene understanding", Proc. Natl. Acad. Sciences 110, 18327 (2013).

[51] C. Bates, I. Yildirim, J. B Tenenbaum, and P. W Battaglia, "Humans predict liquid dynamics using probabilistic simulation", Proc. 37th Annu. Conf. Cogn. Science Soc. Pasadena, CA, 172 (2015).

[52] J. Wu, I. Yildirim, J. J. Lim, B. Freeman, and J. Tenenbaum, "Galileo: Perceiving physical object properties by integrating a physics engine with deep learning", Adv. Neural Inf. Process. Syst. 28, 127 (2015).

[53] T. D. Ullman, A. Stuhlmüller, N. D. Goodman, and J. B. Tenenbaum, "Learning physical parameters from dynamic scenes", Cogn. Psychol. 104, 57 (2018).

[54] N. R. Bramley, T. Gerstenberg, J. B. Tenenbaum, and T. M. Gureckis, "Intuitive experimentation in the physical world", Cogn. Psychol. 105, 9 (2018).

[55] A. Lerer, S. Gross, and R. Fergus, "Learning physical intuition of block towers by example", *Preprint* (2016), arXiv: 1603.01312.

[56] D. Zheng, V. Luo, J. Wu, and J. B. Tenenbaum, "Unsupervised learning of latent physical properties using perception-prediction networks", *Preprint* (2018), arXiv: arXiv:1807.09244.

[57] P. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, and K. Kavukcuoglu, "Interaction networks for learning about objects, relations and physics", Proc. 30th Int. Conf. on Neural Inf. Process. Syst., 4509 (2016).

[58] M. B. Chang, T. Ullman, A. Torralba, and J. B. Tenenbaum, "A compositional object-based approach to learning physical dynamics", *Preprint* (2016), arXiv: 1612.00341.

[59] D. Raposo *et al.*, "Discovering objects and their relations from entangled scene representations", *Preprint* (2017).

[60] B. Lusch, J. N. Kutz, and S. L. Brunton, "Deep learning for universal linear embeddings of nonlinear dynamics", *Preprint* (2017), arXiv: 1712.09707.

[61] N. Takeishi, Y. Kawahara, and T. Yairi, "Learning Koopman invariant subspaces for dynamic mode decomposition", *Preprint* (2017), arXiv: arXiv:1710.04340.

[62] S. E. Otto and C. W. Rowley, "Linearly-recurrent autoencoder networks for learning dynamics", *Preprint* (2017), arXiv: arXiv:1712.01378.

[63] T. S. Cubitt, J. Eisert, and M. M. Wolf, "Extracting dynamical equations from experimental data is NP hard", Phys. Rev. Lett. 108, 120503 (2012).

[64] M. Fraccaro, S. Kamronn, U. Paquet, and O. Winther, "A disentangled recognition and nonlinear dynamics model for unsupervised learning", Adv. Neural Inf. Process. Syst., 3601 (2017).

[65] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman, "Building machines that learn and think like people", Behav. Brain Sciences 40 (2017).

[66] M. Abadi *et al.*, TensorFlow: Large-scale machine learning on heterogeneous systems, 2015, `https://www.tensorflow.org/`.

[67] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)", *Preprint* (2015), arXiv: 1511.07289.

[68] G. Cybenko, "Approximation by superpositions of a sigmoidal function", Math. Control. Signals Syst. 2, 303 (1989).

[69] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators", Neural Networks 2, 359 (1989).

[70] J. Lee, *Introduction to Smooth Manifolds*, Graduate Texts in Mathematics, 2 ed. (Springer-Verlag, New York, 2012).

[71] N. Pitelis, C. Russell, and L. Agapito, "Learning a manifold as an atlas", IEEE Conf. on Comput. Vis. Pattern Recognit., 1642 (2013).

[72] E. O. Korman, "Autoencoding topology", *Preprint* (2018), arXiv: 1803.00156.

[73] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*, (Cambridge University Press, 2010).