

PAPER • OPEN ACCESS

## Operationally meaningful representations of physical systems in neural networks

To cite this article: Hendrik Poulsen Nautrup *et al* 2022 *Mach. Learn.: Sci. Technol.* **3** 045025

View the [article online](#) for updates and enhancements.

You may also like

- [Enhancing wildfire spread modelling by building a gridded fuel moisture content product with machine learning](#)  
Tyler C McCandless, Branko Kosovic and William Petzke
- [Quantum computation with machine-learning-controlled quantum stuff](#)  
Lucien Hardy and Adam G M Lewis
- [Comparison between SOLPS-4.3 and the Lengyel Model for ITER baseline neon-seeded plasmas](#)  
D. Moulton, P.C. Stangeby, X. Bonnin et al.



## PAPER

## OPEN ACCESS


RECEIVED  
12 July 2022REVISED  
11 October 2022ACCEPTED FOR PUBLICATION  
17 October 2022PUBLISHED  
16 December 2022

Original Content from  
this work may be used  
under the terms of the  
[Creative Commons  
Attribution 4.0 licence](#).

Any further distribution  
of this work must  
maintain attribution to  
the author(s) and the title  
of the work, journal  
citation and DOI.



# Operationally meaningful representations of physical systems in neural networks

Hendrik Poulsen Nautrup<sup>1,\*</sup> , Tony Metger<sup>2,\*</sup>, Raban Iten<sup>2</sup>, Sofiene Jerbi<sup>1</sup>, Lea M Trenkwalder<sup>1</sup>, Henrik Wilming<sup>2</sup>, Hans J Briegel<sup>1,3</sup> and Renato Renner<sup>2</sup>

<sup>1</sup> Institute for Theoretical Physics, University of Innsbruck, Technikerstr. 21a, A-6020 Innsbruck, Austria

<sup>2</sup> Institute for Theoretical Physics, ETH Zürich, 8093 Zürich, Switzerland

<sup>3</sup> Department of Philosophy, University of Konstanz, 78457 Konstanz, Germany

\* Authors to whom any correspondence should be addressed.

E-mail: [hendrik.poulsen-nautrup@uibk.ac.at](mailto:hendrik.poulsen-nautrup@uibk.ac.at) and [tmetger@ethz.ch](mailto:tmetger@ethz.ch)

**Keywords:** representation learning, neural networks, reinforcement learning, Bloch vector, quantum physics

## Abstract

To make progress in science, we often build abstract representations of physical systems that meaningfully encode information about the systems. Such representations ignore redundant features and treat parameters such as velocity and position separately because they can be useful for making statements about different experimental settings. Here, we capture this notion by formally defining the concept of operationally meaningful representations. We present an autoencoder architecture with attention mechanism that can generate such representations and demonstrate it on examples involving both classical and quantum physics. For instance, our architecture finds a compact representation of an arbitrary two-qubit system that separates local parameters from parameters describing quantum correlations.

## 1. Introduction

Neural networks are among the most versatile and successful tools in machine learning [1–3] and have been applied to a wide variety of problems in physics (see [4–6] for recent reviews). Many of the earlier applications have focused on solving specific problems that are intractable analytically and for which conventional numerical methods deliver unsatisfactory results. Conversely, neural networks may also lead to new insights into how the human brain develops physical intuition from observations [7–14].

Recently, the potential role that machine learning might play in the scientific discovery process has received increasing attention [15–22]. This direction of research is not only concerned with machine learning as a useful numerical tool for solving hard problems, but also seeks ways to establish artificial intelligence methodologies as general-purpose tools for scientific research.

An important step in the scientific process is to convert experimental data, which can be seen as a very high-dimensional and noisy representation of a physical system, into a more succinct representation that is amenable to a theoretical treatment by a human user. For example, when we observe the trajectory of an object, the natural experiment is to record the position of the object at different times; however, our theories of kinematics do not use time series of positions as variables, but rather describe the system using quantities, or *parameters*, such as initial velocity and initial position. The description of a system in terms of initial velocity and initial position is both succinct and *operationally meaningful* since the concept of velocity by itself can be used to perform prediction tasks in many different physical settings. If we instead described the system by e.g. the sum and difference of initial velocity and initial position (in some fixed units), this would still succinctly represent the same information, but it would not be operationally meaningful and mostly useless to a human interpreter. This is because the sum of initial velocity and initial position by itself is generally not a useful quantity for making predictions. This notion is motivated by the following general criterion for physical theories. We want a theory to describe a system in such a way that if different agents perform different experiments on a systems, they only need to know a subset of the parameters describing this system. We call this criterion *efficient communicability*: we imagine that one agent has a full description

of the system in terms of a set of parameters and wants to tell other agents about the aspects of the system relevant to their experiments by sending a subset of parameters. Then, the parameters describing the full system should be chosen in such a way as to minimize the required communication.

In this work, we formally introduce the concept of operationally meaningful representations and present a neural network architecture for the automated discovery of such representations. To this end, we define an operationally meaningful representation of a physical system as the minimal set of parameters that describes the full system such that each parameter is relevant for different experiments one can actually perform on the system. In line with this definition, we design a neural network architecture that can generate a meaningful representation through an attention mechanism. We demonstrate that this architecture can indeed identify parameters such as charge, mass, or even amplitudes of a quantum state in an operationally meaningful way from high-dimensional experimental data.

## 2. Operationally meaningful representations

The field of representation learning is concerned with feature detection in raw data. While, in principle, all deep neural network architectures learn some representation within their hidden layers, most work in representation learning is dedicated to defining and finding *good* representations [23]. A desirable feature of such representations is the interpretability of their parameters (stored in different neurons in a neural network). Standard autoencoders, for instance, are neural networks which compress data during the learning process. In the resulting representation, different parameters in the representation are often highly correlated and do not have a straightforward interpretation. A lot of work in representation learning has recently been devoted to separating or *disentangling* such representations in a meaningful way (see e.g. [24–28] and appendix A).

When using neural networks to find such parameterisations, one encounters the limitation of standard techniques from representation learning: typically, statistically independent factors of variation in the training data set are disentangled. That is, disentanglement arises implicitly from the statistical distribution of the data set. This works well for many practical problems [24]; however, for scientific applications, it is desirable that parameters be separated according to a more fundamental and transparent criterion than the distribution of experimental data.

To this end, we introduce *operationally meaningful* representations. In such a representation, parameters that are useful for different operational tasks on a physical system should be stored in separate neurons. As an example, take the physical system to be a charged mass. We consider a situation where one agent  $E$  has performed a *data collection experiment* so that this agent has a high-dimensional representation of all potentially relevant information about this system, e.g. its mass, charge, and colour. Other agents want to predict the outcome of various *evaluation experiments* on this charged mass (their ‘operational task’), e.g. colliding it with another (uncharged) mass or placing it in an external electric field. For this, these agents will receive information about the system from agent  $E$ . The key restriction is that communication from agent  $E$  to the other agents should be minimized, i.e. agent  $E$  will try to find a low-dimensional representation of the system and only send the relevant parameters in this representation to each of the other agents. For example, an agent  $D_1$  who performs a collision experiment between the system and an uncharged particle with fixed mass only needs to know about the system’s mass. The requirement that communication be minimised then means that agent  $E$  has to store the mass as a separate parameter so that it is possible to communicate only the mass and no other information about the system to agent  $D_1$ . Another agent  $D_2$  might want to predict the force exerted on the system in an electric field, which requires knowledge of the system’s charge, forcing agent  $E$  to store the charge in another separate neuron for efficient communicability.

More formally, consider experiments that are performed on a physical system which can be described by some hidden parameter space  $\Omega$  of objects and their interactions. We generally do not have direct access to these hidden parameters of the physical system, but only to high-dimensional observational data  $\mathcal{O}$  generated by performing data collection experiments  $\mathcal{E}_d : \Omega \rightarrow \mathcal{O}$ . Our goal is to extract an operationally meaningful representation from such observational data. To evaluate whether a representation is operationally meaningful, we consider multiple *evaluation experiments*  $\mathcal{E}_i : \Omega \times \mathcal{Q}_i \rightarrow \mathcal{A}_i$ , which depend on the systems parameters in  $\Omega$  as well as additional variables or *questions*  $q \in \mathcal{Q}_i$  and produce an outcome or *answer*  $a \in \mathcal{A}_i$ . Questions are known reference parameters that are required to make a correct prediction. For example, in the collision experiment we considered above, the additional question variable might be (some encoding of) the mass of the reference particle that we collide with our system, and the answer might be a prediction of the system’s position one second after the collision. In this example, the evaluation experiment only requires a subset of the system’s parameters as input, namely the mass.

In general, a *representation*  $E$  of a physical system is a map  $E : \mathcal{O} \rightarrow \mathbb{R}^L$  that represents the observations by variables in a real space of dimension  $L$ . Note that in the context of this paper, we refer both the agent and the

representation to the same mapping  $E$ . We denote the dimension of a representation as  $\dim(E) = L$ , where we use  $\dim(E)$  as shorthand for  $\dim(\text{Im}(E))$ . A *sub-representation*  $E' : \mathcal{O} \rightarrow \mathbb{R}^{L'}$  (for  $L' \leq L$ ) is a map that coincides with  $E$  on  $L'$  of the output dimensions of  $E$ . That is, one can think of  $E'$  as a ‘filtered’ map that first applies  $E$ , but then only outputs  $L'$  of the  $L$  output dimensions. With the above setup of data collection and evaluation experiments, we can define an operationally meaningful representation as follows.

**Definition 1 (Operationally meaningful representation).** A representation  $E : \mathcal{O} \rightarrow \mathbb{R}^L$  with an image set  $E[\mathcal{O}] = \mathcal{R} \subset \mathbb{R}^L$  is called *operationally meaningful* w.r.t. evaluation experiments  $\{\mathcal{E}_i\}_{i=1,\dots,Q}$  if and only if

- $E$  is *sufficient* w.r.t.  $\{\mathcal{E}_i\}_{i=1,\dots,Q}$ : There exists a map  $D_i : \mathcal{R} \times \mathcal{Q}_i \rightarrow \mathcal{A}_i$  such that  $D_i(E(\mathcal{E}_d(\omega)), q) \approx \mathcal{E}_i(\omega, q)$  for all  $q \in \mathcal{Q}_i$ ,  $\omega \in \Omega$  and  $i = 1, \dots, Q$ .
- $E$  is *minimal* w.r.t.  $\{\mathcal{E}_i\}_{i=1,\dots,Q}$ : Any other representation  $E' : \mathcal{O} \rightarrow \mathbb{R}^{L'}$  that is sufficient w.r.t.  $\{\mathcal{E}_i\}_{i=1,\dots,Q}$  must have  $\dim(E') \geq \dim(E)$ .
- $E$  is *efficiently communicable* w.r.t.  $\{\mathcal{E}_i\}_{i=1,\dots,Q}$ :  $\forall i = 1, \dots, Q$ , there exist sub-representations  $E_i$  that are sufficient with respect to  $\mathcal{E}_i$  and  $\sum_{i=1}^Q \dim(E_i)$  is minimal, i.e. for any other representation  $E'$  with sufficient sub-representations  $E'_i$ ,  $\sum \dim(E'_i) \geq \sum \dim(E_i)$ .

Consider the example of the colliding charged particles above. A *sufficient* and *minimal* representation would store information about the mass and charge, albeit not necessarily in separate neurons. In an *efficiently communicable* representation however, the parameters mass and charge need to be stored in separate neurons since there are evaluation experiments that only require one of these parameters but not the other. Here, the existence of sub-representations implies that the representation  $E$  can be split into parts  $E_{\text{Im}(E_i)}$  that can be communicated to the respective agent  $D_i$ . Parameters which are not relevant to any of the evaluation experiments, e.g. the color of particles, will not be stored in the representation at all.

### 3. Architecture

We can construct a neural network architecture that autonomously generates operationally meaningful representations as defined above. The architecture is based on an autoencoder modified with an attention mechanism to enable the generation of efficiently communicable representations.

The neural network architecture is presented in figure 1. An *encoder*  $E : \mathcal{O} \rightarrow \mathbb{R}^L$  receives the high-dimensional data  $o \in \mathcal{O}$  of the physical system from the data collection experiment  $\mathcal{E}_d$  and maps it onto a latent space of some specified dimension  $L$ . In order to minimise the dimensionality of the representation, we add a global *filter*  $\varphi_E : \mathbb{R}^L \rightarrow \mathbb{R}^l$  that outputs only  $l \leq L$  of its  $L$  inputs, where  $l$  now is a parameter that is optimized during the training of the neural network<sup>4</sup>. For each evaluation experiment  $\mathcal{E}_i$  we add another filter  $\varphi_i : \mathbb{R}^l \rightarrow \mathbb{R}^{l_i}$  with  $l_i \leq l$  and a *decoder*  $D_i : \mathcal{R}_i \times \mathcal{Q}_i \rightarrow \mathcal{A}_i$  with  $\mathcal{R}_i \subset \mathbb{R}^{l_i}$  (see figure 1)

We can now define separate loss functions for our criteria of *sufficiency*, *minimality* and *efficient communicability*; the total loss function will be a weighted sum of these.

- *Sufficiency*:  $\mathcal{L}_s = \sum_{i=1}^Q (a_i(o, q_i) - a_i^*(o, q_i))^2$ ,
- *Minimality*:  $\mathcal{L}_m = \dim(\varphi_E)$ ,
- *Efficient communicability*:  $\mathcal{L}_c = \sum_{i=1}^Q \dim(\varphi_i)$ .

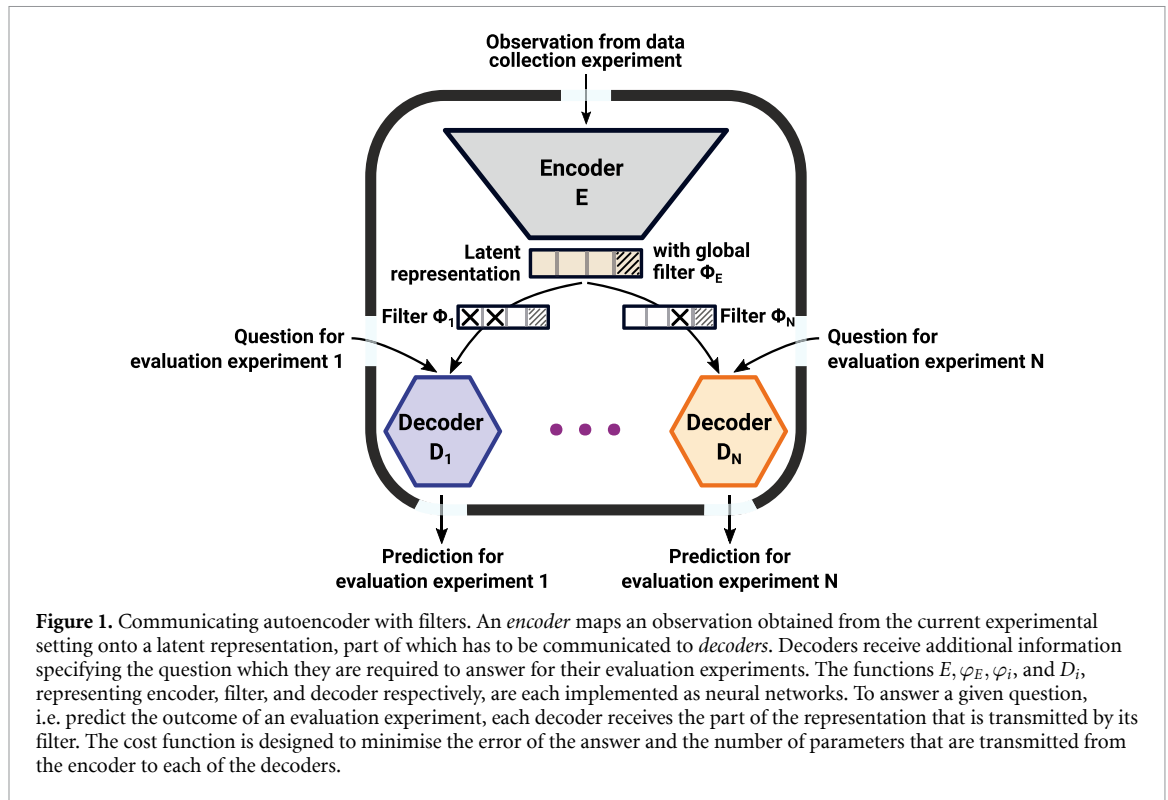
Here,  $a_i^*(o, q_i)$  is the outcome of the evaluation experiment for the question  $q_i$  of decoder  $D_i$  given the physical system that generated the observations  $o$ . Similarly,  $a_i$  is the corresponding prediction made by the decoder  $D_i$ .<sup>5</sup>

Due to the difficulty of implementing a function with a binary output in neural networks, we need to replace the ideal losses  $\mathcal{L}_m$  and  $\mathcal{L}_c$  by a smoothed filter function. Such a smoothed filter specifies how much noise should be added to a latent neuron; little noise means that the neuron is transmitted through the filter, and lots of noise means that the neurons is blocked, as in that case the filter’s output will contain essentially no information about the input neuron’s value. To sample the noise, we use the renormalisation trick [29], which allows gradients to propagate through the sampling step. More details about the filters are provided in appendix B. An extension of our architecture to multiple encoders can be found in appendix C.

In general, multiple rounds of hyperparameter optimization are necessary in order to find the operationally meaningful representation. The best current candidate for such a representation is easily

<sup>4</sup> Of course, we could equivalently just consider the output dimension  $L$  of  $E$  a trainable parameter that should be minimised. However, to implement such an optimization in a neural network, it is easier to consider a separate filter function, which can then be ‘smoothed’ as explained at the end of this section.

<sup>5</sup> For  $a_i^*(o, q_i)$  to be well-defined, we implicitly assume that the mapping from the hidden parameters  $\Omega$  to the observations is injective.



identified as the one that separates most latent variables while minimizing the evaluation loss. The number of latent variables and their separation can be directly inferred from the filter values for the respective neurons. These numbers can be used to design an automated search of parameters including the relative weights for the losses  $\mathcal{L}_s, \mathcal{L}_m, \mathcal{L}_c$ . In this way, we can guide the automated hyperparameter optimization to improve the efficiency of our method while maintaining its representational power.

### 4. Examples

We demonstrate our method on two examples, one from classical mechanics, and one from quantum mechanics. In appendix D we discuss and demonstrate how our architecture can be generalised to reinforcement learning environments [30] as evaluation experiments. In all cases, the network finds representations that comply with our operational requirements.

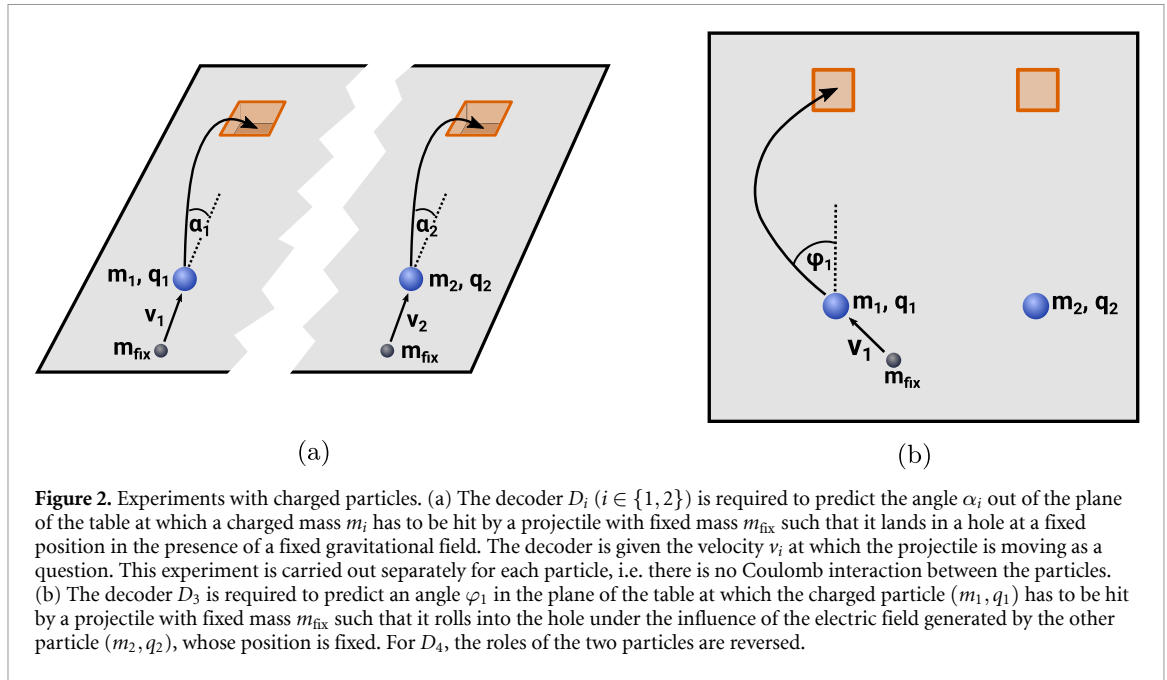
#### 4.1. Experiments with charged particles

Here, we present an illustrative example from classical mechanics where our architecture generates an operationally meaningful representation of charged particles.

Consider particles with masses  $m_1, m_2$  and charges  $q_1, q_2$ , where both masses and charges are the hidden parameters that are varied between training examples. We perform a data collection experiment  $\mathcal{E}_d : (m_1, m_2, q_1, q_2) \mapsto (x_{c,1}, x_{c,2}, x_{em,1}, x_{em,2})$  as follows:

- (a) To generate  $x_{c,i}$ , we elastically collide particle  $i$  (with mass  $m_i$ ), initially at rest, with a reference mass  $m_{ref}$  moving at a fixed reference velocity  $v_{ref}$ , and observe a time series of positions  $x_{c,i} = (x_{c,i}^1, \dots, x_{c,i}^n)$  of the particle after the collision. In practice, we use  $n = 10$ .
- (b) To generate  $x_{em,i}$ , we place particle  $i$  (with mass  $m_i$  and charge  $q_i$ ) at the origin at rest, and place a reference particle with fixed mass  $m_{ref}$  and charge  $q_{ref}$  at a fixed distance  $d_0$ . Particles  $i = 1, 2$  are free to move while the reference mass remains fixed. We observe a time series of positions  $x_{em,i} = (x_{em,i}^1, \dots, x_{em,i}^n)$  of particle  $i$  as it moves due to the Coulomb interaction between itself and the reference particle.

Different decoders now are required to predict the outcome of different evaluation experiments, pictured in figure 2.



- Decoders  $D_1$  and  $D_2$  are each given projectiles with a fixed mass  $m_{\text{fix}}$ . As question input,  $D_i$  is given the (variable) velocity  $v_i$  with which this projectile will hit  $m_i$ . The projectile hits  $m_i$  at an angle  $\alpha_i$  in the  $yz$ -plane, and the mass will fly towards the target hole under the influence of gravity. The decoder is asked to predict the angle  $\alpha_i$  for which  $m_i$  lands directly in the hole, similar to a golfer hitting a perfect lob shot that lands in the hole without bouncing.
- Decoders  $D_3$  and  $D_4$  are given projectiles. The velocities of these projectiles are again given as a question input. The decoder  $D_3$  has to predict the angle  $\varphi_1$  in the  $xy$ -plane so that when  $m_1$  is hit with the projectile at this angle and moves in the Coulomb field of  $m_2$  (which stays fixed), it will roll into the hole. For  $D_4$ , the roles of  $m_1$  and  $m_2$  are reversed.

In both cases, we restrict the velocities given as questions to ones where there actually exists a (unique) angle that makes the particle land in the hole.

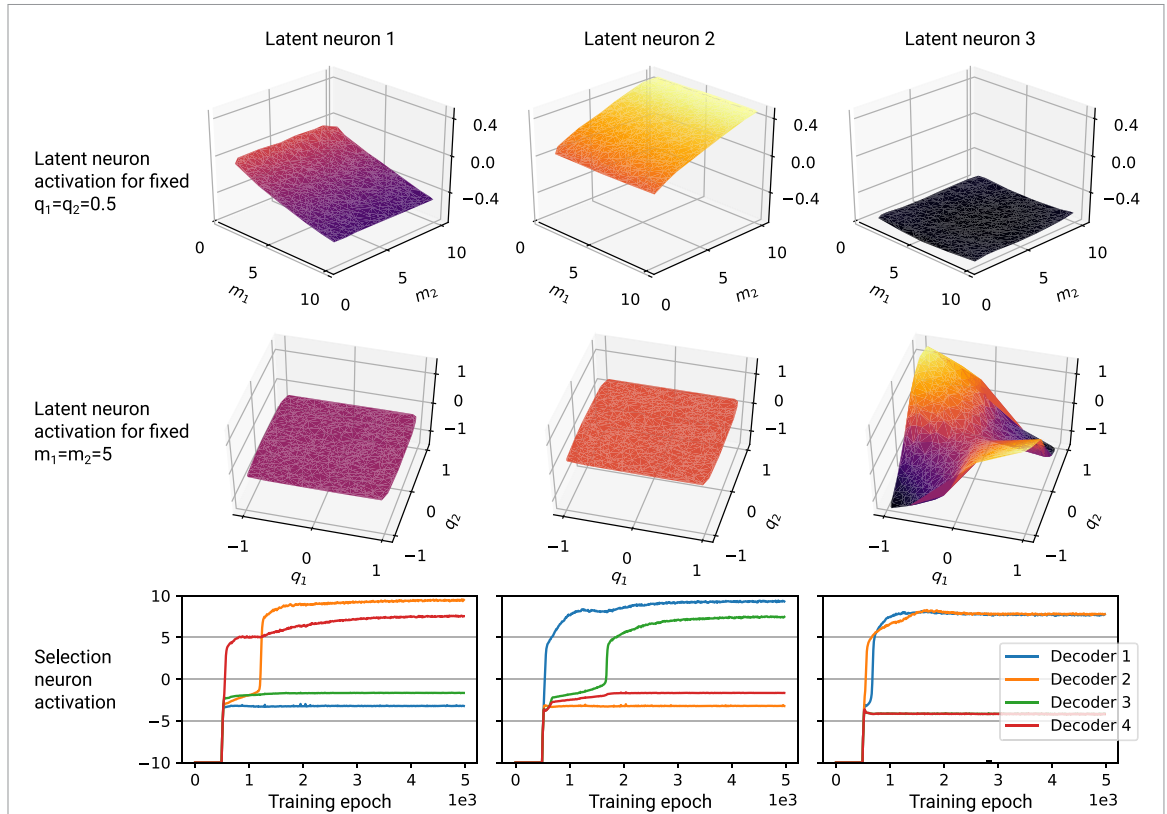
To analyse the learnt representation, we plot the activation of the latent neurons for different examples with different (known) values of  $m_1, m_2, q_1, q_2$  against those known values. This corresponds to comparing the learnt representation to a hypothesised representation that we might already have. The plots are shown in figure 3. The first and second latent neurons are linear in  $m_1$  and  $m_2$ , respectively, and independent of the charges; the third latent neuron has an activation that resembles the function  $q_1 \cdot q_2$  and is independent of the masses. This means that the first and second latent neurons store the masses individually, as would be expected since the evaluation experiments in figure 2(a) only require individual masses and no charges. The third neuron roughly stores the product of the charges, i.e. the quantity relevant for the strength of the Coulomb interaction between the charges. This is used by the decoders dealing with the evaluation experiments in figure 2(b), where the particle's trajectory depends on the Coulomb interaction with the other particle. As demonstrated in appendix C.2, a setting with multiple encoders can lead to an additional separation of the two charges.

#### 4.2. Quantum state tomography experiments

Here, we demonstrate that our architecture generates an operationally meaningful representation of a two-qubit, i.e. a four dimensional, quantum system. Finding a representation of such a system from measurement data is a non-trivial task called quantum state tomography [31]. We consider evaluation experiments that collect the outcomes of many different local and nonlocal measurements, respectively. We find that our architecture generates a representation that autonomously separates local, single-qubit parameters from nonlocal parameters that represent quantum correlations.

In this example, the encoder  $E$  has access to a data collection experiment consisting of two devices, where the first device creates (many copies of) a quantum system in a state  $\rho$ , which depends on the (hidden) parameters of the device. The second device can perform binary measurements (with output 0 or 1),





**Figure 3.** Results for the charged collision experiments. The used network has 3 latent neurons and each column of plots corresponds to one latent neuron. For the first row we generated input data with fixed charges  $q_1 = q_2 = 0.5$  and variable masses  $m_1, m_2$  in order to plot the activation of latent neurons as a function of the masses. We observe that latent neuron 1 and 2 store the masses  $m_1, m_2$  respectively while latent neuron 3 remains constant. In the second row, we plot the neurons' activation in response to  $q_1, q_2$  with fixed masses  $m_1, m_2 = 5$ . Here, the third latent neuron approximately stores  $q_1 \cdot q_2$ , which is the relevant quantity for the Coulomb interaction while the other neurons are independent of the charges. The third row shows which decoder receives information from the respective latent neuron. Roughly, the  $y$ -axis quantifies how much information of the latent neuron is transmitted by the 4 filters to the associated decoder as a function of the training epoch. Positive values mean that the filter does not transmit any information. Decoders 1 and 2 perform non-interacting collision experiments with objects  $(m_1, q_1)$  and  $(m_2, q_2)$ , respectively. Decoders 3 and 4 perform the corresponding electromagnetic collision experiments. As expected, we observe that the information about  $m_1$  (latent neuron 1) is received by decoders 1 and 3 and the information about  $m_2$  (latent neuron 2) is used by decoders 2 and 4. Since decoders 3 and 4 answer questions about electromagnetism experiments, the product of charges (latent neuron 3) is received only by them (the green line of decoder 3 in the last plot is hidden below the red one).

described by projections  $|\psi\rangle\langle\psi|$ , where  $|\psi\rangle$  is a pure state of two qubits<sup>6</sup>. For all data collection experiments, we fix 75 randomly chosen observables  $|\psi_1\rangle\langle\psi_1|, \dots, |\psi_{75}\rangle\langle\psi_{75}|$ . For a given state  $\rho$ , the input to the encoder  $E$  then consists of the probabilities to measure each of the fixed 75 observables, respectively. The state  $\rho$  is varied between training examples.

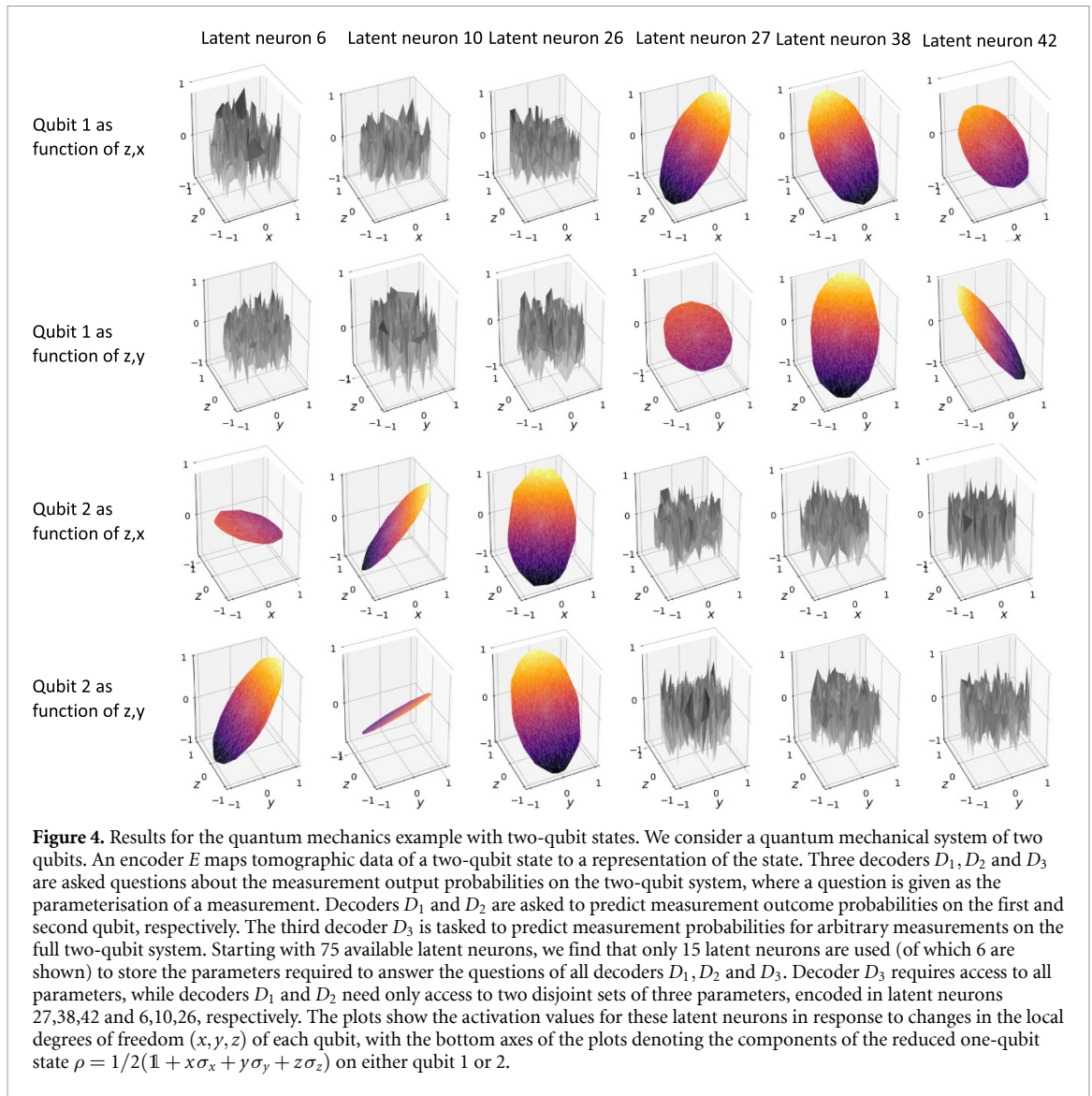
Three decoders  $D_1, D_2$  and  $D_3$  are now required to predict different evaluation experiments with the two-qubit system:

- Decoders  $D_1$  and  $D_2$  each have to predict measurement probabilities on the first and second qubit, respectively, given a parameterisation of a single-qubit measurement device.
- Decoder  $D_3$  is provided has to predict joint measurement output probabilities on both qubits, given a parameterisation of a two-qubit measurement device.

A measurement device measuring the projector  $|\omega\rangle\langle\omega|$  is parametrised again by 75 randomly chosen fixed projectors  $|\varphi_1\rangle\langle\varphi_1|, \dots, |\varphi_{75}\rangle\langle\varphi_{75}|$ , i.e. the  $i$ -th question input corresponds to the probabilities  $p(\omega, \varphi_i) := |\langle\varphi_i|\omega\rangle|^2$  for all  $i \in \{1, \dots, 75\}$ .

We find that three latent neurons are used for each of the local qubit representations as required by decoders  $D_1$  and  $D_2$ . These local representations store combinations of the  $x$ -,  $y$ - and  $z$ -component of the Bloch sphere representation  $\rho_i = 1/2(\mathbb{1} + x\sigma_x + y\sigma_y + z\sigma_z)$  of the single-qubit reduced states  $\rho_1, \rho_2$

<sup>6</sup> The probability to get outcome 1 for a measurement  $|\psi\rangle\langle\psi|$  is given by the Born rule  $p(\rho, \psi) := \langle\psi|\rho|\psi\rangle$ .



(see figure 4), where  $\sigma_x, \sigma_y, \sigma_z$  denote the Pauli matrices. In general, a two-qubit mixed state  $\rho$  is described by 15 parameters, since a Hermitian  $4 \times 4$  matrix is described by 16 parameters, and one parameter is determined by the others due to the unit trace condition. Indeed, we find that the decoder  $D_3$  who has to predict the outcomes of the joint measurements accesses 15 latent neurons, including the ones storing the two local representations. Having chosen a network structure with 75 latent neurons (corresponding to the dimension of the input to the encoder), the global filter successfully recognizes 60 superfluous latent neurons. These numbers correspond to the numbers found in the analytical approach in [32].

It is worth noting that learning arbitrary quantum state representations from measurement data is generally hard [33] and we do not expect our method to be scalable for the general task of quantum state reconstruction. Instead, our example gives an interesting new perspective on it: if we are only interested in a subsystem representation, we may require much less measurement data than for full-state tomography while our architecture would still minimize the number of parameters needed to fully represent the subsystem. In this way, we could avoid full-state tomography in favour of generating an operationally meaningful representation of a subsystem.

### 5. Conclusion

Deep neural networks, while performing very well on a variety of tasks, often lack interpretability [34]. Therefore, representation learning, and in particular methods for learning disentangled, interpretable representations, have recently received increased attention [17, 24, 27, 28, 35, 36]. However, while



methods of disentangling representations are widespread and well established, they lack an operational meaning.

In the scientific discovery process in particular, representations of physical systems and their operational meaning play a central role. To this end, we have developed a neural network architecture that can generate operationally meaningful representations with respect to experimental settings. Roughly, we call a representation operationally meaningful if it can be shared efficiently to predict different experiments. We have demonstrated our methods for examples in classical and quantum mechanics. Moreover, in appendix D, we also consider cases where the experimental process may be framed as an interactive reinforcement learning scenario [16]. Our architecture also works in such a setting and generates representations which are physically meaningful and relatively easy to interpret. Deep learning methods have already been applied across all scientific disciplines and quantum information in particular [37–39]. Our method may find application here to facilitate the interpretability of these tools to a human user.

In this work, we have interpreted the learnt representation by comparing it to some known or hypothesised representation. Instead, we could also seek to automate this process by employing unsupervised learning techniques that categorise experimental data by a metric defined by the response of different latent neurons. For the examples that we considered here, the learnt representation is small and simple enough to be interpretable by hand. However, for more complex problems, additional methods for making the representation more interpretable may be required. For example, instead of using a single layer of latent neurons to store the parameters, a recent work has shown the potential of semantically constrained graphs for this task [40]. In a complementary approach one could even generate a translation model for the communication channels between agents [41]. The resulting translation of a latent space could then help to interpret generated representations. We expect that these methods can be integrated into our architecture, which may allow to produce interpretable and meaningful representations even for highly complex latent spaces.

### Data availability statement

The data that support the findings of this study are openly available at <https://doi.org/10.5281/zenodo.7113611> (for the classical and quantum examples) and <https://doi.org/10.5281/zenodo.7113627> (for the reinforcement learning example). The networks were implemented using the Tensorflow [42] and PyTorch [43] library, respectively.

### Acknowledgments

H P N, S J, L M T and H J B acknowledge support from the Austrian Science Fund (FWF) through the DK-ALM: W1259-N27 and SFB BeyondC F7102. R I, H W and R R acknowledge support from from the Swiss National Science Foundation through SNSF Project No. 200020\_165843 and 200021\_188541. T M acknowledges support from ETH Zürich and the ETH Foundation through the *Excellence Scholarship & Opportunity Programme*, and from the IQIM, an NSF Physics Frontiers Center (NSF Grant PHY-1125565) with support of the Gordon and Betty Moore Foundation (GBMF-12500028). S J also acknowledges the Austrian Academy of Sciences as a recipient of the DOC Fellowship. H J B acknowledges support by the Ministerium für Wissenschaft, Forschung, und Kunst Baden Württemberg (AZ:33-7533.-30-10/41/1) and by the European Research Council (ERC) under Project No. 101055129. This work was supported by the Swiss National Supercomputing Centre (CSCS) under Project ID da04.

### Author contributions

H P N, T M, and R I contributed equally to the initial development of the project, performed the numerical work, and composed the manuscript. S J and L M T contributed to the theoretical and numerical development of the reinforcement learning part. H J B and R R initialised and supervised the project. All authors have discussed the results and contributed to the conceptual development of the project.

### Appendix A. Related work

The field of representation learning is concerned with feature detection in raw data. While, in principle, all deep neural network architectures learn some representation within their hidden layers, most work in representation learning is dedicated to defining and finding *good* representations [23]. A desirable feature of such representations is the interpretability of their parameters (stored in different neurons in a neural

network). Standard autoencoders, for instance, are neural networks which compress data during the learning process. In the resulting representation, different parameters in the representation are often highly correlated and do not have a straightforward interpretation. A lot of work in representation learning has recently been devoted to *disentangling* such representations in a meaningful way (see e.g. [24–28]). In particular, these works introduce criteria, also referred to as *priors* in representation learning, by which we can disentangle representations.

In natural language processing, specifically machine translation and summarisation, attention mechanisms and multi-agent communication are used to efficiently predict words given a source text [44–46]. In this context, attention mechanisms enable agents to focus on the relevant parts of an (encoded) source sentence. Despite the prospect of using attention mechanisms to facilitate disentanglement, these works are rarely concerned with the resulting representation produced by the encoding neural network.

### A.1. $\beta$ -variational autoencoders

Autoencoders are one particular architecture used in the field of representation learning, whose goal is to map a high-dimensional input vector  $x$  to a lower-dimensional latent vector  $z$  using an encoding mapping  $E(x) = z$ . For autoencoders,  $z$  should still contain all information about  $x$ , i.e. it should be possible to reconstruct the input vector  $x$  by applying a decoding function  $D$  to  $z$ . The encoder  $E$  and the decoder  $D$  can be implemented using neural networks and trained unsupervised by requiring  $D(E(x)) = x$ .  $\beta$ -variational autoencoders ( $\beta$ -VAEs) are autoencoders where the encoding is regularised in order to capture statistically independent features of the input data in separate parameters [24].

In [15] a modified  $\beta$ -VAE, called *SciNet*, was used to answer questions about a physical system. The criterion by which the latent representation is disentangled is statistical independence equivalent to standard  $\beta$ -VAE methods. In the present work, we use a similar architecture but impose an operational criterion in terms of communicating agents for the disentanglement of parameters.

Another prior that was recently proposed to disentangle a latent representation is the *consciousness prior* [35]. There, the author suggests to disentangle abstract representations via an attention mechanism by assuming that, at any given time, only a few internal features or concepts are sufficient to make a useful statement about reality.

### A.2. Graph neural networks

Recently, graph neural networks (GNNs)[47] have been used to learn a dynamical model of interacting systems [48, 49]. In these scenarios, GNNs predict the behaviour of dynamical systems while encoding a model of the system in an interaction graph. In [48], the interaction structures are modelled explicitly by a latent interaction graph embedded in a VAE architecture. By adding prior beliefs about the graph structure such as sparsity, interpretable interaction graphs may be produced. In the present work, we refrain from using GNNs since we do not want to make any assumptions about the underlying model. Indeed, using GNNs requires prior knowledge about how to separate a physical system into sub-systems and poses certain restrictions on the functions being implemented by the encoder and decoder.

### A.3. Attention mechanisms

In [44], an attention mechanism is introduced to facilitate machine translation with encoder-decoder models. To that end, an encoder produces a sequence of annotations encoding the information about an arbitrary input sentence. Then, an attention mechanism is used to filter (or weigh) the annotations in accordance with their current relevance. The resulting context vector can be used by an encoder to produce a translation. This attention mechanism has been extended to multiple encoder-decoder models for multilingual neural machine translation [45] and summarisation [46]. In particular, in [46], a significant improvement over existing methods for text summarisation has been achieved by allowing encoders to communicate while producing a context vector. Here, we introduce a simplified attention mechanism that facilitates the generation of concise and structured latent representations within a communication setting. This is in contrast to other works employing attention mechanisms where the interpretability of latent representations is usually considered irrelevant.

### A.4. State representation learning

State representation learning (SRL) is a branch of representation learning for interactive problems [50]. For instance, in reinforcement learning [30] it can be used to capture the variation in an environment created by an agent's action [27, 28, 36, 51]. In [27] the representation is disentangled by an *independence prior* which encourages that independently controllable features of the environment are stored in separate parameters. A similar approach was recently introduced in [28] where model-based and model-free reinforcement learning

are combined to jointly infer a sufficient representation of the environment. The abstract representation becomes expressive by introducing *representation* and *interpretability priors*. Similarly, in [36] *robotic priors* are introduced to impose a structure reflecting the changes that occur in the world and in the way a robot can interact with it. As shown in [28] and [36], such requirements can lead to very natural representations in certain scenarios such as creating an abstract representation of a labyrinth or other navigation tasks.

In [52] many reinforcement learning agents with different tasks share a common representation which is being developed during training. They demonstrate that learning auxiliary tasks can help agents to improve learning of the overall objective. One important auxiliary task is given by a *feature control prior* where the goal is to maximise the activations of hidden neurons in an agent's neural network as they may represent task-relevant high-level features [53, 54]. However, this representation is not expressive or interpretable to the human eye since there is no criterion for disentanglement.

### A.5. Quantum state representation learning

Interestingly, various representation learning methods have been developed specifically for quantum state representation [33, 55–60]. Despite the large dimensionality of the Hilbert spaces, these methods have become increasingly efficient at representing specific [59, 60] and arbitrary [33, 55–58] quantum states. Specifically, attention-based methods have recently been shown to exhibit an empirical learning advantage over other methods for the task of quantum state tomography [33]. However, while the learning problem is similar, these works do not address the separability of variables for interpretability. We expect that such efficient methods for state tomography [33, 56] can be combined with our filter methods to autonomously generate operationally meaningful representations for complex quantum systems.

### A.6. Projective simulation

The projective simulation (PS) model for artificial intelligence [61] is a model for agency which employs a specific form of an episodic and compositional memory to make decisions. It has found applications in various areas of science, from quantum physics [16, 62, 63] to robotics [64, 65] and the modelling of animal behaviour [66]. Its memory consists of a network of so-called *clips* which can represent basic episodic experiences as well as abstract concepts. Besides the usage for generalisation [67, 68], these clip networks have already been used to represent abstract concepts in specific settings [17, 65]. In [17], PS was used to infer the existence of unobserved variables such as mass, charge or size which make an object respond in certain experimental settings in different ways. In this context, the authors point out the significance of exploration when considering the design of experiments, and thereby adopt the notion of reinforcement learning similar to [16]. In line with previous works, we will also discuss reinforcement learning methods for the design of experimental settings. Unlike previous works however, we provide an interpretation and formal description of decision processes which are specifically amenable to representation learning. Moreover, we employ neural networks architectures to infer continuous parameters from experimental data. In contrast, PS is inherently discrete and therefore better suited to infer high-level concepts.

In this work, we suggest to disentangle a latent representation of a neural network according to an operationally meaningful principle, by which agents should communicate as efficiently as possible to share relevant information to solve their tasks. Technically, we disentangle the representation through an attention mechanism according to different questions or tasks, as described in more detail in the main text.

## Appendix B. Implementation of filters

Due to the difficulty of implementing a binary value function with neural networks, we need to replace the ideal cost  $\mathcal{L}_c$  by a comparable version with a smooth filter function. To this end, instead of viewing the latent layer as the deterministic output of the encoder (the generalisation to multiple decoders is immediate), we consider each latent neuron  $j$  as being sampled from a normal distribution  $\mathcal{N}(\mu_j, \sigma_j)$ . The sampling is performed using the renormalisation trick [29], which allows gradients to propagate through the sampling step. The encoder outputs the expectation values  $\mu_j$  for all latent neurons. The logarithms of the standard deviations  $\log(\sigma_j)$  are provided by neurons, which we call *selection neurons*, that take no input and output a bias; the value of the bias can be modified during training using backpropagation. Using the logarithm of the standard deviation has the advantage that it can take any value, whereas the standard deviation itself is restricted to positive values. The ideal filter loss  $\mathcal{L}_c = \sum_j \dim(\varphi_j)$  is replaced by  $\tilde{\mathcal{L}}_c = -\sum_j \log(\sigma_j)$ .

Analogously, we replace the minimisation loss  $\mathcal{L}_m$  by a smoothed version  $\tilde{\mathcal{L}}_m$ .

The intuition for this scheme is as follows: when the network chooses  $\sigma_j$  to be small (where the standard deviation of  $\mu_j$  over the training set is used as normalisation), the decoder will usually obtain a sample that is close to the mean  $\mu_j$ ; this corresponds to the filter transmitting this value. In contrast, for a large value of  $\sigma_j$ , a sample from  $\mathcal{N}(\mu_j, \sigma_j)$  is usually far from the mean  $\mu_j$ ; this corresponds to the filter blocking this value. The loss  $\tilde{\mathcal{L}}_c$  is minimised when many of the  $\sigma_j$  are large, i.e. when the filter blocks many values.

Instead of thinking of probability distributions, one can also view this scheme as adding noise to the latent variables, with  $\sigma_j$  specifying the amount of noise added to the  $j$ -th latent neuron. If  $\sigma_j$  is large, the noise effectively hides the value of this latent neuron, so the decoder cannot make use of it.

We also note that  $\tilde{\mathcal{L}}_c$  is in principle unbounded. However, in practice this does not present a problem since the decoder can only approximately, but not perfectly, ignore the noisy latent neurons. For sufficiently large  $\sigma_j$ , the noise will therefore noticeably affect the decoders' predictions, and the additional loss incurred by worse predictions dominates the reduction in  $\tilde{\mathcal{L}}_c$  obtained from larger values for  $\sigma_j$ .

The success of this method to lead to an approximation of a binary filter depends on the weighting of the success loss in relation to the communication loss. This weight is a hyperparameter of the machine learning system.

## Appendix C. Multiple encoders

### C.1. Architecture

Up to now, we have assumed that there exists one encoder  $E$  who has access to the entire system to make an observation and to communicate its representation. However, just as different decoders  $D_i$  only deal with a part of the system, we can consider the more general scenario of having multiple encoders  $E_1, \dots, E_j$ . In this scenario, each encoder  $E_i$  makes different measurements on the system. For example, one encoder might observe a collision experiment between two particles, while another observes the trajectory of a particle in an external field. Here, only the aggregate observations of all encoders  $E_1, \dots, E_j$  provide sufficient information about the system required for the decoders  $D_1, \dots, D_k$  to make predictions.

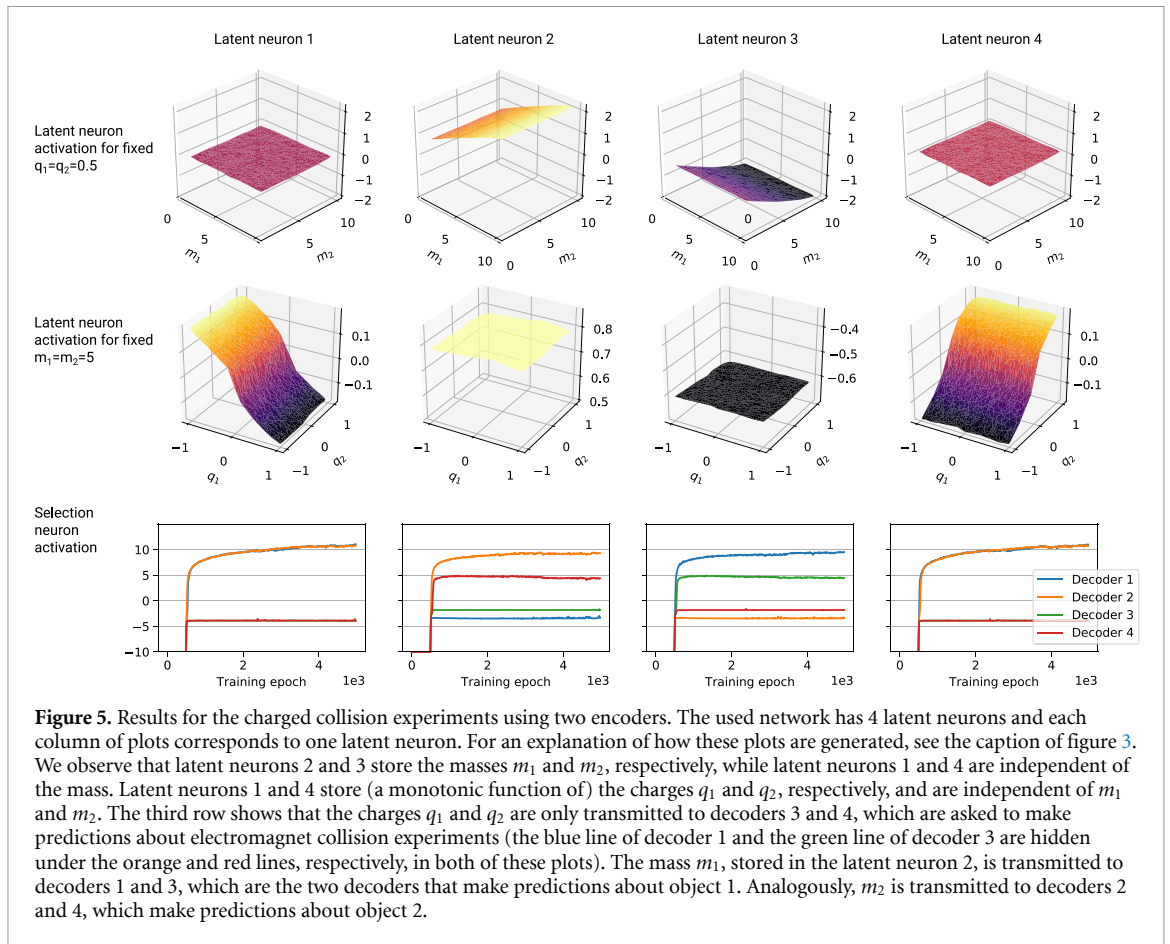
The formalisation is analogous to section 3 and we only sketch it here: each encoder function  $E_i : \mathcal{O}_i \rightarrow \mathbb{R}^{L_i}$  has its own domain of observations and latent spaces. The domain of the filter functions of the decoders  $D_1, \dots, D_k$  is now a cartesian product of the output spaces of the encoders (i.e. the output vectors of the encoders are concatenated and used as inputs to the filters).

In the case where a physical system has an operationally natural division into  $k$  interacting subsystems, a typical case would be to have the same number of encoders  $E_1, \dots, E_k$  as decoders  $D_1, \dots, D_k$ , where both  $E_i$  and  $D_i$  act on the same  $i$ -th subsystem. Here, we expect that  $E_i$  and  $D_i$  are highly correlated, i.e. the filter for  $D_i$  transmits almost all information from  $E_i$ , but less from other agents  $E_l$ . In this case, one can intuitively think of a single agent per subsystem  $i$ , that first makes an observation about that subsystem, then communicates with the other agents to account for the interaction between subsystems, and uses the information obtained from the communication to make a prediction about subsystem  $i$ .

### C.2. Results

In this appendix, we provide details about the representation learnt by a neural network with two encoders for the example involving charged masses introduced in section 4.1. The setup is the same as that in section 4.1, with the only difference being that we now use two encoders (the number of decoders and the predictions they are asked to make remain the same). Accordingly, we split the input into two parts: the measurement data from the data collection experiments involving object 1 are used as input for encoder 1, and the data for object 2 are used as input for encoder 2. Each encoder has to produce a representation of its input. We stress that the two encoders are separated and have no access to any information about the input of the other encoder. The representations of the two encoders are then concatenated and treated like in the single-encoder setup; that is, for each decoder, a filter is applied to the concatenated representation and the filtered representation is used as input for the decoder.

The results for this case are shown in figure 5. Comparing this result with the single-encoder case in the main text, we observe that here, the charges  $q_1$  and  $q_2$  are stored individually in the latent representation, whereas the single encoder stored the product  $q_1 \cdot q_2$ . This is because, even though the decoders still only require the product  $q_1 \cdot q_2$ , no single encoder has sufficient information to output this product: the inputs of encoders 1 and 2 only contain information about the individual charges  $q_1$  and  $q_2$ , respectively, but not their product. Hence, the additional structure imposed by splitting the input among two encoders yields a representation with more structure, i.e. with the two charges stored separately.



## Appendix D. Reinforcement learning

In the main text, we have considered scenarios where decoders make predictions about specific experimental settings and disentangle a latent representation by answering various questions. There, we understood *answering* different questions as making *predictions* about different aspects of a subsystem. Instead, we could have understood answers as *sequences of actions* that achieve a specific goal. For example, such a (delayed) goal may arise when building experimental settings that bring about a specific phenomenon, or more generally when designing or controlling complex systems. In particular, we may view a prediction as a one-step sequence.

In the case of predictions, it is easy to evaluate the quality of a prediction, since we are predicting quantities whose actual value we can directly observe in Nature. In contrast, the correct sequences of actions may not be easily accessible from a given experimental setting: upon taking a first action, we do not yet know whether this was a good or bad action, i.e. whether it is part of a ‘correct’ sequence of actions or not. Instead, we might only receive a few, sparsely distributed, discrete rewards while taking actions. In the typical case, there is only a binary reward at the end of a sequence of actions, specifying whether we reached the desired goal or not. Even in a setting where a single action suffices to reach a goal, such a binary reward would prevent us from defining a useful answer loss in the same manner as before. To see this, consider the toy example in figure 2(a) again: the decoder had to predict an angle  $\alpha_i$ , given a (representation of the) setting, specified by the parameters  $(m_{fix}, m_1, q_1, m_2, q_2)$  and a question  $v_i$ , in order to shoot the particle into the hole. We assumed that we can evaluate the ‘quality’ of the angle chosen by the decoder by comparing it to the optimal angle. Instead, this evaluation experiment could be viewed as a game where an agent is required to shoot the object into a hole. In this case, the agent only has access to a binary *reward* specifying whether or not it successfully hit the (finite-sized) hole. Then, we cannot define a smooth answer loss, which is required for training a neural network.

The problem that the feedback from the environment, i.e. the reward, is discrete or delayed can both be solved by viewing the situation as a reinforcement learning environment: given a representation of the setting (described by the masses and charges) and a question (a velocity), the agent can take different actions



(corresponding to different angles at which the mass is shot) and receives a binary reward if the mass lands in the hole. Therefore, we can employ reinforcement learning techniques and learn the optimal answer.

In reinforcement learning [30], an agent learns to choose actions that maximise its expected, cumulative, future, discounted reward. In the context of our toy example, we would expect a trained agent to always choose the optimal angle. Hence, predicting the behaviour of a trained agent would be equivalent to predicting the optimal answer and would impose the same structure on the parameterisation. In this example, the optimal solution consists of a single choice. In a more complex setting, it might not be possible to perform a (literal and metaphorical) hole-in-one. Generally, an optimal answer may require sequences of (discrete or continuous) actions, as it is for example the case for most control scenarios. In the settings we henceforth consider, questions might no longer be parameterised or given to the agent at all. That is, the question may be constant and just label the task that the agent has to solve.

In this appendix, we impose structure on the parameterisation of an experimental setting by assuming that different agents only require a subset of parameters to take a successful sequence of actions given their respective goals. To this end, we explain how experimental settings may be understood in terms of instances of a reinforcement learning environment and demonstrate that our architecture is able to generate an operationally meaningful representation of a modified standard reinforcement learning environment by predicting the behaviour of trained agents.

Moreover, in appendix F, we lay out the details for the algorithm that allows us to generate and disentangle the parameterisation of a reinforcement learning environment given various reinforcement learning agents trained on different tasks within the same environment. There, we also prove that this algorithm produces agents which are at least as good as the trained agents while only observing part of the disentangled abstract representation. The detailed architecture used for learning is described in appendix G and is combining methods from GPU-accelerated actor-critic architectures [69] and deep energy-based models [70] for projective simulation [61].

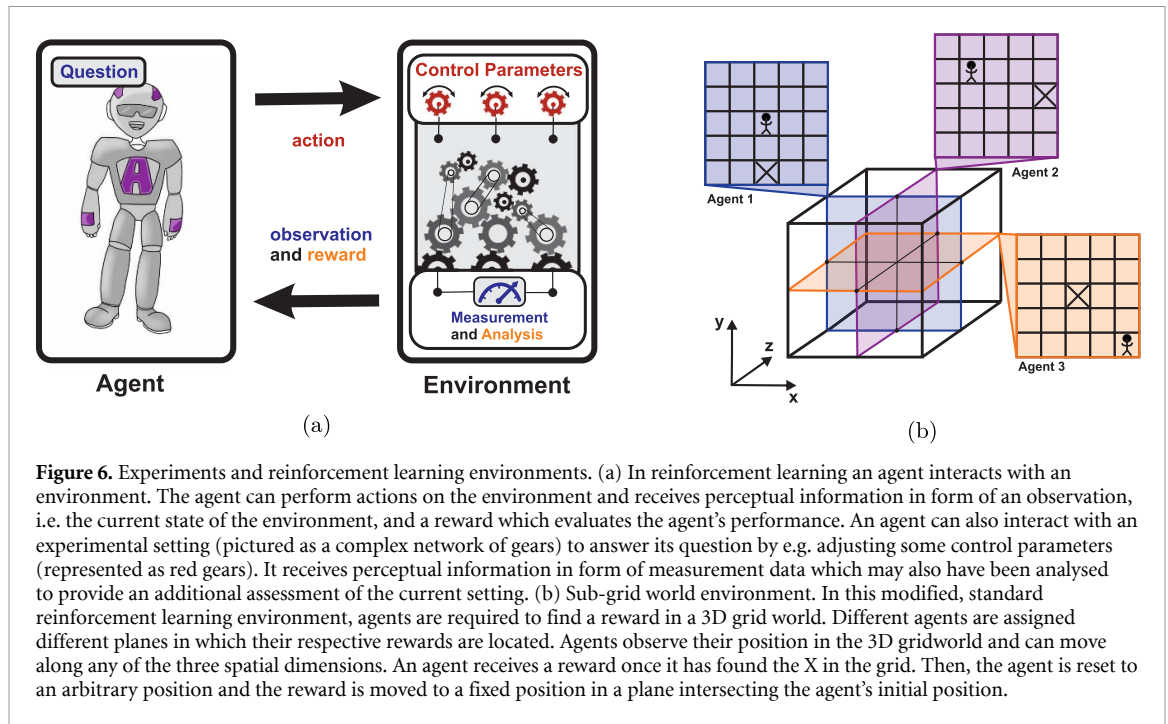
### D.1. Experiments as reinforcement learning environments

In [16] the design of experimental settings has been framed in terms of reinforcement learning [30] and here we formulate a similar setting: an agent interacts with experimental settings to achieve certain results. At each step the agent observes the current measurement data and/or setting and is asked to take an action regarding the current setting. This action may for instance affect the parameters of an experimental setting and hence might change the obtained measurement data. The measurement results are subsequently evaluated and the agent might receive a reward if the results are identified as ‘successful’. The correspondence between experiments as described in this appendix and reinforcement learning environments can be understood as follows (cf figure 6(a)). An *experimental setting* is interpreted as the current, internal state of an environment. The *measurement data* then corresponds to the observation received from the environment. The agent performs an action according to the current observation and its question. Actions may affect the internal state of the experimental setting. For instance, the *experimental parameters* describing the setting can be adjusted or chosen by an agent through actions. The reward function, which takes the current measurement data as input, describes the *objective* that is to be achieved by an agent.

Since the same experiment can serve more than one purpose, we can have many agents interact with the same experimental setting to achieve different results. In fact, we can expect most experiments to be highly complex and have many applications. For instance, photonic experiments have a plethora of applications [71] and various experimental and theoretical gadgets have been developed with these tools for different tasks [72–74]. In this context, we may task various agents to develop gadgets for different task. At first, we assume that all reinforcement learning agents have access to the entire measurement data. Once they have learnt to solve their respective tasks, we can employ our architecture from the main text to predict each agent’s behaviour. Effectively, we can then factorise the representation of the measurement data by imposing that only a minimal amount of information be required to predict the behaviour of each trained reinforcement learning agent. That is, we interpret the space of possible results in an experiment as high-dimensional manifold. When solving a given task however, an agent may only need to observe a submanifold which we want to parameterise.

Due to the close resemblance to reinforcement learning, we consider a standard problem in reinforcement learning in the following and demonstrate that our architecture is able to generate an operationally meaningful representation of the environment. More formally, we consider partially-observable Markov decision processes [75] (POMDP). Given the stationary policy of a trained agent, we impose structure on the observation and action space of the POMDP by discarding observations and actions which are rarely encountered. This structure defines the submanifold which we attempt to parameterise with our architecture. A detailed description of these environments is provided in appendix E.





## D.2. Example with a standard reinforcement learning environment

### D.2.1. Setup

Here, we consider the simplest version of a task that is defined on a high-dimensional manifold while the behaviour of a trained agent may become restricted to a submanifold. Consider a simple grid world task [30] where all agents can move freely in a three-dimensional space whereas only a subspace is relevant to finding their respective rewards (see figure 6(b)). Despite the apparent simplicity of this task, actual experimental settings may be understood as navigation tasks in complicated mazes [16]. This reinforcement learning environment can be phrased as a simple game.

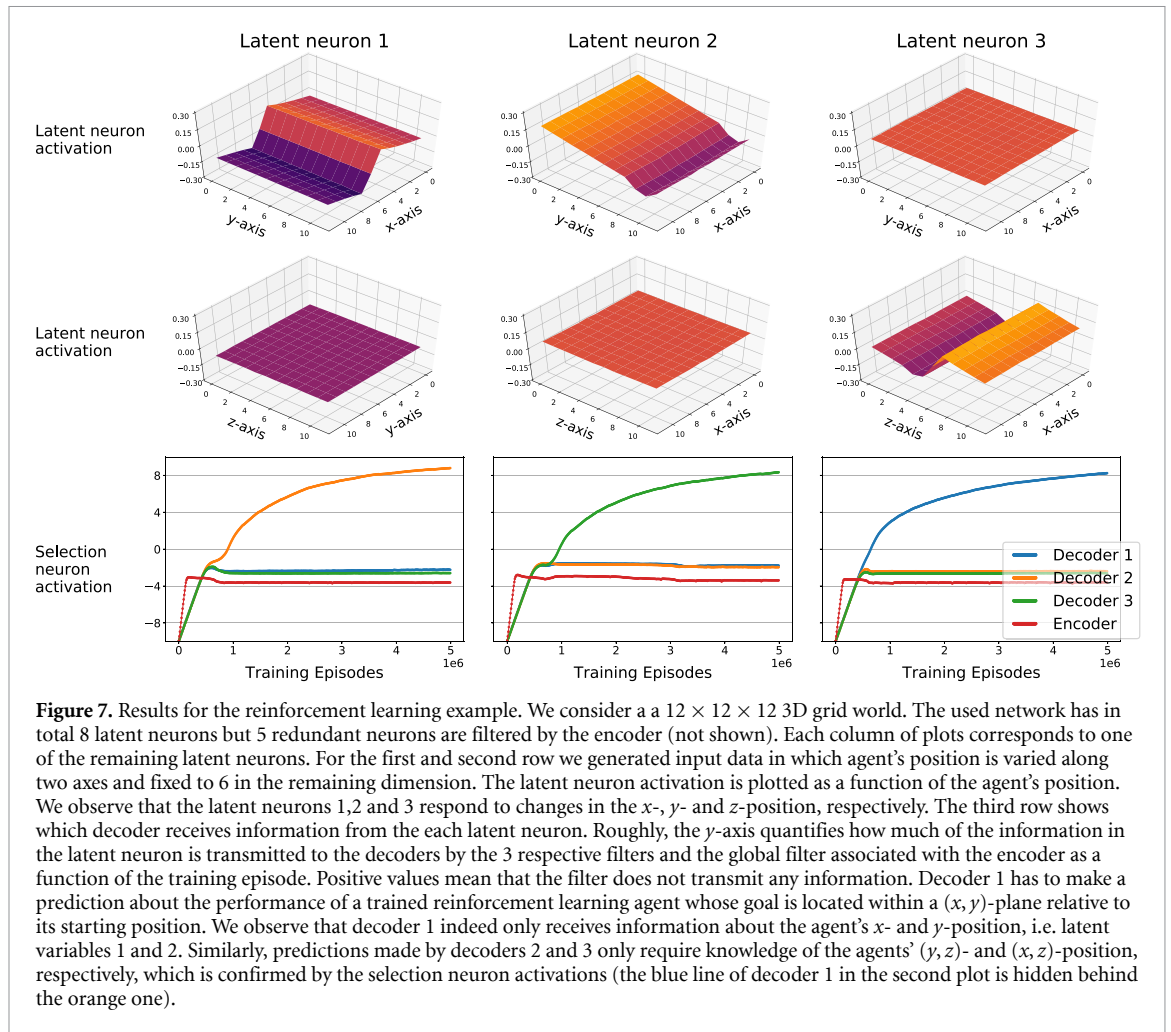
- Three reinforcement learning agents are positioned randomly within a discrete  $12 \times 12 \times 12$  grid world.
- The rewards for the agents are located in a  $(x, y)$ -,  $(y, z)$ - and  $(x, z)$ -plane relative to their respective initial positions. The locations of the rewards in their respective planes are fixed to  $(6, 11)$ ,  $(11, 6)$  and  $(6, 6)$ .
- The agents observe their position in the grid, but not the grid itself nor the reward.
- The agents can move freely along all three spatial dimension but cannot move outside the grid.
- An agent receives a reward if it can find the rewarded site within 400 steps. Otherwise, it is reset to a random position and the reward is re-positioned appropriately in the corresponding plane.

Generally, in reinforcement learning the goal is to maximise the expected future reward. In this case, this requires an agent to minimise the number of steps until a reward is encountered. Therefore, the optimal policy of an agent is to move on the shortest path towards the position of the reward within the assigned plane. Clearly, to predict the behaviour of an optimal agent, we require only knowledge of its position in the associated plane. We refer to appendix F for a concise protocol to *predict behaviour* of a reinforcement learning agent. A detailed description of the architecture can be found in appendix G.

### D.2.2. Results

The optimal policy of an agent is to move on the shortest path towards the position of the reward within its assigned plane. Predicting the behaviour of an optimal agent, we require only knowledge of its position in the associated plane. Indeed, we observe that 5 additional, redundant latent neurons are filtered by the encoder’s filter  $\varphi_E$  such that the remaining dimension of the representation is 3. In addition, the information about the coordinates should be separated such that the different agents have access to  $(x, y)$ ,  $(y, z)$  and  $(x, z)$ , respectively. Using the minimal number of parameters, this is only possible if the encoding agent  $A$  encodes the  $x, y, z$  coordinates of the agents  $B_1, B_2$  and  $B_3$  and communicates their respective position in the plane<sup>7</sup>.

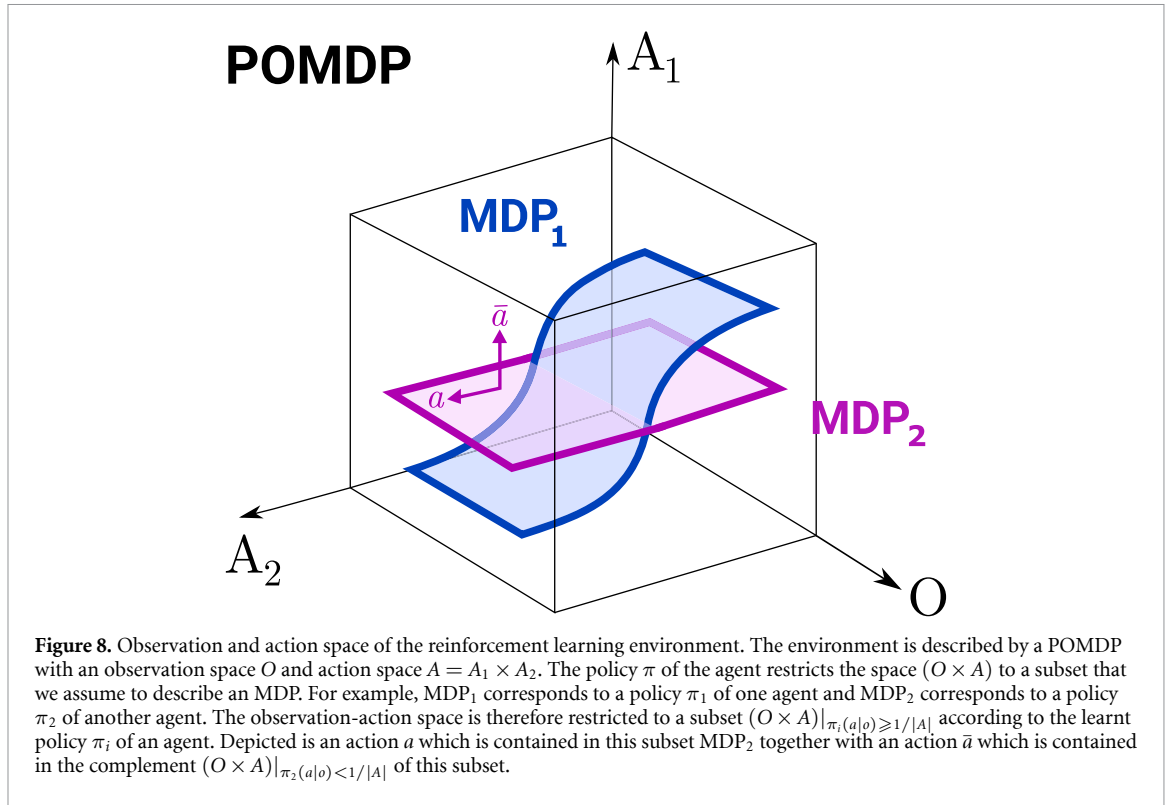
<sup>7</sup> Because the observation space is discrete, an encoding agent can, in principle, ‘cheat’ and encode multiple coordinates into a single neuron. In practice, this does not happen for sufficiently large state spaces.



We verify this by comparing the learnt representation to a hypothesised representation. For instance, we can test whether certain neurons respond to certain features in the experimental setting, i.e. reinforcement learning environment. Indeed, it can be seen from figure 7 that the neurons of the latent layer only respond separately to changes in the  $x$ ,  $y$  or  $z$  position of an agent respectively. Note that the encoding agent uses a nonlinear encoding of the  $x$ - and  $z$ -parameters. Interestingly, this reflects the symmetries in the problem: the reward is located at position  $x = z = 6$  whenever  $x$  or  $z$  are relevant coordinates for an agent, whereas for the  $y$ -coordinate, the reward is located at position 11. The encoding used by the network in this example suggests that an encoding of discrete bounded parameters may carry additional information about the hidden reward function, which may eventually help to improve our understanding of the underlying theory.

## Appendix E. Reinforcement learning environments for representation learning

In this appendix, we give a formal description of the reinforcement learning environments that we consider for representation learning. As we will see, the sub-grid world example in appendix D is a simple instance of such a class of environments. In general, we consider a reinforcement learning problem where the environment can be described as a Partially Observable Markov Decision Process [75] (POMDP), i.e. a MDP where not the full state of the environment is observed by the agent. We work with an observation space  $O = \{o_1, \dots, o_N\}$ , an action space  $A = \{a_1, \dots, a_M\}$  and a discount factor  $\gamma \in [0, 1)$ . This choice of environment does not reflect our specific choice of learning algorithm used to train the agent, as the latter does not construct so-called belief states that are commonly required to learn optimal policies in a POMDP. Rather, we want to show that our approach is applicable to slightly more general environments than Markov Decision Processes (MDPs) for which the learning algorithms we use are proven to converge to optimal policies in the limit of infinitely many interactions with the environment [30, 76]. The generalisation to POMDPs still preserves the 'Markovianity' of the environments and allows to consider only stationary (but not necessarily deterministic) policies  $\pi(a|o)$ , associated to stationary expected returns  $R_\pi(o)$ .



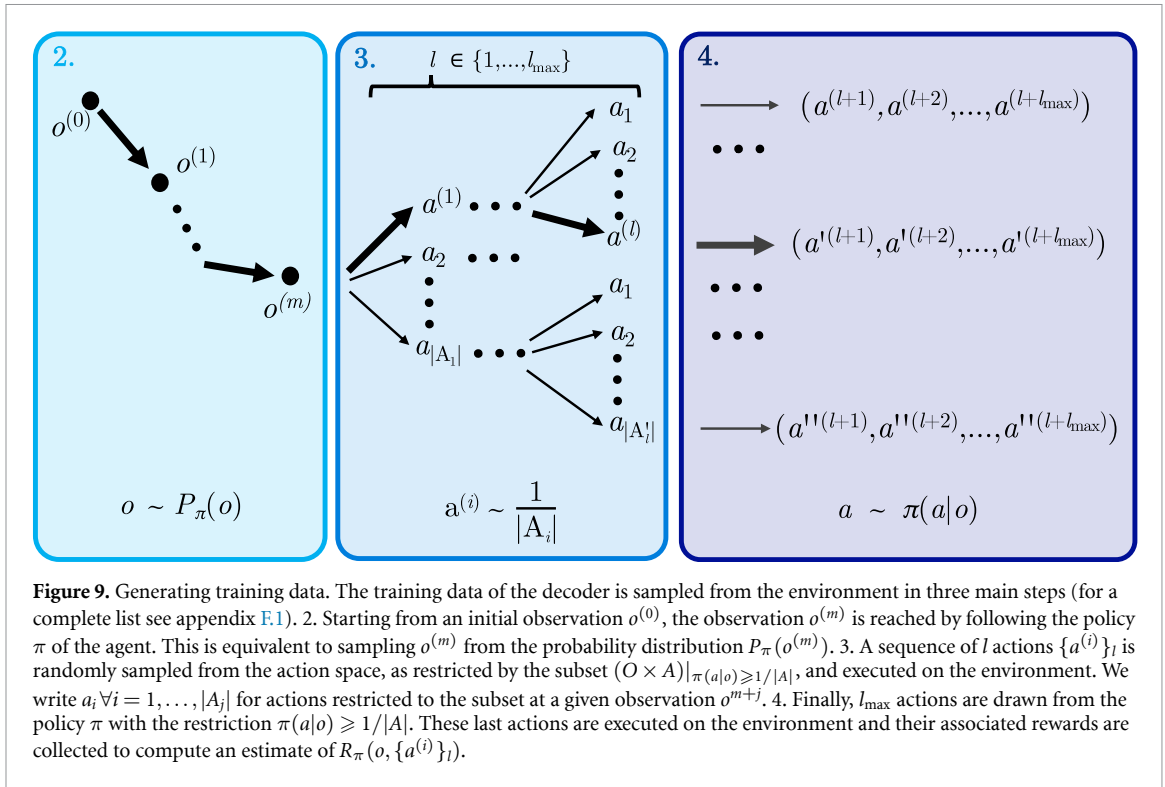
Now consider an agent which exhibits some *non-random* behaviour in this environment, which is characterised by a larger expected return than from a completely random policy. Such a stationary policy may restrict observation-action space  $(O \times A)$  to a subset  $(O \times A)|_{\pi(a|o) \geq 1/|A|}$  of observations and actions likely to be experienced by the agent depending on its learnt policy  $\pi$  and the environment dynamics. This notation indicates that, in any given observation, we discard actions that have probability less than random (i.e. less than  $\frac{1}{|A|}$ ) of being taken by the agent, indicating that the agent’s policy has learnt (un)favouring actions. In general, discarding actions also restricts the observation space. The subset  $(O \times A)|_{\pi(a|o) \geq 1/|A|}$ , along with the POMDP dynamics, describes a new environment. For simplicity, we assume that the restricted environment can be described by an MDP. This is trivially the case if the original environment is itself an MDP, and also the case for the sub-grid world environment discussed in the main text. The MDP inherits the discount factor  $\gamma \in [0, 1)$  of the original POMDP, which allows us to consider w.l.o.g. finite-horizon MDPs<sup>8</sup>, which are MDPs of finite episode lengths (here, we set the maximum length to  $3l_{\max}$ ). A conceptual view on this POMDP restricted by policies is provided in figure 8.

### Appendix F. Representation learning in reinforcement learning environments

In our approach to factorising abstract representations of reinforcement learning agents, we assume that an agent’s policy can impose structure on an environment (as described in appendix E) and we want this structure to be reflected in its latent representation. Therefore, decoders need to predict the *behaviour* of a reinforcement learning agent while requiring minimal knowledge of the latent representation. However, we still lack a definition of what it means for a decoder to predict the behaviour of an agent. Here, we consider decoders predicting the expected rewards for these agents given the representation communicated by the encoder. Later, we show that this is enough to produce a policy which is at least as good as the policy of the reinforcement learning agent.

To be precise, each decoder attempts to learn the expected return  $R_\pi(o, a)$  given an observation-action pair  $(o, a) \in (O \times A)|_{\pi(a|o) \geq 1/|A|}$  under the policy  $\pi$  of an agent. For observation-action pairs outside the restricted subset we assign values 0. The input space of the decoder and the restriction to the subset is illustrated in figure 8. In fact, decoders not only learn to predict  $R$  for a single action but for a sequence of actions  $\{a^{(1)}, \dots, a^{(l)}\}_l$  with length  $l \geq 1$ . This is because it can help stabilise the latent representation of

<sup>8</sup> An infinite-horizon MDP with discount factor  $\gamma \in [0, 1)$  can be  $\epsilon$ -approximated by a finite-horizon MDP with horizon  $l_{\max} = \log_\gamma(\frac{\epsilon(1-\gamma)}{\max_{\sigma} |R(\sigma)|})$ .



**Figure 9.** Generating training data. The training data of the decoder is sampled from the environment in three main steps (for a complete list see appendix F.1). 2. Starting from an initial observation  $o^{(0)}$ , the observation  $o^{(m)}$  is reached by following the policy  $\pi$  of the agent. This is equivalent to sampling  $o^{(m)}$  from the probability distribution  $P_\pi(o^{(m)})$ . 3. A sequence of  $l$  actions  $\{a^{(i)}\}_l$  is randomly sampled from the action space, as restricted by the subset  $(O \times A)|_{\pi(a|o) \geq 1/|A|}$ , and executed on the environment. We write  $a_i \forall i = 1, \dots, |A_j|$  for actions restricted to the subset at a given observation  $o^{m+j}$ . 4. Finally,  $l_{\max}$  actions are drawn from the policy  $\pi$  with the restriction  $\pi(a|o) \geq 1/|A|$ . These last actions are executed on the environment and their associated rewards are collected to compute an estimate of  $R_\pi(o, \{a^{(i)}\}_l)$ .

environments with small actions spaces and simple reward functions. In practice however,  $l = 1$  is sufficient to obtain a proper representation. In the same way, we can help to stabilise the latent representation by forcing an additional decoder to reconstruct the input from the latent representation. For brevity, we write  $\{a^{(i)}\}_l$  for sequences of actions of length  $l$ .

The method described in this appendix, allows us to pick a number of reinforcement learning agents that have learnt to solve various problems on a specific kind of reinforcement learning environment (see appendix E) and parameterise the subspaces relevant for solving their respective tasks. Specifically, the procedure splits into three parts:

- (a) Train reinforcement learning agents.
- (b) Generate training data for representation learning from reinforcement learning agents (see appendix F.1).
- (c) Train encoders with decoders on training data such that they can reproduce (w.r.t. performance) the policy of the reinforcement learning agents (see appendix F.2).

The purpose of this appendix is to prove that the trained decoders contain enough information to derive policies that perform as well as the ones learnt by their associated agents. Only if this is the case, we can claim that the structure imposed by the decoder reflects the structure imposed on the environment by an agent’s policy. To that end, we start by (ii) introducing the method to generate the training data, followed by (iii) a construction of a policy from a trained decoder with given performance bounds.

**F.1. Training data generation**

The decoders are trained to predict the return values  $R_\pi(o, \{a^{(i)}\}_l)$  for observations  $o$  and sequences of actions  $\{a^{(i)}\}_l$  of arbitrary length  $l \leq l_{\max}$ , given a policy  $\pi$ . The training data is then generated as follows (see figure 9):

- (a) Sample two numbers  $m, l$  uniformly at random from  $\{1, \dots, l_{\max}\}$ .
- (b) Start an environment rollout with the trained agent’s policy  $\pi$  for  $m$  steps until the observation  $o^{(m)}$  is reached.
- (c) Continue the rollout with  $l$  actions which are sampled uniformly at random from the action space as restricted by the subset  $(O \times A)|_{\pi(a|o) \geq 1/|A|}$ <sup>9</sup>.

<sup>9</sup> Note that these actions need to be sampled sequentially from the current policy of the agent, given an observation.

- (d) The rollout is completed with  $l_{\max}$  steps according to the policy  $\pi$  of the agent restricted to the subset.
- (e) The rewards  $r_j$  associated to the last  $l_{\max}$  steps are collected and used to evaluate an estimate of  $R_{\pi}(o, \{a^{(i)}\}_l) = \sum_{j=1}^{l_{\max}} \gamma^{j-1} r_j$ .
- (f) Collect a tuple consisting of observation  $o^{(m)}$ , actions  $\{a^{(i)}\}_l$  and reward  $R_{\pi}(o, \{a^{(i)}\}_l)$ .
- (g) Collect tuples  $(o^{(m)}, \{\bar{a}^{(i)}\}_l, 0)$  for all actions  $\bar{a}^{(i)}$  which are not in the restricted subset  $(O \times A)|_{\pi(a|o) \geq 1/|A|}$ .
- (h) Repeat the procedure.

Note, that this algorithm does not require any additional control over the environment beyond initialisation and performing actions. That is, it can be generated *on-line* while interacting with the environment. In the case of a deterministic MDP and policy, one iteration of this algorithm yields the exact values of  $R_{\pi}(o, \{a^{(i)}\}_l)$ . In the case of a stochastic MDP or policy, one obtains instead an unbiased estimate of these values due to the possible fluctuations caused by the stochasticity of the environment dynamics and the policy. Repeated iterations of the algorithm followed by averaging of the estimates allows to decrease the estimation error. We neglect this estimation error in the next section.

The collected tuples are used to train the encoder and decoder through the answer loss  $\mathcal{L}_a$  as discussed in the main text. In practice, short action sequences are sufficient to factorise the abstract representation of the trained agents. In the example of the main text,  $l = 1$  was used. We kept the general description of the return function with arbitrary sequence lengths as a possible extension for more stable factorisations.

### F.2. Reinforcement learning policy from trained decoders

Let us call  $R_{\text{NN}}$  the function learnt by the decoder. We prove that a policy  $\pi'$  satisfying  $R_{\pi'}(o^{(0)}) \geq R_{\pi}(o^{(0)}) \forall o^{(0)}$  in the MDP can be constructed from the decoder if it was trained with a certain loss  $\varepsilon$ .

**Theorem 1.** *Given a POMDP with observation-action space  $O \times A$  and a policy  $\pi$  that restricts the POMDP into an MDP with observation-action space  $(O \times A)|_{\pi(a|o) \geq 1/|A|}$ , there exists a policy  $\pi'$  that satisfies  $R_{\pi'}(o^{(0)}) \geq R_{\pi}(o^{(0)}) \forall o^{(0)}$  in the MDP and that can be derived from a function which is  $\varepsilon$ -close (in terms of a mean squared error), with  $\varepsilon > 0$ , to:*

$$\tilde{R}_{\pi}(o, a) = \begin{cases} R_{\pi}(o, a) & \text{if } (o, a) \in (O \times A)|_{\pi(a|o) \geq 1/|A|} \\ 0 & \text{otherwise} \end{cases}$$

**Proof.** For clarity, we first prove that the construction of  $\pi'$  is possible if the return values are learnt perfectly, i.e. the training loss  $\mathcal{L}$  is zero. Later, we relax this assumption and show that the proof still holds for non-zero values of the loss.

We choose the loss function to be a weighted mean square error on the subset extended to arbitrary length action sequences, i.e.  $(O \times \bigcup_{k=1, \dots, l_{\max}} A^k)|_{\pi(a|o) \geq 1/|A|}$ ,

$$\mathcal{L} = \sum_{o, \{a^{(i)}\}_l} P_{\pi}(o) \frac{1}{l_{\max} \prod_i |A_i|} (R_{\pi}(o, \{a^{(i)}\}_l) - R_{\text{NN}}(o, \{a^{(i)}\}_l))^2.$$

An analogous approach yields similar results for other loss functions. Here,  $P_{\pi}(o)$  is the probability that the observation  $o$  is obtained given that the agent follows the policy  $\pi$  and  $A_i$  is the action space from which the action  $a^{(i)}$  is sampled, as restricted by the subset. Now, let us further restrict the sum to action sequences of length one, i.e.

$$\mathcal{L}' = \sum_{o, a} P_{\pi}(o) \frac{1}{l_{\max} |A_1|} (R_{\pi}(o, a) - R_{\text{NN}}(o, a))^2,$$

for which it is easily verified that  $\mathcal{L}' \leq \mathcal{L}$ .

Using  $R_{\text{NN}}$ , we derive the following policy:

$$\pi'(a|o) = \begin{cases} 1 & \text{if } a = \operatorname{argmax}_{a'} R_{\text{NN}}(o, a') \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Since  $R_{\text{NN}}(o, a)$  corresponds to the return of the policy  $\pi$  after observing  $o$  and taking action  $a$ , maximising this return hence leads to a return  $R_{\pi'}(o) \geq R_{\pi}(o) \forall o \in O_{\text{MDP}}$ .

In the following, we discuss the implications of the decoder not learning to reproduce  $R_\pi$  perfectly, i.e.  $\mathcal{L} = \varepsilon > 0$ . More precisely, we derive a bound on  $\varepsilon$  under which a policy  $\pi'$  satisfying  $R_{\pi'}(o^{(0)}) \geq R_\pi(o^{(0)}) \forall o^{(0)}$  in the MDP can still be constructed from the decoder.

The decoder can be used to construct the policy  $\pi'$  defined in equation (1) if the approximation error of  $R_{\text{NN}}$  is small enough to distinguish the largest and second-largest return values  $R_\pi(o, a)$  given an observation  $o$ . In the worst case, this difference can be as small as the smallest difference between any two returns given an observation

$$\varepsilon' = \gamma^{l_{\max}} \delta_R,$$

where  $\delta_R = \min_i |r_{i+1} - r_i|$  is the minimal non-zero difference between any two values the reward function of the environment can assign (including a reward  $r = 0$ ).

Let us set,

$$\mathcal{L}' \leq \varepsilon = \frac{\gamma^{2l_{\max}} \delta_R^2 \delta_\pi}{16|A|l_{\max}}$$

where  $\delta_\pi = \min_{o \in O_{\text{MDP}}} \{P_\pi(o) \mid P_\pi(o) \neq 0\}$ . That is,

$$\sum_{o,a} P_\pi(o) \frac{1}{l_{\max}|A_1|} (R_\pi(o, a) - R_{\text{NN}}(o, a))^2 \leq \frac{\gamma^{2l_{\max}} \delta_R^2 \delta_\pi}{16|A|l_{\max}}$$

and hence,  $\forall (o, a) \in (O \times A) \mid_{\pi(a|o) \geq 1/|A|}$

$$\begin{aligned} P_\pi(o) \frac{1}{l_{\max}|A_1|} (R_\pi(o, a) - R_{\text{NN}}(o, a))^2 &\leq \frac{\gamma^{2l_{\max}} \delta_R^2 \delta_\pi}{16|A|l_{\max}} \\ (R_\pi(o, a) - R_{\text{NN}}(o, a))^2 &\leq \frac{\gamma^{2l_{\max}} \delta_R^2}{16} \\ |R_\pi(o, a) - R_{\text{NN}}(o, a)| &\leq \frac{\varepsilon'}{4}. \end{aligned}$$

It is sufficient for  $R_{\text{NN}}$  to approximate  $R_\pi$  with precision  $\frac{\varepsilon'}{4}$ . Therefore, it is sufficient to bound the error of the loss function  $\mathcal{L}$  by

$$\varepsilon \leq \frac{\gamma^{2l_{\max}} \delta_R^2 \delta_\pi}{16|A|l_{\max}}.$$

□

This worst case analysis shows that the error needs to be exponentially small with respect to the parameters of the problem so that we can derive strong performance bounds of the policy on the entire subset. In practice, we expect to be able to derive a functional policy even with higher losses during the training of the decoder.

## Appendix G. Model implementation for representation learning in reinforcement learning environments

In this appendix, we give the details for the architecture that has been used to factorise the abstract representation of a reinforcement learning environment. The code has been made available at [https://github.com/HendrikPN/reinforced\\_scinet](https://github.com/HendrikPN/reinforced_scinet). For convenience, we repeat the training procedure here:

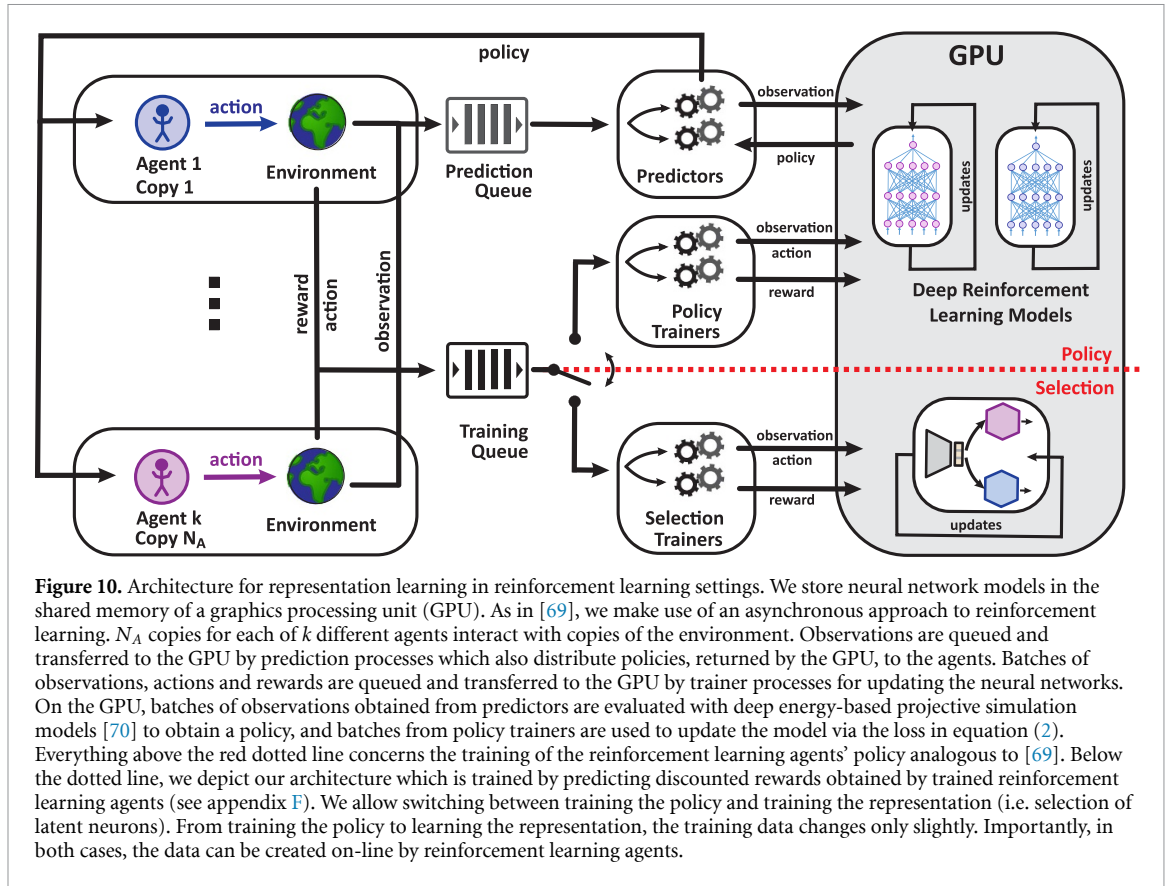
- (a) Train reinforcement learning agents.
- (b) Generate training data for representation learning from reinforcement learning agents (see appendix F.1).
- (c) Train encoders with decoders on training data to learn an abstract representation (see appendix F.2).

The whole procedure is encompassed by a single algorithm (see figure 10).

### G.1. Asynchronous reinforcement and representation learning

Due to the highly parallelisable setting, we make use of asynchronous methods for reinforcement learning [69]. That is, at all times, we have stored the neural network models in the shared memory of a graphics processing unit (GPU). Both, predicting and training, are therefore outsourced to the GPU while interactions of various agents with their environments are happening in parallel on central processing units (CPUs). The interface between the GPU and CPU is provided by two main processes which are assigned their





own threads on CPUs, *predictor*<sup>10</sup> and *training* processes. Predictor processes get observations from a *prediction queue* and batch them in order to transfer them to the GPU where a forward pass of the deep reinforcement learning model is performed to obtain the policies (i.e. probability distributions over actions) which are redistributed to the respective agents. Training processes batch training data as appropriate for the learning model in the same way as predictors batch observations. This data is transferred to the GPU to update the neural network. In our case, we need to be able to switch between two such training processes. One for training a policy as in [69] and as required by step (i) of our training procedure, and one for representation learning as required by step (iii). Interestingly, the training data which is used by the policy trainers in step (i) is very similar to the training data which is used by the selection trainers in step (iii). Therefore, in the transition from step (i) to (iii), we just have to slightly alter the data which is sent to the training queue as required by the algorithm in section F.1. Note that the similarity of the training data for the two training processes is due to the specific deep reinforcement learning model under consideration as described in the following section. For further details on the implementation of asynchronous reinforcement learning methods on GPUs see [69].

## G.2. Deep energy-based projective simulation model

The deep learning model used for the numerical results obtained here is a deep energy-based projective simulation (DPS) model as first presented in [70]. We chose this model because it allows us to easily switch between training the policy and training the decoders since the training data is almost the same for both. In fact, besides different initial biases and network sizes, the models used as reinforcement learning agents and the models used for decoders are the same.

The DPS model predicts so-called  $h$ -values  $h(o, a)$  given an observation  $o$  and action  $a$ . The loss function aims to minimise the distance between the current  $h$ -value  $h_t(o, a)$  and a target  $h$ -value  $h_t^{\text{tar}}(o, a)$  at time  $t$ , given as

$$\mathcal{L} = |h_t(o, a) - h_t^{\text{tar}}(o, a)|. \quad (2)$$

<sup>10</sup> Here we adopt the notation from [69]. That is, the predictor processes used here are not related to the prediction process associated with decoders in the main text.

Note that we are free to choose other loss functions such as the mean square error, or a Huber loss. We want the current  $h$ -value to be updated such that it maximises the future expected reward. Approximating this reward at time  $t$  for a given discount factor, we write

$$R_t = \sum_{j=1}^{l_{\max}} (1 - \eta)^{j-1} r_{t+j}$$

where  $\eta \in (0, 1]$  is the so-called *glow* parameter accounting for the discount of rewards  $r_{t+j}$  obtained after observing  $o$  and taking action  $a$  at time  $t$  up to a temporal horizon  $l_{\max}$ . The target  $h$ -value can then be associated with this discounted reward as follows,

$$h_t^{\text{tar}}(o, a) = (1 - \gamma_{\text{PS}})h_t(o, a) + R_t,$$

where  $\gamma_{\text{PS}} \in [0, 1)$  is the so-called *forgetting* parameter used for regularisation. The  $h$ -values are used to derive a policy through the softmax function,

$$\pi(a|o) = \frac{e^{\beta h(o, a)}}{\sum_{a'} e^{\beta h(o, a')}},$$

where  $\beta > 0$  is an inverse temperature parameter which governs the drive for exploration versus exploitation. The tabular approach to projective simulation has been proven to converge to an optimal policy in the limit of infinitely many interactions with certain MDPs [76] and has shown to perform as good as standard approaches to reinforcement learning on benchmarking tasks [77]. For a detailed description and motivation of the DPS model we refer to [70].

Note that the training data required to define the loss in equation (2) consists of tuples containing observations, actions and discounted rewards  $(o, a, R)$ . Since this is in line with the training data required for training the decoders as described in appendix F.1, this model is particularly well suited for the combination with representation learning as introduced in this paper.

## Appendix H. Classical mechanics derivation for charged masses

In this appendix, we provide the analytic solution to the evaluation experiment in section 4.1 that we use to evaluate the cost function for training the neural networks. This is a fairly direct application of the generic Kepler problem, but we include the derivation for the sake of completeness. We use the notation of [78].

The setup we consider is shown in figure 11. Our goal is to derive a function  $v_0(\varphi)$  that, for fixed  $q, Q, d_0$  and given  $\varphi$ , outputs an initial velocity for the left mass such that the mass will reach the hole. Introducing the inverse radial coordinate  $u = \frac{1}{r}$ , the orbit  $r(\theta)$  of the left mass obeys the following differential equation (see e.g. [78, section 4.3]):

$$\frac{d^2 u}{d\theta^2} + u = \frac{k}{l^2}, \quad (3)$$

with the constant

$$k = \frac{-qQ}{4\pi\epsilon_0 m} \quad (4)$$

and the mass-normalised angular momentum

$$l = r^2 \frac{d\theta}{dt}. \quad (5)$$

This is a conserved quantity and we can determine it from the initial condition of the problem

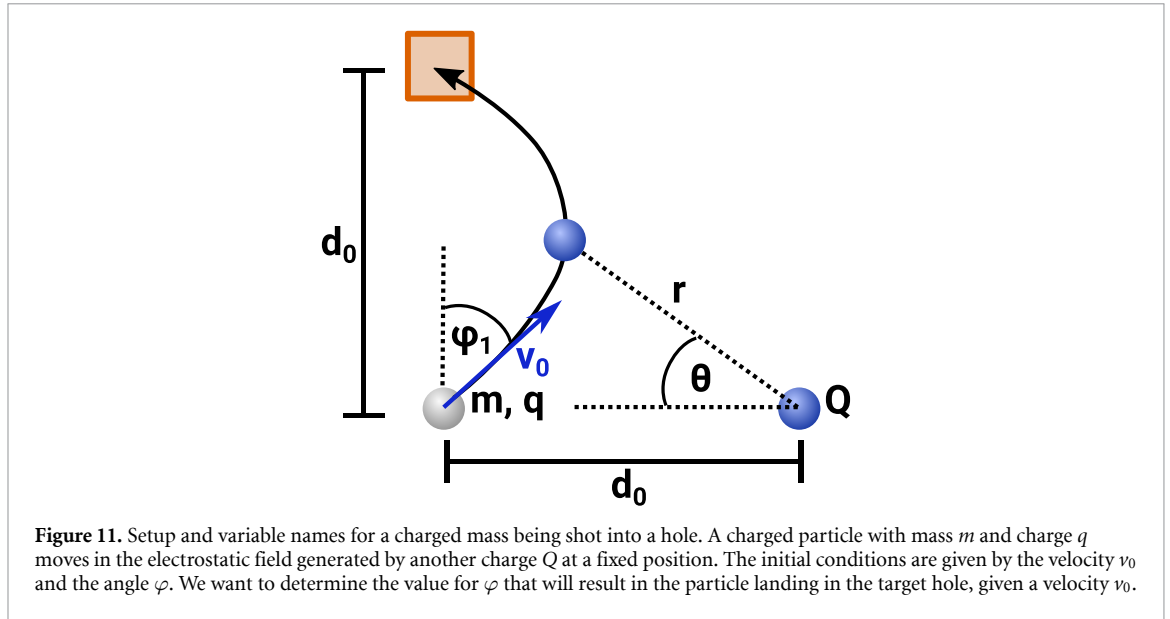
$$l = d_0 v_0 \cos \varphi. \quad (6)$$

The general solution to equation (3) is given by

$$u = A \cos(\theta - \theta_0) + \frac{k}{l^2}, \quad (7)$$

where  $A$  and  $\theta_0$  are constants to be determined from the initial conditions. The initial conditions are

$$r(\theta = 0) = \frac{1}{A \cos(\theta_0) + \frac{k}{l^2}} = d_0, \quad (8)$$



**Figure 11.** Setup and variable names for a charged mass being shot into a hole. A charged particle with mass  $m$  and charge  $q$  moves in the electrostatic field generated by another charge  $Q$  at a fixed position. The initial conditions are given by the velocity  $v_0$  and the angle  $\varphi$ . We want to determine the value for  $\varphi$  that will result in the particle landing in the target hole, given a velocity  $v_0$ .

$$\left. \frac{dr}{d\theta} \right|_{\theta=0} = \frac{-A \sin \theta_0}{\left(A \cos \theta_0 + \frac{k}{l^2}\right)^2} \frac{v_0 \cos \varphi}{d_0} = v_0 \sin \varphi. \tag{9}$$

Combining these yields

$$A \cos \theta_0 = \frac{1}{d_0} - \frac{k}{l^2}, \tag{10}$$

$$A \sin \theta_0 = -\frac{1}{d_0} \tan \varphi. \tag{11}$$

The condition that the mass reaches the hole is expressed in terms of  $r(\theta)$  as follows:

$$r\left(\theta = \frac{\pi}{4}\right) = \frac{1}{A \cos\left(\frac{\pi}{4} - \theta_0\right) + \frac{k}{l^2}} = \sqrt{2}d_0. \tag{12}$$

Using  $\cos(\pi/4 - \theta_0) = \cos(\theta_0)/\sqrt{2} + \sin(\theta_0)/\sqrt{2}$  and the definition of  $l$  as well as equations (10) and (11), we can solve this for  $v_0$ :

$$v_0^2 = \frac{(\sqrt{2} - 1)k}{d_0} \frac{1}{\cos \varphi \sin \varphi}. \tag{13}$$

Restricting  $\varphi$  to a suitably small interval, this function is injective and has a well-defined inverse  $\varphi(v_0)$ . The neural network has to compute this inverse from operational input data. To generate valid question-answer pairs, we evaluate  $v_0(\varphi)$  on a large number of randomly chosen  $\varphi$  (inside the interval where the function is injective).

### ORCID iD

Hendrik Poulsen Nautrup  <https://orcid.org/0000-0001-7815-7006>

### References

- [1] Nielsen M A 2018 *Neural Networks and Deep Learning* (Determination Press)
- [2] LeCun Y, Bengio Y and Hinton G 2015 Deep learning *Nature* **521** 436
- [3] Silver D *et al* 2016 Mastering the game of Go with deep neural networks and tree search *Nature* **529** 484
- [4] Dunjko V and Briegel H J 2018 Machine learning and artificial intelligence in the quantum domain: a review of recent progress *Rep. Prog. Phys.* **81** 074001
- [5] Roscher R, Bohn B, Duarte M F and Garcke J 2020 Explainable machine learning for scientific insights and discoveries *IEEE Access* **8** 42200
- [6] Carleo G, Cirac I, Cranmer K, Daudet L, Schuld M, Tishby N, Vogt-Maranto L and Zdeborová L 2019 Machine learning and the physical sciences *Rev. Mod. Phys.* **91** 045002

- [7] Bates C, Battaglia P W, Yildirim I and Tenenbaum J B 2015 Humans predict liquid dynamics using probabilistic simulation *Proc. of the 37th Annual Conf. Cognitive Science Society* vol 1 p 172
- [8] Wu J, Yildirim I, Lim J J, Freeman B and Tenenbaum J 2015 Galileo: perceiving physical object properties by integrating a physics engine with deep learning *Advances in Neural Information Processing Systems* vol 28 (Curran associates, Inc.) pp 127–35
- [9] Bramley N R, Gerstenberg T, Tenenbaum J B and Gureckis T M 2018 Intuitive experimentation in the physical world *Cogn. Psychol.* **105** 9
- [10] Rempe D, Sridhar S, Wang H and Guibas L J 2019 Learning generalizable physical dynamics of 3D rigid objects (arXiv:1901.00466)
- [11] Kissner M and Mayer H 2019 Adding intuitive physics to neural-symbolic capsules using interaction networks (arXiv:905.09891)
- [12] Ehrhardt S, Monszpart A, Mitra N and Vedaldi A 2018 Unsupervised intuitive physics from visual observations (arXiv:1805.05086)
- [13] Ye T, Wang X, Davidson J and Gupta A 2018 Interpretable intuitive physics model (arXiv:1808.10002)
- [14] Zheng D, Luo V, Wu J and Tenenbaum J B 2018 Unsupervised learning of latent physical properties using perception-prediction networks (arXiv:1807.09244)
- [15] Iten R, Metger T, Wilming H, del Rio L and Renner R 2020 Discovering physical concepts with neural networks *Phys. Rev. Lett.* **124** 010508
- [16] Melnikov A A *et al* 2018 Active learning machine learns to create new quantum experiments *Proc. Natl Acad. Sci.* **115** 1221
- [17] Ried K, Eva B, Müller T and Briegel H 2019 How a minimal learning agent can infer the existence of unobserved variables in a complex environment (arXiv:1910.06985)
- [18] Briegel H J 2012 On creative machines and the physical origins of freedom *Sci. Rep.* **2** 522
- [19] Wu T and Tegmark M 2019 Toward an artificial intelligence physicist for unsupervised learning *Phys. Rev. E* **100** 033311
- [20] De Simone A and Jacques T 2019 Guiding new physics searches with unsupervised learning *Eur. Phys. J. C* **79** 289
- [21] D’Agnolo R T and Wulzer A 2019 Learning new physics from a machine *Phys. Rev. D* **99** 015014
- [22] Rahaman N, Wolf S, Goyal A, Remme R and Bengio Y 2019 Learning the arrow of time (arXiv:1907.01285)
- [23] Bengio Y, Courville A and Vincent P 2012 Representation learning: a review and new perspectives *IEEE Trans. Pattern Anal. Mach. Intell.* **35** 1798
- [24] Higgins I *et al* 2017 Beta-VAE: learning basic visual concepts with a constrained variational framework *ICLR*
- [25] Chen T Q, Li X, Grosse R B and Duvenaud D 2018 Isolating sources of disentanglement in variational autoencoders (arXiv:1802.04942)
- [26] Kim H and Mnih A 2018 Disentangling by factorising (arXiv:1802.05983)
- [27] Thomas V *et al* 2018 Disentangling the independently controllable factors of variation by interacting with the world (arXiv:1802.09484)
- [28] François-Lavet V, Bengio Y, Precup D and Pineau J 2019 Combined reinforcement learning via abstract representations *The 33rd AAAI Conf. on Artificial Intelligence, AAAI 2019* p 3582
- [29] Kingma D P and Welling M 2013 Auto-encoding variational bayes (arXiv:1312.6114)
- [30] Sutton R S and Barto A G 1998 *Reinforcement Learning: An Introduction* (Cambridge: MIT press)
- [31] Paris M and Reháček J (eds) 2004 *Quantum State Estimation (Lecture Notes in Physics)* (Berlin: Springer)
- [32] Gamel O 2016 Entangled bloch spheres: bloch matrix and two-qubit state space *Phys. Rev. A* **93** 062320
- [33] Cha P *et al* 2021 Attention-based quantum tomography *Mach. Learn.: Sci. Technol.* **3** 01LT01
- [34] Olah C, Satyanarayan A, Johnson I, Carter S, Schubert L, Ye K and Mordvintsev A 2018 The building blocks of interpretability *Distill* **3** e10
- [35] Bengio Y 2017 The consciousness prior (arXiv:1709.08568)
- [36] Jonschkowski R and Brock O 2015 Learning state representations with robotic priors *Auton. Robots* **39** 407
- [37] Fösel T, Tighineanu B, Weiss T and Marquardt F 2018 Reinforcement learning with neural networks for quantum feedback *Phys. Rev. X* **8** 031084
- [38] Venderley J, Khemani V and Kim E-A 2018 Machine learning out-of-equilibrium phases of matter *Phys. Rev. Lett.* **120** 257204
- [39] Fösel T, Niu M Y, Marquardt F and Li L 2021 Quantum circuit optimization with deep reinforcement learning (arXiv:2103.07585)
- [40] Krenn M, Häse F, Nigam A, Friederich P and Aspuru-Guzik A 2020 Self-referencing embedded strings (SELFIES): a 100% robust molecular string representation *Mach. Learn.: Sci. Technol.* **1** 045024
- [41] Andreas J, Dragan A and Klein D 2017 Translating neuralese *Proc. of the 55th Annual Meeting of the Association for Computational Linguistics* vol 232
- [42] Abadi M *et al* 2015 TensorFlow: large-scale machine learning on heterogeneous systems (available at: [www.tensorflow.org/](http://www.tensorflow.org/))
- [43] Paszke A *et al* 2017 Automatic differentiation in PyTorch *NIPS Autodiff Workshop*
- [44] Bahdanau D, Cho K and Bengio Y 2015 Neural machine translation by jointly learning to align and translate (arxiv:1409.0473)
- [45] Firat O, Cho K, Sankaran B, Yarman Vural F T and Bengio Y 2017 Multi-way, multilingual neural machine translation *Comput. Speech Lang.* **45** 236–52
- [46] Celikyilmaz A, Bosselut A, He X and Choi Y 2018 Deep communicating agents for abstractive summarization *Proc. of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* vol 1662
- [47] Scarselli F, Gori M, Tsoi A C, Hagenbuchner M and Monfardini G 2009 The graph neural network model *IEEE Trans. Neural Netw.* **20** 61
- [48] Kipf T, Fetaya E, Wang K-C, Welling M and Zemel R 2018 Neural relational inference for interacting systems *Proc. 35th Int. Conf. on Machine Learning* vol 80 p 2688
- [49] Battaglia P, Pascanu R, Lai M, Rezende D J and kavukcuoglu K 2016 Interaction networks for learning about objects, relations and physics *Proc. 30th Int. Conf. on Neural Information Processing Systems* pp 4509–17
- [50] Lesort T, Diaz-Rodriguez N, Goudou J-F and Filliat D 2018 State representation learning for control: an overview *Neural Netw.* **108** 379
- [51] Bengio E, Thomas V, Pineau J, Precup D and Bengio Y 2017 Independently controllable features (arXiv:1703.07718)
- [52] Jaderberg M *et al* 2017 Reinforcement learning with unsupervised auxiliary tasks *5th Int. Conf. on Learning Representations, ICLR 2017 (Toulon, France, 24–26 April 2017) (Conf. Track Proc.)*
- [53] Mnih V *et al* 2015 Human-level control through deep reinforcement learning *Nature* **518** 529
- [54] Zahavy T, Zrihem N B and Mannor S 2016 Graying the black box: understanding DQNs *Proc. 33rd Int. Conf. on Int. Conf. on Machine Learning* vol 48 p 1899
- [55] Wang C, Zhai H and You Y-Z 2019 Emergent Schrödinger equation in an introspective machine learning architecture *Sci. Bull.* **64** 1228

- [56] Carrasquilla J, Torlai G, Melko R G and Aolita L 2019 Reconstructing quantum states with generative models *Nat. Mach. Intell.* **1** 155
- [57] Neugebauer M, Fischer L, Jäger A, Czischek S, Jochim S, Weidemüller M and Gärtner M 2020 Neural-network quantum state tomography in a two-qubit experiment *Phys. Rev. A* **102** 042604
- [58] Carrasquilla J, Luo Di, Pérez F, Milsted A, Clark B K, Volkovs M and Aolita L 2021 Probabilistic simulation of quantum circuits using a deep-learning architecture *Phys. Rev. A* **104** 032610
- [59] Carleo G and Troyer M 2017 Solving the quantum many-body problem with artificial neural networks *Science* **355** 602
- [60] Glasser I, Pancotti N, August M, Rodríguez I D and Cirac J I 2018 Neural-network quantum states, string-bond states and chiral topological states *Phys. Rev. X* **8** 011006
- [61] Briegel H J and De las Cuevas G 2012 Projective simulation for artificial intelligence *Sci. Rep.* **2** 400
- [62] Poulsen Nautrup H, Delfosse N, Dunjko V, Briegel H J and Friis N 2019 Optimizing quantum error correction codes with reinforcement learning *Quantum* **3** 215
- [63] Wallnöfer J, Melnikov A A, Dür W and Briegel H J 2020 Machine learning for long-distance quantum communication *PRX Quantum* **1** 010301
- [64] Hangl S, Ugur E, Szedmak S and Piater J H 2016 Robotic playing for hierarchical complex skill learning *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS 2016 (Daejeon, Republic of Korea)* p 2799
- [65] Hangl S, Dunjko V, Briegel H and Piater J H 2017 Skill learning by autonomous robotic playing using active learning and creativity (arXiv:1706.08560)
- [66] Ried K, Müller T and Briegel H J 2019 Modelling collective motion based on the principle of agency: general framework and the case of marching locusts *PLoS One* **14** 1
- [67] Melnikov A A, Makmal A, Dunjko V and Briegel H J 2017 Projective simulation with generalization *Sci. Rep.* **7** 14430
- [68] Flamini F, Hamann A, Jerbi S, Trenkwalder L M, Nautrup H P and Briegel H J 2020 Photonic architecture for reinforcement learning *New J. Phys.* **22** 045002
- [69] Babaeizadeh M, Frosio I, Tyree S, Clemons J and Kautz J 2017 Reinforcement learning through asynchronous advantage actor-critic on a GPU *5th Int. Conf. on Learning Representations, ICLR 2017 (Toulon, France)*
- [70] Jerbi S, Poulsen Nautrup H, Trenkwalder L M, Briegel H and Dunjko V 2019 A framework for deep energy-based reinforcement learning with quantum speed-up (arXiv:1910.12760)
- [71] Erhard M, Fickler R, Krenn M and Zeilinger A 2018 Twisted photons: new quantum perspectives in high dimensions *Light Sci. Appl.* **7** 17146
- [72] Krenn M, Malik M, Fickler R, Lapkiewicz R and Zeilinger A 2016 Automated search for new quantum experiments *Phys. Rev. Lett.* **116** 090405
- [73] Krenn M, Hochrainer A, Lahiri M and Zeilinger A 2017 Entanglement by path identity *Phys. Rev. Lett.* **118** 080401
- [74] Krenn M, Gu X and Zeilinger A 2017 Quantum experiments and graphs: multiparty states as coherent superpositions of perfect matchings *Phys. Rev. Lett.* **119** 240403
- [75] Kaelbling L P, Littman M L and Cassandra A R 1998 Planning and acting in partially observable stochastic domains *Artif. Intell.* **101** 99
- [76] Boyajian W L, Clausen J, Trenkwalder L M, Dunjko V and Briegel H J 2020 On the convergence of projective-simulation-based reinforcement learning in Markov decision processes *Quantum Mach. Intell.* **2** 13
- [77] Melnikov A A, Makmal A and Briegel H J 2018 Benchmarking projective simulation in navigation problems *IEEE Access* **6** 64639
- [78] Tong D 2019 Lectures on dynamics and relativity (arXiv:1903.10563)