Discovering symmetry invariants and conserved quantities by interpreting siamese neural networks

Sebastian J. Wetzel,[1] Roger G. Melko,[1,2] Joseph Scott,[3] Maysum Panju,[4] and Vijay Ganesh[5]
[1]*Perimeter Institute for Theoretical Physics, Waterloo, Ontario, Canada N2L 2Y5*
[2]*Department of Physics and Astronomy, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1*
[3]*School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1*
[4]*Department of Statistics and Actuarial Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1*
[5]*Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1*

(Received 25 June 2020; accepted 27 August 2020; published 25 September 2020)

We introduce interpretable siamese neural networks (SNNs) for similarity detection to the field of theoretical physics. More precisely, we apply SNNs to events in special relativity, the transformation of electromagnetic fields, and the motion of particles in a central potential. In these examples, SNNs learn to identify data points belonging to the same event, field configuration, or trajectory of motion. We demonstrate that in the process of learning which data points belong to the same event or field configuration, these SNNs also learn the relevant symmetry invariants and conserved quantities. Such SNNs are highly interpretable, which enables us to reveal the symmetry invariants and conserved quantities without prior knowledge.

DOI: 10.1103/PhysRevResearch.2.033499

I. INTRODUCTION

Machine learning (ML) algorithms have experienced a surge in the physical sciences. This is based on the introduction of ML methods to fulfill tasks beyond the scope for which they were originally designed. These include finding phase transitions [1–12], simulating quantum systems [13–19], and rediscovering physical concepts [20–27].

Even though ML in theoretical physics is a young discipline, it has so far been successful in reproducing results in many complicated systems in just a few years. This success often comes at the cost of a lack of understanding of what ML algorithms intrinsically learn. Physics, as a scientific discipline, benefits from a "deeper understanding" of the underlying models used for making predictions.

The question of whether ML models can "understand" physics is a deeply philosophical one and we do not presume to address it in all its complexity. Assuming that a ML algorithm is successfully trained to predict the outcome of a physical experiment or calculation, it is not always clear whether the algorithm has deduced physical concepts or has merely managed to perform some basic pattern matching. However, if the ML model is "interpretable" in the sense that we can recover a compact and simple mathematical representation of a physical equation by analyzing the said model, then we take the position that such a model has indeed learned to understand the underlying physics.

The most successful ML algorithms are artificial neural networks (ANNs), which are famously inscrutable. Nevertheless, there have been many recent attempts at interpreting the learned features of a fully trained ANN. The simplest way to interpret a neural network is to examine the weights and biases of individual neurons, which can only yield successful results in shallow ANNs. In the field of explainable artificial intelligence, there are different methods that determine which features of the given input are responsible for a learned model's classification [28,29]. Similarly, in the field of computer vision, there have been many developments to examine the contribution of the pixels in an image to the ANN prediction [30–34]. One of the most popular methods is feature visualization by enhancing learned patterns on input images [35].

In physics, one has a distinct advantage when it comes to interpreting ANNs. In the field of computer vision or natural language processing, it is very hard to come up with mathematical equations uniquely describing the ground truth. In contrast, physicists have worked for hundreds of years to formulate their theories and experimental measurements in terms of mathematical equations. This means that if we can recover such an equation by analyzing an ANN, we have immediate access to its interpretation. This also opens up the possibility to check for consistency and reveal new concepts. Indeed, a few recent works have presented successful interpretations of ANNs in physics [22,36–39].

In this article, we propose a change in traditional siamese neural network (SNN) architectures that makes them easier to interpret. Specifically, the key feature is a bottleneck layer, where the SNN is forced to compress all available information from previous layers. The output of this bottleneck can be analyzed, for example, by applying known regression methods. A similar approach has been taken in [22]. While there does not exist a related interpretation procedure in computer

vision, the idea of interpreting bottleneck layers is also seen in disentangling autoencoders [40].

The ANNs we are considering in this work are a variant of the previously proposed SNNs, a class of ANNs that have been applied to object tracking, face recognition, and image similarity detection [41–44]. An SNN consists of two (identical) ANNs that are applied to a pair of input data points. The two networks share their weights and biases, which are updated simultaneously during training. The goal of the network is to map the input pairs to a set of latent variables that determine the similarity of the pair.

The general problem an SNN attempts to solve can be stated as follows: Given two data points x and y related by an equivalence relation (e.g., the same event in a relativistic setting measured by two observers in different reference frames), is it possible to correctly and automatically classify them as related? Further, if x and y are not related, then we require the ANN to classify them as not related.

Siamese neural networks can solve an extension of a classification problem with relatively little training data per class. Instead of training a traditional neural network to distinguish between a fixed number of classes, an SNN can probe the similarity of one data point with another prototypical data point for a certain class. This reformulation bears many advantages. First, the number of classes does not need to be fixed. Further, it is no longer necessary to train on all of the classes. A successfully trained SNN might be able to share its learned representation to distinguish between classes that are not in the training set. These properties become important in the limit of many (possibly infinitely many) classes or in the case where only a few data points are available in each class.

The following are the contributions we make in this paper.

(i) We introduce the SNN to the field of theoretical physics.

(ii) We demonstrate its usage in the well known contexts of special relativity, electromagnetism, and the motion of particles in a central potential. In the case of special relativity, these SNNs learn whether or not two different observations of physical phenomena correspond to the same event. In the case of electromagnetism, these SNNs learn whether or not given two field configurations, one can be transformed into the other via a Lorentz transformation. In the case of motion of particles, these SNNs discover whether or not two observations of position and momenta describe the same particle.

(iii) Further, we successfully interpret the intermediate output representations of the SNN and recover the mathematical formulations of known physical conserved quantities and invariants, e.g., the space-time interval or the angular momentum.

(iv) Since the interpretation of the SNN yields signatures of known physical equations, we argue that our SNN has indeed learned to understand physical concepts instead of merely performing basic pattern matching.

II. NEURAL NETWORK ARCHITECTURE

In this paper, we employ SNNs to determine whether or not two samples belong to the same class (see Fig. 1). In this context, our input data are a pair of samples $X_i = (x_i, x_i')$. In order to formulate a supervised learning problem, we associate the label $y_i = 0$ with pairs that correspond to the same class (i.e., x_i and x_i' are related via an equivalence relation) and $y_i = 1$ to pairs that belong to different classes (i.e., the input pairs are not related). In this sense, we can reformulate a classification problem with many (possibly infinitely many) classes into a traditional binary classification problem.

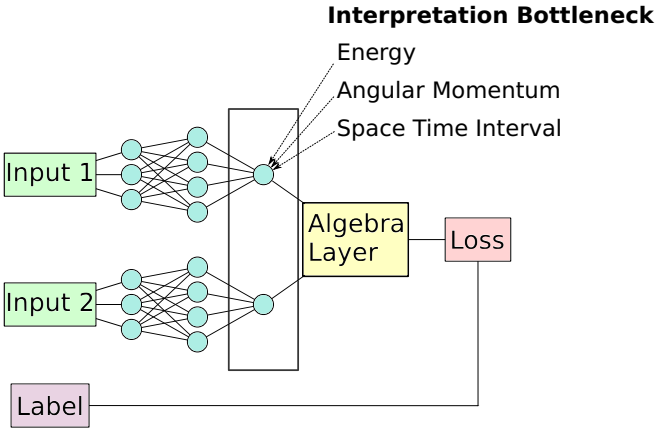

FIG. 1. Schematic architecture of an interpretable SNN. Our SNN contains a bottleneck of only a single neuron; the output of this layer is called the intermediate output of the network. We observe that this bottleneck encodes quantities which are strongly correlated with invariants such as the energy or the space-time interval.

For this purpose, we construct our SNN consisting of several building blocks. The first building block is composed of a pair of identical neural networks. This pair of networks is applied simultaneously to each of the samples in a data point pair x_i and x_i'. The last layer of the network pair only contains a single neuron; we refer to this layer as the bottleneck. The output of the bottleneck layer is the intermediate output of the SNN. The intermediate output is merged by performing appropriate algebraic operations. Let us denote by $f(x_i)$ and $f(x_i')$ the output of each of the neural networks. Then the algebra layer calculates $[f(x_i) - f(x_i')]^2$ before supplying it to a sigmoid neuron such that the output of the full SNN can be written as

$$F(X_i) = \text{sigmoid}\{w[f(\mathbf{x}_i) - f(\mathbf{x}_i')]^2 + b\}. \tag{1}$$

The SNN outputs a probability that signifies whether the two samples belong to the same class or not (see Fig. 2). For the purpose of training, we minimize the binary cross-entropy loss function between the SNN $F(X_i)$ and the label y_i on the training set. After training is complete, the generalization performance is measured on the test set. While training the SNN, we enforce weight sharing in the network pair to make sure these networks learn the same function. We note that, in the context of our physical examples, a natural minimization of the binary cross-entropy loss function is achieved if $f(x_i)$ learns a symmetry invariant or a conserved quantity.

After having successfully trained the SNN, our goal is to answer the question of which features this neural network bases its decision on. In general, there is no easy answer to this question, since analyzing even small neural networks can be extremely challenging. So far, there does not exist a comprehensive theory of what is learned by artificial neural networks.

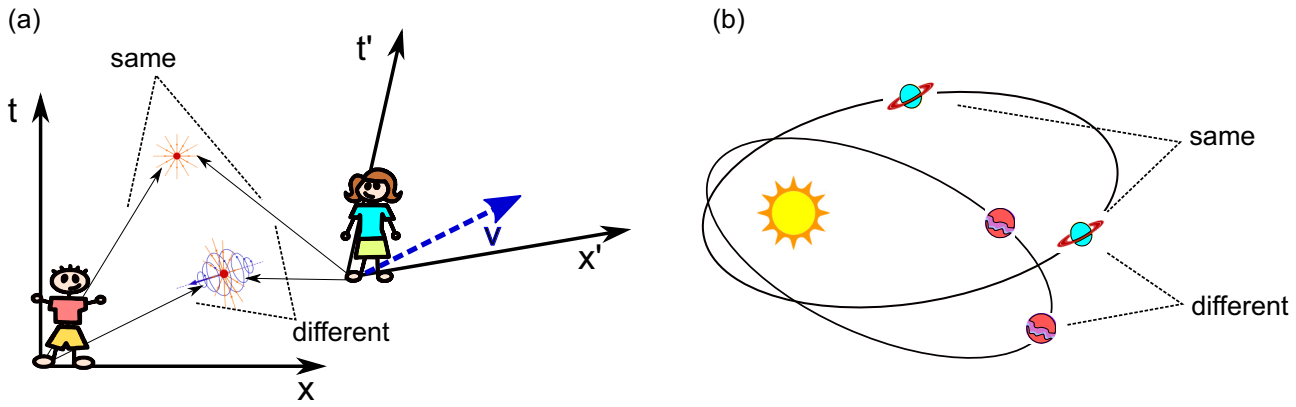

FIG. 2. Schematic description of the task solved by the siamese neural network. (a) In the case of special relativity and electromagnetism, our network is tasked to learn if two descriptions from different perspectives correspond to the same event or the same electromagnetic field configuration. (b) In the context of Newtonian gravity, we train our network to detect if two observations of velocities and positions correspond to the same particle moving in a central potential.

One of our crucial insights is that in order to interpret what our SNN learns, we have designed the SNN to include a bottleneck at the output of the first building block before merging (see Fig. 1). We will see later that our SNN learns conserved quantities and invariants at the bottleneck in order to make its decision about whether two samples belong to the same class. Further, by interpreting the network, we can predict conserved quantities and invariants with no additional prior knowledge.

If the number of neurons in the bottleneck layer increases, one can achieve better accuracy at the cost of interpretability. The interpretability can in principle be retained if one enforces decorrelated intermediate outputs. More details about our neural network architecture and the learning procedure can be found in Appendix B.

III. PERFORMING MACHINE LEARNING

A. Space-time in special relativity

1. Introduction

The first physical system we consider in this work is the Minkowski space-time in special relativity. An event is a four-vector $(t, x, y, z) \in \mathbb{R}^4$ that combines spatial coordinates and a moment in time. The Minkowski space-time is $\mathbb{R}^4$ with a scalar product induced by the metric $\eta_{\mu\nu} = \mathrm{diag}(-1, 1, 1, 1)$,

$$\langle \mathbf{x}, \mathbf{y} \rangle = \eta_{\mu\nu} x^\mu y^\nu = x_\mu y^\nu, \tag{2}$$

where we have used $x_\mu = \eta_{\mu\nu} x^\nu$. Thereby we define the space-time interval s by

$$\langle \mathbf{x}, \mathbf{x} \rangle = -t^2 + x^2 + y^2 + z^2 = s^2. \tag{3}$$

The Lorentz group is the set of transformations which preserve the scalar product on the Minkowski space-time

$$O(3, 1) = \{\Lambda \in \mathcal{M}(\mathbb{R}^4) : \langle \Lambda \mathbf{x}, \Lambda \mathbf{y} \rangle = \langle \mathbf{x}, \mathbf{y} \rangle \, \forall \, \mathbf{x}, \mathbf{y} \in \mathbb{R}^4\} \tag{4}$$

and thus also preserve the space-time interval.

2. SNN training

In this section, we discuss how to teach the neural network to identify, in special relativity, whether two observations by different observers correspond to the same event. These observers are at the same position but move with a relative velocity in some direction. For this purpose, we prepare positive training data of pairs of observations that correspond to the same event and negative data where a pair of measurements does not describe the same event.

More specifically, in order to train our neural networks with data we prepare a training data set consisting of pairs of measurements of the same event in Minkowski space-time seen from two different observers $X^\mu = (\mathbf{x}^\mu, \mathbf{x}'^\mu = \Lambda_\nu{}^\mu \mathbf{x}^\nu) = ((t, x, y, z), (t', x', y', z'))$. Here Λ is a random Lorentz transformation which is sampled from all possible Lorentz transformations. More details can be found in Appendix A. We sample 50 000 space-time events $\mathbf{x}^\mu$ and Lorentz transformations Λ to create pairs of events that form the positive data set. We associate with each pair the label $y = 0$. Further, we create a negative data set where each pair of space-time coordinates is not related by a Lorentz transformation. In practice, we implement this by randomly permuting among all second elements of all pairs of space-time events in the positive data set. Each pair in the negative data set is labeled with $y = 1$. In addition to this training set, we prepare a similar test data set of 5000 positive pairs and 5000 negative pairs.

The SNN is trained to predict if a pair of observations describe the same event or not. This is done by optimizing the weights of the neural network via backpropagation to minimize the binary cross-entropy loss between network output y_p and true label y_t. After training, the neural network is able to correctly predict if a pair of observations belong to the same event with an accuracy of approximately 94% on the training data set and approximately 92% on the test data set. The training and testing accuracies during training can be seen in Fig. 9.

Following the successful training of our SNN, we want to understand what the neural network has learned. This can be achieved by examining the intermediate output of the neural network, which acts as an interpretable bottleneck. We perform a hierarchy of linear regressions with polynomial

TABLE I. Regression scores of the regression on the intermediate output in the case of special relativity, which measures the normalized distance between regression and data. The score metric is known as the coefficient of determination or R^2 score. The best possible score is 1; the score can be negative.

Order	Train score	Test score
1	0.0013	0.0005
2	0.9894	0.9893
3	0.9900	0.9899
4	0.9907	0.9906

features (i.e., polynomial regression) on the intermediate output with respect to the input. If we assume that the Taylor expansion of the decision function is sufficiently accurate at the decision boundary, we can hope to get insightful results.

We perform ridge regression with polynomial features of the input on the intermediate output of the SNN. We start with polynomials of degree 1 and increase the degree of the polynomial features until the regression becomes accurate. From Table I one can immediately infer that the optimal degree of the polynomial features is 2.

The result of the regression in an ordered manner is

$$\begin{aligned} f(\mathbf{x}) \approx & -87.41t^2 - 60.48 - 0.11x \\ & - 0.10yz + 0.04ty + 0.06z \\ & + 0.07y + 0.10tx + 0.12tz \\ & + 0.15xz + 0.21xy + 2.50t \\ & + 88.10z^2 + 88.61y^2 + 88.63x^2 \\ \approx & 88\underbrace{(-t^2 + x^2 + y^2 + z^2)}_{=s^2} - 60. \end{aligned} \tag{5}$$

We can see that four nontrivial features dominate all others. If we assume that the regression includes small approximation errors, we can infer that the SNN has learned the invariant quantity $s^2 = -t^2 + x^2 + y^2 + z^2$. This quantity is the space-time interval, a known invariant of the Lorentz group. In cases where the regression does not yield a clear result, one can cross-check the second-order regression result with higher orders of regression and observe if the dominant features stay the same. Another option is to do the whole training procedure with a different random seed and see what parts of the results keep the same ratio.

To summarize, as long as the ANN is only able to use a single scalar function to decide if two events are the same, it calculates the space-time interval. If the space-time interval is the same, the ANN predicts that both coordinates in a pair belong to the same event. While it is often difficult to decide if neural networks learn to understand physical concepts to make decisions, here we argue that our SNN does so. To confirm our derivation, we draw a scatter plot for a subset of our data points of the intermediate output versus the space-time interval in Fig. 3 and observe a nearly perfect nonlinear correlation between these two. Note that we have cross-checked the second-order regression result with higher orders of regression and found that the dominant features stay the same.

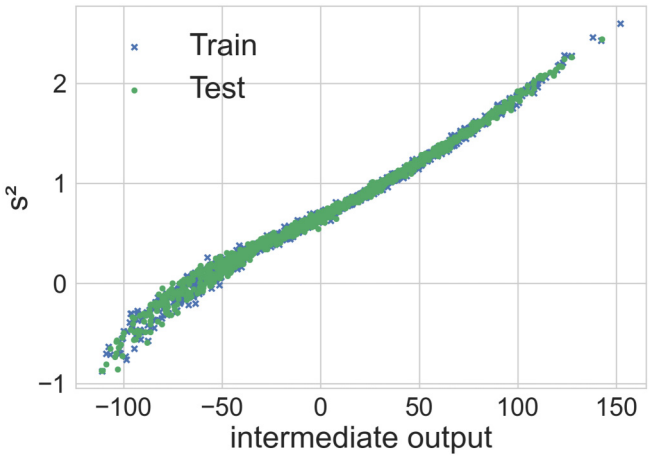

FIG. 3. Special relativity: correlation between the intermediate output of the siamese neural network at the bottleneck layer and the space-time interval.

Finally, we examine whether the SNN can also learn a different quantity to decide if two observations from different observers belong to the same event. For this purpose, we again prepare a training and a test data set, as explained above. However, in the preparation of the data set, we keep the space-time interval fixed. We attempt to train the SNN to learn to associate corresponding observations. However, the ANN fails to train in this case. After the best training cycle, the ANN can only predict if two observations belong to the same event with an accuracy of 58% on the training set or 57% on the testing set, which is barely better than random. This fact leads to the conclusion that the SNN is unable find another invariant of the Lorentz group besides the space-time interval.

The fact that the SNN fails to distinguish observations in this reduced data set hints that all observations with the same space-time interval can be transformed into each other by a Lorentz transformation. Further, it indicates that there is no other symmetry invariant. Both of these statements are of course known to be true. However, one needs to be careful since the same conclusions could be drawn if the neural network is not powerful enough to learn an underlying invariant.

B. Motion in a central potential

1. Introduction

As a second system we consider the motion of a particle in a central potential, such as the movement of a planet in the gravitational potential of the sun. Newtonian gravity can be formulated via the Hamiltonian

$$H = \frac{\mathbf{p}^2}{2m} - \frac{GmM}{r}. \tag{6}$$

Here $\mathbf{p}$ is the momentum, $r = \sqrt{x^2 + y^2}$ is the distance from the potential center, m is the mass of the planet, M is the mass of the sun, and G is Newton's constant of gravitation. Given an initial position $\mathbf{x}$ and velocity $\mathbf{v}$ one can calculate the trajectory of motion by solving Hamilton's equations

$$\begin{aligned} \dot{x} &= \partial_{p_x} H, \quad \dot{p}_x = -\partial_x H, \\ \dot{y} &= \partial_{p_y} H, \quad \dot{p}_y = -\partial_y H. \end{aligned} \tag{7}$$

TABLE II. Regression scores of the regression on the intermediate output in the case of the motion of a particle in a central potential.

Order	Train score	Test score
1	0.0003	−0.0003
2	0.9936	0.9939
3	0.9937	0.9940
4	0.9952	0.9858

There are conserved quantities in this system: the energy E, the components of the angular momentum $\mathbf{L}$, and the components of the Laplace-Runge-Lenz vector $\mathbf{A}$. They are related by two equations, which effectively reduces the number of scalar conserved quantities to five.

2. SNN training

When examining the motion of particles in a central potential, the SNN is tasked with determining whether two observations of the same particle correspond to the same particle trajectory.

We simulate particles of fixed mass m moving in a Newtonian static gravitational potential produced by a stationary mass M by solving the Hamilton equations for a set of random initial positions and velocities. For simplicity we set $m = 1$ and $GmM = 1$. We measure the position and the velocity of the particle at two different times to get pairs of inputs $X = (\mathbf{x}, \mathbf{x}') = ((x, y, v_x, v_y), (x', y', v_x', v_y'))$. We generate 50 000 pairs belonging to the same particle trajectories to form the positive training data set labeled by $y_i = 0$. By permuting the second entry in the pairs, we create a negative data set labeled with $y_i = 1$. Similarly, a testing set is produced with 5000 positive and 5000 negative examples.

The SNN is then trained to correctly predict if a pair of coordinates belong to the same trajectory. After being successfully trained, the network achieves an accuracy of approximately 98% on the training set and approximately 97% on the test set.

In order to interpret on what quantity the neural network bases its decision, we again examine the bottleneck at the intermediate output. We again perform a hierarchy of linear regressions with increasing polynomial features on the intermediate output. The optimal degree of the regression is 2 (see Table II).

The result of the regression in an ordered manner is

$$\begin{aligned} f(\mathbf{x}) \approx & -403.71xv_y - 4.85x - 0.58xy \\ & - 0.17xv_x - 0.02v_y^2 - 0.01v_xv_y \\ & + 0.00v_y^2 + 0.01v_y + 0.02v_x \\ & + 0.45x^2 + 0.66y^2 + 0.74 \\ & + 0.99yv_y + 1.24y + 402.44yv_x \\ \approx & -403\underbrace{(xv_y - yv_x)}_{=L_z}. \end{aligned} \tag{8}$$

This quantity is an approximation to the angular momentum $L_x = m(xv_y - yv_x)$. A confirmation of this result is visualized

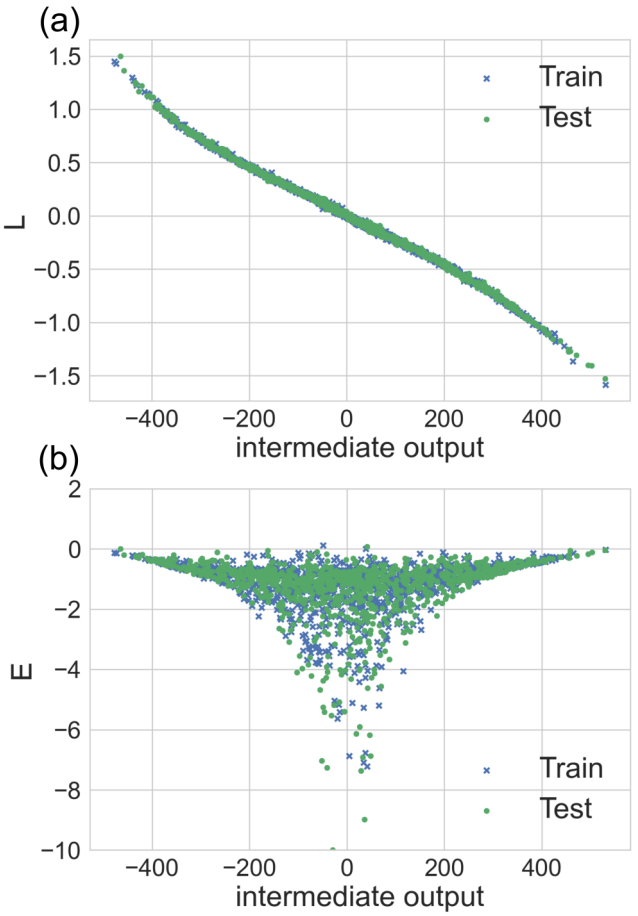

FIG. 4. Particle in a central potential: correlation between the intermediate output and (a) the angular momentum or (b) the energy.

in the very good correlation between the angular momentum and the intermediate output, illustrated in Fig. 4(a). This means that the SNN learns to distinguish between pairs originating from the same trajectory and different trajectories, by calculating the angular momentum. Another conserved quantity in this system is the energy. In Fig. 4(b) we see that the intermediate prediction is not correlated with the energy.

We now fix the angular momentum and perform the simulation again to produce 50 000 positive and 50 000 negative data pairs. We train the SNN again to distinguish if a pair of observations belong to the same trajectory. Even though the neural network cannot use the angular momentum to determine if the pair correspond to the same trajectory, it still manages to perform well on this task. The SNN achieves an accuracy of approximately 95% on both the training and the test set.

When using linear regression with polynomial features to determine what the SNN has learned in order to make its prediction we fail. On the one hand, there is no clear optimal degree of the polynomial regression. On the other hand, all the regression results do not yield a clear dominant feature. If we compare the intermediate output to the remaining invariants in the system, we find that the intermediate output is strongly correlated with the energy of the system (see Fig. 5). This means that the SNN probes the pair of observations for energy

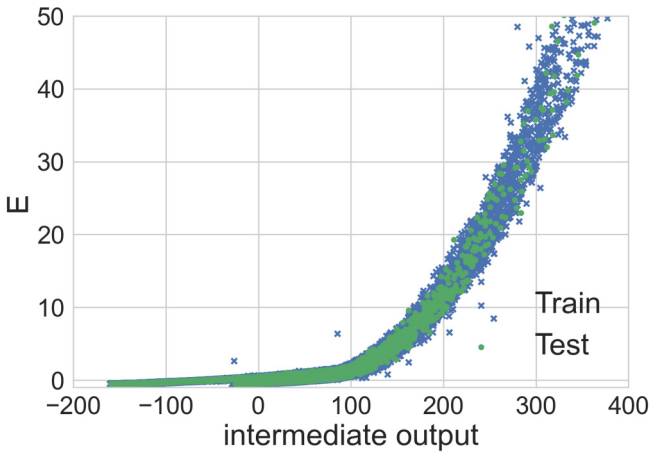

FIG. 5. Particle in a central potential with fixed angular momentum: correlation between the intermediate output and the energy.

conservation. However, the energy cannot be well approximated by a polynomial function with which we perform our regression.

To circumvent this problem we extend the input features to include the term $1/r = 1/\sqrt{x^2+y^2}$ such that the input feature vector reads $(x, y, v_x, v_y, 1/r)$. From this point we can start the polynomial regression, go on to identify the best polynomial order (see Table III), and formulate the findings in the equation

$$\begin{aligned} f(\mathbf{x}) \approx & -174.57 - 88.28\frac{1}{r} - 87.39 y v_x \\ & - 1.43\frac{1}{r^2} + \cdots + 1.27\frac{x}{r} \\ & + 46.22 v_x^2 + 46.53 v_y^2 + 87.18 x v_y \\ \approx & -175 + 87\underbrace{(x v_y - y v_x)}_{=L_z=\text{const}} + 90\underbrace{\left(\frac{1}{2}v_x^2 + \frac{1}{2}v_y^2 - \frac{1}{r}\right)}_{=E}. \end{aligned} \tag{9}$$

We see that the result includes the energy and the angular momentum. The angular momentum evaluates to a constant. Since constants can be absorbed, we conclude that the SNN has learned energy conservation. One might ask whether the SNN is able to find the Laplace-Runge-Lenz vector, which remains open for further investigation.

TABLE III. Regression scores of the regression on the intermediate output in the case of the motion of a particle in a central potential with fixed angular momentum.

Order	Train score	Test score
1	0.0019	0.0069
2	0.9145	0.9077
3	0.9258	0.7925
4	0.9498	−0.0359

TABLE IV. Regression scores of the regression on the intermediate output in the case of electromagnetism.

Order	Train score	Test score
1	0.0000	0.0000
2	0.9902	0.9902
3	0.9902	0.9902
4	0.9946	0.9946

C. Electromagnetism

1. Introduction

Finally, we consider electric **E** and magnetic fields **B** and their behavior under Lorentz transformations. For this purpose we incorporate the fields in the electromagnetic field strength tensor

$$F_{\mu\nu} = \begin{pmatrix} 0 & E_x & E_y & E_z \\ -E_x & 0 & -B_z & B_y \\ -E_y & B_z & 0 & -B_x \\ -E_z & -B_y & B_x & 0 \end{pmatrix}. \tag{10}$$

The Lorentz transformation of the field strength tensor

$$F'_{\mu\nu} = F_{\alpha\beta}\Lambda^{\alpha}{}_{\mu}\Lambda^{\beta}{}_{\nu} \tag{11}$$

implies the transformations for the electric and magnetic fields. The known Lorentz invariants of the electromagnetic fields are the determinant of the field strength tensor $\mathbf{B}\cdot\mathbf{E} = \det F$ and $|\mathbf{B}|^2 - |\mathbf{E}|^2 = 1/2 F_{\mu\nu}F^{\mu\nu}$.

2. SNN training

In this section, we study the behavior of electromagnetic fields under Lorentz transformations with SNNs. For this purpose, we again produce 200 000 true pairs $X = ((E_x, E_y, E_z, B_x, B_y, B_z), (E'_x, E'_y, E'_z, B'_x, B'_y, B'_z))$ of electric and magnetic field configurations, which are connected by a Lorentz transformation, and 200 000 negative pairs of fields by permuting the positive pairs.

We again train the SNN to predict if the two measurements belong to the same field configuration. After having successfully trained the neural network, we find that the neural network can fulfill the task to the high accuracy of approximately 95% on the training set and approximately 94% on the test set.

In order to determine what the neural network has learned, we perform polynomial regression on the intermediate output of the neural network. The function which approximates the output best is of degree 2 (see Table IV) and is given by

$$\begin{aligned} f(\mathbf{x}) \approx & -170.53 E_2 B_2 - 170.22 E_1 B_1 - 170.20 E_3 B_3 \\ & - 4.13 B_3^2 + \cdots + 4.92 E_2^2 + 53.43 \\ \approx & -170\underbrace{(E_1 B_1 + E_2 B_2 + E_3 B_3)}_{=E\cdot B} + 53. \end{aligned} \tag{12}$$

This function is an approximation to a known invariant, the determinant of the field strength tensor $\mathbf{B}\cdot\mathbf{E} = \det F$. A confirmation of this deduction is the correlation between $\det F$ and the intermediate output as depicted in Fig. 6.

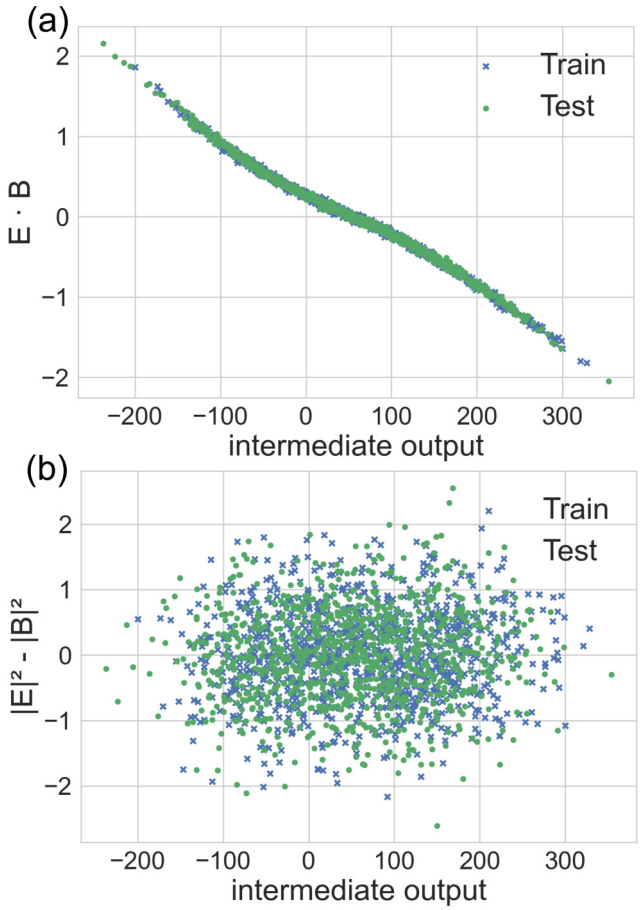

FIG. 6. Electromagnetism: correlation between the intermediate output and (a) the determinant of the field strength tensor and (b) a specific contraction of two field strength tensors.

Let us perform the same experiment again, however this time we fix the determinant of the field strength tensor when sampling the pairs of electromagnetic field configurations. The neural network still trains successfully and performs well in identifying pairs of data belonging to the same fields, with an accuracy of approximately 91% on the training set and approximately 90% on the test set. Performing the bottleneck regression on the intermediate output of the neural network reveals the remaining invariant to be of degree 2 (see Table V) and is approximated by

$$f(\mathbf{x}) \approx -216.26E_2^2 - 216.016E_1^2 - 215.59E_3^2 - 1.83E_1B_2 + \cdots + 5.55E_3B_3 + 13.59$$

TABLE V. Regression scores of the regression on the intermediate output in the case of electromagnetism with fixed determinant of the field strength tensor.

Order	Train score	Test score
1	0.0002	−0.0003
2	0.9956	0.9956
3	0.9957	0.9956
4	0.9962	0.9962

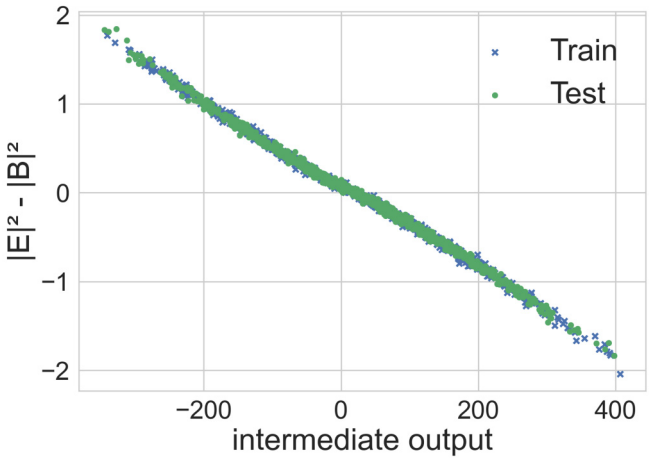

FIG. 7. Electromagnetism with fixed determinant of the field strength tensor: correlation between the intermediate output and a specific contraction of two field strength tensors.

$$+215.80B_3^2 + 216.57B_2^2 + 217.31B_1^2 \approx -216\underbrace{\left(E_1^2 + E_2^2 + E_3^2 - B_1^2 - B_2^2 - B_3^2\right)}_{=|E|^2-|B|^2} + 14. \quad (13)$$

This function is another known invariant of the field strength tensor $|\mathbf{B}|^2 - |\mathbf{E}|^2 = 1/2F_{\mu\nu}F^{\mu\nu}$, confirmed in Fig. 7. To summarize, in the context of electromagnetism, we have revealed the two invariants of the electric and magnetic fields which are preserved under Lorentz transformations.

IV. CONCLUSION AND FUTURE DIRECTIONS

We have introduced siamese neural networks to the field of theoretical physics. They are successful in predicting whether two data instances are connected by a deterministic transformation. We examined space-time events and electromagnetic fields which transform under Lorentz transformations, as well as the movement of particles in a central potential. By interpreting our neural network, we found that it learns the underlying symmetry invariants and conserved quantities to perform its prediction. Most interestingly, we were able to interpret our SNNs via the use of polynomial regression. This procedure revealed an excellent approximation of the underlying symmetry invariants and conserved quantities. These

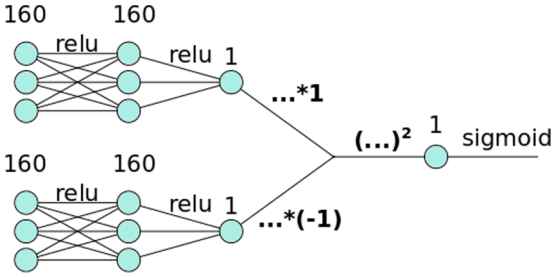

FIG. 8. Detailed architecture of an interpretable siamese neural network.

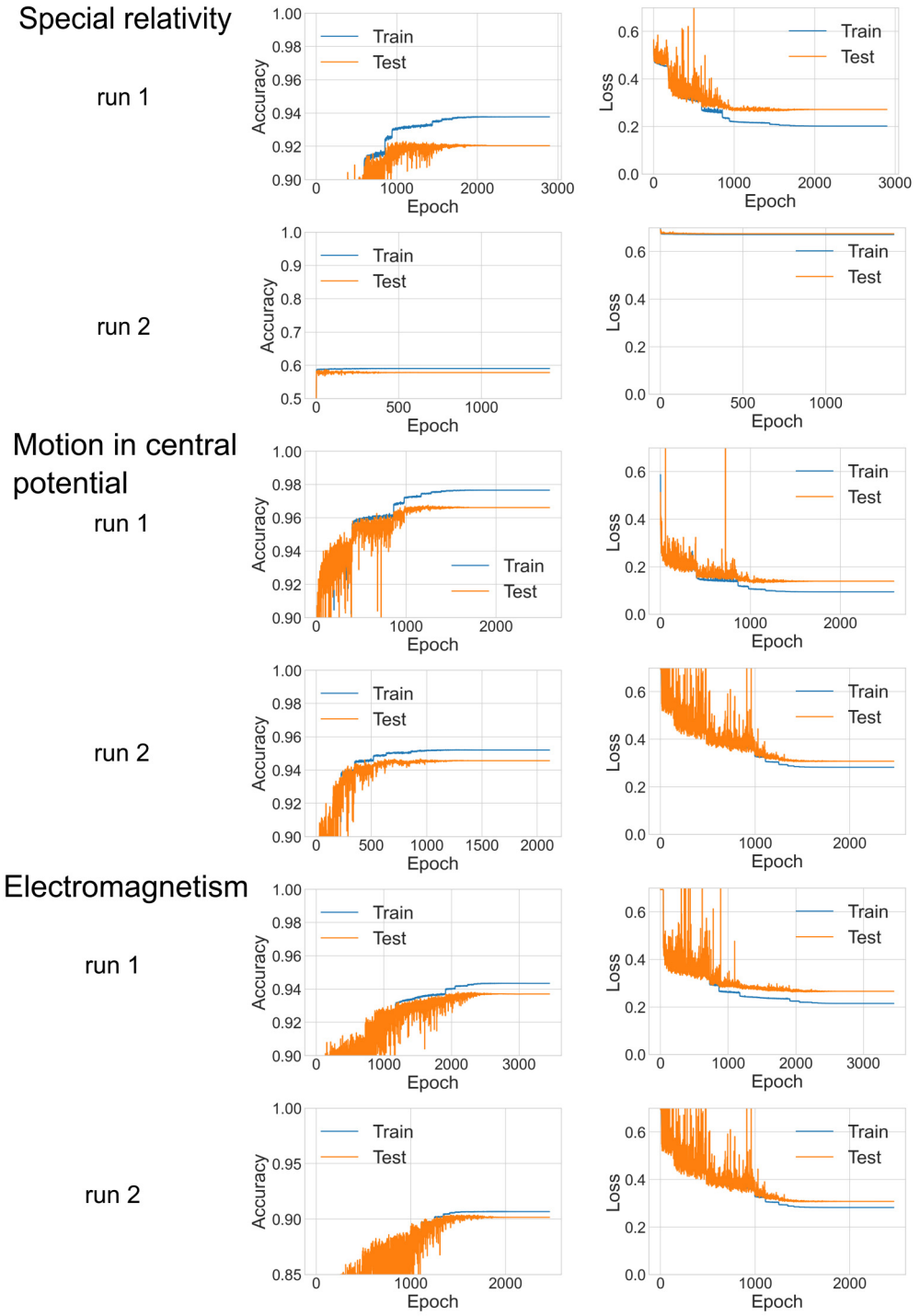

FIG. 9. Losses and accuracies.

invariants range from the space-time interval over angular momentum conservation to the determinant of the field strength tensor. If the underlying system does not contain human readable invariants, the neural network could act as an approximation to such an invariant.

Future directions of this work include an upgrade of the polynomial regression to symbolic regression [45]. Another exciting direction is to combine interpretable SNNs with semiautomated mathematical reasoning tools, e.g., solvers or theorem provers. The idea is to check the physical law learned by the SNN for consistency against known laws and invariants by leveraging such reasoning tools [46]. It does not take much imagination to envision how this technology can be used in applications such as quantum error correction or in particle tracking at the Large Hadron Collider.

It remains to be seen if SNNs will ever find an invariant or conserved quantity unknown to modern physics. Even if this does not happen, the contribution of this work is the introduction of SNNs as a useful tool in theoretical physics. Furthermore, we challenged the black box nature of artificial

neural networks by a very clear interpretation that reveals polynomial quantities without prior knowledge. The interpretation procedure might also be adopted into the field of computer science, where the interpretability of neural networks poses a major problem.

ACKNOWLEDGMENTS

We thank Isaac Tamblyn for helpful discussions. R.G.M. and J.S. were supported by NSERC. R.G.M. was further supported by the Canada Research Chair Program and the Perimeter Institute for Theoretical Physics. We thank the National Research Council of Canada for their partnership with Perimeter on the PIQuIL. Research at Perimeter Institute was supported in part by the Government of Canada through the Department of Innovation, Science and Economic Development Canada and by the Province of Ontario through the Ministry of Colleges and Universities.

APPENDIX A: LORENTZ TRANSFORMATION

Let us describe the representation of the Lorentz transformations which are used to generate the data pairs in the special relativity and electromagnetism sections. An arbitrary Lorentz transformation can be decomposed as

$$\Lambda = D_1 \Lambda_v D_2. \tag{A1}$$

Here Λ_v is a Lorentz boost in the x direction,

$$\Lambda_v = \begin{pmatrix} \gamma & -\gamma\beta & 0 & 0 \\ -\gamma\beta & \gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{A2}$$

where

$$\beta = \frac{v}{c}, \quad \gamma = \frac{1}{\sqrt{1-\beta^2}}. \tag{A3}$$

Here c is the speed of light, which we conveniently set to $c = 1$. The matrices D_1 and D_2 perform the rotation in the three-dimensional subspace

$$D = \begin{pmatrix} 1 & 0 \\ 0 & \mathcal{R} \end{pmatrix}, \tag{A4}$$

where $\mathcal{R} \in O(3)$.

APPENDIX B: NEURAL NETWORK DETAILS

In this Appendix we explain the details of the training of the SNN on pairs of data with a number of data points N between 50 000 and 200 000. For the sake of understandability we use the same architecture and hyperparameters for all learning tasks. The architecture of the SNN is depicted in Fig. 8.

The training of the neural network is the adjustment of the weights w_{ij}^L and biases b_i^L of the neural network to achieve a minimum of the binary cross-entropy loss function for all N training data points

$$L(y_t, y_p) = -\frac{1}{N}\sum_{i=0}^{N} y_{i,t}\ln(y_{i,p}) + (1-y_{i,t})\ln(1-y_{i,p}), \tag{B1}$$

where y_t denotes the true label, while y_p is the neural network prediction. Our neural networks are trained using the Adadelta optimizer. We found that starting learning rates of $\eta = 100$ are needed to train the neural network; this learning rate is much higher than normally used in traditional classification problems. Each update is performed by calculating the gradient on a batch of size 256. We employ learning rate decay callbacks which reduce the learning rate by a factor of 2 if the training loss has not improved for 50 epochs. We train our networks for 10 000 epochs; however, we employ an early stopping callback which aborts the training process if the training loss has not improved over 200 epochs. We do not use any kind of regularization in our neural networks. The evolution of the losses and accuracies during training are depicted in Fig. 9

APPENDIX C: INTERPRETING NEURAL NETWORKS

If a neural network intrinsically learns a physical observable $\mathcal{O}(x)$ as a function of the input data x, this observable is often encoded in an elusive manner distributed among many neurons. The bottleneck interpretation forces all information of this observable through a single neuron. In general, this observable is encoded in a deformed manner such that the output of the bottleneck neuron is $h(\mathcal{O}(x))$. If we restrict ourselves to a small output range, the function h can be linearized such that $h(\mathcal{O}(x)) = h_0 + h_1 \times \mathcal{O}(x)$. This form helps us to perform linear regression and h_0 and h_1 can be identified as adjustments to the weights and bias of the neuron.

[1] J. Carrasquilla and R. G. Melko, Nat. Phys. **13**, 431 (2017).
[2] E. P. L. van Nieuwenburg, Y.-H. Liu, and S. D. Huber, Nat. Phys. **13**, 435 (2017).
[3] L. Wang, Phys. Rev. B **94**, 195105 (2016).
[4] S. J. Wetzel, Phys. Rev. E **96**, 022140 (2017).
[5] Y. Zhang and E.-A. Kim, Phys. Rev. Lett. **118**, 216401 (2017).
[6] F. Schindler, N. Regnault, and T. Neupert, Phys. Rev. B **95**, 245134 (2017).
[7] W. Hu, R. R. P. Singh, and R. T. Scalettar, Phys. Rev. E **95**, 062122 (2017).
[8] T. Ohtsuki and T. Ohtsuki, J. Phys. Soc. Jpn. **86**, 044708 (2017).
[9] P. Broecker, J. Carrasquilla, R. G. Melko, and S. Trebst, Sci. Rep. **7**, 8823 (2017).
[10] D.-L. Deng, X. Li, and S. D. Sarma, Phys. Rev. B **96**, 195145 (2017).
[11] K. Ch'ng, J. Carrasquilla, R. G. Melko, and E. Khatami, Phys. Rev. X **7**, 031038 (2017).
[12] P. Huembeli, A. Dauphin, and P. Wittek, Phys. Rev. B **97**, 134109 (2018).
[13] G. Torlai and R. G. Melko, Phys. Rev. B **94**, 165134 (2016).
[14] G. Carleo and M. Troyer, Science **355**, 602 (2017).
[15] E. M. Inack, G. E. Santoro, L. Dell'Anna, and S. Pilati, Phys. Rev. B **98**, 235145 (2018).

[16] M. Hibat-Allah, M. Ganahl, L. E. Hayward, R. G. Melko, and J. Carrasquilla, Phys. Rev. Res. **2**, 023358 (2020).
[17] J. Carrasquilla, D. Luo, F. Pérez, A. Milsted, B. K. Clark, M. Volkovs, and L. Aolita, arXiv:1912.11052.
[18] F. Ferrari, F. Becca, and J. Carrasquilla, Phys. Rev. B **100**, 125131 (2019).
[19] O. Sharir, Y. Levine, N. Wies, G. Carleo, and A. Shashua, Phys. Rev. Lett. **124**, 020503 (2020).
[20] M. Schmidt and H. Lipson, Science **324**, 81 (2009).
[21] R. Iten, T. Metger, H. Wilming, L. del Rio, and R. Renner, Phys. Rev. Lett. **124**, 010508 (2020).
[22] S. J. Wetzel and M. Scherzer, Phys. Rev. B **96**, 184410 (2017).
[23] P. Ponte and R. G. Melko, Phys. Rev. B **96**, 205146 (2017).
[24] J. Greitemann, K. Liu, and L. Pollet, Phys. Rev. B **99**, 060404(R) (2019).
[25] Y.-i. Mototake, arXiv:2001.00111.
[26] S.-M. Udrescu and M. Tegmark, Sci. Adv. **6**, eaay2631 (2020).
[27] C. Wang, H. Zhai, and Y.-Z. You, Sci. Bull. **64**, 1228 (2019).
[28] S. M. Lundberg and S.-I. Lee, in *Proceedings of the 31st Conference on Neural Information Processing Systems, Long Beach, 2017*, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Curran, Red Hook, 2017), pp. 4765–4774.
[29] D. Gunning, XAI: Explainable Artificial Intelligence, DARPA report, 2017, available at https://www.darpa.mil/attachments/XAIProgramUpdate.pdf
[30] G. Montavon, W. Samek, and K.-R. Müller, Digital Signal Process. **73**, 1 (2018).
[31] M. T. Ribeiro, S. Singh, and C. Guestrin, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, New York, 2016).
[32] K. Simonyan, A. Vedaldi, and A. Zisserman, Workshop at the International Conference on Learning Representations, arXiv:1312.6034 (2014).
[33] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, PLoS One **10**, e0130140 (2015).
[34] M. D. Zeiler and R. Fergus, in *Computer Vision—ECCV 2014*, edited by D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Lecture Notes in Computer Science Vol. 8689 (Springer, Cham, 2014), pp. 818–833.
[35] A. Mordvintsev, DeepDream—A code example for visualizing neural networks, 2015, available at https://ai.googleblog.com/2015/07/deepdream-code-example-for-visualizing.html.
[36] D. Kim and D.-H. Kim, Phys. Rev. E **98**, 022138 (2018).
[37] P. Suchsland and S. Wessel, Phys. Rev. B **97**, 174435 (2018).
[38] Y. Zhang, P. Ginsparg, and E.-A. Kim, Phys. Rev. Res. **2**, 023283 (2020).
[39] S. Bluecher, L. Kades, J. M. Pawlowski, N. Strodthoff, and J. M. Urban, Phys. Rev. D **101**, 094507 (2020).
[40] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner, arXiv:1804.03599.
[41] J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, Y. Lecun, C. Moore, E. Säckinger, and R. Shah, Int. J. Pattern Recog. Artif. Intell. **07**, 669 (1993).
[42] S. Chopra, R. Hadsell, and Y. LeCun, in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)* (IEEE, Piscataway, 2005).
[43] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, *Proceedings of the 2014 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (IEEE, Piscataway, 2014).
[44] S. Appalaraju and V. Chaoji, arXiv:1709.08761.
[45] J. Koza, Stat. Comput. **4**, 87 (1994).
[46] J. Scott, M. Panju, and V. Ganesh, *Proceedings of the 34th AAAI Conference on Artificial Intelligence, New York, 2020* (AAAI, Palo Alto, 2020).