DPhil: Transfer of Status Literature Review

Shaan Desai

September 22, 2020

Contents

1 Introduction

Humans have long tried to understand how the mind works. From a biological perspective, connectomics has paved a pathway to understanding how we think [10]. Inspired by these ideas and the advent of computers and increased processing power, many researchers in the 50's began to ask whether the human mind can be replicated using a computer. Early work by Walter and Warren was one of the first to point out a means to build neural logic in computers that loosely resembles neurons in the brain [10]. Their seminal work opened up a new door to AI research and was later proven to hold universal function approximation properties [19]. Given neural networks are inspired by brain activity and consist of mathematical guarantees, AI researchers have focused intensely on developing them further. Arguably, they have been the most used method in the past decade having shown remarkable performance across a whole host of domains including image classification [17], machine translation [13] and robotic manipulation [42]. Despite these grand successes, neural networks have numerous limitations. Firstly, they are considered 'black box' models meaning they aren't readily interpretable. Secondly, they require a significant number of data points (> 5000 points) to learn and thirdly, they are highly sensitive to hyper-parameter tuning. Given such limitations, the physics community has been apprehensive about their adoption. However, over the past two years, significant research in *scientific machine learning* has created a new road to tackle the limitations of neural networks by embedding knowledge from physics into them. This prior knowledge has been shown to make neural networks more data efficient, interpretable and stable. In this review, we canvas state-of-the-art methods that incorporate physics into deep neural networks. For each method, we summarize the key insights that drive performance and also mention possible pain points worth investigating further.

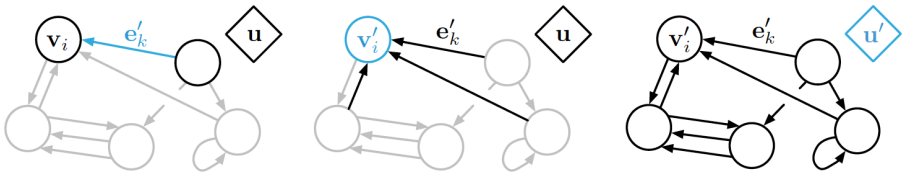
2 Inductive Biases

An inductive bias in machine learning is prior information used to guide model building. In simple linear regression, for example, we may assume the distribution of the noise in our data follows a Gaussian. This is a natural inductive bias because it imparts prior knowledge on the noise model of our data. In neural networks, architectures with variable depth, form and activations for example, serve as inductive biases. In essence, inductive biases allow us to encode initial assumptions about the complexity of our data.

A feedforward deep neural network naturally encodes non-linearity as a prior and arguably makes the least assumptions about the underlying data. Convolutional Neural Networks assume spatial representations can be captured. Recurrent Neural Networks assume temporal data as inputs [3].

In what follows, we highlight some of the recent advances in inductive biases, particularly as they relate to solving problems in physics.

2.1 Graph Neural Networks



(a) Edge update (b) Node update (c) Global update

Figure 1: Graph attribute updates as presented in [3]. GNNs carry out a sequence of computations and aggregations at an attribute level to determine the next set of values.

The state of a dynamic physical system can be represented by a graph G consisting of vertices V, edges E and global parameters u often denoted as $G = (u, V, E)$ [3]. For example, a node (V) can be used to represent a particle in an N-body problem. These nodes can be used to contain the core features of the particle e.g. position, momentum, mass and particle constants. Edges (E) can represent forces between the particles and 'Globals' (u) can represent constants such as air density, the gravitational constant etc. We note that graphs can be fully connected or sparse, depending on the specific use case. In representing physical systems this way, we impart structure on our data which forms an important inductive bias when learning physics [4, 3, 38, 39, 12, 40, 37, 24, 11]. Representations of this form allow us to carry out learning in multiple complex domains because such graphs impart a strong relational bias on the data. Graph Neural Networks are designed to translate graphs from one state to another. They work by computing sequential updates to node, edge and global parameters and aggregating them to define a new output graph. Graph Neural Networks are considered to impart a relational inductive bias, as convolutional networks are considered to impart a spatial bias and recurrent networks to impart a temporal bias.

Some major applications of GNNs include:

- Interaction-physics based problems naturally benefit from GNNs [3].
- GNNs trained to learn Hamiltonians achieve impressive results in rolling out trajectories of large N-body systems [36].
- GNNs have shown significant promise in explaining the phase transitions of glassy materials [2].
- GNNs can learn complex fluid like systems from visual data [37].

Bottom Line: Inspired by this work we see two major steps in moving this research forward. Firstly, their use in physics opens up a tried-and-tested pathway to solve more complex problems particularly in accounting for material

interactions. As such, we see good scope to use graph networks for large interacting systems as has been shown in [37]. Secondly, most systems to date have looked at graphs for classical physics, but literature from 2004 suggests graphs, inherent in their relational structure, can also capture Ising-like Hamiltonian structure [31]. In addition, although Hamiltonian Graph Networks show strong performance [3], not much work has been done to uncover the learned relations and whether they are consistent with ground truth interactions.

2.2 Integrative Biases

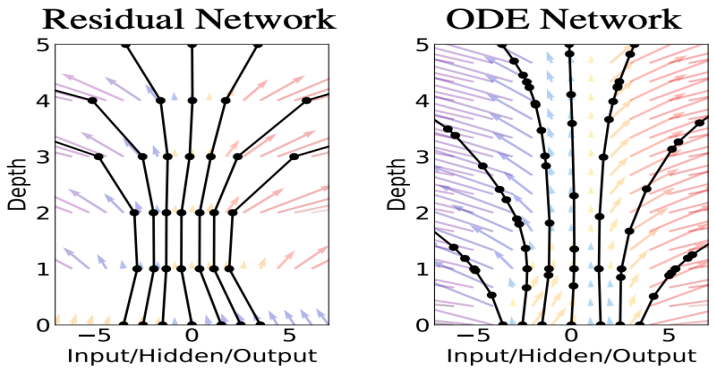


Figure 2: ODE Networks [7] show that in the continuous time limit, residual networks look like integrated neural networks.

NeuralODE brought to light a meaningful connection between residual networks and integrative steps [6, 7]. Residual networks are able to model complex functions by sequentially transforming a hidden state. Mathematically, this might look as follows:

$$h_{t+1} = h_t + f_\theta(h_t) \tag{1}$$

where h_t is a hidden layer of the network and $f_\theta(h_t)$ is a neural network parametrized by θ that operates on a hidden layer. By combining these discrete transformations over t, one can obtain more complex, non-linear functions that may be needed for downstream tasks [18].

If more steps are taken at shorter time intervals, in the continuous limit, equation 1 becomes:

$$\frac{dh(t)}{dt} = f(h(t), t, \theta) \tag{2}$$

Using this logic, we can parametrize any differential equation as a neural network and integrate it as long as we know its initial condition and final time step evaluation. However, one of the challenges of passing a neural network output

to an integrator is the fact that the integrator induces additional operations on the weights i.e. this process becomes a continuous depth neural network if the final time evaluation is large in comparison to the discrete time steps between layers. Neural ODEs via the adjoint method allow us to integrate this neural network with constant memory cost [7] using an ODESolver.

NeuralODE is used in numerous systems aimed at learning continuous time dynamics. In many settings a one-step integration is applied. For this, a complex ODESolver is replaced in favour of a simple explicit integrator e.g. Runge-Kutta 4. There are indeed settings where a multi-step integration is used, such as in [50] and in [35] for which continuous depth networks cause severe memory issues if NeuralODE is not used.

Despite their success, the work in [14] identifies the instability of using the adjoint method to compute continuous depth networks and propose a method to resolve this issue.

Bottom Line: Extensions of this work have been adopted across numerous methods we are about to discuss. Future work in this direction should look at identifying the right integrator of choice and establishing whether neural integrators can circumvent the need for an ODESolver.

2.3 Physics priors

Broadly, the intersection of physics and AI falls into one of two domains, physics for AI or AI for physics. The former uses techniques from physics to develop and improve learning algorithms in general. The latter uses existing learning approaches (with adaptations) to performance inference over physical systems, for example ML for materials [34, 41]. In this review, we focus our attention on the latter.

Physicists have long been interested in using learning tools to predict physics based systems. Some of these include predicting magnetic properties of 2-D materials [32], predicting the time evolution of N-body systems [3] and even using AI to understand phase transitions [2]. However, numerous challenges still remain in terms of 1. data-efficient learning, 2. reducing computational cost, 3. improving predictive accuracy and 4. learning better representations of the underlying physical process. Researchers have identified methods to address these challenges, but arguably the most promising hinges on physics-informed priors embedded in learning. It has been shown that models enriched with physically-informed priors i.e. models which consist of some knowledge about the physical system apriori, significantly outperform traditional methods in terms of data-efficiency and predictive accuracy. This has sparked a sharp interest in building both task-specific and general physics priors to improve learning. In this section, we summarize some of the core developments over time in physics informed inductive biases.

2.4 Gradient Learning

Although most modern methods cite gradient learning by Witkoskie [45] as one of the earliest efforts designed to improve learning of physics in neural networks, we actually find that a more sophisticate approach was developed prior to this effort.

In 1996, James Howse [20] presented a paper that highlights an approach to identify dynamical systems. The paper introduces a model inspired by defining a generalized functional form for dynamical systems. Using a potential $V(x)$ the authors show that we can partition an n-dimensional phase space into two components. The first space is normal to a level surface $V(x) = k$, and the second is tangent to $V(x) = k$. Systems that always move downhill are gradient like systems: $\dot{x} = -P(x)\nabla_x V(x)$. Systems that remain at constant potential are Hamiltonian like: $\dot{x} = Q(x)\nabla_x V(x)$. The paper shows that by casting a dynamic problem into these two components, one can learn the coefficients of the dynamical system.

In addition, the notion of embedding physically-informed inductive biases in neural networks can be found in numerous early work aimed at modelling materials [45, 29, 41, 34, 47]. For example, early efforts by Witkoskie and Doren [45] demonstrate that in contrast to directly learning a potential energy surface, the inclusion of gradient learning can drive a network to accurately model the forces. By gradient learning we mean that instead of a state vector being defined as x over which we want to optimize, our state vector becomes $[x, \dot{x}]$. This addition means that we can supplement the learning process with additional information and hence improve the learnt potential surface. The fundamental idea being that if we have access to supplemental data such as the gradients, but fewer data points, we might actually learn a surface with higher accuracy than if we had many data points with no gradient information.

Bottom Line: This result inspires us to look more closely at combining inductive biases. Namely, by adding well known priors together, we may make learning data-efficient and more accurate. In addition, we might want to look more closely at the results in [20] to see if we can develop a more generalizable approach to learning dynamics.

2.5 PINNs: Physics Informed Neural Networks

PINNs is one of the first methods to use the backpropagation technique from neural networks to actually compute the gradients of a function with respect to the inputs of that given function. The core idea in PINNs is to solve the following equation:

$$u_t + \mathcal{N}[u] = f = 0 \tag{3}$$

where $u(t, x)$ is a function that depends on time t and state-vector x and where

$\mathcal{N}$ is a non-linear differential operator. $u(t, x)$ denotes the latent hidden solution. To solve this equation, when only the initial conditions of u and specific collocation points are provided, PINNs compute u(t,x) with a neural network. Using backpropagation, they then compute partial derivatives with respect to the input variables t, x. The final results can thus be stored in f. A simple L2 loss to minimize the predicted u_{pred} vs. the ground truth u_{gt} (for initial and boundary conditions) coupled with a penalization of the function f (since it should be zero) at specific collocation points results in PINNs.

PINNs are used for both data-driven solutions and data-driven discovery. In other words, PINNs can be used to identify a system's governing equations, or given a set of equations, be used to continuously solve the system given an initial condition and points of evaluation.

One limitation to this approach is the need to know the functional form of $\mathcal{N}$. However, the authors of PINNs also developed deep hidden physics models [30] that are designed to learn the functional form without any prior knowledge of the function $\mathcal{N}$. The idea behind these deep hidden physics models is to compute u and then compute up to the k-th order derivatives of u. Using a regressive search, one can then identify the components that make up the underlying equation.

Bottom Line: PINNs are a versatile method that use backpropagation to compute both the loss term and the gradients of the function u w.r.t its inputs.

2.6 Energy Conserving Networks

As the section title suggest, energy conserving networks aim at conserving the energy of a given trajectory. The section looks carefully at methods that attempt to learn the Hamiltonian or Langrangian of a system.

2.6.1 Hamiltonian Neural Networks

[16] demonstrate that dynamic predictions through time can be improved using Hamiltonian Neural Networks (HNNs) which endow models with a Hamiltonian constraint. The Hamiltonian is an important representation of a dynamical system because it is one of two approaches that generalizes classical mechanics. The Hamiltonian $\mathcal{H}$ is a scalar function of position $\mathbf{q} = (q_1, q_2, ..., q_M)$ and momentum $\mathbf{p} = (p_1, p_2, ..., p_M)$. In representing physical systems with a Hamiltonian, one can simply extract the time derivatives of the inputs by differentiating the Hamiltonian with respect to its inputs (see Eqn. 4.)

$$\frac{\mathrm{d}\mathbf{q}}{\mathrm{d}t} = \frac{\partial \mathcal{H}}{\partial \mathbf{p}}, \quad \frac{\mathrm{d}\mathbf{p}}{\mathrm{d}t} = -\frac{\partial \mathcal{H}}{\partial \mathbf{q}} \tag{4}$$

As a consequence, it is noted in [16] that by accurately learning a Hamiltonian, the system's dynamics can be naturally extracted through backpropagation,

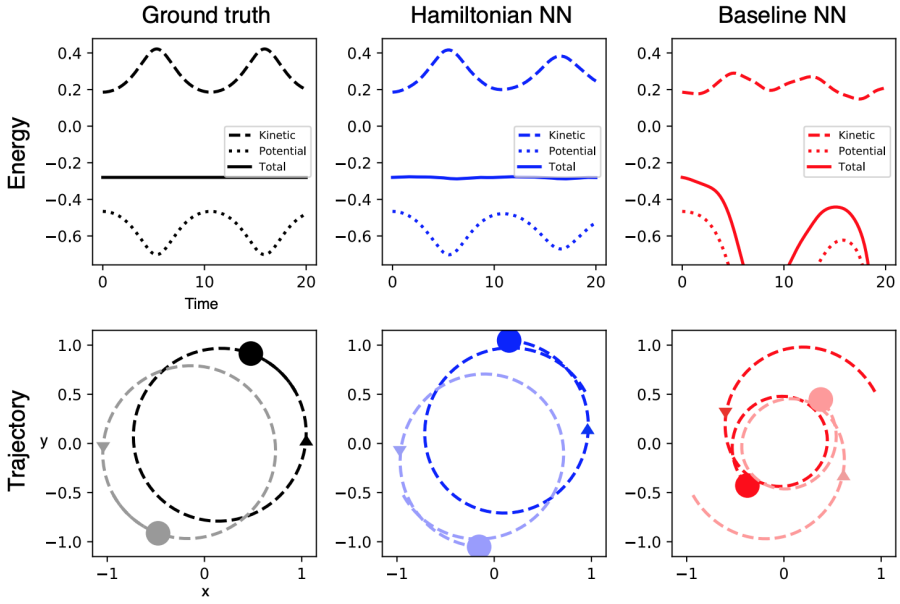


Figure 3: Trajectories of a 2-body system as predicted in [16].

similar to [30]. This information allows us to build two 1st-order differential equations which can be used to update the state space, $(\mathbf{q}, \mathbf{p})$. Equation 5 shows this integral, in which we define the symplectic gradient $\mathbf{S} = \left[\frac{\partial \mathcal{H}}{\partial \mathbf{p}}, -\frac{\partial \mathcal{H}}{\partial \mathbf{q}}\right]$:

$$(\mathbf{q}, \mathbf{p})_{t+1} = (\mathbf{q}, \mathbf{p})_t + \int_t^{t+1} \mathbf{S}(\mathbf{q}, \mathbf{p}) \mathrm{d}t \tag{5}$$

It can be shown that the Hamiltonian in many systems also represents the total energy of the system. Therefore, the Hamiltonian is a powerful inductive bias that can be utilised to evolve a physical state while maintaining energy conservation.

As we will discuss later, HNNs are great at learning in low-dimensional settings for a few integration steps. However, scaling this network to more challenging problems proves difficult. Fortunately, many alternatives have been proposed.

Bottom Line: One limitation we hope to investigate further in this domain is adding an energy penalization term between the predicted and ground truth energies to make learning more data-efficient. Our preliminary experiments find that penalizing the loss function with an energy term actually hurts the learning process indicating instability issues linked to HNNs.

2.6.2 Variational Integrator Networks

Lagrangian mechanics offers an alternative to the Hamiltonian in generalizing a dynamical system. Rather than position and momentum (canonical coordinates) defining the state space, Lagrangian mechanics is defined using a generalized coordinate state space $(\mathbf{q}, \dot{\mathbf{q}})$. This is particularly useful in physical settings where the description and measurement of generalized coordinates may be easier to work with than canonical coordinates [27]. Given these coordinates, Joseph-Louis Lagrange showed that a scalar value $\mathcal{A}$, referred to as the action, can be defined as the integral of a Lagrangian, $\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}})$:

$$\mathcal{A} = \int_{t}^{t+1} \mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}) \mathrm{d}t \tag{6}$$

The integral can be thought as inducing multiple paths between points in state space i.e. multiple walks in the domain of $(\mathbf{q}, \dot{\mathbf{q}})$. However, only one path is a stationary state of the action integral. This state lets us move from $t \rightarrow t+1$ with minimal energy. It can be shown, through variational calculus, that this stationary state must satisfy the Euler-Lagrange equation:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \mathcal{L}}{\partial \dot{\mathbf{q}}}\right) = \frac{\partial \mathcal{L}}{\partial \mathbf{q}} \tag{7}$$

Although complex in form, the action integral and the Euler-Lagrange equations can be discretized and collectively form the basis for variational integrators. The work in [35] shows that, by adopting this approach, one can develop VINs which make network learning in noisy data-settings more robust. Similar to Hamiltonians, Lagrangians in classical mechanics are also connected to the kinetic energy $\mathcal{T}$ and potential energy $\mathcal{V}$ via:

$$\mathcal{L} = \mathcal{T}(\mathbf{q}, \dot{\mathbf{q}}) - \mathcal{V}(\mathbf{q}, \dot{\mathbf{q}}) \tag{8}$$

Furthermore, Variational Integrators are symplectic and momentum conserving [25].

Bottom Line: The paper introduces the importance of symplectic integrators but does not discuss how to improve accuracy and scale up to higher dimensions. It is with this in mind that VIGNs was borne.

2.6.3 Deep Lagrangian Network

DeLAN net is the first network to use a Lagrangian embedded in a neural network to learn the dynamics of a system [26].

The paper defines the Langrangian $\mathcal{L}(q, \dot{q})$ and Euler-Lagrange equations to be:

$$\mathcal{L} = T - V \tag{9}$$

and

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = \tau \tag{10}$$

where τ represents generalized forces, T is kinetic energy and V is potential energy.

If we are dealing with a rigid body then: $T = \dot{q}^T M(q)\dot{q}$ where M is the inertia matrix. Replacing this equation in the Euler-Lagrange equation results in:

$$\frac{d}{dt}(M(q)\dot{q}) - \frac{\partial V}{\partial q} = 0; \quad M(q)\ddot{q} = \dot{M(q)}\dot{q} + \frac{\partial V}{\partial q} \tag{11}$$

To obtain $\ddot{q}$, the method proposes computing M, $\dot{M}$ and $\frac{\partial V}{\partial q}$ separately and then combining them.

Bottom Line: As a result of this framing, DeLAN net uses multiple network heads to compute individual components of the equation before combining them. [11] show that it is indeed possible to circumvent this problem. The important result here is that DeLAN can learn individual representations. If the non-conservative forces can be distinguished e.g. damping vs forcing, then DeLAN could be used to learn individual damping/driving forces from data which could be used on other systems.

2.6.4 Unsupervised Learning of Lagrangian Dynamics

In [51], the authors show that the full Lagrangian dynamics can be learned from visual data. The paper introduces a co-ordinate aware VAE to encode the latent space. Using the encoded latent space, a Lagrangian is learnt so that the time derivatives of the latent space can be extracted and used to integrate the latent space. The paper shows that the motion of a pendulum can be learnt from visual data. More importantly, via energy shaping, the pendulum can be constrained into a state-space $q*$ of choice.

Bottom Line: This result is quite powerful because it empirically proves that a controlled Lagrangian system can be learned from visual data. This inspires us to ask the question of whether a chaotic system such as heinon-heiles or the double-pendulum can be learned from visual data.

2.6.5 Lagrangian Neural Networks

The main premise of LNNs [11] is to tackle the problem of dealing with canonical coordinate spaces. Many datasets do not usually consist of canonical position and momentum, rather they use generalized coordinates. As such, LNNs aim to address learning from generalized coordinates. In addition, they provide a

more general framework than DeLANs which were designed to work well with continuous control applications. The unique innovation that LNNs introduce over other methods is that they do not assume any form for the Lagrangian. As such, they vectorized the Euler-Lagrange:

$$\frac{d}{dt}\nabla_{\dot{q}}\mathcal{L} = \nabla_q \mathcal{L} \tag{12}$$

Then, using the chain rule to expand the time derivative by allowing $\nabla_{\dot{q}}\mathcal{L}$ to be a function of q and $\dot{q}$ they obtain:

$$(\nabla_{\dot{q}}\nabla_{\dot{q}}^T\mathcal{L})\ddot{q} + (\nabla_q\nabla_{\dot{q}}^T\mathcal{L})\dot{q} = \nabla_q\mathcal{L} \tag{13}$$

Using matrix inversion, one can obtain $\ddot{q}$.

Bottom Line: The paper shows promising results on the double pendulum, relativistic particle in a uniform potential and on the wave equation. Upon inspection of the training scheme for the highly sensitive double pendulum, we do find that training times and data points are quite large, creating a space to find a more optimal learning scheme.

2.6.6 Modeling System Dynamics with PINNs on Lagrangian Mechanics

Unlike LNNs, this paper introduces a functional form for the Lagrangian. The Lagrangian in this paper [33] is assumed to be $L = T - V$. In addition, they introduce non-conservative forces and the final form is:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = Q^{ncons}$$

The approach taken by this paper is to feed in the respective components e.g. q,$\dot{q}$ into separate neural network heads designed to predict one of M, C and G. Using this technique, $\ddot{q}$ is backed out and then fed to an RK-4 integrator. In other words, the approach is a simplified version of DeLANs.

Bottom Line: The approach is tethered to a more traditional way of using NNs for predictions and is one of the reasons why model performance is not as good as it can be if backpropagation was used to compute the second derivative of q. In addition, the wrong choice of integrator is a bottleneck for long range predictions in this setting.

2.7 Symplectic Networks

Many researchers have identified that embedding integrators into the learning process of dynamic systems alleviates the need for the derivatives $[\dot{q}, \dot{p}]$ of the state vector $[q, p]$. That is, with an integrator embedded, the state-vector is

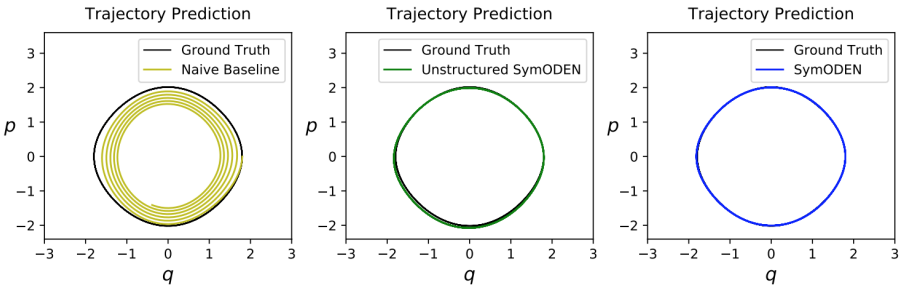


Figure 4: Symplectic Integrators from [50] show that their capacity at predicting long range trajectories preserve phase-space volume

enough to compute the dynamics. However, as [52] and a number of other researchers highlight, the choice of integrator can play a significant role in long range predictions. In principle, integrators that preserve symplecticity show significantly better long range results because they enforce phase-space volume conservation.

2.7.1 Symplectic ODE Net

Symp-ODEN [50] extends HNNs into the control domain. If external control is affine and influences the change in generalized momenta then the state-space derivatives can be written as:

$$\begin{bmatrix} \dot{q} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} \frac{\partial H}{\partial p} \\ -\frac{\partial H}{\partial q} \end{bmatrix} + \begin{bmatrix} 0 \\ g(q) \end{bmatrix} u \tag{14}$$

where $g(q)u$ represents a control variable. If rank(g(q)) is rank(q) the system is fully actuated. For such systems, a controller $u = \beta(q) + v(p)$ can be designed to change the potential energy landscape so as to force a system toward a specific configuration.

Arguably this is the first paper to think extensively about the bottlenecks presented in HNNs including control, energy shaping and symplecticity.

2.7.2 Symplectic Recurrent Neural Network

Symplectic RNN [8] makes two major contributions to the physics neural network space. Firstly, they take Hamiltonian Neural Networks and convert the integration scheme from euler to a leapfrog. Secondly, they adopt a NeuralODE style integrator to integrate across multiple timesteps from an initial condition. However, the results in this paper overlap greatly with Symp-ODEN.

2.7.3 Deep Hamiltonian Networks based on symplectic integrators

This paper [52] reviews the integrator of choice for HNNs. One of its primary objectives is to establish the difference in performance between using a symplectic vs non-symplectic integrator from a theoretical perspective. Quite evidently, using a symplectic integrator allows us to conserve the phase-space volume of the system - a crucial component in preserving energy. Their results reiterate the need for symplectic integrators when dealing with Hamiltonian-like systems.

2.7.4 SympNets

A brand new class of methods is proposed in [22]. It avoids the need for a separable Hamiltonian and more importantly, is designed to eliminate the need for backpropagating the Hamiltonian with respect to the input. In this framework, a sequence of symplectic maps are used to transform the input into the output. The symplectic map can be split into an upper triangular matrix and lower triangular matrix with diagonals set to 1. Non-diagonal terms are parametrized by a neural network and by stacking a range of symplectic maps together, one can learn dynamics better. In fact, the results show that by doing this, the learned trajectory is significantly more accurate than HNNs with symplectic integrators.

Bottom Line: The introduction of SympNets takes us in a direction tangential to the concept of backpropagating a function with respect to its input. This might be more flexible in the long run since it avoids complicating backpropagations.

2.8 Gaussian-Based Physics Networks

2.8.1 Learning Constrained Dynamics with Gauss' Principle adhering Gaussian Processes

[15] introduces a GP model that utilises mechanical constraints as prior knowledge for learning dynamics of systems. The GP is constrained to satisfy Gauss' Principle. Using Udwadia Kalaba Equation, the acceleration of a particle can be disentangled into an unconstrained acceleration, an ideal constraint and a non-ideal constraint. Using this knowledge, one can feed this structural form into the mean of a gaussian process regression. The process can be used to set priors on unknown parameters linked to the constraints.

The proposal is important because it tackles constraint optimisation of physics based networks using GPs which can give us explicit uncertainty bounds on our learned trajectories.

2.8.2 Bayesian Hidden Physics Models

In [1], the author presents a novel approach to tackle learning from noisy data. The paper is arguably the first method to combine Gaussian Processes with physics informed neural networks in a unified manner.

Problem Statement

Given, x, t and some initial conditions, learn the governing equation $\mathcal{N}$ of a dynamic system that obeys:

$$u_t + \mathcal{N}[u] = 0$$

The learning takes place in 3 stages. The first is, given inputs x and t, compute $u(x,t)$ using a neural network. The loss of this predicted term is computed using a log likelihood s.t.:

$$L_i^u = logp(D_i^u|\theta_i^u) = \sum_{j=1}^{n_{st}} logp(\hat{u}_j^i|u(x_j,t_j;\theta_i)$$

where D is the dataset, θ are the network parameters and $\hat{u}$ is the ground truth function.

Then, using the computed u, define a prior over the derivatives of u e.g. $u_t, u_x, u_x x$. The purpose of doing this is for data-driven discovery. In other words, rather than computing k partial derivatives of u w.r.t the input x, we place a GP prior on the k partial derivatives. The prior is assumed to be Gaussian and is of the form:

$$L^f = p(\hat{u_t}|V) = \mathcal{N}(\hat{u}_t|\mu(V), K_f + \sigma_f^2 I)$$

As such, one can obtain the posterior using variational inference since we do not know the distribution of L^f.

The main breakthrough of using this approach is uncertainty quantification of the learned operators as well as the convergence in v-space the method illustrates.

2.9 Data Driven Model Discovery

2.9.1 Discovering Physical Concepts with Neural Networks

Although this review has looked closely at embedding physical laws into neural networks, some approaches such as [21] take on a different approach of not assuming any prior. The paper sets the 'scientific process' as the prior. By feeding inputs into an encoder to learn a representation and then querying this

representation before decoding, the experimental process of learning laws is designed to become a part of the network.

The only caveat to this modelling is the use of disentangled VAEs that are designed to produce orthogonal representations. The assumption here being that every new variable added to the representation should be independent and alter the output landscape separately i.e. no linear combinations/the variables should span the output space.

Results illustrate that such a system can learn accurate latent representations necessary to evolve the time dynamics of a system, including conservation laws. The network also learns how to switch co-ordinate systems and the dimension of underlying quantum systems.

This presents an alternative view to learning physical systems. Motivated by this result, one can easily draw a comparison across methods to illustrate how much more data/time is needed to establish the right representation in comparison to methods naturally designed to embed physical laws.

2.9.2 Discovering Governing equations from data by sparse identification of nonlinear dynamical systems

The proposed method in [5] tackles learning the governing equations of a physical system by inspecting the sparsity of a function f that satisfies:

$$dx/dt = f(x(t))$$

For most systems, the paper identifies that only a small group of derivatives are relevant to compute f.

The method first computes higher order polynomials of the inputs and then uses sparse regression to select the parameters.

This approach of 'selecting' from a bag of potential functions has been explore extensively in determining governing equations. However, the method relies on having to explicitly build a large matrix of permutations.

2.9.3 Symbolic Pregression

Fundamental early work aimed at devising unique ways to find regression coefficients using a supervised learning framework. Symbolic pregression combines this approach with that of the hamiltonian neural net to build an unsupervised framework for discovery of differential equations [43]. The paper still does not reveal the functional form of the underlying constraint but the results look deeply promising in dealing with visual data.

2.10 Physics Informed Generative Adversarial Networks

GANs have been shown to emulate complex physical concepts such as turbulent flows. In addition, by embedding physical constraints into the generator, GANs can be used to sample deterministic physical constraints. In [46], the authors define a general physical constraints s.t.:

$$H[u] \leq 0$$

where H is a differential operator. Then, by adding this constraint to the generator, we get:

$$V_C(D, G) = V(D, G) + \lambda C_{phys}$$

where:

$$C_{phys} = \mathbb{E}_Z(Max(H(G(Z)), 0))$$

The authors show that such a system can be used to generate samples given specific constraints e.g. generate samples on circle of radii 3. However, they also show that approximate samples can be drawn for which constraints might be inequalities.

Extending this work to build physics informed images might be particularly useful in animation.

2.11 Physics Informed Neural Network Applications

2.11.1 Chaos

Using Hamiltonian Neural Networks, [9] shows that it is possible to predict the trajectory of a chaotic heinon-heiles system. In addition, they are able to predict trajectories of billiard balls undergoing complex potential functions. [11] also work to show that the double pendulum system can be learned from data but their approach shows significant practical limitations relating to the complexity of the model used.

Our preliminary idea was to see if HNNs can be used to learn the dynamics of a double pendulum. However, our preliminary results show that high energy systems are quite difficult to learn. However, this line of work can establish new ways of developing neural networks that are more sensitive to their inputs.

2.11.2 Materials

In numerous papers, [34, 45, 29, 41, 47] the use of physics priors have been used to model materials. Some have taken a more traditional ML approach

to predict materials properties such as [34], while others have looked at using deep networks such as [44] in which convolutional networks are used to extract meaningful Hamiltonian representations of magnetic materials.

2.12 Causality

Bayesian networks are probabilistic graphical models that can represent conditional dependencies using a Directed Acyclic Graph. The main logic behind these methods is that a probabilistic approach to determining a marginal distribution can be linked to a graph structure if certain conditional dependencies are met.

Learning DAGs from data is an NP-hard problem, but the NOTEARS algorithm [49] presents a way to tackle this combinatorial search by converting the problem into a continuous optimization problem. The main contribution of the paper is a penalization of the weight matrix so that it is acyclic. From studies of graphs we know that the n'th power of an adjacency matrix gives us the lengths of walk k between two nodes. As such, all we need to do is enforce that all the diagonals of the powers of the adjacency matrices are set to zero.

In other words:

$$tr(I - B)^{-1} = d$$

must be satisfied. Embedding this into the learning process is imparting an inductive bias on learning.

It was with this in mind that DYNOTEARS [28] were able to extend the work into the temporal setting and learn dynamic bayesian networks. In principle, by adding an additional weight matrix to learning and penalizing it in the same way that NOTEARS does.

Furthermore, it has been shown that the NOTEARS constraint can be embedded in graph neural networks [23, 48].

References

[1] S. Atkinson. Bayesian Hidden Physics Models: Uncertainty Quantification for Discovery of Nonlinear Partial Differential Operators from Data. *arXiv:2006.04228 [cs, stat]*, June 2020. arXiv: 2006.04228.

[2] V. Bapst, T. Keck, A. Grabska-Barwińska, C. Donner, E. D. Cubuk, S. S. Schoenholz, A. Obika, A. W. R. Nelson, T. Back, D. Hassabis, and P. Kohli. Unveiling the predictive power of static structure in glassy systems. *Nature Physics*, 16(4):448–454, Apr. 2020. Number: 4 Publisher: Nature Publishing Group.

[3] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. Dahl, A. Vaswani, K. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu. Relational inductive biases, deep learning, and graph networks. *arXiv:1806.01261 [cs, stat]*, Oct. 2018. arXiv: 1806.01261.

[4] P. W. Battaglia, R. Pascanu, M. Lai, D. Rezende, and K. Kavukcuoglu. Interaction Networks for Learning about Objects, Relations and Physics. *arXiv:1612.00222 [cs]*, Dec. 2016. arXiv: 1612.00222.

[5] S. L. Brunton, J. L. Proctor, and J. N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, Apr. 2016.

[6] B. Chang, L. Meng, E. Haber, L. Ruthotto, D. Begert, and E. Holtham. Reversible Architectures for Arbitrarily Deep Residual Neural Networks. *arXiv:1709.03698 [cs, stat]*, Nov. 2017. arXiv: 1709.03698.

[7] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural Ordinary Differential Equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 6571–6583. Curran Associates, Inc., 2018.

[8] Z. Chen, J. Zhang, M. Arjovsky, and L. Bottou. Symplectic Recurrent Neural Networks. *arXiv:1909.13334 [cs, stat]*, Apr. 2020. arXiv: 1909.13334.

[9] A. Choudhary, J. F. Lindner, E. G. Holliday, S. T. Miller, S. Sinha, and W. L. Ditto. Physics enhanced neural networks predict order and chaos. *arXiv:1912.01958 [physics]*, Nov. 2019. arXiv: 1912.01958.

[10] J. D. Cowan. Neural Networks: The Early Days. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 828–842. Morgan-Kaufmann, 1990.

[11] M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, and S. Ho. Lagrangian Neural Networks. *arXiv:2003.04630 [physics, stat]*, Mar. 2020. arXiv: 2003.04630.

[12] M. D. Cranmer, R. Xu, P. Battaglia, and S. Ho. Learning Symbolic Physics with Graph Networks. *arXiv:1909.05862 [astro-ph, physics:physics, stat]*, Nov. 2019. arXiv: 1909.05862.

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]*, May 2019. arXiv: 1810.04805.

[14] E. Dupont, A. Doucet, and Y. W. Teh. Augmented Neural ODEs. *arXiv:1904.01681 [cs, stat]*, Oct. 2019. arXiv: 1904.01681.

[15] A. R. Geist and S. Trimpe. Learning Constrained Dynamics with Gauss Principle adhering Gaussian Processes. *arXiv:2004.11238 [cs, eess, stat]*, Apr. 2020. arXiv: 2004.11238.

[16] S. Greydanus, M. Dzamba, and J. Yosinski. Hamiltonian Neural Networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d. Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 15379–15389. Curran Associates, Inc., 2019.

[17] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. *arXiv:1703.06870 [cs]*, Jan. 2018. arXiv: 1703.06870.

[18] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. *arXiv:1512.03385 [cs]*, Dec. 2015. arXiv: 1512.03385.

[19] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, Jan. 1989.

[20] J. W. Howse, C. T. Abdallah, and G. L. Heileman. Gradient and Hamiltonian Dynamics Applied to Learning in Neural Networks. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 274–280. MIT Press, 1996.

[21] R. Iten, T. Metger, H. Wilming, L. del Rio, and R. Renner. Discovering physical concepts with neural networks. *Physical Review Letters*, 124(1):010508, Jan. 2020. arXiv: 1807.10300.

[22] P. Jin, Z. Zhang, A. Zhu, Y. Tang, and G. E. Karniadakis. SympNets: Intrinsic structure-preserving symplectic networks for identifying Hamiltonian systems. *Neural Networks*, 132:166–179, Dec. 2020.

[23] S. Lachapelle, P. Brouillard, T. Deleu, and S. Lacoste-Julien. GRADIENT-BASED NEURAL DAG LEARNING. page 23, 2020.

[24] L. Lamb, A. Garcez, M. Gori, M. Prates, P. Avelar, and M. Vardi. Graph Neural Networks Meet Neural-Symbolic Computing: A Survey and Perspective. *arXiv:2003.00330 [cs]*, Mar. 2020. arXiv: 2003.00330.

[25] A. Lew, J. E. Marsden, M. Ortiz, and M. West. AN OVERVIEW OF VARIATIONAL INTEGRATORS. *Finite Element Methods*, page 18.

[26] M. Lutter, C. Ritter, and J. Peters. Deep Lagrangian Networks: Using Physics as Model Prior for Deep Learning. *arXiv:1907.04490 [cs, eess, stat]*, July 2019. arXiv: 1907.04490.

[27] J. E. Marsden and M. West. Discrete mechanics and variational integrators. *Acta Numerica*, 10:357–514, May 2001.

[28] R. Pamfil, N. Sriwattanaworachai, S. Desai, P. Pilgerstorfer, P. Beaumont, K. Georgatzis, and B. Aragam. DYNOTEARS: Structure Learning from Time-Series Data. *arXiv:2002.00498 [cs, stat]*, Apr. 2020. arXiv: 2002.00498.

[29] A. Pukrittayakamee, M. Malshe, M. Hagan, L. M. Raff, R. Narulkar, S. Bukkapatnum, and R. Komanduri. Simultaneous fitting of a potential-energy surface and its corresponding force fields using feedforward neural networks. *The Journal of Chemical Physics*, 130(13):134101, Apr. 2009.

[30] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, Feb. 2019.

[31] I. Rezek and S. J. Roberts. An Operator Interpretation of Message Passing. page 8.

[32] T. D. Rhone, W. Chen, S. Desai, A. Yacoby, and E. Kaxiras. Data-driven studies of magnetic two-dimensional materials. *arXiv:1806.07989 [cond-mat]*, June 2018. arXiv: 1806.07989.

[33] M. A. Roehrl, T. A. Runkler, V. Brandtstetter, M. Tokic, and S. Obermayer. Modeling System Dynamics with Physics-Informed Neural Networks Based on Lagrangian Mechanics. *arXiv:2005.14617 [cs, stat]*, May 2020. arXiv: 2005.14617.

[34] M. Rupp, A. Tkatchenko, K.-R. Müller, and O. A. von Lilienfeld. Fast and Accurate Modeling of Molecular Atomization Energies with Machine Learning. *Physical Review Letters*, 108(5):058301, Jan. 2012. Publisher: American Physical Society.

[35] S. Saemundsson, A. Terenin, K. Hofmann, and M. P. Deisenroth. Variational Integrator Networks for Physically Meaningful Embeddings. *arXiv:1910.09349 [cs, stat]*, Oct. 2019. arXiv: 1910.09349.

[36] A. Sanchez-Gonzalez, V. Bapst, K. Cranmer, and P. Battaglia. Hamiltonian Graph Networks with ODE Integrators. *arXiv:1909.12790 [physics]*, Sept. 2019. arXiv: 1909.12790.

[37] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. W. Battaglia. Learning to Simulate Complex Physics with Graph Networks. *arXiv:2002.09405 [physics, stat]*, Feb. 2020. arXiv: 2002.09405.

[38] A. Sanchez-Gonzalez, N. Heess, J. T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell, and P. Battaglia. Graph networks as learnable physics engines for inference and control. *arXiv:1806.01242 [cs, stat]*, June 2018. arXiv: 1806.01242.

[39] S. Seo and Y. Liu. Differentiable Physics-informed Graph Networks. *arXiv:1902.02950 [cs, stat]*, Feb. 2019. arXiv: 1902.02950.

[40] S. Seo, C. Meng, and Y. Liu. PHYSICS-AWARE DIFFERENCE GRAPH NETWORKS FOR SPARSELY-OBSERVED DYNAMICS. page 15, 2020.

[41] J. S. Smith, O. Isayev, and A. E. Roitberg. ANI-1: an extensible neural network potential with DFT accuracy at force field computational cost. *Chemical Science*, 8(4):3192–3203, 2017.

[42] M. Toussaint, K. Allen, K. Smith, and J. Tenenbaum. Differentiable Physics and Stable Modes for Tool-Use and Manipulation Planning. In *Robotics: Science and Systems XIV*. Robotics: Science and Systems Foundation, June 2018.

[43] S.-M. Udrescu and M. Tegmark. Symbolic Pregression: Discovering Physical Laws from Raw Distorted Video. *arXiv:2005.11212 [physics, stat]*, May 2020. arXiv: 2005.11212.

[44] J. Wang, S. Olsson, C. Wehmeyer, A. Pérez, N. E. Charron, G. de Fabritiis, F. Noé, and C. Clementi. Machine Learning of Coarse-Grained Molecular Dynamics Force Fields. *ACS Central Science*, 5(5):755–767, May 2019. Publisher: American Chemical Society.

[45] J. B. Witkoskie and D. J. Doren. Neural Network Models of Potential Energy Surfaces: Prototypical Examples. *Journal of Chemical Theory and Computation*, 1(1):14–23, Jan. 2005.

[46] Z. Yang, J.-L. Wu, and H. Xiao. Enforcing Deterministic Constraints on Generative Adversarial Networks for Emulating Physical Systems. *arXiv:1911.06671 [physics, stat]*, Nov. 2019. arXiv: 1911.06671 version: 1.

[47] K. Yao, J. E. Herr, D. Toth, R. Mckintyre, and J. Parkhill. The TensorMol-0.1 model chemistry: a neural network augmented with long-range physics. *Chemical Science*, 9(8):2261–2269, 2018.

[48] Y. Yu, J. Chen, T. Gao, and M. Yu. DAG-GNN: DAG Structure Learning with Graph Neural Networks. page 10.

[49] X. Zheng, B. Aragam, P. Ravikumar, and E. P. Xing. DAGs with NO TEARS: Continuous Optimization for Structure Learning. *arXiv:1803.01422 [cs, stat]*, Nov. 2018. arXiv: 1803.01422.

[50] Y. D. Zhong, B. Dey, and A. Chakraborty. Symplectic ODE-Net: Learning Hamiltonian Dynamics with Control. *arXiv:1909.12077 [physics, stat]*, Sept. 2019. arXiv: 1909.12077.

[51] Y. D. Zhong and N. E. Leonard. Unsupervised Learning of Lagrangian Dynamics from Images for Prediction and Control. *arXiv:2007.01926 [cs, eess, stat]*, July 2020. arXiv: 2007.01926.

[52] A. Zhu, P. Jin, and Y. Tang. Deep Hamiltonian networks based on symplectic integrators. *arXiv:2004.13830 [cs, math]*, Apr. 2020. arXiv: 2004.13830.