DPhil: Thesis

Shaan Desai

July 19, 2020

Contents

1 Literature Review (WIP)

1.1 Artificial Intelligence

- Humans have long tried to understand how the mind works - from a biological perspective it is clear that connectomics paves a pathway to understanding how we think - With the advent of computers and 'large' processing power, many researchers began to ask whether the human mind can be replicated using a computer. - Early work by walter and warren was the first to point out a means to build neural network logic. - Followed by Hebbian - Then an AI winter - Geof Hinton backprop - slew of advents in neural processing, convnets, rnns, graphs, tensor nets, deep RL, - all seeking to solve more tasks/increase generalizability. - to date, there has been no general purpose solution (AGI) for a vast array of tasks e.g. a human can learn to create art, sit math exams, play sports, yet a robot so far can only do one of these tasks well. - Numerous limitations in terms of processing power and memory access still exist

- Despite these limitations, neural networks have brought about significant advancements to society in robotics, self driving cars, cancer detection.

- In what follows, we outline some of the major advances, those which have been fundamental pillars to research conducted during this phd. We follow no particular order.

1.2 Inductive Biases

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

A feedforward deep neural network naturally encodes 'nonlinearity' 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.

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

1.3 Graph Neural Networks

Graph Neural Networks are neural networks which use graphs as inputs. As such, they can be used to understand and learn the relationships which exist between the nodes of a graph. More formally, a graph is a tuple (Vertices,Edges,Globals). These are attributes of the graph where the nodes contain information, edges contain relationship weights and globals consist of macro variables general to all graphs.

interaction-physics based problems naturally benefit from graphs. We see that HOGNs and OGNs achieve great results in rolling out trajectories of N body systems.

Although they account for a new set of relational inductive biases, graphs still involve complex message passing schemes that involve significant fine-tuning. In addition, they assume data streams are already structured. However, graphs have shown significant promise in physics e.g. in explaining the phase transition of glassy materials. Their use in physics opens up a tried-and-tested pathway to solve more complex problems particularly in accounting for material interactions.

They're even shown to model complex fluid like systems from visual data.

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.

1.4 Integrative Biases

- ODEnet - new paper by tegmark

1.5 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, for example xx. The latter uses existing learning approaches (with adaptations) to predict physics, for example ML for materials. 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, predicting the time evolution of N-body systems and even using AI to understand phase transitions. However, numerous challenges still remain in terms of data-efficient learning, reducing computational cost, improving predictive accuracy and 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.

Gradient Learning

Although most modern methods cite gradient learning(next section) by Witkoskie as one of the earliest efforts designed to improve learning of physics in neural networks, we actually find that more sophisticated approaches were developed prior to this effort.

In 1996 James Howse/NIPS presented a paper which highlights a few things on identifying dynamical systems:

- define ML as model selection and parameter estimation
- paper introduces a model inspired by defining a generalized functional form for dynamical systems
- using a potential V(x) we can partition an n-dimensional phase space
- first space is normal to some level surface $V(x) = k$, second is tangent to $V(x) = k$.
- systems which always move downhill are gradient like systems: $\dot{x} = -P(x)\nabla_x V(x)$

- systems which remain at constant potential are hamiltonian like: $\dot{x} = Q(x)\nabla_x V(x)$
- can combine these two to get total dynamics.
- define a loss function inspired by this model
- use the learnt parameters to then evolve a system

The entire framework is a parameter fitting one because we have a sense for the prior functional form of the dynamics.

The notion of embedding physically-informed inductive biases in neural networks can be found in numerous early work aimed at modelling materials [**?**, **?**, **?**, **?**, **?**]. For example, early efforts by Witkoskie and Doren [**?**] 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.

Therefore, the loss function takes the form:

$$\left\| \begin{bmatrix} \hat{y} \\ \hat{\dot{y}} \end{bmatrix} - \begin{bmatrix} y \\ \dot{y} \end{bmatrix} \right\|_2^2$$

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.

Hamiltonian Neural Networks

ing neural networks to accurately learn classical dynamics from data problems has been Hamiltonian Neural Networks [**?**]. In demonstrated 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. 1.)

$$\frac{\mathrm{d}\mathbf{q}}{\mathrm{d}t} = \frac{\partial \mathcal{H}}{\partial \mathbf{p}}, \quad \frac{\mathrm{d}\mathbf{p}}{\mathrm{d}t} = -\frac{\partial \mathcal{H}}{\partial \mathbf{q}} \tag{1}$$

As a consequence, it is noted in [**?**] that by accurately learning a Hamiltonian, the system's dynamics can be naturally extracted through backpropagation. 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 2 shows this integral, in which we define the symplectic gradient $\mathbf{S} = \left[\frac{\partial \mathcal{H}}{\partial \mathbf{p}}, -\frac{\partial \mathcal{H}}{\partial \mathbf{q}}\right]$:

$$(\mathbf{q}, \mathbf{p})_{t+1} = (\mathbf{q}, \mathbf{p})_t + \int_t^{t+1} \mathbf{S}(\mathbf{q}, \mathbf{p}) \mathrm{d}t \tag{2}$$

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.

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 [**?**]. 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, $\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}})$:

$$\mathcal{A} = \int_t^{t+1} \mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}) \mathrm{d}t \tag{3}$$

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:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \mathcal{L}}{\partial \dot{\mathbf{q}}}\right) = \frac{\partial \mathcal{L}}{\partial \mathbf{q}} \tag{4}$$

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 [**?**] 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:

$$\mathcal{L} = \mathcal{T}(\mathbf{q}, \dot{\mathbf{q}}) - \mathcal{V}(\mathbf{q}, \dot{\mathbf{q}}) \tag{5}$$

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

symplectic ODE Net

The paper extends hamiltonian neural networks into the control domain. If external control is affine and influences the change in generalized momenta then:

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

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.

If the hamiltonian is a newtonian then -

$$\beta(q) = g^T(gg^T)^{-1}(\partial V/\partial q - \partial V_d/\partial q)$$

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. https://arxiv.org/pdf/2005.11212.pdf

deep energy net

neurips review for steve

A new method is introduced to tackle the challenge of learning energy conserving physics more broadly. The method claims to be more general as it can learn from PDEs as well as ODEs, and it avoids discretisation errors that arise from using RK integrators. The method does this by a) introducing a general formalism which highlights how the rate of change of states can be cast into a matrix G times the gradient of a Hamiltonian and b) by discretising the gradient of the hamiltonian with a frechet derivative. The paper is well written and detailed.

$$\dot{u} = G\nabla H$$

Positives

Can solve a broader class of problems compared to current solutions in the field by modelling an additional G matrix e.g. Friction systems (ODE and PDE) Discrete PDEs (PDE) Maintains energy and mass conservation via discretisation which preserves the geometric structure (a.k.a volume preservation) Good form for discrete auto differentiation Great PDE dataset motivations Can extend research to identify discrete PDEs Cons

No mention of momentum conservation No study (at least not explicit) of higher order systems e.g. all the data descriptions talk about single precision (does this mean Euler/RK2??) what happens if we have data that has/requires higher precision? Novelty in the math - G matrix is mentioned in other papers ODE systems are not appropriately benchmarked: Dissipative systems are

talked about but SympODEN dissipative is not benchmarked * although dissipative SympODEN is technically new and not required to be benchmarked Precision is talked about in hamiltonian systems but VIN isn't benchmarked Not readily extensible to N particle systems (could plug VIGNs/graph based networks for them to cite) Computational overheard (2X slower than HNN which could suffer in large data/state space domains 'links to prev. bullet) A mention of how control might be incorporated via G would be interesting

2 Notes

A section to document some of my learnings over time.

2.1 GANs for Physics (Oct '19 - Jan '20)

Preliminary Ideas

Can generative networks capture the laws of physics?

1. Generative networks are able to learn complex mappings from data to latent spaces
2. Often, these latent spaces are able to perfectly generate the original data which might indicate that the fundamental laws of physics are being captured by the network
3. This leads us to the question - can we use a generative network to learn dynamics of a particle e.g. in the 2 body problem, and observe whether the generator learns the underlying equations of motion.

Key point: if i can generate the data, I must understand something about the laws of physics.

Inspired by greydanus, we wondered if a GAN-like approach can be taken to model and understand the phase-space.

GAN Theory

What is a Generative Adversarial Network?

Intuition is that there is an art forger (G) and art critique (D) who are tasked with making and evaluating art respectively.

1. $\hat{x} = G(z)$ where z is noise and $\hat{x}$ has to match the distribution of $p(x)$.
2. $D(x)$ is a mapping into a probability. Discriminates between real and fake art.

They are adversaries, as such, they have different cost functions to optimize. This induces Nash equilibria:

It is a game between 2 players in which the equlibria is a local minima for each loss. This means the generator draws samples perfectly from $p(x)$ and the discriminator cannot discriminate so predicts 1/2 for all samples.

The discriminator loss follows that of a log bernoulli distribution or the cross entropy loss function.

$$H(p, q) = -\sum_i p_i \log q_i$$

For binary classification we get:

$$H((x_1, y_1), D) = -y_1 \log D(x_1) - (1 - y_1) \log(1 - D(x_1))$$

If we sum over each value in the dataset, we then obtain:

$$H((x_i, y_i)_i^N, D) = -\sum_{i=1}^{N} y_i \log D(x_i) - (1 - y_i) \log(1 - D(x_i))$$

Now ideally, we like to take this and make some changes.

1. we want data to be split evenly, so if the total dataset size is N. We set N/2 of the samples to be class 1 and N/2 to be class 0. This automatically reduces the above y variables to 1/2's.

2. The x values come from 2 sources. The legitimate source and the sampled source. We want this to be probabalistic so sums become expectations.

The discriminator needs to minimze the above loss. It is the same as the cross-ent loss for a NN doing binary classification with sigmoid output.

If we optimize this loss holding G constant, we get:

$$D(x) = \frac{p_{data}}{p_{model} + p_{data}}$$

This is nash equilbrium.

Zero-sum game means any loss is conserved between entities.

The zero sum game solution is minimax where we want to minimize the maximum loss. The discriminator wants a higher payoff so tries to maximize while the generator wants a lower loss so minimizes. To optimize this, an interative approach is used.

The sources used to understand these concepts come from ucla tutorial and Takeshi github link.

Training GANs

To start we train a GAN using MNIST (Adam optimizers and saw convergence). We attempted GAN with pendulum motion and saw multiple convergence issues. We notice some challenges with training GANs - generator usually diverges while discriminator rapidly converges. Many tips online i.e. make target

values 0.9 instead of 1 to reduce certainty.Using 'hacks' only can see marginal improvement. Larger issue of GANs being unstable.

Furthermore, do we actually want to use GANs for the application? Seems that the only way to constrain the noise input is to encode it which essentially gives us an autoencoder.

VIN network is a good idea though we have no code base/architecture used.

Attempted to use AE-GAN to encode physics and also build a discriminator: challenge at the moment is discriminator loss is increasing (rare) and predicted output images look nothing like the ground truth.

Some ideas:

1. Changed optimizers
2. Changed network architecture
3. Removed discriminator and still noticed issues
4. Even when architecture is consistent with HNN
5. Perhaps the only difference is torch.randperm and torch shuffle They use shuffling of data with replacement and don't use epochs but steps
6. Takens theorem here

Not necessarily evident what the benefits are of having the AE-GAN from the simple autoencoder network/ perhaps better images but this might not improve the latent space we learn.

An alternative approach is to sample the latent space in a way which is consistent with the laws of physics (i.e. sample p and q) sequentially, as we sample the ground truth images - then we can see how p,q can 'map' into a 2D image (might be potential to learn from the hidden layers) but this is fully supervised and we are not learning anything immediately useful.

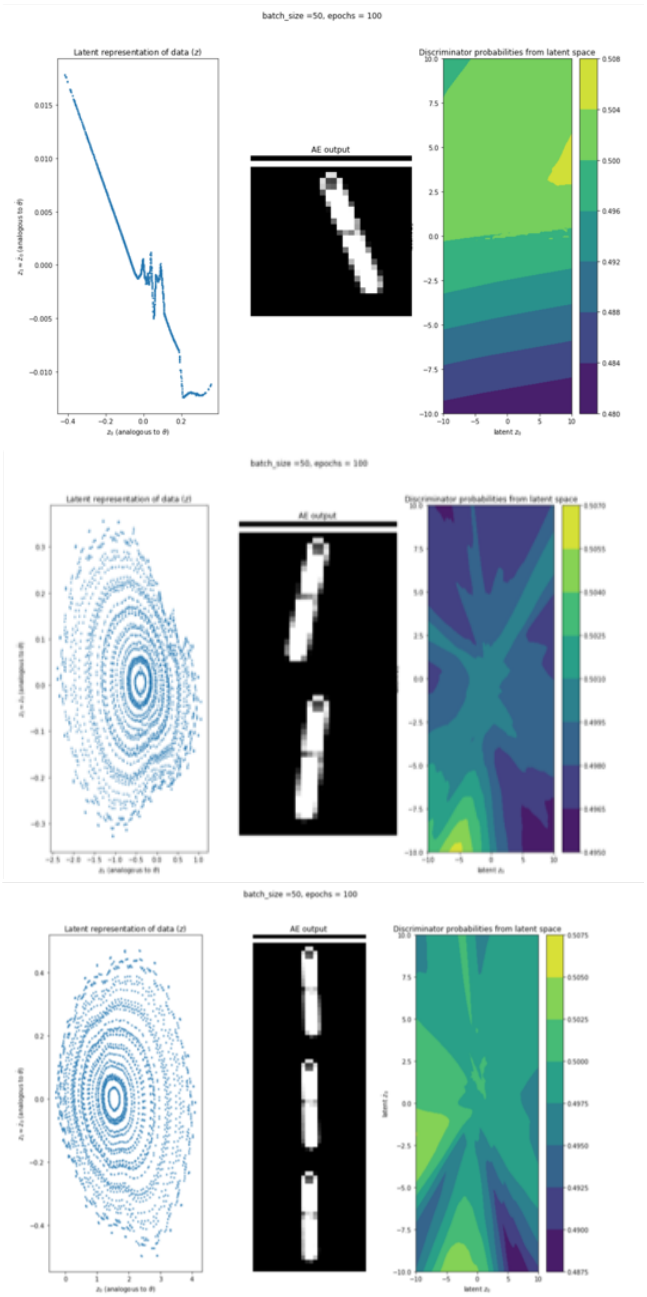
Some experimental results

We carried out an extensive study between batch size and epoch. However, we find no consistency in the generated phase space.

key takeaways

Preliminary results show that GANs are difficult to work with (this could be an interesting topic to cover on its own). I believe gradient clipping is one way to solve this issue. More importantly, Variational Integrator Networks proves that the learnt phase space from an AE will not, in general, represent the true phase space. In order to do this, one needs to encode the Lie group dynamics. Constrained GANs have already been looked at for generating consistent results.

We still have yet to establish why a potential Lie group constrained GAN can help the world. i.e. why is it useful to be able to generate random samples of a pendulum swinging?



2.2 VIGN

Need to fill this section out in detail.

2.3 Lagrangian Dynamics (WIP)

The Euler-Lagrange formulation equates to:

$$\frac{\partial}{\partial t}(\frac{\partial L}{\partial \dot{q}}) = \frac{\partial L}{\partial q}$$

where:

$$L = T - U$$

In the example of a pendulum:

$$T = 1/2mv^2 = 1/2m(l\dot{\theta})^2 = 1/2ml^2\dot{\theta}^2$$

$$U = mgh = mgl(1 - \cos\theta)$$

$$L = 1/2ml^2\dot{\theta}^2 - mgl(1 - \cos\theta)$$

Placing the above into the euler-lagrange form gives us:
LHS:

$$\frac{\partial L}{\partial \dot{q}} = ml^2\dot{\theta}$$

$$\frac{\partial}{\partial t}(\frac{\partial L}{\partial \dot{q}}) = ml^2\ddot{\theta}$$

RHS:

$$\frac{\partial L}{\partial q} = -mgl\sin\theta$$

Therefore:

$$\ddot{\theta} = \frac{g}{l}\sin\theta$$

Now, to transition to the Hamiltonian version:
We can write the hamiltonian as the sum of the energies:

$$H = T + U$$

$$H = p^2/2m + U$$

$$H = -L + 2K$$

$$H = -L + mv^2$$

$$H = -L + p^2/m = -L + p * (m\dot{q})/m = -L + p\dot{q}$$

$$p = \frac{\partial L}{\partial \dot{q}}$$

3 Running Ideas

3.1 VIGN extension

- non conserved energy domain is there a link between the port-hamiltonian view and lagrangians with generalized force for a damped system we have:

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}} - \frac{\partial \mathcal{L}}{\partial q} = \mathbf{F}^{ext}\frac{\partial \mathbf{r}}{\partial q}$$

see here for details on how this equation was built.

- distributional assumptions for practical purposes

Assume $H = E_c$ for a trajectory of an initial condition, then:

$$H = K + U$$

$$U = H - K$$

$$L = K - U$$

$$L = 2K - H$$

Now, for L to be constant, K needs to be constant. In most cases, $K = p^2/2m$. For this to be constant p should be constant. For p to be constant, $\dot{p} = 0$. This would imply, for most settings, that:

$$\frac{dp}{dt} = \frac{dU}{dq} = 0$$

This means that U is zero, which is not true.

We have proven that as long as energy is conserved in a system exhibiting $K = p^2/2m$ the Lagrangian will never be constant unless the system has 0 potential.

As such, the Lagrangian depends on the distribution of the kinetic energy/potential energy. In the large N limit, these distributions tend toward a Gaussian.

Is it beneficial to have a target variable distributed more broadly like a gaussian vs a delta?

3.2 Covid

SEIR model is differentiable and has some hamiltonian/energy conserving property. how might we build a neural network on this?

We start out using SEIR to model china and it works well.

We have issues modeling beijing which is seeing a second spike, we could reshape R naught (t) as an exponential sine function.

$$R_0(t) = Ae^{-(t-t_0)}|sin(B(t-t_0))|$$

This is too specific. How can we encode the SEIR model into a graph neural network. Note, graph networks and graph neural networks are different frameworks encompassing the same core operations. In graph networks, however, the intermediate functional representations (hidden layers) are computed first, before an aggregation scheme is used. In graph neural networks, e.g. GCN, the operations involve computing a hidden layer first using neighbouring nodes, before feeding these hidden layers into the graph neural network.

This reversal means one needs to be careful in how they define graph neural network architectures between DGL and deepmind graph nets library.

We use graph convolutions to train on simulated data, we find hypersensitivity to initial conditions. More importantly, because of the time dependent nature of the system we are unsure of how many lag-variables are necessary to evolve the system.

3.3 Inductive biases for RL

MDPs with policies which are deterministic or stochastic. Model free which involves sampling at random. Model based which involves sampling the learnt environment.