Modeling Dynamical Systems with Deep Neural Networks By Leveraging Quantum Compute

Kelsey J. Harvey
Electrical & Computer Engineering
Carnegie Mellon University
Pittsburgh, PA, U.S.A. 15213
kelseyjh@andrew.cmu.edu

Abhishek Anand
Electrical & Computer Engineering
Carnegie Mellon University
Pittsburgh, PA, U.S.A. 15213
aanand3@andrew.cmu.edu

Chen Zhao
Electrical & Computer Engineering
Carnegie Mellon University
Pittsburgh, PA, U.S.A. 15213
chenzha3@andrew.cmu.edu

Abstract*—Underlying the core principles of the fields of Machine Learning, Robotics, and Quantum Computing are applications of the same fields of Mathematics. Specifically, these fields are Linear Algebra, Probability Theory, Multivariable Calculus, and Mathematical Optimization, as well as components of more abstract mathematical fields such as Partial Differential Equations, Differential Geometry, and Perturbation Theory. We are interested in the investigation of whether or not Quantum Computers, being naturally tied to these fields of Mathematics, lend themselves more favorably to solving problems in Machine Learning, specifically Deep Learning, and Robotics as compared to classical computing architectures. In this paper we investigate the similarities and differences in the implementation of a Deep Neural Network architecture designed for learning the Lagrangian Mechanics of an ideal Double Pendulum being the Dynamical System of interest in both the Classical and Quantum Computing paradigms, as well as a Hybrid variation thereof. The metrics with which we concern ourselves regarding each of these network instantiations is primarily the time necessary to train the network on each piece of hardware. Secondary metrics of interest are the forward propagation time for each input of the test data set as well as the corresponding accuracies of each network to the ground-truth value of the system. For the development, we utilized PyTorch, NumPy, MatPlotLib and Qiskit libraries along with several high-end Nvidia graphics processing units as well as the IBM Quantum Lab platform. The classical training of the double pendulum model, even on high-end GPU hardware took on average of 10 hours and yields an accuracy of approximately 50% in each case. Related to the quantum approach, our team implemented said hybrid quantum design and furthered our research by utilising the very new Qiskit Machine Learning library, enabling to rapidly implement and evaluate our fully Quantum Neural Network design to learn the Lagrangian Mechanics of a conventional Double Pendulum Dynamical System.

***Index Terms*—Machine Learning, Deep Learning, Robotics, Quantum Computing, Dynamical Systems, Deep Neural Network, Linear Algebra, Probability Theory, Multi-variable Calculus, Mathematical Optimization, PDE, Differential Geometry, Perturbation Theory, Lagrangian Mechanics, Double Pendulum**

I. Introduction

The contemporary theories comprising the fields of Machine Learning, Robotics, and Quantum Computing share many commonalities between them, namely the underlying mathematics involved in their respective contrivances and development. Naturally, any student of each field observing these parallelisms, would begin to ponder the potential existence of some greater, overarching, amorphous body of knowledge relating these disciplines similar to the strange loop phenomenon [1], which has yet to be fully understood or recognized by the modern schools of thought of each subject matter. Such a question is deeply philosophical to the fundamental nature of these fields, and indeed, a hypothetical, respective answer to this question is equally crucial to determining the trajectory of which these areas of investigation will follow in the future as the theories become more rigorous and approach respective maturity. Furthermore, one might also wonder if the paradigm of Quantum Computing, given its fundamentally mathematical nature, is better tailored towards performing those calculations which are necessary to the fields of Machine Learning and Robotics, as compared to traditionally utilized classical computing architectures and techniques which have historically struggled with regards to scalability with regards to the levels of compute necessary for solving such classes of problems, as well as, minimizing the financial cost incurred to render these decently accurate models of their corresponding solutions. [2], [3] It is precisely this question which our team shall be investigating through an objective, discriminating, and scientific lens in this document.

II. Motivation

A. Problem Description

The topic of interest for this project is in the investigation of the possibility of creating a pure quantum or classical-quantum hybrid model of a classical Deep Neural Network (DNN) designed to learn and parameterize arbitrary an arbitrary Lagrangian value of a given Dynamical System, known in contemporary literature as a Lagrangian Neural Networks (LNN) [4], and leveraging Quantum Computers for faster training of said Deep Neural Networks as compared to classical GPU hardware. In addition, there is equal interest in the comparison between the feedforward latencies of such a trained network on both classical and quantum domains.

B. Significance of the Solution

The investigation being conducted and reported in this document concerns the Deep Neural Network which will be designed to learn and provide approximate solutions to the

Forward and Inverse Lagrangian Mechanics of an ideal Double Pendulum. The importance of solving this problem is made evident by the Introduction Section of this paper as each of these technologies, Quantum Computing, Machine Learning, and Robotics, will prove themselves to be predominant in the near future in both academics and in society as a whole. Moreover, a demonstration of the solution to such a problem encapsulating and leveraging the anatomical and physiological commonalities between these disciplines would prove seminal to future work being conducted at the intersection thereof.

III. Literature Review

This section outlines the state of the art literature related to the problem of interest in this work. The articles are categorized by subject according to which domain they belong. i.e. Classical or Quantum. The section is concluded by outlining the work to be done from here as well as the desired outcome of these efforts as they impact each field in part.

A. Background

1) ***Double Pendulum****:* A Double pendulum, is a time-independent Hamiltonian system exhibiting chaotic behaviour, consisting of two point masses connected in a serial, tip-to-tail, fashion via mass-less rods. Moreover, the joints possess full rotational freedom in the plane but are translationally constrained. That is, the translational motion of the double pendulum system is governed by a set of coupled ordinary differential equations. A direct result of this relationship is that the double pendulum demonstrates nonlinear dynamics with one pendulum fixed at the other end of another pendulum in that its motion is represented by the Euler-Lagrange Equation (5). In a planar Cartesian coordinate system the origin-centered simplified double pendulum may be modelled as a Free Multi-Body Diagram, as depicted in *Figure 1*.

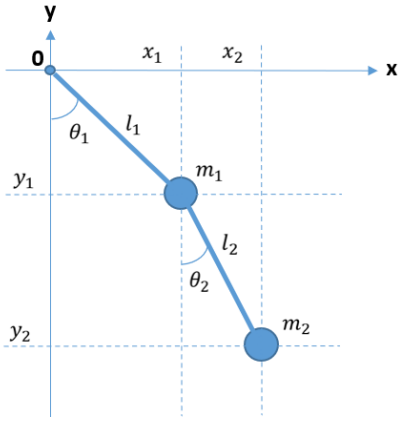

Fig. 1. Simplified model of Double Pendulum.

Here, m_1, m_2 corresponds to masses of two point mass, l_1, l_2 corresponds to length of mass-less rods, θ_1, θ_2 corresponding displaced angles, where all pivots points assumed as friction-less.

2) ***Lagrangian Mechanics****:* The equations describing the motion of double pendulum can be expressed using Newtonian, Hamitonian and Lagrangian mechanics. In this work focuses in the Lagrangian mechanics more prominently utilising the Euler-Lagrange formulation of motion taking into account of non-conservative forces and generalized coordinates. Basically Lagrangian describes the complete dynamics of a given system that need not be unique and is valid when yields correct equations of motion. The Lagrangian is generally formulated as,

$$\mathcal{L} = T - V \tag{1}$$

where T corresponds to the kinetic energy and V corresponds to the potential energy.

3) ***Deep Lagrangian Network for Modeling Lagrangian mechanics*** *:* In a similar vein of work is the paper by Lutter et al. [5] which expands the scope of the target dynamical systems to full robotic systems. The Deep Neural Network employed in the paper has Lagrangian Mechanics imposed upon it as the objective criteria, given the sobriquet "Deep Lagrangian Network" or "DeLaN". The DeLaN learns the mechanics of the system it is modelling online, that is to say, in real-time, as compared to classical Deep Neural Nets which are trained on a batch of prior data and deployed thereafter. The authors found that as a result of the online learning and the robustness of the Lagrangian Objective, that training of the DeLaN to learn the robot model was indeed even faster and yielded better results as compared to traditional Deep Learning training techniques.

4) ***Lagrangian Neural Network for Modeling Lagrangian mechanics*** *:* In their work, Gupta and Menda [6] discuss methods for unifying prior knowledge about the behavior of a dynamical system with a deep learning based approach that allows for parameterizing generic mechanical dynamic system models. It emphasizes on employing neural networks to parameterize the Lagrangian of a system with generalized forces acting on it. In this paper we focus our interest to the investigation of the double pendulum problem, as it is the canonical example of dynamical system in the study of Lagrangian Mechanics. This research is expounded upon by Cranmer et al. [4] that discusses to develop a neural network design that learn symmetries in the input data corresponding to physical conservation laws acting upon the dynamical system, that in this case corresponds to a double pendulum. In his work Cranmer [4] exploit these symmetries to improve output performance and reduce training times as compared to prior Hamiltonian-based approaches to modelling physical systems using neural networks.

B. Classical Approaches

A mathematical description of a problem in computational physics requires solutions of partial differential equations restricted by initial and boundary conditions. In the field of dynamics, it is imperative that an accurate representation of the system model is available when implementing control

methods in an environment. In classical approach we used Cranmer et al. [4] Lagrangian Neural Network in modeling the chaotic dynamic system of double pendulum to generate the data for quantum implementation. Here, State Representation definition is given as,

$$x = [\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2]^T \tag{2}$$

$$\dot{x} = [\dot{\theta}_1, \dot{\theta}_2, \ddot{\theta}_1, \ddot{\theta}_2]^T \tag{3}$$

Lagrangian equation for motion of double pendulum, related to mass and length is expressed as

$$\mathcal{L} = T - V \tag{4}$$

$$\frac{\partial \mathcal{L}}{\partial f_i} = \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial f_i'}\right) \tag{5}$$

$$\begin{aligned}\mathcal{L} = \frac{1}{2}(m_1+m_2){l_1}^2\dot{\theta}_1^2 + \frac{1}{2}m_2{l_2}^2\dot{\theta}_2^2 \\ +m_2 l_1 l_2 \dot{\theta}_1\dot{\theta}_2 \cos(\theta_1-\theta_2) \\ +(m_1+m_2)g l_1 \cos\theta_1 \\ +m_2 g l_2 \cos\theta_2\end{aligned} \tag{6}$$

Rearranging the equations of motion for classical encoding, for data generation,

$$\begin{aligned}(m_1+m_2)l_1\ddot{\theta}_1 + m_2 l_2 \ddot{\theta}_2 \cos(\theta_1-\theta_2) + \\ m_2 l_2 \ddot{\theta}_2^2 \sin(\theta_1-\theta_2) + \\ (m_1+m_2)g\sin(\theta_1) = 0\end{aligned} \tag{7}$$

$$\begin{aligned}m_2 l_2 \ddot{\theta}_2 + m_2 l_1 \ddot{\theta}_1 \cos(\theta_1-\theta_2) \\ -m_2 l_1 \dot{\theta}_1^2 \sin(\theta_1-\theta_2) \\ +m_2 g \sin(\theta_2) = 0\end{aligned} \tag{8}$$

C. Classical Baseline Architecture

For this work, we are implementing a Structured Mechanical Model [7] which will be trained by minimizing the discrete Euler-Lagrange residual [8] as the baseline architecture. This technique defines $\boldsymbol{q}$ as the system configuration, and parameterizes a mechanical system into 4 different networks, where $\boldsymbol{M(q)}$ is the system Mass-Matrix, *V(q)* represents the system's potential energy, $\boldsymbol{B(q)}$ represents the input Jacobian for control inputs, and $\boldsymbol{F(q,\dot{q})}$ is the dissipative forces that act on the system. For calculating loss during training, we will calculate the discrete Euler-Lagrange (DEL) residual, where we utilize the following definitions from Menda et. al previous work [8]:

Lagrangian Equation for the Double Pendulum:

$$\mathcal{L}(q,\dot{q}) = \frac{1}{2}\dot{q}^T \boldsymbol{M(q)}\dot{q} - V(q) \tag{9}$$

And the subsequent Discrete Lagrangian is defined as:

$$\mathcal{L}_d(q_1, q_2, h) = h\mathcal{L}\left(\frac{q_1+q_2}{2}, \frac{q_2-q_1}{h}\right) \tag{10}$$

where h is defined as the time step between q_1, q_2

The discrete Euler-Lagrange equation is then defined as:

$$DEL(q_1,q_2,q_3,h) = D_2\mathcal{L}_d(q_1,q_2,h) + D_1\mathcal{L}_d(q_2,q_3,h) = 0 \tag{11}$$

where D_i is defined as the slot derivative, where the partial derivative of the function is taken with respect to the $i-th$ argument [8]

Which, leveraging the d'Alembert Principle of Virtual Work, is equivalent to:

$$\begin{aligned}DEL(q_1,q_2,q_3,h) = D_2\mathcal{L}_d(q_1,q_2,h) + D_1\mathcal{L}_d(q_2,q_3,h) \\ +\frac{1}{2}(F_d(q_1,q_2) + F_d(q_2,q_3)) = 0\end{aligned} \tag{12}$$

where F_d is defined as the discrete external forces acting on the system, where u_i are defined as the control inputs at the $i-th$ time step:

$$\begin{aligned}F_d(q_1,q_2,h) = hF\left(\frac{q_1+q_2}{2}, \frac{q_2-q_1}{h}\right) \\ = h\left(\boldsymbol{B}\left(\frac{q_1+q_2}{2}\right)\left(\frac{u_1+u_2}{2}\right) + \boldsymbol{F}\left(\frac{q_1+q_2}{2}, \frac{q_2-q_1}{h}\right)\right)\end{aligned} \tag{13}$$

The DEL residual is calculated by providing system configurations q_1, q_2, and q_3 into the DEL equation, which should optimally result in a value of zero, so we update the network parameters to minimize this value.

To ensure that the of the mass-matrix remains positive-definite throughout the training, we parameterize the Cholesky factor, and indirectly learn the mass-matrix [6], also applying a small regularization penalty during the loss calculation as described by Gupta et. al, which helps to ensure that the matrix does not become singular during training [8].

D. Quantum Approaches

In the Quantum approach, the work contributed by Weibe et al. [9] elucidates the capability and indeed feasibility of quantum computer to train deep neural networks. In their work, it mentioned that not only is the amount of time needed to train Deep Neural Networks (DNNs) reduced by the utilization of quantum compute, but that a quantum approach leads to a richer and more comprehensive framework for training DNNs in contrast to classical hardware. Furthermore, the methods created by the authors permit the efficient training of full Boltzmann Machines and Fully-Connected Networks

of multiple layers, for which the classical counterparts do not have well-known implementations.

Specific details for training Deep Neural Networks via Quantum Computers are well documented by Verdon et al. [10] who introduce their quantum-based analog to the classical back propagation technique, dubbed 'Baqprop' in the paper. More specifically, the authors outline a quantum instance of the canonical Gradient Descent Algorithm, which they dub 'Quantum Dynamical Descent' or 'QDD', for minimizing the objective criterion to learn the weights of the Deep Neural Network. Further still, the team also define and investigate a quantum version of the Momentum Gradient(MoMGrad), optimizer algorithm which is a common modification to vanilla gradient descent. The primary intent of the authors is to achieve a universal training algorithm for Quantum Deep Learning.

IV. Intended Impact of This Work

Our focus is to partially recreate these results by modelling the Lagrangian mechanics specifically of double pendulum using available quantum computing platform. With this implementation we are aiming to synthesize robust proof-of-concept to derive an efficient means of training a compliant robotic control and manipulation that would ultimately impact the future of in this fields.

V. Description of Data set

For training and testing, Data set of double pendulum is generated randomly using equation 7 and 8. Mass, length, $\theta, \dot{\theta}$ are initialized as below.

mass:

$$m1 = 1$$
$$m2 = 1$$

length:

$$l1 = 1$$
$$l2 = 1$$
$$\theta \; bound, \quad \theta_1, \theta_2 \in (-\pi, \pi)$$
$$\dot{\theta}_1, \dot{\theta}_2 \in (-\pi/12, \pi/12)$$

Both generated training and test data is categorized as input and label data. Each input and label is in the form $[\dot{\theta}_1, \dot{\theta}_2, \ddot{\theta}_1, \ddot{\theta}_2]$ represent the state of pendulum masses from pivot. By randomly generating the value of $\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2$, 500000 (500K) training and test data was generated.

VI. Training Instances

For the development, and performance of the implemented hybrid and full Quantum DNN for Lagrangian mechanics of double pendulum, we explored the IBM quantum lab having 8 gigabyte of random access memory with Qiskit libraries. For development activities, we utilized Google Collab with IBM Qiskit. Code for the model and the training is available at the teams' GitHub Repo located here.

VII. Design & Results

Each of these DNN designs as outlined in their respective papers constitute the state-of-the-art techniques in their fields. It will be the job of this team to partially recreate these results as well as synthesize them into a robust proof-of-concept for an efficient means of compliant robotic control and manipulation which we hope will impact the future of each of the fields outlined previously.

A. Classical Approach

In Classical model, we utilised PyTorch for modeling the 7 fully-connected hidden layers of Lagrangian Neural Network with softplus as activation function with calculation of Hessian and Jacobian.

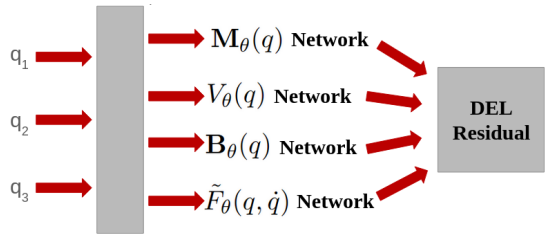

Fig. 2. Visualization of the DEL Network Pipeline.

Regarding preliminary results of the DNN training on classical hardware, the most promising result obtained with respect to the training time was approximately 20.5 hours with approximately 51.7% accuracy to the ground-truth Lagrangian values. This result was obtained on a locally owned and operated Nvidia RTX 5000 GPU. Even more promisingly, highly valuable results were obtained via Google Colab, Google Cloud Platform (GCP), and Amazon Web Services (AWS). The remote instance operating on Google Colab was equipped with an Nvidia Tesla P100 GPU on which the Classical DNN was trained for four Epochs averaging 10 hours and 17 minutes each with mean prediction time of 133.8 seconds and an accuracy of approximately 51%. Likewise, the remote instance operating on GCP was equipped with an Nvidia Tesla P4 GPU on which the Classical DNN was trained for three epochs, before experiencing vanishing gradient issues due to the formulation of the discredited, numerical solution to the Euler-Lagrange Equation 5 occasionally resulting in poorly conditioned output matrices. As a result, the mean training time for the same architecture DNN on this GPU hardware was moderately improved to approximately 7 hours and 29 minutes each with mean prediction time of 85.52 seconds, but an accuracy of approximately 48.2% with respect to the ground-truth Lagrangian Values. Finally, in similar fashion to the prior training instance, the DNN was trained via AWS on a `g4dn.xlarge` instance for three epochs before experiencing the same numerical stability issues resulting in a mean training time similar to the Google Colab instance of approximately 10 hours and 27 minutes; however, the AWS instance in this case experienced significantly faster prediction times than the

Colab instance, averaging around 69.16 seconds for validation processing.

B. Hybrid-Quantum Approach

Inspired from the work of Maxwell et al. [11], quantum circuits are modelled taking into account the classical input data and output from the circuit. Here the output obtained is fed to classical layer in the hybrid design and in other full quantum approach to the quantum neural network (QNN). Quantum circuit is static and input data is encoded in encoded into the initial states of each individual qubit of quantum circuit.

With taking into account the input and output data size from classical neural network, we came up with the simple quantum circuit *Fig.3* with single qubit capable of handling only single classical input at a time. This qubit is followed by hadamard gate, random rotation and then measurement.

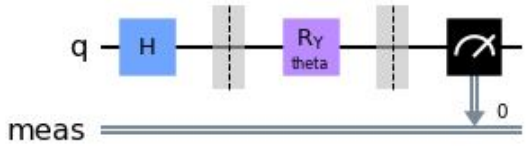

Fig. 3. Quantum Circuit (Single qubit).

With an aim to process one complete set of data at a time, we designed the quantum circuit *Fig.4*, with four qubit taking into account of four input value, that results to four valued output from the circuit which is directly fed to the classical layer. Circuit design consists of initial hadamard gate for initialization in first stage, followed by entanglement between qubits in second stage and finally measurements of the results.

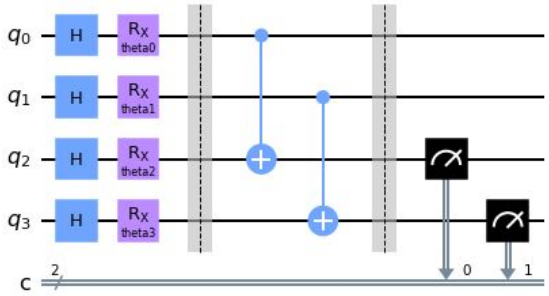

Fig. 4. Quantum Circuit (Four qubit).

1) ***Classical-Quantum****:* In this classical-Quantum design approach, Classical layer is followed by quantum circuit. Classical input data is fed to the classical neural network layer whose output is further fed to the quantum circuit. The quantum circuit is shown in *Fig.5*.

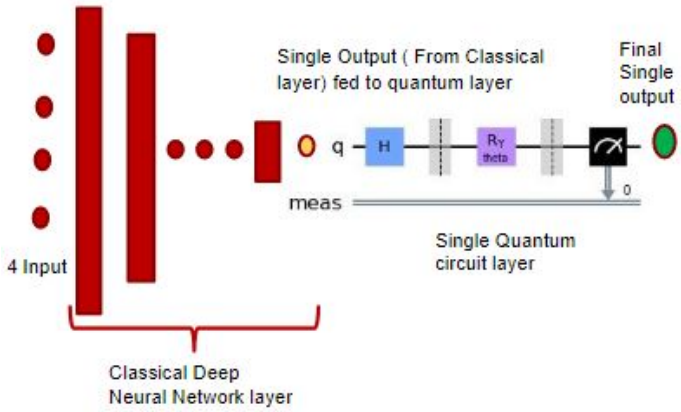

Fig. 5. Hybrid Classical-Quantum Neural Network design with single qubit.

The loss response versus training iterations in *Fig.6* obtained after comparing the training result and reference result for the implemented design as in *Fig.5*.

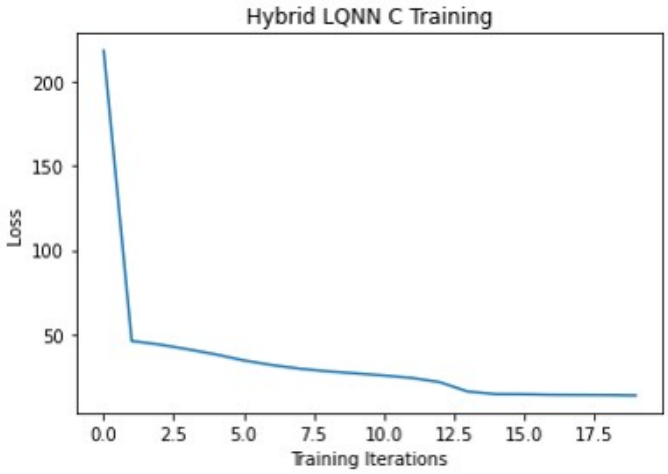

Fig. 6. Hybrid Classical-Quantum loss response.

2) ***Quantum-Classical(Single-Qubit)****:* In this Quantum-Classical design approach, quantum circuit is followed by classical layer. Classical input data is fed to the quantum circuit whose output is further fed to the classical neural network layer. The quantum circuit is shown in *Fig.7*. This hybrid single qubit quantum circuit which take single value input at a time and generate single value output. In case of double pendulum input data, which is a four valued $i.e$ [a, b, c, d]. Each classical input is fed to single-qubit quantum circuit one after the other (calling same circuit four times to process each input) to final four valued result say [g,h,i,j] $i.e$ [a yields g], [b yields h], [c yields i], [d yields j].

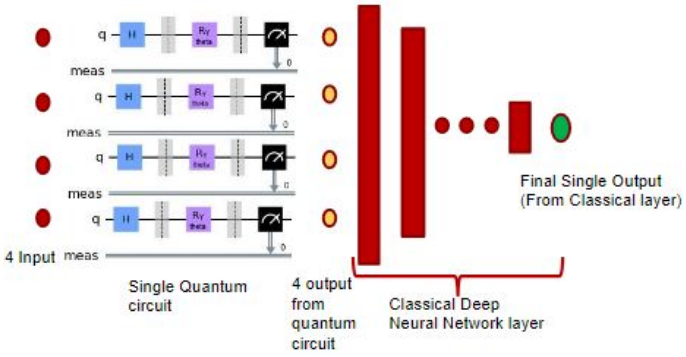

Fig. 7. Hybrid Quantum-Classical Neural Network with single qubit.

The loss response versus training iterations in *Fig.8* obtained after comparing the training result and reference result for the implemented design as in *Fig.7*.

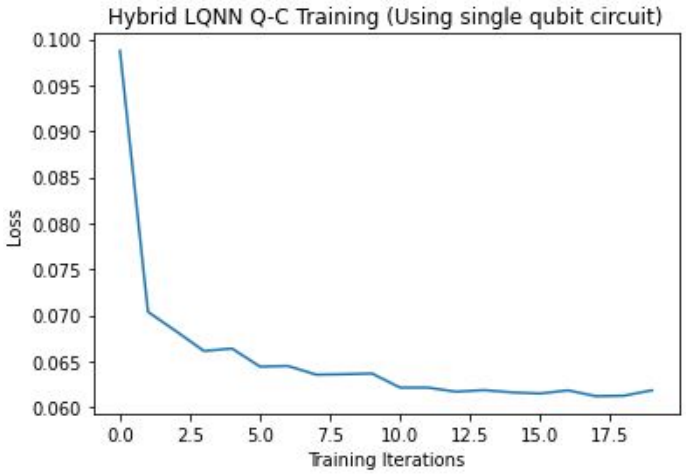

Fig. 8. Hybrid Quantum-Classical (single qubit) loss response.

3) ***Quantum-Classical(Four-Qubit)****:* In this Quantum-Classical design approach, quantum circuit with four qubit take four valued input at a time and generate four value output. Classical input data is fed to the quantum circuit whose output is further fed to the classical neural network layer. The quantum circuit is shown in *Fig.9*. The four valued double pendulum input data say $[a, b, c, d]$ when fed to quantum circuit yields four valued output say $[g, h, i, j]$.

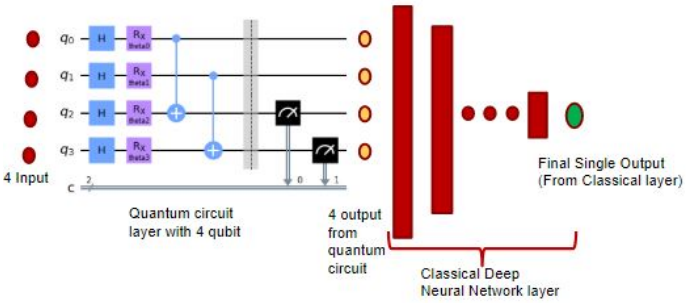

Fig. 9. Hybrid Quantum-Classical Neural Network with four qubit.

The loss response versus training iterations in *Fig.10* obtained after comparing the training result and reference result for the implemented design as in *Fig.9*.

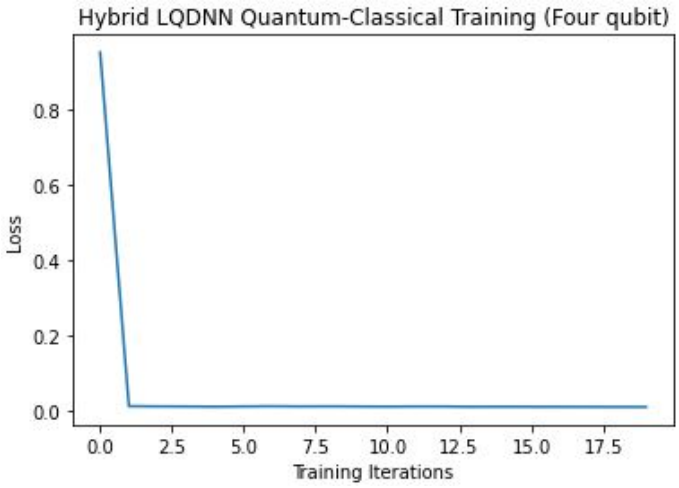

Fig. 10. Hybrid Quantum-Classical (four qubit) loss response

C. Full-Quantum Approach

Encouraged from the machine learning algorithm available with IBM Qiskit platform, especially for neural network training helped us to come up with Full quantum design for double pendulum.

For a simple step to implement full QNN, we leveraged neural network regressor [12]. This simple approach requires our input data to be encoded first before feeding to the regressor. So we designed a quantum circuit *Fig.11* with four qubit, followed by individual random rotation and finally measurement. Random number of output for four valued input, is manipulated by calculating the probability of occurrence of 1 from total output so to reduce the final output size to one.

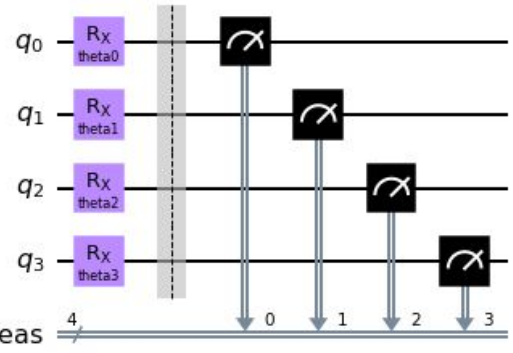

Fig. 11. Simple quantum circuit for classical data encoding (Required for TwoLayerQNN).

IBM Qiskit provides neural networks algorithms, especially, Neural Network Regressor is leveraged with TwoLayerQNN as shown in listing 1, a special case of Operational Flow Quantum Neural Network (OpFlowQNN). Classical data is encoded

using quantum circuit, further calculating the probability of occurrence of 1 from the result obtained.

```
# construct simple feature map
param_x = Parameter('x')
feature_map = QuantumCircuit(1, name='fm')
feature_map.ry(param_x, 0)
print(feature_map)

# construct simple ansatz
param_y = Parameter('y')
ansatz = QuantumCircuit(1, name='vf')
ansatz.ry(param_y, 0)
print(ansatz)

# construct TwoLayerQNN
twolayerqnn = TwoLayerQNN(1, feature_map, ansatz,
    quantum_instance=quantum_instance)

# construct the regressor from the neural network
nnr = NeuralNetworkRegressor(neural_network=
    twolayerqnn,loss='l2',optimizer=L_BFGS_B(),
    callback=callback_graph)
```

Listing 1. Neural Network Regressor with TwoLayerQNN

The objective function value response versus iterations in *Fig.12* obtained for the full quantum implementation of double pendulum with 1481.9600680875417 as minimum value of objective function.

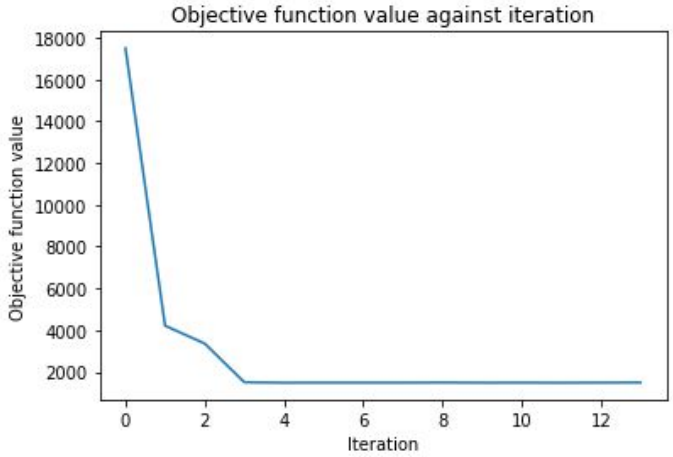

Fig. 12. Full Quantum response.

VIII. Impact of noise

In the design of quantum circuits for small or large computations, it is important to consider the impact of noise that dominates the near-term quantum computers either due to to infidelities of quantum gates, measurement, devices or due to unwanted environment interactions. In Depolarisation error on a qubit means to convert it to a completely mixed state. Here, we have analysed the impact of noise on the full quantum circuit of double pendulum, for this we used *one* qubit *Fig.13* and *two* qubit *Fig.14* depolarising noise model.

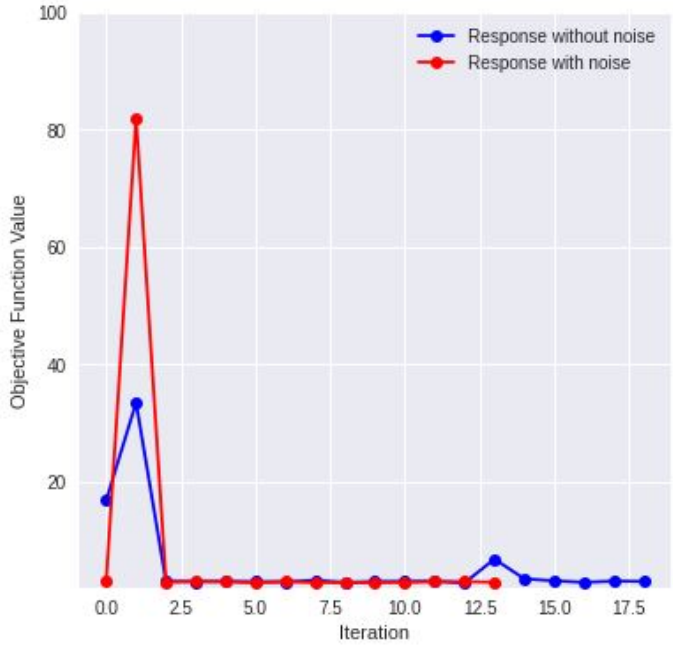

Fig. 13. Full Quantum circuit response comparison with one qubit gate noise.

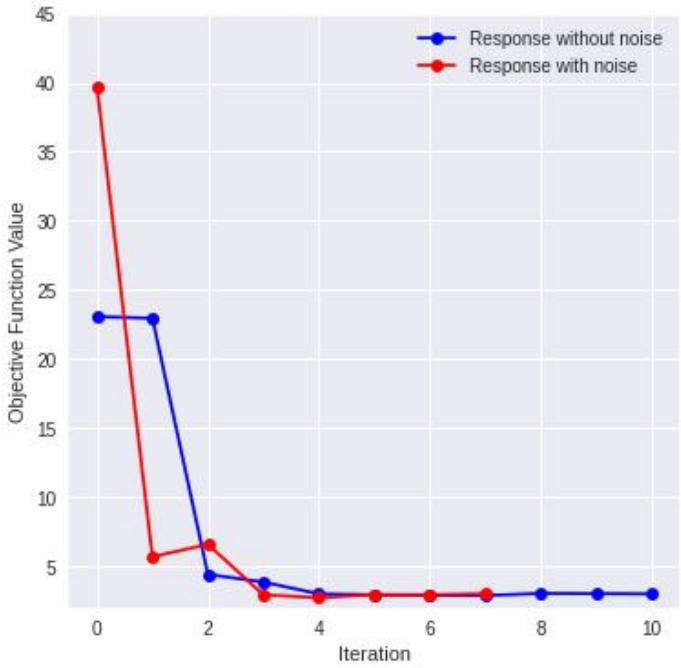

Fig. 14. Full Quantum circuit response comparison with two qubit gate noise.

IX. Conclusion

In this paper, we have formulated and modelled the dynamic systems, describing Lagrangian mechanics with the Lagrangian neural network both in classical and quantum approach. This is first attempt to model the Lagrangian neural network of double pendulum with the quantum computing platform. We are able to analyse LNN with full quantum

replacement with the comparison to reference and training model. With the available quantum environment, with large training data, training time is longer when compared to classical, on the simulated and the actual quantum machines. With promising results obtained thus far we want to continue further with the implementation to have direct mapping of data in the quantum intermediate neural layer to achieve better fit, loss and accuracy prediction with the complete data set.

REFERENCES

[1] D. R. Hofstadter, *Gödel, Escher, Bach: An eternal golden braid*. Basic Books, 1999.

[2] R. Sagar, "Why deep learning is a costly affair," May 2020.

[3] B. Dickinson, "Why reducing the costs of training neural networks remains a challenge," Oct 2020.

[4] M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, and S. Ho, "Lagrangian neural networks," 2020.

[5] M. Lutter, C. Ritter, and J. Peters, "Deep lagrangian networks: Using physics as model prior for deep learning," 2019.

[6] J. K. Gupta, K. Menda, Z. Manchester, and M. J. Kochenderfer, "A general framework for structured learning of mechanical systems," *arXiv preprint arXiv:1902.08705*, 2019.

[7] J. K. Gupta, K. Menda, Z. Manchester, and M. J. Kochenderfer, "Structured mechanical models for robot learning and control," *arXiv preprint arXiv:2004.10301*, 2020.

[8] K. Menda, J. K. Gupta, Z. Manchester, and M. J. Kochenderfer, "Training structured mechanical models by minimizing discrete euler-lagrange residual," 2021.

[9] N. Wiebe, A. Kapoor, and K. M. Svore, "Quantum deep learning," 2015.

[10] G. Verdon, J. Pye, and M. Broughton, "A universal training algorithm for quantum deep learning," 2018.

[11] M. Henderson, S. Shakya, S. Pradhan, and T. Cook, "Quanvolutional neural networks: Powering image recognition with quantum circuits," 2019.

[12] "Qiskit machine learning api reference machine learning base classes - neural network regressor," *Qiskit Documentation*.