California State University,
Long Beach

Hamiltonian Neural Network Exploration for Electron Particle Tracking

THESIS PROJECT

submitted in partial satisfaction of the requirements
for the degree of

MASTER OF SCIENCE

in Applied Mathematics

by

Mikaela Meitz

Thesis Project Committee:
Dr. Paul Sun, CSULB, Chair
Dr. Jen-Mei Chang, CSULB
Dr. Xiyue Liao, CSULB
Dr. Lipi Gupta, Lawrence Berkeley National Laboratory

May 2023

DEDICATION

For mom, dad, and Lance.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to express my gratitude to the Lawrence Berkeley National Laboratory (LBNL) for providing me with the invaluable opportunity to pursue this research. I am also deeply appreciative of the support extended to me by the team at the National Energy Research Scientific Computing Center (NERSC).

My sincere appreciation goes to my esteemed mentor, Dr. Lipi Gupta from LBNL, for the insightful topic and inspiration behind my thesis project, and for her invaluable guidance throughout the journey.

A special word of thanks goes to my graduate advisor, Dr. Paul Sun, for his consistent support and guidance throughout my graduate career. Thank you to my thesis project committee, Dr. Jen-Mei Chang and Dr. Xiyue Liao, for their continued support.

Lastly, I extend my heartfelt gratitude to my beloved family for their constant encouragement and unwavering support in my pursuit of a master's degree.

VITA

Mikaela Meitz

EDUCATION

Master of Science in Applied Mathematics **2023**
California State University, Long Beach *Long Beach, California*

Bachelor of Science in Mathematics **2020**
California State University, San Marcos *San Marcos, California*

RESEARCH EXPERIENCE

Graduate Research Assistant **2022**
Lawrence Berkeley National Laboratory *Berkeley, California*

EXPERIENCE

Graduate Assistant **2022**
California State University, Long Beach *Long Beach, California*

Teaching Assistant **2021**
California State University, Long Beach *Long Beach, California*

FEATURED ARTICLES

NERSC Summer Student Applies Machine Learning to Particle Accelerator Design Challenge **Aug 2022**
Berkeley Lab Computing Sciences, News & Events

https://cs.lbl.gov/news-media/news

ABSTRACT

Hamiltonian Neural Network Exploration for Electron Particle Tracking

By

Mikaela Meitz

Master of Science in Applied Mathematics

California State University, Long Beach, May 2023

Dr. Paul Sun, CSULB, Chair

In the field of accelerator physics, there is a burgeoning interest in using machine learning methods for aiding in the design and optimization of charged particle accelerators. A periodic circular accelerator, called a synchrotron, emits ultraviolet and soft x-ray beams by accelerating electron bunches nearly as fast as the speed of light. These accelerators are prone to beam instability resulting in particle loss and consequently creating less x-ray brightness. The dynamic aperture (DA) measures the beam stability and is determined through computationally intensive particle tracking iterations. During the design or upgrade process, particle tracking is needed to ensure the particle dynamics are sufficient for the intended scientific use. Since the optimization of DA is iterative it is imperative to find ways to reduce each iteration length. Machine learning methods may alleviate some of the need for these expensive computations by making particle integration faster and easier to parallelize. This research explores electron particle tracking with the use of Hamiltonian Neural Networks. Our proposed method of numerical particle tracking involves solving Hamiltonian's equations which emulate the behavior and motion of particles within a particle accelerator. Machine learning based Hamiltonian Neural Networks (HNN) constrain the model learning to obey Hamiltonian mechanics so that the neural network can learn conservation laws from data [16]. We compare the performance of HNN to other machine learning based models [17].

Chapter 1

Particle Accelerators

In the early days of the COVID-19 pandemic, there was an urgent need to research the characteristics of the SARS-CoV-2 virus. The Advanded Light Source (ALS) at Lawrence Berkeley National Laboratory, a particle accelerator, launched several experiments using nonliving samples of the SARS-CoV-2 [9]. The ALS is a cyclic particle accelerator facility that produces x-rays to image crystallized forms of samples to create 3D structures of proteins, viruses (including the SARS-CoV-2 virus), and various other samples [9]. With the 3D structure of the SARS-CoV-2 virus, pharmaceutical scientists had the ability to further research on drug creation to attack the virus [9].

1.1 Brief History

In 1895, Röntgen discovered x-rays which lead to an interest in high-energy collisions and to the the creation and popularity of x-ray tubes from 1898-1905 [18]. By the 1930's, many experiments had been done to study methods for creating higher energy charged particle beams. The first "high energy" accelerator, developed by Crockroft and Walton, created

the nuclear transmutation produced by a discharge in hydrogen gas [3]. Other creations at the time such as Van-de-Graaff accelerators and Wideroe's "ray transformer" aided in the creation of circular accelerators called cyclotrons.

The first successful operating cyclotron was built in 1931 by E.O. Lawrence and M.S. Livingston near the University of California Berkeley campus, at a facility called The Radiation Laboratory [11]. After winning the Nobel Prize in 1939 for the invention of the cyclotron, Lawrence improved the accelerator by creating a machine with a larger magnet and increasing the circumference of the accelerating chamber [11]. The Radiation Laboratory was then relocated to its present-day location and after his death in 1958, the Radiation Laboratory was re-named the Lawrence Radiation Laboratory in honor of E.O. Lawrence and his work on particle accelerators. In 1995, the laboratory was officially renamed the Ernest Orlando Lawrence Berkeley National Laboratory [11].

Lawrence Berkeley National Laboratory (LBNL) is home to the building that housed this cyclotron, built in 1946, and now holds the Advanced Light Source, or ALS. Today, the ALS provides world class synchrotron light source capabilities to the scientific community. A synchrotron emits ultraviolet and soft x-ray beans by accelerating electron particle bunches nearly as fast as the speed of light. The ALS and many other particle accelerators around the world are used for radiation diffraction imaging used for research in chemistry, biology, material sciences, and other fields.

1.2 Fundamentals of Particle Accelerators

Particle accelerators are large machines that accelerate charged particles. A depiction of a typical circular particle accelerator is shown in Figure 1.1. Particles travel in bunches and traverse the machine inside a vacuum tube, and periodically through alternating electro-

magnets. While traveling through a vacuum tube, the electromagnets steer and focus the particle beam. Particle beams are accelerated with the use of alternating electric fields that accelerate the particle beam to speeds close to the speed of light [1]. The radiation created from this acceleration can be optimized and used in various ways [2]. Light from the particle beam hits the samples, and particle detectors then record the data that was created by the diffraction. Computers are then able reconstruct the samples structure revealing an image. This process is also known as diffraction imaging [1]. Information such as composition, structure, and dates of samples can be gathered for fundamental science research.

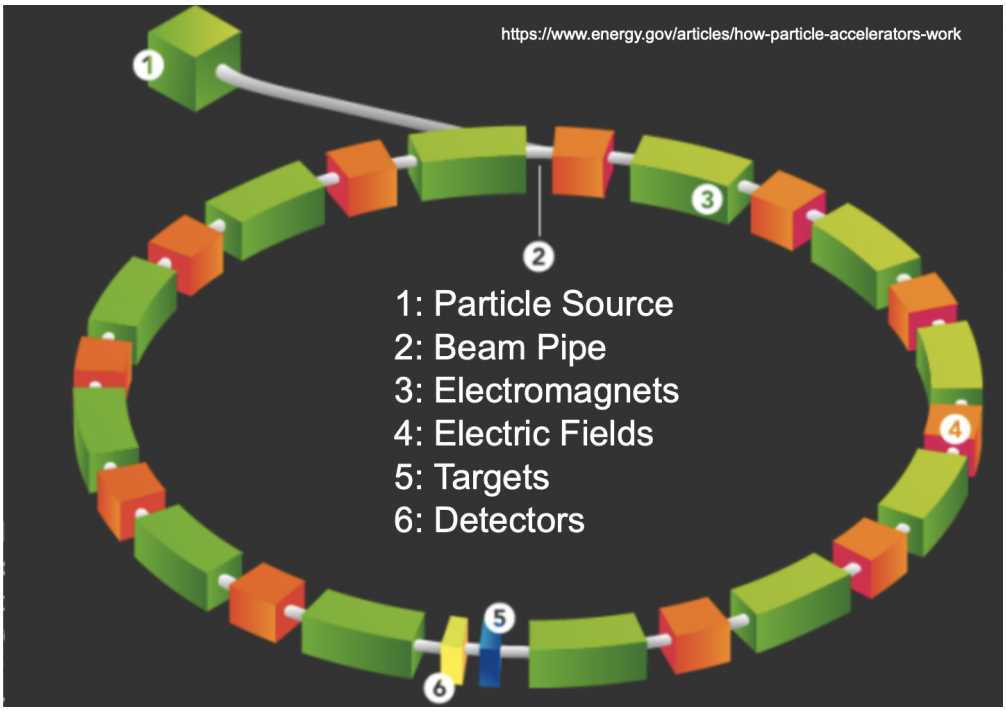

Figure 1.1: Schematic of a general circular particle accelerator [1]. The particles emanate from a particle source (1) in bunches and traverse through a beam pipe (2) that alternates between electromagnets (3) and electric fields (4). The electromagnets steer the particle beam while the electric fields alternate between positive and negative poles, which accelerates the particle bunches. The particles collide with a stationary target (5) and interact with detectors (6). The detectors record the radiation that results from the collision with the target.

Chapter 2

Particle Accelerator Upgrades and Design Challenges

Particle accelerators are upgraded often to expand their capabilities and to align with emerging technologies. During the design or upgrade process for these machines, electron particle tracking is needed to ensure the particle dynamics are sufficient for the intended scientific use. However, multi-dimensional particle tracking can be computationally expensive and the optimization process requires iterative simulations, which compounds the compute time. We will asses if machine learning models can be an efficient method for particle tracking with the use of simulated data for a generalized circular particle accelerator.

For example, the ALS-U is an ongoing upgrade at LBNL that will enhance the Advance Light Source's capabilities to produce bright steady beams of high-energy light. The replacement of the electron storage ring will allow the machine to produce results that no other particle accelerator in the world can accomplish [10]. Upon the completion of ALS-U, the newly designed machine will undergo a series of tests to verify that it is ready for operational deployment. One of these tests includes rigorous numerical particle tracking to ensure that

the stability of the particle beam lies within a desired region of phase space.

2.1 Stability

All particle accelerators are prone particle loss, which can result from the inherent instability of particle beams due to external fields and perturbation [15]. In this case, stability refers to the longevity of a particle bunch circulating within the particle accelerator. The preservation of a particle's beam stability over numerous revolutions is an important factor and contributes significantly to the overall performance of the accelerator. Thus, limited stability can causes particle loss, leading to a decline in both the luminosity and the brightness of x-rays. When upgrading a particle accelerator or designing a new particle accelerator, it is necessary to perform comprehensive stability studies. Numerical particle tracking provides crucial insight into the theoretically achievable stability region. The stability region can vary across different accelerator configurations. Typically, this region is depicted and analyzed in a two-dimensional mapping, as shown in Figure 2.1.

2.2 Numerical Particle Tracking

Numerical particle tracking tracks the position and momentum, or time derivative, of the electron particle trajectory for numerous revolutions around the particle accelerator. This tracking can give insight as to where the particles land in the region of phase space as a result of magnetic structures and perturbations along the accelerator.

The dynamic aperture is the stability region of phase space in a particle accelerator and is found by particle tracking. If the dynamic aperture is too small, then adjustments are made, and the numerical particle tracking process is repeated until a desired result is achieved.

However, this process can take hours to days to complete. An application of machine learning techniques may have the ability to significantly expedite this process by leveraging previous trajectory data to predict future particle trajectories. Consequently, machine learning may serve as an efficient and powerful tool for particle tracking for the optimization of stability.

In Figure 2.1, the boundaries of distinct regions, namely the pink, blue, and red areas, is a representation of particle survival following a varying number of revolutions within a particle accelerator. The pink region represents the optimal stability zone, where particles are theoretically capable of surviving an infinite number of revolutions within the particle accelerator. The blue region denotes limited stability, where particles may be lost after a finite number of revolutions. The red region exhibits a complete lack of stability, characterized by significant particle loss. The region of stability can be improved by numerical particle tracking during the tuning and upgrade process to ensure optimal accelerator performance. However, numerical particle tracking is very computationally expensive since when any change is done to the particle accelerator, numerical tracking is required.

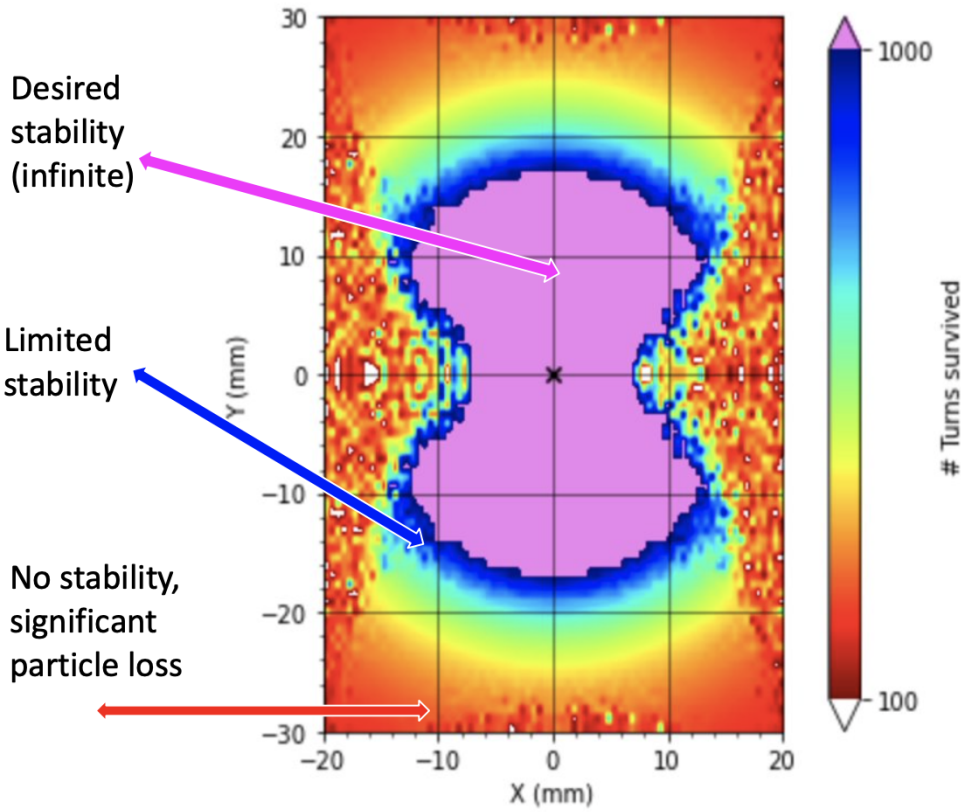

Figure 2.1: Stability plot in physical space within a beam pipe, created by using numerical particle tracking via computer simulation [8]. (Pink) Desired stability, no particle loss, desired region. (Blue) Limited stability, some particle loss. (Red) No stability, undesired region

Numerical particle tracking involves solving Hamiltonian's equations, as will be outlined in Chapter 3, which emulate the behavior and motion of particles within a particle accelerator. Leveraging the power of machine learning, methods are devised to enhance the efficiency and speed of solving Hamiltonian's equations. In essence, Numerical particle tracking is the numerical integration of the Hamiltonian equations.

2.3 Stability of Chosen Data

Figure 2.2 displays the trajectory mapping onto phase space, position and momentum, for 400 individual particle trajectories each with $1,000$ points. Within the context of this study, the region of stability is noted by the presence of closed trajectories, indicating that a particle's trajectory can be perpetuated indefinitely. Trajectories in close proximity to the origin are deemed more stable, as they maintain consistent position and momentum throughout the particle's trajectory lifespan, as depicted in Figure 2.3 (1). Although certain trajectories may not exhibit closure due to limited iterations, they are still regarded as closed, as shown in Figure 2.3 (2). Conversely, particle trajectories that intersect with other trajectories or exhibit non-receptive/chaotic behavior, such as lacking consistent position and momentum as shown in Figure 2.3 (3), are considered unstable.

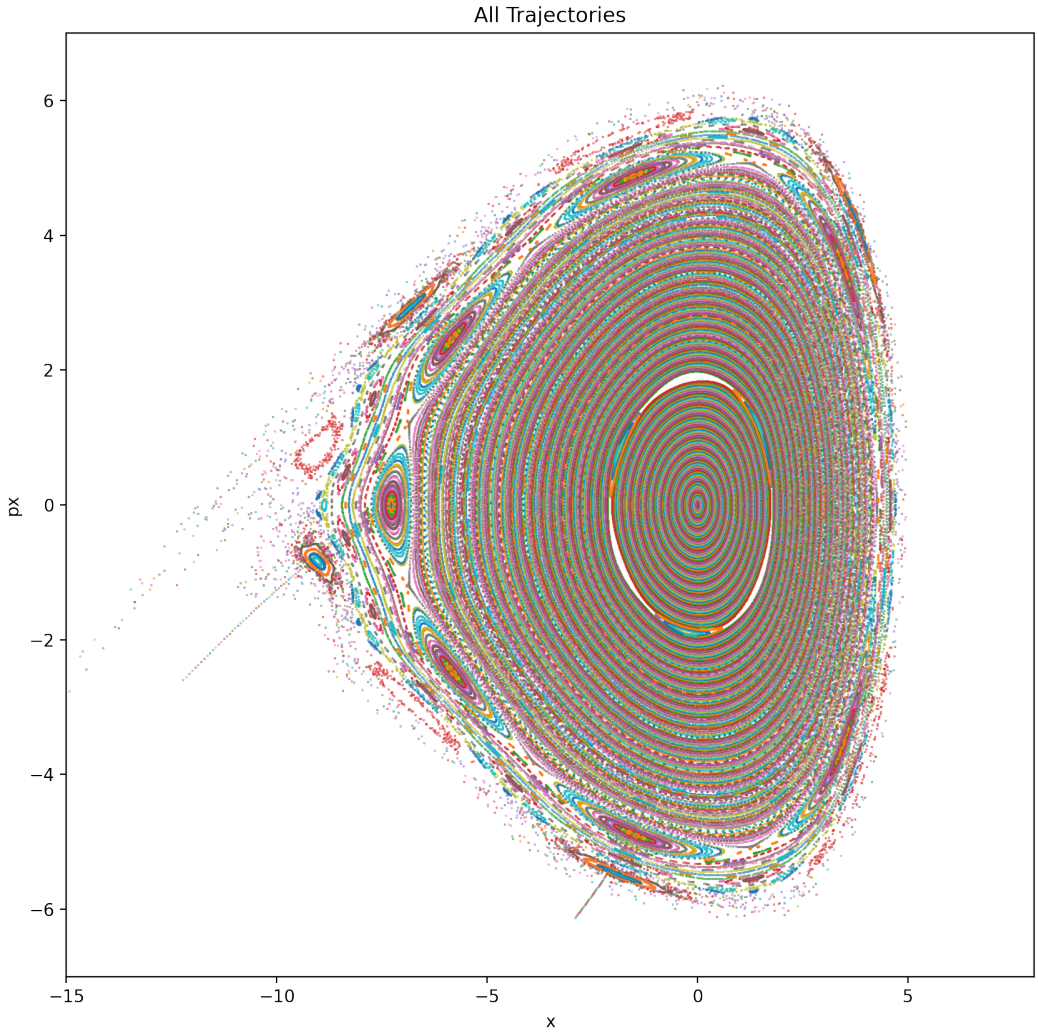

Figure 2.2: The simulated particle trajectory data set from [7] with 400 individual trajectories mapped onto phase space, each with 1,000 points with a (x, px) coordinate system, position and momentum.

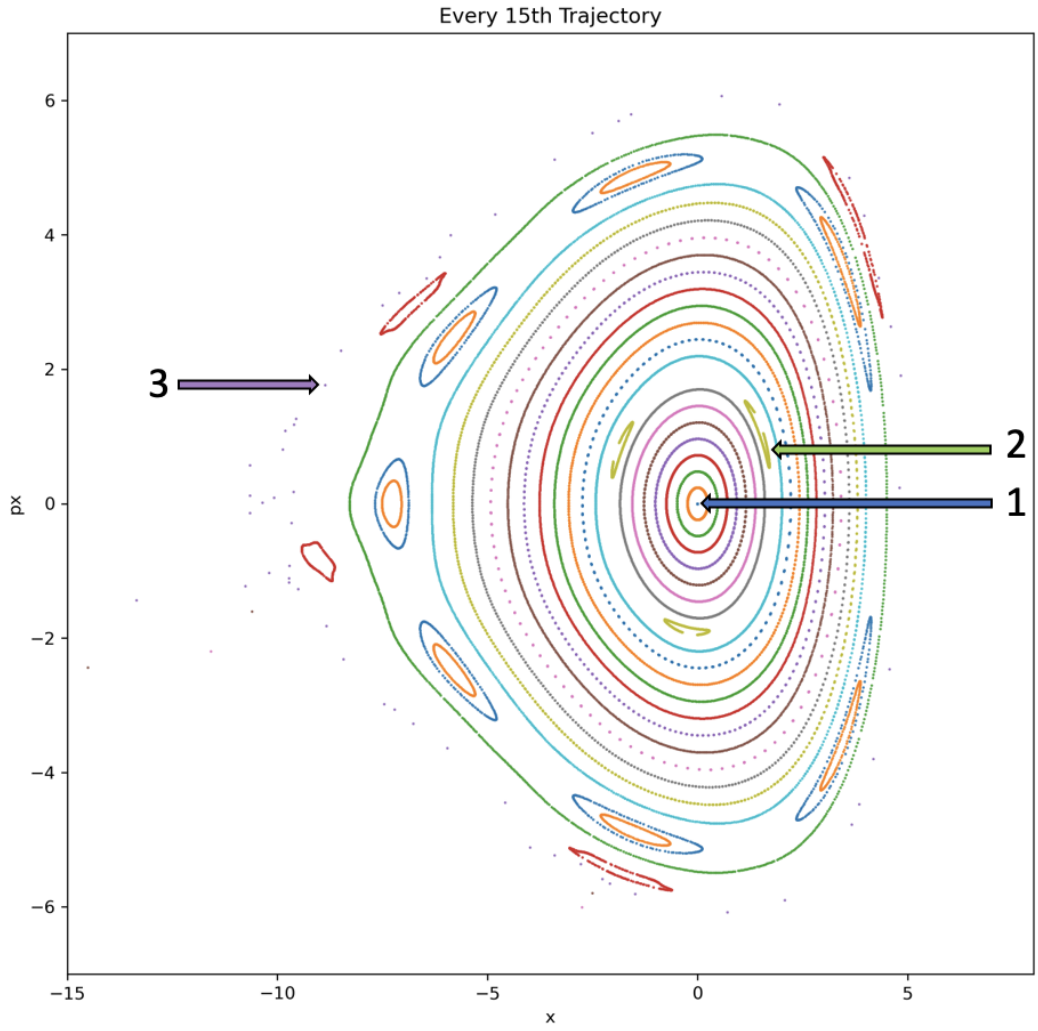

Figure 2.3: Same simulated data from Figure 2.2 but every 15^{th} trajectory mapped onto phase space, position and momentum. (1) Origin, trajectories close to this value will be considered stable (2) Islands or dotted particle trajectories that do not trail off nor exhibit chaotic behavior are still considered to be a closed trajectory (3) Chaotic particle trajectory, unstable region, ideally we want to avoid this area

Chapter 3

Hamiltonian Equations

Hamiltonian mechanics was formulated in the early 19^{th} century as an approach to model the motion of particles in a physical system [19]. A Hamiltonian is the sum of the kinetic energy, T, and potential energy, U, which is equal to the total energy of the system, E_{total}, as in Equation 3.1[6].

$$\mathcal{H} = T + U = E_{total} \tag{3.1}$$

The Hamiltonian equations consist of a set of first-order differential equations that relate the positions and momenta of particles in a system to their associated time derivatives. Let t be the time step, $\mathbf{q}$ be the position of the particle, and $\mathbf{p}$ be corresponding momentum. Then the Hamiltonian equations are given by:

$$\frac{d\mathbf{q}}{dt} = \frac{\partial \mathcal{H}}{\partial \mathbf{p}} \ , \ \frac{d\mathbf{p}}{dt} = -\frac{\partial \mathcal{H}}{\partial \mathbf{q}} \tag{3.2}$$

$\mathcal{H}$ conserves energy in a given system if Equations 3.2 are satisfied. The Hamiltonian equations are formulated to allow for the elimination for cyclic variables which reduces the number of degrees of freedom [19]. This is ideal when dealing with physical a system that exhibits a

circular behavior.

3.1 Deriving Hamiltonian's Equations

In classical mechanics, many dynamical systems are Lagrangian. The Lagrangian, $\mathcal{L}$, is a quantity that characterizes a physical system and is given by,

$$\mathcal{L} = T - V \tag{3.3}$$

where T is the kinetic energy and V is the potential energy. Consider a system with a generalized position vector, $\mathbf{q} = (q_0, q_1, ..., q_n)$. The Lagrangian of the system is a function of $\mathbf{q}, \dot{\mathbf{q}}$, and t, where $\dot{\mathbf{q}}$ is the time derivative of $\mathbf{q}$ with respect to t. The energy, $E(\mathbf{q}, \dot{\mathbf{q}}, t)$, of a system is given by [13],

$$E(\mathbf{q}, \dot{\mathbf{q}}, t) \equiv \frac{\partial \mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}, t)}{\partial \dot{\mathbf{q}}} \dot{\mathbf{q}} - \mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}, t) \tag{3.4}$$

and let

$$\frac{\partial \mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}, t)}{\partial \dot{\mathbf{q}}} \equiv \mathbf{p} \tag{3.5}$$

So, $\dot{\mathbf{q}}$ is written as $\mathbf{p}$ which is traditionally the generalized momentum or the conjugate momentum associated with the position $\mathbf{q}$ [13]. We can solve Equation 3.5 for $\dot{\mathbf{q}}$ in terms of $\mathbf{q}$, $\mathbf{p}$, and t resulting in the function $\dot{\mathbf{q}}(\mathbf{q}, \mathbf{p}, t)$. We replace the term $\frac{\partial \mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}, t)}{\partial \dot{\mathbf{q}}}$ with $\mathbf{p}$ in Equation 3.4 thus yielding the Hamiltonian function, $\mathcal{H}$, below in terms of $\mathbf{q}$, $\mathbf{p}$, and t.

$$\mathcal{H}(\mathbf{q}, \mathbf{p}, t) \equiv \mathbf{p}\dot{\mathbf{q}}(\mathbf{q}, \mathbf{p}, t) - \mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}(\mathbf{q}, \mathbf{p}, t), t) \tag{3.6}$$

This Legendre transformation of $\mathcal{L}$ is defined as a mapping of $(\mathbf{q}, \dot{\mathbf{q}}) \rightarrow (\mathbf{q}, \mathbf{p})$. For n coordinates, $\mathbf{q} = (q_0, q_1, ..., q_n)$ and $\mathbf{p} = (p_0, p_1, ..., p_n)$, the new Hamiltonian $\mathcal{H}$ can be written as [19],

$$\begin{aligned} \mathcal{H}(\mathbf{q}, \mathbf{p}, t) &\equiv p_0\dot{q}_0(\mathbf{q}, \mathbf{p}, t) + p_1\dot{q}_1(\mathbf{q}, \mathbf{p}, t) + ... + p_n\dot{q}_n(\mathbf{q}, \mathbf{p}, t) - \mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}(\mathbf{q}, \mathbf{p}, t), t) \quad (3.7) \\ &\equiv \sum_{i=0}^{n} \mathbf{p}_i\dot{\mathbf{q}}_i(\mathbf{q}, \mathbf{p}, t) - \mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}(\mathbf{q}, \mathbf{p}, t), t) \quad (3.8) \end{aligned}$$

where $\mathcal{H}$ is now a function of $2n$ coordinates and

$$\frac{\partial \mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}(\mathbf{q}, \mathbf{p}, t), t)}{\partial \dot{\mathbf{q}}_i} \equiv \mathbf{p}_i \quad (3.9)$$

Looking at the partial derivative of Equation 3.6 with respect to $\mathbf{p}$ we get the following,

$$\frac{\partial \mathcal{H}(\mathbf{q}, \mathbf{p}, t)}{\partial \mathbf{p}} = \frac{\partial(\mathbf{p}\dot{\mathbf{q}}(\mathbf{q}, \mathbf{p}, t))}{\partial \mathbf{p}} - \frac{\partial \mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}(\mathbf{q}, \mathbf{p}, t), t)}{\partial \mathbf{p}} \quad (3.10)$$

Since $\mathbf{p}\dot{\mathbf{q}}(\mathbf{q}, \mathbf{p}, t)$ is dependant on $\mathbf{p}$, we get that

$$\frac{\partial(\mathbf{p}\dot{\mathbf{q}}(\mathbf{q}, \mathbf{p}, t))}{\partial \mathbf{p}} = \dot{\mathbf{q}}(\mathbf{q}, \mathbf{p}, t) + \mathbf{p}\frac{\partial \dot{\mathbf{q}}(\mathbf{q}, \mathbf{p}, t)}{\partial \mathbf{p}} \quad (3.11)$$

Also, $\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}(\mathbf{q}, \mathbf{p}, t), t)$ depends on $\mathbf{p}$ since it is dependant on $\dot{\mathbf{q}}$, thus,

$$\frac{\partial \mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}(\mathbf{q}, \mathbf{p}, t), t)}{\partial \mathbf{p}} = \frac{\partial \mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}(\mathbf{q}, \mathbf{p}, t), t)}{\partial \dot{\mathbf{q}}} \frac{\partial \dot{\mathbf{q}}(\mathbf{q}, \mathbf{p}, t)}{\partial \mathbf{p}} \quad (3.12)$$

By substituting Equations 3.11 and 3.12 into 3.10 and using the fact of Equation 3.9, we obtain

$$\frac{\partial \mathcal{H}(\mathbf{q}, \mathbf{p}, t)}{\partial \mathbf{p}} = \dot{\mathbf{q}}(\mathbf{q}, \mathbf{p}, t) + \mathbf{p}\frac{\partial \dot{\mathbf{q}}(\mathbf{q}, \mathbf{p}, t)}{\partial \mathbf{p}} - \mathbf{p}\frac{\partial \dot{\mathbf{q}}(\mathbf{q}, \mathbf{p}, t)}{\partial \mathbf{p}} \quad (3.13)$$

Simplifying notation we get,

$$\frac{\partial \mathcal{H}}{\partial \mathbf{p}} = \dot{\mathbf{q}} + \mathbf{p}\frac{\partial \dot{\mathbf{q}}}{\partial \mathbf{p}} - \mathbf{p}\frac{\partial \dot{\mathbf{q}}}{\partial \mathbf{p}} \tag{3.14}$$

$$= \dot{\mathbf{q}} \tag{3.15}$$

$$= \frac{d\mathbf{q}}{dt} \tag{3.16}$$

Similarly, the partial derivative of Equation 3.6 with respect to $\mathbf{q}$, we get the following,

$$\frac{\partial \mathcal{H}(\mathbf{q}, \mathbf{p}, t)}{\partial \mathbf{q}} = \frac{\partial(\mathbf{p}\dot{\mathbf{q}}(\mathbf{q}, \mathbf{p}, t))}{\partial \mathbf{q}} - \frac{\partial \mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}(\mathbf{q}, \mathbf{p}, t), t)}{\partial \mathbf{q}} \tag{3.17}$$

Since $\mathbf{p}\dot{\mathbf{q}}(\mathbf{q}, \mathbf{p}, t)$ is dependant on $\mathbf{q}$ we get that

$$\frac{\partial(\mathbf{p}\dot{\mathbf{q}}(\mathbf{q}, \mathbf{p}, t))}{\partial \mathbf{q}} = \mathbf{p}\frac{\partial \dot{\mathbf{q}}(\mathbf{q}, \mathbf{p}, t)}{\partial \mathbf{q}} \tag{3.18}$$

$\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}(\mathbf{q}, \mathbf{p}, t), t)$ also depends on $\mathbf{q}$ thus,

$$\frac{\partial \mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}(\mathbf{q}, \mathbf{p}, t), t)}{\partial \mathbf{q}} = \frac{\partial \mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}(\mathbf{q}, \mathbf{p}, t))}{\partial \mathbf{q}} + \frac{\partial \mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}(\mathbf{q}, \mathbf{p}, t))}{\partial \dot{\mathbf{q}}}\frac{\partial \dot{\mathbf{q}}(\mathbf{q}, \mathbf{p}, t)}{\partial \mathbf{q}} \tag{3.19}$$

The Euler-Lagrange equation in n dimensions is,

$$\frac{\partial \mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}(\mathbf{q}, \mathbf{p}, t), t)}{\partial \mathbf{q}} = \frac{d}{dt}\frac{\partial \mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}(\mathbf{q}, \mathbf{p}, t), t)}{\partial \dot{\mathbf{q}}} \tag{3.20}$$

$$\rightarrow \dot{\mathbf{p}} \tag{3.21}$$

By substituting Equations 3.18 and 3.19 into 3.17 and using the fact of Equations 3.9 and 3.20, and simplifying notation we obtain

$$\begin{aligned} \frac{\partial \mathcal{H}}{\partial \mathbf{q}} &= \mathbf{p}\frac{\partial \dot{\mathbf{q}}}{\partial \mathbf{p}} - (\dot{\mathbf{p}} + \mathbf{p}\frac{\partial \dot{\mathbf{q}}}{\partial \mathbf{q}}) && (3.22) \\ &= -\dot{\mathbf{p}} && (3.23) \\ &= -\frac{d\mathbf{p}}{dt} && (3.24) \end{aligned}$$

Thus, the Hamiltonian equations in $2n$ coordinate are,

$$\frac{\partial \mathcal{H}}{\partial \mathbf{p}} = \dot{\mathbf{q}} = \frac{d\mathbf{q}}{dt} \tag{3.25}$$

and

$$-\frac{\partial \mathcal{H}}{\partial \mathbf{q}} = \dot{\mathbf{p}} = \frac{d\mathbf{p}}{dt} \tag{3.26}$$

The Hamiltonian equations, 3.25 and 3.26, form the symplectic gradient, a canonical transformation that preserves the form of the Hamiltonian equations and tells us the direction of the moving coordinates, $\mathbf{S}_{\mathcal{H}}$ below [17],

$$\mathbf{S}_{\mathcal{H}} = (\frac{\partial \mathcal{H}}{\partial \mathbf{p}}, -\frac{\partial \mathcal{H}}{\partial \mathbf{q}}) \tag{3.27}$$

The symplectic gradient can be integrated with respect to time, as in Equation 3.28, to solve the Hamiltonian equations by finding the next iteration of coordinates, (q_1, p_1), given (q_0, p_0) as the initial value.

$$(q_1, p_1) = (q_0, p_0) + \int_{t_0}^{t_1} (\frac{\partial \mathcal{H}}{\partial \mathbf{p}}, -\frac{\partial \mathcal{H}}{\partial \mathbf{q}})dt \tag{3.28}$$

3.2 Simplified System

A toy Hamiltonian system, $\mathcal{H}(x, y, p_x, p_y)$, is used to model the behavior of electrons within a particle accelerator. The motion of a particle is governed by the Hamiltonian,

$$\mathcal{H}(x, y, p_x, p_y) = E_{total} \tag{3.29}$$

Consider the Hamiltonian $\mathcal{H}(x, y, p_x, p_y)$ given by

$$\mathcal{H}(x, y, p_x, p_y) = \frac{1}{2}(p_x^2 + p_y^2 + x^2 + y^2) + x^2 y - \frac{1}{3}y^3 \tag{3.30}$$

where x is the position of a particle and the p_x is the associated momentum. This Hamiltonian represents a particle in a accelerator consisting of only vacuum tubes (no magnetic fields), a sextupole magnet ($V(x) \propto x^3$), and a focusing element which can focus particles in both planes simultaneously. This type of focusing element is constructed mathematically for the purpose of studying the nonlinear dynamics in the presence of a sextupole. Although particle tracking is conventionally conducted in multidimensional spaces, we aim to demonstrate a proof of concept in 2 dimensions for the sake of simplification. Thus, the Hamiltonian becomes Equation 3.31. Let the Hamiltonian, $\mathcal{H}(x, p_x)$, be a function such that

$$\begin{aligned} \mathcal{H}(x, p_x) &= \frac{1}{2}(x^2 + p_x^2) - \frac{\alpha}{3}x^3 && (3.31) \\ &= \frac{1}{2}(x^2 + p_x^2) + V(x) && (3.32) \end{aligned}$$

where α is a constant parameter and $V(x)$ is the sextupole potential. The toy system, Equation 3.31, has all the elements that exist in a real particle accelerator but with the addition of a certain potential, $V(x)$ which exhibits non-linear behavior. Thus Equation 3.31

cannot be solved exactly by traditional methods of solving partial differential equations. The toy system is discretized, as it operates with discrete time points and discrete changes in time. The time step for the system is set at $dt = 1$, representing a particle's complete single revolution around the particle accelerator. Since we cannot directly solve the Hamiltonian in its continuous form, iterative numerical integration methods are employed to solve the Hamiltonian. It is important to note that real-world systems are continuous. The associated Hamiltonian equations for Equation 3.31 are,

$$\frac{\partial \mathcal{H}}{\partial \mathbf{p_x}} = \frac{\partial \mathbf{x}}{\partial t} \ , \ -\frac{\partial \mathcal{H}}{\partial \mathbf{x}} = \frac{\partial \mathbf{p_x}}{\partial t} \tag{3.33}$$

The proposed toy Hamiltonian, Equation 3.31, should adhere to the conservation of energy following successful training of the HNN on a chosen dataset of particle trajectories. We parameterize the Hamiltonian with a neural network then test the neural network in its ability to learn the Hamiltonian equation, Equation 3.31, from the given data. A comparison of the total energy is made between the predicted trajectories, generated based on a given initial position and momentum, and the ground truth. By inserting the predicted data points, outputs obtained from the HNN, into our toy Hamiltonian, we can verify if Equations 3.33 are satisfied. If it is, then the Hamiltonian is conserved. Successful training an HNN to learn and reproduce Hamiltonian mechanics will enable the use of a HNN for fast optimization of beam dynamics through out the design phase, without the need of computationally expensive simulations which integrate the Hamiltonian directly.

Chapter 4

Machine Learning

Machine learning enables computer systems to learn and adapt autonomously without explicit instructions using algorithms that allow them to learn from statistical data. With the advancement of computers in the 1950's, the idea that computers could program themselves became very appealing to research scientists. The concept of a computer working like a neuron in the brain gave rise to neural networks [14]. Neural networks are a specific group of algorithms in machine learning that model data using neuron like structure with hidden layers. These hidden layers, as shown in Figure 4.1, extract data from the input neuron layer and produce values for the output neuron layer. This results in the discovery of the various relationships between different inputs. This very basic fully connected multi-layer neural network, shown in Figure 4.1 part (b), is also known as a multi-layer perceptron (MLP). The first multilayered unsupervised network was created in 1975 [14]. Today, neural networks are widely used in various areas of research such as predicting new virus strains to weather forecasting. Machine learning for particle tracking may offer a more efficient way to predict particle trajectories compared to traditional numerical tracking methods. To compare the performance of the HNN, we also train and test a MLP and Dissipative Hamiltonian Neural Network (DHNN) on the same dataset. The predicted trajectories from each neural network

are then mapped in phase space. The squared errors and total energies of the system are also calculated and graphed.

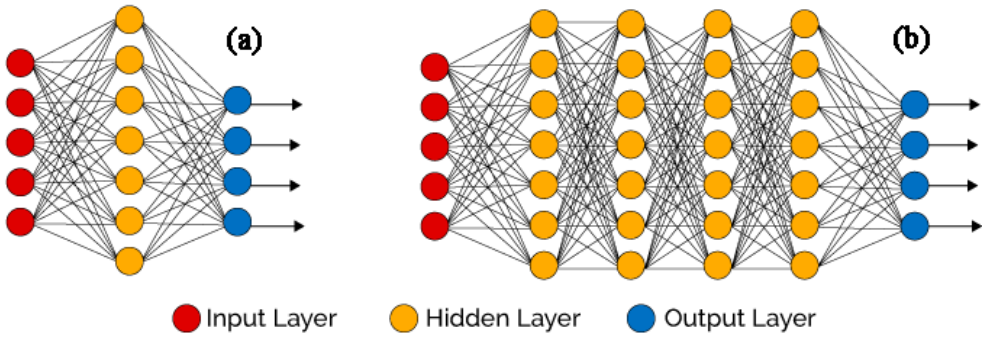

Figure 4.1: Schematic of the simple structure of a basic feed forward neural network. (a) one hidden layer, (b) multiple hidden layers [4]

4.1 Hamiltonian Neural Network

There are three main types of machine learning: supervised, unsupervised, and semi-supervised. In our case, we will be utilizing a supervised approach, since the Hamiltonian Neural Network (HNN) is trained on a labeled dataset, where each input has an associated output. The primary objective of the algorithm is to discern the relationship between these values, specifically in the context of the conservation of energy. The HNN utilizes its understanding of a pre-trained dataset, consisting of a particles position and momentum, to predict the next set of time derivatives. We then compare those predicted values to that of the reserved ground truth trajectory data. The total energy is created by plugging in the predicted trajectory data from the neural networks into our toy Hamiltonian and checks how well the energy in the system is conserved.

A Hamiltonian Neural Network consists of various steps as shown in Figure 4.2. The first set of training coordinates, (x_0, p_{x_0}), are passed through an instance of a differentiable model which will then return a single like "energy-like" value, $\mathcal{H}$, similar to a Hamiltonian value as in chapter three [17]. $\mathcal{H}$ is passed through another instance of the same differentiable model

and returns the sum of gradients of outputs with respect to inputs [17]. This new gradient is split and swapped to create the symplectic gradient, given in Equation 4.1.

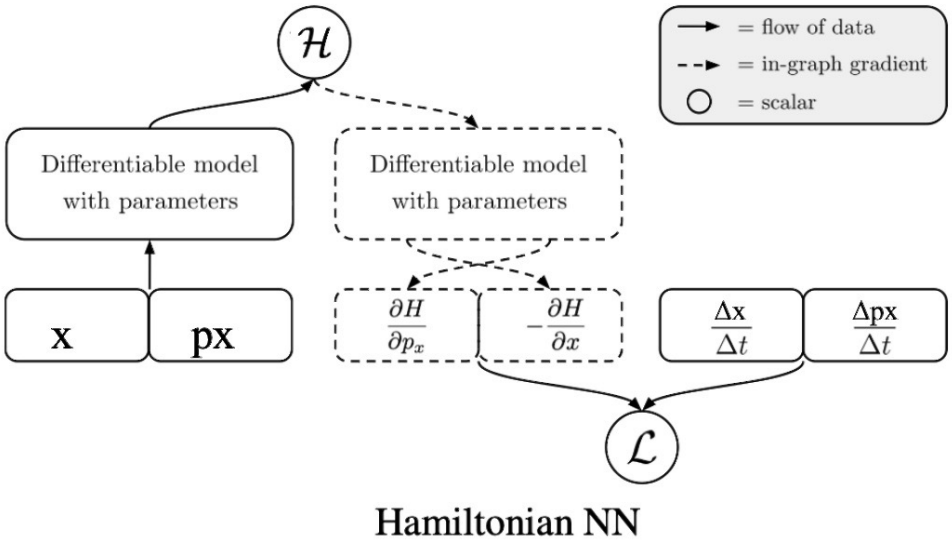

Figure 4.2: The general set up of how the Hamiltonian equations are used within the neural network. It consists of two forward passes through a baseline network with the outputs inserted into a modified loss function, Equation 4.2. Adapted from [17].

$$S_{\mathcal{H}} = (\frac{\partial \mathcal{H}}{\partial p_x}, -\frac{\partial \mathcal{H}}{\partial x}) \tag{4.1}$$

Once the symplectic gradient is mapped to its approximate time derivative, it is then used to compute and optimize the L_2 loss shown in Equation 4.2, [17]. This modified loss function ensures that the Hamiltonian equations govern the neural network.

$$L_{HNN} = \| \frac{\partial \mathcal{H}}{\partial p_x} - \frac{\partial x}{\partial t} \|_2 + \| \frac{\partial \mathcal{H}}{\partial x} + \frac{\partial p_x}{\partial t} \|_2 \tag{4.2}$$

To train the HNN, we minimize the error between the time derivative of the coordinates, $(\frac{\partial x}{\partial t}, \frac{\partial p_x}{\partial t})$, and symplectic gradient, Equation 4.1. Then the neural network is told to only predict the gradient if $\frac{\partial \mathcal{H}}{\partial p_x} = \frac{\partial x}{\partial t}$ and $-\frac{\partial \mathcal{H}}{\partial x} = \frac{\partial p_x}{\partial t}$, satisfying the Hamiltonian Equations

which implies that energy is conserved. The key distinction between a basic neural network and a HNN lies in the loss functions.

4.2 Dissipative Hamiltionan Neural Network

A Dissipative Hamiltionan Neural Network (DHNN) combines Hamiltonian mechanics and Helmholtz decomposition, also known as the fundamental theorem of vector calculus. This gives a DHNN the ability to separate dissipative effects such as friction and conserved quantities such as energy [17]. A DHNN has a similar algorithm but with a different loss function given by,

$$L_{DHNN} = \| \left(\frac{\partial H}{\partial p_x} + \frac{\partial D}{\partial x}\right) - \frac{\partial x}{\partial t} \|_2 + \| \left(\frac{\partial H}{\partial x} + \frac{\partial D}{\partial p_x}\right) - \frac{\partial p_x}{\partial t} \|_2 \tag{4.3}$$

4.3 Conclusion

We seek to evaluate the ability of the HNN to learn Hamilton's equations (referring to Equation 3.2). To accomplish this, we conduct a comparison of the HNN's performance with the ground truth dataset, which holds reserved data that the neural network has not been exposed to during training. Additionally, we train and test a DHNN and a MLP on the same dataset, aiming to identify any significant differences among these neural network architectures in comparison to the HNN. This will allows us to determine the proficiency of the HNN, DHNN, and MLP in learning Hamilton's equations and their performance relative to the ground truth dataset.

Chapter 5

Analysis and Results

5.1 The Data

The simulated trajectory data used in this project was created by using a numerical integration process where a starting point, (x_1, p_{x_1}), was chosen and the equations of motion were given. These parameters were then use to solve the initial value problem, Equation 5.1. Let (x_2, p_{x_2}) be the next set of coordinates for a particle's trajectory such that,

$$(x_2, p_{x_2}) = (x_1, p_{x_1}) + \int_{t_1}^{t_2} (\frac{\partial \mathcal{H}}{\partial p_x}, -\frac{\partial \mathcal{H}}{\partial x})\, dt \tag{5.1}$$

where t is the generalized time step. Note that one step in t is considered one full revolution around the particle accelerator. This numerical tracking processes was then iterated 1000 times each for 400 individual particle trajectories. It is important to acknowledge that the trajectory number serves as an arbitrary, numeric index for labeling trajectory data. Trajectory 1 is designated as the particle trajectory originating from the origin, while trajectory 400 represents the outermost particle trajectory in the dataset. The generated data is mapped onto a phase space plane in Figure 5.1.

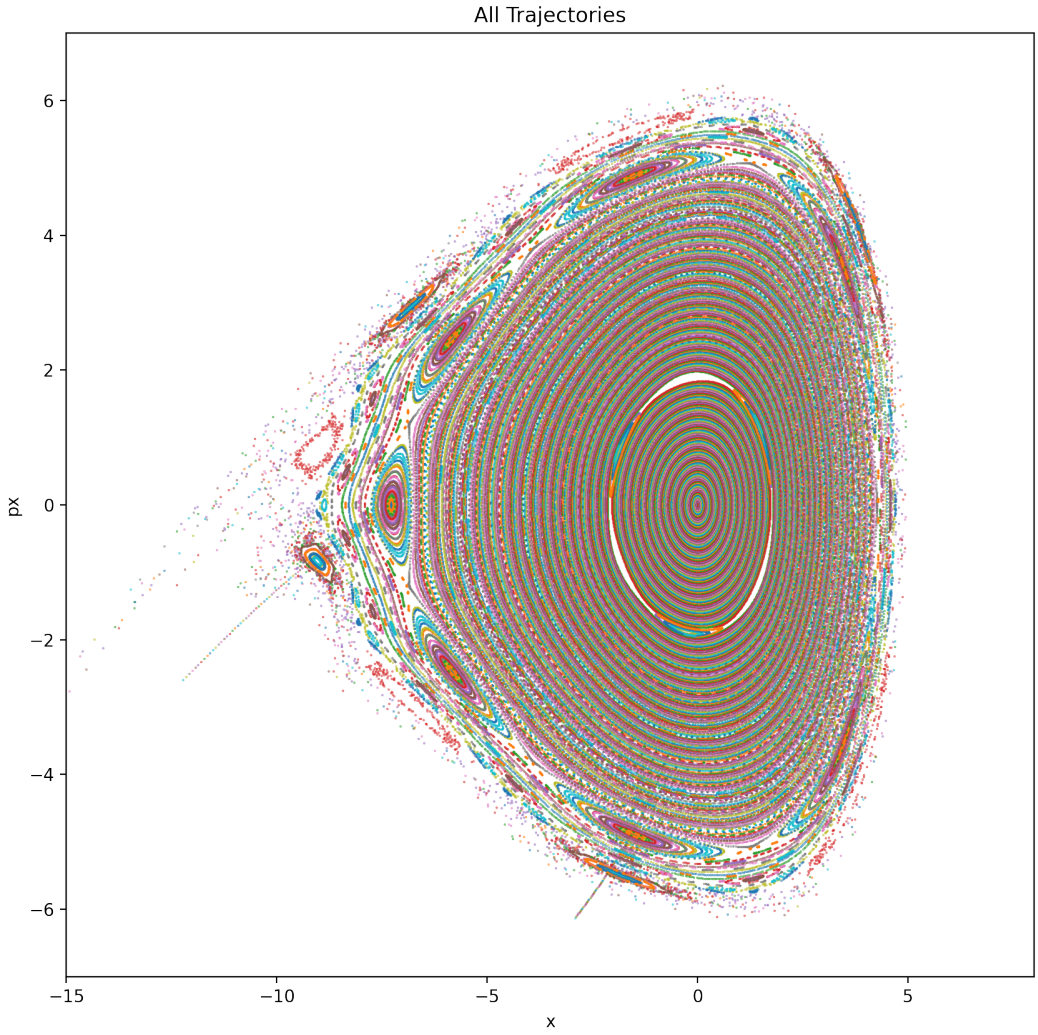

Figure 5.1: All 400 individual particle trajectories with 1,000 iterations per trajectory. Each iteration is one full time step, which is considered to be one complete rotation around the particle accelerator

Experiments begin on trajectory 20 since any trajectory positioned prior to trajectory 20 is already presumed to have a stable orbit. While there is an interest in the neural networks predicting the coordinates accurately, there is also are interest in seeing if they can predict the long term behavior, if the trajectory remains closed or not, sufficiently. Thus, an approx-

imation of the particle's trajectory will suffice. The primary focus is on discerning whether the beam has been successfully maintained within the region of stability. Again, the region of stability in this research is noted by the presence of closed trajectories, indicating that a particle's trajectory can be perpetuated indefinitely. Looking at only enclosed trajectories, and those that do not exhibit a dissipating nature, we choose data sets from trajectories 20 to 300 to employ in the notebook.

5.2 Idea behind MainExp.py

MainExp.py, from GitHub [12], was created with the purpose to compile and organize all data, training, testing, and results with code adapted from [17]. All mathematical functions written can be found in **Meitz_utils.py**, from GitHub [12].

5.2.1 Average Amplitude

MainExp.py iterates for i trajectories as i experiments for $i = 20, 21, ..., 300$. Let D_i be the set of data points in one single arbitrarily chosen particle trajectory given by,

$$D_i = \{(x_1, p_{x_1}), (x_2, p_{x_2}), ..., (x_{1000}, p_{x_{1000}})\} \tag{5.2}$$

where each (x_m, p_{x_m}) for $m = 1, 2, ..., 1000$ is the associated position and momentum. First, the notebook will calculate the approximate average amplitude of each trajectory by finding the Euclidean distance of each data point, $(x_m, p_{x_m}) \in D_i$, in reference to the origin, $(0, 0)$, in the function *get_AMP*. Let $\mathcal{A}_i$ be the average amplitude for the i^{th} trajectory with 1000

coordinates calculated by,

$$\mathcal{A}_i = \frac{1}{1000}\sum_{m=1}^{1000}\sqrt{(x_m-0)^2+(p_{x_m}-0)^2} \tag{5.3}$$

$$= \frac{1}{1000}\sum_{m=1}^{1000}\sqrt{(x_m)^2+(p_{x_m})^2} \tag{5.4}$$

A new array is made where each i^{th} trajectory then has an associated average amplitude.

5.2.2 Scaled Trajectories

During the initial phases of this research, both training and testing were conducted on the same trajectory where the dataset was divided into an 80/20 train to test ratio. Once the model parameters and training paradigm was optimized, we moved on to understand how well these models generalize to adjacent trajectories relative to the training trajectory. The optimization was evaluated to be sufficient once the predicted particle trajectories remained within the desired region of stability (e.i. in a closed trajectory).

The objective of the following procedure is to generate a list of trajectories that are scaled by the i^{th} trajectory that contains certain percentages. Given an i^{th} trajectory, we create scaled trajectories for testing which are proportionately larger or smaller to i. This allows us to evaluate how well the models generalize to other trajectories when trained on a specific i^{th} trajectory.

A designated percentile list, in our case $p = [0.9, 0.95, 1.0, 1.05, 1.10]$, is used to select certain amplitudes from the average amplitude array, $\mathcal{A}$. The function *get_scale* calculates the scaled amplitude by taking each $\mathcal{A}_n$ for $n = 20, 21, ..., 300$ and scaling it to a single arbitrarily chosen average amplitude $\mathcal{A}_i$. Let S_i be the scaled amplitude for each average amplitude of

trajectories given by,

$$S_i = \frac{\mathcal{A}_n}{\mathcal{A}_i} \tag{5.5}$$

All amplitude values are rounded to two decimal places. Then the function will choose the index of the trajectories where it's S_i value is in the percentage array p. This process is repeated 280 times until each individual trajectory has an associated percentage array with scaled amplitude values relative to itself, $\mathcal{A}_n$.

Lastly, the function *find_amp_index* will find all values where $S_i \in p$ and stores the i^{th} index in a array. If there exists more than one trajectory with the same scaled amplitude, the function will only store the first i^{th} index it finds. The scaled amplitude index array contains the i^{th} trajectory since $n = i$ then $S_i = 1$. The i^{th} trajectory data set is used for training and all other trajectory indexes are reserved for testing.

5.2.3 Subsequent Steps

The newly created scaled amplitude index array will run through training and testing for each HNN, DHNN, and MLP. All predicted trajectories and numerical results are stored in dictionaries that can easily be accessed.

5.3 Results

We run **MainExp.py** on trajectories 20 to 300 as trajectories below 20 are presumed to fall within the desired stability range, while those above 300 are expected to be excessively chaotic for the intended purpose.

We choose a random sample of ten data trajectories, shown in Figure 5.2, to compare their

predicted trajectories, squared errors, and total energies in Figures 5.3 to 5.21.

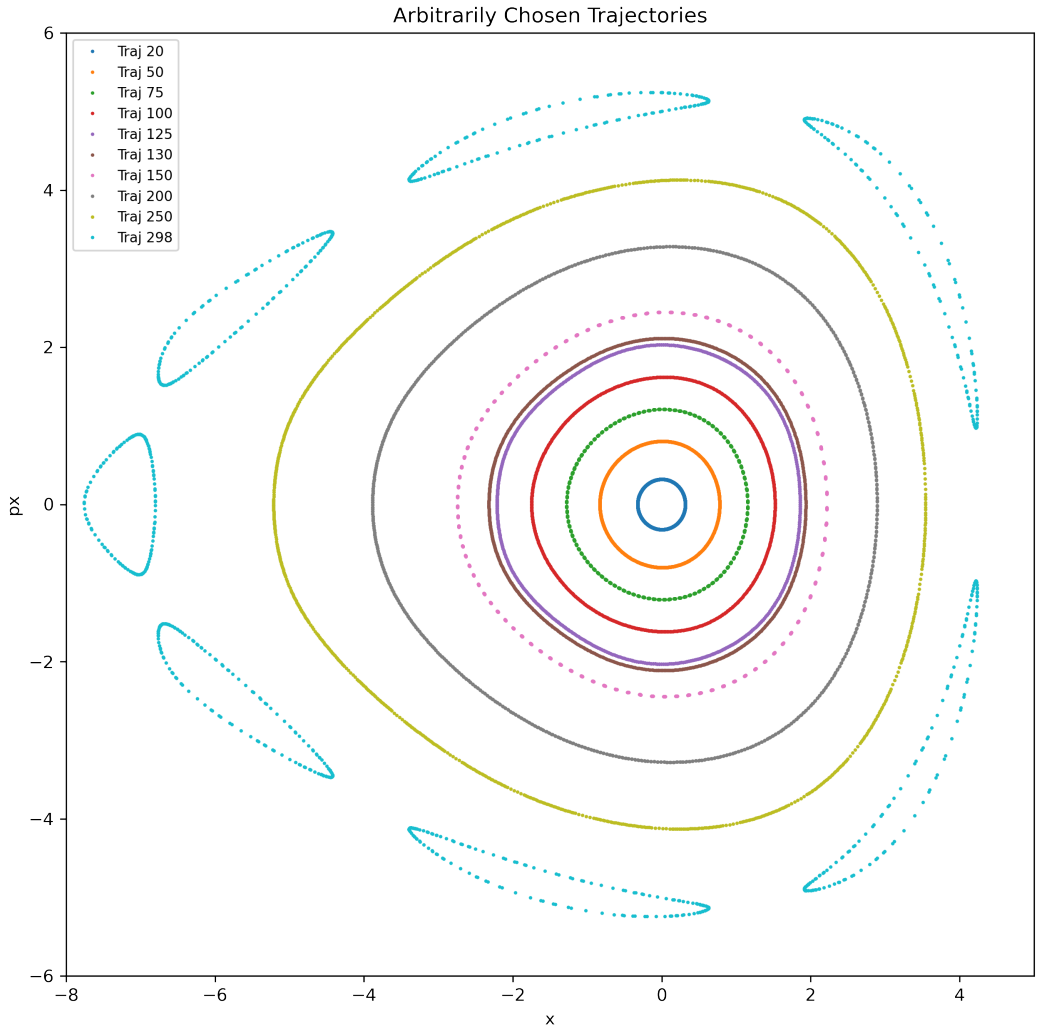

Figure 5.2: Arbitrarily chosen trajectories from the original dataset

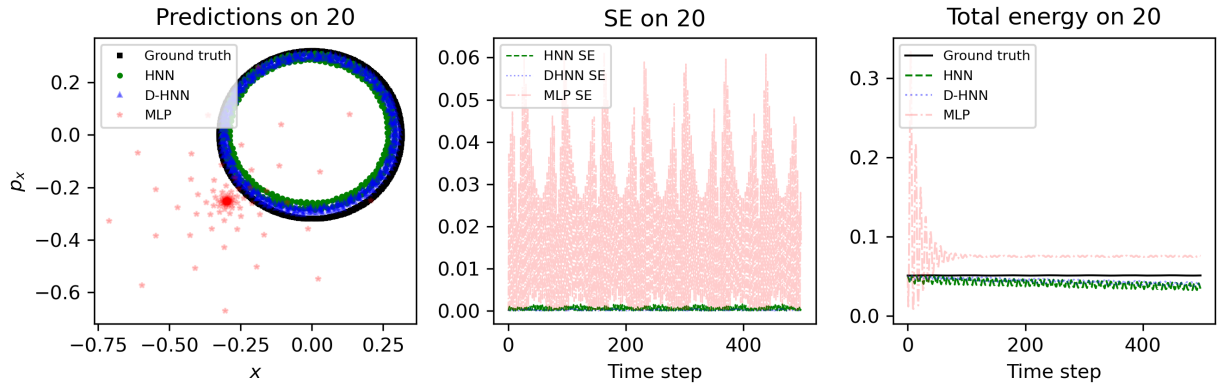

Figure 5.3: Experiment 20, Trial 20

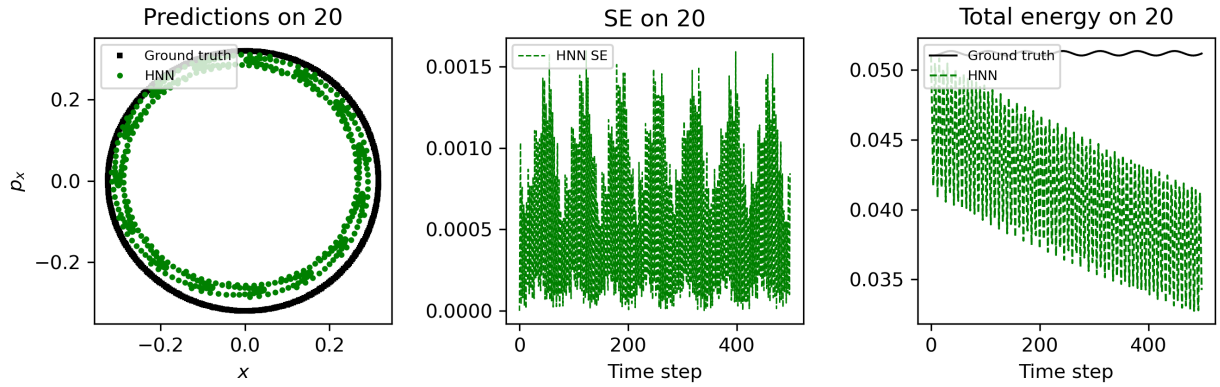

Figure 5.4: Experiment 20, Trial 20: HNN only

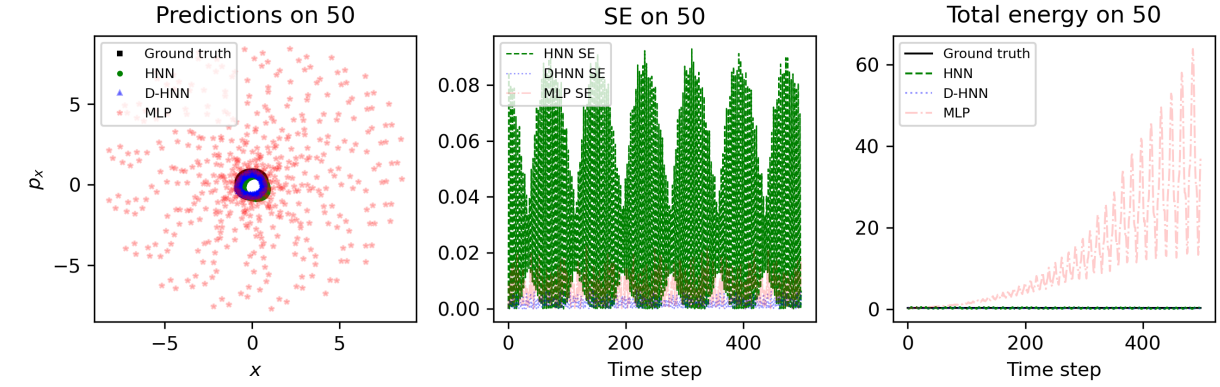

Figure 5.5: Experiment 50, Trial 50

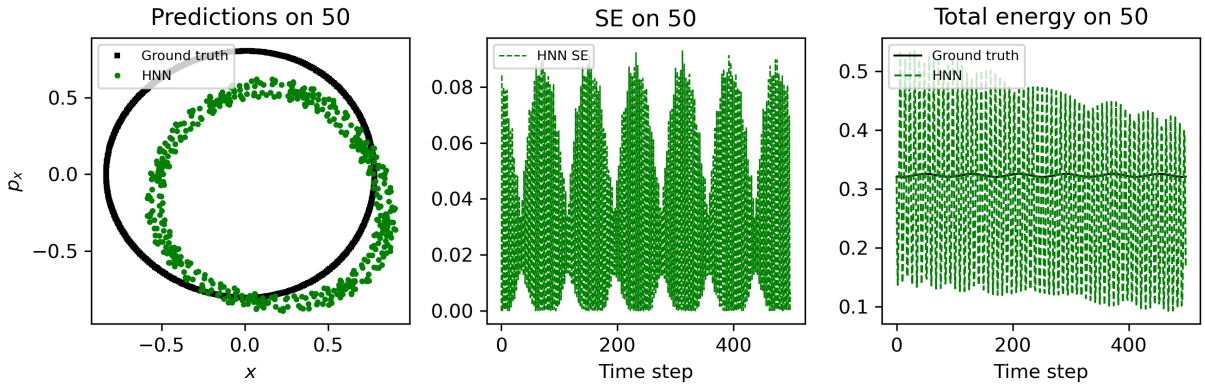

Figure 5.6: Experiment 50, Trial 50: HNN only

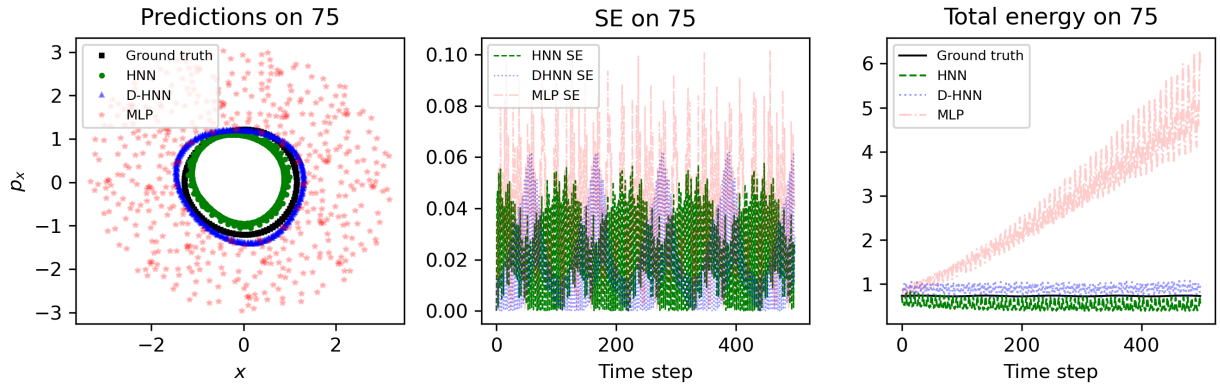

Figure 5.7: Experiment 75, Trial 75

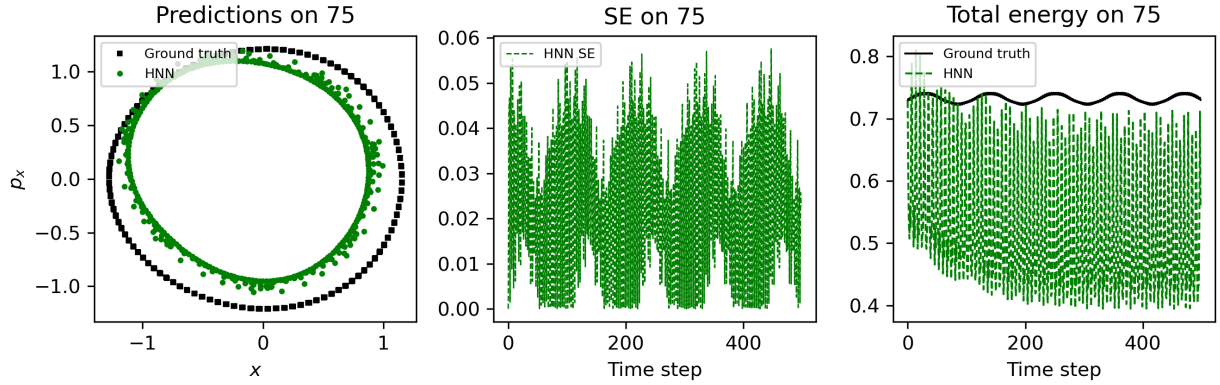

Figure 5.8: Experiment 75, Trial 75: HNN only

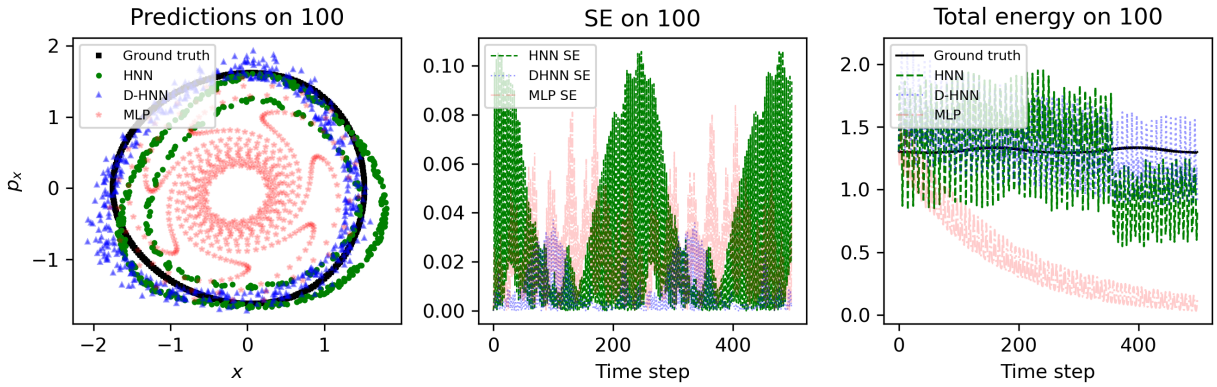

Figure 5.9: Experiment 100, Trial 100

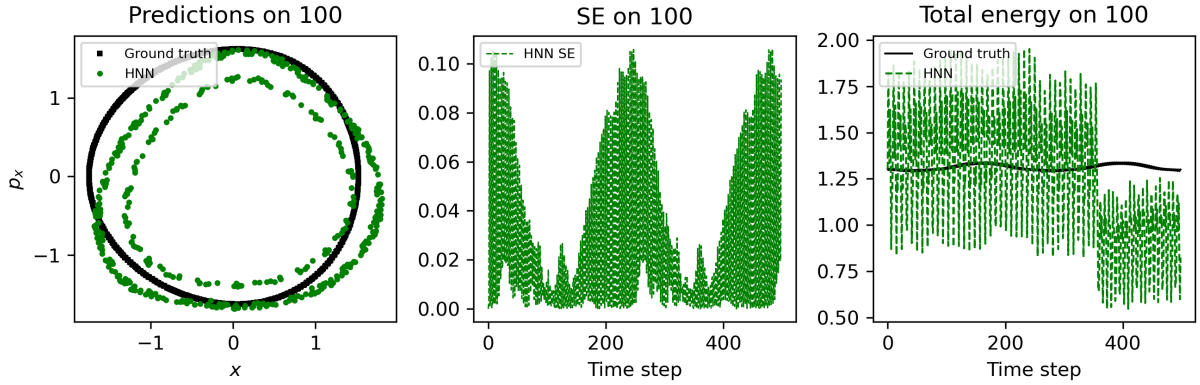

Figure 5.10: Experiment 100, Trial 100: HNN only

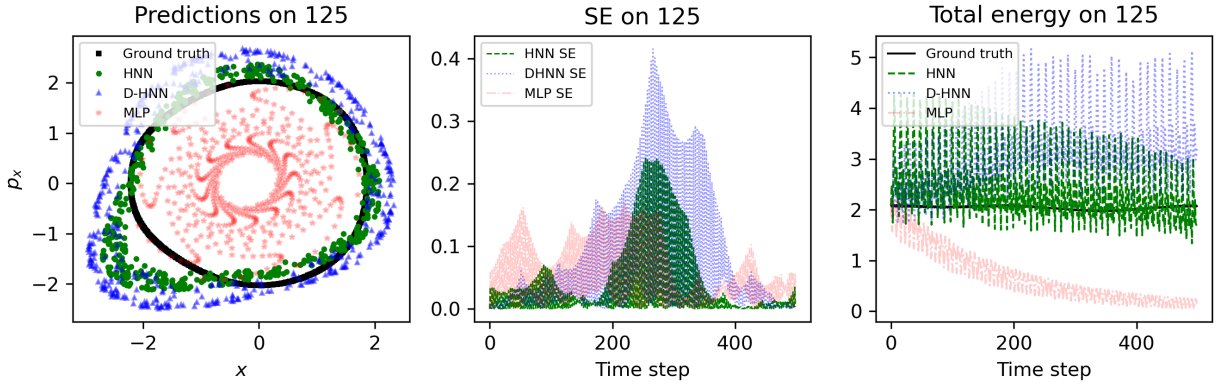

Figure 5.11: Experiment 125, Trial 125

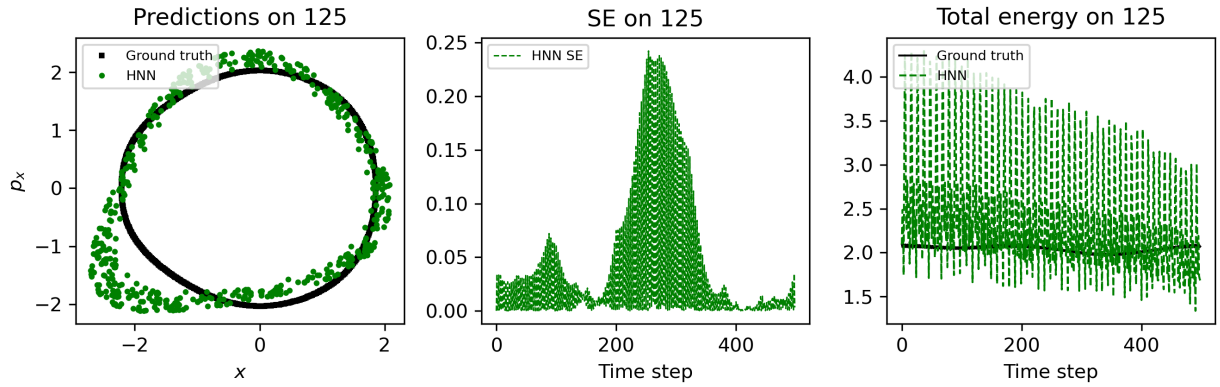

Figure 5.12: Experiment 125, Trial 125: HNN only

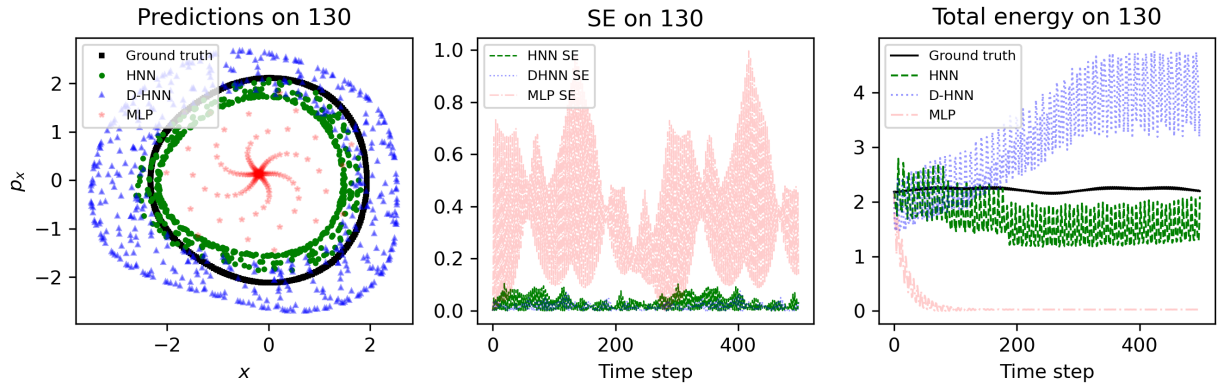

Figure 5.13: Experiment 130, Trial 130

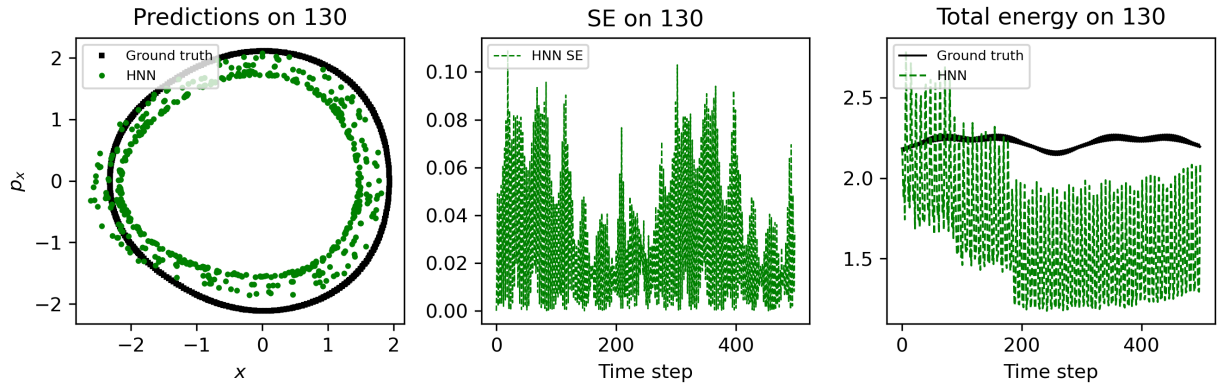

Figure 5.14: Experiment 130, Trial 130: HNN only

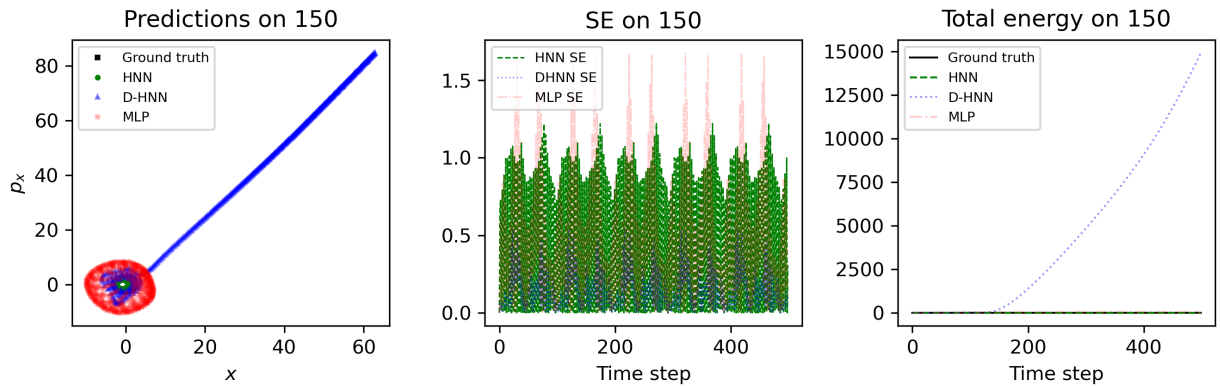

Figure 5.15: Experiment 150, Trial 150

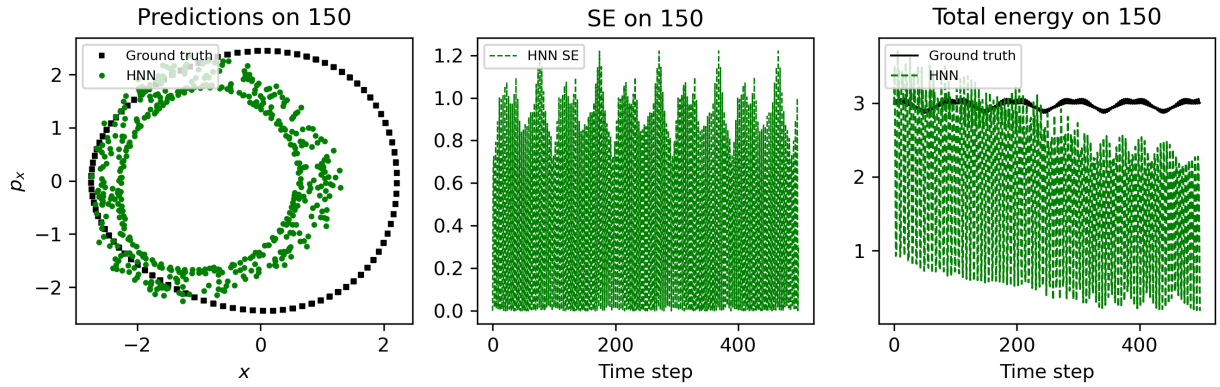

Figure 5.16: Experiment 150, Trial 150: HNN only

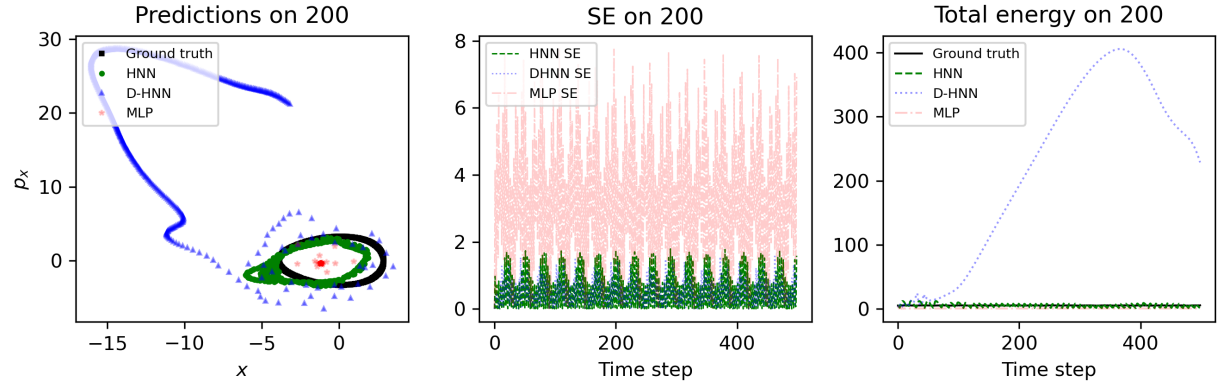

Figure 5.17: Experiment 200, Trial 200

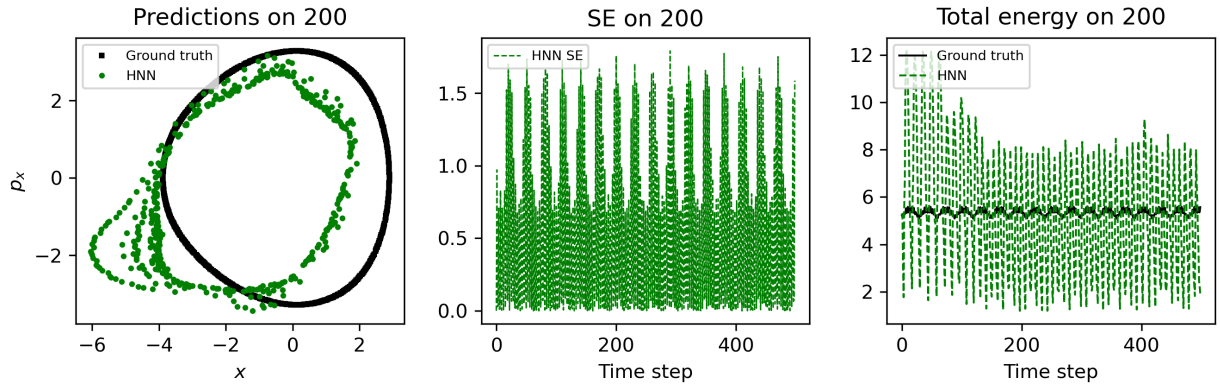

Figure 5.18: Experiment 200, Trial 200: HNN only

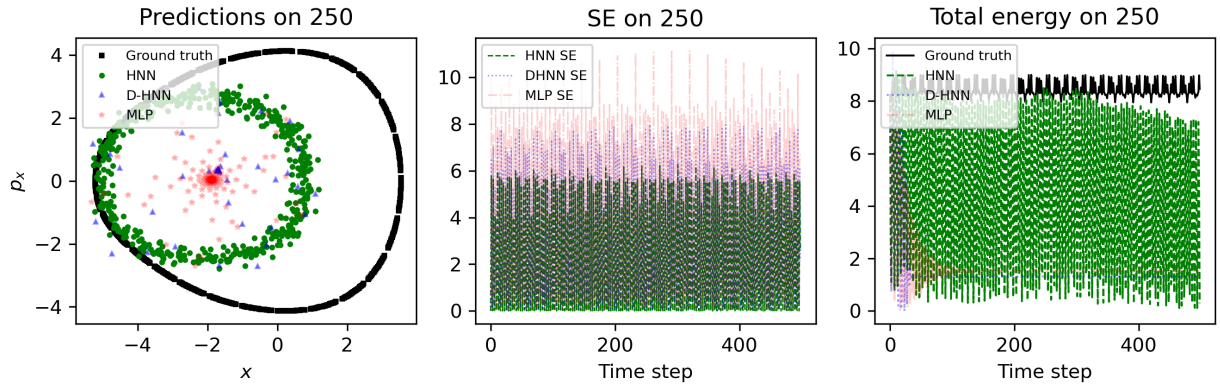

Figure 5.19: Experiment 250, Trial 250

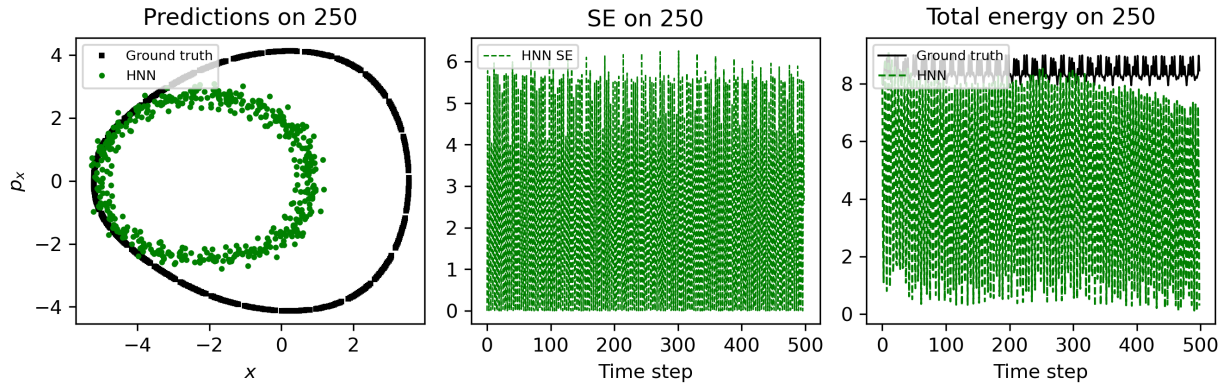

Figure 5.20: Experiment 250, Trial 250: HNN only

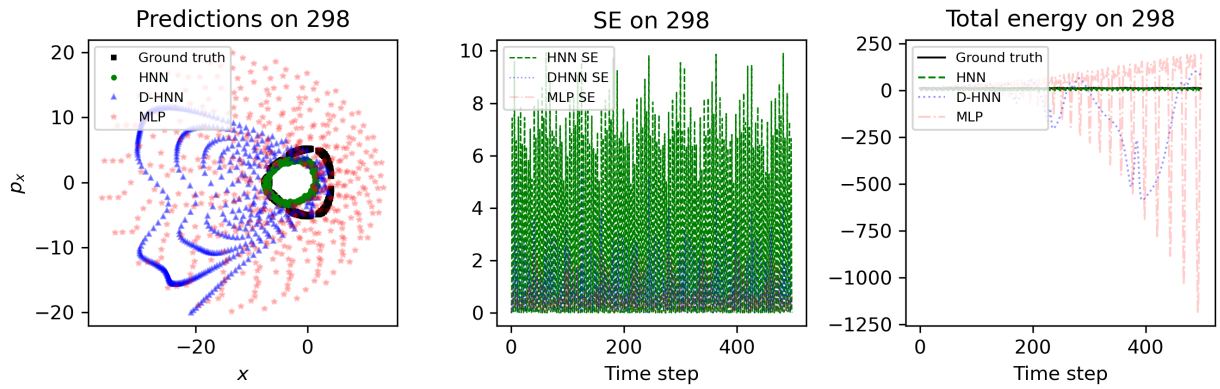

Figure 5.21: Experiment 298, Trial 298

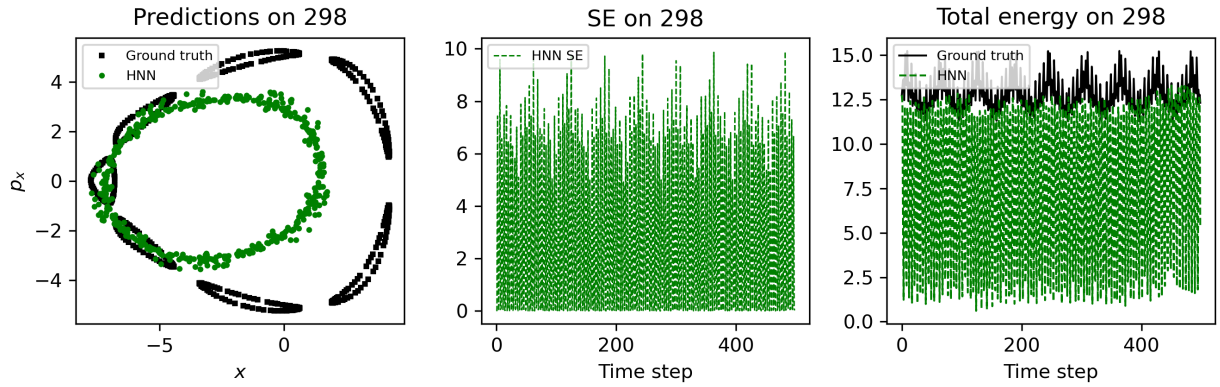

Figure 5.22: Experiment 298, Trial 298: HNN only

Through trial and error, the DHNN and MLP are unable to effectively predict the "islands"; that is, trajectories that produce circular patterns within their own cycle of motion as shown in Figure 5.21. As the trajectories traverse farther from the origin, the DHNN and MLP fail to comprehend the underlying behavior of the training coordinates, despite being trained on the same data set. The precise landing location of the particle beam trajectory holds a subordinate priority since the main focus is whether the predicted trajectories land within the desired region of phase space.

5.4 Mean Squared Error

The mean squared error (MSE) is calculate to evaluate the performance of the HNN, DHNN, and MLP models in relation to the ground truth trajectories. Recognizing that the order of predicted coordinates, (x, p_x), holds minimal significance, we adopt a proximity-based strategy. We determine the closest predicted point in reference to a single ground truth point. This is done for all ground truth points, resulting in an array of ground truth points each paired with the associated closest predicted point. Let the i^{th} trajectory have reserved ground truth data such that $(g_{x_j}, g_{p_{x_j}}) \in \left\{(g_{x_1}, g_{p_{x_1}}), (g_{x_2}, g_{p_{x_2}}), ..., (g_{x_{1000}}, g_{p_{x_{1000}}})\right\}$ where $j = 1, 2, ..., 1000$. We calculate the squared error (SE) of each predicted point with respect to the i^{th} ground truth point and place the SE values into an array. Next, this array is sorted in ascending order, and the SE value at the first index is chosen and appended to a new array. This new array contains SE values associating each $(g_{x_j}, g_{p_{x_j}})$ with its corresponding SE value for the i^{th} trajectory.

The reason for iterating through all predicted values for each individual $(g_{x_j}, g_{p_{x_j}})$, is that the order of predicted points holds little importance in this analysis. Our focus lies in identifying the closest predicted point to a fixed ground truth point, and evaluating the error between them using SE. The MSE helps to determine whether the predicted trajectory has stayed within the desired region of phase space, regardless of the order of predicted points.

The squared error calculations are done in the function *get_true_pred* and the mean squared error is calculated in the **MainExp.py** notebook. The function calculates the squared error between a single ground truth point, $(g_{x_j}, g_{p_{x_j}})$. Let the set of all predicted points output by the HNN be $(p^*_{x_k}, p^*_{p_{x_k}}) \in \left\{(p^*_{x_1}, p^*_{p_{x_1}}), (p^*_{x_2}, p^*_{p_{x_2}}), ..., (p^*_{x_{1000}}, p^*_{p_{x_{1000}}})\right\}$ where $k = 1, 2, ..., 1000$ for the i^{th} trajectory. Let SE_{HNN_j} be the SE for a fixed ground truth point j for the HNN

predictions be given by,

$$SE_{HNN_j} = (g_{x_j} - p^*_{x_k})^2 + (g_{p_{x_j}} - p^*_{p_{x_k}})^2 \tag{5.6}$$

where $k = 1, 2, ..., 1000$. This SE calculation is iterated for each $j = 1, 2, ..., 1000$ until the new SE array contains all of the least SE for each predicted point. Let MSE_{HNN_i} be the mean squared error for the i^th trajectory such that,

$$MSE_{HNN_i} = \frac{1}{1000} \sum_{j=1}^{1000} SE_j \tag{5.7}$$

The MSE's for DHNN and MLP are calculated the same way but with their associated predicted trajectory datasets.

Tables 5.1 to 5.3 display the MSE for each neural network per trial for Experiments 129, 135, and 141. The choice of these trajectories was decided based on the fact that somewhere between trajectories 129 and 145 we see a change in prediction behavior for the DHNN. Note that the experiment number is the trajectory number and refers to the particle trajectory index that other particle trajectories were scaled too. The other trajectory numbers are the trials within a certain experiment. The "{ number }" in the following tables represents the scaled index. For example, in Table 5.1, trajectory 135 is 5% larger than trajectory 129.

MSE from Experiment 129			
Neural Network	129 {1}	135 {1.05}	141 {1.10}
HNN	0.02813184	0.20931989	0.22367826
DHNN	0.02644155	0.00698289	0.00782255
MLP	0.06427188	0.04919715	0.02762560

Table 5.1: Scaled in reference to 129^{th} trajectory

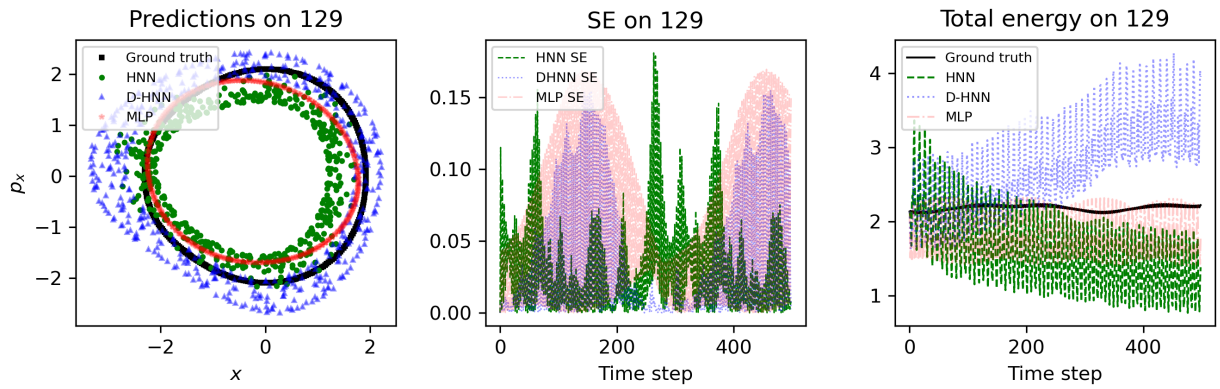

Figure 5.23: Experiment 129, Trial 129

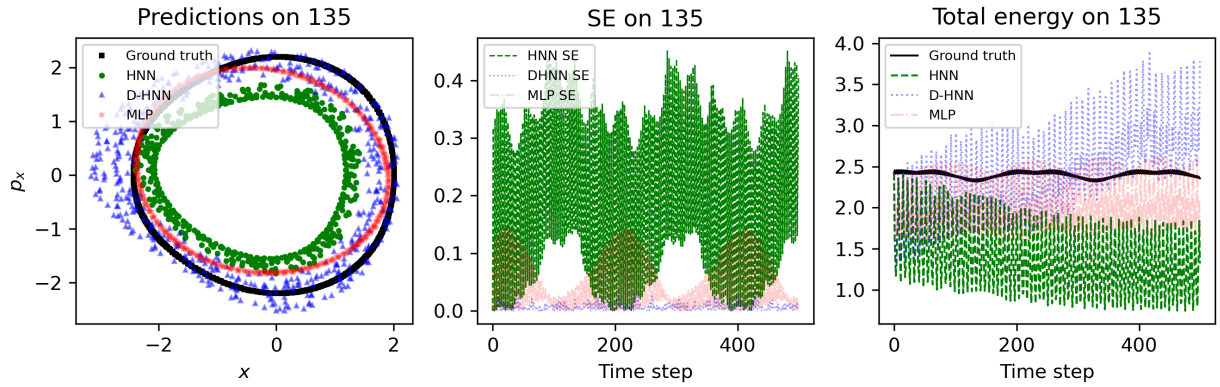

Figure 5.24: Experiment 129, Trial 135

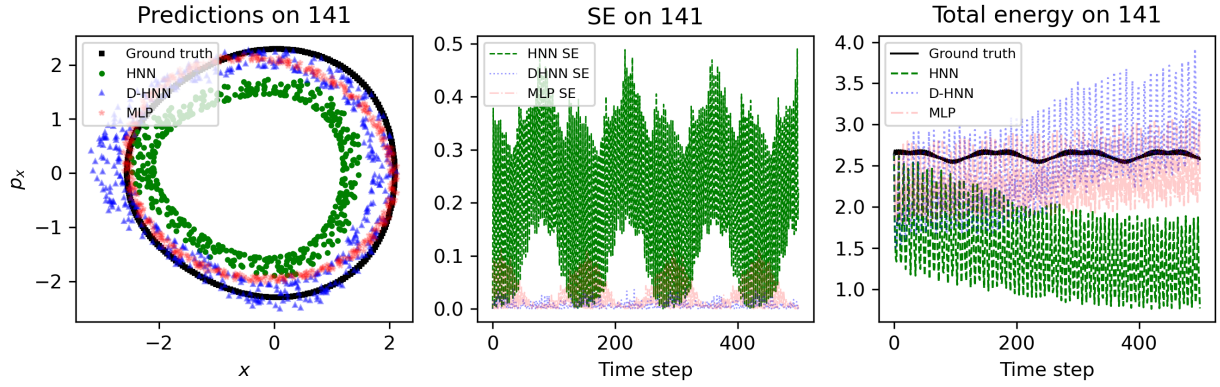

Figure 5.25: Experiment 129, Trial 141

The DHNN produces the smallest MSE for all trajectories in Table 5.1. Visually, all predicted trajectories for experiment 129 are closed and exhibit no chaotic behavior, thus would be considered to be in the desired region of phase space. However, it is worth noting that quantitatively, the MSE's for the HNN are the largest for trajectories 135 and 141. This may be attributed to the fact that the HNN was trained on trajectory 129 and not 135 or 141, assuming that it would preform better if it was trained on it's own trajectory data.

MSE from Experiment 135			
Neural Network	129 {0.95}	135 {1}	141 {1.05}
HNN	0.07857367	0.05073854	0.13720985
DHNN	0.05153979	0.13761024	0.07504052
MLP	0.16765303	0.16765303	0.16638069

Table 5.2: Scaled in reference to 135^{th} trajectory

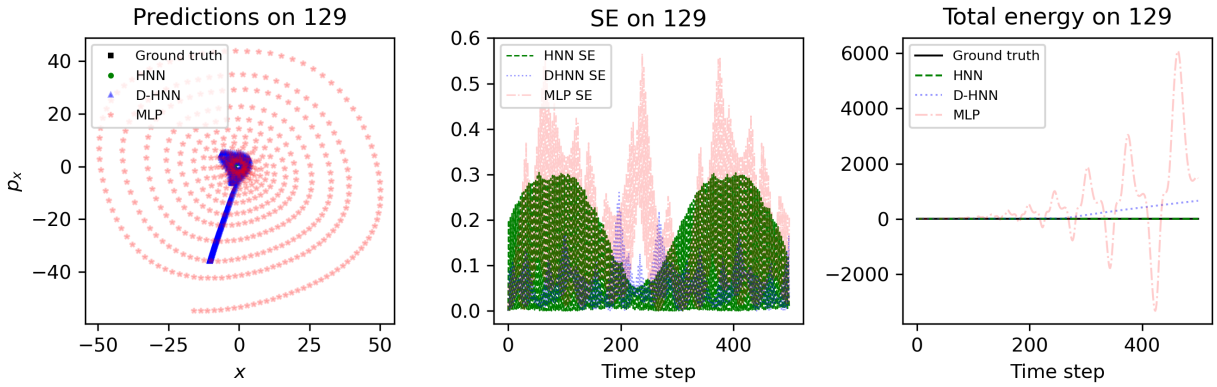

Figure 5.26: Experiment 135, Trial 129

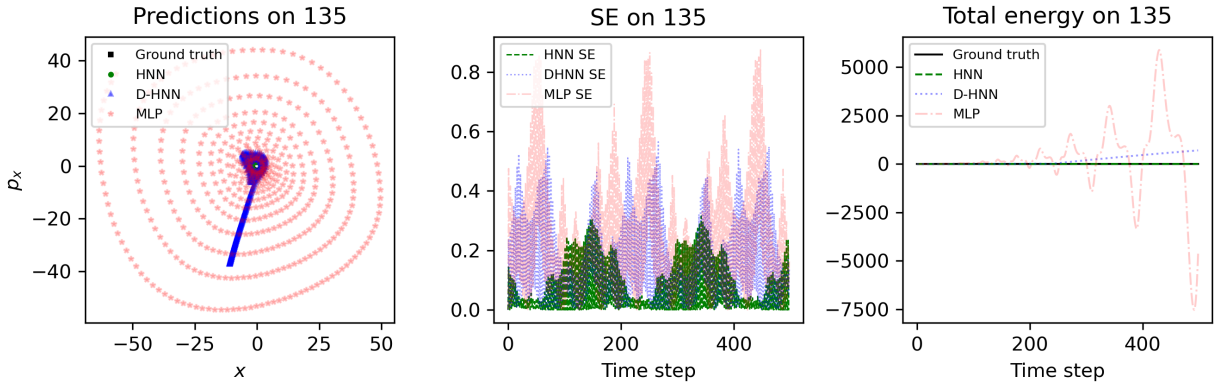

Figure 5.27: Experiment 135, Trial 135

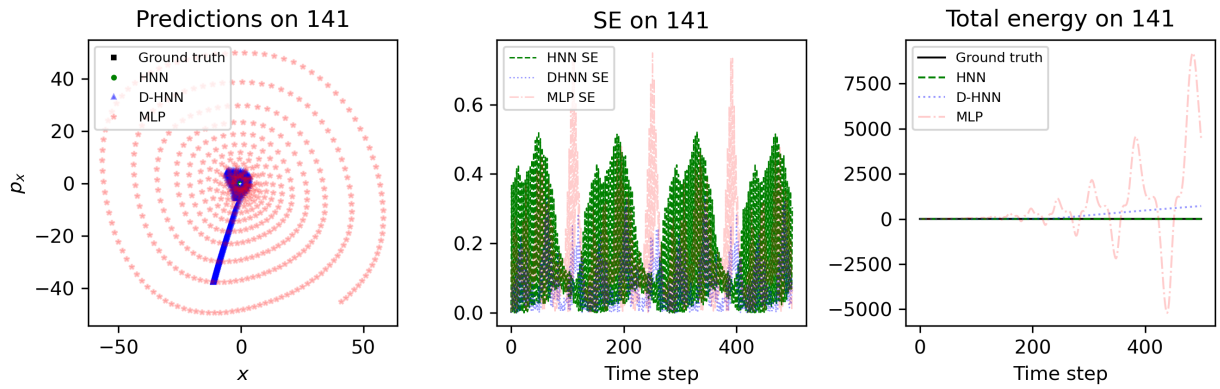

Figure 5.28: Experiment 135, Trial 141

HNN outperforms both DHNN and MLP in terms of visual accuracy, Figures 5.26 to 5.28, when predicting particle trajectories that are within the desired region of phase space. For the HNN, the MSE for trajectory 135 is the lowest only for experiment 135. Again, suggesting that training and testing from the same trajectory dataset may yield optimal results for the HNN. On the other hand, the DHNN produces predictions that appear chaotic and may cause the particles to exit the desired region of stability. However, the DHNN produced the lowest MSE's for trajectory 129 and 141. This may suggest that an MSE calculation is may not be the best quantitative measure to represent how well the neural networks preform. The MLP exhibits a large oscillation, in Figures 5.26 to 5.28, in the total energy which indicates that energy may not be conserved. The DHNN also displays an increasing slop in all of its total energy plots for all trials in experiment 141, also implying that energy is not conserved.

Since the average amplitudes are rounded to only two decimal places in *get_scale* and not in *get_AMP*, more than one trajectory may be the same value after being scaled. Thus, in Experiment 141, trajectories 129 $\equiv$ 127 and 135 $\equiv$ 134. These trajectory numbers are just indexes used to keep track of which data set was used. A comparison of ground truth trajectories can be seen in Figure 5.32.

MSE from Experiment 141			
Neural Network	127 {0.9}	134 {0.95}	141 {1}
HNN	1.05090004	0.89932434	1.87989783
DHNN	0.10775759	0.13520611	1.60056335
MLP	0.25546958	0.32467957	1.01974611

Table 5.3: Scaled in reference to 141^{st} trajectory

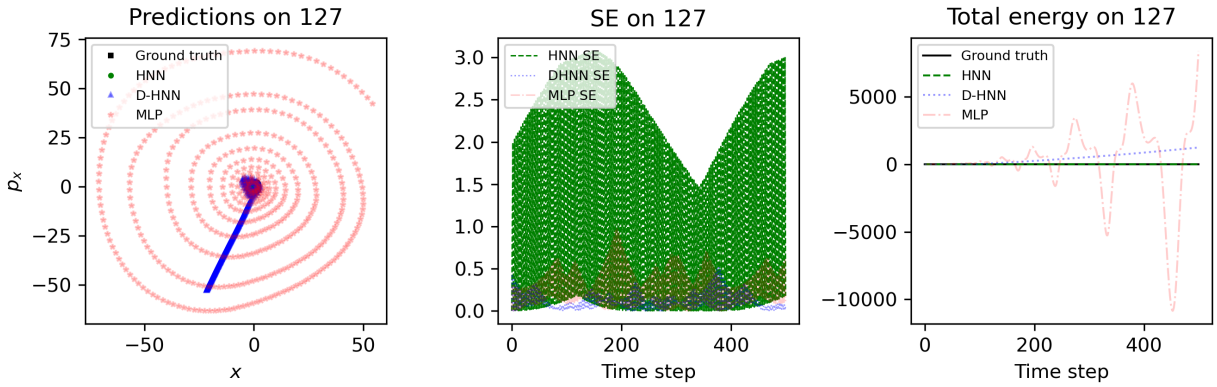

Figure 5.29: Experiment 141, Trial 127

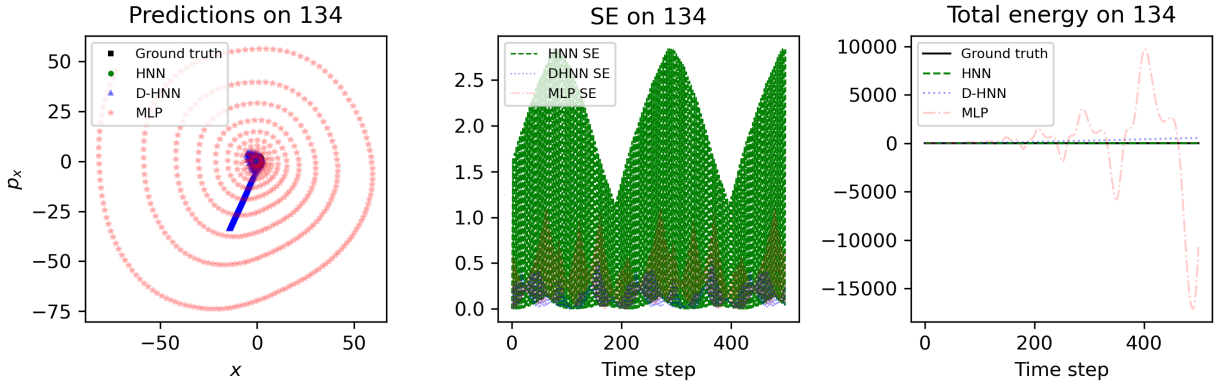

Figure 5.30: Experiment 141, Trial 134

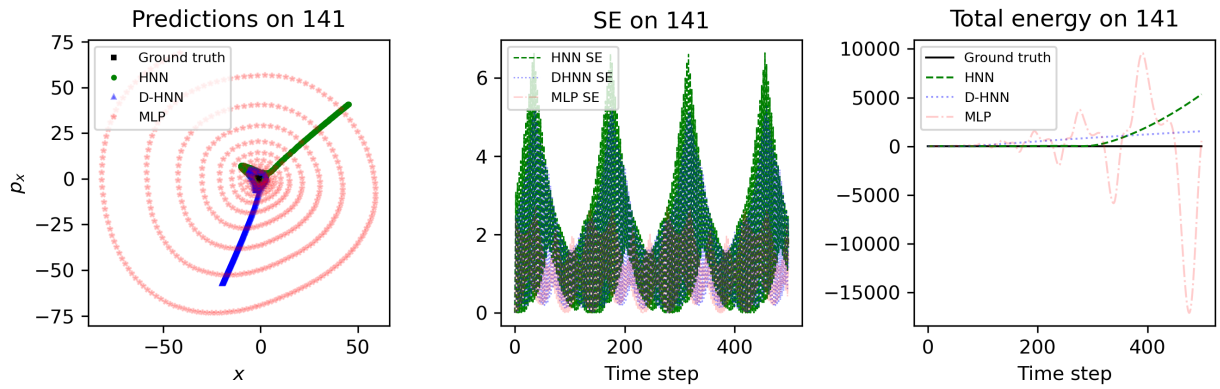

Figure 5.31: Experiment 141, Trial 141

The HNN produced the largest MSE's for all trajectories in experiment 141. The DHNN and MLP exhibit chaos which implies that the predicted particle trajectories for all trials in experiment 141 are outside of the desired region of phase space. In the case of experiment 141 trial 141, Figure 5.31, the HNN also exhibits chaotic behavior and does not conserve energy. The MLP exhibits large oscillations in the total energy which implies that energy may not be conserved over time.

For reference, the full original trajectory data is plotted below,

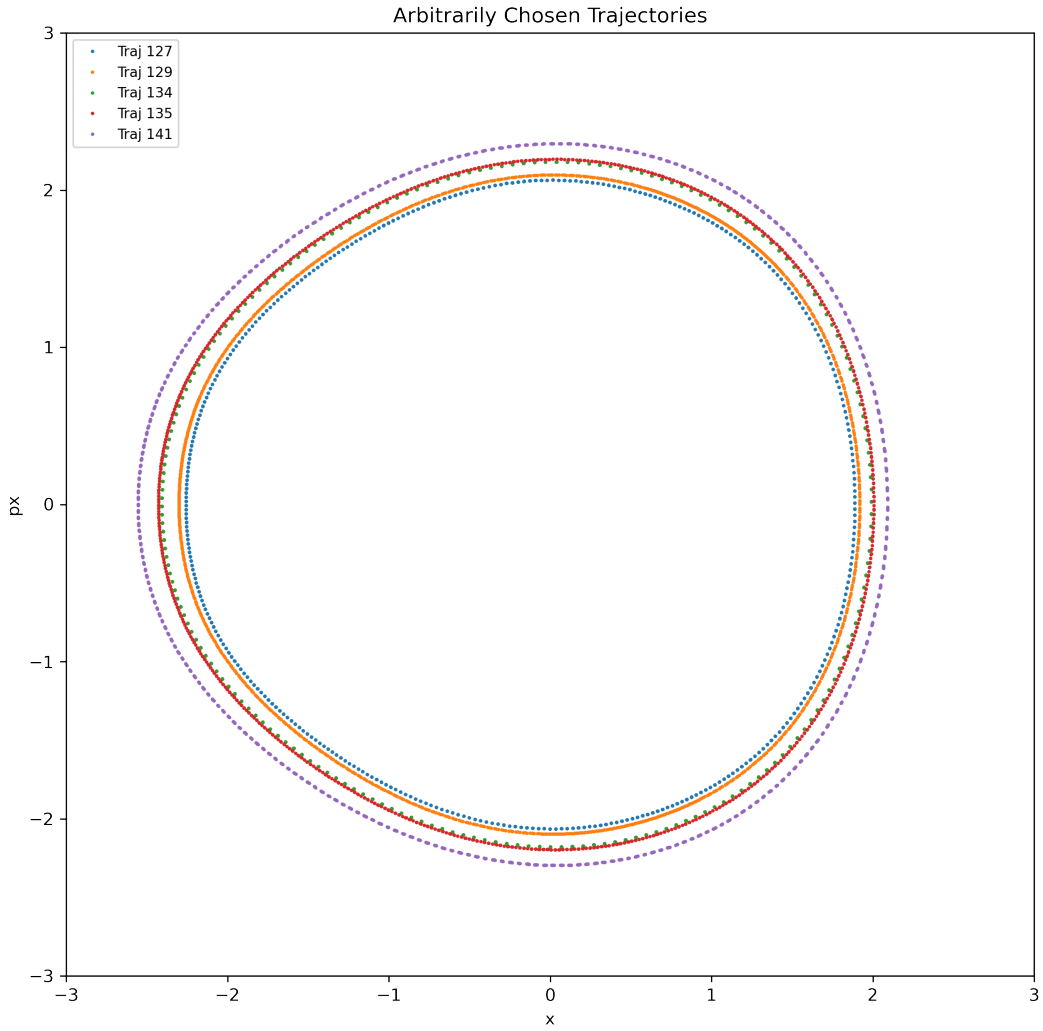

Figure 5.32: True trajectory data from trajectories 127, 129, 134, 135, 141

All of the trajectories in Figure 5.32 are closed without exhibiting chaotic behavior, which indicates that they are considered to be within the desired region of stability.

5.5 Total Energy

The toy Hamiltonian, as described in Equation 3.31, is used to calculate the total energy for both ground truth and predicted trajectories. By passing all trajectory points through the toy Hamiltonian function, a single "energy-like" value is generated and subsequently plotted for analysis as in Figures 5.3 to 5.22 .

In the case of ground truth trajectories, denoted in black, it is observed that the total energy exhibits oscillations, attributable to the constraint of solely considering two dimensions. This behavior is specifically shown in Figures 5.3 to 5.22 where the HNN and ground truth trajectories are exclusively plotted for analysis. These figures explicitly reveal the behavior of the total energy over time, as calculated by the toy Hamiltonian, and showcase the oscillatory nature of the ground truth trajectories. However, it is imperative to note that the overall energy conservation remains intact, as energy may be transferred to other dimensions not examined in this study. The primary focus is to ascertain if the total energy is conserved, which implies that the moving average should exhibit minimal slope or ideally a slope of 0, implying no net energy change over time.

5.5.1 Moving Average

In our analysis of energy conservation with the trajectory data in the toy Hamiltonian system, we utilize the moving average to obtain a clearer understanding of the total energy behavior. The moving average is a statistical method that involves calculating the arithmetic mean of a subset of data points over a period of time or a certain number of points in the dataset [20]. This approach helps to reduce small fluctuations and noise in the total energy data, providing a smoother representation of the total energy dynamics during revolutions of the particle trajectory in a particle accelerator.

By employing the moving average, we aim to gain insights into the ground truth total energy trends and compare them with the total energies of the predicted trajectories from the HNN, MLP, and DHNN. This methodology allows us to effectively analyze the energy conservation properties of the system and make meaningful comparisons among different neural network models.

Let MA_w be the moving average such that for any trajectory dataset of any model be $p_1, p_2, ...p_m$ of size m. Let $p_{n-w+1} + p_{n-w+2} + ... + p_n$ be the chosen subset of the full dataset with window size of length w,

$$MA_w = \frac{p_{n-w+1} + p_{n-w+2} + ... + p_n}{w} = \frac{1}{w} \sum_{i=n-w+1}^{n} p_i \tag{5.8}$$

We apply the moving average, with window size 100, to the total energy plots for the arbitrarily chosen trajectory set, $\{20, 50, 75, 100, 125, 130, 150, 200, 250, 298\}$, as in Figure 5.2 for Figures 5.33 to 5.42 in **MovingAvgTotalEnergy.ipynb**. Although some figures may appear to being conserving energy better than others, it is important to note that each figure has a different y-axis scale.

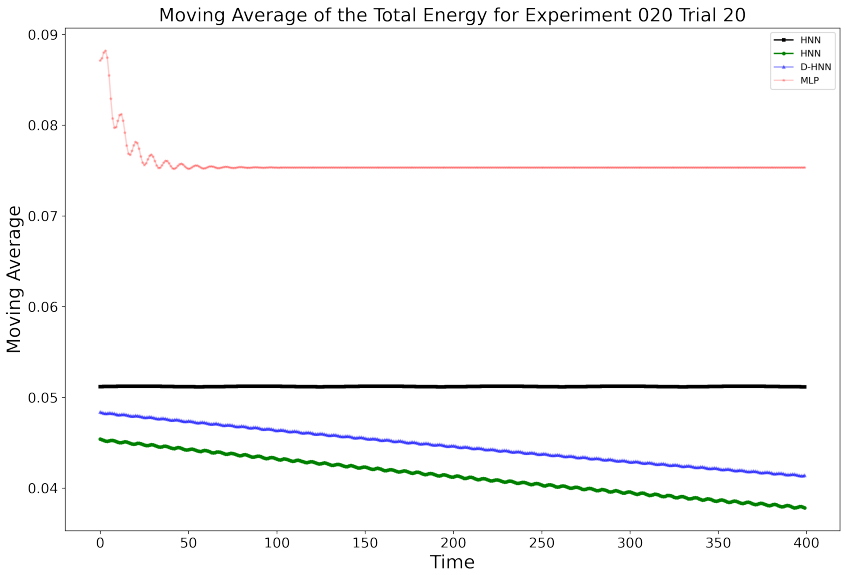

Figure 5.33: Experiment 20, Trial 20

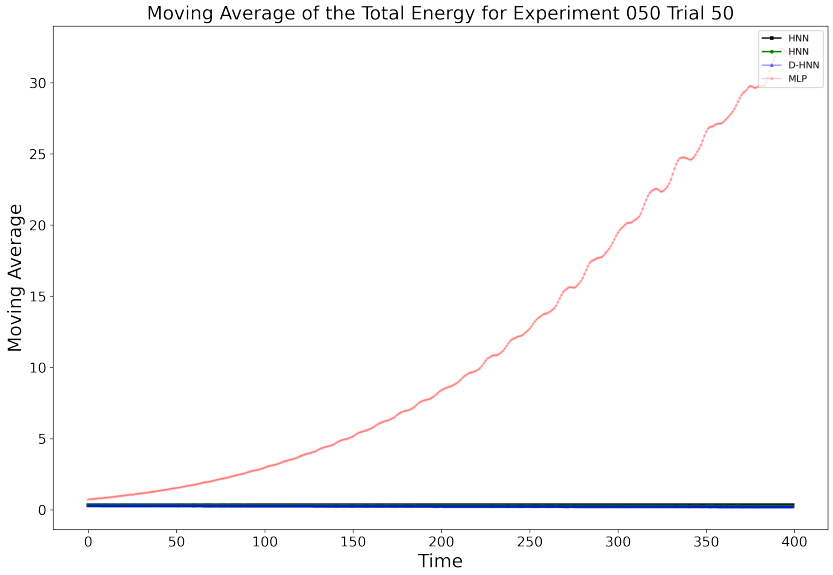

Figure 5.34: Experiment 50, Trial 50

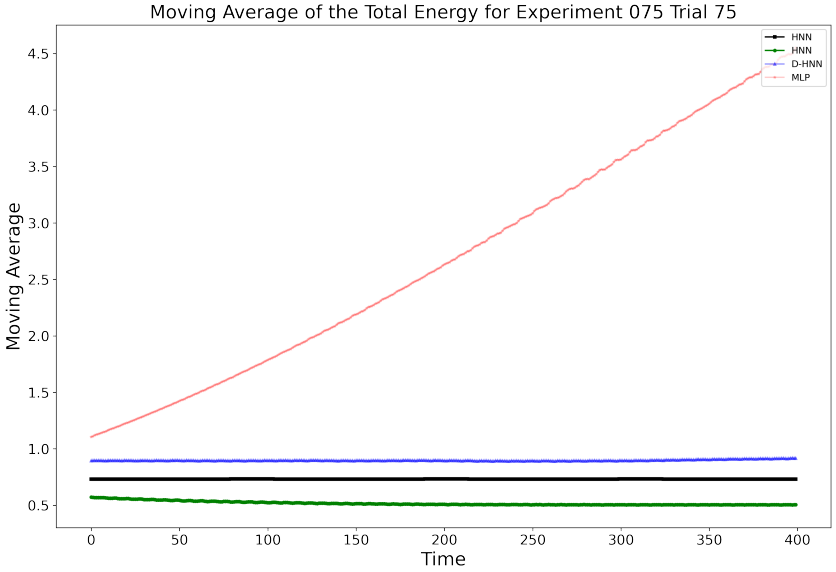

Figure 5.35: Experiment 75, Trial 75

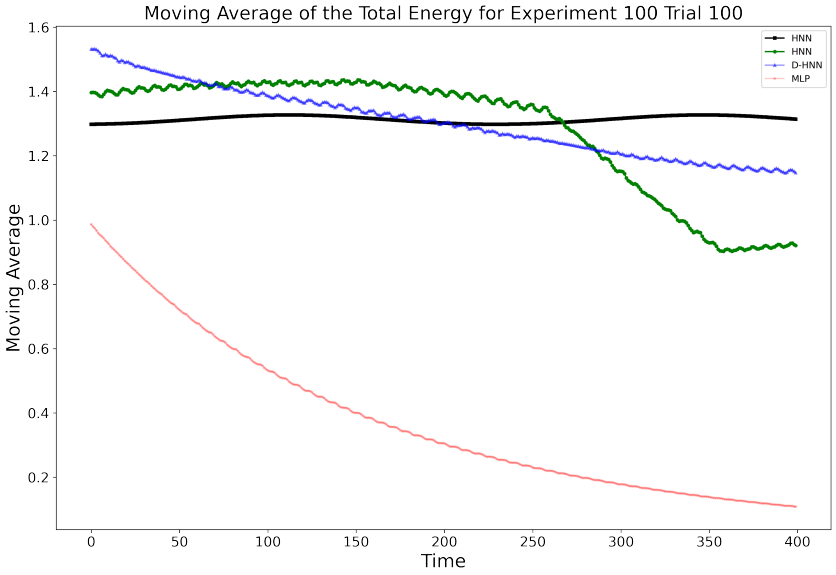

Figure 5.36: Experiment 100, Trial 100

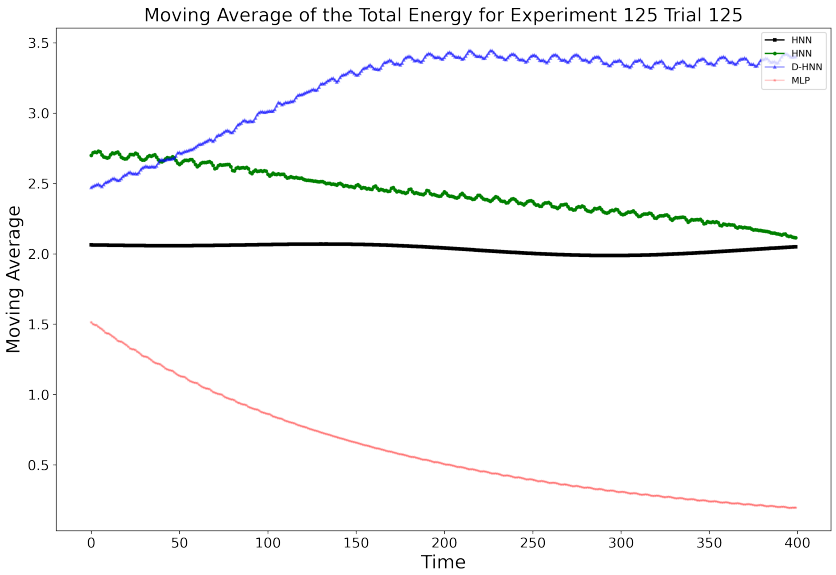

Figure 5.37: Experiment 125, Trial 125

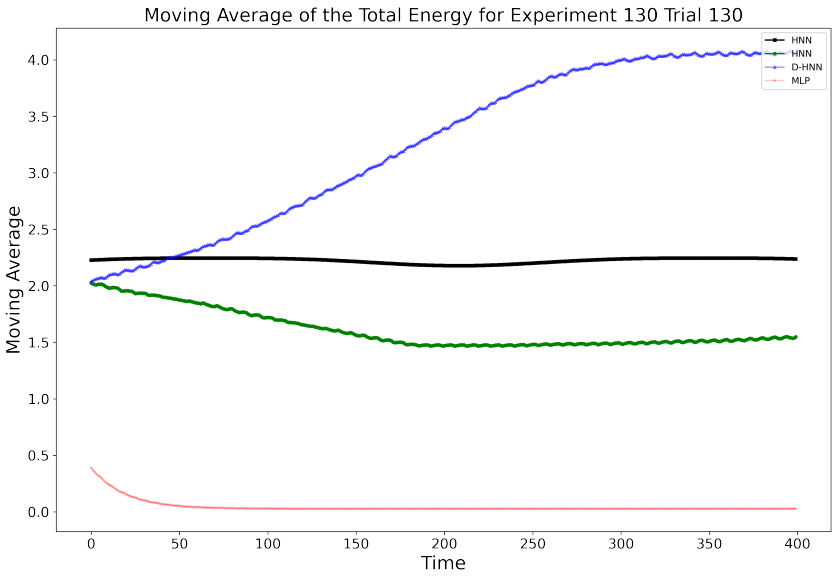

Figure 5.38: Experiment 130, Trial 130

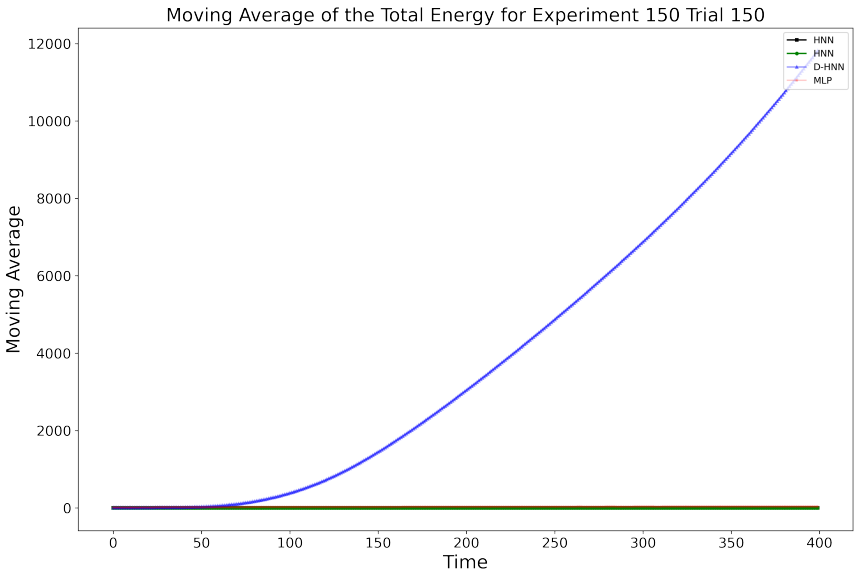

Figure 5.39: Experiment 150, Trial 150

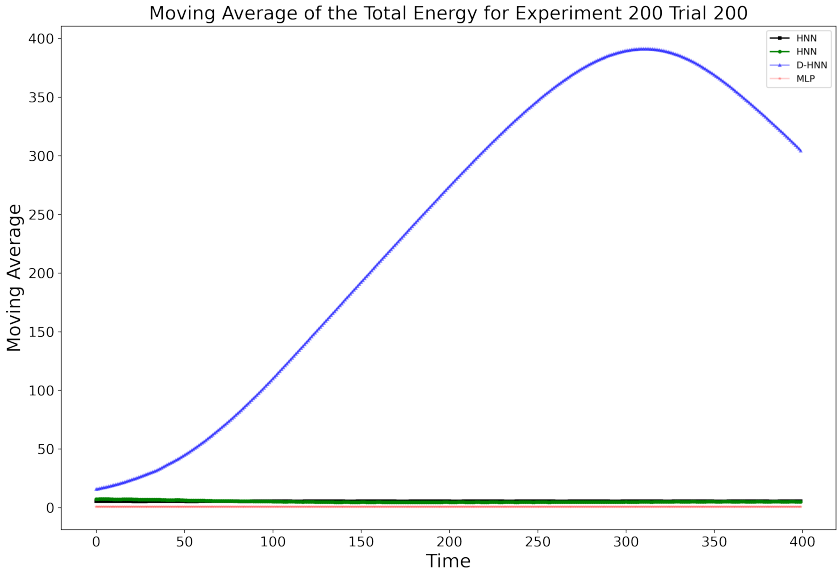

Figure 5.40: Experiment 200, Trial 200

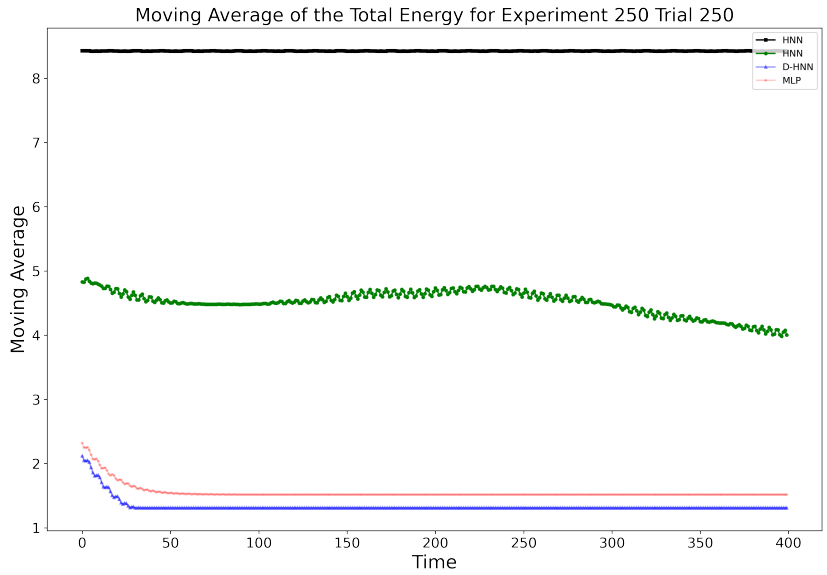

Figure 5.41: Experiment 250, Trial 250

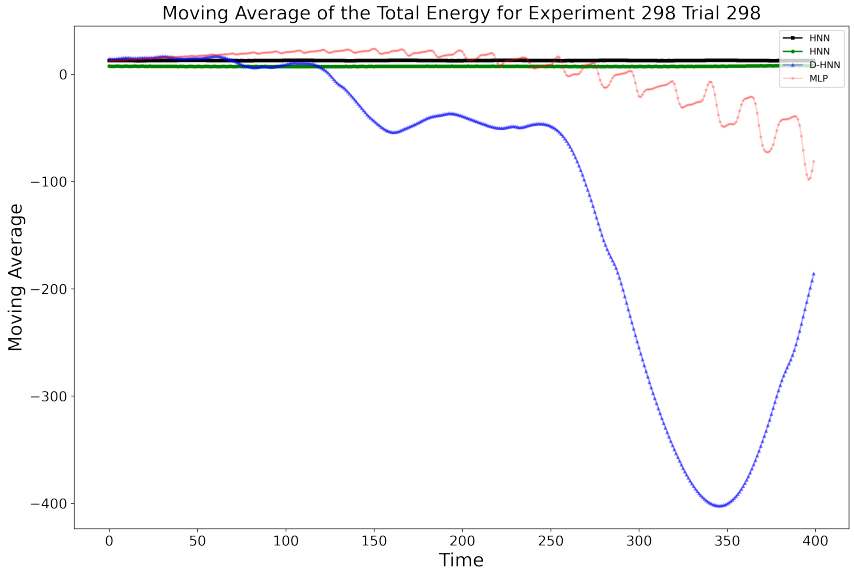

Figure 5.42: Experiment 298, Trial 298

The DHNN shows some exponential behavior indicating a lack of energy conservation. On the contrary, the HNN displays greater performance with the best moving average among

most figures, suggesting that it is more effective in conserving energy than both the DHNN and MLP.

5.5.2 Efficiency

As reported by [5], traditional numerical tracking methods may require approximately 17 hours to track 500 turns of 61 particle trajectories, even while employing computational parallelization. In contrast, with the utilization of neural networks, the time required to train, test, and analyze each experiment was significantly reduced, with an average time of 24 minutes per HNN, MLP, and DHNN.

In our study, each trajectory (ranging from 20 to 300) was considered as an experiment, involving calculations of average amplitude, scaled amplitude, mean squared error (MSE), and total energies, along with training and testing of each neural network. Let H_{total} be the number of hours for 280 experiments be given by,

$$H_{total} = \frac{24min * 280}{60min} = 112hr \tag{5.9}$$

In total, 280 experiments were conducted, with each experiment consisting of up to 5 trials with a the total run time of approximately 112 hours. In contrast, traditional methods of numerical tracking would take up to 4,704 hours which is 196 days for this set of data, 280 particle trajectories.

By leveraging the parallel computing capabilities of NERSC's supercomputers, we were able to run multiple jobs concurrently, resulting in a significant reduction in run time. Specifically, with 12 separate jobs running simultaneously, each processing a equal range of experiments, the overall run time was decreased to 9.5 hours. This highlights the efficiency and scalability of utilizing neural networks in our research, allowing for faster and more streamlined

experimentation and analysis of particle trajectories.

Chapter 6

Conclusion

Particle accelerators that are being designed or frequently undergo upgrades, such as the ALS, require stability studies to ensure optimal performance. These stability studies serve the purpose of providing information about a particle beam's survival within the particle accelerator and are preformed by numerical particle tracking. Traditional numerical particle tracking methods and optimization can be computationally expensive due to their iterative nature. As such, this research aimed to explore alternative approaches, leveraging machine learning and Hamiltonian mechanics, to make particle tracking more efficient.

The longevity of particle beams within the accelerator is crucial for achieving optimal luminosity and x-ray brightness. By utilizing Hamiltonian mechanics, which provides a way to model particles in a physical system based on the total energy (Hamiltonian) of the system, we were able to investigate if energy was being conserved in the system.

Through the use of machine learning techniques, we explored the possibility of more efficient particle tracking. By combining the power of machine learning with the principles of Hamiltonian mechanics, we aimed to determine the feasibility of utilizing these approaches for particle tracking in the context of stability studies for particle accelerators.

6.1 Future Endeavors

Although the use of Hamiltonian Neural Networks (HNN) for particle tracking is a relatively new research topic, there is still limited understanding on how to effectively track particles in multidimensional phase space using HNN. In this study, we focused on tracking particles in only 2 dimensions using simulated data.

A potential avenue for future research could involve utilizing real particle trajectory data to further validate and refine the HNN-based tracking approach. Additionally, there are numerous other parameters and factors that can influence particle tracking in a particle accelerator, such as beam optics and external fields, which could be incorporated into the training of the HNN to improve its performance and narrow the scope of the learning algorithm.

As this field continues to evolve, further investigations and developments are needed to fully explore the capabilities of HNN in particle tracking. The incorporation of real data and consideration of additional parameters could provide valuable insights and enhance the accuracy and applicability of this approach.

6.2 Final Thoughts

The utilization of Hamiltonian Neural Networks (HNN) as a method for predicting particle trajectories in phase space holds significant potential for efficiency gains compared to traditional numerical methods. The HNN approach has demonstrated some ability to track particles within the desired region of phase space, with predicted trajectories exhibiting closed non-chaotic behavior, in contrast to the performance of multi-layer perceptron and Dissipative Hamiltonian Neural Network.

The proof of concept provided by HNN in this research is promising, suggesting that it could

be a viable method for particle tracking. However, further research is needed to validate and solidify this idea, through rigorous testing with real data and comprehensive analysis of the addition of more parameters. The potential time-saving benefits of neural networks in predicting particle trajectories make it a compelling area of study for the future development of efficient particle tracking methods, and further investigations could potentially establish HNN as a reliable and efficient approach in this domain.

Bibliography

[1] How particle accelerators work.

[2] Particle accelerators and radiation research.

[3] Cern accelerating science, 2006.

[4] T. Faraj Al-Janabi and S. A, architecture of the simple neural network classifier b, architecture ... https://www.researchgate.net/figure/A-Architecture-of-the-simple-neural-network-classifier-B-Architecture-of-the-deep$_fig3_318652568$. *Accessed : April*20, 2023.

[5] Michael Borland. Beam optics challenges in the next generation of light sources, 2006.

[6] David B. Cline. Hamiltonian invariance, 2016. Accessed: March 9, 2023.

[7] Lipi Gupta. "Analytic and Machine Learning Methods for Controlling Nonlinearities in Particle Accelerators". PhD diss. University of Chicago 2021.

[8] Nikita Kuklev. Personal correspondence.

[9] Lawrence Berkeley National Laboratory. Staff at berkeley lab's x-ray facility mobilize to support covid-19-related research, 2020.

[10] Lawrence Berkeley National Laboratory. Advanced light source upgrade (als-u) overview, n.d.

[11] Lawrence Berkeley National Laboratory. Early years at the lawrence berkeley national laboratory, n.d.

[12] Mikaela Meitz. Hnn-research. https://github.com/MikaelaMeitz/HNN-Research, 2023. GitHub repository.

[13] David Morin. Chapter 15 - electric charge, coulomb's law, and electric field, 2023.

[14] Eric Roberts. A brief history of neural networks, n.d.

[15] G. Rumolo, A. Boccardi, R. Bruce, F. Cerutti, M. Kalliokoski, S. Redaelli, A. Rossi, B. Salvachua, M. Scapin, and G. Valentino. Beam-induced heating in the lhc - measurements and simulations. *Journal of Instrumentation*, 10(05):P05007, 2015.

[16] S. Greydanus, A. Sosanya. "Dissipative Hamitonian Neural Networks: Learning Dissipative and Conservative Dynamics Separately". arXiv preprint arXiv:2201.10085, 2022.

[17] S. Greydanus, M. Dzamba, J. Yosinski. "Hamiltonian Neural Networks". arXiv preprint arXiv:1906.01563, 2019.

[18] The American Physical Society. The early history of the american physical society, 2001.

[19] Wikipedia. Hamiltonian mechanics. https://en.wikipedia.org/wiki/Hamiltonian_mechanics. Accessed on April 11, 2023.

[20] Wikipedia. Moving average, 2023.