Using Neural Networks to Solve Differential Equations in Classical Mechanics

Julie Butler
June 22, 2022
DSECOP June Workshop

MICHIGAN STATE UNIVERSITY

Theory Alliance
FACILITY FOR RARE ISOTOPE BEAMS

Outline of Presentation

- Introduction to the Module Topic
- Overview of the Module Contents
- How can this module be worked into a classical mechanics course?
- Conclusions and Future Works

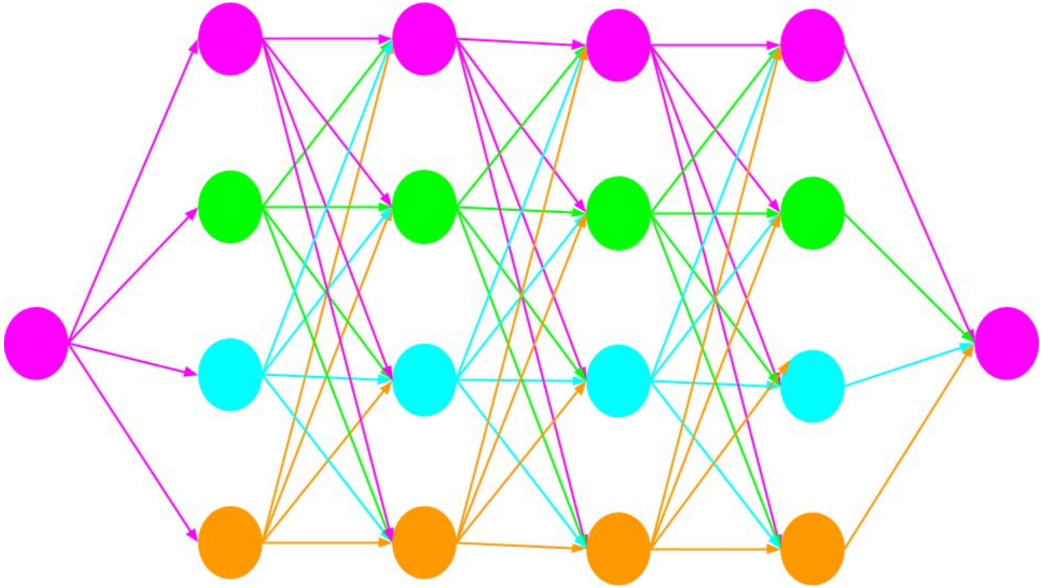
Introduction to the Module Topic

Why Incorporate Machine Learning in Classical Mechanics?

- Programming and machine learning are becoming increasingly integrated in many areas of physics
  - Important for undergraduates studying physics to be exposed to these areas
- Students in Classical Mechanics
  - Typically sophomores or juniors
  - Typically a physics majors only class

What is a Neural Network?

- Computational system that learns to match inputs to outputs by tuning a set of weights



Solving Differential Equations Numerically

- Method to solve second order differential equations by discretizing the inputs
    - Acceleration and position
- Euler's Method, Euler-Cromer Method, Velocity-Verlet Method

$$v_{i+1} = v_i + a_i \Delta t$$

$$y_{i+1} = y_i + v_{i+1} \Delta t$$

Solving Differential Equations with Neural Networks

- Assume we are given the acceleration for an object, want to model its position using a neural network

$$y_{trial}(t) = A + Bt + t^2 NN(W, t)$$
$$\mathrm{y}(0) = \mathrm{A},\ \mathrm{v}(0) = \mathrm{B}$$

- Requires taking derivatives of the neural network--automatic differentiation library called JAX

Overview of the Module Contents

Outline of Module

1. Solving Differential Equations Numerically
2. What is a Neural Network?
3. Solving Differential Equations with Neural Networks
4. Further Problems

Prerequisites:

- Python programming including NumPy and Matplotlib
- Newton's second law and related
- Derivatives, integrals, matrix-vector math

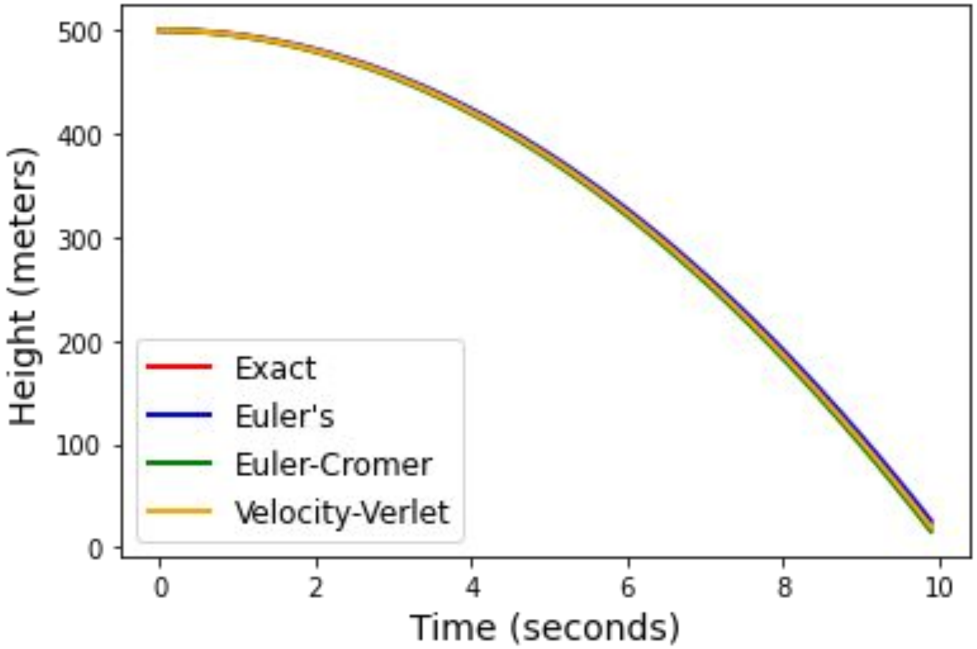
Overview of Each Notebook Contents

- Introduction to the topic containing text and equations
- Python code broken into small pieces with text explanation and comments
- Short exercises scattered throughout the notebook
- Longer “Practice What You Have Learned” at the end

```python
def neural_network(W, x):
    """
        Inputs:
            W (a list of length 2): the weights of the neural
                network
            x (a float): the input value of the neural network
        Returns:
            Unnamed (a float): The output of the neural network
        Defines a neural network with one hidden layer.  The
        number of neurons in the hidden layer is the length of
        W[0]. The activation function is the sigmoid function
        on the hidden layer an none on the output layer.
    """
    # Calculate the output for the neurons in the hidden layer
    hidden_neuron = sigmoid(jnp.dot(x, W[0]))
    # Calculate the result for the output neuron
    return jnp.dot(hidden_neuron, W[1])
```

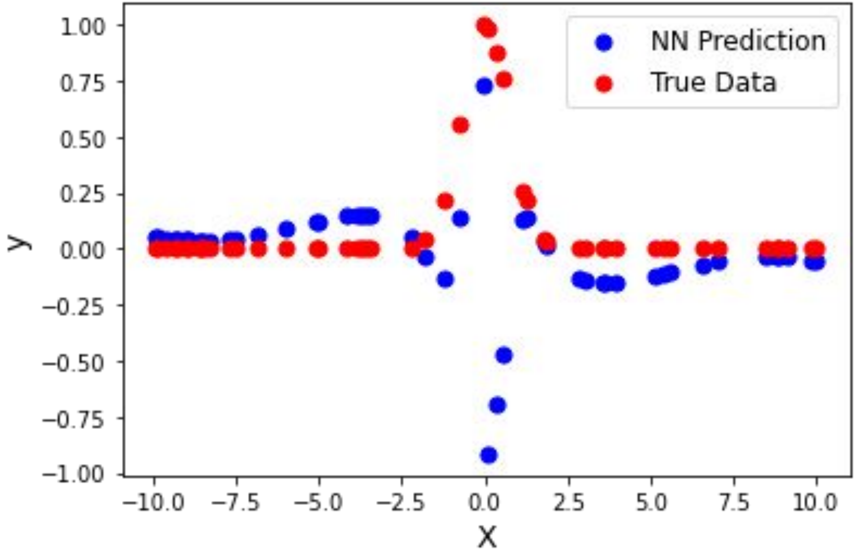
Notebook 1: Solving Differential Equations Numerically

- Introduction to differential equations
- Euler's Method
- Euler-Cromer Method
- Velocity-Verlet Method
- Walkthrough Example
  - Freefall in the presence of linear drag
- Practice What You Have Learned
  - Simple harmonic oscillator



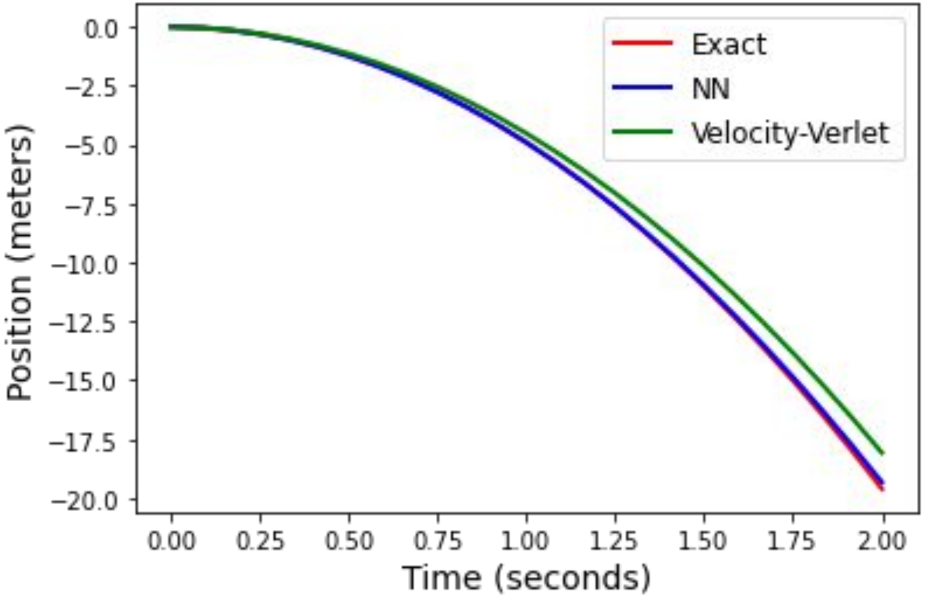
Notebook 2: What is a Neural Network?

- What is a neural network?
- Creating a neural network from scratch using JAX
- Creating neural networks with popular Python libraries
- Practice What You Have Learned
    - Optimizing the from scratch neural network



Notebook 3: Solving Differential Equations with Neural Networks

- How to solve differential equations with neural networks
- Setting up a neural network to solve for position given acceleration
  - Freefall with linear drag
- Improving the results
  - Hyperparameter Tuning and Step Size
- Practice What You Have Learned
  - Sliding with friction



How can this module be worked into a classical mechanics course?

When?

- Most advanced physics topic covered is the exact solution for freefall with linear drag
  - Sections 2.1-2.2 in *Classical Mechanics* by John Taylor
- Could be added anywhere in a classical mechanics class
- Post-exam break from learning new material or in place of instruction if professor is absent

How to Incorporate it In Class

Each Notebook Contains:

- Walkthrough of the module containing text and code
- Short exercises problems dispersed through the text
- "Practice What You Learned" problem at the end

Option 1: All Three Notebooks in Class

- Out of class time: ~3hr
- In class time: ~3hr

Option 2: Only Notebook 3 In Class

- Out of class time: ~5hr
- In class time: ~1hr

Option 3: Self-Study

- Out of class time: ~6hr
- In class time: 0hr

Conclusion and Future Works

Conclusions and Future Works

- This module introduces neural networks to students using simple physics models so the machine learning takes center stage
- The knowledge given in this module should allow students to investigate further uses of neural networks on their own

- Add more problems with different types to Notebook 4: Further Problems
  - Exam questions, clicker questions

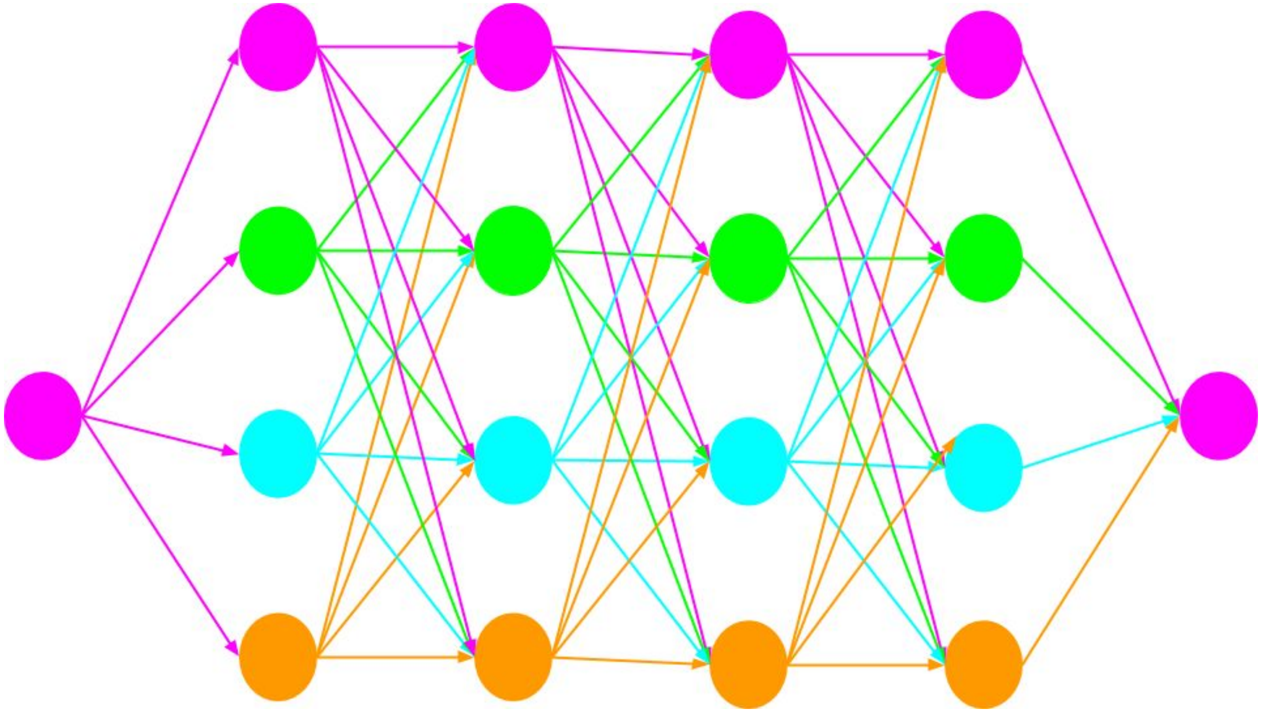
Thank You!

GitHub:

Module Feedback:

Extra Slides

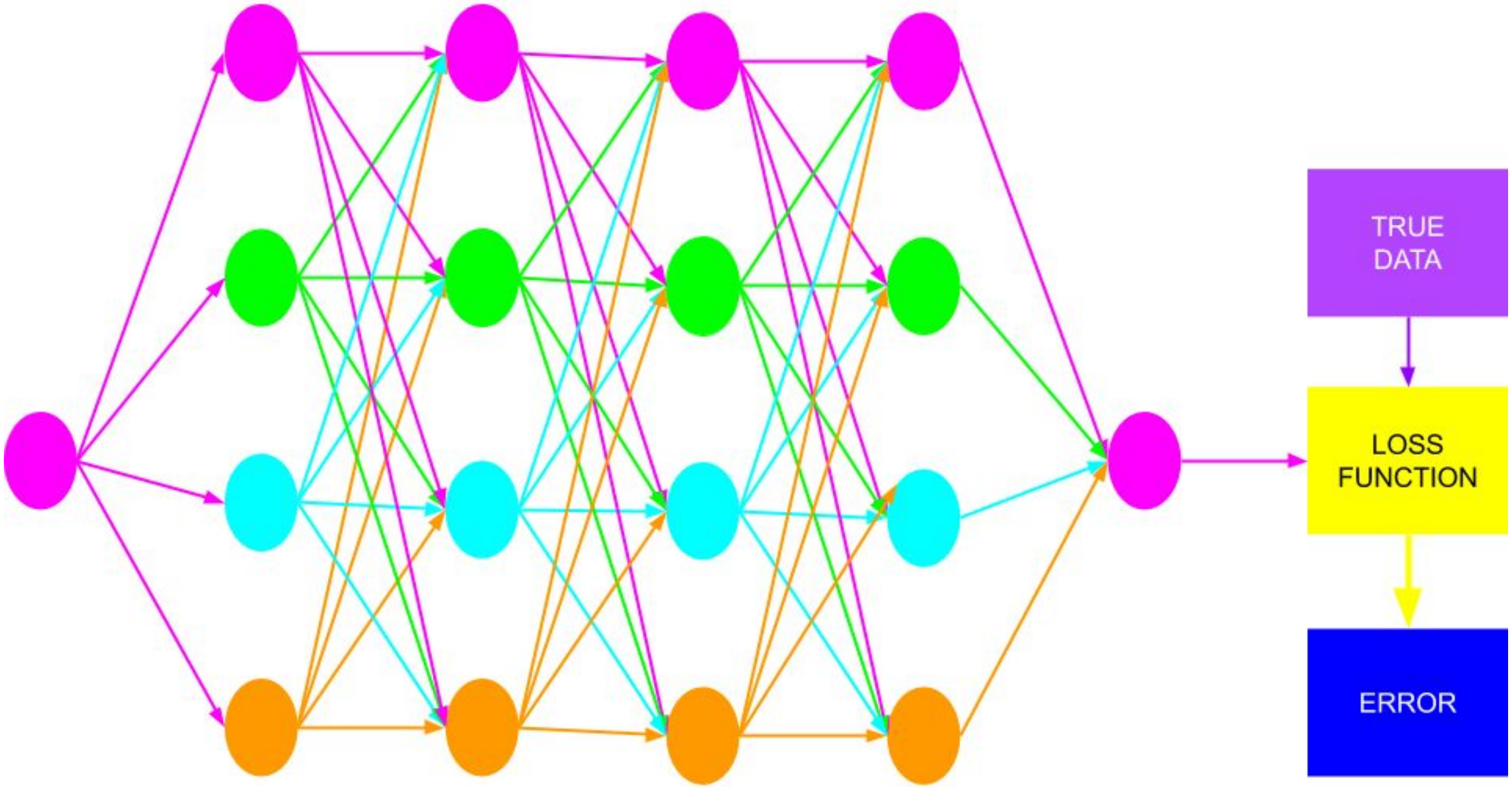
What is a Neural Network?



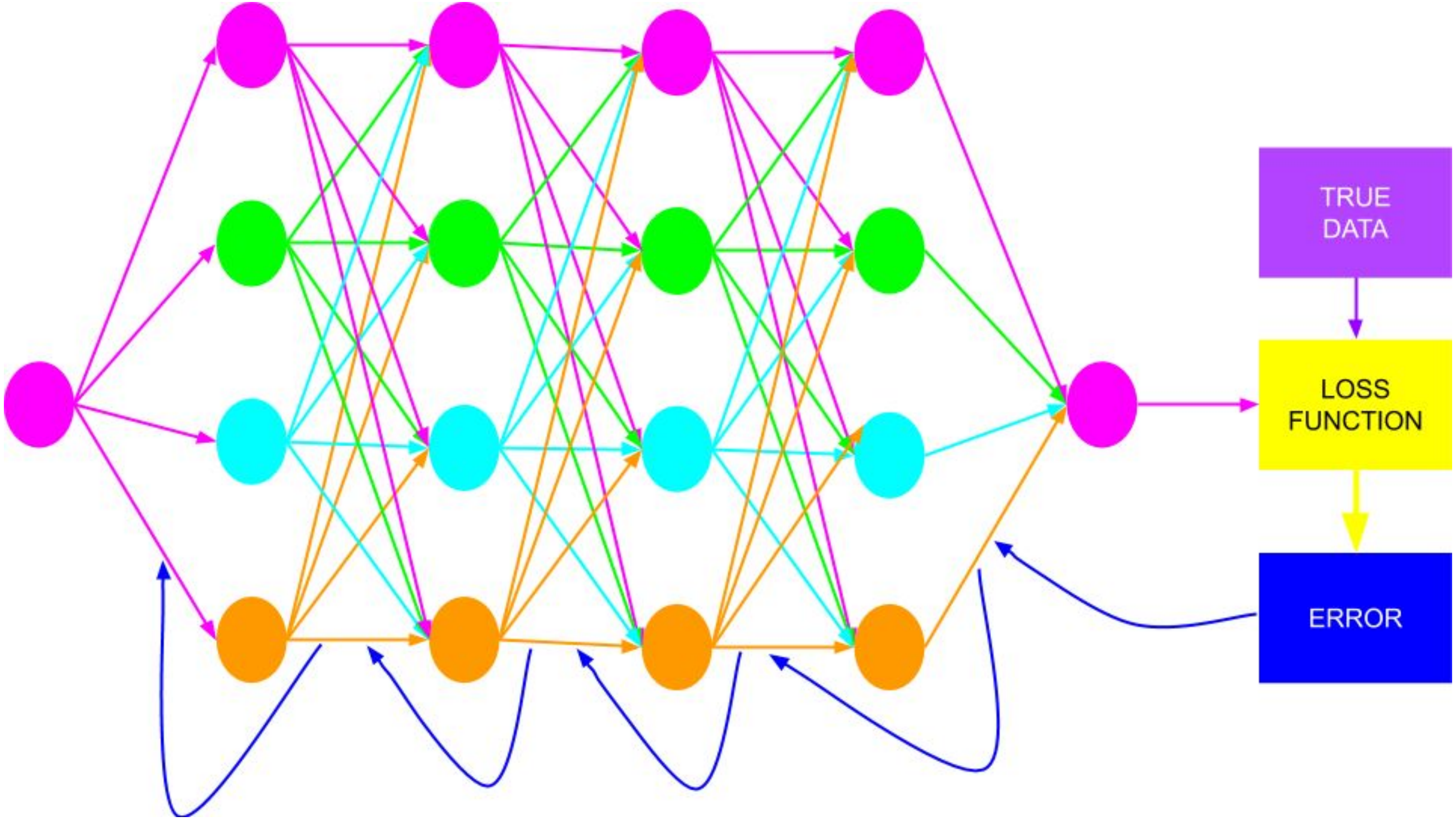
Neural Network: Computational system that is trained to match a given input to a given output

Exact mathematical form for each neuron and layer

Training a Neural Network



Training a Neural Network



Training a Neural Network: Gradient Descent

$$J(W) = \frac{1}{N} \Sigma_{i=i}^{N} (y_i - \hat{y}_i)^2$$

$$\mathrm{W} = \mathrm{W} - \mathrm{r}_l \frac{\partial J(W)}{\partial W}$$

Training a Neural Network: Many iterations of a forward pass followed by backpropagation to update the weights