Soft Robotic Gait Optimization Using Deep Learning

Joel Valley

November 16, 2025

Contents

Chapter 1

Outline

1.1 Background

The development of an untethered soft robot that uses electromagnetic leg actuation to drive locamotive action requires specific input parameters to create a desired gait. The desired optimized variable for this project is the robot's forward velocity. The goal is to not only predict the maximum velocity that the robot can move, but also the input parameters controlling the gait to achieve this desired maximum.

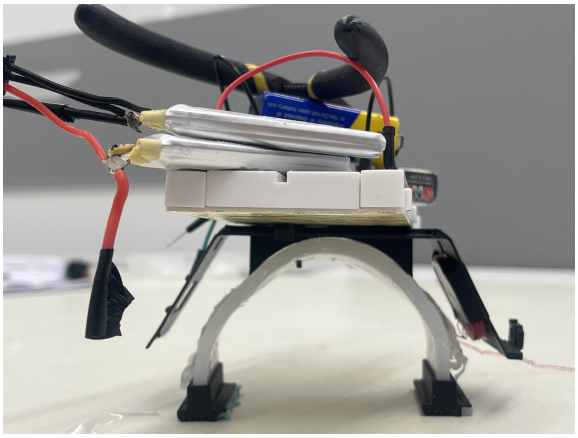

Figure 1.1: Soft Robot Prototype

As shown in Figure 1.1, the soft robot consists of a rigid plastic shell with to electromagnetic solenoids connected on either side. The soft portion of the robot is its silicone legs which are embedded with permenant magents. At the ends of the

silicone legs are plastic feet with silicone embedded on only half of each foot, which allows for directional friction to ensure the robot is propelled forward. By pulsing the electromagnetic solenoids on each leg, the robot builds oscillatory momentum, and this rocking motion is what propels the prototype forward.

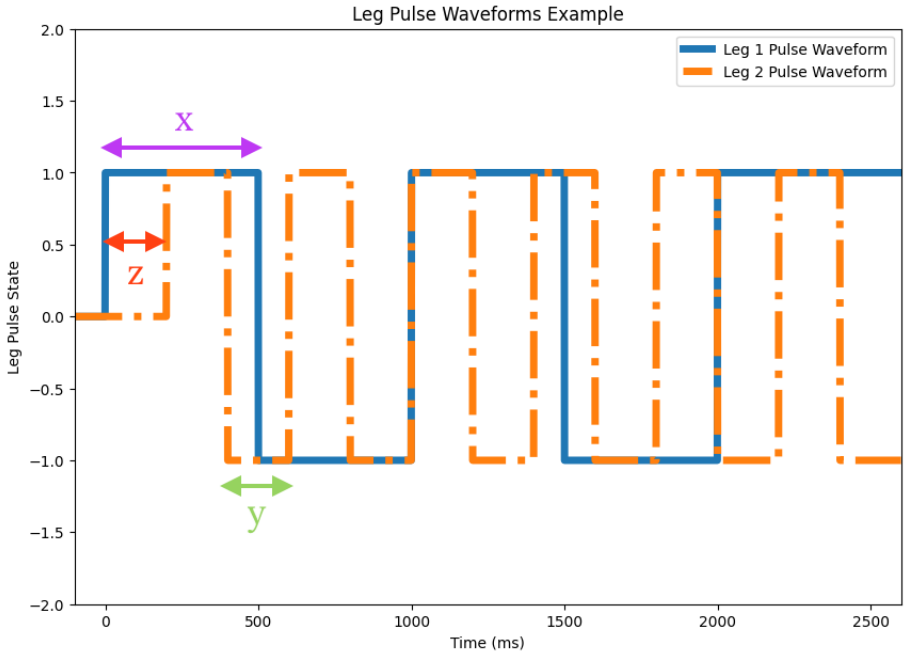

Figure 1.2: Leg Pulse Waveform Example Plot

The current hypothesis is that three main parameters contribute the most to the output velocity: first leg pulse width, second leg pulse width, and the offset between the waveforms. As seen in Figure 1.2, the parameters from here onward denoted as x, y and z will be the two leg pulse durations, and the pulse offset, respectively. Each leg has its own square waveform where the current passes through the solenoid one way, then after the set pulse duration, the motor drivers will be used to change the direction of the current, thus switching the electromagnet's polarity. The offset, z, is the time at which one leg's pulse waveform lags the other. By varying these parameters, the forward velocity of the robot should change, and there should be some value for each of these parameters that will maximize that velocity. It is important to note that Figure 1.2 is only for illustrative purposes.

Chapter 2

Offline Synthetic Data Optimization

2.1 Introduction

To familiarize ourselves with the concepts of what is to be implemented on the prototype, the first phase of the gait optimization is performing the optimization offline using synthetic data. A 3-dimensional function is used to generate a dataset of speed values (w) using random inputs (x, y, z). This dataset is then passed to a multiple regression model that is to learn how to map the inputs to the outputs. This can then be taken a step further by optimizing an initial guess of the input parameters and performing gradient ascent on the model's predictions to find the maximum value of the model's predictions to locate the maximum predicted output. Once the model's predictions are maximized, the remaining optimized input represents the model's estimate of the parameters that produce the maximum output. This estimated maximum can then be compared to the true maximum of the underlying function (if known), allowing the error and accuracy of the model to be evaluated.

2.2 Paraboloid Dataset

2.2.1 Dataset Generation

The first synthetic dataset used to perform multiple regression is a simple concave-down 4D paraboloid with some shifting to have the vertex at a known, non-zero point.

$$w = -((x - 200)^2 + (y - 150)^2 + (z - 25)^2) + 20 + \epsilon \tag{2.1}$$

where:

w = speed of the robot

x = first leg pulse width

y = second leg pulse width

z = leg pulses waveforms offset

ϵ = random noise $[-0.1, 0.1]$

From the mathematical model in (2.1), it is evident that the maximum value of the paraboloid is $w \in [20 - |\epsilon_{max}|, 20 + |\epsilon_{max}|]$, and occurs at the vertex $(200, 150, 25)$.

Five-hundred sample points were generated using a normal distribution with a mean at the vertex of the paraboloid, and a standard deviation of 70 for each input. These values were chosen so that more datapoints near the maximum of the paraboloid would be generated, allowing the model to more accurately predict the maximum. A medium value for the standard deviation was chosen so that the model will still learn a general pattern for the data and not solely values near the peak.

Before passing the dataset to the neural network, it was normalized centered at zero using scikit-learn's StandardScaler. The data was split into a 80:20 train/test split, and the data was batchified for improved model training.

2.2.2 Model Architecture

The architecture of the neural network used to learn this dataset is a feed-forward MLP neural network designed for multiple regression, with four linear layers and three GELU layers. The GELU activation function was chosen through experimentation, and was used instead of ReLU to avoid dying ReLU, since our data is normalized and centered at zero, meaning we can have negative values. The model was trained using three input parameters (x, y, z), 1024 hidden units, and one output (w).

2.2.3 Model Training

The model was trained for 2500 epochs using an ADAM optimizer for faster convergence, and MSE loss was chosen for its sensitivity to outliers. The learning rate of the optimizer is 1×10^{-5}, and was selected through testing trial-and-error. The model architecture and training parameters will be optimized in future sections using a ML pipeline library of sorts.

2.2.4 Results

After 2500 epochs of training, the model has a training loss of 0.0000018, and a test loss of 0.0003811. The loss decreased very fast in training, however, since the data is normalized, it was noticed that even a tiny amount of error in the scaled values results in a large error when the predictions are upscaled back to their original sizes. This is why the model was trained for so many epochs on a relatively simple dataset.

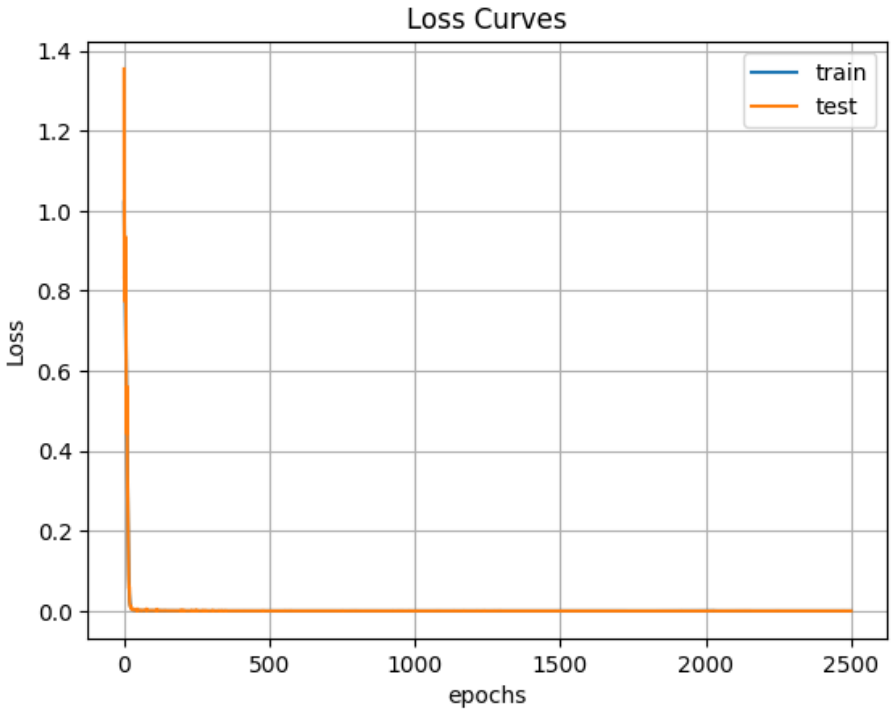

Figure 2.1: Loss Curves

The model was then used to predict some points on the paraboloid and compare

them to the actual values.

x	y	z	Pred (Scaled)	Actual (Scaled)	Pred (Real)	Actual (Real)
200	150	25	1.19187	1.19070	31.95191	20.07000
0	0	0	-4.86145	-5.03438	-61351.35938	-63104.92000
400	300	50	-4.93359	-5.03438	-62082.93359	-63104.91000
150	78	100	-0.12526	-0.12177	-13324.37207	-13288.96000

Table 2.1: Sample Model Predictions

As shown in Table 2.1, the scaled real values are quite close to the actual values calculated using Equation 2.1, however, there is still some unecessary error. Methods to further improve upon this error will be implemented and experimented with in future sections. For this initial test, these values are adequate.

After the model learned the dataset, to test if the model could be used to accurately predict the input values that create the maximum speed, a tensor of random input values (x, y, z) is optimized using another ADAM optimizer, and the loss function is the negative of the model's predictions to perform gradient ascent. After the random input tensor is optimized, the resulting input parameters to create the optimal speed predicted by the model were found to be:

$$(x, y, z)$$

2.3 Spherical Dataset

2.3.1 Dataset Generation