Developing Artificial Neural Networks Surrogate Model for Simple Inverted Pendulum Simulation

Ikhwan Syatricha Hidayatullah

Faculty of Mechanical and Aerospace Engineering, Bandung Institute of Technology
Jalan Ganesha No. 10 Bandung 40132, Indonesia

*Corresponding E-mail: ikhwansyatricha@gmail.com

Abstract. With the rise of artificial intelligent and its sub categories like machine learning and artificial neural network in recent decades. Machine learning also found a lot of use in engineering such as in design and optimization. In this paper we will explore the possibility of implementation of machine learning in the form of multi-perceptron artificial neural network model as a substitute for numerical simulation of a simple mechanical system. We managed to trained two neural network models to mimics the behavior of numerical inverted pendulum model. The neural network model with more parameters perform better than the other one, in which the latter deviate wildly from the numerical model. We also found that neural network model over time will have small deviation from numerical model. It is clear that the neural network models that we have developed still a long way from substituting a numerical model, even if it is quite accurate for a short period of time before it's deviating significantly or even detached completely from the numerical model.

Keywords: machine learning, deep learning, simulations, euler integration, inverted pendulum, model comparison

1. Introduction

In the last few decades, artificial neural network and artificial intelligent field in general has been on the rise. Especially in an online environment such as at YouTube's recommendation system and Ad Targeting System of Google, Facebook, and the like [2]. Not only on online tech field, artificial intelligent and it's sub categories such as machine learning and artificial neural network which would be the focus of this paper were have also been implemented in engineering. Mainly to assist on engineering design and optimization [3].

In this paper, we are developing a neural network model to replace a numerical simulation model of inverted pendulum system. The neural network model will be trained with data generated using numerical simulation.

2. Inverted Pendulum Simulation and Data Generation

To be able to train neural network model, we need to obtain the data first. A numerical simulation model will be used to generate our training data.

2.1 Inverted Pendulum Equation

To actually create a numerical simulation model, first we need to determine what model that we will be using. The following model is what we will be using

There are few assumptions that need to be made. First is that friction will be ignored. And the second is that the mass of the rod connecting the cart and the pendulum assume to be insignificant or massless. We then can derived the following equations for acceleration of the pendulum and the cart

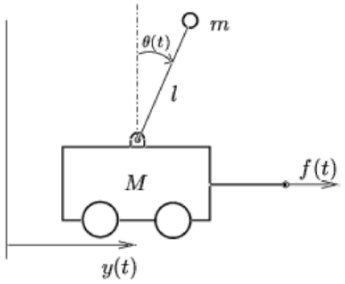

Fig. 1: Inverted Simulation Model. [1]

$$\ddot{y} = \frac{F(x) - mg\sin\theta\cos\theta + ml\dot{\theta}^2\sin\theta}{M + m - m\cos^2\theta} \tag{1}$$

$$\ddot{\theta} = \frac{g\sin\theta - \ddot{y}\cos\theta}{l} \tag{2}$$

Which then can be used to calculate the velocity and position for the next frame of our simulation. This can be done by integrating acceleration ($\ddot{y}$ and $\ddot{\theta}$) using Euler method for a selected delta-time dt.

To calculate cart velocity ($\dot{y}$) and pendulum angular velocity ($\dot{\theta}$)

$$\dot{y}_{t+1} = \dot{y}_t + \ddot{y}dt \tag{3}$$

$$\dot{\theta}_{t+1} = \dot{\theta}_t + \ddot{\theta}dt \tag{4}$$

Equations (3) and (4) then can be used to calculate angular position (θ) and cart position (y)

$$y_{t+1} = y_t + \dot{y}dt \tag{5}$$

$$\theta_{t+1} = \theta_t + \dot{\theta}dt \tag{6}$$

2.2 Data Generation with Python Scripts

With the model and its respected governing equation obtained, we then wrote a Python scripts based on said equation. For our simulation we will set delta-time as 0.01 seconds. Snipped of what will be captured from the python simulation into CSV files can be seen in Table. I

TABLE I: Snipped of Captured Simulation Data

Index	Features					Labels	
	$F(x)$	$\sin\theta_{In}$	$\cos\theta_{In}$	$\dot{y}_{In}$	θ_{In}	$\Delta\dot{y}_O$	$\Delta\theta_O$
1	-61.0	0.529919	-0.848048	0.000000	0.000000	-0.114379	-0.022507
2	-61.0	0.530110	-0.847929	-0.114379	-0.022507	-0.114376	-0.022489
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
2999	-61.0	0.679259	-0.733898	-332.716778	-0.747652	-0.111219	-0.007494

We will generate 60000 of those tables with varying initial condition (θ_0 and $F(x)_0$). All the files then will be joined into three separated CSV files. The three separate partition were training data which is 60% of total data and also evaluation and test data which is 20% of total data each.

3. Deep Neural Networks Model

With our data ready, now is the time to actually develop our neural network model. In this section we will discuss the process of feature extraction and labels selection. The main idea of our surrogate model is to behave more or less the same with the numerical model. What that means is that our neural network model will receive current inverted pendulum state ($F(x)$, θ, $\dot{\theta}$, $\dot{y}$) and then produce the state of our inverted pendulum (θ, $\dot{\theta}$, $\dot{y}$) for the next time-step. But to just use the raw data to train our NN model will not yield the best results. And that is why we have to perform some features extraction and label selection so that our neural network can focus on the important characteristic of the inverted pendulum model.

3.1 Tilt Angle of Pendulum's Arm (θ)

Tilt angle (θ) cannot be directly used as one of our feature. The reason is that since tilt angle (θ) is a periodic data, its range are limited between 0 $radians$ and 2π $radians$. This results in a sudden jump of value every time the angle θ move from positive to negative value or when it's move into value larger than 2π. This behavior can be observed on Fig. 2

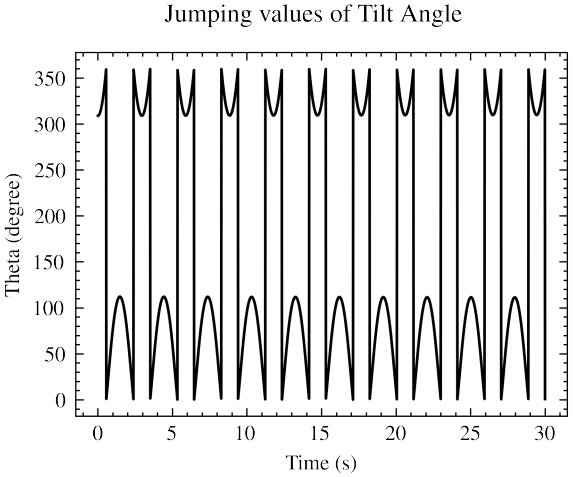

Fig. 2: Angle θ Sudden Jump.

This is a problem because if we take the value of 0.5° and 359.5°, the difference between the two will be 359°. In reality the difference between the two is only 1°. This will caused problem on our neural network model to follow sharp jump near the 0° and 360° points. That will in turn limit our model's ability to make an accurate prediction.

To mitigate this problem, we can separate angle θ into two features $\cos\theta$ and $\sin\theta$ instead of only one . $\sin\theta$ and $\cos\theta$ is a continues periodic function which doesn't have a sharp jump as seen in Fig. 3. And we can also convert it back into angle θ by using $\tan^{-1}\frac{y}{x}$ function [4].

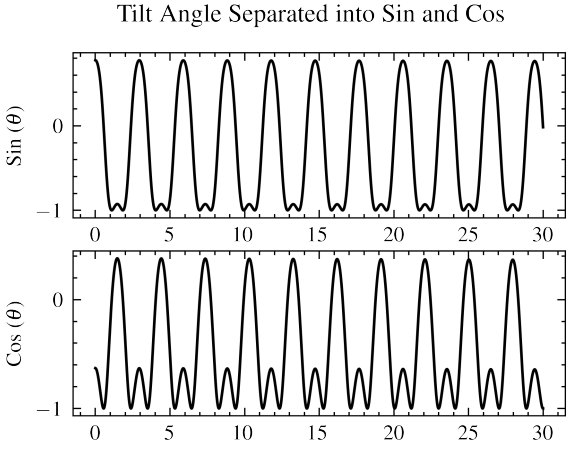

Fig. 3: θ Separated.

3.2 Labels Selection

Obviously we can just directly use θ, $\dot{\theta}$, $\dot{y}$ as our labels to train our model. But we can lift some strain from our model by perform some processing to our label before we use it for training. This is where our understanding of the system become really helpful.

We knew from equations (3), (4), (5), and (6) that y and θ can be calculated using Euler integration of $\ddot{y}$ and $\ddot{\theta}$ which in turn calculated from $\dot{y}$ and $\dot{\theta}$. So rather than using θ,

$\dot{\theta}$, $\dot{y}$, and y as our labels, we can instead use $\Delta\dot{\theta}$ and $\Delta\dot{y}$ which is closer to $\dot{y}$ and $\ddot{\theta}$ but also cut the integration part. By doing this we can let our model 'learn' the important characteristics of our system instead of needing to also 'learn' about integrating $\ddot{\theta}$ into $\dot{\theta}$ and then integrating it into θ. In which we can do that process manually.

3.3 Neural Networks Topology and Training

Now we have $F(x)$, $\sin\theta$, $\cos\theta$, $\dot{y}$, and $\dot{\theta}$ as our features with $\Delta\dot{\theta}$ and $\Delta\dot{y}$ as our labels. Which means our neural network will have 5 input and 2 output. For the hidden-layers we will use two configurations. The first one is three hidden-layers with 30 nodes on the first hidden-layer, 30 nodes on the second, and 10 nodes for the last hidden-layer or we can write it as [30, 30, 10] configuration. The second configuration is [50, 50, 20] configuration. We can latter compare both configuration. A preview of our neural networks topology can be seen at Fig. 4, although in reality the first and second hidden layers has more nodes than shown in the Figure.

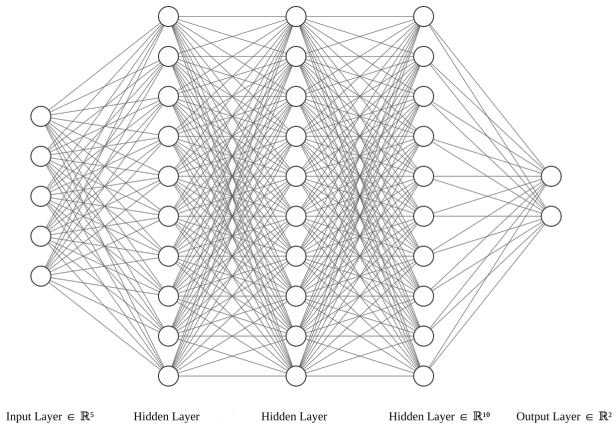

Fig. 4: Diagram of Neural Network Topology.

As stated before, our data will be separated into three data sets. The first one is training set which will be used to train our model. The second one is evaluation set, this data will be used to evaluate our model to see whether our model has been starting to overfit or not, this data used to signal early stopping. And the last one is test set which used to measure how good our model's predictions is.

4. Results and Discussion

After training both model with each training session spend more than 13 hours. Our model is now ready to be evaluated and make some predictions.

4.1 Mean Square Error (MSE)

First measure of our models ability in making prediction on unseen data is of course its MSE. Below is our model's MSE on the test data set for both configurations.

From Table. II we can see that the [50, 50, 20] model have better MSE, as it is smaller than MSE of [30, 30, 10] model. Although it's important to notes that the difference is not too significant.

TABLE II: Model's MSE Against Test Data

Model Configuration	Mean Square Error (MSE)
[30, 30, 10]	9.3160e-07
[50, 50, 20]	6.6212e-07

4.2 Comparison with Numerical Model

To actually see our models ability to mimics the numerical model we need to put it side by side with the numerical model. From the test data set we can take the best and the worst prediction made by our neural network models. The best prediction made by our models can be seen in Fig. 5. And the worst prediction can be seen in Fig. 6

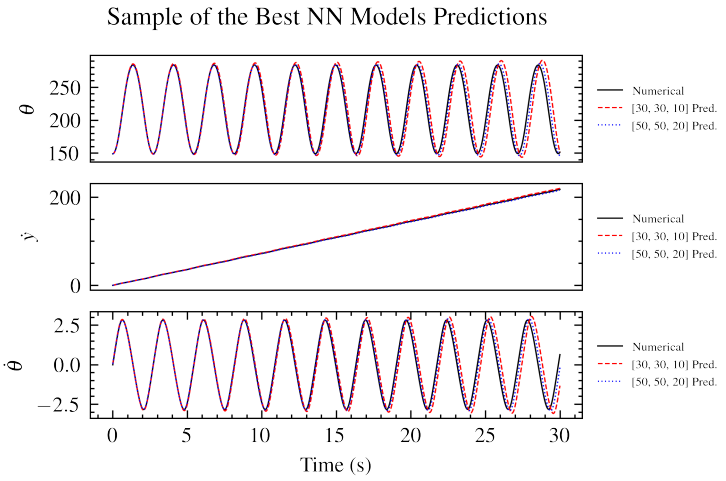

Fig. 5: Best Prediction of Neural Network Models.

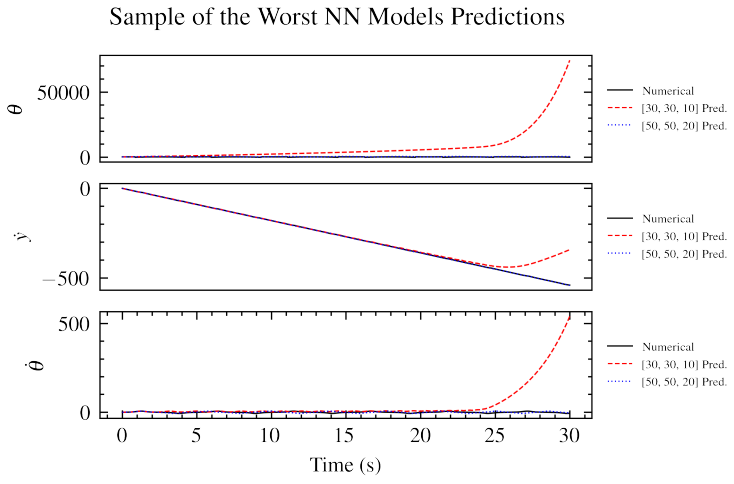

Fig. 6: Worst Prediction of Neural Network Models.

As what can be seen from figures above, both neural networks models deviate from the numerical model. The amount deviations at $t = 30\ seconds$ are as written at Table. III. It's clear from Fig. 6 that once our model deviate enough, it will run wildly and its deviation will keep growing as what happen to our [30, 30, 10] model. This could indicate that [30, 30, 10] topology is still underfit

and doesn't able to captures characteristics of the inverted pendulum as well as the [50, 50, 20] models.

TABLE III: Deviation of Model's Predictions from Numerical Model

Model's Prediction		Deviation		
		θ (rad)	$\dot{y}$ (m/s)	$\dot{\theta}$ (rad/s)
Best	[30, 30, 10]	0.000007	2.216417	1.966594
	[50, 50, 20]	0.094312	1.848403	0.823167
Worst	[30, 30, 10]	1298.365944	197.545242	545.15207
	[50, 50, 20]	8.993692	1.455308	1.113509

It is also very clear that our models with [50, 50, 20] configurations perform better. But even it in the best scenario still deviate from the numerical model, although the amount of the deviation is comparable with [30, 30, 10] configuration. This means that as long as the use case of the surrogate model requires more accurate prediction, the surrogate models still is not a preferred substitute for numerical model. Because it's not as accurate and also will deviate wildly at some edge cases as seen at Fig. 6.

5. Conclusion and Recommendation

After developing two configurations of neural network models aimed as a substitute for a numerical models. We can conclude that neural network models did managed to mimics the behavior of numerical model for a period of time before it detached from the numerical model. With neural network model with [50, 50, 20] configuration performed better it's still attached to the numerical model with little deviation compared to model with [30, 30, 10] at worst scenario. This can be seen at Table. III also at Fig. 5 and Fig. 6. The MSE at Table. II also shows that model [50, 50, 20] is better than [30, 30, 10] model, although not by much.

For further research, it is recommended to implement some kind of topology optimization method to actually get the best topology for the neural network models. Such as Bayesian methods or even just a simple iterative search. I would also recommend to used GPU (Graphical Processing Unit) version of TensorFlow to speed up training process compared to CPU version of TensorFlow. Future research can also try different distribution of initial condition on data generation process rather than just to randomize both $F(x)$ and θ. We also recommend future research to develop the surrogate models with different machine learning structure such as recurrent neural network (RNN) for example.

Appendix A

In the making of this paper, a few resources such as python scripts, trained models, and generated data were made. Few of this files can be accessed through the following GitHub repository: https://github.com/perfect-less/Simulter-HPC Although not everything can be uploaded into the repositories as the size of generated data are quite large, the important components of this projects are available at the said repositories.

References

[1] Simple inverted pendulum. Last accessed 20 December 2021.
[2] Adilin Beatrice. Top companies using machine learning in a profitable way, 2021. Last accessed 19 December 2021.
[3] Jean Thilmany. Giving design jobs to machines, 2020. Last accessed 20 December 2021.
[4] Lyndon White. Encoding angle data for neural network, 2016. Last accessed 22 December 2021.