THE HONG KONG POLYTECHNIC UNIVERSITY
香港理工大學

INDUSTRIAL CENTRE
工業中心

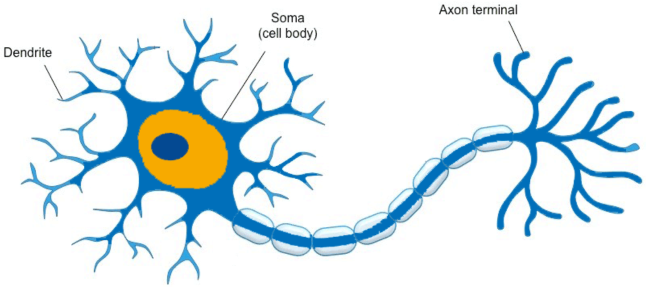
TM1117 AI
Basic Deep Learning

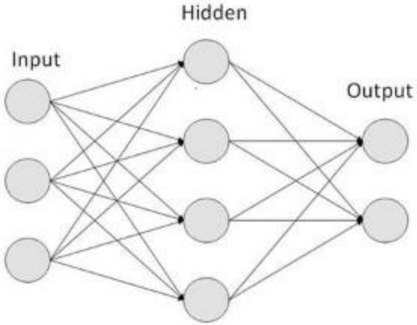
Instructor: Summer Lo

Opening Minds • Shaping the Future
啟迪思維 • 成就未來

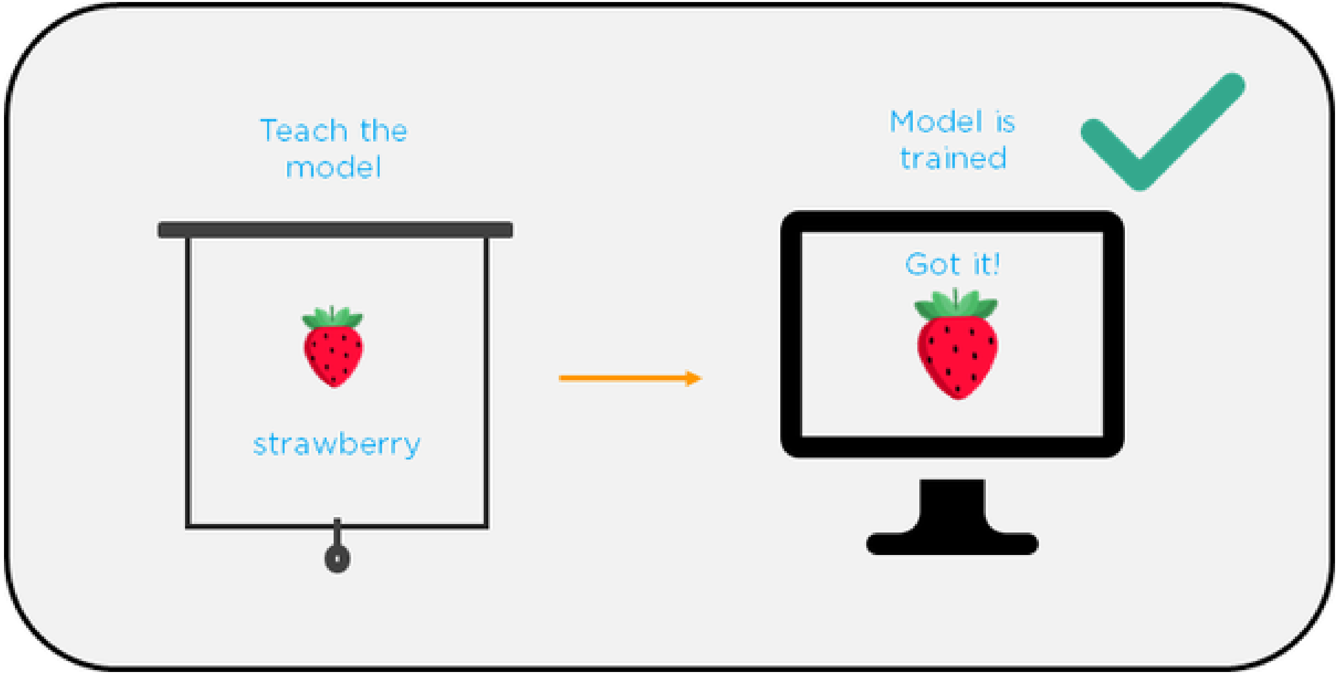
Neural network

> Neural networks, also known as artificial neural networks (ANNs) , are a subset of machine learning and are at the heart of **deep learning** algorithms. Their name and structure are inspired by the human brain, mimicking the way that biological neurons signal to one another.

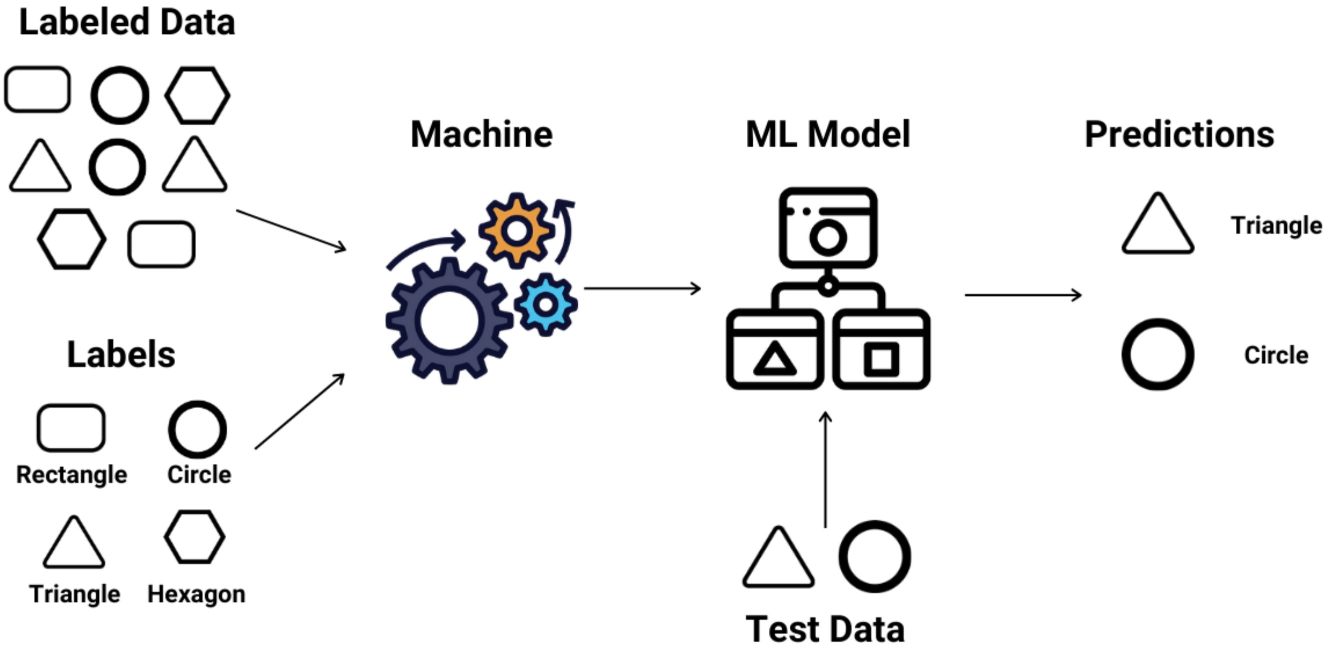




Supervised Learning



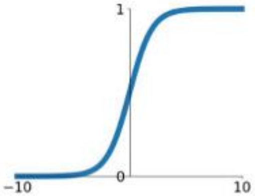
Supervised Learning



Activation function

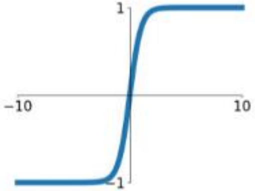
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



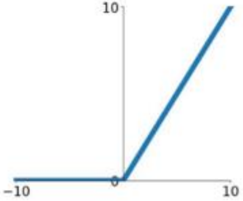
tanh

$\tanh(x)$



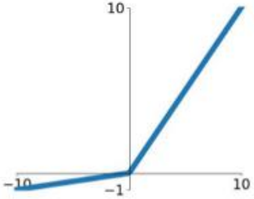
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

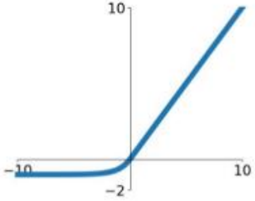


Maxout

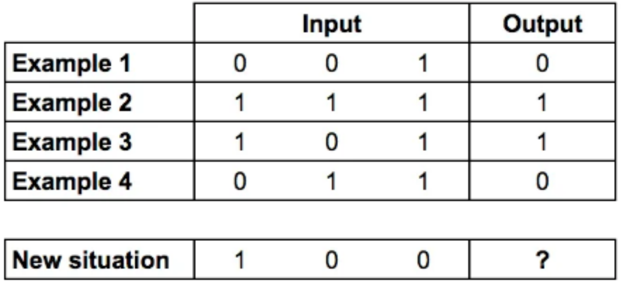
$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

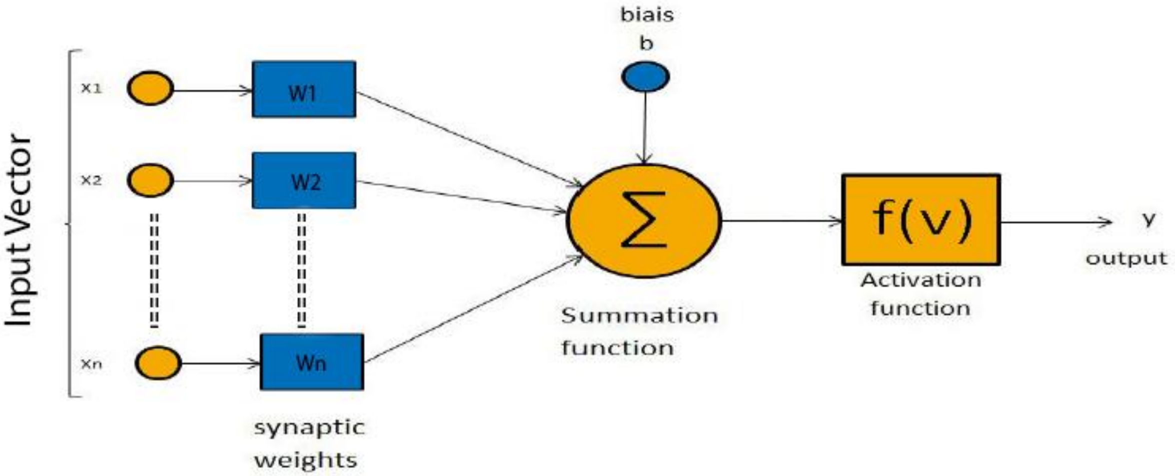


Single-layer Neural Network



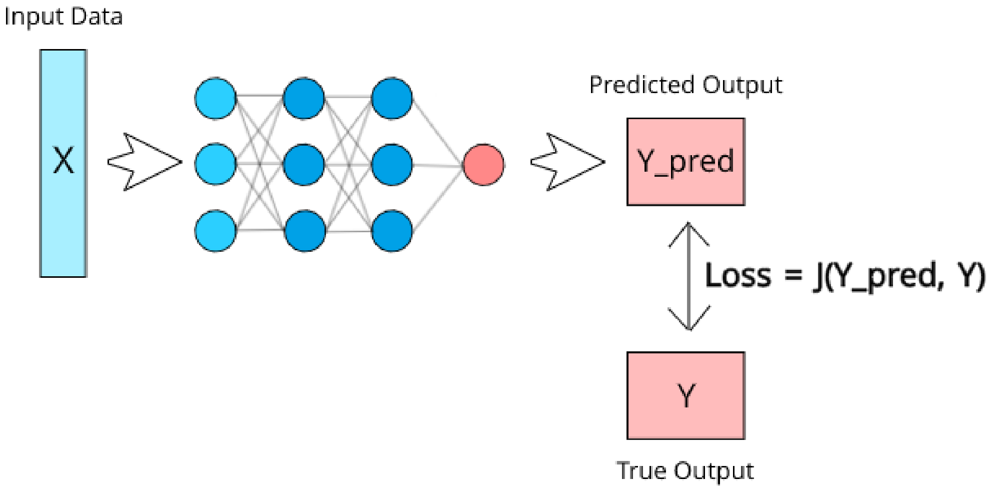
	Input			Output
Example 1	0	0	1	0
Example 2	1	1	1	1
Example 3	1	0	1	1
Example 4	0	1	1	0

New situation	1	0	0	?



Single-layer Neural Network

- A loss function is a function that compares the target and predicted output values; measures how well the neural network models the training data.
- When training, it is aimed to minimise this loss between the predicted and target outputs.



An example of a Loss function – MSE

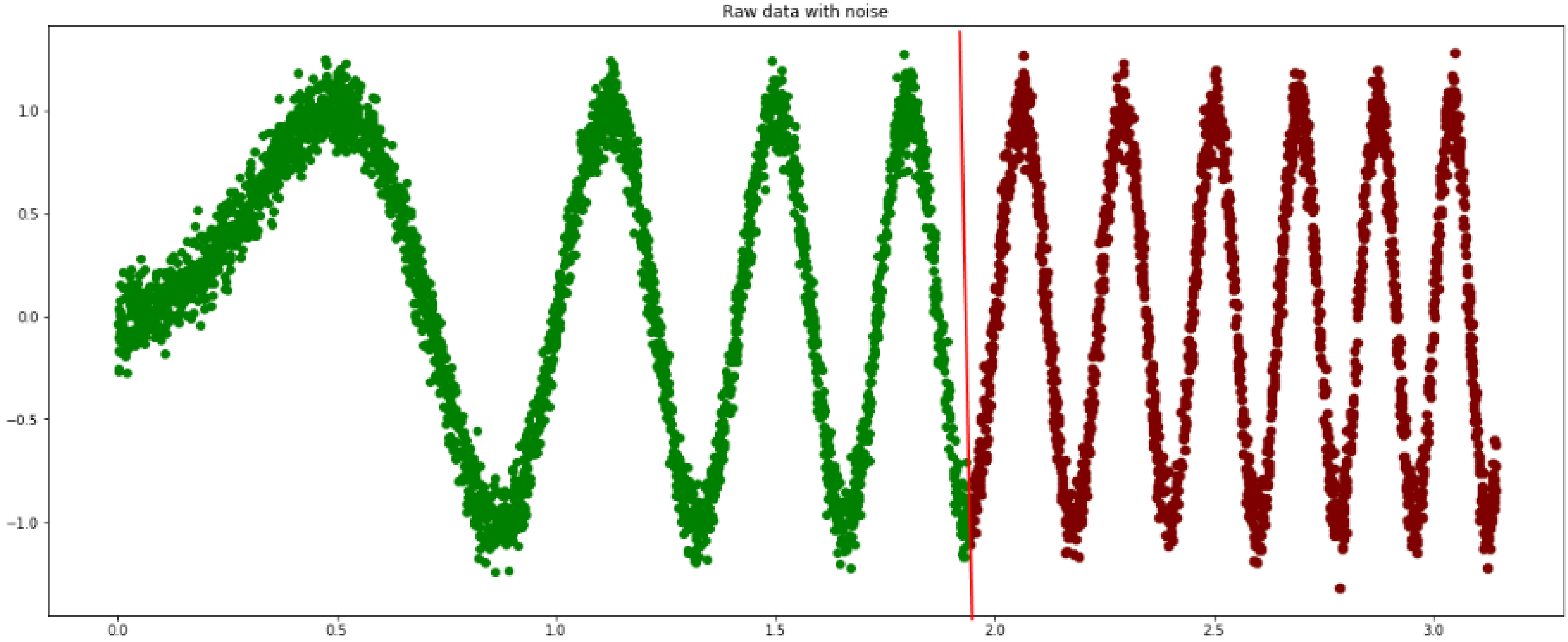
$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

MSE	= mean squared error
n	= number of data points
Y_i	= observed values
$\hat{Y}_i$	= predicted values

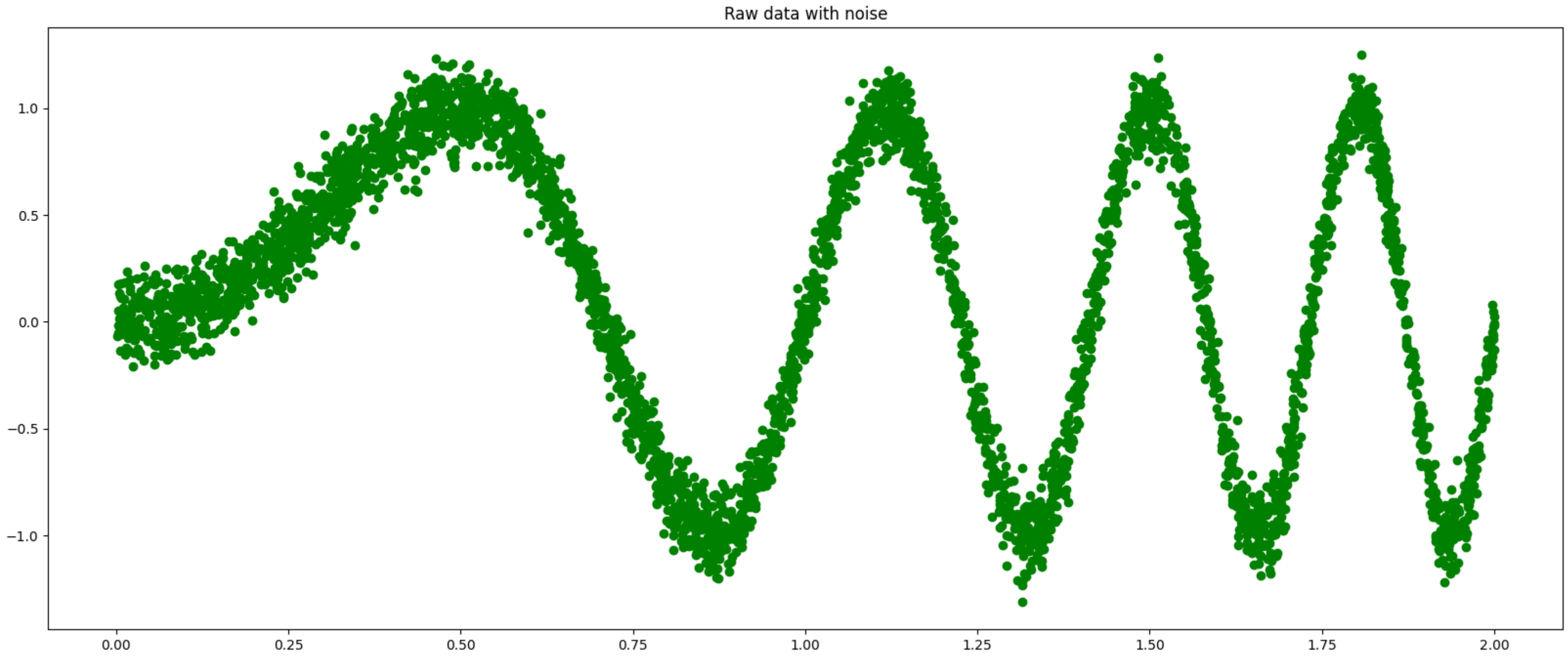
Task 1

Predict the future - function modeling

Prediction – Function modeling

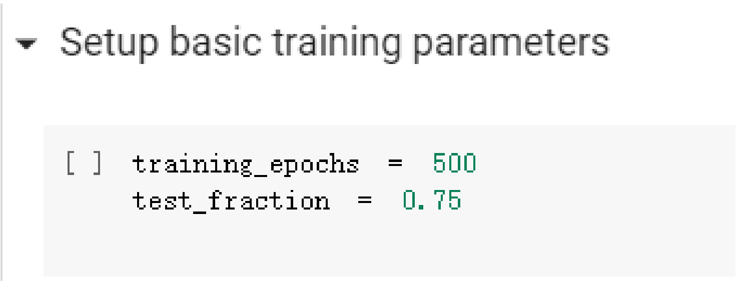


Prediction – Raw data



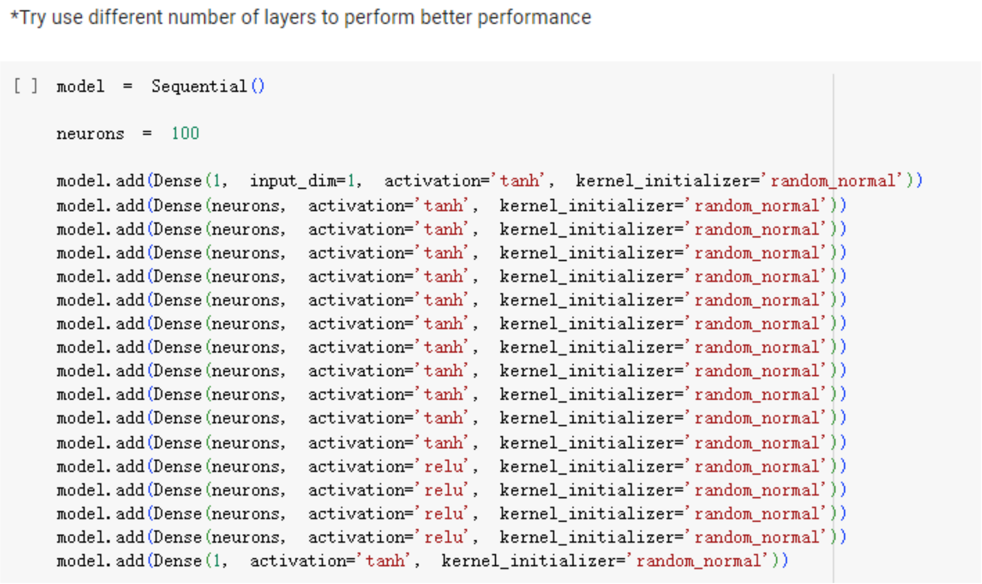
Prediction – Fine-tuning the model

>Modify the parameter and architecture to obtain better result



Setup basic training parameters

```
[ ]  training_epochs  =  500
     test_fraction  =  0.75
```



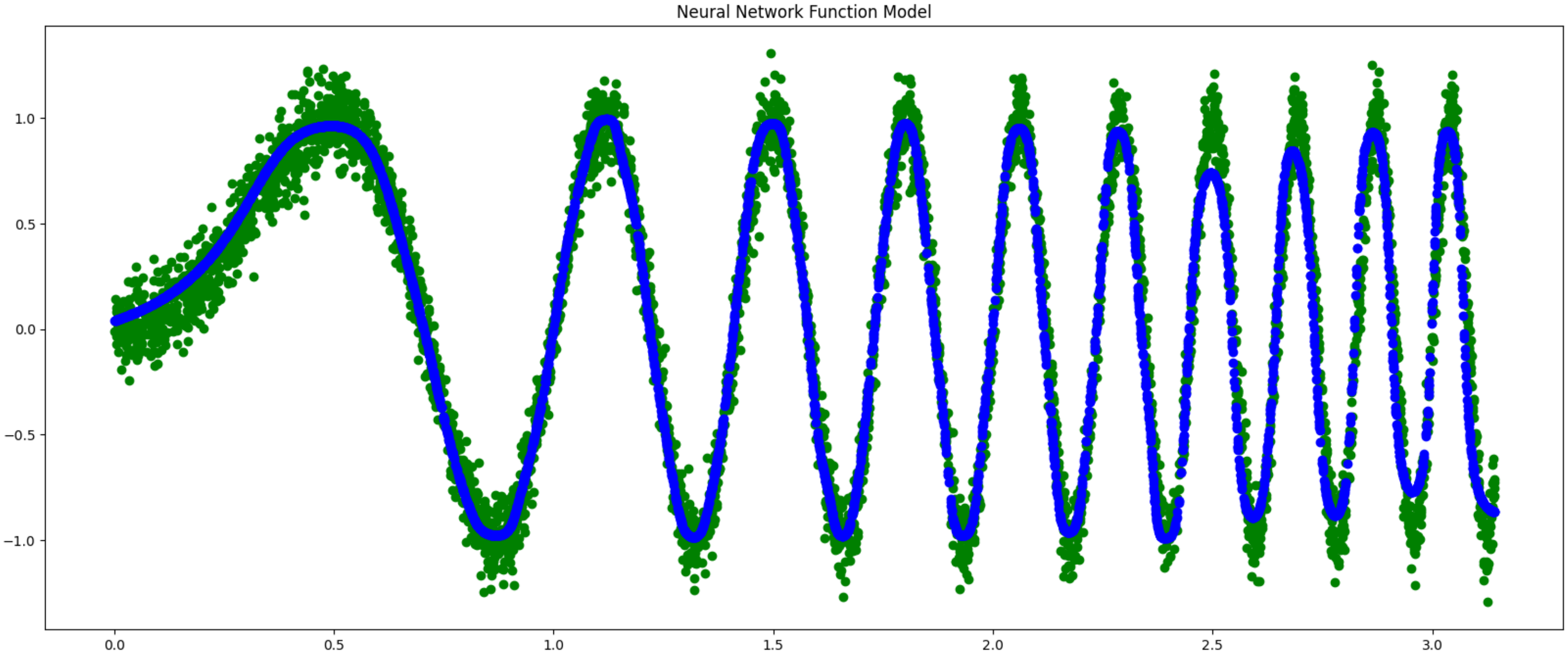
*Try use different number of layers to perform better performance

```
[ ] model  =  Sequential()

    neurons  =  100

    model.add(Dense(1,  input_dim=1,  activation='tanh',  kernel_initializer='random_normal'))
    model.add(Dense(neurons,  activation='tanh',  kernel_initializer='random_normal'))
    model.add(Dense(neurons,  activation='tanh',  kernel_initializer='random_normal'))
    model.add(Dense(neurons,  activation='tanh',  kernel_initializer='random_normal'))
    model.add(Dense(neurons,  activation='tanh',  kernel_initializer='random_normal'))
    model.add(Dense(neurons,  activation='tanh',  kernel_initializer='random_normal'))
    model.add(Dense(neurons,  activation='tanh',  kernel_initializer='random_normal'))
    model.add(Dense(neurons,  activation='tanh',  kernel_initializer='random_normal'))
    model.add(Dense(neurons,  activation='tanh',  kernel_initializer='random_normal'))
    model.add(Dense(neurons,  activation='tanh',  kernel_initializer='random_normal'))
    model.add(Dense(neurons,  activation='tanh',  kernel_initializer='random_normal'))
    model.add(Dense(neurons,  activation='tanh',  kernel_initializer='random_normal'))
    model.add(Dense(neurons,  activation='relu',  kernel_initializer='random_normal'))
    model.add(Dense(neurons,  activation='relu',  kernel_initializer='random_normal'))
    model.add(Dense(neurons,  activation='relu',  kernel_initializer='random_normal'))
    model.add(Dense(neurons,  activation='relu',  kernel_initializer='random_normal'))
    model.add(Dense(1,  activation='tanh',  kernel_initializer='random_normal'))
```

Prediction – Final Result


Neural Network Function Model