SIMULATION STUDY OF ORBIT CORRECTION BY NEURAL NETWORK IN TAIWAN PHOTON SOURCE

M. S. Chiu[†], C Felix, Y. S. Cheng, F. H. Tseng, H. J. Tsai, G. H. Luo

National Synchrotron Radiation Research Center, Hsinchu, Taiwan

Abstract

Machine learning has been applied in many fields in recent decades. Many research articles also presented remarkable achievements in either operation or designing of the particle accelerator. This paper focuses on the simulated orbit correction by neural networks, a subset of machine learning, in Taiwan Photon Source. The training data for the neural network is generated by accelerator toolbox (AT).

INTRODUCTION

The Taiwan Photon Source (TPS) [1,2] is designed as a 3 GeV synchrotron light source, encompassing a 518.4 m circumference. The lattice structure of the storage ring consists of 24 Double-Bend Achromat (DBA) cells, providing 18 short straight sections (7m) and 6 long straight sections (12 m). Three long straight sections, located at 3-fold symmetric position, adopt symmetrical double mini-β_y lattice in which a set of quadrupole triplet is installed in the middle of the long straight section to accommodate double undulators. Figure 1 shows the optical functions of the double mini-β_y lattice for 1/3 TPS.

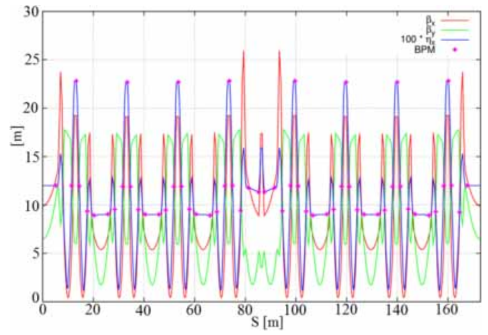


Figure 1: Optical functions of the double mini-β_y lattice for 1/3 TPS storage ring.

Each DBA cell is outfitted with 7 beam position monitors (BPMs). Two BPMs installed in the injection section are unused. There are six additional BPMs installed in three double mini-β_y sections. The TPS storage ring employs 72 horizontal and 96 vertical corrector magnets to define the electron golden orbit, which is monitored by 172 BPMs. In routine operation for user experiments and maintaining long-term orbit stability, each insertion device is equipped with orbit feed-forward table to compensate itself 1st and 2nd order residual integral fields, while gap or phase of the insertion device are moving. Additionally, TPS storage ring is also equipped with a fast orbit feedback systems involving 96 fast correctors (FC) in horizontal and vertical direction, and the RF feedback system.

Figure 2 shows the positions of BPMs, slow orbit corrector magnets (trim coil wound on the sextupoles) and fast orbit corrector magnets in a DBA cell.

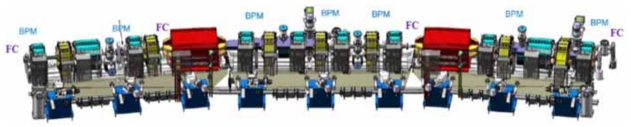


Figure 2: DBA cell in TPS storage ring. Dipole is printed in red colour. Quadrupole is in blue colour. Sextupole is in yellow. Slow orbit corrector magnets are the trim coils wound on the sextupole magnets.

In daily operation for user experiments, the orbit correction and control uses a measured orbit response matrix and singular value decomposition (SVD) algorithm. BPMs are used to monitor the electron beam's orbit, apply SVD to calculate the pseudoinverse of the orbit response matrix, the desired strengths of the corrector magnets can be derived, then apply the calculated current to corrector magnets to bring the electron orbit closer to the target orbit. This traditional method is rooted in physics and well-established principles of beam dynamics in particle accelerators. However, applying machine learning to particle accelerators is growing. D. Schirmer [3, 4] published a paper talking about orbit correction with machine learning at the synchrotron light source DELTA. We also try to create a neural network model to do orbit correction to explore the benefits and drawbacks of utilizing machine learning for orbit correction in TPS storage ring.

MACHINE LEARNING

Machine learning (ML) is a subset of artificial intelligence (AI). It can enable computers to learn from large amounts of data and make predictions or decisions without being explicitly programmed. ML can be roughly classified into three types: supervised learning, unsupervised learning, and reinforcement learning. In supervised learning, the model is trained on labeled datasets, meaning that the input data is paired with output or target values. The goal of supervised learning is to search for a mapping from inputs to outputs. Once adeptly trained, the model can make precise predictions or classifications when encountered with new or unseen data. In unsupervised learning, the model works with unlabeled data and learns patterns without predefined outcomes. It's often used for clustering and dimension reduction. In reinforcement learning, agents learn by interacting with an environment to achieve specific goals.

† chiu.ms@nsrrc.org.tw

Neural network, a subset of machine learning, is a computational model inspired by the human brain's structure. It can assist us to find out the rules or relationships between the input and output of a nonlinear or complex system. A typical architecture of the feedforward neural network is shown in Fig. 3. For simplicity, we will use one hidden layer as an example. The circles in each layer stand for neurons, called nodes. The linking arrows in-between layers show the signal transduction pathways. The input layer is used to feed data. The output layer gives us the predictions by the neural network. The hidden layer is the main processing units in the neural network to process the data. Usually in each neuron, it executes two things: (a) sum the data passed from the previous layer multiplied by a weight matrix W and then add a bias value B, (b) pass the weighted sum of the data to an activation function f to make a transformation. After that, the transformed data is sent to next layer.

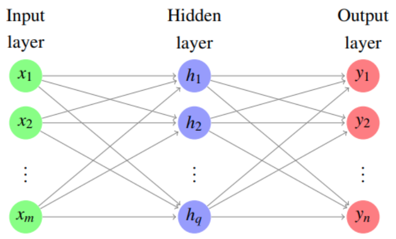


Figure 3: Scheme of a typical feedforward neural network model, considering input, output and one hidden layer

The signal transduction from input layer to the hidden layer can be formulated as the following equation:

$$H = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_q \end{bmatrix} = f(W_1 X + B_1) = f\left(\begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1q} \\ w_{21} & w_{22} & \cdots & w_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ w_{q1} & w_{q2} & \cdots & w_{qm} \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}\right),$$

resulting from the following equation to give the prediction of Y.

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = W_2 H + B_2 = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1q} \\ w_{21} & w_{22} & \cdots & w_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1} & w_{n2} & \cdots & w_{nq} \end{bmatrix} \times \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_q \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix},$$

where X is the input data, Y is the output of the neural network. H stands for the output from the hidden layer. W_1 and W_2 are weight matrices. B_1 and B_2 are bias vectors. The subscript m, q, and n are the neuron number in the input layer, hidden layer, and output layer, respectively. f is an activation function. Here, we assume the activation in the output layer is linear. Figure 4 shows commonly used activation functions.

Neural Network Workflow

The flow chart of the neural network application is the following:

1. *Data collection:* Scaling and normalizing data, then splitting data into training, validation and test sets
2. *Build a neural network:* Select an appropriate neural network architecture (e.g. feedforward, recurrent, convolution) based on problem type (e.g. regression, classification, *et al.*), and assign the number of layers, neuron number in each layer, activation function (e.g. sigmoid, tanh, ReLu, *etc.*);
3. *Compile the model:* Specify the loss function (e. g. mean square error, etc.), optimizer (e.g. adam, sgd, *etc.*) that adjusts the model's weights and bias, evaluation metrics (e.g. accuracy, mean absolute error) to monitor during training;
4. *Train the model:* Specify the batch size, the number of epochs (training iteration times), and using training set of data;
5. *Evaluate the model:* Evaluate the model's performance by using validation data set;
6. *Visualize the training progress:* Plot training and validation loss over epochs to assess how well the model is learning.
7. *Hyperparameter tuning:* Training model with different learning rates (step size during training), batch size (number of data sets used in each iteration of training, epoch (training times of passing data sets through network model) to avoid underfitting and overfitting, number of layers, neurons per layer;
8. *Make predictions:* Use the trained model to make prediction on new data.
9. *Model deployment:* If the trained model performs well, save it for deployment.

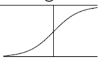
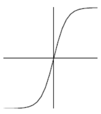
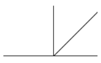
Name	Function	Derivative	Figure
sigmoid	$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1 - f(x))^2$	
tanh	$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$f'(x) = 1 - f(x)^2$	
ReLU	$f(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0. \end{cases}$	$f'(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0. \end{cases}$	
softmax	$f(x) = \frac{e^x}{\sum_i e^x}$	$f'(x) = \frac{e^x}{\sum_i e^x} - \frac{(e^x)^2}{(\sum_i e^x)^2}$	

Figure 4: Basic activation function used in neural network

Neural Network Training

Before starting the training process, the weight matrix elements are randomly assigned. During the training process, the optimizer will update the weight matrix and bias values to minimize the loss function, which is defined as the square of the difference between the outputs of the neural network and target values.

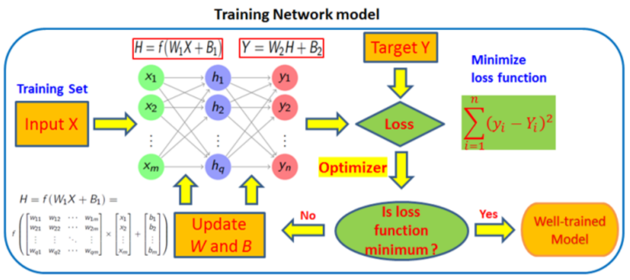


Figure 5: Flow chart for training neural network model.

SIMULATION DETAIL

Accelerator Toolbox (AT) [5] is used to generate training data. Three-thousand sets of 72 horizontal correctors (HC) strengths within ± 2.5 μrad are randomly assigned with the matlab [6] command `rand`. Using a for-loop selects a set of 72 random numbers to assign the strengths of 72 corrector magnets respectively, followed by using the MML command `getx` to get the orbit. Eventually, we have 3000 different orbits associated with 3000 sets of different strengths of the 72 corrector magnets.

Python is used to develop the machine learning application. Tensorflow [7] and keras [8], machine learning packages, are used to build the neural network model. Scikit-learn [9], data mining toolbox, is used to pre-process data, e. g. normalization and split data into training and validation sets of data. For TPS orbit correction in the horizontal plane, the number of input neurons is 172 BPMs, number of output neurons is 72, number of hidden neurons is 172. Figure 6 shows the loss function for training and validation sets of data. The loss function of the training and validation sets of data converges after several training iterations. That means no overfitting and underfitting phenomenon is observed. After training, the trained model is saved with the keras package. Figure 7 shows the accuracy of the trained neural network.

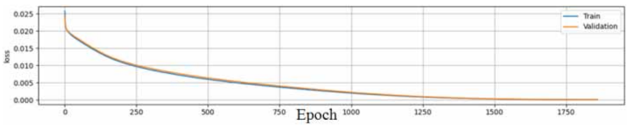


Figure 6: Loss function for training and validation sets of data during training process.

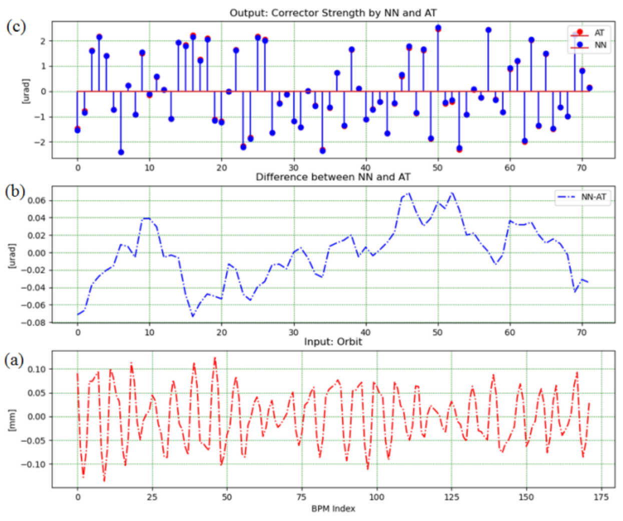


Figure 7: Accuracy of the trained neural network. (a) Test set of input data (172 BPMs' data), (b) Difference of the corrector strength between the prediction from the trained neural network (NN) and AT simulation. (c) Corrector strength: Red is by AT simulation; blue is the prediction by the trained neural network (NN).

In the following, we are going to verify the performance of the trained neural network on orbit correction in AT simulator. Figure 8 shows the implementation of the neural network on orbit correction. The unknown orbit distortion shown in red color in Fig. 9 (b), generated by shifting 249 quadrupoles randomly within ± 3 μm in horizontal plane with the AT command 'setshift', is feed into the trained neural network. The trained network will predict one set of 72 corrector strengths. Using the predicted corrector strengths to correct the orbit distortion in AT simulator and iterate three times. The corrected orbit by the trained neural network is shown in Fig. 9 (b).

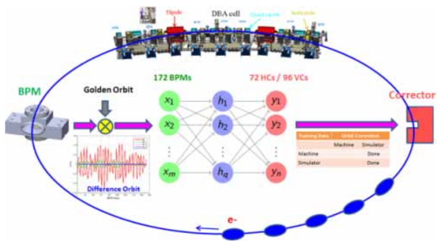


Figure 8: Implement neural network for orbit correction.

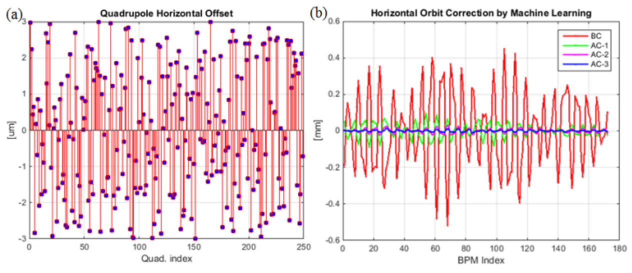


Figure 9: (a) Misalignment quantities of 249 quadrupole magnets within ± 3 μm to generate orbit distortion in TPS storage ring simulated by AT. (b) Orbit correction by neural network: Red is the orbit before correction (BC), green, magenta, and blue are the orbit after correction (AC), iterate 3 times (AC-1, AC-2, AC-3).

SUMMARY

This paper demonstrates the preliminary results of the machine learning application for the orbit correction at TPS storage ring by AT simulator. Even though there is minor residue of orbit distortion after orbit correction by the neural network, it did show great potential to use a neural network for orbit correction. There is still room for improving the performance of the neural network. The next step will to be to apply machine learning on the real machine. Detailed machine studies need to verify the performance of the neural network for orbit correction at TPS storage ring.

REFERENCES

[1] C.C. Kuo *et al.*, "Commissioning of the Taiwan Photon Source", in *Proc. IPAC'15*, Richmond, VA, USA, May 2015, paper TUXC3, pp. 1314-1318. doi:10.18429/JACoW-IPAC2015-TUXC3
[2] M.S. Chiu *et al.*, "Double mini-betay lattice of TPS storage ring", in *Proc. IPAC'11*, San Sebastián, Spain, May 2011, paper WEPC035, pp. 2082-2084.

[3] D. Schirmer, "Orbit correction with machine learning techniques at the synchrotron light source DELTA", in *Proc. ICALEPCS'19*, paper WEPHA138, pp. 1426-1430. doi:10.18429/JACoW-ICALEPCS2019-WEPHA138

[4] D. Schirmer, "A machine learning approach to electron orbit control at the 1.5 GeV synchrotron light source DELTA", in *Proc. IPAC'22*, Bangkok, Thailand, May 2022, paper TUPOPT058, pp. 1137-1140. doi:10.18429/JACoW-IPAC2022-TUPOPT058

[5] https://atcollab.github.io/at/

[6] https://www.mathworks.com/

[7] https://www.tensorflow.org/

[8] https://keras.io/

[9] https://scikit-learn.org/stable/