A Generalized Framework for Machine Re-learning of Complex Process and Kinetic Models

Yuening Wang

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science in Chemical Engineering

University of Washington
2019

Committee:
David Beck
Jim Pfaendtner

Program Authorized to Offer Degree:
Chemical Engineering

University of Washington

Abstract

A generalized framework for machine re-learning of complex process and kinetic models

Yuening Wang

Chair of the Supervisory Committee:
David Beck
Chmical Engineering

Computer modelling is commonly used to simulate complex systems and thus give insight to the problems with lower cost and more efficiently. However, since they usually characterize real system by using complicated mathematical equations, it could be very time-consuming and expensive to run them on the computer. To tackle this problem, artificial neural network (ANN) model, in many cases, has turned out to be successful surrogate model to reduce the computational burden and generate data in faster manner. However, though there is plenty of software available to create neural network, it still takes time to determine every detail for a good model. Therefore, this study was motivated to build a Python module which builds a multi-layer perceptron (MLP) neural network and searches the best model architecture to substitute a numerical simulation model. A synthetic data has been utilized to prototype the module. After every function works out with synthetic data, data from Pseudo-two-dimensional (P2D) battery model has been employed to test the functinality of the module.However, the best prediction gained from over 2000 models trained gives R^2 as only 0.3245. There are several possible causes of not finding a useable model, which could be improper processing of data, unsuitable choices of activation functions, not enough iterations went through,inappropriate optimization method, etc. The module thus needs more work to be perfected.

TABLE OF CONTENTS

LIST OF FIGURES…… 5

LIST OF TABLES…….. 5

Chapter 1: BACKGROUND……... 6

Chapter 2: METHOD

2.1 MODEL IMPLEMENTATION AND SETUP

2.1.1 KERAS INTRODUCTION…….. 9

2.1.2 HYPERPARAMETER SETUP……11

2.2 SYNTHETIC DATA…… 12

2.3 CASE STUDY – PSEUDO-TWO-DIMENSIONAL (P2D) BATTERY MODEL…….. 13

Chapter 3: RESULTS AND DISCUSSION

3.1 RESULTS…….. 16

3.2 ANALYSES AND POSSIBLE SOLUTIONS

3.2.1 DIFFERENT DATA PROCESSING METHOD…… 18

3.2.2 LARGER DATASET…….. 18

3.2.3 ADDING BUILDING BLOCKS FOR COMPLEX-VALUED NEURAL NETWORK…… 19

3.2.4 DIFFERENT SEARCHING ALGORITHM……... 20

3.2.5 DIFFERENT OPTIMIZING ALGORITHM……....21

Chapter 4: CONCLUSION…….. 22

BIBLIOGRAPHY……. 23

LIST OF FIGURES

Figure 1: Workflow of project…… 7
Figure 2: Overall model implementation…… 9
Figure 3: Best fitting results of synthetic data…… 13
Figure 4: Test*100 vs Predicted*10000 of 21544 spectra……. 18
Figure 5: Iteration count vs Best R^2 during grid search process…… 20
Figure 6: Possible forward stepwise searching method……. 20

LIST OF TABLES

Table 1: Running Time Comparison on Different Hardware Platform…….. 10
Table 2: Model Components Setup…… 11
Table 3: Synthetic Data…….. 13
Table 4: Processed EIS Data…… 15
Table 5: Heatmap of Model Configurations of Every Layer……... 17

Chapter 1: BACKGROUND

Mathematical models and virtual prototypes are widely employed to simulate complex kinetic or flux systems to avoid having to conduct expensive results repeatedly, saving time and cost. These models and prototypes generate data on a computer based on certain equations and algorithms. A computer model typically consists of an input vector with several state variables, a mathematical equations system defining the system and an output vector representing the behavior of a real system.[1] Simulation models can capture many more details than analytical models and thus contribute to more robust solutions.[2]For example, the Activated Sludge Model 1 (ASM1) was created by the International Water Association (IMA) for the removal of organic carbon and nitrogen concentration. The model helped to gain insight into wastewater treatment and hence the protection of the water resources.[3]EnergyPlus has also performed well in estimating the total energy consumption of a building over a long period, which is not feasible to achieve with actual measurement due to the high cost and time requirements.[4]

However, many processes are strongly nonlinear, which makes qualifying complex model computation demanding. Further, two types of uncertainty variance – randomness caused by system inherent variability and epistemic uncertainty – resulting from a lack of knowledge of the system can affect the reliability and sensibility of analysis.[1]Given the need to improve model accuracy and processing time, ANN models have been applied to combine/replace simulation models. ANN was inspired by biological neural networks and mimics the learning process of brains.[5][6] It is a framework that can incorporate different machine learning algorithms in working together and performing tasks, such as image recognition, text processing, classification and regression. Neural networks are not constrained by a predefined mathematical relationship between dependent and independent variables and they are highly adaptive to complicated nonlinear relationships.[7] In other words, an ANN model can learn complex nonlinear systems to predict values for the original model. Therefore, for certain problems, ANN can act as a powerful tool for obtaining accurate mathematical model results within a reasonable time, without going through every detail of that process. This capability makes it useful when systems require real-time outcomes.

Many researchers have applied ANN to modify or optimize their computer models. For instance, a three-layer ANN model was used to predict the decolorization efficiency of reactive red 33 in a bubble column reactor. It performed better at predicting reactive red 33 removal compared to response surface methodology (RSM).[8] ANN has also been utilized as a chemistry integrator for large eddy simulation (LES), which represents chemical state-space through optimal chemical kinetic mechanisms.[9]

However, although various software is available for users to build and train a neural network, they require a developer's theoretical background knowledge to wisely choose algorithms, structures or tuned hyperparameters. It is also time consuming and tedious to slightly reset model structures and train them. Therefore, it was motivated for our study to develop Python module which builds a multi-layer perceptron (MLP) neural network for numerical simulation models. This module enables iterations over different model configurations and the identification of the best fitting one among them. It is intended for automatic search of a surrogate MLP model of a nonlinear simulation model.

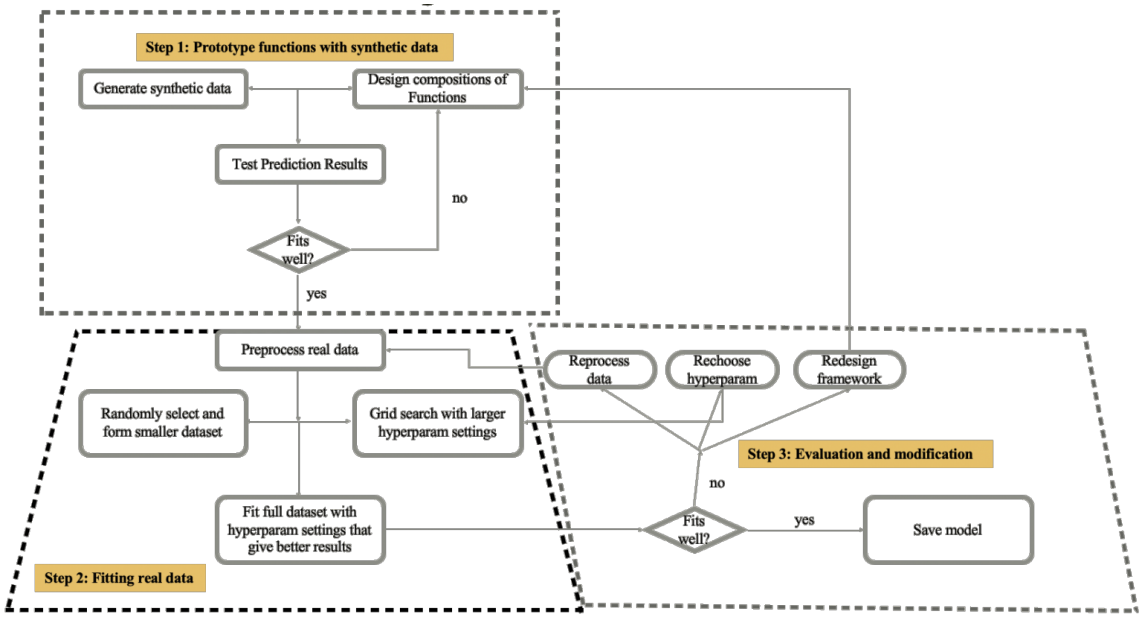

Figure 1. Workflow of project. *Step 1*: A set of nonlinear synthetic data was created to prototype and test components in the module. Model building functions were written using Keras. *Step 2*: As the functions work as expected, data from real simulation models are fitted to find a surrogate machine learning model. A larger number of model configurations are trained with a smaller dataset, among which the models providing

better results are stored. Those selected models are then trained with full datasets, after which the best one is obtained.*Step 3*: If no models can produce predictions as well as the original model, evaluation is conducted. This step repeats designing functions, choosing different hyperparameters or processing data in a different method, so as to identify a neural network that can reproduce the results from the original model.

Chapter 2: METHOD

2.1 MODEL IMPLEMENTATION AND SETUP

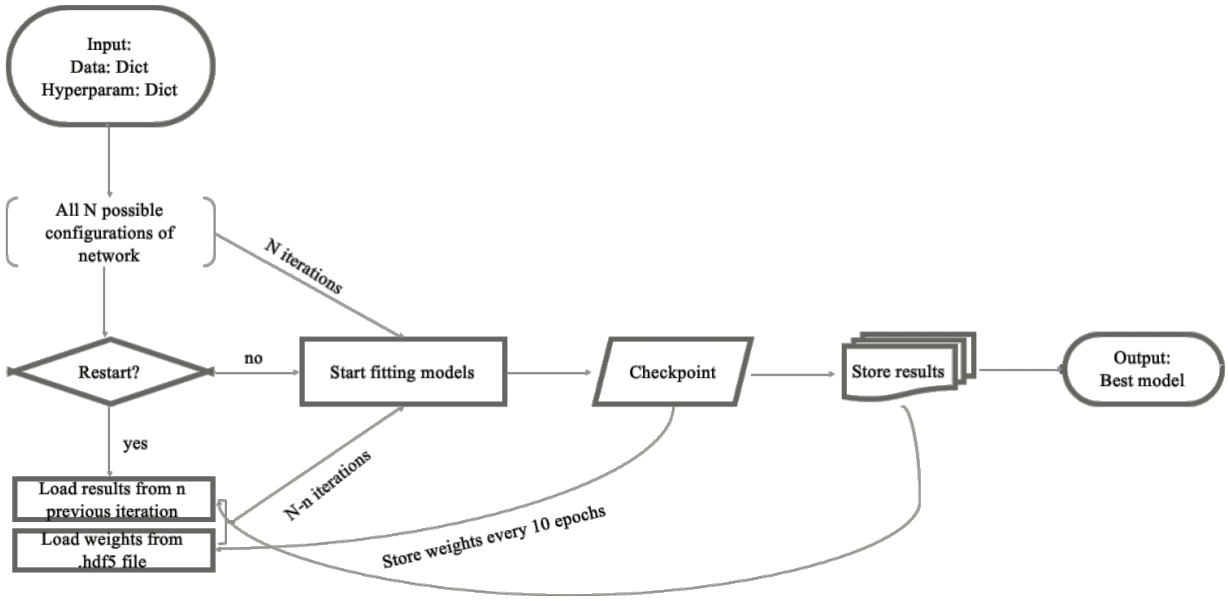

Figure 2. Overall model implementation. The procedure is as follows: (1) data and hyperparameters are input. This step requires knowledge of the original models and data. The data need to be preprocessed and hyperparameters need to be chosen based on analysis of data. (2) Model building and iterations. In this step, the function calculates all possible configurations of the neural network based on the hyperparameter options given at input. Every model is then trained iteration by iteration. Weights acquired during training are stored so that the training can be resumed without repeating trained models if it is stopped before all configurations have been processed. (3) Model saving. The best model will be stored as a JSON file, which allows it to be reproduced.

2.1.1 KERAS INTRODUCTION

There are many types of deep-learning software that can develop neural networks. This study is built over the top of Keras, a high-level neural network API written in Python. It was chosen because it is simple to understand and manipulate and because it offers flexibility, which enables developers to configure the model structures and implement more tasks in the base

language. It also supports multiple backend engines – Tensorflow, Theono and CNTK. This makes it more compatible in different ecosystems. More importantly, a Keras model can be trained seamlessly on CPU and GPU[10]. Running thousands of iterations is very time consuming, while fitting results on GPU can be obtained more significantly than on CPU. Therefore, it is convenient to use Keras because models can be trained on different hardware platforms without revising codes. One more attractive feature of Keras is that it supports multi-GPU data parrallelism, which is very useful in training large numbers of different models.

Table 1. Running Time Comparison on Different Hardware Platform

	Iteration #	Cumulative time(s)	Cumulative time(h)	Time(s)/iteration
CPU(waffle)	1544	1113507.5	309.3	721.2
Interactive node (hyak)	1842	108042.3	30.0	58.7

Notes: 1.Waffle is a CPU cluster sponsored by eScience Studio of University of Washington. The CPU type: Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz

2. Hyak is a service of UW-IT.[22] The nodes type are Intel E5-2650, Intel E5-2650 v2 and Intel E7-4820

2.1.2 HYPERPARAMTER SETUP

Table 2. Model Components Setup

Parameter	Setting	Parameter	Setting
units	function input	biase_constraint	none
activation	function input	epoch	300
use_bias	none	batch_size	10
kernel_initializer	glorot_uniform	patience	20
bias_initializer	zeros	period	10
kernel_regularizer	none	validation_split	0.2
bias_regularizer	none	loss	MSE
activity_regularizer	none	optimizer	adam
kernel_constraint	none	metrics	R_squared
batch_normalization	yes	checkpoint	yes

The model is set up as one nonlinear layer followed by one batch normalization layer. The units or node numbers of each layer, number of hidden layers and the activation function of each layer are chosen for tuning, and need to be determined at input. Epoch size defines the the number of complete passes through the entire training set and batch_size is the number of samples processed before the model is updated. A larger epoch size allows better minimization of error rate. A larger batch size makes a learning process converge slowly with more accurate estimates of the error gradient, while a smaller batch size improves computational efficiency at

the cost of noise in the training process[11]. In the balance of computational time and accuracy, the epoch size was set as 300 and the batch_size as 10. For the purpose of saving time, earlystopping was used. The training would be early-stopped if the loss value or mean squared error did not improve for 20 epochs. There are two parts of checkpoints: Keras built-in checkpoint, which records weight matrix to .hdf5 file every 10 epochs, and external checkpoint, which checks the iteration count and model structure for resuming model searching. An adaptive moment estimation (Adam) optimization algorithm was chosen. This is an adaptive learning rate optimization algorithm that can find individual learning rates for each parameter. It was chosen because it has a fast speed of convergence and good performance in deep-learning models[12][13]. The metric to determine the efficacy of the model is R^2, which is made possible by a customized function feature in Keras. The other hyperparameters were kept as Keras default settings. Additionally, the batch normalization layers were adopted because it enabled a much higher learning rate, reduced the effect of initialization and could eliminate the need for a dropout layer.This method also addressed the problem of internal covariate shift by performing normalization for each training mini-batch[14][15].

2.2 SYNTHETIC DATA

The synthetic data have 10,000 data points and were generated from the following highly nonlinear equation:

$$y = |A|^{|B|} - |B|^{|C|} + e^{|C|} + C - \frac{C^3}{3!} - \frac{C^5}{5!} - \frac{C^7}{7!} - 2.5$$

Table 3. Synthetic Data

A	B	C	y
–1	–1	–1	–0.6065198
–1	–1	–0.9	–0.8138812
…	…	…	…
–1	–1	1	1.04308342
–1	–0.9	–1	–0.5065198
–1	-0.9	–0.9	–0.7234138
…	…	…	…
–0.9	–1	–1	–0.7065198
–0.9	–1	–0.9	–0.9138812
–0.9	–1	–0.8	–1.0863535
…	…	…	…

The data loading process, model building, model training, results store, early stopping, search resuming and model saving processes are all prototyped and tested with synthetic data.

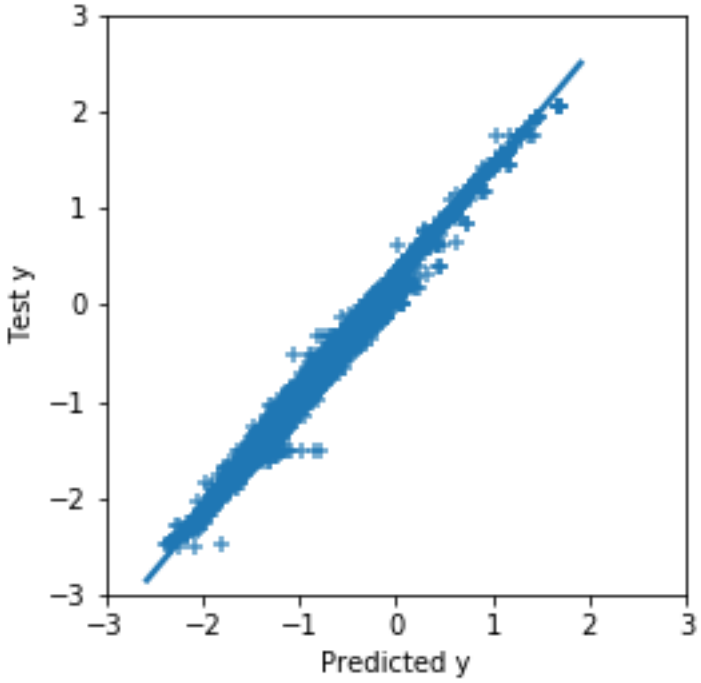

Figure 3. Best fitting results of synthetic data. The hyperparameter combination was [['tanh',50],['tanh',18], 'linear'] and the R^2 was 0.992 with slope of 0.91.

It was found that the Python module that was built could successfully predict simple nonlinear data. The intermediate results were recorded for analysis. The next step was to test the module using data from real simulation models.

2.3 CASE STUDY – PSEUDO-TWO-DIMENSIONAL (P2D) BATTERY MODEL

The P2D model is a set of partial differential equations that can predict publicly available experimental electrochemical impedance spectroscopy (EIS) data. The quantitative analysis of EIS data is of importance for both characterizaion and prognostic applications in many electrochemical systems. M. D. Murbach and D. T. Schwartz developed an open-source platform – the impedance analyzer – and demonstrated its performance for physics-based analysis of experimental EIS spectra with 38800 P2D-based impedance spectra simulation data. 31 parameters were chosen and there were 25 spectra frequency responses[16].

This dataset has been used to test the ability and feasibility of finding a surrogate machine learning model using the functions developed in this study. The challenge of using this

dataset is that the frequency signals are complex numbers. The majority of artificial neural networks rely on real-valued weights and represent information in the same way as human brains, which means it would be odd to use complex numbers[17]. Although there are already various studies that extend the usage of neural networks to complex domains by creating new neuron models and new learning paradigms, this research, due to its time limitation, studied the simplest way to deal with complex numbers: splitting a separate complex number into two parts. This means the responses of data doubled from 25 to 50. The architecture and learning process of neural networks guarantees that the order of reponses will not be mixed during training, which makes it work to propogate real parts and imaginary parts of the complex numbers. 10,000 data points were first tried with different input hyperparameter combinations, followed by training on the full dataset with the hyperparameter setup that gave the best results.

Table 4. Processed EIS Data

run	l_neg[m]	l_sep[m]	l_pos[m]	…	0.001_real	0.001_imag
1	0.00027538	1.11E-05	0.00062949	…	0.00482302	–0.0045896
2	0.00096508	3.27E-05	0.00102063	…	0.00583941	–0.0035232
3	0.00027538	1.11E-05	0.00062949	…	0.00858403	–0.0058215
4	0.00027538	1.11E-05	0.00062949	…	0.00482225	–0.0045886
5	0.00027538	1.11E-05	0.00062949	…	0.00412464	–0.0039947
6	0.00027538	1.11E-05	0.00062949	…	0.00313152	–0.0007604
7	0.00027538	1.11E-05	0.00062949	…	0.01247074	–0.0726213
	…	…	…	…	…	…

Chapter 3. RESULTS AND DISCUSSION

3.1 RESULTS

There were over 2000 iterations with different sets of hyperparameter input running; however, the best R_squared acquired was 0.3250. This result was gained when two hidden layers were involved. The activation functions used from the first layer to the last layer were softplus, tanh, softmax and elu, while the node numbers were 100, 100, 100 and 50. Analyses were undertaken to determine why the results were so poor and how they could be improved.

Table 5. Heatmap of Model Configurations of Every Layer

Layer 1	Activation_Func / Nodes_N	tanh	relu	linear	softplus			
	120				?			
	100							
	50			?				
	18							
Layer 2	Activation_Func / Nodes_N	tanh	linear					
	100							
	50	?						
	18		?					
Layer 3	Activation_Func / Nodes_N	tanh	relu	linear	softmax			
	100	?						
	50			?				
	18							
Layer 4	Activation_Func	tanh	linear	softplus	selu	elu	linear	relu
	Output Dimension	50	50	50	50	50	50	50

The table above showed that different layer has slightly different preference of configuration.Though no good fiiting was acquired, the heatmap could also give insight of better choices of hyperparameters. Based on the heatmap, a new model has also been trained. The comfiguration from input layer to output layer is tanh+100, relu+150,relu+135, tanh+200, relu+120 and linear. The epoch size was set to be 6000, and patience to be 500. The R^2 got from this model fitting is 0.395, which is higher than best result got from all previous iterations. This

implies: (1). Grid search could help narrow down the model configurations used for further training. (2) More layers could possibly help enhance performance of the model.

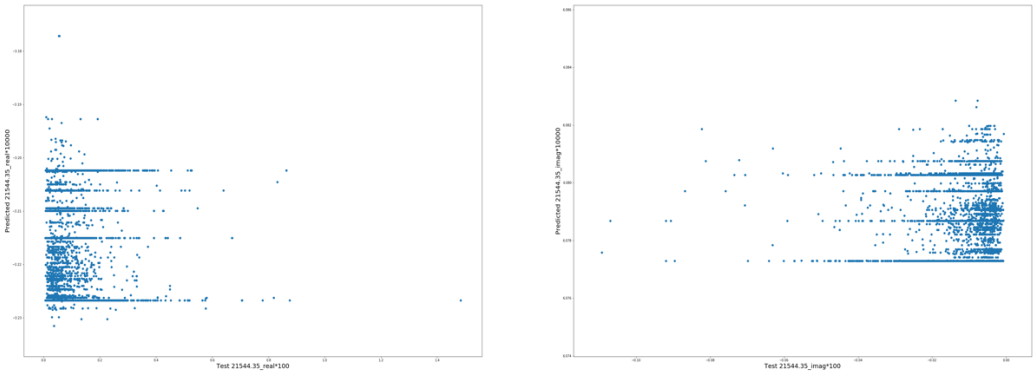

Figure 4. Test*100 vs Predicted*10000 of 21544 spectra.

.

3.2 ANALYSES AND POSSIBLE SOLUTIONS

3.2.1 DIFFERENT DATA PROCESSING METHOD

The problem could be that when the complex numbers have been split into two numbers, the correlation of the real parts and imaginary parts was reduced. The neural networks that were built were not complicated enough to learn the inner relationship between two parts of the neural network. To solve this problem, different ways of processing data could be utilized. For example, M. D. Murbach used principal component analysis (PCA) to concatenate two parts of complex numbers. The advantage of the PCA method is that it can preserve more correlated features of real and imaginary parts of the complex numbers – in other words, although still converted to a real number, the number maintains its integrity as a single number.

3.2.2 LARGER DATASIZE

The data is also heavily high dimensional, with 31 parameters but only 38,800 data points. The performance of the neural networks could have been constrained when large datasets were not available. Getting more data processed in the same way is likely to enhance the fitting

of the model.The epoch size of 300 is also relatively small. Thus, increasing the epoch size would provide more opportunities to minimize training errors to get better results.

3.2.3 ADDING BUILDING BLOCKS FOR COMPLEX-VALUED NEURAL NETWORK

Although the results have not yet shown all iterations for all combinations, the heatmaps suggest that every layer has different preferences with activation functions. Further, more nodes do not necessarily result in a better fit. Only the built-in activation functions of Keras have been applied, which can only handle real numbers. However, there are already many efforts to add building blocks for designing complex-valued neural networks. For example, modReLU has been proposed, which is differentiable with respect to the real and imaginary parts of a complex number.[18][19] Complex batch normalization and complex weight initialization [20] have also been suggested for complex values. In light of these suggestions, the fitting results could possibly be enhanced by leaving the complex frequencies as they are and incorporating advanced complex activation functions and complex algorithms. Keras allows the use of customized functions and flexibility in the utilization of different algorithms.[11] A P2D model could benefit from more hyperparameter options and the module could have broader uses as well.

3.2.4 DIFFERENT SEARCHING ALGORITHM

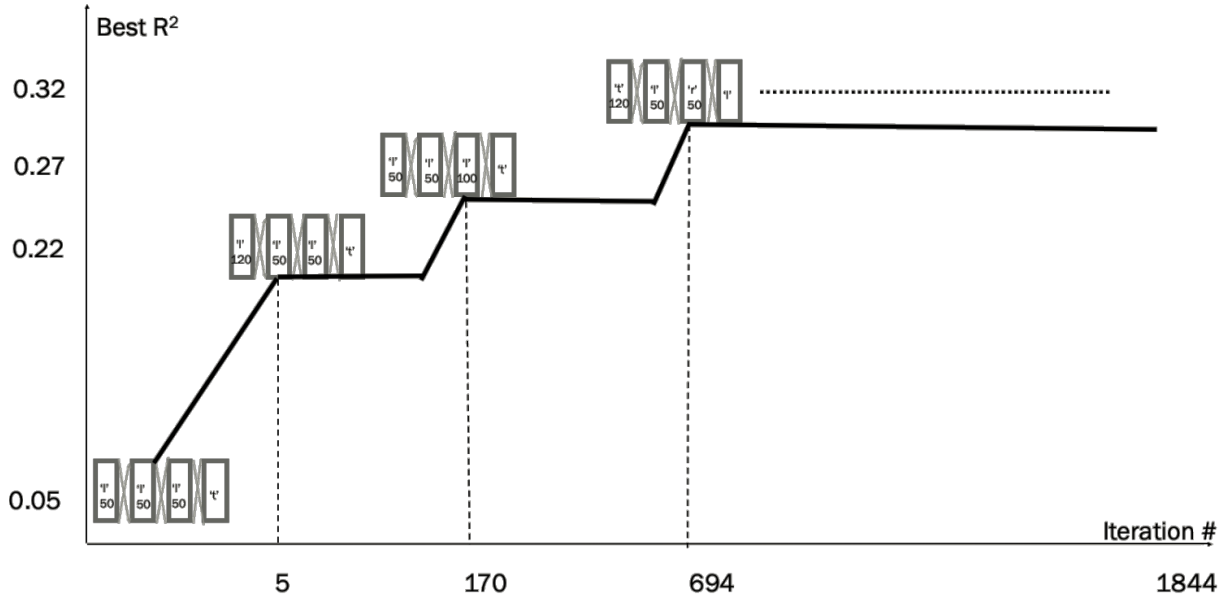

Figure 5. Iteration count vs Best R^2 during grid search process. The figure on the plot indicates the model architecture that gives better prediction at certain iteration. 'l' represents 'linear', 'r' represents 'relu' and 't' represents 'tanh'.

Figure 5 implies that the grid search method has quite low efficiency. The combination that gives the best R^2 does not change for numbers of iterations. Therefore, forward stepwise selection could be referred to while modifying the search algorithm, adding hyperparameters in layers one by one. Starting with a controlled model architecture in each layer configuration, only one hyperparameter combination that gives a single best improvement would be used to replace the controlled model before moving on to the next layer.[21]

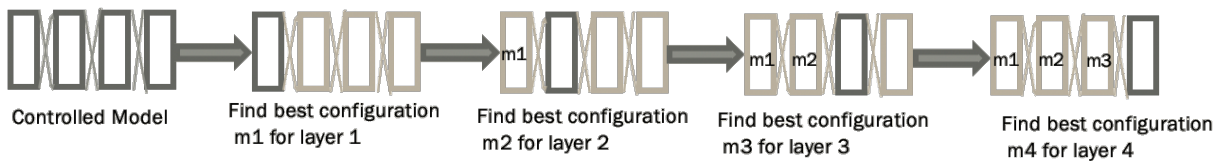

Figure 6. Possible forward stepwise searching method

3.2.5 USING DIFFERENT OPTIMIZAING ALGORITHM

Using an Adam optimizer could be another reason for the fitting getting bad results. Adam has performed well in deep-learning neural networks, but it could offer worse generalization compared to other optimizers, such as stochastic gradient descent. The neural networks that have been trained in this study mostly have only two hidden layers, in which the advantage of the Adam optimizer could be over-weighted by its limitations. The solution could be to train with other built-in optimizers in Keras. Many researchers have also implemented a revised optimization approach to tackle the issues of the Adam optimizer, which can be added into a module to improve its training process.

Chapter 4. CONCLUSION

In this study, A Python module has been developed to implement grid search for an MLP model to take place of a computation demanding nonlinear simulation model. It could act as a 'black box' which reduce the time of running through every mathematical equation. The module performs well with relatively simple, real-valued nonlinear synthetic data. However, the module failed to find a usable model that could replace P2D model which is high-dimensional and complex-valued. The modification to improve the functionality of the module could be adding advance building blocks for network, increasing dataset size or alternating learning algorithms.

BIBLIOGRAPHY

1. Oparaji, U., Sheu, R., Bankhead, M., Austin, J., & Patelli, E. (2017). Robust artificial neural network for reliability and sensitivity analyses of complex non-linear systems. *Neural Networks, 96*, 80-90. doi:10.1016/j.neunet.2017.09.003
2. Use of Simulation. (n.d.). Retrieved May 5, 2019, from https://www.anylogic.com/use-of-simulation/
3. Miron, M., Frangu, L., Ifrim, G., & Caraman, S. (2016). Modeling of a wastewater treatment process using neural networks. 2016 20th International Conference on System Theory, Control and Computing (ICSTCC). doi:10.1109/icstcc.2016.7790667.
4. Neto, A. H., & Fiorelli, F. A. (2008). Comparison between detailed model simulation and artificial neural network for forecasting building energy consumption. Energy and Buildings, 40(12), 2169-2176. doi:10.1016/j.enbuild.2008.06.013
5. Porto-Pazos, A. B., Veiguela, N., Mesejo, P., Navarrete, M., Alvarellos, A., Ibáñez, O., . . . Araque, A. (n.d.). Artificial Astrocytes Improve Neural Network Performance. Retrieved March 28, 2019, from https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0019109
6. Gerven, M. V., & Bohte, S. (n.d.). Retrieved from https://www.frontiersin.org/research-topics/4817/artificial-neural-networks-as-models-of-neural-information-processing#authors
7. Tu, J. V. (1996). Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes. Journal of Clinical Epidemiology, 49(11), 1225-1231. doi:10.1016/s0895-4356(96)00002-9
8. Begin, J. (2016). Response surface methodology and artificial neural network modeling of reactive red 33 decolorization by O3/UV in a bubble column reactor. Advances in Environmental Technology, 1, 33-44
9. Sen, B. A., & Menon, S. (2009). Turbulent premixed flame modeling using artificial neural networks based chemical kinetics. Proceedings of the Combustion Institute, 32(1), 1605-1611. doi:10.1016/j.proci.2008.05.077
10. Why use Keras? (n.d.). Retrieved from https://keras.io/why-use-keras/

11. A Gentle Introduction to Mini-Batch Gradient Descent and How to Configure Batch Size. (2019, May 14). Retrieved from https://machinelearningmastery.com/gentle-introduction-mini-batch-gradient-descent-configure-batch-size/
12. 1245284952193863. (2018, October 22). Adam - latest trends in deep learning optimization. Retrieved from https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c
13. Synced. (2019, March 07). ICLR 2019 | 'Fast as Adam & Good as SGD' - New Optimizer Has Both. Retrieved from https://medium.com/syncedreview/iclr-2019-fast-as-adam-good-as-sgd-new-optimizer-has-both-78e37e8f9a34
14. Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. AISTATS, 9
15. Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift
16. Murbach, M. D., & Schwartz, D. T. (2018). Analysis of Li-Ion Battery Electrochemical Impedance Spectroscopy Data: An Easy-to-Implement Approach for Physics-Based Parameter Estimation Using an Open-Source Tool. Journal of The Electrochemical Society, 165(2). doi:10.1149/2.1021802jes
17. Trabelsi, C., Pal, C. J. , *et al*(2018). DEEP COMPLEX NETWORKS. ICLR.
18. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Ournal of Machine Learning Research, 15, 1929-1958.
19. Cogswell, M., & Batra, D. (2016). Reducing Overfitting in Deep Networks by Decorrelating Representations. ICLR
20. Martin Arjovsky, Amar Shah, and Yoshua Bengio.(2015) Unitary evolution recurrent neural networks. *arXiv preprint arXiv:1511.06464*.
21. Hyak Node Hardware - Hyak User Documentation. (n.d.). Retrieved from https://wiki.cac.washington.edu/display/hyakusers/Hyak Node Hardware