Project report

Authors: *Tran Huong Giang Pham*
Master of Computer Science.
Emails: t.pham1@studenti.unipi.it
Subject code: 654AA, Academic Year: 2020/2021
Date: 04/01/2021
Type of project: **A**

Abstract

We implement a multilayer perceptron (MLP) and backpropagation learning algorithm with momentum and regularization technique. By employing grid search and k-fold validation schema for model selection, we decide to use a multilayer perceptron with 5 hidden layers for the CUP.

1 Introduction

Our aim is to implement and apply the implemented multilayer perceptron to solve various tasks such as binary classification and regression. Besides, we want to examine how different configurations of the hyper-parameters affect the performance of the network. To achieve this aim, we use the back-propagation learning algorithm (batch version) together with momentum and regularization technique with our MLP.
To decide a model for the CUP competition, we use grid search to find a model that gives the best validation result by k-fold validation schema on 80% of the giving training set with k = 5. The remaining 20 % of the giving training data is used as the internal test set to get the assessment result of our chosen model.

2 Method

To implement the multilayer perceptron, we use three classes: MLP, Unit and Activation as depicted in Figure 1. In which, MLP is the multilayer perceptron which contains multiple units and these units are connected to each other according to the architecture of the network. Inside each unit, there is an activation function belonging to one of these types: sigmoid, tanh, relu, softplus, leaky relu, gauss and linear. Units also contain the weights that connect the unit with other units of the next layer in the network. With these components, we can construct varied networks with different architectures such as various number of hidden layers or different activation function for each layer.

The learning flow chart is depicted in Figure 2. When one input comes, the input will be fed forward to the network in feed forward phase. After this process, we will get the output value o_{pj} and the derivative $f'(net_{pj})$ (with f is the activation function of this unit) for each unit j of the net with the input p.
In the next back propagation phase, the error signal is spread back to all the units of the network, in this phase, the value δ_{pj} of each unit j with the input p is calculated.
For the batch learning version, after getting all the needed value (o_{pj} and δ_{pj}) of the input p for all units of the net, we can compute the value Δw for the input p and then accumulate this value until we finish all the training input data for one epoch. Then we update the weights of each unit with the accumulated Δw.
To apply momentum and regularization, when we calculate the value Δw, we add the momentum term and the regularization term.
We choose to implement k-fold validation as the validation schema with the number k of folds can be chosen so that it will be suitable for dataset of different sizes.
Our program is implemented by C++ with standard libraries. To speed up the running time, we exploit multithreads in computing the mean square error of the training data and also in the accumulating process for batch learning.

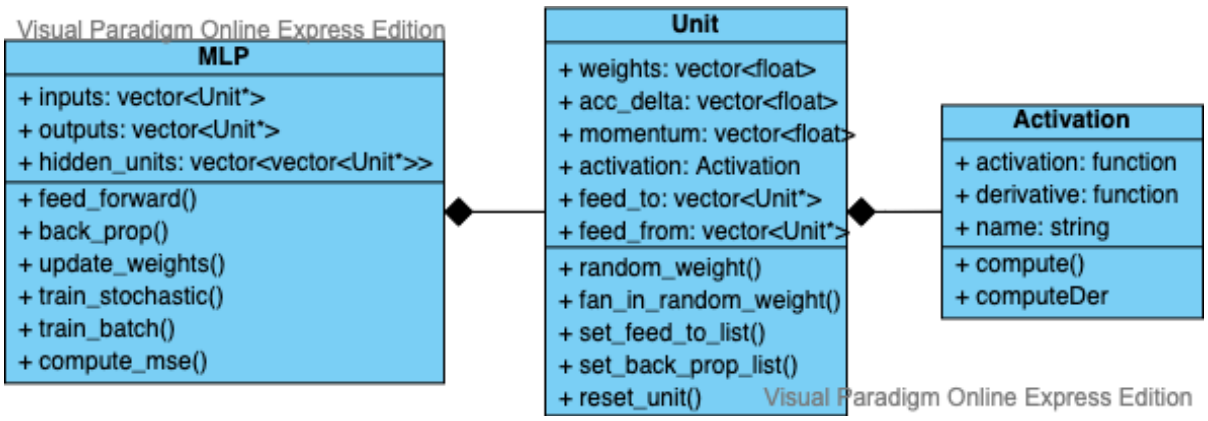


Figure 1: Class diagram of the program

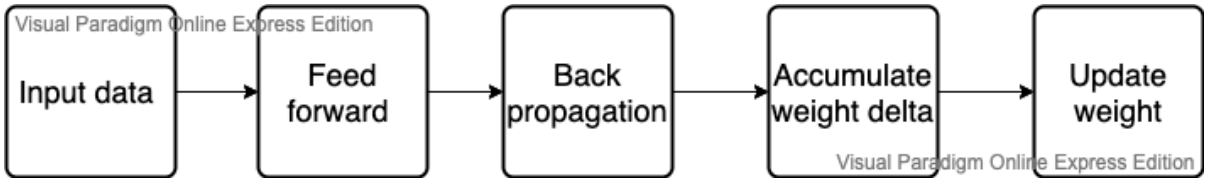


Figure 2: The learning flow of the MLP

3 Experiments

Firstly, we use Monk dataset as a benchmark to examine the correctness of our MLP model, then the CUP dataset will be used to evaluate the performance of the model.

3.1 Monk Results with MLP

For Monk dataset, we use one hot encoding for the input data.
The average result over 10 trials for 3 Monk datasets is report in Table 1.
Figure 3 to 6 depict the learning curve for each dataset.

Task	**#Units, eta, lambda**	**MSE (TR/TS)**	**Accuracy (TR/TS)(%)**
MONK1	2 hidden layers: 4, 2 units; eta = 0.7, alfa = 0.0007, lambda = 0.001, batch learning	0.039/0.049	0.95/0.95
MONK2	2 hidden layers: 6, 2 units; eta = 0.3, alfa = 0.0003, lambda = 9e-5, batch learning	0.058/0.07	0.95/0.94
MONK3	2 hidden layers: 8, 4 units; eta = 0.3, alfa = 0.0, batch learning	0.0049/0.065	0.998/0.91
MONK3+reg	2 hidden layers: 8, 4 units: eta = 0.07, alfa = 0.001, lambda = 0.0007, batch learning	0.15/0.14	0.932/0.97

Table 1: Average results of 3 Monk datasets over 10 trials with specified hyper parameters.

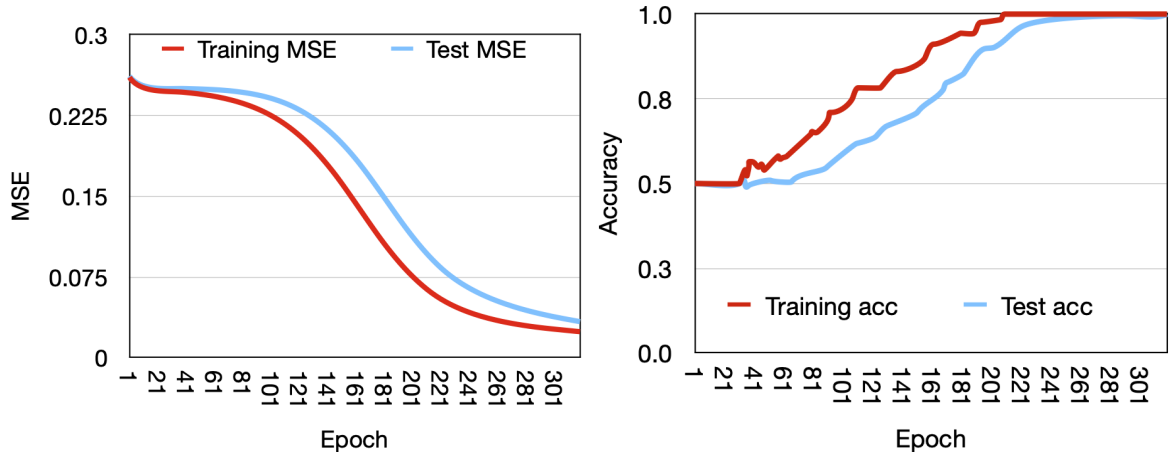
3.2 Cup Results

3.2.1 Validation schema and preprocessing

For assessing the model, we use a hold out internal test set contains 20(%) of all the data from the training file. The remaining data will be used in k-fold validation scheme with k = 5. Before splitting these sets, we shuffle the data. We will use the validation result from this method to choose the best model for the grid search in the grid search subsection.
Given the range of the target outputs are large, we decide to normalize the target to the range [0, 1] by the formula: $\frac{x-min}{max-min}$.

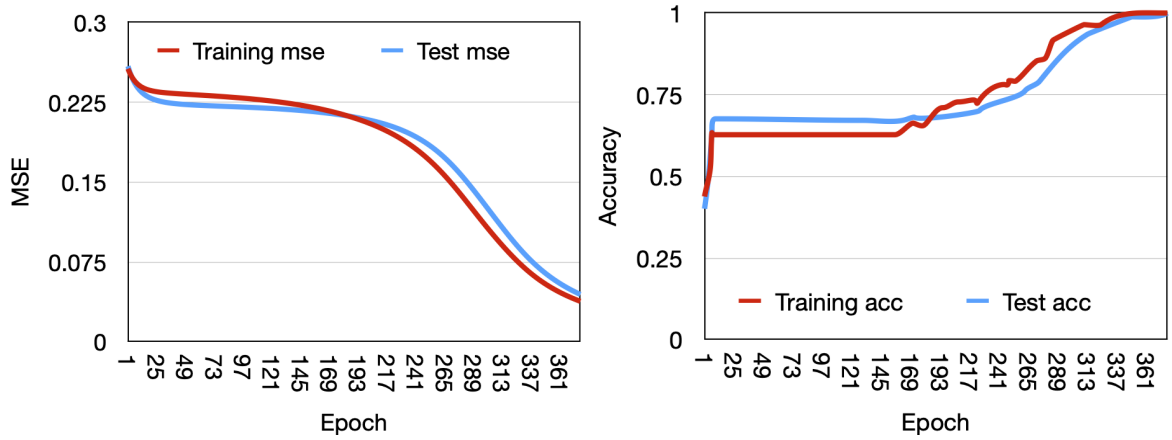
3.2.2 Screening phase

Figure 7 shows some interesting preliminary trials with the MLP models.



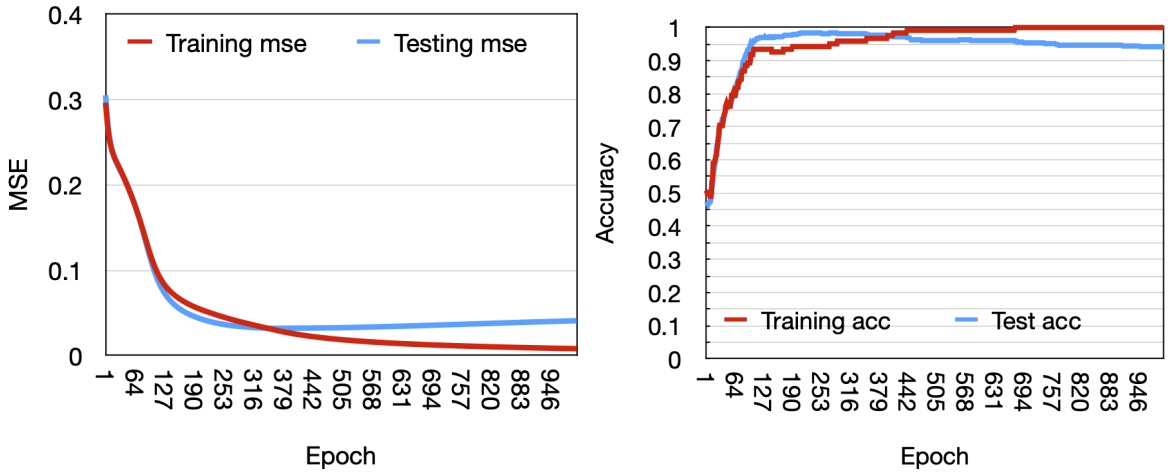
(a) (b)

Figure 3: Monk 1 dataset MSE (a) and accuracy (b) with model consists of 2 hidden layers with 4 and 2 units, with gaussian activation function for hidden layers and sigmoid activation for output units, eta = 0.7, alfa = 0.0007 and lambda = 0.001, batch learning.



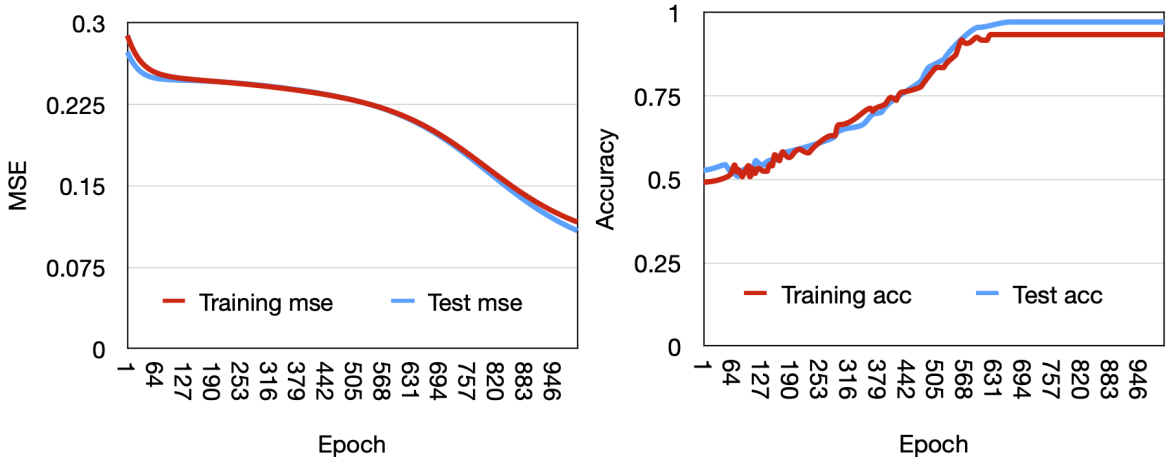
(a) (b)

Figure 4: Monk 2 dataset MSE (a) and accuracy (b) with model consists of 2 hidden layers with 6 and 2 units, with gaussian activation function for hidden layers and sigmoid activation for output units, eta = 0.3, alfa = 0.0003 and lambda = 9e-5, batch learning.



(a) (b)

Figure 5: Monk 3 dataset MSE (a) and accuracy (b) with model consists of 2 hidden layers with 8 and 4 units, with gaussian activation function for hidden layers and sigmoid activation for output units, eta = 0.3, alfa = 0, lambda = 0, batch learning.



(a) (b)

Figure 6: Monk 3 dataset MSE (a) and accuracy (b) with model consists of 2 hidden layers with 8 and 4 units, with gaussian activation function for hidden layers and sigmoid activation for output units, eta = 0.07, alfa = 0.001, lambda = 0.0007, batch learning.

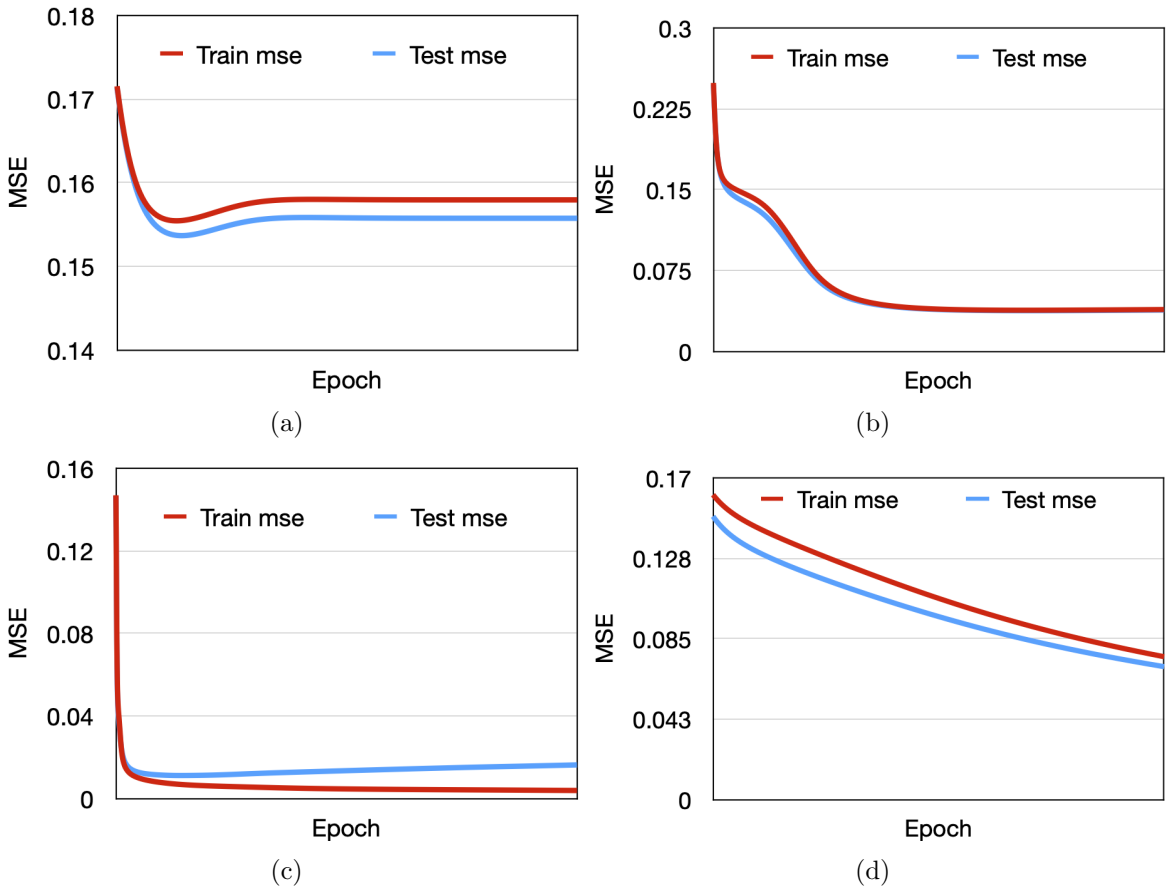


Figure 7: Screen demo of some MLP models (a) Underfitting caused by inappropriate hyper-parameters (eta is too small and lambda is too large) (b) A "good" learning curve with convergence and no overfitting (c) Overfitting caused by overtraining (d) Underfitting caused by undertraining.

3.2.3 Grid search

To choose the best neural network with the most appropriate combination of hyper-parameters, first, we perform the grid search with a large range of parameters e.g:

1. eta $\in$ {0.2, 0.4, 0.7}
2. alfa $\in$ {1e-5, 0.001, 0.01}
3. lambda $\in$ {1e-5, 0.0001, 0.001}

with a number of networks with different architecture and setup. After getting the first round results of the grid search as shown in Table 2 we pick 2 best models to go into the second grid search with the refine searching range around the best values we got from the first grid search.
The second grid search we perform the searching procedure with the 4^{th} and 6^{th} model from Table 2. For the 4^{th} model we search in the range:

1. eta $\in$ {016, 0.21, 0.25}
2. alfa $\in$ {0.00097, 0.00011, 0.0013}
3. lambda $\in$ {0.000095, 0.00011, 0.00013}.

The searching range for the 6^{th} model is:

1. eta $\in$ {0.37, 0.41, 0.45}
2. alfa $\in$ {0.008, 0.011, 0.015}
3. lambda $\in$ {0.000095, 0.00011, 0.00015}.

The result is shown in the Table 3. For more detail on the searching result of these two model, check the Table 5 in the section 5. Because there is no significant improve (the MEE reduces with the amount large than 0.1), we stop the grid search and pick the 4^{th} model with the refined parameters as shown in Table 3.
All of the grid searches here are run with 5000 epochs and with batch learning.

3.2.4 Decide a model for the CUP

Given the grid search result, we pick the MLP model with 5 hidden layers (16, 12, 8, 8, 4 units in each layer, respectively) with gaussian activation function for the hidden units and linear activation function for the output units as the "warrior" for the CUP. The hyper parameters are: eta = 0.21, alfa = 0.0011, lambda = 9.5e-5. This model give us the best validation result we can get among all of the other models (3.09664).

Idx	#hidden units, activation (hidden and output)	Best eta - alfa - lambda	Validation MEE	Train MSE
1	20-10-4, gauss, sigmoid	e=0.7 - a=0.01 - l=0.0001	3.2763	0.00780417
2	12-12-4, gauss, linear	e=0.2 - a=1e-05 - l=0.0001	3.15669	0.00804169
3	16-8-8-8-4, gauss, linear	e=0.2 - a=0.001 - l=0.0001	3.17622	0.00783045
4	16-12-8-8-4, gauss, linear	e=0.2 - a=0.001 - l=0.0001	3.11322	0.00789323
5	16-12-12-12-6, gauss, linear	e=0.2 - a=1e-05 - l=0.0001	3.14262	0.00801552
6	16-12-12-8-4, gauss, linear	e=0.4 - a=0.01 - l=0.0001	3.11865	0.00777457

Table 2: First grid search result with MLP: The combination of hyper parameters that gives the best validation result for each model.

Model id	Best eta - alfa - lambda	Validation MEE	Train MSE
4^{th}	e=0.21 - a=0.0011 - l=9.5e-05	3.09664	0.0078837
6^{th}	e=0.41 - a=0.015 - l=9.5e-05	3.13529	0.00779311

Table 3: Result of the second grid search with refine range for 2 best models from the first grid search.

3.2.5 Training procedure and final result

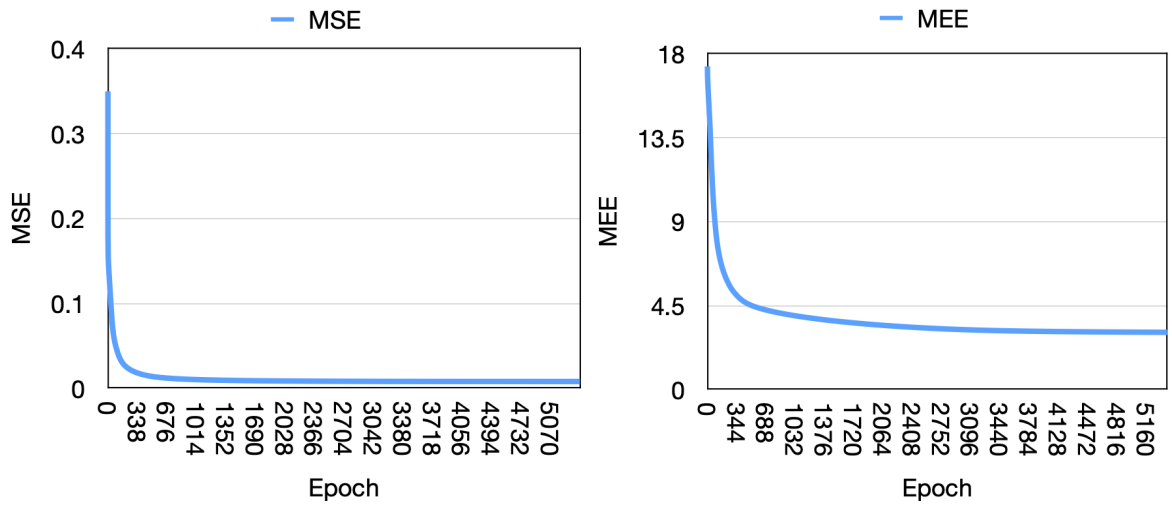
After figuring out the best model, we retrain the model with all the training data (80% of the given training set) and use early stopping technique to prevent overtraining. The training process will stop when the MEE of the validation set (for the early stopping technique) increases significantly (in term of number of continuously increasing epochs and also in term of the increasing amount). 15% of the training data is used for the early stopping.
The training curve is show in Figure 8. And all the information about the training procedure is report in the Table 4.

#Epoch	Final training MSE	Final validation MEE	Test result
5408	0.00762326	3.05394	**3.07844**

Table 4: Information about the training procedure of the final model.

3.2.6 Running time

To train 5408 epochs with 1037 training samples, in each epoch we also calculate the MEE with 183 validation samples, it takes 333815267 microsecond. That means on average, the training and validation process takes 61726 microsecond (about 0.06 second) for each epoch. The experiment is conducted on Macbook Air, version 2018 with Dual-core Intel Core i5, 1.6GHz.



(a) (b)

Figure 8: Learning curve of the final model: (a) MSE line of the training set (b) MEE line of the validation set used for early stopping

4 Conclusion

From this project, we have learnt how to implement a feed forward fully connected neural network with different configurations. Through the experiments, we also have an insight of how the hyper parameters affect the performance of the model. Moreover, the grid search and validation process also give us some experience with model selection and the usage of dataset.

BLIND TEST RESULTS:

Team name: Noob

Name of the result file: noob_ML-CUP20-TS.csv

5 Appendix

Idx	eta - alfa - lambda	MEE	eta - alfa - lambda	MEE
1	e=0.16-a=0.00097-l=9.5e-05	3.22857	e=0.37-a=0.008-l=9.5e-05	3.30068
2	e=0.16-a=0.00097-l=0.00011	3.22866	e=0.37-a=0.008-l=0.00011	3.19512
3	e=0.16-a=0.00097-l=0.00013	3.20171	e=0.37-a=0.008-l=0.00015	3.3749
4	e=0.16-a=0.0011-l=9.5e-05	3.20629	e=0.37-a=0.011-l=9.5e-05	3.25154
5	e=0.16-a=0.0011-l=0.00011	3.24076	e=0.37-a=0.011-l=0.00011	3.19558
6	e=0.16-a=0.0011-l=0.00013	3.22485	e=0.37-a=0.011-l=0.00015	3.27419
7	e=0.16-a=0.0013-l=9.5e-05	3.22471	e=0.37-a=0.015-l=9.5e-05	3.15192
8	e=0.16-a=0.0013-l=0.00011	3.23695	e=0.37-a=0.015-l=0.00011	3.3294
9	e=0.16-a=0.0013-l=0.00013	3.2354	e=0.37-a=0.015-l=0.00015	3.31565
10	e=0.21-a=0.00097-l=9.5e-05	3.19087	e=0.41-a=0.008-l=9.5e-05	3.23683
11	e=0.21-a=0.00097-l=0.00011	3.14882	e=0.41-a=0.008-l=0.00011	3.2108
12	e=0.21-a=0.00097-l=0.00013	3.12517	e=0.41-a=0.008-l=0.00015	3.42285
13	e=0.21-a=0.0011-l=9.5e-05	3.09664	e=0.41-a=0.011-l=9.5e-05	3.21649
14	e=0.21-a=0.0011-l=0.00011	3.15629	e=0.41-a=0.011-l=0.00011	3.28108
15	e=0.21-a=0.0011-l=0.00013	3.29651	e=0.41-a=0.011-l=0.00015	3.39181
16	e=0.21-a=0.0013-l=9.5e-05	3.17146	e=0.41-a=0.015-l=9.5e-05	3.13529
17	e=0.21-a=0.0013-l=0.00011	3.25202	e=0.41-a=0.015-l=0.00011	3.29397
18	e=0.21-a=0.0013-l=0.00013	3.14147	e=0.41-a=0.015-l=0.00015	3.31813
19	e=0.25-a=0.00097-l=9.5e-05	3.18136	e=0.45-a=0.008-l=9.5e-05	3.19806
20	e=0.25-a=0.00097-l=0.00011	3.24186	e=0.45-a=0.008-l=0.00011	3.21905
21	e=0.25-a=0.00097-l=0.00013	3.26715	e=0.45-a=0.008-l=0.00015	3.37409
22	e=0.25-a=0.0011-l=9.5e-05	3.19207	e=0.45-a=0.011-l=9.5e-05	3.36569
23	e=0.25-a=0.0011-l=0.00011	3.13987	e=0.45-a=0.011-l=0.00011	3.2531
24	e=0.25-a=0.0011-l=0.00013	3.22473	e=0.45-a=0.011-l=0.00015	3.24846
25	e=0.25-a=0.0013-l=9.5e-05	3.15639	e=0.45-a=0.015-l=9.5e-05	3.21131
26	e=0.25-a=0.0013-l=0.00011	3.14944	e=0.45-a=0.015-l=0.00011	3.15227
27	e=0.25-a=0.0013-l=0.00013	3.444427	e=0.45-a=0.015-l=0.00015	3.24056

Table 5: Grid search result of the 4^{th} model (two columns on the left) and the 6^{th} model (two columns on the right) with refine range (we report only the validation result here).