# Neural Networks for Prediction of functional data

Weiru Han[1*] Lu Lu[1†] Jiangfeng Zhou[2‡]

**Abstract**
Neural networks are popular for classification and prediction involving a large number of inputs and a small number of output variables. However, they have not been broadly used for predicting functional output. Motivated from a physical science study which involves predicting the spectrum curve from metamaterial optical device with different geometrical parameter settings, we developed neural network models that utilized the multi-layer perceptron for predicting the functional output. To optimize the network architecture, first a screening design was used to explore the impact of model hyperparameters such as the numbers of layers and nodes and the choices of activation functions. Then a two-phased genetic algorithm called NSGA-II [Deb et al., 2002] was developed to search the optimal structures to simultaneously minimize the training and prediction errors as well as the model complexity. The first phase used a discrete non-dominated sorting genetic algorithm to seek top-ranking combinations of the number of nodes, activation functions and optimization methods. The second phases used a continuous search focusing on optimizing the number of nodes. Finally, desirability functions were used to further select the most robust models from pareto front. The proposed method will be illustrated with an optical device optimization example.

**Key Words:** multi-layer perceptron, functional principal component analysis, factorial design experiment, evolutionary algorithm, multiple objective optimizations, desirability functions

## 1. Introduction

The term "functional data analysis" was first presented by [Ramsay, 1982], but the history of the field is much older and dates back to [Grenander, 1950, Rao, 1958]. Typical functional data consists of a sample of observations that are also regarded as a set of curves or real-valued functions $X_1(t), X_2(t), \cdots$ on a compact interval $\tau = [0, t]$ [Wang et al., 2016]. Much work dealing with functional data usually aims at smoothing curves, clustering and classification and so forth. A growing area concentrates on functional regression, where the relationship between functional covariates and vector of responses or functional responses are of main interest. However, few studies have focused on predicting functional data using a vector of predictors. One of the main reasons is that it is invariably easier to extract features from information with higher dimension compared to those with lower dimension.

In our experiment, there are 4 types of geometric parameters of discs and device that are related to the shape of reflection and transmission curves, which are dis-alignment distance in X-axis denoted by $a$, dis-alignment distance in y-axis denoted by $b$, thickness of the device denoted by $t_{BCB}$ and radius of the disc denoted by $r$ where $r = d_{GDA}/2$. An example of discs and device is illustrated in figure 1 as well as these 4 geometric features.

While the overall goal of this project is to predict reflection and transmission curves as functional data using 4 geometric parameters as scalar variables, several issues are needed to be addressed prior to that. Indeed, we mainly focus on two problems:

1. How to find a group of promising models to predict function data regardless of dimension using limited number of scalar variables?

[*]Department of Mathematics and Statistics, University of South Florida, weiruhan@usf.edu
[†]Department of Mathematics and Statistics, University of South Florida, weiruhan@usf.edu
[‡]Department of Physics, University of South Florida, jiangfengz@usf.edu

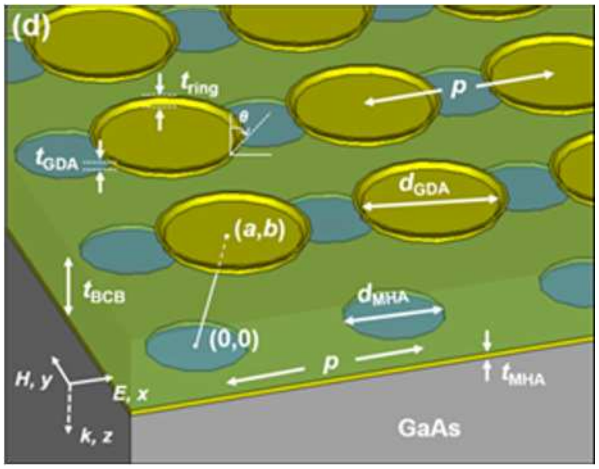


**Figure 1**: An example of a photonic device

2. How to form a systematic way in selecting the near-optimal models?

Datasets were generated from a simulation of a 500-run space filling design which is shown in figure 2. Inside the plot, $t$ represents $t_{BCB}$ as the thickness of the device and $d$ represents $d_{GDA}$ in figure 1 as the diameter of the disc. In our research, we used radius of the disc $r = d_{GDA}/2$ as one of 4 predictor variables to train the model.

While reflection and transmission curves could not fit well with polynomial regression models, we adopted classic neural networks which belong to the multi-layer perceptron with backpropagation to predict functional data in that neural networks are advantageous in capturing nonlinear response surface. To address the first issue, we utilized a common dimension reduction method called functional principal component analysis (short for FPCA) to reduce the dimension of those curves. The benefit of this approach is that it can guarantee that functional data can be represented by a relatively small number of coordinates while still preserving a majority of information of the data, which is highly useful especially when there are too many sampling points of functional data After dimension reduction, these coordinates can be fed into multi-layer perceptron as response to train models. After carrying out the training process, predicted coordinates are projected back into the original space using basis expansion to compare with original curves. Predicted performance of curves can be unfavorable either because the model is too simple or too complicated. Therefore, models that could be considered as promising if they can nicely balance the trade-off between model complexity and predicted accuracy. To address the second problem, notice that different model hyperparameters have impact on predicted performance, we applied an evolutionary method called a fast and elitist non-dominated sorting genetic algorithm (short for NSGA-II) to find a group of near-optimal models. However, typical genetic algorithm may require a large search space for obtaining the best models which is of huge time and cost. Therefore, we conducted a screening design experiment prior to employing NSGA-II to narrow down the search space first. At the end, desirability functions were given in order to select a small group of best solutions from pareto front.

The structure of the paper is formulated as follows. Section 2 introduce the basic idea behind functional principal component analysis and how it relates to the general principal component analysis for multivariate data. Section 3 illustrates how we design a screening experiment as well as their corresponding results. Section 4 demonstrates the application of NSGA-II algorithm and how it is imposed considering different targets in 2 phases. Section 5 presents the use of desirability functions in selecting a small subset of optimal models from pareto front.

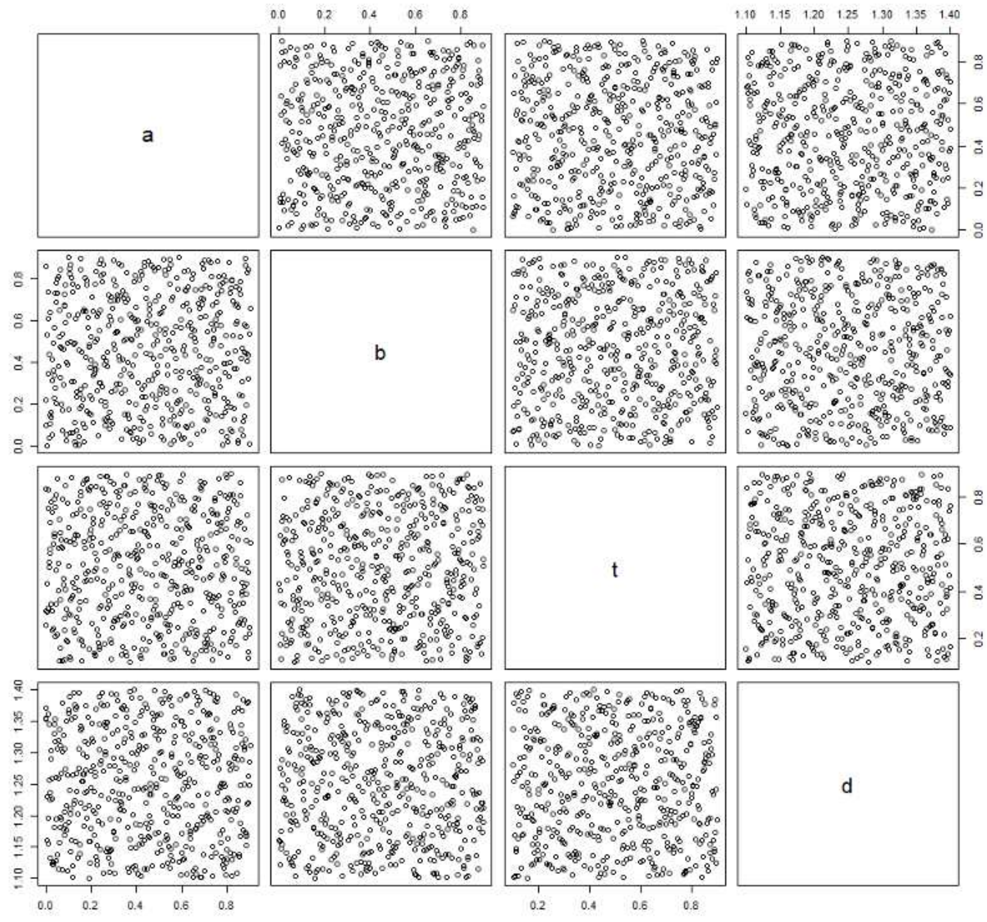


**Figure 2**: Input space for 4 geometric parameters

## 2. Functional Principal Component Analysis

Suppose we have $N$ curves denoted by

$$x_1(t), x_2(t), \cdots, x_N(t) \tag{1}$$
$$t \in \tau = [0, T] \tag{2}$$

where $\tau$ is a compact interval, thus predicting curves is actually predicting functions which is unrealistic for general methods. To reduce the dimension of functional data, [Karhunen, 1946] independently discovered the FPCA expansion

$$x_i(t) = \mu(t) + \sum_{k=1}^{\infty} y_{ik}\xi_k(t) \tag{3}$$
$$y_{ik} = \int_{\tau} (x_i(t) - \mu(t))\xi_k(t)dt \tag{4}$$

where $y_{ik}$ are the functional principal component scores (short for FPCs) of the curve $x_i(\cdot)$, $\xi_k(t)$ is the weight function measured at point $t$, and $\mu(t)$ is the mean curve (function) of $\mathbf{x}_1, \cdots, \mathbf{x}_N$ measured at point $t$. It is evident to show that the first $k$ terms of above equation for large enough $k$ could provide a good approximation to $x_i(t)$. In fact, as a counterpart of principal component analysis for multivariate data, FPCA also shares the same properties where $y_{ik}$ and $\xi_k$ are obtained by eigen-decomposition for covariance function denoted by

$$V(s,t) = \frac{1}{N}\sum_{i}^{N} x_i(s)x_i(t) \tag{5}$$

To obtain eigen-components, we need to solve the equation

$$\int_{\tau} V(s,t)\xi_(t) = \lambda\xi(s) \tag{6}$$

where $\lambda$ represents eigenvalue. The challenge here is how to find a method to compute the integral. In general, there are 3 approaches that can be utilized to approximate the integral, including discretizing functions into $n$ equally spaced values [Rao, 1958], using pre-specified basis such as fourier basis to express those functions [Ramsay and Silverman, 1997], or using numerical integral [Stoer and Bulirsch, 2002].

For our study, we choose to discretize functional data into $n$ equally spaced values and convert our problem into multivariate case. After discretization, we get an $n \times N$ matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N]$, of which each function $x_i(\cdot)$ is replaced by a vector $\mathbf{x}_i$ consists of $n$ values $\mathbf{x}_i(t_1), \mathbf{x}_i(t_2), \cdots, \mathbf{x}_i(t_n)$ where $t_1, t_2, \cdots, t_n$ are $n$ equally spaced points from the compact interval $\tau$. We subtract $\mathbf{X}$ from its row mean to obtain a $n \times N$ matrix $\mathbf{X}^*$ ensuring that each row has zero mean; thus, we can calculate covariance matrix $\Sigma$ as

$$\Sigma = \frac{1}{N}\mathbf{X}^*(\mathbf{X}^*)^T \tag{7}$$

We then obtain the eigenvalues $\lambda_i$ and eigenvectors $\psi_i$ by calculating the equation

$$\Sigma\psi_i = \lambda_i\psi_i \; w.r.t \tag{8}$$

$$\|\psi_i\|_2^2 = 1, \; \psi_i^T\psi_j = 0, \; \forall i \neq j \tag{9}$$

We rank eigenvalues in a decreasing order to get $\lambda_1, \lambda_2, \cdots$ as well as the associated eigenvectors $\psi_1, \psi_2, \cdots$. We select $k$ terms of eigenvectors $\psi_1, \cdots, \psi_k$ either manually or by how much percentage these eigen-components explain to the total variation of the data which can be calculated as $\frac{\sum_{i=1}^{k}\lambda_i}{\sum_{i=1}\lambda_i}$. At last, $k$ terms of eigenvectors constitute an eigen-basis projecting $\mathbf{X}$ to $\mathbf{Y} = \mathbf{X}^T\psi$, from a higher $n$-dimensional space to a lower $k$-dimensional space. A detailed process of PCA is summarized in algorithm 1

**Algorithm 1:** Principal component analysis

```
  Input: observation matrix $\mathbf{X} = \begin{bmatrix}\mathbf{x}_1 & \cdots & \mathbf{x}_N\end{bmatrix}_{n\times N}$
         threshold $\sigma \in (0,1)$
  Output: Principal component weight matrix $\Psi = \begin{bmatrix}\psi_1 & \cdots & \psi_k\end{bmatrix}$
          Principal component score matrix $\mathbf{Y} = \begin{bmatrix}\mathbf{y}_1 & \cdots & \mathbf{y}_k\end{bmatrix}_{N\times k}$
1 For each row of observation matrix $\mathbf{X}$, subtract them from mean value of the row
    to get a $n \times N$ matrix $\mathbf{X}^*$ ensuring each row has zero mean
2 Calculate covariance matrix $\Sigma = \frac{1}{N}\mathbf{X}^*(\mathbf{X}^*)^T$
3 Compute eigenvalues $\lambda$ and eigenvectors $\psi$ such that $\Sigma\psi = \lambda\psi$
4 Rank eigenvalues in a decreasing order to get $\lambda_1, \lambda_2, \cdots$
5 Obtain $\psi_i$ corresponding to $\lambda_i$
6 Choose $k$ manually or select $\psi_1, \cdots, \psi_k$ such that
      $\frac{\sum_i^{k-1}\lambda_i}{\sum_i\lambda_i} < \sigma \;\&\; \frac{\sum_i^{k}\lambda_i}{\sum_i\lambda_i} \geq \sigma$
7 Output $\mathbf{Y} = (\mathbf{X})^T\Psi$
```

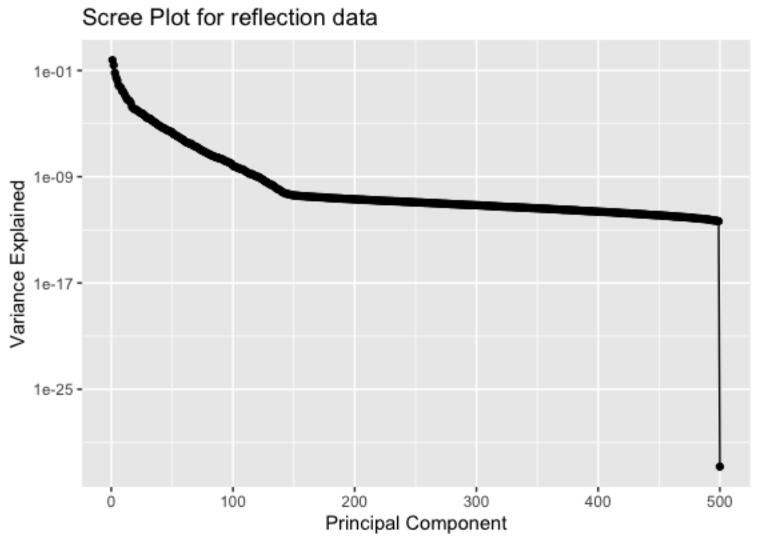


**Figure 3**: Scree plot for reflection data

The rank of $\mathbf{X}^*$ is at most $N-1$, thus for an $n \times n$ covariance matrix $\Sigma$, we can obtain at most min$(n, N-1)$ nonzero eigenvalues $\lambda_i$. In our case, we have 500 curves each for the reflection and transmission data, with 1001 equally spaced frequency points from interval [40,60], thus we can obtain at most 499 nonzero eigenvalues. Figure 3 is a scree plot with a log scale in y-axis showing the variance explained after carrying out principal component analysis to reflection data. In effect, we find out that among 500 principal components, the first 6 principal components cover more than $99\%$ of the variance of the data showing that we can use only 6 principal components as the basis to reduce the dimension of the data without losing much information. However, since the main goal is to find a promising model with a higher accuracy, we select all 500 principal components as the basis so that each observation is represented by 500 principal component scores in the later experiments.

## 3. Screening design experiment

To narrow down the search space, we design a screening experiment to select a group of candidate hyperparameters of neural networks, which is summarized in table 1. For this experiment, network models have 3 options for choosing number of hidden layers including 2,4,6 hidden layers. After fixing the number of layers, number of hidden units are the same through all hidden layers and contains 5 possible values which are adopted from default values in general convolutional neural networks for convenience; **Act1** represents the activation function connecting from the input layer to the first half hidden layer, and **Act2** represents the activation function connecting from the first half hidden layer to the last hidden layer; search space is the same for both activation functions including 11 common function types. Figure 4 illustrates a network with 4 hidden layers whose number of nodes for all hidden layers is 512; **Act1** is "**relu**" and **Act2** is "exponential". For regression problems, typical choice of activation function for output layer is linear function, thus we choose not to add any activation function between the last hidden layer and output layer to guarantee a linear combination.

For different model architectures, we fix the training epochs as 100 so as to observe which model will converge the best in a limited update time. Functional data is discretized by 1001 equally distributed frequency points and reduced to 500 dimensions by FPCA as the training output. Optimizer is chosen as "**adam**" as it has been proved to be the

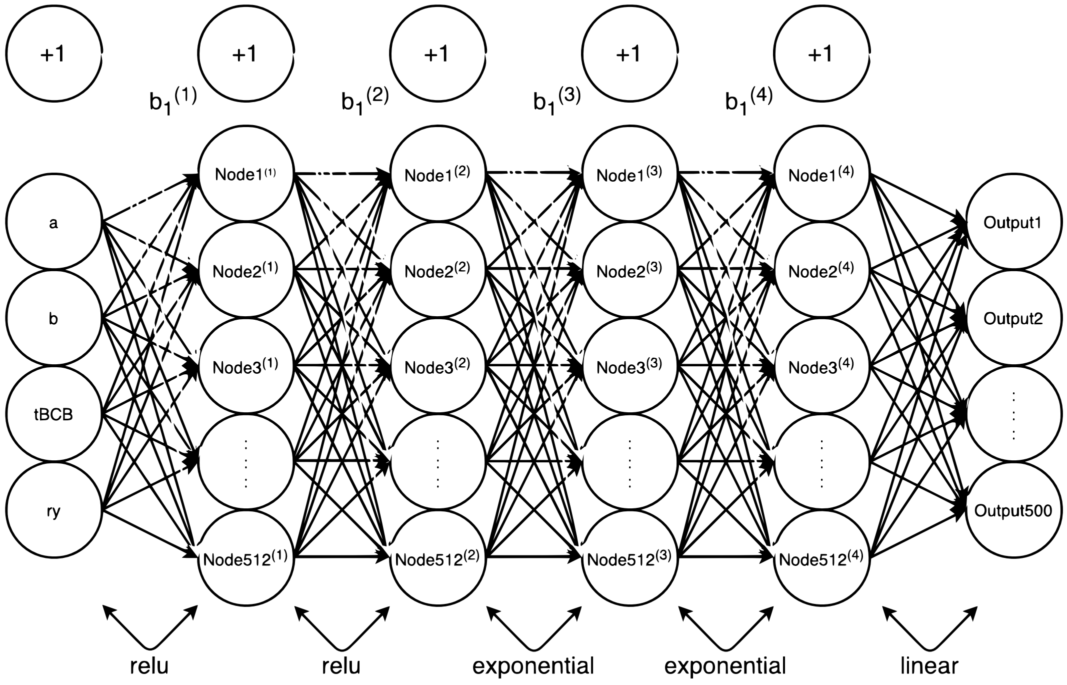


**Figure 4**: Multi-layer perceptron with 4 hidden layers

most effective optimizer in terms of the training cost in many scenarios, one can also refer to [Ruder, 2016] where it benchmarked multiple optimizers on the multi-layer perceptron and convolutional neural networks, respectively. Data is divided into 5 folds in order to calculate 5 folds cross validation errors. To select a group of superior model structures, the criteria is to investigate the main and interaction effect measured by cross validation errors. Due to the fact that prediction of network is the principal component scores with the lower dimension, we need to project scores using eigen-basis to the original space to obtain the fitted data; thus, errors are computed by comparing the fitted data with those discretized curves. Suppose we denote N discretized curves with $n$ equally spaced points as $\mathbf{x}_i(t_j)$ and fitted curves as $\hat{\mathbf{x}}_i(t_j), i = 1, 2, \cdots, N, j = 1, 2, \cdots, n$, mean square error would be

$$MSE = \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{x}_i - \hat{\mathbf{x}}_\mathbf{i}\|_2^2 \tag{10}$$

| Hyperparameters | Levels |
|---|---|
| # Layers | 2,4,6 |
| # Nodes | 32,64,128,256,512 |
| Act1 and Act2 | selu, relu, elu, hard_sigmoid, exp, tanh, softplus, linear |

**Table 1**: Selection of hyperparameters

Hence the cross-validation error would be the average of MSE for 5 folds, calculated as

$$CVerror = \frac{1}{5}\sum_{p=1}^{5}\frac{1}{N_{f_p}}\sum_{r\in f_p}\|\mathbf{x}_r - \hat{\mathbf{x}}_\mathbf{r}\|_2^2 \tag{11}$$

where $p$ is the fold index, $f_p$ is the observation index set and $N_{f_p}$ represents the number of observations in $p's$ fold.

For each level of layers number, there are 605 factor combinations. Through all experiments, only networks with 2 hidden layers can export valid results with errors between 0 and 1. For networks with 4 layers, 5 combinations fail for reflection data and 6 combinations fail for transmission data, and for networks with 6 layers, 23 models each for both type of data fail.

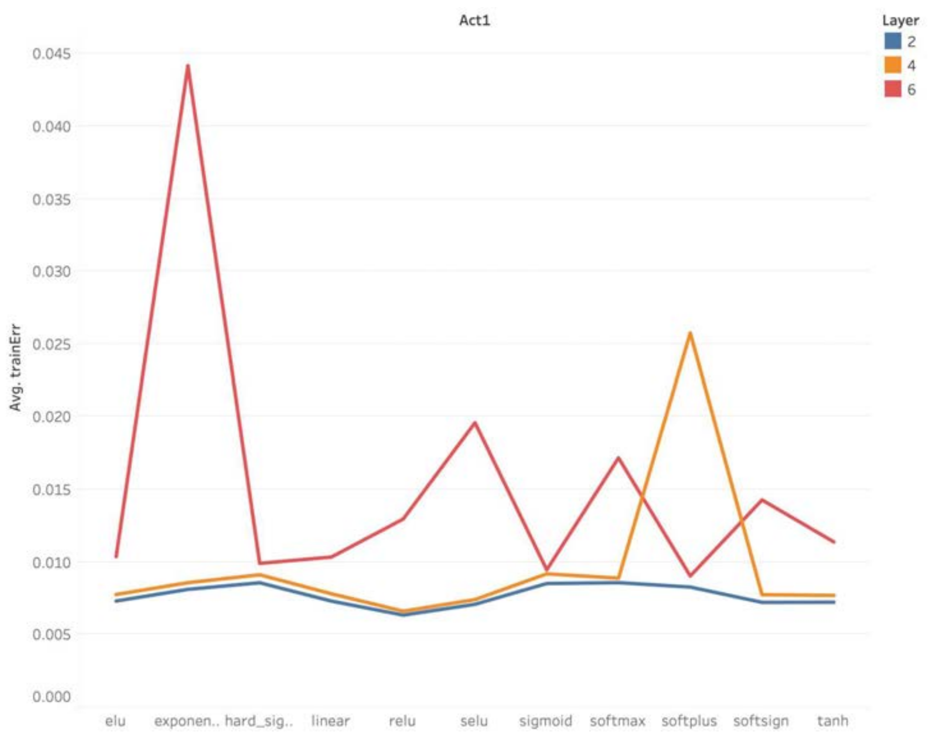
The main reason is evident either in that the exploding gradient or vanishing gradient. We notice that all failed models contain "**exponential**" as activation function whose gradient has no limitation; then by using backpropagation algorithm, gradient may be multiplied to an extremely high value so as to overflow and obtain **NA** values. Multiplicative gradient values with respective to functions such as "**sigmoid**" or "tanh" are apt to be zero especially for deep neural networks, which will cease updating gradient, that is, the gradient vanishing problem. All failed cases manifest activation function 1 and 2 are either all "**exponential** or a combination of "**exponential**" and functions like "sigmoid" that are not capable of resolving gradient exploding problem.

Through all valid cases, we average their cross-validation errors for single factor or two factors combination. Most marginal effects are less than 0.01, nevertheless there are marginal effects larger than 0.1 regardless of the option for **Act1** when **Act2** is "**SoftMax**". We then remove "**SoftMax**" as the option in **Act2** considering the fact that those large values may increase the bias in observing the marginal effects for **Act1**.

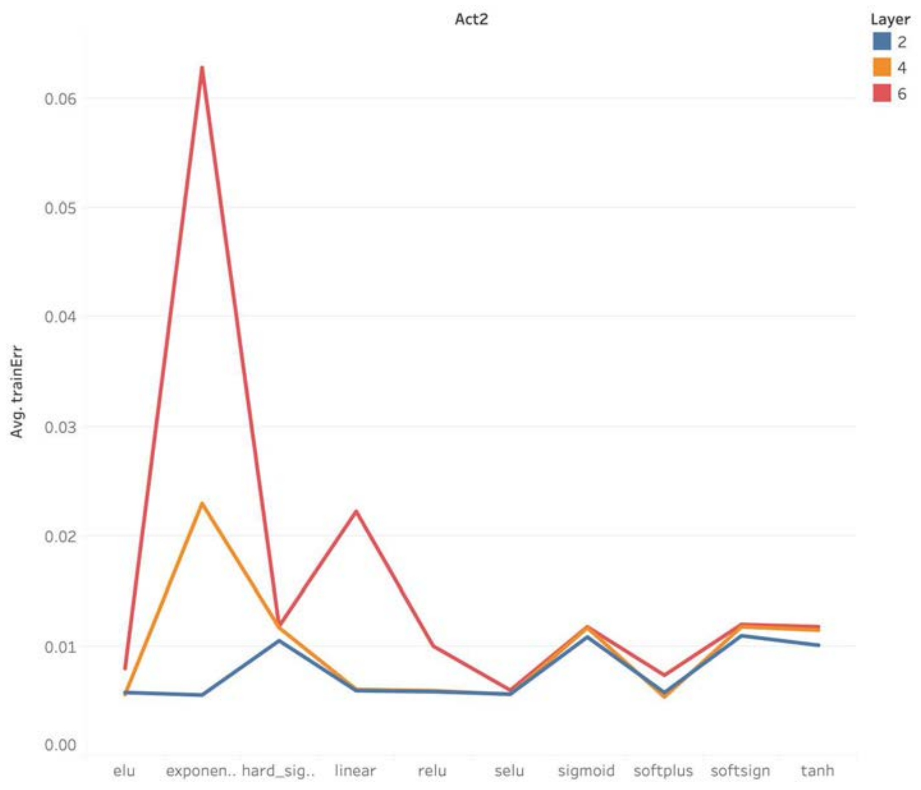
Figure 5a and Figure 5b depict marginal effects of **Act1** and **Act2** in terms of different layers. While blue, orange and red lines represent models with 2,4 and 6 hidden layers respectively, we can easily observe the instability exists in networks with 6 hidden layers while models with 2 or 4 hidden layers are more stable; Both graphs show that for most activation functions, 2-layer model has no difference in predicted performance compared to 4-layer model and can be even better especially when activation function is "**softplus**" and "**exponential**" for **Act1** and **Act2**, respectively.

To lower down the randomness of the training, we repeat aforementioned procedures by shuffling the data to have different splits. We collect those consistent results, and find out that for reflection data:

- Models with 2 hidden layers have the better and more stable mean performance than 4 and 6 hidden layers
- When number of hidden layers is 2 and **Act1 =**'**relu**','**elu**,'**selu**','**linear**','**softsign**','**tanh**', marginal effect of **Act1** is always better than other 5 activation functions
- When number of hidden layers is 2 and **Act2** = '**elu**','**selu**','**softplus**','**exponential**', marginal effect of **Act2** is always better than other 7 activation functions



(a) marginal effect of act1 for reflection



(b) marginal effect of act2 for reflection

**Figure 5**: marginal effect of activation function 1 and activation function 2 with respect to different layers and data

## 4. NSGA-II algorithm

In design of experiments usually multiple criterions are to be considered, all of which cannot be optimized at the same time; that is, any criterion must be optimized at the cost of diminishing at least one of other criterions. Instead of finding solutions that can perform the best only in one aspect, we aim to find pareto optimal set that can best balance the trade-off between different criterions. Suppose we denote solutions as $S_i, i = 1, 2, \cdots$, and we relate each solution to a criterion vector $F(S_i)$ comprised of $C$ objective functions $f_1(S_i), f_2(S_i), \cdots, f_C(S_i)$, we say $S_i$ pareto dominates $S_j$ or $S_j$ is dominated by $S_i$ if

$$f_k(S_i) \geq f_k(S_j) \; for \; k = 1, 2, \cdots, C \tag{12}$$

$$\text{at least one } k \text{ from } 1, 2, \cdots, C \text{ such that} \tag{13}$$

$$f_k(S_i) > f_k(S_j) \tag{14}$$

As such, pareto optimal set is the collection of solutions which dominates all other solutions that are not in the pareto optimal set, and solutions in the pareto optimal set cannot dominate each other. The collection of the corresponding criterion vectors for pareto optimal set is called pareto front. To optimize multiple objectives and find those pareto optimal solutions a general method is the genetic algorithms. Usually, pareto optimal set contains an ocean of possible solutions, thus two different strategies can be applied to further select the smaller group of optimal solutions from pareto front. The first strategy usually is applied prior to the optimization process which is to scalarize vector of multiple objective functions into a scalar function. Methods include weighted linear combination of objective functions, distance functions, desirability functions, etc. However, selected optimal solutions are highly sensitive to different weighting schemes using these methods, and oftentimes require multiple GA search for different weights, which could not apparently be the best strategy considering time and cost. The second strategy is to apply multiple criteria to the pareto front after optimization [Chapman et al., 2018]; methods include additive desirability functions, utopia point method and so on. Details of the second strategy and corresponding results are illustrated in the section 5.

While our final objective is to find a promising model to predict the functional data, it is also noteworthy to lessen the model complexity. Thus, how to balance the trade-off between model complexity and prediction performance is vital to the research. Here we use number of trainable parameters to signify the model complexity, which is directly related to the number of hidden units in different layers. For a network with 2 hidden layers, the number of trainable parameters is calculated as

$$\begin{gathered}\#\text{trainable parameters} = (\text{input}_{dim} + 1) * \text{first hidden layer}_{dim} + \\ (\text{first hidden layer}_{dim} + 1) * \text{second hidden layer}_{dim} + \\ (\text{second hidden layer}_{dim} + 1) * \text{output layer}_{dim}\end{gathered}$$

In our research, we focus on minimizing training and predicted errors as well as minimizing the model complexity. However, it is more likely that as model complexity decreases, errors increase. Depending on the dimension for input layer and output layer, number of units for different hidden layers should be adjusted within a reasonable interval. To optimize our problem, we employ a fast and elitist non-dominated sorting algorithm (short for NSGA-II) [Lu et al., 2011] to find the optimal set.

After carrying out screening design, a group of candidate solutions is chosen as a new search space to explore the near-optimal networks. From screening experiment, we observe that networks with 2 hidden layers can always output efficient results regardless of how we

| Unit 1 | 64,128,192,256,320,384,448,512 |
|---|---|
| Unit 2 | 64,128,192,256,320,384,448,512 |
| Act1 | Relu, elu, selu, linear, softsign, tanh |
| Act2 | elu, selu, softplus, exponential |
| Optimizer | adadelta, adagrad. adamax, adam |

**Table 2**: Search space of model for reflection data in Phase I

configure the model, which means as layer stacks, there is more chance that the model predicts worse than being more accurate. For this reason, we select networks with 2 hidden layers as a fixed structure in the subsequent experiment. To verify whether **adam** is the best optimizer, we expand the search space of optimizer with other 3 different optimizer functions. Our algorithm contains 2 phases, of which the first aims to find the best configuration of activation function and optimizer by doing a quick search, and the second aims to further explore the relationship between hidden unit number and model performance while fixing the structure of pareto optimal solutions from the last generation of phase I.

We encode network architecture into a chromosome consists of 5 genes which is shown in Figure 6. Table 2 represents the search space of models for predicting reflection data in the phase I; unit 1 and unit 2 represent the number of hidden units for the first and the second hidden layers, respectively. For simplicity, we choose integer to represent activation function and optimizer types but keep unit number as the true number. For example, in finding the optimal models for reflection data, a chromosome labeled as **64-128-1-4-4** represents a 2-layer model, whose first hidden layer contains 64 units, and second hidden layer contains 128 units; the activation function connecting between input layer and first hidden layer is **relu**, and the activation function connecting between the first hidden layer and the second hidden layer is exponential function; the optimizer type is **adam**.

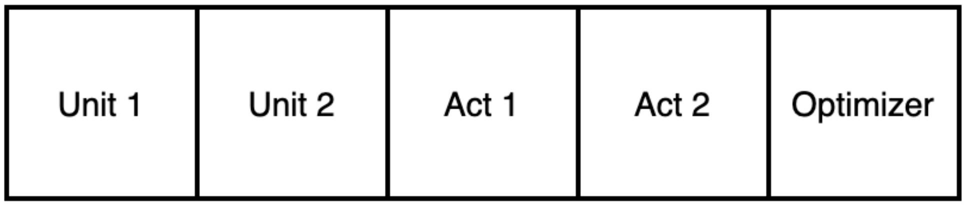


**Figure 6**: chromosome-alike solution structure

### 4.1 Phase I

In this section, we propose a quick search for a candidate set of optimal models by selecting a small number of values for the unit number. We can stop the searching process after a fixed generations, a fixed processing time, or improvement of objective functions reach a threshold [Chapman et al., 2018]. In our experiment, we manually set up a fixed generations to terminate the search process. We first introduce several terminologies and notation

| | Unit 1 | Unit 2 | Act1 | Act2 | Optimizer |
|---|---|---|---|---|---|
| Parent 1 | 32 | 64 | 1 | 2 | 1 |
| Parent 2 | 64 | 128 | 2 | 4 | 4 |
| | $\downarrow$ | $\downarrow$ | $\downarrow$ | $\downarrow$ | $\downarrow$ |
| Offspring | 32 | 128 | 1 | 4 | 1 |

**Table 3**: Cross over by randomly picking up genes from both sides; for offspring, gene 1,3,5 are from parent 1,gene 2,4 are from parent 2

| | Unit 1 | Unit 2 | Act1 | Act2 | Optimizer |
|---|---|---|---|---|---|
| Parent 1 | 64 | 128 | 2 | 4 | 4 |
| | $\downarrow$ | $\downarrow$ | $\downarrow$ | $\downarrow$ | $\downarrow$ |
| Offspring | 32 | 128 | 1 | 4 | 1 |

**Table 4**: Randomly choose genes to mutate. Gene 1,3,5 are selected to mutate in this example

used in the genetic algorithm:

- Parent is a set of solutions that will generate the offspring
- Offspring is a set of solutions obtained by cross over and mutation from parent;
- Population is the selected set of solutions that will move to the next generation as the new parent
- $G$ represents number of generations
- $N_{P_g}$ represents population size of $g's$ generation, $N_{P_1}$ represents initialized population size
- $N_{PF_g}$ represents pareto front size at the beginning of $g's$ generation
- $cg$ represents minimum number of offspring solutions created in $g's$ generation
- $ag$ represents minimum number of solutions included in the population

At the first generation, we initialize $N_{P_1}$ random structures from search space, and train them to obtain the objective functions; then we rank solutions by their front levels and crowding distances; thereafter, we compute the number of solutions in tier 1 front to obtain first tier pareto front size at the first generation $N_{PF_1}$

We use a similar method as in [Chapman et al., 2018] with a slight difference to update population size and offspring size by not specifying the maximum population size and maximum offspring size. To guarantee fast convergence, each solution from the first tier pareto front will be selected to do cross over and mutation; Mutation process is also slightly different for different generation in the first phase. Suppose phase I contains G generations. Then procedure is formulated as follows.

1. For the first $G/2$ generations when $1 \leq g \leq G/2$

   (a) **Cross over** Select two solutions with the probability proportional to the rank from the whole population, randomly exchange genes from both sides to generate a new offspring. Repeat this procedure $cg/2$ times.

| | Unit 1 | Unit 2 | Act1 | Act2 | Optimizer |
|---|---|---|---|---|---|
| Parent 1 | 200 | 100 | 1 | 1 | 4 |
| Parent 2 | 120 | 210 | 1 | 1 | 4 |
| | $\downarrow$ | $\downarrow$ | $\downarrow$ | $\downarrow$ | $\downarrow$ |
| Offspring | 160 | 160 | 1 | 1 | 4 |

**Table 5**: In this phase, results of activation functions and optimizer has converged; gene 1 and gene 2 of offspring are obtained by averaging gene 1 and gene 2 from two parents

| | Unit 1 | Unit 2 | Act1 | Act2 | Optimizer |
|---|---|---|---|---|---|
| Parent 1 | 120 | 210 | 1 | 1 | 4 |
| | $\downarrow$ | $\downarrow$ | $\downarrow$ | $\downarrow$ | $\downarrow$ |
| Offspring | 120 | 30 | 1 | 1 | 4 |

**Table 6**: For the whole population, few chromosome has a value of 210 for the second gene. then the second gene is most likely to be chosen to mutate

(b) **Cross over** For each solution from first tier pareto front, select another solution in the same front with the probability proportional to the rank, randomly exchange genes from both sides to generate a new offspring. This procedure takes $N_{PF_g}$ times. Process of $(a)$ and $b$ are illustrated in table 3.

(c) **Mutation** Select one solution with the probability proportional to their rank from the whole population. Select a random number of genes to mutate from the search space randomly. Repeat this procedure $cg/2$ times.

(d) **Mutation** For each solution from first tier pareto front, Select a random number of genes to mutate from the search space randomly. This procedure takes $N_{PF_g}$ times. Process of $(c)$ and $(d)$ are illustratedd in table 4

(e) **Fitness** For $cg + 2N_{PF_g}$ offspring solutions, obtain objective functions.

(f) **Rank** Combine parent and offspring; rank solutions based on front levels and crowding distances.

(g) **Selection** Choose the top $ag+N_{PF_{g+1}}$ solutions as the new population (parent) to the next generation

(h) **Iteration** g = g+1

2. For the second $G/2$ generations when $G/2 < g \leq G$

(a) **Cross over** Select two solutions with the probability proportional to the rank from the whole population, randomly exchange genes from both sides to generate a new offspring. Repeat this procedure $cg/2$ times.

(b) **Cross over** For each solution from the first tier pareto front, select another solution in the same front with the probability proportional to the rank, randomly exchange genes from both sides to generate a new offspring. This procedure takes $N_{PF_g}$ times. Process of $(a)$ and $(b)$ is the same with previous steps, which can also be illustrated in 3

(c) **Mutation** Select one solution with the probability proportional to their rank from the whole population. Calculate the frequency of each gene over the whole population; gene that occurs less is more likely to be chosen to mutate. Repeat this procedure $cg/2$ times.

(d) **Mutation** For each solution from first tier pareto front, calculate the frequency of each gene over the whole population; gene that occurs less is more likely to be chosen to mutate. This procedure takes $N_{PF_g}$ times. Process of $(c)$ and $(d)$ is illustrated in 6, which is different from previous steps

(e) **Fitness** For $cg + 2N_{PF_g}$ offspring solutions, obtain objective functions.

(f) **Rank** Combine parents and offspring; rank solutions based on front levels and crowding distances.

(g) **Selection** Choose the top $ag+N_{PF_{g+1}}$ solutions as the new population (parent) to the next generation

(h) **Iteration** g = g+1

After $G$ generations, solutions from the first tier pareto front of the last generation will be put into the Phase II where all optimizer and activation functions are consistent and fixed.

### 4.2 Phase II

At phase II, we are interested in the relationship between number of hidden units at different layers and predicted performance. As such, 3 genes that represent activation functions and optimizer type are removed in that phase I has found out the best combination of activation functions and optimizer; The cross over and mutation procedure are for the first 2 genes that represent number of hidden units for the first hidden layer and the second hidden layer. Recall that in the Phase I, there are only 8 levels for choosing the number of hidden units. For phase II, we change the search space from $64, 96, 128, \cdots, 512$ to $30, 40, \cdots, 510$. Procedure is formulated as follows:

1. Select pareto optimal set from the last generation of phase I as the parent in the beginning of phase II
2. For $G$ generations when $1 \leq g \leq G$

(a) **Cross over** Select two solutions with the probability proportional to the rank from the whole population, average two genes from both sides and round to ten digits to generate a new offspring. Repeat this procedure $cg/2$ times.

(b) **Cross over** For each solution from first tier pareto front, select another solution in the same front with the probability proportional to the rank, average two genes from both sides and round to ten digits to generate a new offspring. This procedure takes $N_{PF_g}$ times. Process of $(a)$ and $(b)$ in this phase is illustrated in table 5, which is different from phase I

(c) **Mutation** Select one solution with the probability proportional to their rank from the whole population. Calculate the frequency of each gene over the whole population; gene that occurs less is more likely to be chosen to mutate. Repeat this procedure $cg/2$ times.

(d) **Mutation** For each solution from the first tier pareto front, calculate the frequency of each gene over the whole population; gene that occurs less is more likely to be chosen to mutate. This procedure takes $N_{PF_g}$ times. Process of $(c)$ and $(d)$ is illustrated in table 6

(e) **Fitness** For $cg + 2N_{PF_g}$ offspring solutions, obtain objective functions.

(f) **Rank** Combine parent and offspring; rank solutions based on front levels and crowding distances.

(g) **Selection** Choose the top $ag+N_{PF_{g+1}}$ solutions as the new population (parent) to the next generation

(h) **Iteration** g = g+1

## 5. Results

At the beginning of the phase I, we initialize 20 random structures from the search space ($N_{P_1} = 20$), and set up number of generations $G$ at 20. The minimum number of offspring solutions $cg$ is 20 so that cross over and mutation procedures are executed for the whole population 10 times each. The minimum number of population size $ag$ is 10 which means for each generation, 10 solutions outside the first tier front are selected to be included in the next generation. For the first generation, among 20 solutions there are 6 solutions which are from the first tier. After 3 generations, combinations of activation functions and optimizer are the same for all pareto optimal solutions indicating that the optimal structure in our experiment is a network with 2 hidden layers using '**Adam**' as optimizer, of which the activation function connecting between input layer and the first hidden layer is '**relu**' and activation function connecting between the first hidden layer and the second also the last hidden layer is '**exponential**'. At the last generation of Phase I, there are 20 pareto optimal solutions representing the final group of optimal structures, of which only one structure contains more hidden units in the second hidden layer than the first hidden layer.

We then take those 20 optimal structures from the last generation of phase I as the parent in the first generation of Phase II. Note that in the phase I there are only 64 combinations of unit 1 and unit 2, after refinement, there are totally $2401(49 * 49)$ combinations. While number of optimal structures at the first generation of phase I is 6, at the last generation of phase II there are 96 optimal structures. Among all structures only 1 of them contains more hidden units in the second hidden layer than the first layer.

To further select a smaller subset of optimal solutions from pareto front, we impose pareto approach [Lu et al., 2011, Chapman et al., 2018] to combine 3 objectives functions into 1 objective function with the form of additive desirability function

$$DF = w_1 d_1 + w_2 d_2 + w_3 d_3 \tag{15}$$

where $d_1, d_2, d_3$ represent desirability values [Derringer and Suich, 1980] for 3 objective functions, $w_1, w_2, w_3$ represent the importance of different objectives and $\Sigma_{i=1}^{3} w_i = 1$. We scale each desirability value into $[0, 1]$ using one-sided transformation mentioned in [Derringer and Suich, 1980], where 0 represents the worst objective value and 1 represents the best objective value. Due to lack of enough prior information regarding the importance for each objective, we choose to explore the entire weight space to choose those solutions which can perform the best under most weight combinations. Figure 7 is the mixture plot considering different weighting scheme for additive desirability function. Among 96 solutions from the pareto optimal sets, 13 solutions are selected from pareto front which is

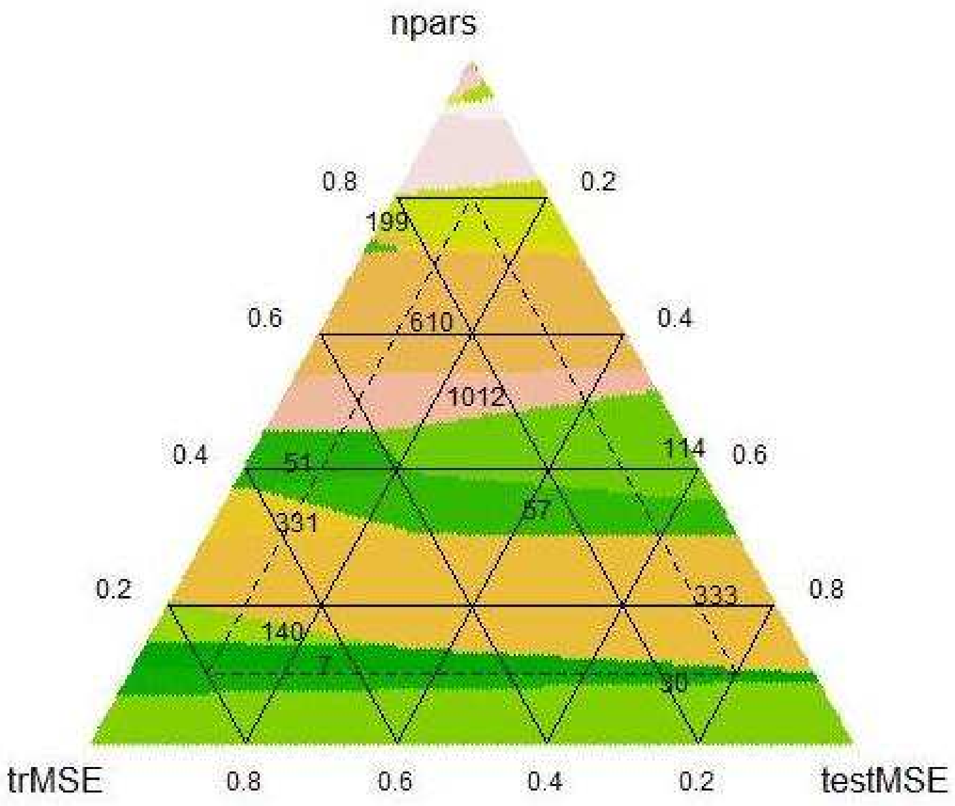


**Figure 7**: Mixture plot using entire weight space

shown in Table 4, 11 of them are labeled by their index. Figure 8 is the trade-off plot for 3 objectives, where x-axis represents the index of each solution and y-axis represents the desirability values. Sets selected from the pareto front are marked as black points in the plot. From graphs we can see that during the training, difference between training error and test error is very small, thus minimize the training error can also at the same time minimize the test error. If we add more weights to the model complexity, then probably the optimal model will be chosen as model with index 199, whose first hidden layer has 390 units, and second hidden layer has 30 units.

## 6. Conclusion and Discussion

In this paper, we show that neural networks are capable of predicting functional data by applying discretization and dimension reduction first. For those periodic curves, one can also use basis such as Fourier basis to express curves as a number of coordinates. Screening design is highly useful before using genetic algorithms since it gives an intuitive of a group of candidate solutions so that researchers can narrow down the search space and find optimal solutions in a more efficient way. In finding the promising models, usually multiple objectives are considered including predicted performance and model complexity. The NSGA-II algorithm with 2 phases is favorable in dealing with multi-objective problems in several aspects. First it divides solutions into multiple front levels and rank them using front levels and crowding distances; this approach can preserve more than the general genetic method. Second, while the search space still contains an ocean of combinations of possible parameters, it focuses on finding the most significant factors at first in the first phase by doing a quick search and takes more time for studying insignificant factors in the

| Index | Unit1 | Unit2 | Train MSE | Test MSE | # paras |
|---|---|---|---|---|---|
| 129 | 512 | 420 | 0.0017 | 0.0019 | 428520 |
| 1351 | 250 | 30 | 0.0039 | 0.0040 | 24280 |
| 333 | 400 | 190 | 0.0020 | 0.0021 | 173690 |
| 331 | 512 | 128 | 0.0021 | 0.0023 | 132724 |
| 57 | 400 | 140 | 0.0022 | 0.0023 | 128640 |
| 140 | 420 | 200 | 0.0019 | 0.0021 | 186800 |
| 1012 | 400 | 100 | 0.0024 | 0.0026 | 92600 |
| 610 | 490 | 70 | 0.0026 | 0.0028 | 72730 |
| 30 | 512 | 320 | 0.0018 | 0.0020 | 327220 |
| 7 | 450 | 240 | 0.0019 | 0.0020 | 230990 |
| 199 | 390 | 30 | 0.0037 | 0.0039 | 29180 |
| 51 | 300 | 140 | 0.0022 | 0.0024 | 114140 |
| 114 | 480 | 110 | 0.0023 | 0.0024 | 110810 |

**Table 7**: Selected optimal model structures from pareto front

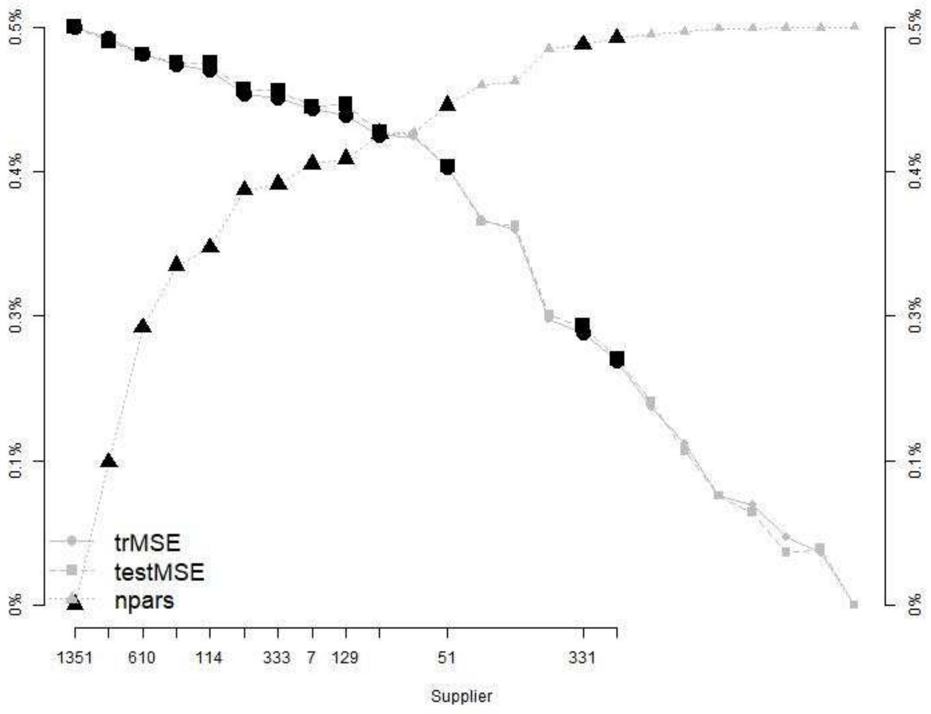


**Figure 8**: Trade-off plot for 3 objective functions

second phase; compared to the method that directly search the entire space, it can tremendously lower down the time and cost. The pareto approach is more acceptable especially when prior knowledge of importance for different objectives is not explicit. Under those circumstances, the promising way is to assign the weight to those objectives from entire space and select solutions from pareto front which perform the best under the majority of weighting schemes. If researchers know the importance of each objective, then one can certainly assign different weights to the objectives and turn multi-objective problem into the single objective problem. Above methods together form a systematic way in finding the promising model when multiple objectives should be taken into consideration, and can be generalized to find more structure especially when we aim to study the impact of batch normalization, dropout rate or learning rate and so on.

## 7. Acknowlodgement

This work is supported by AFOSR/AOARD grant FA2386-18-1-4104

## REFERENCES

[Chapman et al., 2018] Chapman, J. L., Lu, L., and Anderson-Cook, C. M. (2018). Using multiple criteria optimization and two-stage genetic algorithms to select a population management strategy with optimized reliability. *Complexity*, page 18.

[Deb et al., 2002] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Ieee Transactions on Evolutionary Computation*, 6(2):182–197.

[Derringer and Suich, 1980] Derringer, G. and Suich, R. (1980). Simultaneous-optimization of several response variables. *Journal of Quality Technology*, 12(4):214–219.

[Grenander, 1950] Grenander, U. (1950). Stochastic processes and statistical inference. *Arkiv för Matematik*, 1(3):195–277.

[Karhunen, 1946] Karhunen, K. (1946). *Zur Spektraltheorie stochastischer Prozesse*.

[Lu et al., 2011] Lu, L., Anderson-Cook, C. M., and Robinson, T. J. (2011). Optimization of designed experiments based on multiple criteria utilizing a pareto frontier. *Technometrics*, 53(4):353–365.

[Ramsay and Silverman, 1997] Ramsay, J. and Silverman, B. (1997). *Functional Data Analysis*.

[Ramsay, 1982] Ramsay, J. O. (1982). When the data are functions. *Psychometrika*, 47(4):379–396.

[Rao, 1958] Rao, C. R. (1958). Some statistical methods for comparison of growth curves. *Biometrics*, 14(1):1–17.

[Ruder, 2016] Ruder, S. (2016). An overview of gradient descent optimization algorithms. *ArXiv*, abs/1609.04747.

[Stoer and Bulirsch, 2002] Stoer, J. and Bulirsch, R. (2002). *Introduction to numerical analysis*. Texts in applied mathematics. Springer.

[Wang et al., 2016] Wang, J.-L., Chiou, J.-M., and Müller, H.-G. (2016). Functional data analysis. *Annual Review of Statistics and Its Application*, 3(1):257–295.