Multi-Objective Problem Pareto Front Metamodeling Optimization Using NSGA-II

Abstract

Though neural networks have been applied on approximating Pareto fronts of multi-objective optimization problems using surrogate models, existing works have focused on categorizing optimization genetic algorithms with BPNN surrogate models. However, this paper explores specifically the use of the NSGA-II algorithm with a BPNN surrogate model, with the aid of the ZDT1 test function. The exploration concluded the effectiveness of NSGA-II in approximating the Pareto front with a surrogate model of a multi-objective optimization problem using the IGD value and HV indicator of the obtained Pareto front.

Keywords: multi-objective optimization; surrogate model; BPNN; NSGA-II

Contents

1 Introduction

Multi-objective optimization is a branch of mathematical optimization where multiple objective functions of the same inputs are to be optimized. Multi-objective optimization presents a challenge in mathematics and computer science, for the need of an optimal decision in the trade-off between conflicting objective functions – bringing one closer to one objective comes at a cost of moving further away from others. Therefore, there is no single solution in which all of the objective functions are maximized. Rather, a set of solutions, called the Pareto front, is the result of a multi-objective optimization problem, in which the points represent the optimal points where no movements can make an objective better-off without making another worse-off. The set of points form a n-dimensional manifold that separates the space into three regions: an infeasible region that cannot be reached under any input within their domains, a nonoptimal region where changing an input could improve an objective without making another worse-off, and the manifold itself consisting of the optimal solutions. Because of the challenge multi-objective optimization presents, this paper aims to explore the use of neural networks in finding a surrogate model of the Pareto front.

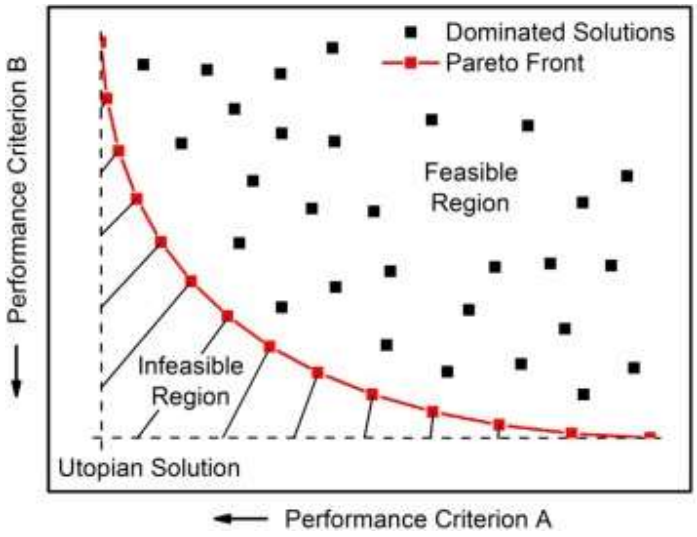


Figure 1. The Pareto front of a 2-objective optimization problem.

A surrogate model, or metamodel, is used to model the outcome of some other model given an input. In the case of finding the Pareto front in a multi-objective problem, since finding the Pareto front given a dataset requires large amounts of computational costs, and finding an exact function describing the Pareto front is impossible, we use a neural network to find a surrogate model, which greatly reduces the computing needed in optimizing multi-objective problems by modeling the Pareto front.

Because of the wide application of multi-objective optimization in fields of not only science, but economics and finance, the demand to find solutions of these problems has led to the use of neural networks, in the assistance of surrogate models, in approximating the Pareto front of multi-objective optimization problems. For example, Díaz-Manríquez et al.[1] takes a mathematical approach and analyzes the integration of different surrogate models into multi-objective evolutionary algorithms, while Schweidtmann et al.[2] takes the TS-EMO algorithm into application, optimizing the performance of chemical processes. This paper, however, aims specifically to explore and optimize the use of NSGA-II in combination of a BPNN surrogate model to approximate the Pareto front of multi-objective optimization, with the aid of the ZDT1 test function.

2 Construction of the Surrogate Model

2.1 BPNN

BPNN, or backpropagation neural network, is a type of artificial neural network that uses the backpropagation algorithm to update weights in the individual neurons of the neural network, based on the gradient of the loss function with respect to the weights, in order to minimize the loss function.

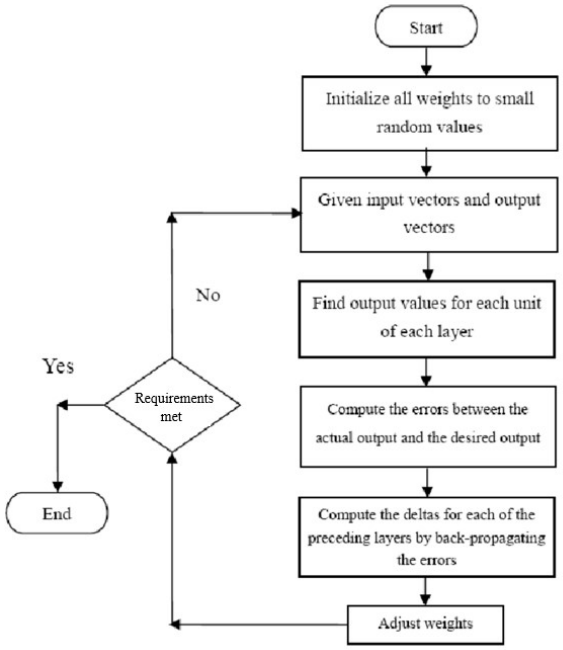


Figure 2. Backpropagation algorithm flow chart.

An artificial neural network consists of individual neurons, organized in layers whose properties include a step function and a vector consisting of weights. Each node, or neuron, of the network receives an input from the neurons in the previous layer, multiplies it with the weight, and runs the result through the step function and outputs the result to the neurons in the next layer, as shown in Figures 3 and 4.

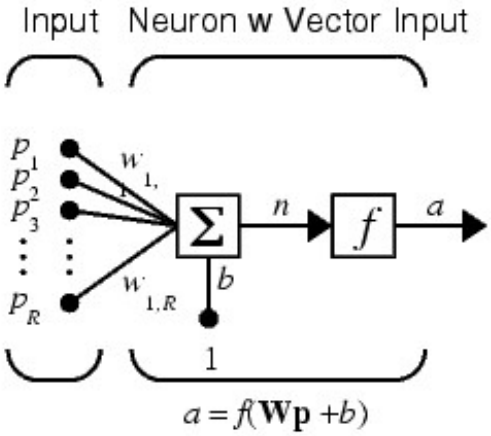


Figure 3. The mechanism of a single neuron.

In back propagation, random weights are selected for each of the neurons. Then, the output is evaluated from a sample input from the training set. The weights of the neurons in the hidden layer is then adjusted to minimize the error between the calculated output and the desired output, given by the training data. Advantages of backpropagation includes its efficiency and simplicity, as well as the normality of using BPNN as a standard method that generally outperforms other algorithms.

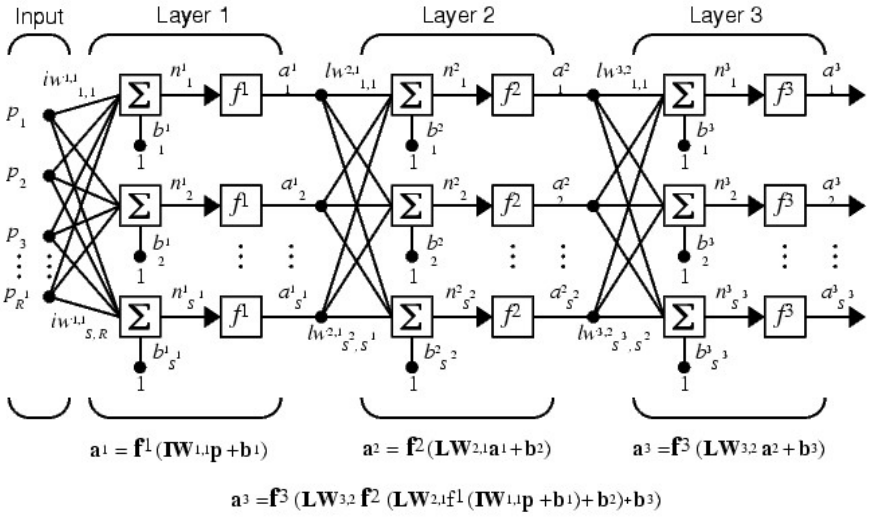


Figure 4. Visual representation of a feedforward artificial neural network.

2.2 Test Function: ZDT1

In order to evaluate the performance of NSGA-II in combination with a BPNN surrogate model, we need data of a multi-objective optimization problem to train the genetic algorithm. Rather than using data from the real world, we use a test function in this case, just to demonstrate the capability of NSGA-II. Since synthetic test functions are generally[3] intentionally difficult for an algorithm to find a solution to but computationally efficient, they are ideal for the exploration here of the genetic algorithm as it can efficiently evaluate an algorithm, not only fast, but comprehensively as it tests the algorithm's capability to bypass difficulties that are present in real life optimization problems.

Therefore, the ZDT1[4] test function can perform optimally in this scenario – genetic algorithms find it difficult to optimize, but the computation is relatively simple. The two objective functions of the bi-objective ZDT1 is expressed as f and g in the following:

$$\begin{cases} g = 1 + 9\left(\sum_{i=2}^{n} x_i\right)(n-1)^{-1} \\ f = \min\left(x_1, g\left(1 - \sqrt{\frac{x_1}{g}}\right)\right) \end{cases}$$

In this evaluation of NSGA-II, the number of input variables to the function is chosen to be five, not only presenting enough difficulty for the algorithm but also allowing the algorithm to optimize quickly. The number of objectives is also chosen to be two because of its simplicity to visualize the results on a two-axes graph. The function is then used to generate 100 data points for the training of the NSGA-II genetic algorithm.

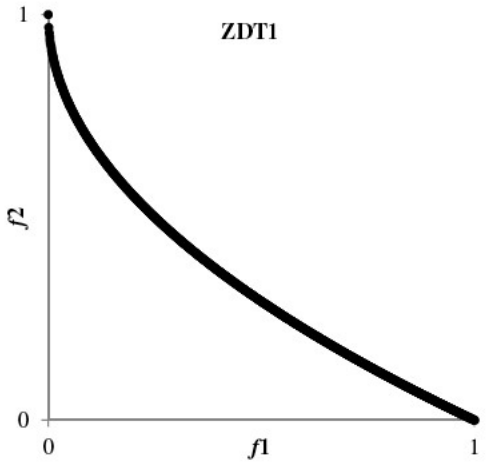


Figure 5. The true Pareto front of the bi-objective ZDT1 test function.

3 Multi-Objective Optimization Based on NSGA-II

NSGA-II is a multi-objective genetic algorithm[5]. It is based off of NSGA, but makes improvements from the original in terms of efficiency and addition of elitism. It has become widely regarded as one of the standard genetic algorithms for multi-objective optimization.

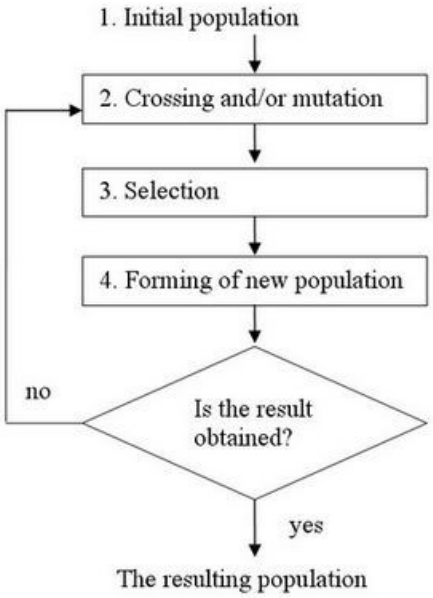


Figure 6. The framework for genetic algorithms.

Like all other genetic algorithms, NSGA-II is based off of the idea of evolution – survival of the fittest. In NSGA-II, an initial population of individuals is inputted or is randomly generated. A child population is then produced randomly through crossing and mutation within the parent population. Then, in traditional genetic algorithms, a fitness function is assigned to each individual in the new population. Based on the traits of each of the individuals, each individual has a fitness value, and the individuals who have the highest fitness survive and reproduce in the next round. However, in NSGA-II, solutions are sorted in fronts. Let individual i have objective function values $f_1(i)$ and $f_2(i)$. Then, an individual i is said to dominate individual j if:

$$f_1(i) \geq f_1(j) \text{ and } f_2(i) \geq f_2(j) \text{ and } \big(f_1(i) > f_1(j) \text{ or } f_2(i) > f_2(j)\big) \quad (1)$$

The individuals that are dominated by no other individual is classified into F_1. Then, individuals in F_1 are removed, and those that are remaining who are dominated by no other

individual are classified into F_2, and so on. The top half of the sorted population remains to reproduce in the next round. However, if a split occurs within a front, the individuals who have the least neighbors survive to preserve diversity and discourage local optima. A diagram of the selection process is shown in Figure 7.

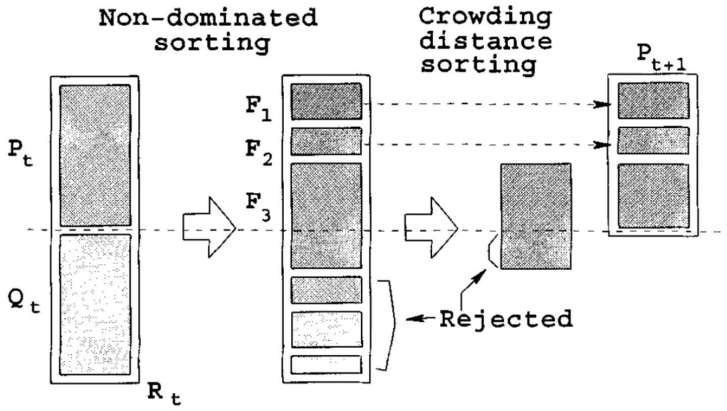


Figure 7. Selection in NSGA-II.

The NSGA-II algorithm is implemented in MATLAB using PlatEMO[6], as shown in Figure 8.

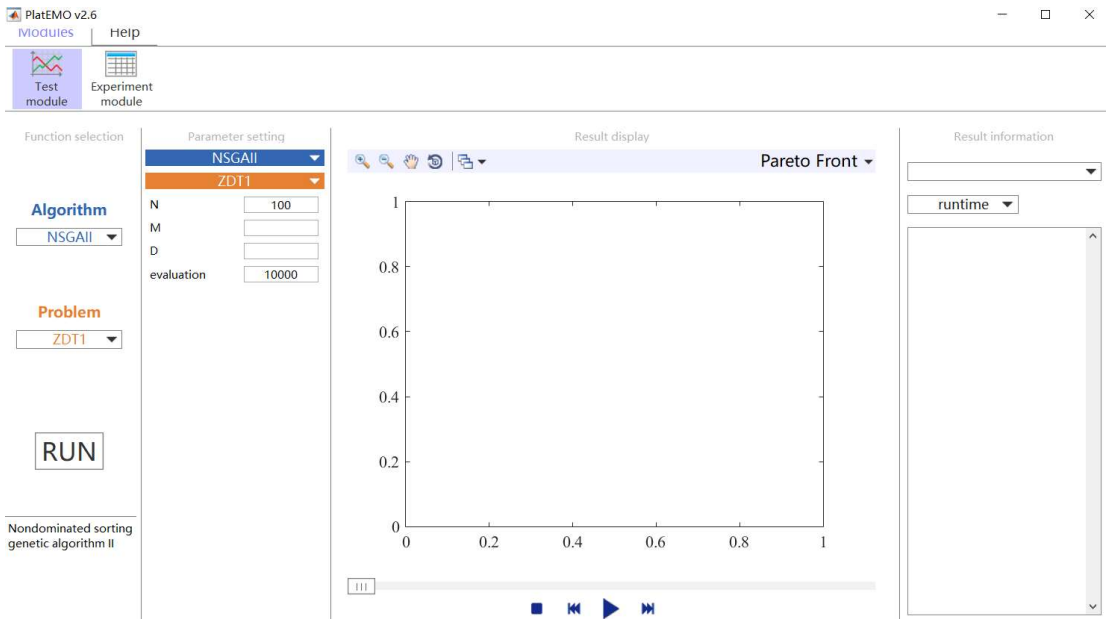


Figure 8.GUI of PlatEMO v2.6.

4 Results and Discussion

Figure 9 illustrates the Pareto optimal front obtained by NSGA-II, combined with a BPNN surrogate model with 8 hidden layers. Figure 10 shows the performance indicators IGD and HV of the optimization.

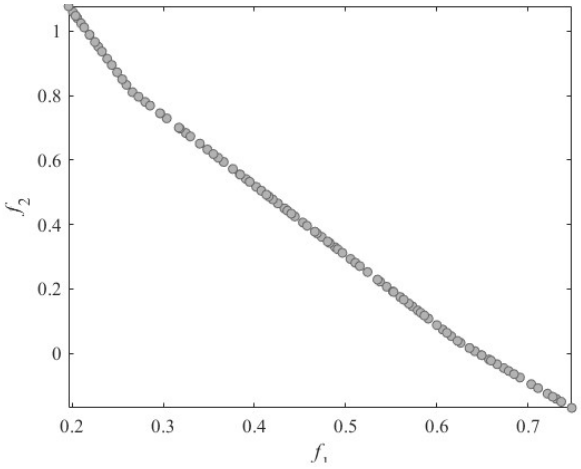


Figure 9. Pareto front obtained by NSGA-II.

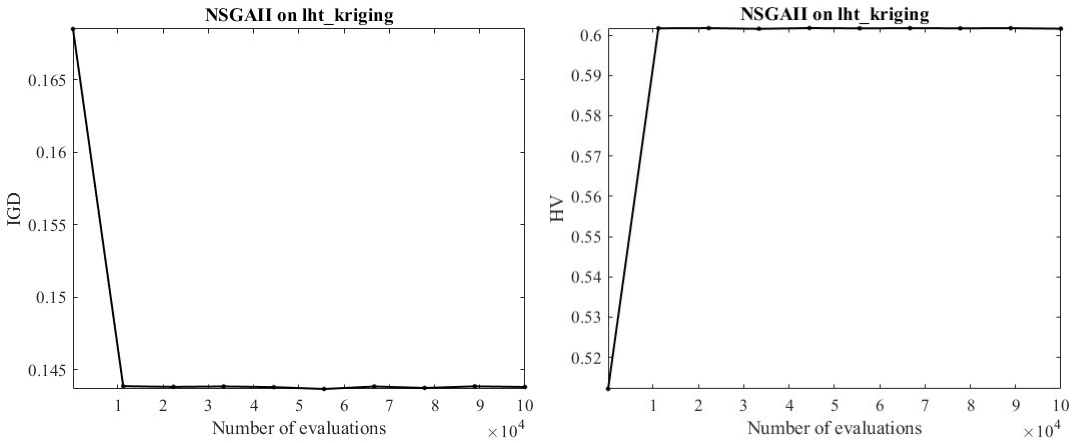


Figure 10. Performance indicators IGD and HV

Compared to real Pareto front of ZDT1 (shown in Figure 11), it can be seen that the POF obtained by NSGA-II fit well with the real front, which indicates the model built in this study is reasonable.

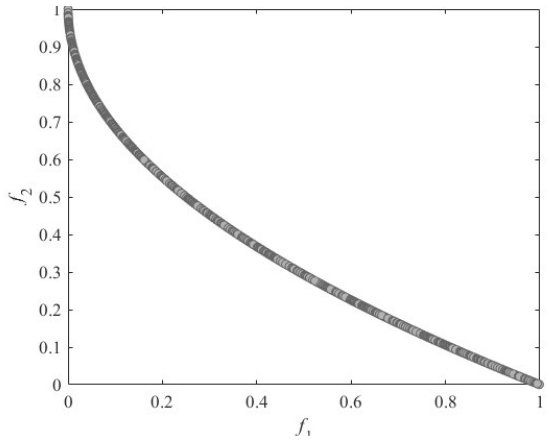


Figure 11. Actual Pareto front of the ZDT1 test function.

5 Result Analysis

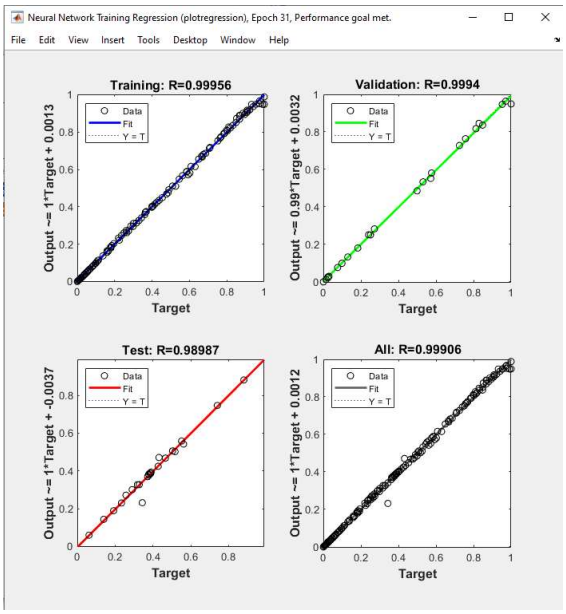
5.1 Error analysis

To better reflect the performance of the BPNN model, two widely used evaluation indicators were employed, which can be calculated by the following equations:

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

$$R^2 = \frac{\sum_1^n (y - \hat{y})^2}{\sum_1^n (y - \bar{y})^2}$$

Where n is the number of test samples, y denotes the true value of responses while $\hat{y}$ denotes the predicted value. Note that: the closer to 1 the R^2 is, the better accuracy the model performs. Figure 9 shows the R^2 analysis results.



5.2 Sensitivity Analysis

The number of hidden layers of the BPNN was modified between four and ten. The value of R remained above 0.93, suggesting the robustness of the model.

6 Conclusion

In this study, a BPNN is trained as the surrogate model to model the Pareto front of a multi-objective optimization test problem, ZDT1, and the accuracy has proved to be high. The NSGA-II algorithm is then employed to optimize the BPNN surrogate model to better fit the real Pareto front of the problem. The results indicate the Pareto front obtained by NSGA-II and BPNN is a good fit ($R^2 = 0.998$) of the actual Pareto front of the ZDT1 test function.

To further expand this study, different test functions that simulate real-world multi-objective optimization problems can be used to test the effectiveness of the NSGA-II algorithm on a BPNN surrogate model. An increased number of objective functions and number of input variables can also be tested using this methodology to further support the effectiveness, or expose the limitations of, the methodology used in this study.

References

[1] Díaz-Manríquez, A., Toscano, G., Barron-Zambrano, J., & Tello-Leal, E. (2016, June 12). A Review of Surrogate Assisted Multiobjective Evolutionary Algorithms. Retrieved September 13, 2020, from https://www.hindawi.com/journals/cin/2016/9420460/

[2] Schweidtmann, A., Clayton, A., Holmes, N., Bradford, E., Bourne, R., & Lapkin, A. (2018, July 04). Machine learning meets continuous flow chemistry: Automated optimization towards the Pareto front of multiple objectives. Retrieved September 13, 2020, from https://www.sciencedirect.com/science/article/pii/S1385894718312634

[3] Rostami, D. (2019, July 25). Synthetic Objective Functions and ZDT1. Retrieved September 13, 2020, from https://datacrayon.com/posts/search-and-optimisation/practical-evolutionary-algorithms/synthetic-objective-functions-and-zdt1/

[4] E. Zitzler, K. Deb, and L. Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Evolutionary Computation, 8(2):173-195, 2000

[5] Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, 6(2), 182-197. doi:10.1109/4235.996017

[6] Ye Tian, Ran Cheng, Xingyi Zhang, and Yaochu Jin, PlatEMO: A MATLAB Platform for Evolutionary Multi-Objective Optimization [Educational Forum], IEEE Computational Intelligence Magazine, 2017, 12(4): 73-87"

Appendix I: BPNN Implementation in MATLAB

```
clc
clear net

input = x;
output = y;
[n, inputnum] = size(input);
outputnum = size(output,2);

hiddenlayers=8;

input_train=input(1:0.8*n,:)';
input_test=input(0.8*n+1:end,:)';
output_train=output(1:0.8*n,:)';
output_test=output(0.8*n+1:end,:)';

net=newff(input_train,output_train,hiddenlayers,{'tansig','purelin'},'trainlm
');

net.trainParam.epochs = 2000;
net.trainParam.lr = 0.1;
net.trainParam.goal = 0.0001;
net.trainParam.max_fail = 100;

[net,per2]=train(net,input_train,output_train);
```

Acknowledgements

I would like to thank Ms. Lin Chen for encouraging students at our school to pursue research in STEM and advising this study. I would also like to thank teachers, friends, and family who showed support during the duration of this process.