IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 13, No. 2, June 2024, pp. 1969~1979
ISSN: 2252-8938, DOI: 10.11591/ijai.v13.i2.pp1969-1979

Deep learning-based prediction of float model performance in floatplanes: A case study on lift-to-drag coefficient ratio

Faisal Fahmi[1,2], Rizqon Fajar[1], Sigit Tri Atmaja[1], Erwandi[3], Daif Rahuna[3]
[1]Department of Information and Library Science, Airlangga University, Surabaya, Indonesia
[2]Research Center for Transportation Technology, National Research and Innovation Agency, South Tangerang, Indonesia
[3]Research Center for Hydrodynamics Technology, National Research and Innovation Agency, Surabaya, Indonesia

Article Info

Article history:

Received Jun 26, 2023
Revised Nov 15, 2023
Accepted Dec 3, 2023

Keywords:

Computational design
Deep learning
Floatplane
Genetic algorithm
Hydrodynamic performance

ABSTRACT

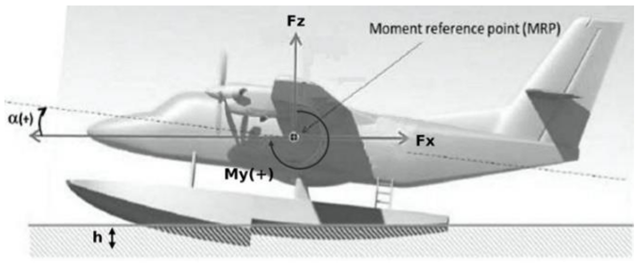
Developing an engineering design is resource-intensive and time-consuming, particularly for the floats of a floatplane design, due to its complexity and limited testing facilities. Intelligent-based computational design (IBCD) techniques, which integrate computational design techniques and machine learning (ML) algorithms, offer a solution to reduce required testing by providing predictions. This paper proposes a deep learning (DL)-based IBCD method for modeling floats' lift-to-drag coefficient ratio (C_L/C_D), where DL is one of the most powerful ML. The proposed method consists of two phases: hyper-parameter optimization and DL model training and evaluation. A genetic algorithm (GA) is employed in the first phase to explore complex hyper-parameter combinations efficiently. Evaluation of the predicted C_L/C_D of the floats using the DL model resulted in a satisfactory R-squared of 0.9329 and the lowest mean squared error (MSE) of 0,001536. These results demonstrate the ability of DL model to predict the float's performance accurately and can facilitate further design optimization. Thus, the proposed method can offer a time-efficient and cost-effective solution for predicting float performance, aiding in optimizing floatplane designs and enhancing their functionalities.

Corresponding Author:

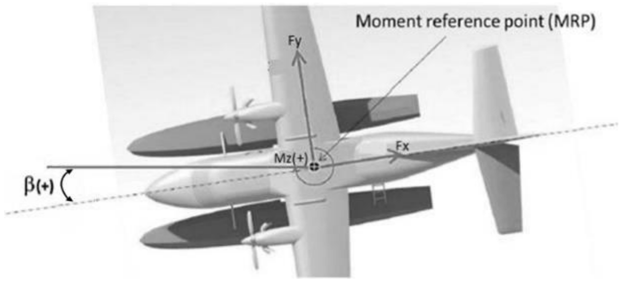
Faisal Fahmi
Department of Information and Library Science, Airlangga University
Jl. Dharmawangsa Dalam, Airlangga, Gubeng, Surabaya, 60286, Indonesia
Email: faisalfahmi@fisip.unair.ac.id

1. INTRODUCTION

A floatplane, i.e., aircraft that can take off and land on the water's surface using a pair of floats under the fuselage, is vital for a country with many islands and remote areas, e.g., Indonesia, to connect people in safe, fast, and reliable ways [1]. However, designing and testing a floatplane poses significant challenges and difficulties due to various factors, including aerodynamics, hydrodynamics, and the complex interaction between the floatplane and water [2], [3]. Predicting the performance of such a design is crucial but can be complex. For instance, the design of each float component can significantly affect the performance of the float and, consequently, the overall functionality of the floatplane. Extensive measurements and analysis have been conducted to study the forces and moments arising from the interaction between floats and water at the Indonesian Hydrodynamics Laboratory (IHL) [4]. The measured parameters include drag force (Fx), lift force (Fz), side force (Fy), angle of attack (α), angle of yaw (β), yaw moment (Mz), and pitch moment (My), as shown in Figures 1(a) and 1(b), which play crucial roles in ensuring the desired performance and maneuverability of the floatplane. The focus of that investigation is to predict the feasibility of smooth take-off and landing for the floatplane, optimize runway requirements, and address other operational considerations. However, the current design and testing process for a single part of aircraft in Indonesia can take time, requiring over a year to complete and incurring very high costs for researchers and industries. Therefore, an urgent improvement to the design and testing process is necessary to address these challenges effectively.



(a)



(b)

Figure 1. A floatplane model along with forces and moments from two different views, i.e., (a) From the side view by the angle of attack and (b) From the top view by the angle of yaw

On the other hand, intelligence-based computational design (IBCD) has emerged as a promising approach by integrating computational design techniques and machine learning (ML) algorithms. This integration can revolutionize the design and testing process, providing a time-efficient, cost-effective solution while fostering sustainability and improving overall design performance [5]–[7]. For the design of a float of the floatplane, this innovative approach holds immense potential to overcome the complexities involved in predicting the performance and its impact on the functionality of the floatplane. The initial study on modeling float characteristics using an ML algorithm is done by predicting the drag force of the float using input data obtained from a numerical simulation [8]. Additionally, the current application of ML in the aerospace field not only employs numerical data but also incorporates numerical data alongside image data [9]. To simplify the discussion, this paper focuses on the numerical data only.

Deep learning (DL) is a powerful ML technique that utilizes computational models of neural network inspired by the structure of the human brain [10], [11] for modeling complex and non-linear design parameters. However, the effectiveness of DL often comes at the cost of demanding significant computational resources due to the complex architecture of the networks. One way to minimize these computational demands is by applying optimization techniques, e.g., the selection of vital parameters and the reduction of the search space. These vital parameters, referred to as hyper-parameters, are pivotal in governing the learning process during the development of an optimal DL model, where this development typically involves hyper-parameter optimization followed by DL training based on the optimized hyper-parameters [12]. However, optimizing a range of hyper-parameter values can be challenging without prior knowledge, and the difficulty increases when the search spaces are extensive. A genetic algorithm (GA) can solve the lack of prior knowledge and big search spaces [13].

The GA is a robust meta-heuristic algorithm inspired by natural selection, where the algorithm can efficiently explore high-dimensional and non-convex search spaces, capable of handling complex relationships between hyper-parameters and network performance [10], [14]. GA creates a population of candidate solutions, subjecting them to selection, crossover, and mutation to converge towards the optimal solution. In the context of DL's hyper-parameter optimization, each candidate solution represents a set of hyper-parameters for the network. The algorithm initiates with a population of randomly selected hyper-parameters, evaluates their fitness using a designated function, and then iteratively refines the candidates through crossover and mutation to improve hyper-parameter configurations.

Building upon the foundation of DL and GA optimization, this research introduces several noteworthy contributions. Firstly, the research achieved a significant milestone in developing an DL model that can accurately predict the performance of a float model designed for a floatplane, explicitly focusing on determining the ratio of lift coefficient to drag coefficient (CL/CD). Secondly, the DL model has been carefully developed to incorporate various critical input variables, including parameters such as speed (v), angle of attack (α), and Draft (h). These inputs are derived from experimental data gathered under precise and well-defined conditions, enhancing the precision and reliability of the DL model. Finally, the DL model for CL/CD provides valuable insights for optimizing float designs, particularly for small floatplanes operating in the maritime of Indonesia, extending its usefulness beyond prediction alone and potentially enhancing operational efficiency in this specific maritime context.

2. A METHOD TO PREDICT THE PERFORMANCE OF A FLOAT

The proposed method to predict the performance of float of floatplane is shown in Figure 2. It consists of seven steps: i) Obtain the dataset, ii) Determine input and output, iii) Pre-process dataset, iv) Optimize hyper-parameters with GA, v) Train DL model, vi) Predict performance using DL model, and vii) Evaluate DL model, where steps i-iii are called data preparation, and steps iv-vii are iterative works to optimize hyper-parameter and DL model, sequentially. The detailed contents for each step are described in Figure 2.

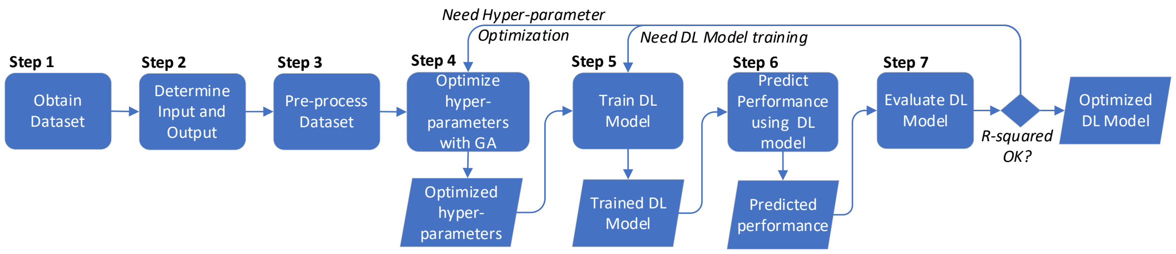


Figure 2. Development process of DL model using GA

2.1. Step 1: Obtain the dataset

To accurately predict the performance of an engineering design using a DL model, obtaining a relevant and high-quality dataset is essential [15]. There are several ways to obtain datasets, including collecting data manually, using existing datasets from public repositories, and generating synthetic data [16]. Due to limited data on the performance of float used in floatplanes, a collaboration with the data owner can be a solution. In Indonesia, the data owner of the float can be a research institution or aircraft manufacturing industry.

2.2. Step 2: Determine input and output

Identifying input and output variables, or features, is paramount when constructing a DL model that exhibits precise predictions of the target variable. Exploratory data analysis (EDA), domain knowledge, and expert opinion can be used to select the most relevant input and target variables. In particular cases, feature engineering techniques used to create new features may be necessary to improve DL model performance [17], [18]. The details of this step are as follows,

– Determine input variables: the input variables are the various settings used to test the performance of the float model, e.g., hydrodynamic performance. EDA techniques, such as data visualization (histograms, scatter plots, heatmaps), domain knowledge, or expert opinion, are used to select input variables. Due to limited data on floatplane models, domain knowledge or expert opinion plays a significant role in choosing relevant input variables [19], [20].
– Determine output variable(s): the output variable is the target variable that the DL model predicts based on the input variable(s). In this research, the output variables are the performance of the float design. Thus, the output variables should be explicitly provided in the training data, where the DL technique works to build a model that can accurately predict these variables for the unseen input data. Like the input variables, domain knowledge or expert opinion is the primary consideration in determining the output variables [20].

- Perform feature engineering (if necessary): if the input and output variables are deemed insufficient, feature engineering techniques can be applied to create more informative features for the DL model [21]. Feature engineering involves extracting relevant information from the data to facilitate the understanding of the model. Well-designed features enhance accuracy and robustness, while poor ones may lead to overfitting, underfitting, or low predictive power [22]. In the same way as input and output variables, domain knowledge and expert opinion are utilized to engineer new features that capture relevant information in the dataset.

2.3. Step 3: Pre-process dataset

Data pre-processing is a crucial preliminary step in DL techniques. It involves manipulating or removing unnecessary data to ensure quality and enhance performance. This step includes data cleaning and splitting the dataset containing input and output variables, i.e., testing settings and performances of the float design, defined in step 2. The details of this step are described below,

- Data cleaning: extensive data cleaning is performed to eliminate errors, inconsistencies, and duplicates. Missing data is handled by imputing or removing the corresponding rows in the dataset. Outliers are checked and removed to prevent their impact on the results.
- Data splitting: the dataset is divided into a training set and a testing set. The training set is used to train the DL model, while the testing set is used to evaluate the performance of the trained DL model. In this research, the splitting of the dataset is determined by criteria from the experts, e.g., the testing set contains the float performance for the critical testing settings.

2.4. Step 4: Optimize hyper-parameters with GA

Hyper-parameters significantly impact the performance of DL models [23]. However, the hyper-parameter settings during the DL training process, including network architectures (e.g., number of hidden layers, activation function, and optimization function), cannot be learned from data. Thus, finding the optimal settings of hyper-parameters can be challenging and time-consuming, particularly for large and complex networks. In order to tackle these challenges, employing a GA for hyper-parameter optimization is a practical approach. GA demonstrates an efficient exploration of high-dimensional and non-convex search spaces, along with the ability to handle complex and non-linear relationships between hyper-parameters and network performance [24].

A GA is a metaheuristic optimization algorithm inspired by natural selection [25], [26]. The algorithm creates a population of candidate solutions and evolves them through selection, crossover, and mutation to find the optimal solution. In hyper-parameter optimization of DL, each candidate solution represents a set of hyper-parameters for the network. The algorithm starts with an initial population of random hyper-parameters. Then, the fitness of each candidate is evaluated using a fitness function. Finally, the algorithm selects the best candidates, combines their traits through crossover and mutation, and generates new candidates in an iterative process. The pseudocode of GA used in this step is shown in Algorithm 1.

Algorithm 1. Hyper-parameter optimization with GA

```
// Define the genetic algorithm parameters
population_size   = <some number>
num_generations   = <some number>
mutation_rate     = <some value between 0 and 1>
parameter_choices = {'nb_neurons': <an array containing options for a number of
neurons>,
                     'nb_layers': <an array containing options for a number of
layers>,
                     'activation': <an array containing options for activation
functions>,
                     'optimizer': <an array containing options for optimizers>}
dataset           = <training set generated in Step 3>
// Create an initial population of random hyper-parameters
population = generate_population(population_size, parameter_choices)
// Repeat for a fixed number of generations
for generation in range(num_generations):
   // Create a new population of individuals through selection and mutation
   new_population = []
   for i in range(population.size):
      // Evaluate the fitness of each individual in the population
      for each individual in the population:
         individual.fitness = evaluate_fitness(individual, normalize(dataset))
      // keep only qualified individuals; otherwise, remove individuals
      new_population = keep_individual(population)
      // produce new individuals to replace the removed ones
```

```
23.        for j in range (population.size-new_population.size)
24.           // Perform selection to choose two parents
25.           parent1 = select_individual(population)
26.           parent2 = select_individual(population)
27.           // Perform breed to create a new individual
28.           child = breed(parent1, parent2)
29.           // Perform mutation on the child
30.           child = mutate(child, mutation_rate)
31.           // Add the new individual to the new population
32.           new_population.append(child)
33.         // Replace the old population with the new population
            population = new_population
      // Select the best individual from the final population
      best_individual = select_best_individual(population)
```

In Algorithm 1, generate_population (population_size) in line 8 generates a population of candidate solutions, where each candidate contains four parameters of hyper-parameter (number of neurons in each hidden layer, number of network layers, activation function, and optimizer) for a DL and each value of the parameters is randomized from parameter_choices, accordingly. evaluate_fitness (individual, normalize (dataset)) evaluates the fitness of an individual by training a DL with the corresponding hyper-parameters and evaluating its performance, i.e., mean squared error (MSE), where the dataset is normalized to prevent domination of feature(s) and split into a training set and validation set used in the training and evaluation process, respectively. keep_individual (population) sorts the fitness of the individual's MSE score and removes the individual(s) with a low score of MSE. Lines 22-31 work to replace the removed individuals with new, better individuals by breed and mutate functions. The select_individual (population) function selects an individual from the population for mating based on their fitness. Mutate (child, mutation_rate) performs a mutation operation on the child with probability set by mutation_rate. select_best_individual (population) selects the individual with the highest fitness from the final population.

2.5. Step 5: Train DL model

After hyper-parameter optimization in step 4, the next step is to train a DL model using the optimized hyper-parameters. The training process involves updating the model using a backpropagation algorithm, which leverages the gradient of the loss function concerning the model parameters. This gradient is used to adjust the weights and biases of the neurons, minimizing the error between predicted and actual output. The loss function evaluates the quality of the model and its ability to capture underlying data patterns [23]. Similar to step 4, the MSE is employed as the loss function. The training process continues for multiple epochs until the DL model converges or when error validation stops decreasing. Additional techniques, such as regularization to prevent overfitting, can be considered [27], [28]. Overfitting occurs when the DL model captures noise in the training data rather than the underlying patterns [29].

2.6. Step 6: Predict performance using the DL model

Once a DL model is trained, the model can be used to make predictions on the testing set defined in step 3. The prediction can be generated by passing the testing set through the trained model. To maintain consistency, we treat the testing set similarly to the training set, e.g., by applying the same normalization techniques [30]. After the prediction is generated, the prediction results will be evaluated in step 7.

2.6. Step 7: Evaluate DL model

This step evaluates the performance of the trained DL model by evaluating the resulting prediction, where the evaluation metric used in this paper is R-squared (coefficient of determination). R-squared can provide valuable insights into model accuracy and generalizability, aiding informed decisions for optimization [31]. R-squared assesses how well the model fits the data by comparing variances of predicted values to actual values, where a higher R-squared value indicates a higher fit [30]. If the performance of the trained DL model is unsatisfactory, the process returns to step 5 for at most three iterations. Otherwise, proceed to step 4 and continue.

3. RESULTS AND DISCUSSION

The experiment describes the results of applying the proposed method in three sub-sections. Firstly, sub-section 3.1 discusses the outcomes of the data preparation process, encompassing steps 1 to 3 of the proposed method. In sub-section 3.2, an in-depth analysis is performed on the results of hyper-parameter optimization that utilizes GA done in step 4. Finally, sub-section 3.3 provides the result of optimizing DL model, detailing the results achieved through the training and evaluation steps performed in steps 5 to 7 in the method. The detailed descriptions are as follows,

3.1. Results of data preparation (Steps 1-3)

The dataset used in this study was obtained from the IHL, a facility of the National Research and Innovation Agency (BRIN), where the dataset is not publicly accessible. This comprehensive dataset encompasses empirical data obtained through experimentations involving a float model of a floatplane. The experiments were done carefully in a water pool with a towing tank to observe the movement of the float. The collected data contains the forces and moments working on the float during these experiments, depicting the behavior of the floatplane in the water.

The obtained dataset is analyzed to determine the input and output for the DL model. The dataset contains five features, including Draft (the depth of the submerged portion of a float), Alpha (angle of attack), Beta (angle of yaw), Vm (the real speed works on the model of a float during the experiment), and Vs (conversion of speed for the actual size of a float), as shown in Table 1. The hydrodynamic performances are forces and moments, where the forces include Fx (drag or resistance), Fy (side force), and Fz (lift), and the moments include Mx (roll moment), My (pitch moment), and Mz (yaw moment). After discussion with the hydrodynamics experts, the input(s) and output(s) used in this research are determined to contain only four inputs and a newly engineered output shown in Table 2, where the inputs are Draft, Alpha, Beta, and Vs, and the output is the result of Fz divided by Fx called C_L/C_D (the ratio of lift to drag coefficient).

Table 1. Dataset of the tested float model

	Draft (mm)	Alpha (deg)	Beta (deg)	Vm (m/s)	Vs (m/s)	Fx (N)	Fy (N)	Fz (N)	Mx (Nm)	My (Nm)	Mz (Nm)
0	570	0	-5	5	15.365	16286	9157.7	-14747	-74821	35276	32808
1	570	0	-5	6	18.336	19396	12567	-8383.4	-101000	33862	44438
2	870	-2	0	3	9.2151	9443.1	-292.07	-31666	5253.5	-8259.3	748.23
3	870	-2	0	4	12.266	14125	51.657	-31648	11945	-3070.4	3591.9
4	870	-2	0	5	15.329	19936	-301.67	-40772	32238	-24777	6904.6
...	...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...	...
229	570	10	2.5	6	18.278	17234	-4509	81623	21584	91800	-7610
230	770	10	2.5	3	9.2208	7204.6	-3365.2	19967	29544	-4060.6	3831.5
231	770	10	2.5	4	12.273	12937	-4070.1	33554	28677	45224	-1964.5
232	770	10	2.5	5	15.332	21740	-5265.3	61005	23578	95500	-12500
233	770	10	2.5	6	18.306	30752	-7447.1	103000	32371	110000	-17283
Min.	370	-2	-5	3	9.1569	1188.6	-9779.8	-40772	-101000	-54044	-36153
Max.	870	10	5	6	18.355	34782	12567	125000	72674	168000	44438
Avg.	636.67	3.15	2.39	4.51	13.81	9904.05	-2551.10	11516.49	16291.49	34780.94	-6272.31

Table 2. Dataset after discussion with hydrodynamics experts

	Draft (mm)	Alpha (deg)	Beta (deg)	Vs (m/s)	C_L/C_D
0	570	0	-5	15.365	-0.905502
1	570	0	-5	18.336	-0.432223
2	870	-2	0	9.2151	-3.353348
3	870	-2	0	12.266	-2.240566
4	870	-2	0	15.329	-2.045144
...	...	...	...	...	...
...	...	...	...	...	...
229	570	10	2.5	18.278	4.736161
230	770	10	2.5	9.2208	2.771424
231	770	10	2.5	12.273	2.593646
232	770	10	2.5	15.332	2.806118
233	770	10	2.5	18.306	3.349376
Min.	370	-2	-5	9.1569	-4.3259
Max.	870	10	5	18.355	7.837481
Avg.	636.667	3.145	2.393	13.807	1.050

After discussion with hydrodynamics experts, an EDA with heatmap is performed to obtain the correlations between each input or control variable (Draft, Alpha, Beta, Vs) and the output or target variable (C_L/C_D), where the result of the heatmap is shown Figure 3. The negative value indicates that the two variables are negatively correlated, while the positive value represents a positive correlation. For instance, Alpha strongly correlates with C_L/C_D. Meanwhile, Draft and Vs show a strong enough correlation with C_L/C_D. A value close to zero shows that the two variables are not correlated or the dataset size is insufficient to establish a correlation, as is the case for Beta. Finally, the input variables used in training the DL model are Draft, Alpha, and Vs only, and the output variable is C_L/C_D.

Finally, the dataset obtained is pre-processed. Since the data is the result of a laboratory experiment, the data is validated, and it is unnecessary to perform cleaning on the data. Due to the limited number of data, i.e., 234 combinations of settings and their corresponding results of forces and moments, the experts are requested to determine the testing data containing the essential settings and their corresponding performances. After splitting, the data is 188 (76%) and 46 (24%) for the training and testing, respectively.

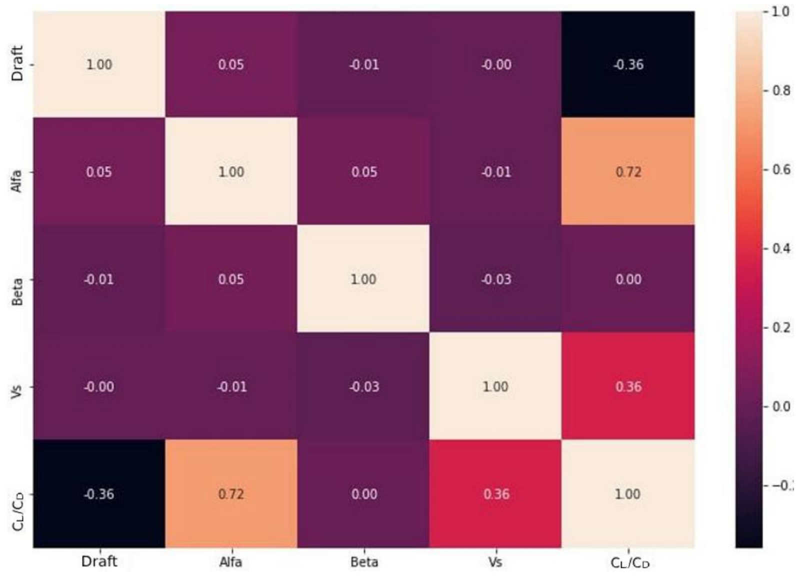


Figure 3. Heatmap for the correlations between input and output variables

3.2. Results of hyper-parameter optimization (Step 4)

In this step, the hyper-parameters used to train a DL model are optimized using GA, as shown in Algorithm 1. Based on the algorithm, some parameters need to be defined, i.e., Lines 2-6, where the parameters are defined as follows,

```
population_size   = 75
num_generations   = 3
mutation_rate     = 0.2
parameter_choices = {'nb_neurons': [10, 20, 30, 40, 50, 60, 70, 80],
                        'nb_layers': [3, 4, 5, 6, 7, 8],
                        'activation': ['relu', 'elu', 'tanh'],
                        'optimizer': ['adamax', 'adagrad', 'adadelta', 'adam']}
```

By using an exhaustive search, the `parameter_choices` will produce 8×6×3×4 (i.e., `nb_neurons` x `nb_layers` x `activation` x `optimizer`) = 576 combinations of hyper-parameters that need to be evaluated. Using GA employed in this study, only 3×75 (i.e., `num_generationsxpopulation_size`) = 225 possible hyper-parameter combinations. The three generations considered in the parameters are based on a comprehensive assessment of various factors. One significant consideration was that, even with fewer combinations, the GA exhibited a remarkable ability to generate solutions of comparable quality to those derived from exhaustive searches. Additionally, running a high number of generations can be computationally expensive, particularly in scenarios where resources are limited. By strategically minimizing the number of generations, a balance between computational efficiency and solution quality can be achieved. Furthermore, limiting the number of generations can mitigate the risk of overfitting or premature convergence, ensuring that the GA does not become excessively specialized to the training data.

The algorithm is performed in the Kaggle platform with a GPU P100 accelerator. The algorithm runs for 4,264.4 seconds, where the resulting hyper-parameters (including network architectures) and their associated MSE for each generation are shown in Figure 4. Since MSE is used for fitness function, the hyper-parameter with the lowest MSE is selected to train the DL model. The lowest, highest, and average of the resulting MSE in each generation are shown in Table 3, and the five best results with the lowest MSE in the last generation are shown in Table 4. From Table 3, it is shown that the average MSE decreases as the number of generations increases. However, the best individual (i.e., hyper-parameter with the lowest MSE) exhibits similar quality across each generation. Therefore, even with a limited number of generations, convergence in quality can be achieved, as the best individual remains consistent across increasing generations.

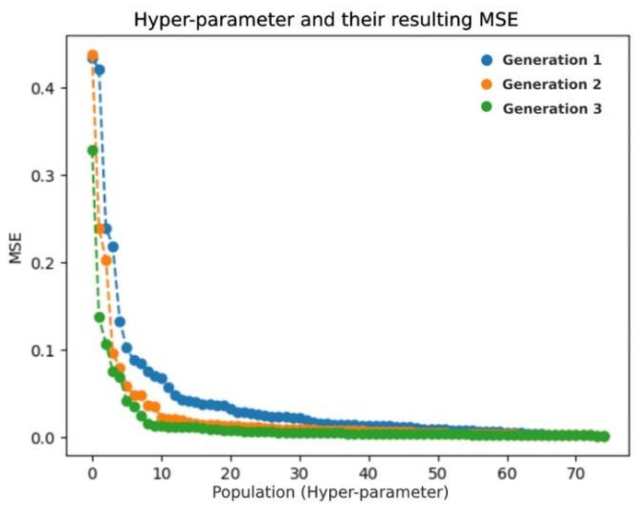


Figure 4. The resulting MSE calculated for each hyper-parameter in three generations of GA

Table 3. Resulting in MSE for each generation

Generation	Lowest MSE	Highest MSE	Average MSE	Best Individual (Hyper-parameters)
1st	0.00173	0.43306	0.03981	{'nb_neurons': 30, 'nb_layers': 5, 'activation': 'relu', 'optimizer': 'adagrad'}
2nd	0.00173	0.43757	0.02406	{'nb_neurons': 30, 'nb_layers': 5, 'activation': 'relu', 'optimizer': 'adagrad'}
3rd	0.00173	0.32887	0.01593	{'nb_neurons': 30, 'nb_layers': 5, 'activation': 'relu', 'optimizer': 'adagrad'}

Table 4. Results of hyper-parameters and networks with the lowest MSE in 3rd generation of GA

Hyper-parameters and network architectures	MSE
{'nb_neurons': 30, 'nb_layers': 5, 'activation': 'relu', 'optimizer': 'adagrad'}	0.001728
{'nb_neurons': 40, 'nb_layers': 5, 'activation': 'relu', 'optimizer': 'adagrad'}	0.002138
{'nb_neurons': 40, 'nb_layers': 6, 'activation': 'elu', 'optimizer': 'adagrad'}	0.002537
{'nb_neurons': 70, 'nb_layers': 6, 'activation': 'tanh', 'optimizer': 'adam'}	0.002577
{'nb_neurons': 60, 'nb_layers': 4, 'activation': 'relu', 'optimizer': 'adagrad'}	0.002597

3.3. Results of DL model training and evaluation (Steps 5-7)

The resulting hyper-parameters in step 4 are then used to train and optimize a DL model in steps 5-7 using the Keras library iteratively. These works are performed in the Kaggle platform with the P100 accelerator. In step 5, the resulting hyper-parameters are used as the parameters to train the DL model, where these parameters can affect the learning process and performance of the DL model. Before step 5 is performed, additional work is done to select the best number of epochs to improve the resulting hyper-parameters, where one epoch represents one complete round of learning from the training data. The results of the trained network with different numbers of epochs are shown in Figure 5, where the lowest MSE is 0.001536 and achieved when the epoch is set as 300. In step 6, the trained DL model is applied to predict C_L/C_D of the testing data, resulting in step 3, based on Draft, Alfa, and Vs. Then, the predicted C_L/C_D is compared to the original C_L/C_D of the testing data for the evaluation with R-squared metrics in step 7. These three steps, i.e., steps 5-7, are iterated for a specific number, where the trained DL model with the best R-squared and above a predefined threshold is selected as a result and stored as an H5 file. If the resulting DL model is unsatisfactory after several steps 5-7 iterations, then go back to step 4 and continue. In this research, a predefined threshold for the R-squared is 0.9. Notably, the DL model derived from 75 iterations exhibits the highest R-squared score of 0.9329, accompanied by an elapsed time of 11,977.9 seconds. The comparison of the original (actual) C_L/C_D and the predicted C_L/C_D of the testing data is shown in Figure 6.

Instances of various settings for the float design variables (h and α) and their corresponding C_L/C_D values, both the original and predicted, are shown in Figures 7(a) and 7(b). These graphs demonstrate how C_L/C_D resulted from the same h=570 mm, different α=2° and α=5°, and varying speeds from 9 to 18 m/s. These examples demonstrate the ability of DL model to predict the performance of float. This ability can be used to predict the performance of the float in certain conditions that cannot be tested in the laboratory due to lack of facilities and to optimize the design, represented in control variables, to achieve the desired target variables, i.e., performances.

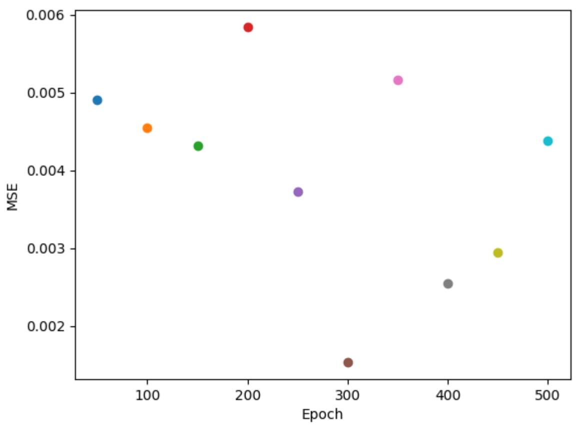


Figure 5. MSE of the optimized network that is trained with different numbers of epochs

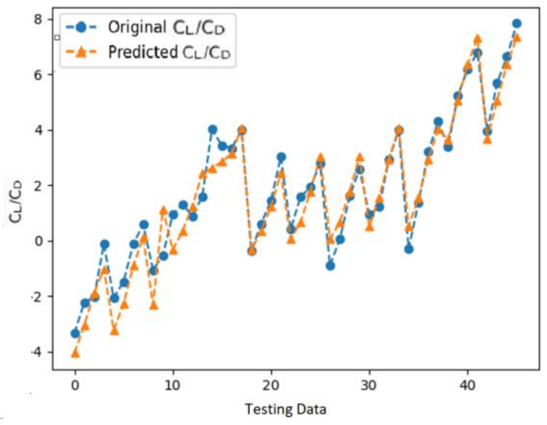
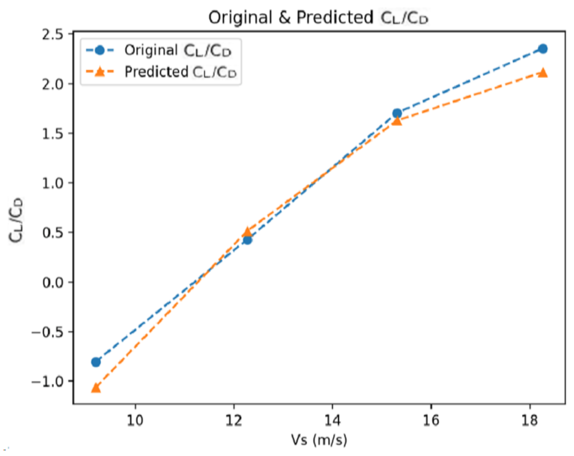
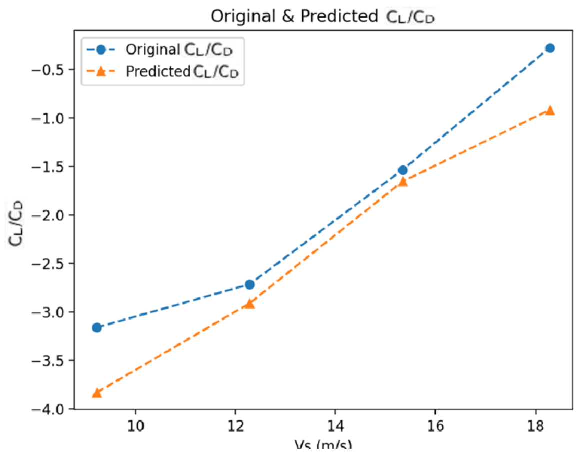


Figure 6. Comparison of the original C_L/C_D and the predicted C_L/C_D of the testing data



(a)



(b)

Figure 7. The examples of two essential settings with their original and predicted C_L/C_D on varying speed (m/s), where the first setting is (a) h=570mm and α=2°, and the second setting is (b) h=570mm and α=5°

4. CONCLUSION

This paper has demonstrated the successful application of GA and DL in predicting the performance of an engineering design. The DL model demonstrated its ability to predict the C_L/C_D ratio based on input variables (Draft, Alpha, Vs) and surpassed the predefined threshold of 0.9 for the R-squared. This result can have significant benefits when obtaining the actual performance of the design is costly, such as the C_L/C_D of float for a floatplane. This research also offers significant implications for the field of engineering design, as it can lead to the development of more efficient and practical designs. Besides, future research topics can be focused on optimizing control variables to obtain the desired performance of float in various designs, dimensions, and materials.

ACKNOWLEDGEMENT

The research conducted for this article received no financial support. We express our appreciation to the BRIN for their invaluable support on postdoctoral fellowship program within the research group of modeling of sustainable transportation. We are also grateful for the generous contribution of essential data by the Indonesian Hydrodynamic Laboratory, which enhanced this research. Moreover, we acknowledge Airlangga University for for creating a conducive research environment.

REFERENCES

[1] S. Syamsuar *et al.*, "Numerical simulation for floater design on the 17 passengers capacity of N219 amphibian in static and dynamic condition," in *AIP Conference Proceedings*, 2023, vol. 2646, no. 1, p. 050118, doi: 10.1063/5.0132289.

[2] M. G. Morabito, "A review of hydrodynamic design methods for seaplanes," *Journal of Ship Production and Design*, vol. 37, no. 3, pp. 159–180, 2021, doi: 10.5957/JSPD.11180039.

[3] J. Masri, L. Dala, and B. Huard, "A review of the analytical methods used for seaplanes' performance prediction," *Aircraft Engineering and Aerospace Technology*, vol. 91, no. 6, pp. 820–833, 2019, doi: 10.1108/AEAT-07-2018-0186.

[4] Erwandi, A. Aribowo, B. Sumantri, D. Rahuna, B. Ali, and M. Nasir, "The influence of sister keelsons on the hydrodynamic performance of N219A floats," in *AIP Conference Proceedings*, 2023, vol. 2941, no. 1.

[5] S. Carta, "Machine learning and computational design," *Ubiquity*, vol. 2020, no. May, pp. 1–10, May 2020, doi: 10.1145/3401842.

[6] O. Owoyele *et al.*, "Application of an automated machine learning-genetic algorithm (AutoML-GA) coupled with computational fluid dynamics simulations for rapid engine design optimization," *International Journal of Engine Research*, vol. 23, no. 9, pp. 1586–1601, Sep. 2022, doi: 10.1177/14680874211023466.

[7] I. Haryanto, T. S. Utomo, N. Sinaga, C. A. Rosalia, and A. P. Putra, "Optimization of maximum lift to drag ratio on airfoil design based on artificial neural network utilizing genetic algorithm," *Applied Mechanics and Materials*, vol. 493, pp. 123–128, Jan. 2014, doi: 10.4028/www.scientific.net/AMM.493.123.

[8] S. T. Atmaja, R. Fajar, S. Syamsuar, and S. Sutiyo, "Implementation of artificial neural network for predicting water drag of the aircraft floater," in *AIP Conference Proceedings*, 2023, vol. 2646, doi: 10.1063/5.0114019.

[9] S. Kasmaiee and M. Tadjfar, "Experimental study of the injection angle impact on the column waves: Wavelength, frequency and drop size," *Experimental Thermal and Fluid Science*, vol. 148, Oct. 2023, doi: 10.1016/j.expthermflusci.2023.110989.

[10] S. Kasmaiee, M. Tadjfar, and S. Kasmaiee, "Optimization of blowing jet performance on wind turbine airfoil under dynamic stall conditions using active machine learning and computational intelligence," *Arabian Journal for Science and Engineering,* pp. 1–25, Jun. 2023, doi: 10.1007/s13369-023-07892-9.

[11] M. Tadjfar, S. Kasmaiee, and S. Noori, "Continuous blowing jet flow control optimization in dynamic stall of NACA0012 airfoil," in *American Society of Mechanical Engineers, Fluids Engineering Division (Publication) FEDSM*, Jul. 2020, vol. 2, doi: 10.1115/FEDSM2020-20149.

[12] P. Josh and G. Adam, *Deep learning a practitioners approach*. O'Reilly Media, Inc., 2017.

[13] E. Wirsansky, *Hands-on genetic algorithms with Python: Applying genetic algorithms to solve real-world deep learning and artificial intelligence problems*. Packt Publishing Ltd, 2020.

[14] M. Tadjfar, S. Kasmaiee, and S. Noori, "Optimization of naca 0012 airfoil performance in dynamics stall using continuous suction jet," in *American Society of Mechanical Engineers, Fluids Engineering Division (Publication) FEDSM*, Jul. 2020, vol. 2, doi: 10.1115/FEDSM2020-20147.

[15] C. C. Aggarwal, *Neural networks and deep learning*, vol. 10, no. 978. Cham: Springer International Publishing, 2018.

[16] F. Cady, *The data science handbook*. Wiley, 2017.

[17] F. Hidiyanto, S. Leksono, R. Fajar, and S. T. Atmaja, "Data exploratory analysis and feature selection of low-speed wind tunnel data for predicting force and moment of aircraft," *Majalah Ilmiah Pengkajian Industri*, vol. 16, no. 2, pp. 87–94, Sep. 2023, doi: 10.29122/mipi.v16i2.5285.

[18] A. Zheng and A. Casari, "Feature engineering for machine learning: Principles and techniques for data scientists," *O'Reilly*, p. 218, 2018.

[19] G. Miner, R. Nisbet, and J. Elder, *Handbook of statistical analysis and data mining applications*. Elsevier, 2009.

[20] F. Provost and T. Fawcett, *Data science for business what you Need to know about data mining and data-analytic thinking*. 2013.

[21] M. Kuhn and K. Johnson, *Feature engineering and selection: A practical approach for predictive models*. Chapman and Hall/CRC, 2019.

[22] M. Kuhn and K. Johnson, *Applied predictive modeling*. New York, NY: Springer New York, 2013.

[23] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[24] J.-H. Han, D.-J. Choi, S.-U. Park, and S.-K. Hong, "Hyperparameter optimization using a genetic algorithm considering verification time in a convolutional neural network," *Journal of Electrical Engineering & Technology*, vol. 15, no. 2, pp. 721–726, Mar. 2020, doi: 10.1007/s42835-020-00343-7.

[25] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: Past, present, and future," *Multimedia Tools and Applications*, vol. 80, no. 5, pp. 8091–8126, Feb. 2021, doi: 10.1007/s11042-020-10139-6.

[26] M. H. Abed and M. N. Mohmad Kahar, "Hybridizing genetic algorithm and single-based metaheuristics to solve unrelated parallel machine scheduling problem with scarce resources," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 12, no. 1, p. 315, Mar. 2023, doi: 10.11591/ijai.v12.i1.pp315-327.

[27] Y. Tian and Y. Zhang, "A comprehensive survey on regularization strategies in machine learning," *Information Fusion*, vol. 80, pp. 146–166, Apr. 2022, doi: 10.1016/j.inffus.2021.11.005.

[28] A. Mathew, P. Amudha, and S. Sivakumari, "Deep learning techniques: An overview," in *Advances in Intelligent Systems and Computing*, vol. 1141, 2021, pp. 599–608.

[29] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*, vol. 53, no. 9. Cambridge University Press, 2014.

[30] Aurélien Géron, "Hands-on machine learning with Scikit-Learn, Keras and TensorFlow: Concepts, tools, and techniques to build intelligent systems," *O'Reilly Media*, p. 851, 2019.

[31] L. D. Schroeder, D. L. Sjoquist, and P. E. Stephan, *Understanding regression analysis: An introductory guide*. 2455 Teller Road, Thousand Oaks California 91320: SAGE Publications, Inc, 2017.

BIOGRAPHIES OF AUTHORS

Faisal Fahmi received an M.Sc. and Ph.D. in Electronic Engineering and Computer Science from National Chiao-Tung University, Hsinchu City, Taiwan. He is currently a lecturer at Airlangga University, Surabaya, Indonesia. His research interests include software engineering, microservices and service-oriented architecture, information development, and machine learning. He can be contacted at email: faisalfahmi@fisip.unair.ac.id and fais009@brin.go.id.

Rizqon Fajar received an M.Sc. in Chemical Engineering from Delft Technical University, Netherlands, and a Dr. in Mechanical Engineering from the University of Indonesia, Indonesia. He is a senior research engineer in the Research Center for Transportation Technology, National Research and Innovation Agency, South Tangerang, Indonesia. His research interests include Fuel, Combustion, Machine Learning, and Data Science. He can be contacted at email: rizq001@brin.go.id.

Sigit Tri Atmaja holds a Master of Engineering (M.T.) from the University of Indonesia. He is an expert in Intelligent Control Systems, the Internet of Things, and Machine Learning in the Research Center for Transportation Technology, National Research and Innovation Agency, South Tangerang, Indonesia. He can be contacted at email: sigi015@brin.go.id.

Erwandi received an M.Eng. and Ph.D. in naval architecture and global architecture from Osaka University, Japan, respectively. He is a senior research engineer in the Research Center for Hydrodynamics Technology, National Research and Innovation Agency, Surabaya, Indonesia. His research interests include hydrodynamics, floatplane design, and submarine. He can be contacted at email: erwa001@brin.go.id.

Daif Rahuna received a Master of Engineering (M.T.) in Ocean Engineering from Sepuluh Nopember Institute of Technology, Surabaya, Indonesia. He is a research engineer in the Research Center for Hydrodynamics Technology, National Research and Innovation Agency, Surabaya, Indonesia. His research interests include hydrodynamics, ocean power plants, and hydro turbines. He can be contacted at daif001@brin.go.id.