An Integrated Approach for Design Optimization in AI-Driven CAE

Younju Kim[a], Dongwook Yang[a,1], Sungil Jang[a]

[a]Hyundai Mobis, Teheran-ro 203, Seoul 06141, Korea

Abstract

Prior to actual testing, performance evaluation is conducted through Computer-Aided Engineering (CAE) in design process. The analysis results are scrutinized to verify whether the design meets the intended performance. If not satisfied, the conditions are modified, and the analysis is iteratively performed until the desired performance is achieved. However, this iterative process poses challenges in terms of both time and cost.

To address this, machine learning and deep learning methodologies have been developed. Nevertheless, some of the existing models focus on identifying a single optimal solution for a given performance. This focus may overlook the potential to discover diverse sets of design parameters that can achieve the same performance criteria. In this study, we propose a method that aims to efficiently identify multiple design parameters that satisfy the desired performance with high accuracy, thereby reducing the need for iterative processes.

We modify the model to predict various design parameters by applying Monte Carlo dropout and Bayesian neural network to the tandem network, which outputs only one design parameter fitted to the training data. Furthermore, we propose a 2-stage methodology for exploring local minima by performing bayes optimization based on multiple candidate values derived from the tandem network. The proposed model significantly reduces the number of iterations required for design optimization and predicts multiple possible design parameters. Experimental results using data from rear-wheel steering and braking simulations demonstrate the overwhelming performance and diverse design parameters provided by our proposed model.

Keywords

Computer-Aided Engineering, Deep Learning, Tandem Network, Bayesian Optimization, Simulation, Design Optimization

1 Introduction

Computer-Aided Engineering (CAE) refers to the use of computer simulations to evaluate performance and solve engineering problems. The CAE process involves inputting design parameters and conditions, performing analysis using computer programs, and verifying the results. If the analysis results do not meet the target performance, the conditions are adjusted, and the analysis is repeated. This iterative process continues until the design parameters that satisfy the target performance are identified.

The developed algorithm is capable of suggesting the optimal design value that satisfies all target values in cases where there are multiple desired performance targets, and each performance target value must be satisfied. The optimal value can be identified through the utilisation of existing data, obviating the necessity for additional calculations for each performance. In existing optimization

[1]Corresponding author.
E-mail address: yang_dw@mobis.com (D. Yang)
https://doi.org/10.59972/j92qm86n

analysis programs, each analysis must be performed separately, depending on the target performance and the type of analysis. Furthermore, a parameter that satisfies one performance criterion in a particular analysis may result in poor performance in another analysis. Therefore, another analysis must be performed to satisfy multiple performance criteria, even if it has previously satisfied the initial performance.

This research aims to predict optimal design parameters that satisfy desired multi-performance criteria. By deriving design factors that meet target performance based on previously conducted analysis data, the number of repeated analyses for parameter tuning can be substantially diminished. To achieve this, we propose a model that derives optimal design parameters satisfying multi-performance criteria via deep learning. Deep learning has recently been successfully applied to various fields, including computer vision, natural language processing, voice processing, signal processing, medical data, financial data, and manufacturing data. However, its application in the field of engineering analysis remains limited. Predicting design parameters that meet target performance within a short timeframe is particularly challenging. Unlike general problems, where the solution is often one-to-one, the design values satisfying multi-performance criteria exhibit a one-to-many relationship. Therefore, we suggest an appropriate solution using the Tandem Neural Network (TNN) [1], leveraging deep learning to predict optimal design parameters effectively.

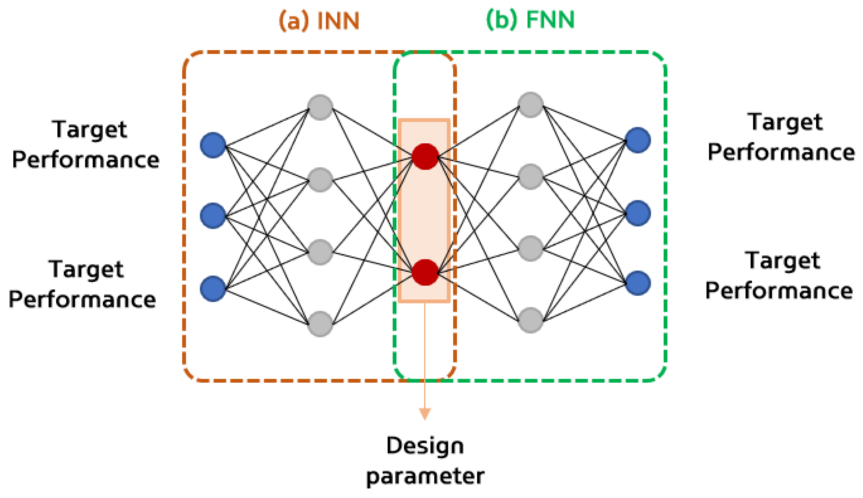

Figure 1. Structure of Tandem Neural Network: a) Inverse Neural Network, b) Forward Neural Network.

As seen in Figure 1, the Tandem Neural Network is composed of a combination of two networks: Forward Neural Network (FNN) and Inverse Neural Network (INN). The first network is the FNN, which predicts performance based on design parameters. While high-performance machine learning models such as tree-based methods, Support Vector Regression (SVR), or K-Nearest Neighbors (KNN) can be used for regression predictions, we adopt a deep learning model with standard layers that functions similarly to traditional analysis programs. The FNN is pre-trained before being integrated with the INN, and the weights of the FNN are frozen during the training of the TNN. The second network is the Inverse Neural Network (INN), which is connected upstream of the FNN. The INN is responsible for predicting design parameters when the target performance is provided as input. The INN and FNN are sequentially combined (forming the TNN), and the training is carried out to ensure that the same input and output performance are produced across the entire network. We then adopt the intermediate values of the TNN (the output values of the INN) as the predicted design parameters.

However, the TNN approach has several drawbacks in analysis. One major issue is that TNN outputs a unique design value that is specifically tailored to the training data, even though multiple design values could satisfy the same performance criteria (one-to-many relationship). This limitation prevents the exploration of potential improvements that might yield better results beyond the training data. Additionally, if the trained TNN provides poor results, it becomes challenging for analysts to derive new, improved outcomes. In such cases, the TNN requires retraining to achieve new results. However, without changes in the data, model hyperparameters need to be adjusted, complicating the process.

Our proposed model effectively addresses the aforementioned issues through a two-stage approach. The first stage combines Monte Carlo dropout (MC dropout) [2] and Bayesian neural networks (BNN) [3] with the TNN. These techniques introduce uncertainty into the model, ensuring a variety of predicted values. However, since only a small amount of uncertainty is added to preserve the model's performance, the predicted values do not fluctuate drastically. By varying the initial weights of the model to create multiple models and employing them in an ensemble, we achieve global optimization by obtaining a range of candidate predicted values. In the second stage, we identify the optimal parameters from the obtained candidate predictions and perform local optimization. Bayesian Optimization (BO) [4] has been proven effective in finding the optimal value that satisfies the objective function among multiple candidates, and we adopt this approach.

An experiment using two simulation datasets from our laboratory demonstrates the effectiveness of proposed model. It showcases superior performance compared to traditional machine learning and deep learning techniques and outperforms individual TNN or BO approaches. Furthermore, our model enables the generation of multiple design parameters that satisfy specific performance criteria, enhancing the amount of information available to users due to the high variability in predicted design parameters.

2 Materials and Methods

This section provides a detailed description of two simulation datasets collected through CAE and proposes a methodology that combines TNN and BO.

2.1 Materials

We conducted simulations using dynamic analysis programs Carmaker and Simulink for Rear Wheel Steering (RWS) and Anti-lock Brake System (ABS). RWS is a technology that actively controls the rear wheel steering angle based on the front wheel steering angle to enhance vehicle stability and maneuverability according to driving conditions. ABS prevents wheel lock phenomena that can occur during sudden braking or on low-friction coefficient roads.

Based on these analytical data, we propose a multi-design optimization methodology using deep learning. We conducted experiments using Python and TensorFlow version 2.5, leveraging four Nvidia V100 GPUs.

2.1.1 Rear Wheel Steering

Exemplary tests to evaluate RWS performance include Double Lane Change (DLC), Ramp, Circle turning, and Slalom tests. This research focuses on DLC and Slalom tests. DLC assesses vehicle safety and agility during evasive steering and lane changes, where the vehicle returns to its original lane after changing lanes. Slalom evaluates cornering performance and vehicle safety through rapid steering around evenly spaced traffic cones. The goal is to define target performance criteria for both tests and identify design parameters that can simultaneously satisfy these criteria.

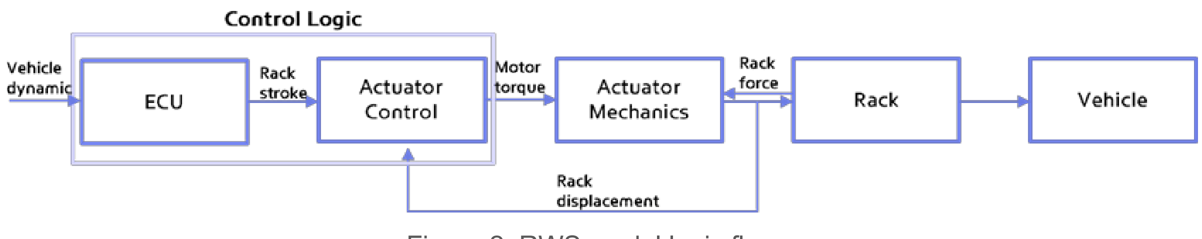

Figure 2. RWS model logic flow.

RWS technology is illustrated in Figure 2. Seven primary design parameters comprising the Control Logic in Figure 2 are identified as design tuning factors, with each factor utilized to generate the target yaw-rate and motor torque values of the Control Logic. Figure 3, 4 depicts the representative target performance values derived from the two tests. In the DLC test, four peak factors are selected from the avoidance, correction, and exit sectors, which contribute to the yaw rate result values. Meanwhile, for the Slalom test, three factors—maneuver time, steering angle, and deviation distance—are chosen. Using the seven obtained target performance indicators as inputs, we employed AI with deep learning applications to derive the corresponding seven design parameters as outputs.

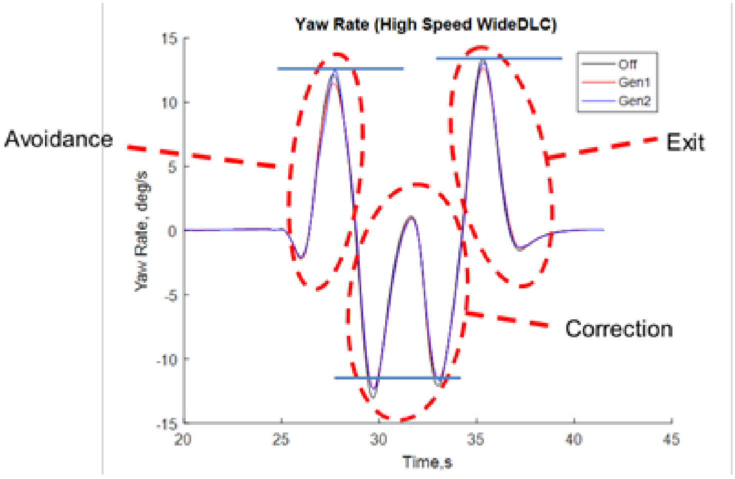

Figure 3. Yaw rate for DLC yaw rate (deg/s).

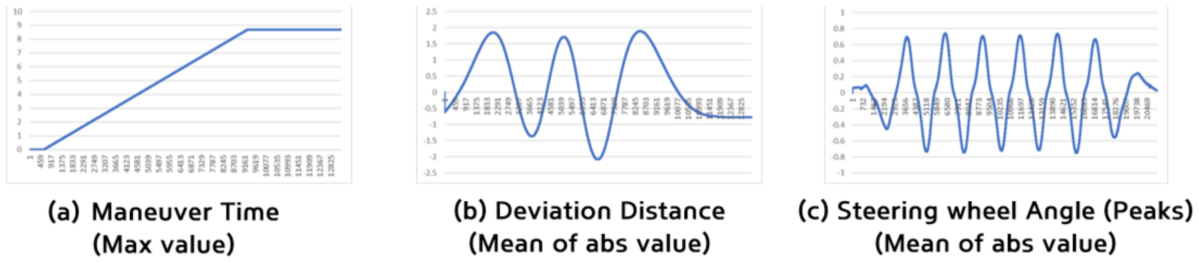

Figure 4. Three performances for SLALOM test.

2.1.2 Anti-lock Brake System

ABS design tuning parameters encompass 12 significant factors that comprise the ABS model within the Hydraulic Electronic Control Unit (HECU) Model in Loop Simulation (MILS) model. Each parameter influences the creation of target wheel brake pressure values within the HECU. Among the evaluation metrics including wheel velocity deceleration difference, brake distance, and wheel slip, which assess the performance of the ABS MILS model in each test, this study selects wheel slip as the representative target performance value.

In Table 1, we present ABS simulation maneuver cases within the scope that can be disclosed. For each of the four road conditions, there are selectable combinations of vehicle velocity, brake pressure, and gear. For instance, on dry asphalt, vehicle velocities of 50 and 100 can be chosen, and brake pressures of 210 and 150 are available. However, not all combinations are possible; in some conditions, only one brake pressure may be applicable for a given vehicle velocity. We conduct simulations for a total of 7 scenarios with combinations of road type, velocity, pressure, and gear, using two wheel slip values as the target performance values for each scenario.

Consequently, the analysis is conducted with 12 design parameters as outputs to satisfy 14 input target performance indicators.

Table 1. ABS simulation maneuver cases.

	Road	Dry asphalt	Wet asphalt	Wet basalt	Wet ceramic
Vehicle Velocity (km/h)	1	50	50	50	50
	2	100	100	80	80
	3	–	–	120	90
	4	–	–	–	100
Brake Pressure (bar)	1	210	210	210	210
	2	150	150	150	150
	3	–	70	70	70
Gear select	1	N	N	N	N
	2	D	D	–	–

2.2 Methods

The framework for proposed model is shown in Figure 5. To effectively address the problem in a given multi-design scenario, we construct a 2-stage model combining TNN and BO. We modify the general TNN into a model that can account for uncertainty, predicting multiple design parameters (global optimization). Subsequently, BO performs local optimization on the parameters considering uncertainty to obtain the optimal parameters.

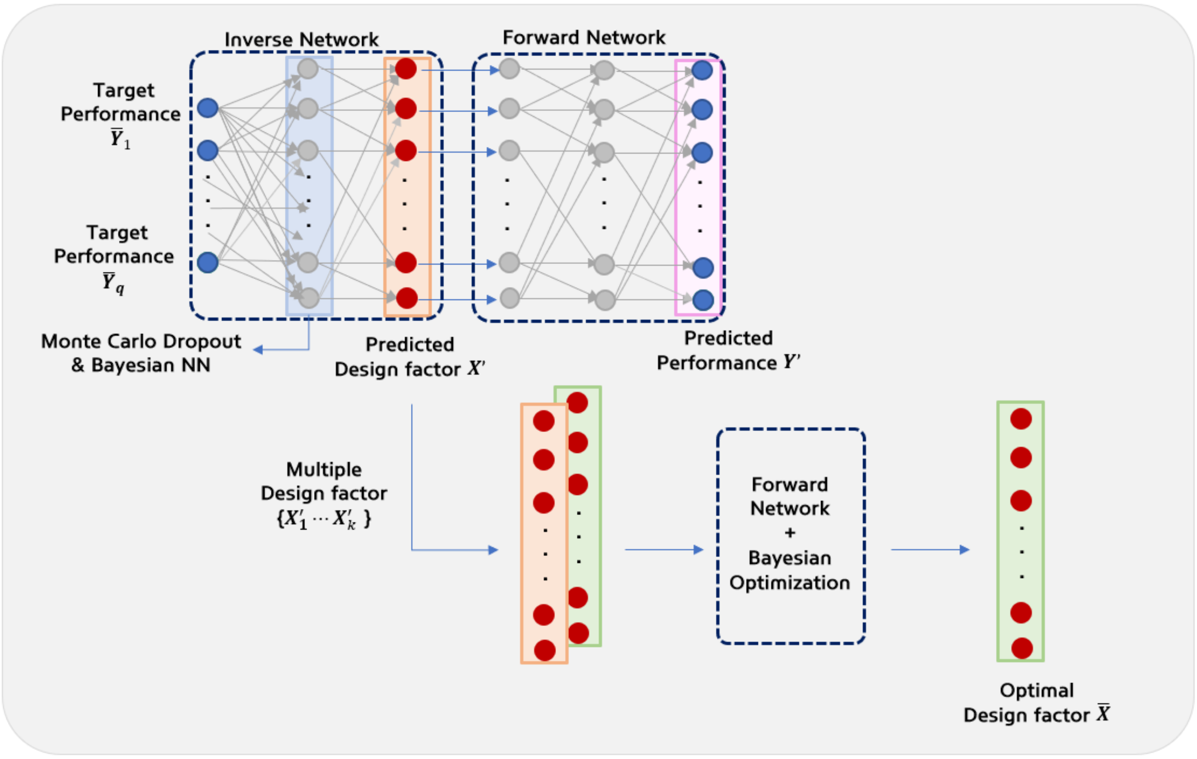
Inverse Network
Forward Network
Target Performance $\bar{Y}_1$
Target Performance $\bar{Y}_q$
Monte Carlo Dropout & Bayesian NN
Predicted Design factor X'
Predicted Performance Y'
Multiple Design factor $\{X'_1 \cdots X'_k\}$
Forward Network + Bayesian Optimization
Optimal Design factor $\bar{X}$

Figure 5. The overall framework of the proposed model.

We define the design factors as $X = [X_1, \ldots, X_n]$ and their corresponding performances as $Y = [Y_1, \ldots, Y_n]$, where n denotes the number of data points. The objective is to determine the optimal design factor $\bar{X}$ given a specified target performance $\bar{Y}$ by the user. This task necessitates the development of a model that accommodates a one-to-many relationship between inputs (performance) and outputs (design factors), moving beyond the conventional one-to-one mapping. We construct an integrated predictive model tailored to this relationship by combining TNN and BO.

The 1-stage TNN model, combining INN and FNN, functions to output candidates for design factors. As the conventional TNN model proposed in [1] retains the same weights after training, leading to repetitive output of identical results, we incorporate MC dropout and BNN (Bayesian Neural Networks) techniques to enhance prediction diversity and account for uncertainty. In the 2-stage, BO (Bayesian Optimization) is utilized to select the optimal design parameter from the candidates. Detailed discussions of these methodologies are presented in Sections 2.2.1 and 2.2.2.

To ensure reliability in our results, we strategically split data for use. For the training and validation of all networks within the TNN, we allocate 90% of the total data. The remaining 10%, designated as test data, is not involved in the training process at all. This test data is exclusively used for performance evaluation, which is discussed in detail in Section 3 and Section 4.

2.2.1 1-stage: Tandem Neural Network

TNN serves the function of outputting a set of candidates predicted design parameters, denoted as X'. One of the components of TNN, the FNN performs a role similar to an analytical simulator, where the input is the design parameter X, and the output is the performance Y. Since FNN is also utilized during the subsequent BO process, it is imperative that it produces reliable results.

Once the training of the FNN network is completed, it is configured to be frozen, ensuring that it solely performs the function from $X \rightarrow Y$ without further involvement in the training of the TNN. The INN, named at the front of the FNN structure, can be added to form a TNN with a $\boldsymbol{Y} \rightarrow \boldsymbol{X} \rightarrow \boldsymbol{Y}$ structure similar to an Auto-Encoder (AE) [5]. The loss function of the TNN utilizes the Mean Squared Error (MSE) formula, as shown in Equation (1).

$$Loss = MSE(\boldsymbol{Y}, \boldsymbol{Y}') = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{Y}_i - \boldsymbol{Y}'_i)^2, \tag{1}$$

When a target performance $\boldsymbol{Y}$ is input into a fully trained TNN, predicted design factors $\boldsymbol{X}'$ can be obtained from the intermediate layer, which includes the output layer of the INN and the input layer of the FNN. However, once the TNN is fully trained, its weights lack uncertainty and remain fixed, thus producing the same $\boldsymbol{X}'$ for identical target performances each time. This results in the elimination of the possibility for other design parameters that could produce the same target performance, confirming only outcomes that fit the training data. To address this issue, we modify the model structure to incorporate uncertainty, enabling the prediction of different outcomes with each input.

MC dropout is similar to general dropout, which is used to reduce overfitting. However, while general dropout is only applied during the training phase and not during inference, MC dropout is used during inference as well to reduce overfitting and introduce uncertainty. We incorporate MC dropout at random locations within the existing TNN layers during training, allowing the model to produce different outputs each time.

BNN is a type of neural network that uses a probabilistic approach to model uncertainty. Unlike conventional neural networks that use fixed values for weights and biases, BNN treat these parameters as probabilistic distributions, thereby incorporating uncertainty. It is known to perform better than other models that merely fit to training data, especially in situations with relatively sparse data. Given the domain of our analytic data, where obtaining big data may not be feasible due to the varying speeds of analytic programs, BNNs are particularly suitable for such scenarios.

Although MC dropout and BNN are not used simultaneously, each is integrated into the Tandem network separately, allowing for varied results during inference Additionally, to further increase the diversity of the learned weights (or weight distributions) based on initial weight settings, we employ an ensemble approach by constructing multiple Tandem networks, each enhanced with either MC dropout or BNN, thereby amplifying the randomness. Upon entering the target performance $\overline{\boldsymbol{Y}}$, a set of predicted design factor candidates $\{\boldsymbol{X}_{\boldsymbol{1}}^{'} \cdots \boldsymbol{X}_{k}^{'}\}$ is output, where k represents the total number of candidates. In the second stage, we use BO to acquire the optimal design parameter $\overline{\boldsymbol{X}}$.

2.2.2 2-stage: Bayesian optimization

Bayesian Optimization is a methodology used to efficiently optimize computationally intensive objective functions, particularly when the actual values of these functions are unknown. It is highly effective for tuning hyperparameters in machine learning or deep learning. This approach calculates the next set of hyperparameters to try by utilizing 'prior knowledge' gathered from previous explorations. The surrogate model used in this process is a probabilistic objective function estimator, which is trained on pairs of input values and objective function values (in this case, MSE). The acquisition function then uses the probabilistic estimates to determine where to experiment next, measures the objective function values at the proposed location, and updates the model accordingly.

$$m(\boldsymbol{X}) = \frac{\sum_{k=1}^{n_{tree}} r_k(\boldsymbol{X})}{n_{tree}} \tag{2}$$

$$\sigma(\boldsymbol{X}) = \sqrt{\frac{\sum_{k=1}^{n_{tree}}\left(r_k(\boldsymbol{X}) - m(\boldsymbol{X})\right)^2}{n_{tree}}} \tag{3}$$

Hwang et al [4] demonstrated significant results using Bayesian Optimization alone in similar data scenarios. They used a random forest as the surrogate model to obtain the mean and standard deviation of the objective function based on Equations (2) and (3). Here, n_{tree} refers to the number of trees in the random forest, and $r_k(\cdot)$ represents the k-th tree. The mean and standard deviation are used in the acquisition function, the Moment-Generating Function of the Improvement (MGFI) function as Equation (4). According to Wang et al [6], the MGFI acquisition function simultaneously considers 'exploration' of the point with maximum variance and 'exploitation' of the point where the objective function is minimized. This formula is detailed in Equation (4).

$$MGFI(\boldsymbol{X},t) = \Phi\left(\frac{f_{min} - m(\boldsymbol{X}) + \sigma^2(\boldsymbol{X})t}{s(\boldsymbol{X})}\right) exp\left((f_{min} - m(\boldsymbol{X}) - 1)t + \frac{\sigma^2(\boldsymbol{X})t^2}{2}\right) \tag{4}$$

In this context, f_{min} represents the minimum value among the objective function values, and t is the balance parameter between exploration and exploitation. A higher t value emphasizes exploration, while a lower t value focuses on exploitation. The optimization is carried out to maximize the value of Equation (4).

However, when applying Bayesian optimization to our problem, a significant drawback arises due to the random selection of the initial input set and its subsequent use in optimization, leading to large variations in model performance. Moreover, the convergence time during optimization is uncertain, and if the initial values are not favorable, it could take a long time to reach a local minimum. To address these drawbacks, we are obtaining initial values through the TNN in the 1-stage.

In summary, in the 1-stage, we obtain candidates for design parameters from multiple TNN, which include MC dropout or BNN structures. These candidates are then inputted into a FNN, from which performance outcomes are used to calculate the MSE (objective function). Using the design parameters and MSE, we train surrogate model. Subsequently, we select the next sampling points based on the MGFI function, adding these points to the design parameter and MSE pairs. This iterative process gradually optimizes until we obtain the optimal design parameter $\overline{\boldsymbol{X}}$.

3 Results and Discussion

We collect two types of simulation data (RWS & ABS) used in Hyundai Mobis Company research facilities to conduct comparative experiments on prediction performance. To ensure the reliability of the experiments, we designate 10% of the total data as test data, which is used solely for testing and validating model performance. In Section 4, we carry out the model's verification and validation processes based on the test data.

Performance comparison with the proposed model involves four representative machine learning algorithms (Multiple Linear Regression (MLR), Support Vector Regression (SVR), Random Forest, XGBoost (eXtreme Gradient Boosting)) commonly used in multi regression and a simple deep learning model composed of dense layers (Deep Neural Network (DNN)). In addition, not only the general TNN but also models that combine MC dropout and BNN with TNN, which generate multiple candidates, are utilized. Since models that produce multiple candidates yield different results each time, experiments are conducted using the average of 10 inference outcomes. Moreover, general BO, which starts with random initial values, and cINN [7], which has shown high performance in recent related studies, are also compared.

The proposed model utilizes an appropriate prediction time that can be applied in practice, as determined in consultation with the simulation task managers. All comparison models and the proposed model take the same analytical performance as input and predict the design parameters accordingly. Since there are multiple viable design parameters that satisfy performance criteria, directly comparing design parameters is not a correct approach. Therefore, for accurate comparison, the design parameters are fed into a fully trained FNN, and the evaluation metrics are derived based on the comparison between the predicted performance and the actual performance.

The proposed model is likely to achieve higher performance by extracting more candidates from the TNN, a hypothesis that can be verified in the third experiment. This experiment will demonstrate a comparison of results based on key hyper-parameters of the proposed model using RWS data.

3.1 Experiment - Rear Wheel Steering

Table 2 presents the experimental results for various models on RWS data. The FNN, used in both TNN and BO and also for performance comparison of design parameters, is selected through hyperparameter tuning, achieving R^2 $score$ of 0.98 and MSE of 0.00026. The optimization methods included in BO and the proposed model adjust the prediction time per parameter, setting the time to predict a single design parameter at approximately 30 seconds, in accordance with worker requests.

Table 2. Model evaluation comparison for RWS data.

Model	$R^2\ score\ (\times 10^{-2})$	$MSE\ (\times 10^{-4})$	$RMSE\ (\times 10^{-2})$	$MAE\ (\times 10^{-2})$
MLR	42.32	24.25	4.79	2.66
SVR	88.12	10.00	2.79	1.90
Random Forest	94.67	5.70	1.84	0.96
XGBoost	89.54	6.58	2.28	1.06
DNN	93.12	5.22	1.98	0.83
TNN	96.05	2.20	1.31	0.66
TNN + MC dropout	96.58	1.69	1.24	0.72
TNN + BNN	81.14	12.02	3.18	2.19
Bayes optimization	96.87	1.74	1.18	0.57
cINN	94.26	6.11	1.86	0.93
Proposed method	98.14	0.71	0.81	0.39

As seen in Table 2, the Random Forest model exhibited the best results among the machine learning models, with the DNN showing similar outcomes. These baseline models use the performance obtained from simulations as input values and design parameters as output values, the results of which are then applied to evaluate the FNN model. BO, which sets initial values randomly, demonstrates significantly lower $RMSE$ and MAE compared to other models (indicated by the blue). While the cINN shows better performance than machine learning models, it performs slightly lower than the TNN-based models or BO that are specifically developed to effectively solve these problems.

Among the three TNN-based methods, those employing MC dropout and the pure TNN both show good results across four metrics. This indicates the superior performance of the basic TNN. In particular, TNN with MC dropout demonstrates high performance, likely due to dropout introducing randomness in the prediction phase and reducing overfitting. The Bayesian neural network structure of TNN, while exhibiting somewhat lower performance due to its high randomness, allows for exploring a broader range of data. Therefore, it is used as an initial model in our proposed model along with TNN applying MC dropout.

Our proposed model demonstrates the best results across four metrics. Although the high $R^2\ score$ of existing models meant that significant improvements are limited, we observe excellent results in terms of MSE, $RMSE$, and MAE. This underscores the importance of initial values in the Bayes optimization process and confirms that TNN is well-suited for this role.

Figure 6 shows a correlation plot comparing the actual performance (ground truth) with the predicted performance obtained by inputting the design parameters, extracted through our proposed model based on RWS performance inputs, into the FNN. The x-axis represents the ground truth, while the y-axis represents the predicted performance derived from the predicted design parameters using the FNN. The results indicate that all variables fit well to the y=x line, suggesting a high correlation between the predicted and actual performances. Instances where the data points do not fit the y=x line mostly occur in situations involving the same data, which can be interpreted as an issue with the data itself. Future models that can account for discrete variables rather than continuous ones are planned for development.

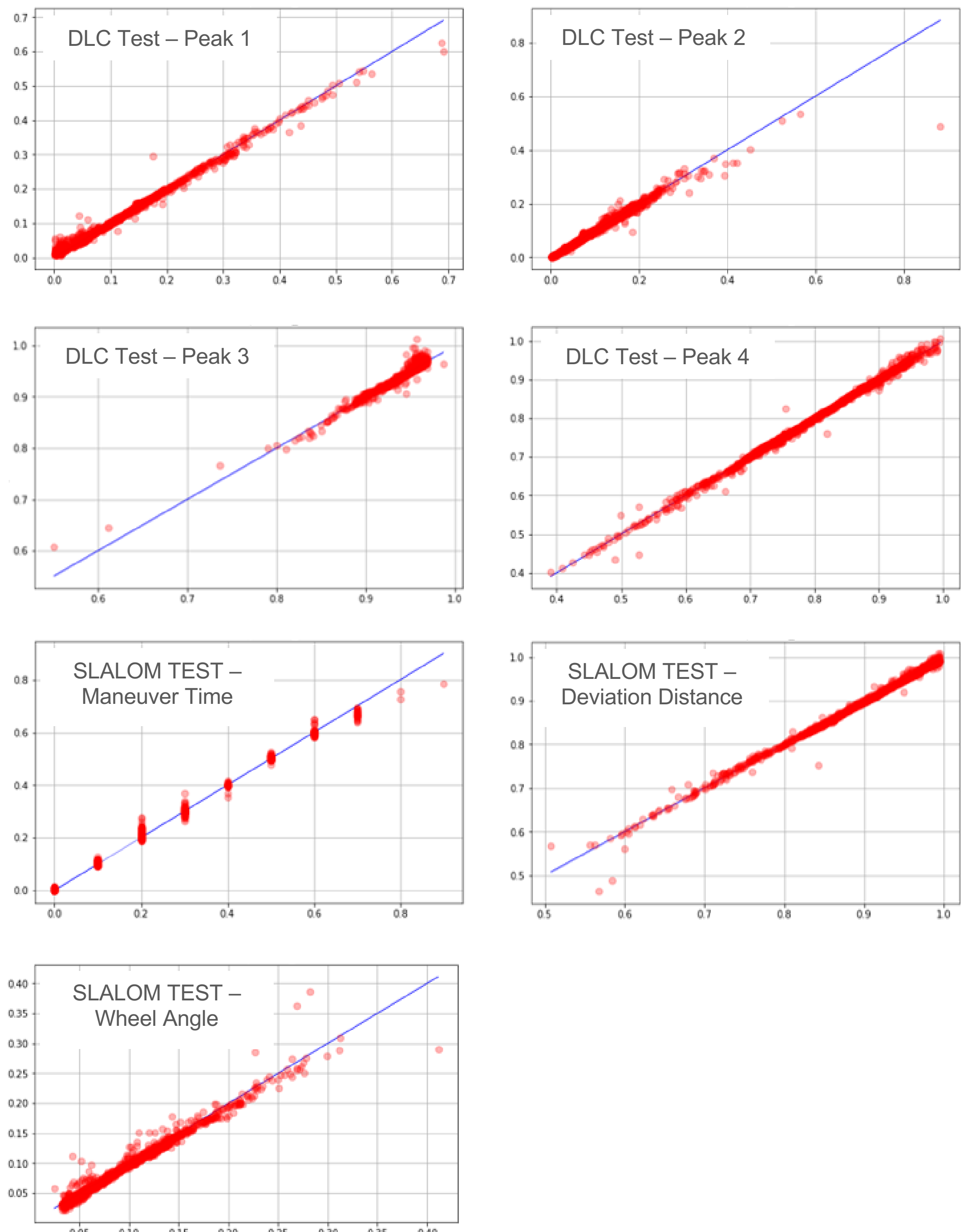

Figure 6. Correlation plot of RWS ground truth performance (x-axis) and the performance when predicted design parameters are input into the FNN (y-axis).

3.2 Experiment - Anti-lock Brake System

To demonstrate the efficacy of our model, we validate its performance on ABS data. We use the same models applied to RWS data for a comparative analysis. In these experiments, we limit the prediction speed (per data) of the proposed model to approximately 30 seconds, despite the time-consuming nature of the Bayesian optimization process involved in optimization. The Forward network employed achieves R^2 $score$ of 0.8699 and MSE of 0.0073.

Table 3. Model evaluation comparison for ABS data.

Model	R^2 $score$ $(\times 10^{-2})$	MSE $(\times 10^{-4})$	$RMSE$ $(\times 10^{-2})$	MAE $(\times 10^{-2})$
MLR	46.64	348.66	17.34	11.38
SVR	70.74	182.39	12.68	8.06
Random Forest	60.70	205.50	13.76	8.82
XGBoost	60.97	186.16	13.25	8.39
DNN	62.88	209.12	13.78	8.82
TNN	90.72	38.72	6.08	4.11
TNN + MC dropout	90.03	47.45	6.71	4.54
TNN + BNN	80.99	87.84	9.17	6.34
Bayes optimization	90.10	45.93	6.63	4.55
cINN	70.78	146.85	12.12	7.63
Proposed method	92.97	31.52	5.47	3.70

The performance of all four representative machine learning models are measured to be somewhat low. Among them, the SVR, which exhibits the highest performance, still shows poorer results compared to models based on TNN. Likewise, the DNN, which consists solely of dense layers, also demonstrates low performance.

The models based on TNN and Bayesian optimization applied in the proposed model record significantly higher performance compared to other models. Notably, the proposed model achieves the highest performance with all metrics. It demonstrates improved performance over the TNN-based models and Bayesian optimization used in the model, similar to what was observed with the RWS data. This indicates that our model is more effective compared to other models.

3.3 Discussion

In this section, we discuss the limitations of our model and explore the types of decisions that can be made based on proposed model.

The most significant limitation of the proposed model is its heavy reliance on the FNN. An FNN is used at every stage of the model, implying that a high performance of the FNN is essential. We have ensured the high performance of the FNN through extensive experimentation with two datasets; however, if the FNN performs poorly in other domains, the use of the proposed model becomes infeasible. Nevertheless, since most design data typically acquire performance through simulation programs, the role of the FNN can be seen as substituting for these programs, allowing us to generally expect high performance.

In addition, the current model inherently performs continuous predictions for discrete data. Typically, in basic $X -> Y$ scenarios, it is straightforward to construct models separately for discrete and continuous Y, but our model cannot employ this approach as it uses intermediate layer outputs as design values. Moreover, while hard constraints guaranteeing the bounds of final result values are prohibitively expensive to implement, they are not feasible. Therefore, our proposed model utilizes soft constraints, which cannot ensure absolute bounds. We intend to address these mentioned limitations as future research projects.

Despite the limitations of the model, we practically utilize the proposed model to provide high-performance test results to workers. The value of the setup parameter corresponding to the results obtained from several simulations is currently employed as the initial tuning value in the actual drive test. Initiating the test with the optimal initial values will enhance the efficiency of the test. Inappropriate initialization can result in an excessive number of tests being required, which is both time-consuming and costly. To circumvent this, we devised an algorithmic approach to derive optimal factors that satisfy multiple performance criteria. By utilizing the algorithmically derived optimal values as initial settings in the actual driving test, the time and number of tests can be significantly reduced.

4 Verification and Validation

In this section, we provide predicted design parameters to research laboratories that use analytical processes, and compare the performance derived from simulation programs with actual performance. Additionally, we demonstrate that the final results consistently vary with each prediction, as previously mentioned. The experimental data consists of 10% of the RWS data used in Chapter 3. This data is not involved in any model training and is used exclusively in this section and for model performance comparison.

4.1 Model Validation

Figure 7 presents a correlation plot between the performance obtained by applying design parameters in a simulation program and the actual performance. We provide plots for seven columns, and it can be observed that the majority of the approximately 1600 data points exhibit a high correlation. In numerical terms, the $R^2\ score$ is 0.93 and the MSE is 0.000089, which are lower than the results obtained when design parameters were input into the FNN (discussed in Section 3-1). This suggests that the performance of the FNN is not fully capable of replacing the simulation program. However, from the perspective of MSE, the model performance does not significantly deteriorate. Given the model's characteristic of producing different results each time, operators can conduct multiple prediction processes to achieve satisfactory results under their discretion, thus there are no issues in replacing the general analytical processes.

4.2 Prediction Diversity

The proposed model, unlike conventional models, produces different results for the same performance level each time. This not only leads to better outcomes but also provides more inspiration and information to operators. For instance, if predicted design parameter remains constant across the same performance level, it can be inferred that this design value holds more significant information. Therefore, in this experiment, we draw multiple design parameters (100 times) for the same performance level and examine the variations in these values considering their standard deviation. We provide a plot comparing randomly selected 10 performance metrics as Figure 8.

Both performance and design parameters are scaled between 0 and 1 (min_max scaler), and for each design parameter, we calculate the average and standard deviation (above and below the mean point) over 100 iterations. Significant deviations are observed in most columns, only column 6 exhibits a very small deviation. Column 6 shows minimal variation across all performance data, indicating it as the most crucial among all design parameters. Additionally, for each set of 100 design parameters corresponding to the same performance, we compare the predicted performance obtained by inputting these into a pretrained FNN with the actual performance. Consequently, the predicted performances across various design parameters are found to be remarkably consistent, with deviations near zero, indicating a level of similarity such that the values obtained are virtually identical.

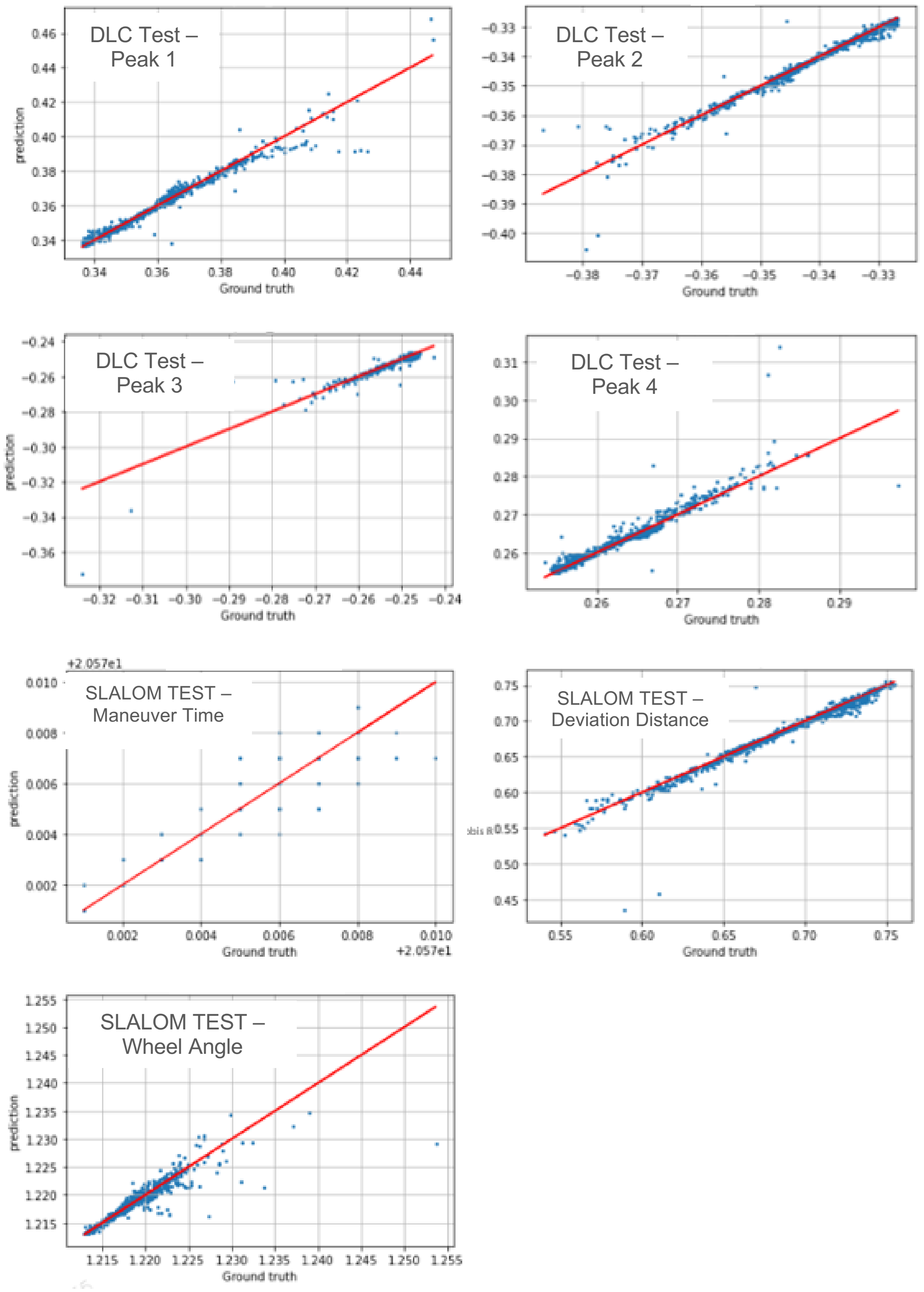

Figure 7. Correlation plot of RWS ground truth performance (x-axis) and the performance when design parameters are input the simulation program (y-axis).

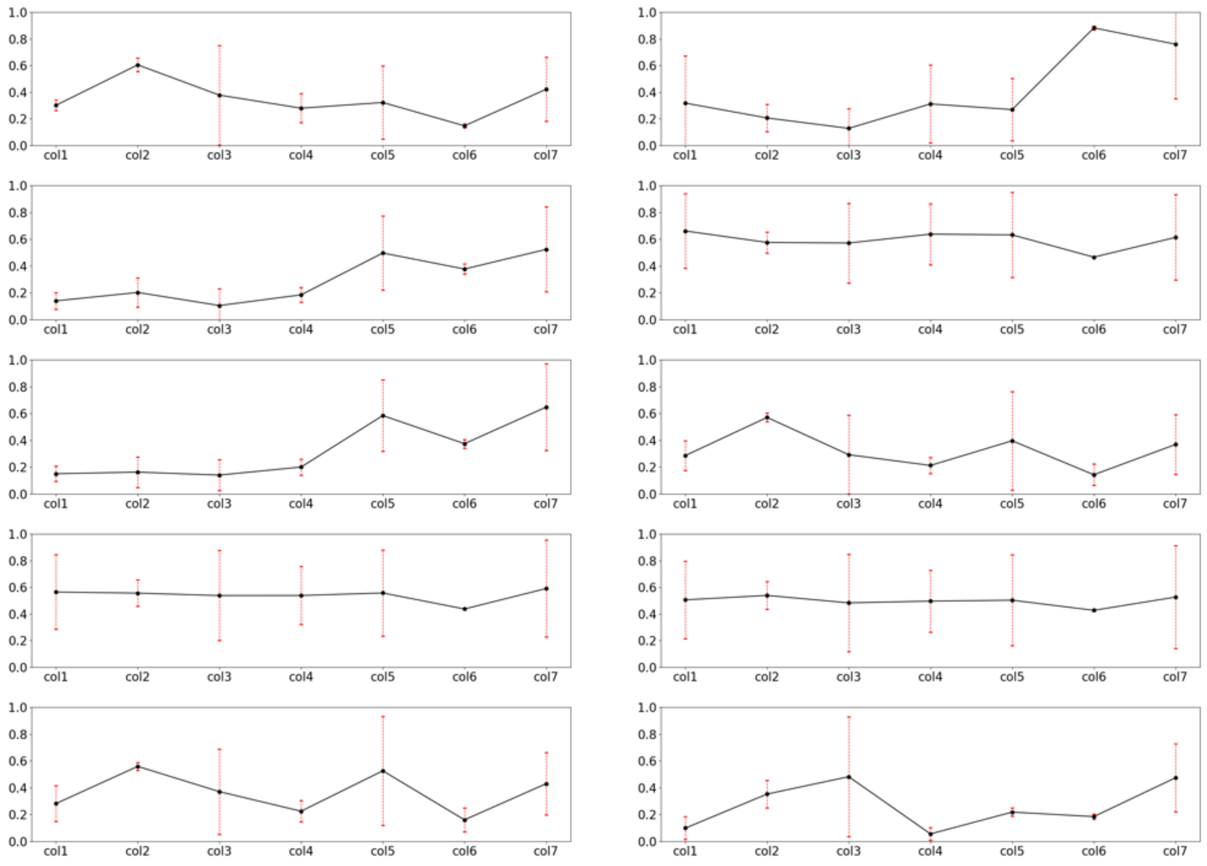

Figure 8. Plot of the average and standard deviation of the predicted design parameters (100 times) for 10 randomly selected performances.

5 Conclusions

The iterative analysis process required to meet target performance involves multiple simulation validations, leading to significant losses in terms of cost and speed. Particularly with the recent advancements in design processes, which incorporate multiple multi-performance criteria, this issue has become more pronounced. Therefore, we introduce a model that predicts design parameters satisfying target performance simultaneously, based on AI, rather than repetitive verification methods.

Our model constructs a new approach combining TNN and BO. The TNN predicts candidate design parameters that meet performance criteria. Specifically, to generate diverse candidates, randomness is introduced to the weights, which involves adding MC dropout or transforming into a BNN structure. Subsequently, BO is performed based on these candidate design parameters to acquire the optimal parameters.

In experiments, proposed model is validated using two analysis items. Data acquired from steering and braking are compared with typical machine learning models and various researched deep learning-based models. Our model demonstrates enhanced results compared to other models. Additionally, the performance resulting from the predicted design parameters input into the simulation program is compared with actual performance, confirming that the proposed model performs as intended. Unlike other models, our model can predict different design parameters each time, a capability that is also validated.

Future developments will focus on creating a model that is not dependent on FNN and adapting the structure to accommodate categorical variables as well as continuous ones. Moreover, we plan to apply sensitivity analysis techniques to explore the impact of design parameters on performance based on this model.

6 Nomenclature

$\bar{\boldsymbol{Y}}$	Target performance (actual performance)
$\boldsymbol{X}'$	Predicted design parameters (candidates)
$\boldsymbol{Y}'$	Predicted performance
$\bar{\boldsymbol{X}}$	Optimal design parameters (Final Result)
n_{tree}	The number of trees in the random forest
$r_k(\cdot)$	k-th tree in the random forest
f_{min}	Minimum value among the objective function values
t	Balance parameter between exploration and exploitation

7 References

[1] Liu. D, Tan. Y, Khoram. E, Yu. Z, “Training deep neural networks for the inverse design of nanophotonic structures”, Acs Photonics, 2018, 5(4), 1365-1369.

[2] Gal. Y, Ghahramani. Z, “Dropout as a bayesian approximation: Representing model uncertainty in deep learning”, In international conference on machine learning, PMLR, 2016, 1050-1059.

[3] Jospin. L. V, Laga. H, Boussaid. F, Buntine. W, Bennamoun, M, “Hands-on Bayesian neural networks—A tutorial for deep learning users”, IEEE Computational Intelligence Magazine, 2022, 17(2), 29-48.

[4] Hwang. H, Cho. Y, Hwang. S, Kim. S, “Optimal Tire Design Using Machine Learning and Bayesian Optimization”, Journal of the Korean Institute of Industrial Engineers, 2022, 48(4), 433-440.

[5] Bank. D, Koenigstein. N, Giryes. R, “Autoencoders”, Machine learning for data science handbook: data mining and knowledge discovery handbook, 2023, 353-374.

[6] Wang. H, van Stein. B, Emmerich. M, Back. T, “A new acquisition function for Bayesian optimization based on the moment-generating function”, In 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2017, 507-512.

[7] Noever-Castelos. P, Ardizzone. L, Balzani. C, “Model updating of wind turbine blade cross sections with invertible neural networks”, Wind Energy, 2022, 25(3), 573-599.