

# Development of Artificial Neural Network System to Recommend Process Conditions of Injection Molding for Various Geometries

Chihun Lee, Juwon Na, Kyongho Park, Hyeonjae Yu, Jongsun Kim, Kwonil Choi, Dongyong Park, Seongjin Park, Junsuk Rho,\* and Seungchul Lee\*

This study combines an artificial neural network (ANN) and a random search to develop a system to recommend process conditions for injection molding. Both simulation and experimental results are collected using a mixed sampling method that combines Taguchi and random sampling. The dataset consists of 3600 simulations and 476 experiments from 36 different molds. Each datum has five process and 15 geometry features as input and one weight feature as output. Hyper-parameter tuning is conducted to find the optimal ANN model. Then, transfer learning is introduced, which allows the use of simultaneous experimental and simulation data to reduce the error. The final prediction model has a root mean-square error of 0.846. To develop a recommender system, random search is conducted using the trained ANN forward model. As a result, the weight-prediction model based on simulated data has a relative error (RE) of 0.73%, and the weight prediction using the transfer model has an RE of 0.662%. A user interface system is also developed, which can be used directly with the injection-molding machine. This method enables the setting of process conditions that yield parts having weights close to the target, by considering only the geometry and target weight.

polymer pellets are melted via pressure and heating, as it passes through a screw. Then, the molten polymer is injected into a mold and cooled until it solidifies. This process is appropriate for complex-shaped products that require short cycle times. More than 1/3 of all thermoplastic materials are created via injection molding to meet the mass throughput of various product industries, including those of electronics, medical devices, and automobile parts.<sup>[1,2]</sup> Apart from its limitations (e.g., material selection for high fluidity polymer, surface-quality problems, high costs of machine and mold, and the need for expert process condition control), injection molding still enables high competitiveness because of its overwhelming throughput compared with additive manufacturing and machining.

Product quality is greatly influenced by process conditions (e.g., time, pressure, velocity, and temperature), which are set by engineers.<sup>[3]</sup> In most cases, the process


conditions are controlled by field experts based on experience. However, in some cases, an engineer uses computer-aided engineering (CAE) software to optimize the injection molding.<sup>[4,5]</sup> Simulation-based process optimization can be divided into two methods: direct discrete and metamodel-based methods.<sup>[6]</sup>

## 1. Introduction

Injection molding is a widely used manufacturing process for the mass production of plastic materials. The method consists of a mold, an injection machine, and molten polymer. First, solid

C. Lee, J. Na, Prof. S. Park, Prof. J. Rho, Prof. S. Lee  
Department of Mechanical Engineering  
Pohang University of Science and Technology (POSTECH)  
Pohang 37673, Republic of Korea  
E-mail: jsrho@postech.ac.kr; seunglee@postech.ac.kr

K. Park, H. Yu  
Advanced Technology R&D Group I, R&D Division  
LS Mtron Ltd.  
Anyang 14118, Republic of Korea

 The ORCID identification number(s) for the author(s) of this article can be found under <https://doi.org/10.1002/aisy.202000037>.

© 2020 The Authors. Published by WILEY-VCH Verlag GmbH & Co. KGaA, Weinheim. This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

DOI: 10.1002/aisy.202000037

Dr. J. Kim  
Molds and Dies R&D Group  
Korea Institute of Industrial Technology (KITECH)  
Bucheon 14441, Republic of Korea

K. Choi  
Development Team  
Virtual Molding Technology (VM Tech)  
Suwon 13350, Republic of Korea

Dr. D. Park  
Extreme Fabrication Technology Group  
Korea Institute of Industrial Technology (KITECH)  
Daegu 42994, Republic of Korea

Prof. J. Rho  
Department of Chemical Engineering  
Pohang University of Science and Technology (POSTECH)  
Pohang 37673, Republic of Korea

Direct discrete numerical optimization does not require explicit objective functions. However, it requires numerous simulations and cannot be used to develop an integrated process optimization system that covers various geometries and materials. Metamodel-based optimization, which is more popular than direct optimization, uses objective functions that are used to make approximate results with acceptable accuracy. Initially, the simplest metamodel-based optimization can predict the quality of products using a simple regression model.<sup>[7]</sup> Moreover, response-surface methodologies, artificial neural networks (ANNs), radial basis functions, and Kriging methods are common metamodels used for injection-molding optimization.<sup>[8,9]</sup>

Among the various metamodels, ANN-based optimization has been widely applied, because they can adopt major technical advancements, such as new activation functions, improved initializers, dropout regularization, and emerging optimization.<sup>[10]</sup> Mok et al. attempted to apply an ANN to set the initial process conditions using a single hidden layer ANN under fixed geometry and material conditions. They demonstrated that ANN-based optimization could surpass trial-and-error-based time and accuracy. Various additional studies have been conducted to improve the performance of ANN-based optimization. Early studies developed ANN models based on a limited number of simulation data. For example, the Taguchi method was applied to 27 data samples for ANN training.<sup>[11–13]</sup> Due to the limited number of data, they select the test data samples from the 27 cases, which cannot ensure the accuracy about arbitrary data. Shen et al. collected the 252 samples that dramatically reduced errors.<sup>[14]</sup> Data sampling methods and data numbers have been studied using various design-of-experiment (DoE) methods to find the optimal sampling method with limited data. Full factorial sampling, the Taguchi method, and combinations have been applied to control process conditions using trained ANN models.<sup>[11,15]</sup>

Input and output features were changed according to each research's objective. Mold and melt temperature, injection time, packing pressure, and packing time are usually applied as process conditions. Dynamic input data, such as pressure and velocity profile data (instead of a single point), are normally preprocessed using self-organizing maps (SOM).<sup>[16]</sup> For quality characteristics, most cases measure warpage and weight. The maximum von Mises stress is used as an output feature to evaluate the product's impact resistance based on the simulation data.<sup>[17]</sup> However, when experimental data are used, quality characteristic features are restricted to easy measurable features, such as temperature and weight (instead of warpage and maximum stress).

Metamodel-based optimization requires both an ANN prediction model (from process condition to quality feature) and an optimization method for the reverse model (from quality feature to process condition). Genetic algorithms and particle-swarm optimization have been used<sup>[18,19]</sup> to reduce computational cost and to find reverse solutions (output feature to input features). However, these studies only used simulation and experimental data. Others, however, used simulation data to develop a prediction model, testing it using experimental data.<sup>[20]</sup> Verification results about injected lens curvature showed 4.28–16.2% relative error (RE) between prediction and experimental results, which is too high for the real applications. Furthermore, previous models could not consider the characteristics of geometrically distinct

components, because they only dealt with single mold. This limitation could be overcome using variety of mold data for training. Hopmann et al. developed a prediction model that could be applied to two different molds. However, each ANN model was used for only one; an integrated ANN model was not achieved.<sup>[21]</sup>

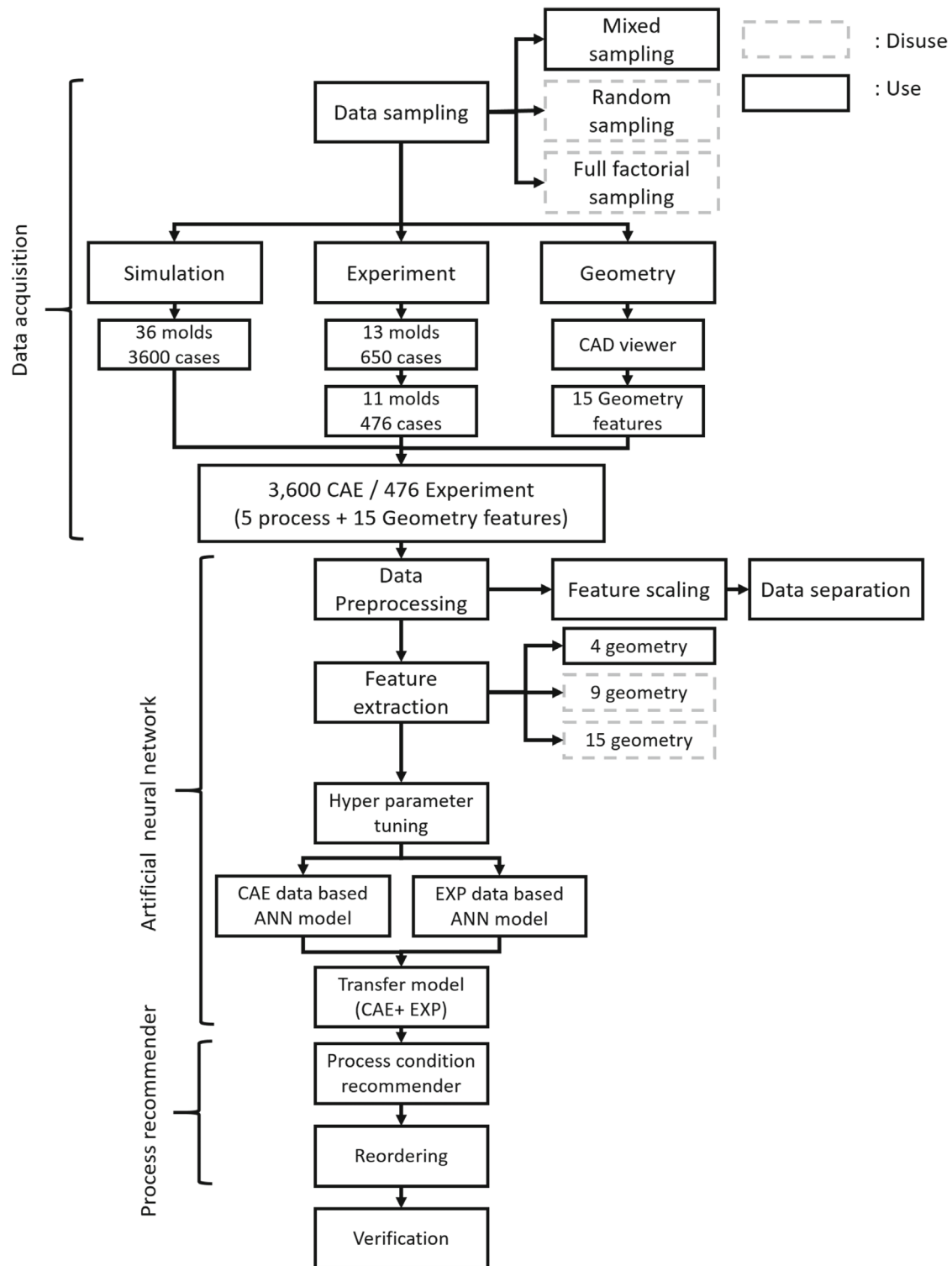
The major contributions of this article are the introduction of versatile geometrical molds and applying transfer learning. Various molds-based ANN model is first introduced for a general prediction model. Most previous studies only changed the process condition under one fixed-geometry condition. Therefore, to apply a process-control system that uses an ANN, an engineer must conduct the simulation or experiment to gather the training data to create a surrogate model. However, we gather both simulation and experimental data from 36 different molds (3600 simulations and 476 experiments). After that, the influence of geometry information is quantified and used as an input feature. We also introduce transfer learning,<sup>[22]</sup> which can combine both simulation and experimental data, to improve accuracy. It can minimize the gap between the simulation and real data, and thus, it can improve the performance with limited experimental data. Furthermore, we conduct the systemic approach to determine data sampling and DoE methods, hyper-parameters, and optimization to maximize the ANN's performance. As a final process, the system is integrated with a real injection-molding machine based on a user interface system. This research develops a system that recommends appropriate process conditions by considering only geometry and the final product information. With this system, non-expert engineers can set up the machine in a short time.<sup>[23]</sup> The remainder of this article is composed as follows. Section 2 describes the method of data acquisition for simulation, experiment, and geometry. Section 3 specifies the ANN algorithm for injection molding. Section 4 develops the process-condition recommender system. Section 5 presents discussion and conclusions. Overall procedure of development of recommender system is shown in **Figure 1**.

## 2. Data Acquisition

A data-driven ANN model required a training dataset of input and output features. Thus, we must first determine the input features, the output features, and the total sample number. Following data acquisition and training, the ANN model can predict its output using arbitrary input feature combinations. The result of the ANN prediction model is a prediction value, and the result of simulation or experiment (not the metamodel) is the actual or true value. An example of a complete dataset is shown in **Table 1**.

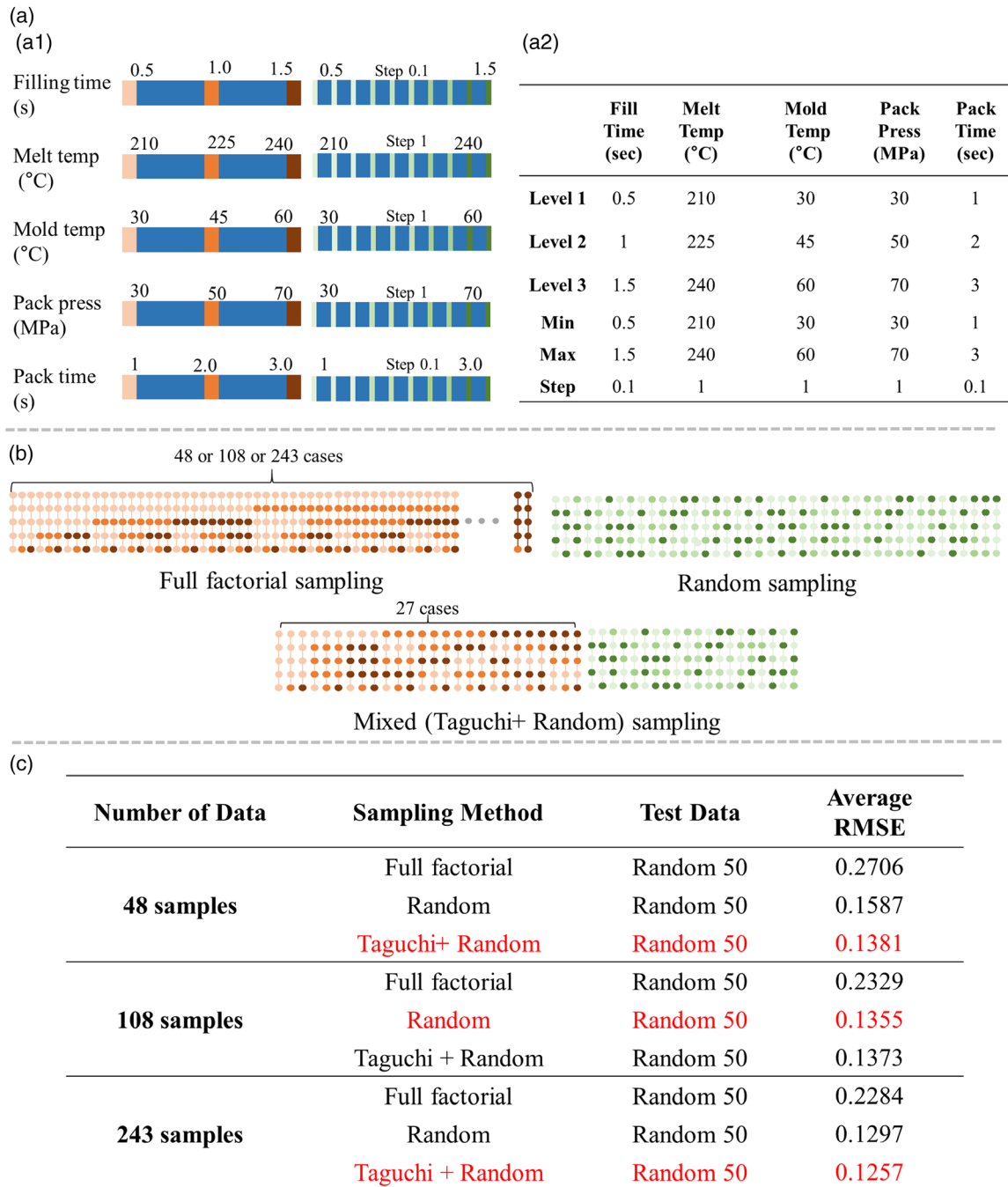
### 2.1. Sampling Method

Prior to gathering data, the process and quality features were determined. The input and output features required several necessary requirements: input features need to have a high influence on the product quality; it should be easy to quantify ANN training, and it should be easily measured by both experiments and simulations. Based on these requirements and previous



**Figure 1.** Overall procedure of the process recommender system development. It comprises three main parts: data acquisition, ANN, and process recommender system. Several sampling methods were considered, but mixed sampling gave the best result. With this sampling method, data from simulation (3600 cases from 36 molds) and experiments (11 molds from 476 cases) were collected. Each datum contained 15 geometric features, which were quantified by CAD viewer software. Then, the data were preprocessed for ANN learning. To increase the prediction model ability, feature extraction and hyper-parameter tuning were applied. Furthermore, transfer learning was introduced to combine data from simulation and experiment to overcome the limitations of experimental data (i.e., high noise, small amount). The well-trained ANN model was used to develop a recommender system by applying a random search method. Finally, simulations and experiments were repeated to verify the recommender system.





**Figure 2.** a) Values of process parameters for sampling. Minimum, middle, and maximum levels (given) were used for the full-factorial and Taguchi's methods. Minimum, maximum and step values were used for the random sampling method. b) Schematics of full factorial, random, and mixed sampling methods. c) Comparisons of results of sampling method and sample number. Mixed sampling method had the smallest RMSE error in most cases.

temperature  $\times$  packing pressure  $\times$  packing time, respectively) levels. Minimum and maximum levels were used for *packing pressure* and *packing time*. Mixed sampling combined 27 samples obtained using Taguchi's method and 87 obtained randomly. The 243-sample case used  $3 \times 3 \times 3 \times 3 \times 3$  (filling time  $\times$  melt temperature  $\times$  mold temperature  $\times$  packing pressure  $\times$  packing time, respectively) levels. Mixed sampling used the 27 cases chosen using Taguchi's method and 216 chosen randomly.

The methods yielded different results (Figure 2c). The error of each model was quantified using root mean-square error (RMSE)

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - t_i)^2} \quad (1)$$

where  $N$  is the number of samples.  $y_i$  is the predicted value of the  $i$ th data sample, and  $t_i$  is its true value.



Full-factorial sampling always showed poor results. Mixed sampling obtained the best results when the number of the sample was 48 or 243, giving similar results as those of random sampling when the number of samples was 108. Therefore, in this article, we used the mixed sampling method to maximize the accuracy of machine-learning algorithms using limited data. A training data example is described in Table 1.

## 2.2. Simulation

Machine-learning algorithms require different datasets to represent different patterns. There are various CAE approaches (e.g., finite volume method [FVM], boundary element method [BEM], and finite-element method [FEM]) for this purpose. FVM is only good for simple geometric processes, owing to its hexahedrons mesh type. BEM has limited capability handling inhomogeneous and nonlinear problems. Polymer behavior from solid to liquid is highly non-linear, making it difficult to solve using BEM. Therefore, FEM is the most appropriate method that can be used to describe the process of injection molding with high accuracy, despite being time-consuming.<sup>[5,26]</sup> Most commercial numerical injection-molding simulations are built using FEM, including Moldflow, Moldex, and Auto desk.

In this research, we simulated 36 molds of different shapes. Injection-molding simulation required 2000–3000 s per case, depending on the size and complexity of the mold. We conducted 100 cases for each mold, and we used a mixed sampling method. The total number of cases generated from 36 molds was 3600. Table S1, Supporting Information, shows an example of one of the molds. The simulations were conducted using Maps 3D (VMtech, Korea) CAE software.<sup>[27]</sup> The material was polypropylene, Hopelen J-150 (Lotte chemical, Korea) for all cases. We also verified the simulation model by comparing the results of arbitrary mold simulation and experimental data, as shown in Figure S1, Supporting Information.

## 2.3. Experiment

A simulation requires a long setup and execution times and has limited accuracy, and, although simulation and experimental results have similar patterns, experimental values are generally more accurate and useful than simulated ones.<sup>[28]</sup> Therefore, the process recommender system built from CAE data is not appropriate for application in real industry. Experimental data must instead be obtained to guide the controls of an injection-molding machine. Unfortunately, experimental data are much more difficult to collect than CAE data. When process conditions are changed, data are only accurate after the system reaches a stable state. For example, when the mold temperature is reduced from 50 to 45 °C, stabilization requires many cycles. Furthermore, to reduce experimental error, the average weight is used after repeating experiments ten times per condition. These constraints limit the amount of data to be collected, compared with the simulation case. We experimented with 50 conditions for each mold and used mixed sampling: 27 cases obtained using Taguchi's method, and 23 cases obtained by random sampling (Table S2, Supporting Information). Using 11 different molds yielded 650 conditions. However, we only used 476,

because the others caused faults (e.g., burning, short shots, and sink marks). The same materials that were used in the simulation were used in the experiments. The measured weight was the total of the runner and the product.

## 2.4. Geometry

Geometry information was quantified and included among the ANN input variables. If the 3D or mesh information were to be used as geometry information, the dimension of the data would become too large, and learning could not be performed as normal. Therefore, only variables that can be automatically quantified using a printing stereolithography (STL) file were used to make injection-molding ANN models.<sup>[29]</sup> The 15 pieces of shape features (Figure 3a) were extracted and jointly developed using computer-aided design (CAD) viewer software (VMtech, Korea). Table S3, Supporting Information, shows the list of 36 molds and the data number of each.

Using the CAD viewer, a user can extract quantified geometry information using the product STL file without simulation. Most geometry features (e.g., volume, surface area, and projection area) can be easily obtained from the STL file without special methods. Some features, however, require analysis using appropriate algorithms. For example, we used the shrink-sphere method to obtain thickness information for the geometry of the injection-molding product.<sup>[28]</sup> In this method, to calculate the thickness of a point, several spheres are created surrounding that point. Then, each sphere's volume is increased until all spheres touch. Then, the thickness of the point is obtained by the smallest diameter that fits within the surrounding spheres.<sup>[30]</sup> Another feature is flow length, which is an index that evaluates the distance from the gate to the filling end, using only the shape information without simulation. To do this, the shortest path between elements is found using the Dijkstra algorithm.<sup>[31]</sup>

The final output from the CAD viewer is json-type data divided into entire information, cavity information, and gate information at the first level. Detailed values are included at the next level. To utilize this information, we developed a Python module that passes the geometry information of json-type data from the CAD viewer.

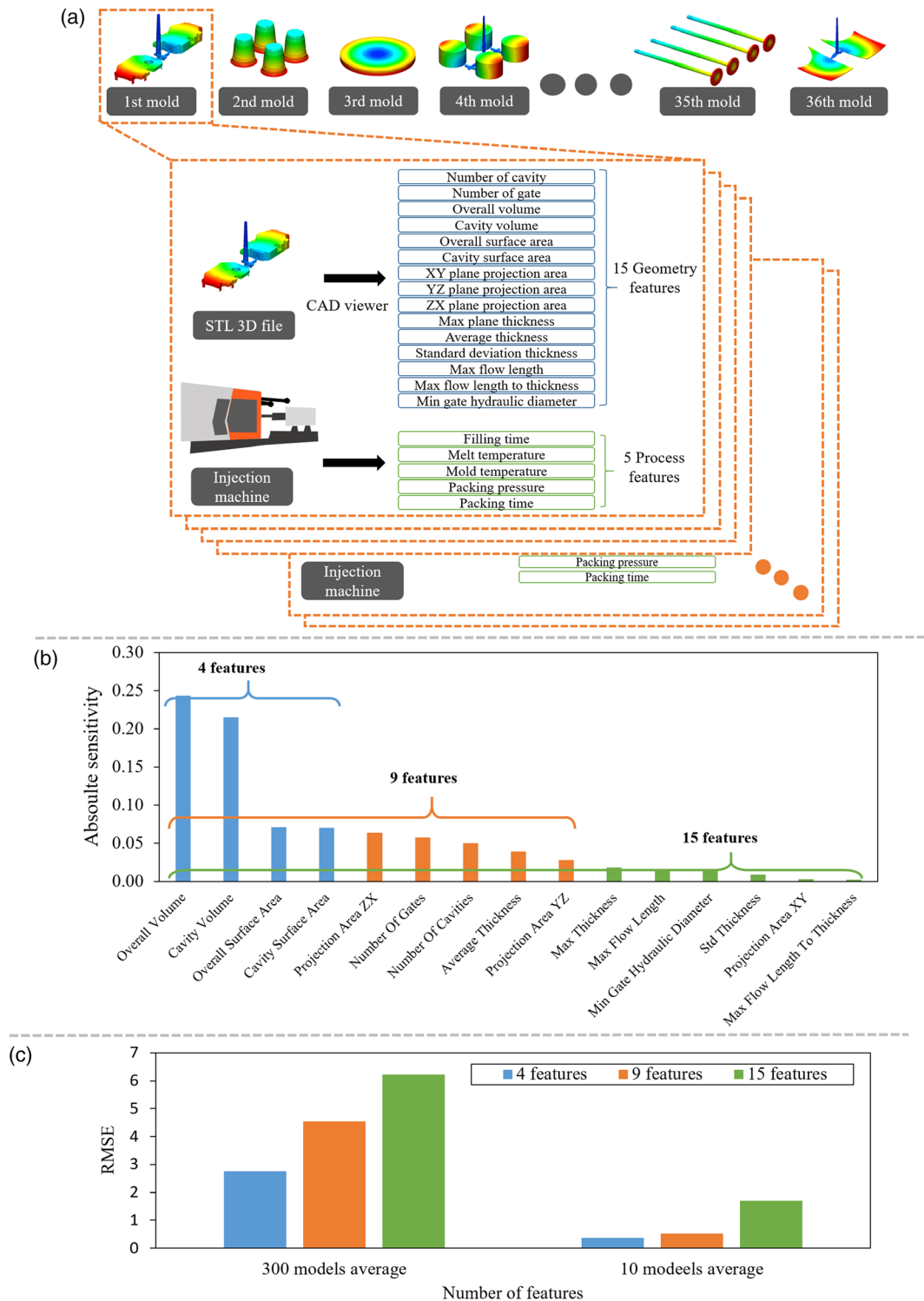
## 3. Artificial Neural Network

An ANN algorithm was used to predict the final product properties. The smallest unit cell is a neuron. Each neuron multiplies its input by a weight, and then uses a transfer function to generate an output  $net_j$  as

$$net_j = \sum_{i=1}^n (x_i w_{ij} + b) \quad (2)$$

where  $w_{ij}$  is the weight value from  $j$  to  $i$ ,  $x_i$  is the  $i$  input value,  $n$  is the number of input, and  $b$  is the bias value. Then,  $net_j$  passes to a non-linear activation function. Various activation functions can be used, depending on the model structure and data.

The neuron is one node of the overall ANN structure, which consists of three parts: the input, output, and hidden layers.



**Figure 3.** a) Schematic of data gathering for 36 different molds. Each mold has 15 geometric features and five process features. b) Sensitivity analysis result depending on the number of geometric features. c) Feature extraction error comparing result. The case using four features case was most accurate for averages of both the best-300 models and best-10 models.

The ANN training procedure consists of forward and backward propagation. Forward propagation calculates the final value using the updated weight value from input to output layers. Thus,

appropriate updating of weight and bias values is the key point of ANN training. Backward propagation updates those weight and bias values to minimize the error between the forward

propagation result and the data output.<sup>[32]</sup> The gap between the target value and forward result is evaluated by calculating the RMSE.<sup>[33]</sup>

Our weight-prediction model was generated in three steps. First, the data were preprocessed for machine learning. This process includes scaling and data separation. Next, the appropriate ANN structure and parameters for the data were selected using hyper-parameter tuning. We compared both grid and random searches for hyper-parameter optimization. Finally, transfer learning was used to combine simulation and experimental data.

### 3.1. Data Preprocessing

Data preprocessing consists of two main parts: unification of the distribution of the input data, and separation of the data into training, validation, and test sets. If the scales of the input values are different, the convergence speed slows, and the accuracy decreases.<sup>[34]</sup> The process conditions also have very different scales. For example, melt temperature varies from 180 to 200 °C, whereas a filling time varies from 0.5 to 1.5 s. Feature scaling is performed to eliminate this scale difference. We want to use information about patterns of data in machine learning via scaling methods.<sup>[35]</sup> In this study, min–max normalization was used

$$P = \frac{D - D_{\min}}{D_{\max} - D_{\min}} \times (P_{\max} - P_{\min}) + P_{\min}$$

$$P_{\max} = 0.9$$

$$P_{\min} = 0.1$$
(3)

where  $P$  is a normalized datum,  $D$  is the input datum,  $D_{\min}$  and  $D_{\max}$  are, respectively, the minimum and maximum values of the original data, and  $P_{\max}$  and  $P_{\min}$  are, respectively, the upper and lower boundaries, after normalization. After normalization, all variables range from 0.1 to 0.9.

After data re-scaling, the data were divided into a training set, a validation set, and a test set. The training set was used to learn the ANN model, and the validation set was used for hyper-parameter tuning. The test set was used to evaluate the accuracy of the tuned model. In general, the ratio between the validation and test sets is 10–15%, depending on the number of data. The validation and test set ratios decrease with the total amount of data. When the number of data is very small, “leave one out” cross-validation is used.<sup>[33]</sup>

In this study, data separation was performed differently for the simulation and experimental cases, because the numbers of simulation data and experimental data were different. The simulation considered 36 molds (3600 samples). The 28 molds (2800 samples) were used as the training set, six molds (600 samples) were used as the validation set, and two molds (200 samples) were used as the test set. The experiment evaluated 50 samples from 11 molds. Some samples were not available. Thus, we used data that had been collected previously. We used ten molds (327 samples) as the training set, two molds (99 samples) as the validation set, and one mold (50 samples) as the test set.

### 3.2. Sensitivity Analysis

A sensitivity analysis was conducted using Equation (4)<sup>[36]</sup> to determine the main features for the prediction model.

$$\text{Sensitivity} \equiv \text{Output change (\%)} / \text{Input change (\%)} \quad (4)$$

The relationship between the input and output of injection molding is not simply linear. Thus, we performed sensitivity analysis for changing ratios. We built an ANN model with simulation data. The volume- and surface-area-related features have a significant effect on the final product weight.

To select the final geometry features for an ANN model, we built several models with different geometries (i.e., different input dimensions) (Figure 3b). Models having 300 different hyper-parameters were generated for each number of geometry features used, and the appropriate number of geometry features was evaluated using the average of the top 10 and 300 ANN models. The averages of top-10 RMSEs were 0.373, 0.530, and 1.698 for 4, 9, and 15 geometry features, respectively. The average of the 300 RMSEs was 2.749, 4.542, and 6.215 for 4, 9, and 15 geometry features, respectively. The tendency was similar to the top-10 result (Figure 3c). Therefore, we used four geometry features for the final ANN model. In the general case, more features will help increase accuracy. However, in this article, the minimum geometry features case had the best RMSE, because the output feature was the only weight of products. Volume and surface area information have significant effects on the final product's weight compared with other features, such as flow length, projection area, and thickness. Moreover, owing to the lack of geometrical diversity (36 different molds), more geometry features were not available.

### 3.3. Hyper-Parameter Tuning

To build an ANN model, many parameters should be considered, (e.g., activation function, optimizer, initializer, learning rate, layer, node, epoch, dropout, and batch size). Those hyper-parameters have significant effects on prediction accuracy. In this section, we compare and analyze various types of hyper-parameters. For the activation function, optimizer, and initializer, we used the widely used methods.<sup>[37]</sup> Then, we conducted a random search to determine the remaining hyper-parameters (i.e., layer, node, epoch, dropout, learning rate, and batch size).

Determining the activation function, optimizer, and initializer is a very complex problem. The appropriate method depends on the data. Some papers have shown that the appropriate initializer depends on the type of activation function.<sup>[38]</sup> Therefore, in this article, we empirically determined this function by combining three hyper-parameters: activation function, optimizer, and initializer. We then used RMSEs to compare the error. For comparison, we applied the 3600 cases of simulation data and fixed the other hyper-parameters. The comparison test used one hidden layer, 30 nodes, a learning rate of 0.01, 3000 epochs, dropout turned off, and a batch size of 1500.

The choices of activation function, optimizer, and initializer affected the accuracy of the result. The average RMSEs, according to the activation function, were 7.691, 6.317, 5.655, and 4.893 for sigmoid,<sup>[39]</sup> tanh, the rectified linear unit (ReLU),<sup>[40]</sup> and the exponential linear unit (ELU),<sup>[41]</sup> respectively. This means that ELU was the most appropriate activation function for injection-molding data. The initializer had a minor effect on the RMSE of the prediction model. For the optimizer, the accuracy was



**Table 2.** Hyper-parameter tuning result for initializations, optimizers, and activation functions. Each of hyper-parameters has widely used methods. The grid combination is  $4 \times 4 \times 4 = 64$  cases. RMSE is calculated for all 64 cases; then, the model with lowest RMSE is chosen: it was He initializer, ELU activation function, and Adam optimizer.

RMSE	Normal	Truncated normal	He	Xavier
<b>ReLU</b>				
Adagrad	8.942	10.99	9.776	11.918
Gradient	5.971	4.175	4.415	4.65
RMSProp	3.096	4.621	5.886	5.589
Adam	2.724	3.171	2.322	2.437
<b>Tanh</b>				
Adagrad	8.806	14.828	14.036	10.757
Gradient	6.273	6.101	4.108	3.937
RMSProp	6.232	6.249	3.957	2.619
Adam	3.8565	2.598	3.7	3.018
<b>Sigmoid</b>				
Adagrad	16.965	14.339	14.128	14.579
Gradient	5.607	4.95	5.598	5.013
RMSProp	5.394	4.773	5.297	3.701
Adam	6.663	5.397	5.376	5.276
<b>ELU</b>				
Adagrad	10.086	11.896	11.205	10.532
Gradient	5.282	3.049	6.981	4.138
RMSProp	2.356	2.414	2.192	4.412
Adam	1.587	0.8614	0.672	0.732

highest using Adam,<sup>[42]</sup> with accuracy descending using RMSProp,<sup>[43]</sup> gradient descent,<sup>[44]</sup> and Adagrad<sup>[45]</sup> in that order. The ELU activation function, the Xavier initialize,<sup>[46]</sup> and the Adam optimizer gave the best accuracy. Therefore, we used them for each model (Table 2).

Random search is an optimization method that can seek additional feature space for a given trial. In this research, we tried 324 random searches. The hyper-parameter list and its minimum, maximum, and step size were specified for each parameter (see Table 3). For example, in the case of the number of layers, from Steps 2 to 5, the step size was 1. Thus, we selected 2, 3, 4, or 5 as the number of layers. The RMSEs of the models obtained using grid search were 0.4, 0.416, and 3.112 for the top average, the top-10 average, and the average of the entire 324, respectively. The RMSEs of models obtained via random search were 0.314, 0.367, and 2.749 in the same order. These four comparisons indicate that the models obtained using the random search were advantageous in finding the global best for the same number of trials. The best model had two layers, 28 nodes, 5500 epochs, a 0.05 learning rate, a 0.9 dropout rate, and a 1152 batch size (Table 3).

### 3.4. Transfer Learning

Thus far, all processes were performed only via simulations. However, as noted, the results of the simulations and the experiments differ (Figure 4a).<sup>[22,41,47]</sup> Furthermore, practical applications

**Table 3.** Hyper-parameter tuning for layers, nodes, epochs, learning rate, dropout rate, and batch size. Each of parameter has minimum, maximum, and step value. To find the optimal hyper-parameter, random search is conducted by combining those parameters (324 different models). As a result, two layers, 28 nodes, 5500 epochs, a 0.05 learning rates, 0.9 dropouts, and 1152 batch size had minimum RMSE error for validation data set.

	Number of layers	Number of nodes	Number of epochs	Learning rate	Dropout	Batch size
Min	2	5	1000	0.005	0.9	512
Max	5	30	20 000	0.1	1.0	2048
Step	1	1	500	0.005	0.1	128
Top 1 RMSE	0.400				0.314	
Top 10 RMSE	0.416				0.367	
Total 324 RMSE	3.112				2.749	

require a model that is derived from experimental data. The amount of experimental data is limited. Thus, the ANN model did not give good results (RMSE 7.178, average RE 13.604%) when using the same hyper-parameters as the simulation data. These inferior results occurred, because the experimental data had more noise than the simulation data, so prediction was inaccurate. To solve this problem, we implemented transfer learning. In deep learning, a model can be trained with sufficient data and labels. However, this method results in overfitting or underfitting when the number of data is insufficient. Injection-molding data have similar problems. Simulation data are cheaper and easier to gather than experimental data. Transfer learning can conquer the immense error that occurs when only experimental data were used.

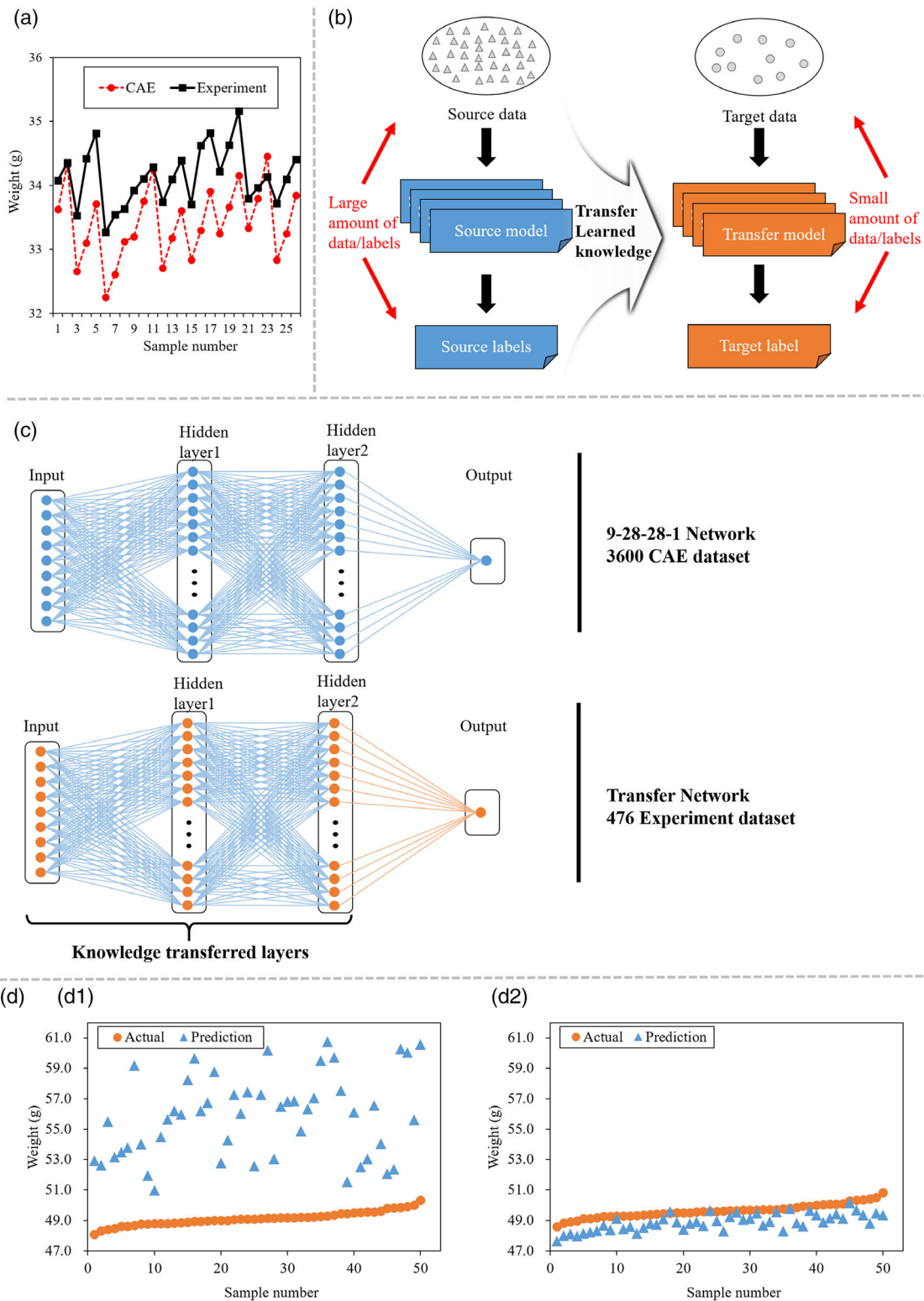
Transfer learning consists of two steps. First, a model is trained using a large number of cheap data. Then, only a few layers of the model are trained with the small amount of expensive target data (Figure 4b). In this article, we use the simulation data as source data and experimental data as target data (Figure 4c). This approach achieved better accuracy (RMSE 0.846, average RE 0.715%) than cases that used experimental data only (Figure 4d).

## 4. Process Condition Recommender System

We used a weight-prediction model to develop a system that recommends initial process conditions. This weight-prediction system uses 20 input features to predict the weight of the final product by combining the shape information and the process conditions. However, the system used to derive initial conditions should reverse the model to recommend process conditions that satisfy the target weight. This task requires an ANN forward model that predicts the weight accurately in real time from process conditions and weight. We also need an optimization algorithm to find the solution.

### 4.1. Random-Search Optimization

Random searching for optimization is very similar to random searching to find the hyper-parameter. The optimization search



**Figure 4.** a) Example of weight difference between experimental and simulation data. They have similar patterns but different values because of simulation errors. b) Concept of transfer learning. c) Application of transfer learning to injection-molding data. “Large amount data” are 3600 cheap simulation data. “Small amount of data” are 476 expensive experimental data. The model first learns using the simulation data. Then, it copies the weight except for the last layer. Then, the final layer’s weights are updated from scratch using the experimental data. Distribution of actual and predicted weights d1) before transfer learning (only experimental data) and d2) after transfer learning (both experimental and simulation data). Both graphs are arranged by ascending order for visualization.

can be completed in real time, because only the forward process of the trained ANN is used. The optimization that conducts the random search proceeded as follows: 1) extract the minimum and maximum values of input process conditions; 2) set a step size for each process condition, and make a candidate list; for example, for fill time, 0.5 s, 0.6 s, 0.7 s ... , and 1.5 s; 3) combine the process features from the list from Step 2 10 000 times; 4) predict the list of 10 000 random process conditions to the trained ANN model; and 5) from the 10 000 weight-prediction results, extract the process conditions that satisfy the target weight range (Figure 5a).

#### 4.2. Ordering System

The system that uses random search to optimize initial conditions yields 70–80 process conditions, depending on the model. However, the user only wants to obtain a few conditions that are approximate to the current state of the injection-molding machine. Therefore, we develop an ordering system that is easy to apply to injection-molding machines. The system comprises of four major steps: 1) normalizing all recommended process conditions and the current state of injection molding; 2) calculating absolute error from recommended process conditions to the current state; 3) obtaining the weight sum (Equation (5)); and 4) selecting the top ten process conditions and re-ordering by mold temperature.

$$\begin{aligned} \text{Weighted sum} = & a|\text{MeltTemp}^{\text{current}} - \text{MeltTemp}^i| \\ & + b|\text{MoldTemp}^{\text{current}} - \text{MoldTemp}^i| \\ & + c|\text{Filltime}^{\text{current}} - \text{Filltime}^i| \\ & + d|\text{Packtime}^{\text{current}} - \text{Packtime}^i| \\ & + e|\text{Packpress}^{\text{current}} - \text{Packpress}^i| \end{aligned} \quad (5)$$

where  $a = b = 0.35$  and  $c = d = e = 0.1$  are weight coefficients.

The first step of normalization unifies each feature's effect to calculate the weighted sum at step 3. In our case, the normalized process conditions were subtracted from normalized values of the current machine state. Then, Equation (5) was used to calculate the weighted sum. We applied high coefficients for melt and mold temperatures, because temperature-related features are difficult to control. We calculated the weighted sum only for the top ten results. As a final step, we sorted these ten process conditions again by mold temperature. Using this ordering system, the user can apply the process condition from low-to-high mold temperatures (Figure 5a).

#### 4.3. Verification

We built an ANN model that estimated product weight by considering geometry information and process conditions. As a result, we obtained two systems: one that was learned using simulation data, and the other that was created by transfer learning using combined simulation and experimental data. To verify that these systems are usable, we used CAE software for the simulation-driven model and conducted experiments for the transfer-learned model. Finally, ten recommendation conditions were re-simulated and re-experimented. The mean REs were 0.656% for a cup mold and 0.804% for a cosmetic container.

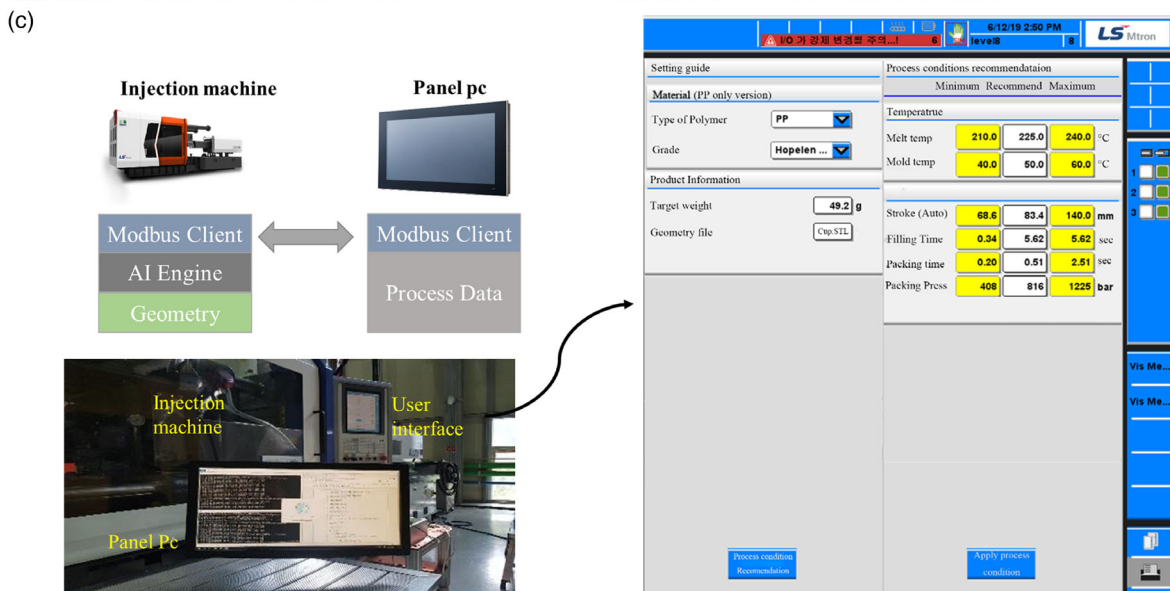
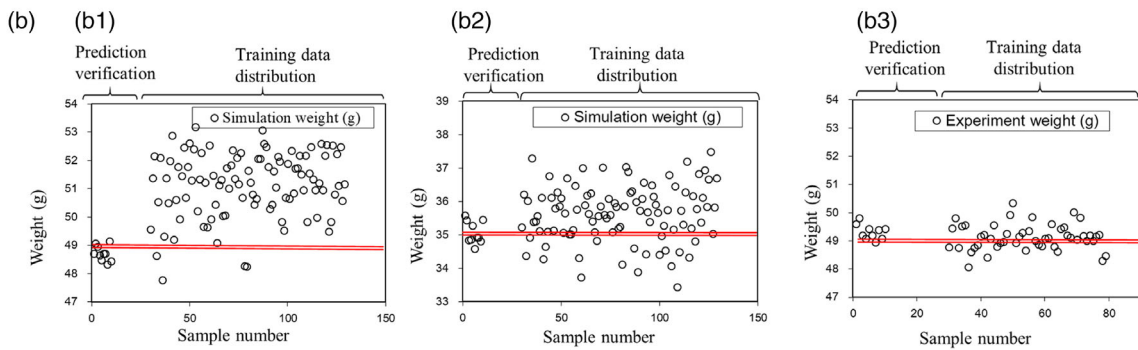
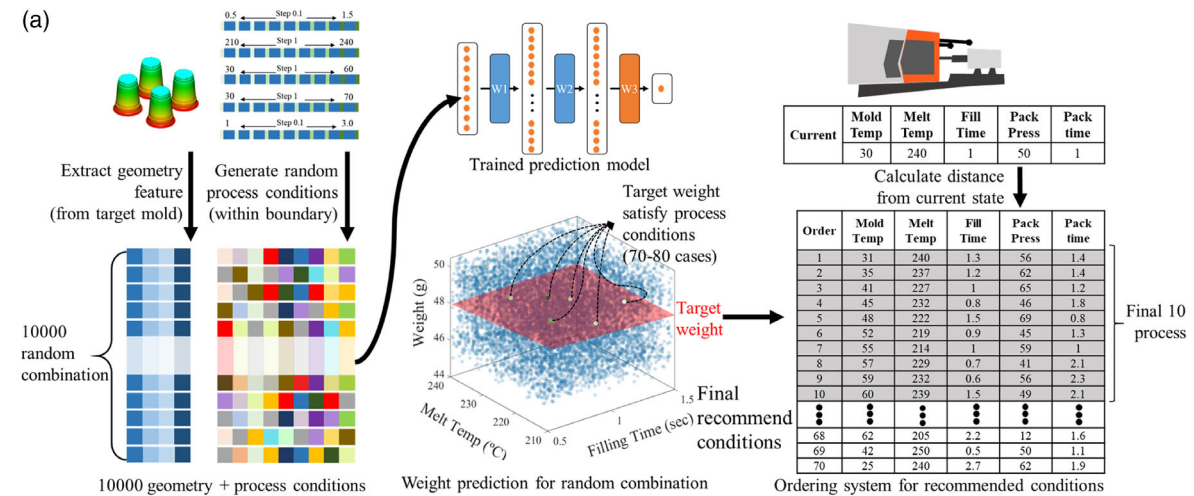
Similarly, the system using transfer learning had an average RE of 0.662% in experiments (Figure 5b). These results indicate that the process recommender system can suggest reasonable process conditions at values near the target weight. The ANN-based system searches the proper process condition using various methods. Past recommended process conditions mainly used temperature, pressure, or velocity. Thus, they used only one optimal solution. Our recommender system, on the other hand, explores multiple optimal solutions. However, for actual system use, users still must manually control the machine to find the optimal process condition due to the error. These errors are caused by various reasons: simulation error, experimental measuring error, and ANN model error. Both simulation and experimental errors are hard to correct. Therefore, in this article, we focused on reducing the ANN-model error via gathering large datasets, hyper-parameter tuning, and transfer learning.

#### 4.4. User Interface

The algorithm was trained using a powerful 2950x thread ripper (Advanced Micro Devices, Inc.) central processing unit workstation, a Titan volta 12 GB graphical processing unit (GPU) with python tensorflow GPU. However, the use of such expensive computers may not be viable in an actual industrial environment. Therefore, we stored only the forward model of the trained ANN-based prediction model and made it into an executable file, including a random search and an ordering system. Such files can be run in real time on a cheap panel computer. A serial communications protocol called Modbus is used to transfer the results from the panel computer to the injection-molding machine to help it automatically apply the process conditions. A user interface system was also developed to utilize and control these systems on the injection-molding machine screen (Figure 5c).

## 5. Discussion and Conclusion

We developed a system that recommends injection-molding process conditions in real time under fixed material conditions. To construct this system, various methods (e.g., data sampling, feature extraction, hyper-parameter tuning, transfer learning, and optimization) were applied. Several widely used methods were compared at each step to find an optimized method for the injection-molding process. Data were sampled using a mixture of Taguchi and random sampling. The 3600 cases of simulation data (from 36 molds) and 476 cases of experimental data (from 11 molds) were collected as training, validation, and testing for ANN model. We also conducted geometry-feature extraction guided by the results of sensitivity analysis. As a result, four geometry features related to surface area and volume were applied for the final ANN model. After gathering data, a random search was conducted to find the optimal hyper-parameters. Then, we used transfer learning to enable the model to use experimental data. The completed weight-prediction ANN model was used to develop a system to recommend process conditions via random search. Then, the final recommendation condition was extracted using an ordering system and verified by simulation and experiment. The final model achieved an average RE of



**Figure 5.** a) Schematic of random search optimization. First, geometric features are extracted from the target molds. Then, 10 000 random combinations are generated within the range of minimum to and maximum of each process condition. Then, 10 000 random product weights are predicted, and the conditions that yielding the target weight are selected. As a last step, the nearest-10 process conditions are sorted and selected. b) Verification result of the process recommender system. Verification of simulation results of b1) cup and b2) cosmetic case and b3) experimental verification of the cup mold. Each graph's left-10 points are verification result, and right-50 or -100 points are the training source data. Comparison confirms that the recommender system can reduce the distribution of the product weight to near the target weight. c) Left: concept and equipment setup of connecting method between injection machine and panel computer; right: user interface for the process recommender system. Material and geometry information are needed to calculate and recommend the optimal process conditions.



0.73% from simulation and 0.63% from experiments. We then developed a user interface.

This study differs from existing studies in two major ways. First, this one considers the geometry conditions. Second, we used data from both simulation and experiment. In general, a manufacturing system can consider three types of inputs: process conditions, geometries, and materials. Most prior studies that combined ANNs and injection molding only considered changes in process conditions of fixed mold shape and molding material. In contrast, in this article, the process was optimized by obtaining a large amount of data from a range of geometric conditions (36 molds). In addition, extant studies used only simulation or experimental data, whereas ours introduced transfer learning for both. This combined use of a small number of expensive experimental data and a large number of cheap simulation data achieved similar performance as methods using large experimental data.

As a result of the final verification, the weight mean RE was 0.63%, which is acceptable in an actual industrial machine. There are various reasons for this level of error. First, it exists in the training data itself. Simulation data have inevitable errors, as shown in Figure S1, Supporting Information. Similarly, as shown in Table S2, Supporting Information, experimental data have small fluctuations, despite the same process conditions. Moreover, the lack of experimental and geometry data increased the error. The number of molds and experimental data was insufficient compared with the complexity of injection molding. However, these data required a great deal of time and cost to acquire. Therefore, it is important to consider an alternative algorithm to perform well using a small amount of data. Nonetheless, this study changed the process and geometry conditions, but it did not consider changes in material. In addition, it used only the weight of the product as the index to evaluate product quality. This decision was made, because the output feature must be easily quantifiable and measurable with both experiments and simulations for various geometries. Therefore, further research should quantify the properties of the material, and data should be collected via simulation and experiments using various materials.

In conclusion, this research developed a system that even a non-expert in injection molding can use to set up process conditions using the artificial intelligence (AI) system. Assuming that the material is polypropylene, the user only needs a 3D file with an STL extension and the target weight of the product. Then, the recommender system suggests ten process conditions having less than 1% RE. As a final step, the engineer selects one condition among the ten recommended conditions and performs fine-tuning. Previous studies have only considered laboratory-level development that combined the ANN and injection molding. This article, on the other hand, shows the possibility of optimizing AI-based injection molding at actual industrial sites. Future work should improve the quality and quantity of data. Qualitative development should consider factors, such as material conditions, environment conditions, process features, and geometric features. Quantitative improvement should consider various molds and materials. As a result, the system will be able to respond quickly to new products and could be used to control the process conditions of a smart factory.

## Supporting Information

Supporting Information is available from the Wiley Online Library or from the author.

## Acknowledgements

This work was financially supported by the Technology Innovation Program (No.10067766) funded by the Ministry of Trade, Industry & Energy (MOTIE), Republic of Korea, and the National Research Foundation (NRF) grants (NRF-2019R1A2C3003129, CAMM-2019M3A6B3030637, NRF-2019R1A5A8080290) funded by the Ministry of Science and ICT (MSIT), Republic of Korea. The authors thank Sunae So (POSTECH) for the help preparing the revised manuscript.

## Conflict of Interest

The authors declare no conflict of interest.

## Author Contributions

C.L.: Data curation, formal analysis, methodology, software, visualization, writing-original draft, and writing review and editing. J.N.: Methodology. K.P.: Project administration and formal analysis. H.Y.: Data curation and validation. J.K.: Data curation and validation. K.C.: Software, simulation, and data curation. D.P.: Writing review. S.P.: Supervision and funding acquisition. J.R.: Writing review and editing, overall editing, conceptualization, resources, and project administration. S.L.: Methodology, writing review and editing, resources, and project administration.

## Keywords

injection-molding computer-aided engineering, manufacturing process optimization, process control, search algorithm, transfer learning

Received: March 9, 2020

Revised: June 24, 2020

Published online:

- [1] A. L. Andrad, M. A. Neal, *Philos. Trans. R. Soc. B* **2009**, 364, 1977.
- [2] M. L. H. Low, K. S. Lee, *Int. J. Prod. Res.* **2008**, 46, 6269.
- [3] C. M. Seaman, A. A. Desrochers, G. F. List, *IEEE Trans. Control Syst. Technol.* **1994**, 2, 157.
- [4] C. A. Hieber, S. F. Shen, *J. Nonnewton. Fluid Mech.* **1980**, 7, 1.
- [5] C. Fernandes, A. J. Pontes, J. C. Viana, A. Gaspar-Cunha, *Adv. Polym. Technol.* **2018**, 37, 429.
- [6] X. P. Dang, *Simul. Model. Pract. Theory* **2014**, 41, 15.
- [7] S. F. Walsh, *J. Reinf. Plast. Compos.* **1993**, 12, 769.
- [8] T. W. Simpson, J. D. Peplinski, P. N. Koch, J. K. Allen, *Eng. Comput.* **2001**, 17, 129.
- [9] G. G. Wang, S. Shan, *J. Mech. Des. Trans. ASME* **2007**, 129, 370.
- [10] A. D. Almási, S. Woźniak, V. Cristea, Y. Leblebici, T. Engbersen, *Neurocomputing* **2016**, 174, 31.
- [11] M. Altan, *Mater. Des.* **2010**, 31, 599.
- [12] B. Ozcelik, T. Erzurumlu, *J. Mater. Process. Technol.* **2006**, 171, 437.
- [13] N. C. Fei, N. M. Mehat, S. Kamaruddin, *ISRN Ind. Eng.* **2013**, 2013, 1.
- [14] C. Shen, L. Wang, Q. Li, *J. Mater. Process. Technol.* **2007**, 183, 412.
- [15] S. Kenig, A. Ben-David, M. Omer, A. Sadeh, *Eng. Appl. Artif. Intell.* **2001**, 14, 819.



- [16] W. C. Chen, P. H. Tai, M. W. Wang, W. J. Deng, C. T. Chen, *Expert Syst. Appl.* **2008**, *35*, 843.
- [17] Y. Xu, Q. W. Zhang, W. Zhang, P. Zhang, *Int. J. Adv. Manuf. Technol.* **2014**, *76*, 2199.
- [18] F. Yin, H. Mao, L. Hua, *Mater. Des.* **2011**, *32*, 3457.
- [19] Z. H. Che, *Comput. Ind. Eng.* **2010**, *58*, 625.
- [20] R. J. Bensingh, R. Machavaram, S. R. Boopathy, C. Jebaraj, *Meas. J. Int. Meas. Confed.* **2019**, *134*, 359.
- [21] C. Hopmann, J. Heinisch, H. Tercan, in *Int. Conf.*, ANTEC, SPE, Orlando, USA **2018**.
- [22] S. J. Pan, Q. Yang, *IEEE Trans. Knowl. Data Eng.* **2010**, *22*, 1345.
- [23] Y. Liao, H. Lee, K. Ryu, *IOP Conf. Ser. Materials Science and Engineering* **2018**, *324*, 012077.
- [24] N. Murata, S. Yoshizawa, S. I. Amari, *IEEE Trans. Neural Networks* **1994**, *5*, 865.
- [25] M. N. Marshall, *Fam. Pract.* **1996**, *13*, 522.
- [26] R. Y. Chang, W. H. Yang, *Int. J. Numer. Methods Fluids* **2001**, *37*, 125.
- [27] K. Choi, B. Koo, *Polymer* **2004**, *1*, 501.
- [28] N. E. H. Potente, H. Schulte, *Int. Polym. Process.* **1993**, *8*, 224.
- [29] H. Zhou, S. Shi, B. Ma, *Int. J. Adv. Manuf. Technol.* **2009**, *40*, 297.
- [30] J. C. Hart, *Vis. Comput.* **1996**, *12*, 527.
- [31] B. Golden, *Oper. Res.* **1976**, *24*, 1164.
- [32] D. Svozil, V. Kvasnieka, J. Pospichal, *Chemometr Intell Lab* **1997**, *39*, 43.
- [33] D. S. Shin, C. H. Lee, S. H. Kim, D. Y. Park, J. W. Oh, C. W. Gal, J. M. Koo, S. J. Park, S. C. Lee, *Powder Technol.* **2019**, *353*, 330.
- [34] R. Leaman, R. I. Doğan, Z. Lu, *Bioinformatics* **2013**, *29*, 2909.
- [35] J. Y. Kumar, S. Kumar Bhandare, *Int. J. Comput. Commun. Technol.* **2011**, *2*, 45.
- [36] I. P. Machado, A. Luísa Gomes, H. Gamboa, V. Paixão, R. M. Costa, *Inf. Process. Manag.* **2015**, *51*, 204.
- [37] T. Masters, *Pract. Neural Netw. Recipes C++* **2014**, *3*, 201.
- [38] D. Mishkin, J. Matas, arXiv preprint arXiv:1511.06422, **2015**.
- [39] J. Han, C. Moraga, *Lect. Notes Comput. Sci.* **1995**, *930*, 195.
- [40] G. E. Dahl, T. N. Sainath, G. E. Hinton, in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, IEEE, Vancouver, BC, Canada **2013**, pp. 8609–8613.
- [41] A. L. Maas, A. Y. Hannun, A. Y. Ng, *Proc. Icml* **2013**, *28*, 6.
- [42] D. P. Kingma, J. L. Ba, arXiv preprint arXiv:1412.6980, **2019**.
- [43] Y. N. Dauphin, H. De Vries, Y. Bengio, *Adv. Neural Inf. Process. Syst.* **2015**, 1504.
- [44] S. Ruder, arXiv preprint arXiv:1609.04747, **2016**.
- [45] R. Ward, X. Wu, L. Bottou, arXiv preprint arXiv:1806.01811, **2018**.
- [46] X. Glorot, Y. Bengio, in *Proc. 13th Int. Conf. on Artificial Intelligence and Statistics*, Vol. 9, Sardinia, Italy **2010**, p. 249.
- [47] L. Torrey, J. Shavlik, *Encycl. Sci. Learn.* **2012**, 3337.