 doi:10.20944/preprints202105.0312.v1

Article

The Steelmaking Process Parameter Optimization with a Surrogate Model Based on Convolutional Neural Networks and the Firefly Algorithm

Yung-Chun Liu[1], Ming-Huwi Horng*[,2], Yung-Yi Yang[3], Jian-Han Hsu[2] , Yen-Ting Chen[3], Yu-Chen Hung[4], Yung-Nien Sun[5], and Yu-Hsuan Tsai[5]

[1] Department of Biomedical Engineering, Da-Yeh University, Changhua, Taiwan,
[2] Department of Computer Science and Information Engineering, National Pingtung University, Pingtung, Taiwan
[3] Green Energy & System Integration Research & Development Department, China Steel Corporation, Kaohsiung, Taiwan
[4] Department of Intelligent Robotics, National Pingtung University, Pingtung, Taiwan
[5] Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan
*correspondence: horng@mail.nptu.edu.tw; Tel.:+886-8-7238700

Featured Application:

Abstract: High-strength low-alloy steels (HSLAs) are widely used in the structural body components of many domestic motor vehicles owing to their better mechanical properties and greater resistance. The real production process of HSLA steelmaking can be regarded as a model that builds on the relationship between process parameters and product quality attributes. A surrogate modeling method is used, and the resulting production process model can be applied to predict the optimal manufacturing process parameters. We used different methods in this paper including linear regression, random forests, support vector regression, multilayer perception, and a simplified VGG model to build such a surrogate model. We then applied three bio-inspired search algorithms, namely particle swarm optimization, the artificial bee colony algorithm, and the firefly algorithm, to search for the optimal controllable manufacturing process parameters. Through experiments on 9,000 test samples used for building the surrogate model, and 299 test samples for making the optimal process parameter selection, we found that the combination of a simplified VGG model and the firefly algorithm was the most successful at reaching a success rate of 100%—in other words, when the product quality attributes of all test samples satisfy the mechanical requirements of the end products.

Keywords: high-strength low-alloy steel; manufacturing process optimization; surrogate model; firefly algorithm; VGG model.

1. Introduction

Under pressure via fierce competition between steel companies, quality improvements in high-strength low-alloy steel (HSLA) products are constantly being pursued. Abnormal variations in upstream process parameters such as alloy composition might cause deviations in mechanical properties and thus lead to unsatisfactory quality and a high rejection rate. To reduce the rejection rate and to effectively improve the competitiveness of China Steel Corporation (CSC) products, we developed a dynamic process control system to predict and monitor the mechanical properties of products before they enter downstream production lines. If the predicted mechanical properties

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 May 2021 doi:10.20944/preprints202105.0312.v1

deviate too much from the usual level, the system can perform quality compensation by calculating and applying appropriate downstream process parameters and thereby meet the final quality requirements. This is done through a sequence of up- and downstream production lines, which include the crude making, hot rolling, cold rolling, and cold-rolled coating lines. Each line should meet quality-level requirements, or else the overall finished products will not achieve their required qualities. In the crude making process, iron ore is first reduced to iron by mixing it with coal/coke and limestone in a blast furnace (BF); then, the iron is converted into steel using a basic oxygen furnace (BOF). The hot-rolling is a mill process in which the steel is rolled at a temperature above its recrystallization temperature. When steel is heated past its recrystallization point, it becomes more malleable and can be properly formed and shaped. This also allows for the 2.1ability to produce larger quantities of steel. The steel is then cooled at room temperature, which "normalizes" it, eliminating the worry for stresses in the material arising when quenching or work-hardening. Cold-rolling steel allows for the creation of precise shapes. Since the process is performed at room temperature, the steel will not shrink as it cools, as it does in the hot-rolled process. The cold-rolled use plastic coating to protect the steel surface.

Although the CSC is a high-quality manufacturer of HSLA, a small number of non-conforming products cause great loss of finance and reputation. The steeling process parameter optimization is important to promote the quality of HSLA. Most manufacturing processes [1, 2] require parameterization to achieve their optimal cost, quality, and other properties. The number of process parameters considered usually exceeds 10 or even 100, and current approaches to their optimization require many expensive and complex experiments. Although some physically precise simulation models have been developed, such as the finite element method [3] and the Taguchi method [4], these need many hours or even days for computation. Another approach is the use of surrogate models, which can effectively decrease the number of simulations needed when applied to the problems of process parameter optimization.

The surrogate models is an easy-to-evaluate approach to construct high-fidelity product models [5–9] which were created by using decision tree, artificial neural network, radial basis function, kernel smoothing, and stochastic processes [10]. Surrogate-based optimization tries to search for the optimal process parameters based on an established surrogate model, and it requires an appropriate mechanism to do so. Genetic algorithms [11] and model-based self-optimizations [12] have been used to iteratively improve candidate process parameters.

Machine learning methods have been widely used in materials design and engineering, suach as the predictive maintenance of anodes [31], casting steel [32] and the predict of the steel's properties [33]. In this paper, we use different methods to build surrogate models for steelmaking process parameter optimization. These methods include linear regression (LR), random forests (RF) [18], support vector regression (SVR) [19], artificial neural networks (ANNs) [20, 37-39], and convolutional neural networks (CNNs) [21]. In addition, we use three different bio-inspired search algorithms — particle swarm optimization (PSO) [22], the artificial bee colony (ABC) algorithm [23], and the firefly algorithm (FA) [24]— to search for optimal process parameters.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 May 2021 doi:10.20944/preprints202105.0312.v1

We further developed a simplified 1D version of the original 2D Visual Geometry Group (VGG) 16 model [13, 14] to establish a surrogate model between the process parameters and the product quality attributes, as shown in Fig. 1. This 1D VGG model is called the simplified VGG (SVGG) model, and it can be regarded as a process for controlling HSLA product quality attributes through its parameter inputs.

To meet the product quality requirements of HSLA, we applied the PSO, ABC and firefly algoithm to search for the optimal adjustable process parameters and then compare their performance. In our experiments, 9000 samples (meeting the quality requirements) were used to train the five different methods for establishing the surrogate model, and 299 samples (not meeting the quality requirements) were used to evaluate three bio-inspired process optimization search algorithms. The FA ultimately produced the most optimal selection of adjustable process parameters. Our experimental results further showed that all the adjustable process parameters used in our test samples were successful in determining whether the corresponding product attributes met the mechanical requirements of the product.

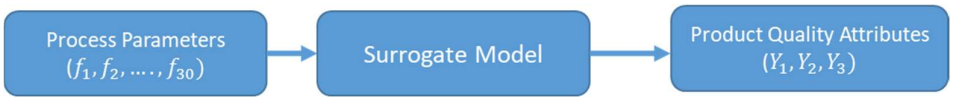

Figure 1: Surrogate model.

The main contributions of this paper are as follows:

1. We addressed the interesting problem of steelmaking process parameter optimization by proposing a simplified VGG model to build a surrogate model, and then compared it with four other machine-learning methods.
2. We applied three different algorithms—PSO, the ABC, and the FA—to search for optimal process parameters, and then evaluated their performance. Our experimental results demonstrated that the FA can achieve high performance and outperforms the other methods.

The remainder of this paper is organized as follows. Section 2 reviews the machine-learning methods and bio-inspired algorithms. Section 3 describes our proposed method called the simplified VGG model + Firefly algorithm to search for optimal process parameters. Section 4 presents our experimental results and discussions. Finally, conclusions and remarks are given in Section 5.

2. Related works

2.1 Surrogate Model

A production process can be represented as a function $\pi: X \to Y$, in which X denotes the process parameters and Y denotes the product attributes. This function maps process parameter configurations $x \in X$ to product attributes $y \in Y.$, and the map can be regarded as a surrogate model $\varphi: X \to Y$ based on all observations (X_i, Y_i). The construction of a surrogate model needs a fitness function to evaluate how well the model predictions match the observations. In general, we need an optimizer to obtain the optimal linear or nonlinear maps. In this study, we use linear regression, random forests, support vector

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 May 2021 doi:10.20944/preprints202105.0312.v1

regression, neural networks, and convolutional neural networks to build surrogate models, with process parameter optimization being dependent on the accuracy of these models. These five methods are described in the following subsections.

2.2 Survey of Machine Learning Method

2.2.1. Linear Regression

Linear regression (LR) is a simple and attractive method for building a surrogate model, as based on Eq. (1):

$$Y_j = \alpha_{1,j} f_1 + \alpha_{2,j} f_2 + \alpha_{3,j} f_3 + \cdots .. + \alpha_{30,j} f_{30} + \beta_j \quad (1)$$

where Y_j and f_i are the j-th product attribute and the i-th dimension of the process parameters, respectively. Particularly, the gradient descent method is used to decide the regression parameters $\alpha_{i,j}$ *and* β_j.

2.2.2 Random Forests

Random forests (RFs) is an ensemble learning method used for classification or regression that constructs a multitude of decision trees at training time and outputs the mode or mean predictions of those individual trees. In general, RFs outperform traditional decision tree methods because they can overcome the problem of overfitting. In the implementation of an RF algorithm, new training sample sets are randomly selected by replacing the original training set. Each training set is separated into two in-of-bag sets including one-third of the samples and one out-of-bag set from the remaining two-thirds of the samples. All samples in the out-of-bag set are collected into test samples, while samples from each of the in-of-bag sets are dependently built by decision tree induction into their own decision tree. This way, the regression results $R(x_i)$ of each test sample x_i are calculated by Eq. (2):

$$R(x_i) = \frac{1}{M} \sum_{m=1}^{M} y_m \quad (2)$$

where the M is the number of decision trees and y_m is the decision value of the m-th decision tree of test sample x_i.

2.2.3. Support Vector Regression

Support vector machines are intelligent statistical learning algorithms used for classification and regression. They solve regression problems through a nonlinear mapping function $\mu(x_i)$, which maps the original samples x_i to a feature space of a higher dimension, and then uses the linear regression method to compute the corresponding targets. Given a set of n training data $\{(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots., (x_n, y_n)\} \in R^n \times R$, where x_i is the input vector and y_i is its target, the following decision function can be defined:

$$f(x) = \omega \cdot \mu(x_i) + b \quad (3)$$

where $\cdot$ is the inner product, ω is the weight vector, and b is the bias parameter.

SVR applies the structured risk minimization principal to generate the above decision function [see Eq. (3)] by minimizing a regularized risk function as presented in Eqs. (4) and (5):

$$\text{Regularized risk} = \frac{C}{n} \sum_{i=1}^{n} L\big(y_i, f(x_i)\big) + \frac{1}{2} \|\omega\|^2 \quad (4)$$

where:

$$L\big(y_i, f(x_i)\big) = \begin{cases} |y_i - f(x_i)|, & if\, |y_i - f(x_i)| \geq \varepsilon \\ 0 & otherwise \end{cases} \quad (5)$$

In SVR, Vapnik's ε-insensitive loss function is used to measure empirical risk, where ε is the tube size, $\frac{1}{2}\|\omega\|^2$ is a regularization term used as a measure of flatness or complexity of the function, and C is a regularized constant that describes the trade-off between the empirical risk and the regularization term.

According to Wolfe duality and the saddle-point condition, the dual optimization problem of the aforementioned primal one is described by the following term:

$$\begin{aligned} &\max_{\alpha,\alpha^*} -\tfrac{1}{2}\textstyle\sum_{i,j=1}^{n}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*) \\ &\text{s.t. } \textstyle\sum_{i,j=1}^{n}(\alpha_i-\alpha_i^*)=0, \alpha_i,\alpha_i^* \in[0, C] \end{aligned} \tag{6}$$

The weight parameters are then described by $w=\sum_{i=1}^{n}(\alpha_i-\alpha_i^*)\,\mu(x_i)$, where α_i,α_i are nonnegative Lagrange multipliers, which can be obtained to solve the convex quadratic programming. Finally, based on Eq. (7) and the radial basis function (RBF) kernel trick, the decision function given by Eq. (3) has the following specific form:

$$f(x)=\sum_{i}^{n}(\alpha_i-\alpha_i^*)\,e^{\frac{-|x_i-x_j|^2}{2\sigma^2}}+b \tag{7}$$

where,

$$b=y_j-\sum_{i=1}^{n}y_i\alpha_i^*\,e^{\frac{-|x_i-x_j|^2}{2\sigma^2}} \tag{8}$$

with σ representing the kernel parameters and $j\in\{j<C\}$. Note there are three penalty parameters: C, the RBF kernel parameter, and the width of ε loss function.

2.2.4 Multilayer Perception

Multilayer perception (MLP) is a powerful class of data-driven function approximation algorithms which represent information through a hierarchy of features. They follow a simple ANN model, beginning with the input layer and ending with the final output layer, with intermediate layers known as hidden layers. By manipulating the number of hidden layers and the size of each, one can learn functions of arbitrary complexity. The input and output layer sizes are fixed, being determined by the dimensionality of the input feature and the output target. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP algorithms also utilize a supervised learning technique, i.e., backpropagation for training. We used an MLP algorithm with two hidden layers; each one has 100 nodes, the dimension of the input layer is 30 (the dimension of our process parameters) and involves three targets (the product attributes).

2.3 Bio-inspired Search Algorithms

Over the last decade, modeling the behavior of social insects such as ants and bees has been used as a way of solving search problems. There are many different bio-inspired search algorithms, including particle swam optimization, artificial bee colony, and the firefly algorithm, which have been widely used in numerical optimization [26], motion estimation [27], image thresholding [28,29], neural network parameter training [30], enhanced near field characteristic [34], simulation-driven spatial phase shifters [35], electromagnetic band-gap resonator antenna [36].

2.3.1 Particle Swarm Optimization

Particle swarm optimization (PSO) is a metaheuristic, stochastic, and population-based evolutionary optimization algorithm, and its standard form was initially developed by Kennedy Eberhart [22]. It searches for an optimal solution in its search space through the modeling of a swarm, where each particle in the swarm survives with a velocity and a position in the solution search space. The lower and upper bounds of each dimension of a particle are denoted in the algorithm by *lb* and *ub*. It improves on the best solution traversed so far by iteratively updating its velocity and position in the search space, as described by Eqs. (9) and (10):

$$v_i(t+1)=\omega v_i(t)+c_1r_1(p_i(t)-x_i(t)+c_2r_2(p_g(t)-x_i(t)) \tag{9}$$

$$x_i(t+1)=x_i(t)+v_i(\mathrm{t+1}) \tag{10}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 May 2021 doi:10.20944/preprints202105.0312.v1

where ω indicates the inertia weight, c_1 and c_2 are the learning rates, r_1 and r_2 are random numbers ranging from 0 to 1, and $p_i(t)$ and $p_g(t)$ represent the personal local best and the global best, respectively.

2.3.2 Artificial Bee Colony Algorithm

The artificial bee colony (ABC) algorithm, as proposed by Karaboga and Basturk [8], has recently become available and is a promising technique for solving real-world optimization problems. It models a colony of artificial bees, containing three different groups: employed bees, onlookers, and scouts. The employed bees carry information about food sources and share it in the dancing area of the hive. The onlookers wait in the dancing area to receive this probability information from the employed bees, and they use it to make decisions regarding the selection of a food source. The computation of this probability is based on the amount of food located at each source. The other kind of bee—scouts carries out random searches for new food sources. An employed bee becomes a scout when its food source is abandoned and becomes an employed bee again as soon as it finds a new food source. Therefore, each cycle of the ABC algorithm contains three steps. First, employed bees are sent to the known food sources and the amounts of nectar are calculated. After receiving that information, onlooker bees visit the food sources and provide updates. When the nectar at a food source is depleted, a scout is sent out to find a new food source.

Within the algorithm, the position of a food source x_i represents a candidate solution to the optimization problem, and the amount of nectar at this food source is denoted as its fitness (*fit*). In general, the number of employed bees or onlookers is equal to the number of food sources. Initially, the ABC algorithm randomly generates a distributed initial population of *K* solutions, denoted by $P = \{x_1, x_2, \dots, x_K\}$, where *K* denotes the number of employed bees or onlookers, and each solution x_i (for *i* = 1, 2,…, *K)* is a *D*-dimensional vector. During each execution cycle *C* (where *C* = 1, 2,…, MCN, the maximum cycle number), the population of solutions is subjected to the search processes of the employed bees, onlookers, and scouts. An employed bee modifies the possible solution as a function of the fitness value (amount of nectar) of the new solution (food source) by using Eq. (11):

$$v_{ij} = x_{ij} + \varphi(x_{ij} - x_{kj}) \tag{11}$$

where $k \in \{1,2,\dots,K\}$, but $k \neq I$ and $j \in \{1,2,\dots,D\}$ are randomly selected indexes, and φ is a random number between −1 and 1.

If the fitness value of the new solution v_i is greater than that of the previous solution x_i, then the employed bee simultaneously remembers the new solution and abandons the old one; otherwise, it will retain the location of the old one in its memory.

When all employed bees have finished their search process, they bring back to the onlookers the information they have on the positions and nectar amounts of all food sources. Each of the onlookers then decides on a particular food source to further call upon, according to a probability proportional to the amount of nectar at that food source. This probability p_i of selecting a food source z_i is determined using the following Eq. (12):

$$p_i = \frac{fit(z_i)}{\sum_{n=1}^{K} fit(z_n)} \tag{12}$$

In practice, each food source z_i sequentially generates a random number between 0 and 1. If this random number is less than the probability p_i, an onlooker is sent to the food source and produces a new solution based on Eq. (13):

$$v_{ij} = z_{ij} + \varphi(z_{ij} - z_{kj}) \tag{13}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 May 2021 doi:10.20944/preprints202105.0312.v1

where $k \in \{1,2,\ldots,K\}$, but $k \neq i$ and $j \in \{1,2,\ldots,D\}$ are randomly selected indexes, and φ is a random number between −1 and 1.

If the fitness value of the new solution is greater than the old one, the onlooker memorizes the new solution and shares this information with the other onlookers. Otherwise, the new solution is discarded. This process is repeated until all onlookers have been distributed to food sources. If the food source could be improved upon (as predetermined by a limiting value), then it is abandoned and the corresponding employed bee becomes a scout. This scout then goes on to discover a new food source to replace $theabandonedsolutionz_j$, as described by Eq. (14):

$$z_{ij} = z_{min}^{j} + \sigma\left(z_{max}^{j} - z_{min}^{j}\right) \tag{14}$$

where z_{min}^{j} and z_{max}^{j} are the lower and upper bounds of the *j*-th component of the solution, and σ is a random number ranging from −1 to 1. If the new solution is better than the $abandonedsolutionz_j$, the scout becomes an employed bee and the new solution is retained.

The search processes of the employed, onlooker, and scout bees are repeated until the execution cycle equals MCN. Of all the methods described so far, the best solutions with the largest fitness are outputted by this ABC algorithm.

2.3.3 Firefly Algorithm

The firefly algorithm (FA) was developed by Xin-She Yang at Cambridge University in 2008 [17]. It has three idealized rules: first, all fireflies are unisex, so each firefly is attracted to all other fireflies, regardless of their sex. Second, attractiveness is proportional to brightness—thus, for any two flashing fireflies, the least bright one will move toward the brighter one. If there is no brighter one, then that particular firefly will move randomly. To model firefly attractiveness, one should select any monotonically decreasing function of the distance $r_{i,j} = d\left(x_j, x_i\right)$ from the chosen (*j*-th) firefly x_j to the target (*i*-th) firefly x_i. This is described by Eqs. (15) and (16):

$$r_{i,j} = \left\|x_i - x_j\right\| \tag{15}$$

$$\beta \leftarrow \beta_0 e^{-\gamma\ i,j} \tag{16}$$

where β_0 is the attractiveness at $r_{i,j} = 0$ and γ is the light absorption coefficient at the source. The movement of firefly *i* when it is attracted to another, more attractive firefly *j* is determined by:

$$x_{i,k} \leftarrow (1-\beta)x_{i,k} + \beta x_{j,k} + u_{i,k} \tag{17}$$

$$u_{i,k} = \alpha\left(\text{rand1} - \frac{1}{2}\right) \tag{18}$$

If a particular firefly x_i is already the brightest (i.e., it is the one with the maximum fitness), then it will move randomly according to the following equations:

$$x_{i^{\max},k} \leftarrow x_{i^{\max},k} + u_{i^{\max},k},\ k = 1, 2, \ldots, D \tag{19}$$

$$u_{i^{\max},k} = \alpha\left(\text{rand2} - \frac{1}{2}\right) \tag{20}$$

where $\text{rand1} \approx U(0,1)$ and $\text{rand2} \approx U(0,1)$ are random numbers obtained from a uniform distribution.

The third rule is that the brightness of a firefly is affected or determined by the landscape of the fitness function $\varphi(\bullet)$. For maximization problems, the brightness *I* of a firefly at a particular location *x* can be chosen as a function $I(x)$ that is proportional to the value of the fitness function $\varphi(x)$.

3. Material and Methods

3.1 Materials and Experimental Setup

All experiments were performed on a PC with an Intel Core i5 3.30 GHz CPU, 8 GB of RAM, and a NVIDIA GeForce GTX 1060 GPU. All used machine-learning methods

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 May 2021 doi:10.20944/preprints202105.0312.v1

were implemented in simultaneously multitask learning and coded the programs by using Python language, and then individually were used to verify the performance of process parameters optimization using the four fold cross-validation method.

The 9,299 HSLA samples were collected from the China Steel Corporation (Kaohsiung, Taiwan) from 2016 to 2010. Each sample has a 30-dimensional process parameter and three corresponding product quality attributes., respectively. The 9,299 samples included the 9,000 training data samples (two of 10 samples as validation samples) which meets quality requirement and 299 false samples (as test samples) of no meeting requirements. In general, the false samples are difficult to collect since the CSC is a high-quality steel manufacturer. Each process parameter x includes temperature, steeling time, iron composition and so on. The product attribute Y includes three quality attributes, i.e., yield stress, tensile stress, and plastic strain ratio. The products of HSLA must meet reasonable quality ranges for these three attributes yield stress [0,185], tensile stress [270, ∞], and plastic strain ratio [39, ∞]. For easy identification, the process parameters are represented by $x_i = (f_{i1}, f_{i2},..., f_{i30})$ for $i = 1, 2,\ldots, 9{,}000$, and the product attributes are represented by $Y_i = (y_{i1}, y_{i2}, y_{i3})$ for training samples. However, the additional 299 test samples have only 25 fixed process parameters, with the other five being adjustable parameters denoted by f_{i2}, f_{i4}, f_{i6}, f_{i8}, and f_{i9} for the i-th test sample.

Table 1 shows the data for the five original training samples, where each includes the 30 dimensions of the process parameters, named f_1, $f_2,\ldots,f_{30}$, and the three product attributes of Y_1, Y_2, and Y_3. The five adjustable process parameters are restricted to positive integers. In total, 9,000 training samples were used to train the simplified VGG surrogate model, each including a complete set of process parameters and product attributes. However, five of the process parameters in each of 299 test samples were adjustable.

3.2 Simplified VGG-16 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a class of deep neural networks that have been widely applied in image and video recognition, recommender systems, image classification, image segmentation, medical image analysis, natural language processing, brain-computer interfaces, and financial time series. A CNN consists of an input layer, hidden layers, and an output layer. In any feed-forward neural network, middle layers are called "hidden" because their inputs and outputs are masked by the activation function and by the final convolution. In a CNN, these hidden layers include layers that perform convolutions. Typically, this includes a layer that does multiplication or finds a dot product, and its activation function is commonly referred to as a rectified linear unit (ReLU). This layer is followed by others, pooling layers, fully connected layers, and normalization layers.

VGG is a popular CNN model, containing 13 convolutional layers and three fully connected layers, commonly applied to image classification and pattern recognition. However, it is not directly suitable for building our surrogate model. In this paper, we instead use a simplified VGG model for this task. As the traditional VGG-16 model has a 2D layer structure consisting of many convolution layers, max-pooling layers, and fully connected layers, it is difficult to directly apply it to our steelmaking process-optimization problem with its 1D mapping. We therefore modified this VGG-16 model into a 1D structure with 10 layers (seven for 1D convolutional layer and three for fully connected layer). The seven convolutional layers generate powerful features and three fully connected layer use the extracted features to regression.

More precisely, the differences between the traditional VGG-16 model and our simplified VGG model are shown in Fig. 2. The simplified VGG model deletes the final softmax procedure and maps the process parameters to a single product attribute. It is a one-dimensional CNN model composed of seven convolutional layers and three fully connected layers. The convolutional layers extract the powerful features, and the fully connected layers match the product attributes used when establishing the surrogate model. Our trained simplified VGG first uses three blocks in which two or three convolutional procedures are followed by max-pooling to obtain feature maps with sizes of 3×32; they

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 May 2021 doi:10.20944/preprints202105.0312.v1

are then straightened into one-dimensional form. The sequential fully connected layers are applied in order to regress the product attribute of Y_1, Y_2, or Y_3.

In this method, the Adam algorithm [25] is used as an optimizer with a loss function equal to the mean square error (MSE) of the true outputs $\varphi(x_l)$ of the surrogate model and the desired outputs y_l, as defined in Eq. (21):

$$LossFunction = \frac{1}{M}\sum_{l=1}^{M}\left(y_l - \varphi(x_l)\right)^2 \tag{21}$$

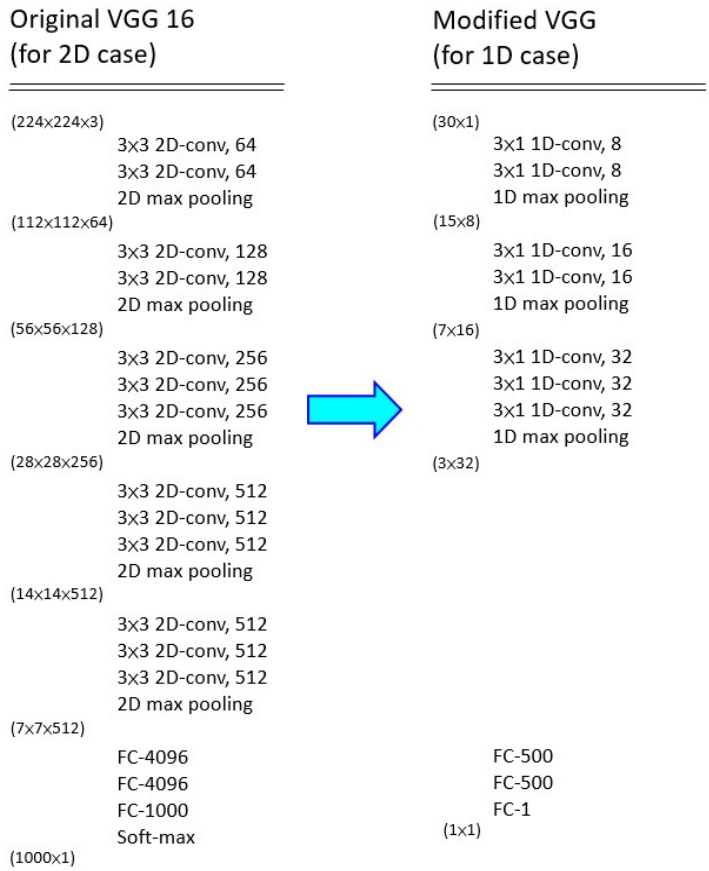

Figure 2: Traditional VGG and simplified VGG models.

3.2. *Process Parameter Optimization Using the Firefly Algorithm*

The process parameters of our test samples are divided into 25 fixed parameters and five adjustable ones, with the latter modified such that the product attributes meet the requirements. Therefore, we first assigned each adjustable process parameter to the i-th firefly solution x_i, with the structure depicted in Fig. 3. Next, we used the FA to search for the optimal solution $x_{best.}$, first, we initialized N firefly solutions, x_i, $i = 1,2,\ldots,N$, where each solution was updated using the FA.

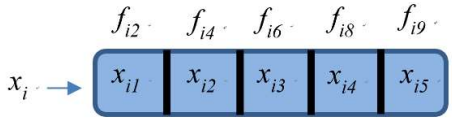

Figure 3: Structure of firefly solution

The fixed process parameters were integrated into each firefly solution x_i into the X_i. Using the built surrogate model $\varphi(X)$, the corresponding product quality attributes,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 May 2021 doi:10.20944/preprints202105.0312.v1

$Y_k = \varphi(X_i)$ for k = 1, 2, and 3, could be calculated. We then defined a fitness function Fitness function as the brightness of the FA for the purposes of searching for the optimum, as shown in Eq. (23).

$$u(X_i) = \frac{|Y_{i1} - m_{Y_1}|}{\sigma_{Y_1}} + \frac{|Y_{i2} - m_{Y_2}|}{\sigma_{Y_2}} + \frac{|Y_{i3} - m_{Y_3}|}{\sigma_{Y_3}} \quad (22)$$

$$Fitness(\mu(X_i)) = 1/(1 + u(X_i)) \quad (23)$$

where the product attributes Y_{i1}, Y_{i2}, and Y_{i3} are the outputs of the surrogate model $\varphi(x)$ with x as its input, and m_{Y_i} and σ_{Y_i} are the mean and the standard derivation of the i-th product attributes of the training data samples.

The steps of the proposed algorithm are described in detail as follows:

Step 1. Generate the initial solutions and the given hyper-parameters

In this step, the initial population of N solutions is generated, as denoted by $D = [x_1, x_2, \ldots, x_N]$, where $x_i = [f_{i2}, f_{i4}, f_{i6}, f_{i8}, f_{i9}]$ for the i-th firefly solution, and the values of x_i are assigned from between −1 and 1. This step assigns the parameters of the FA, which are σ, β_0, the MCN, and γ. The number of cycles l is also set to 0.

Step 2. Firefly movement

Here, each complete process is combined into X_i, and its fitness value $Fitness(\mu(X_i))$ is computed as the corresponding brightness of the firefly. For each firefly solution x_i, this step randomly chooses another brighter solution x_j and then moves toward it, according to the following equations:

$$r_{ij} = \|x_i - x_j\| = \sqrt{\sum_{k=1}^{5}(x_{ik} - x_{jk})^2} \quad (24)$$

$$x_{ik} = (1 - \beta)x_{ik} + \beta x_{jk} + u_{jk},\ \beta = \beta_0 e^{-\gamma r_{i,j}} \quad (25)$$

where $u_{jk} \sim U(0,1)$ is a random number ranged from 0 to 1, and x_{ik} is the k-th element of solution x_i.

Step 3. Select the current best solution

This step will pick the best solution from the solution set and represent it as x_{best}, as described by the following:

$$x_{best} = Argmax_{x_i}\{Fitness(\mu(x_i))\} \quad (26)$$

Step 4. Check the termination criterion

If the cycle number l is equal to the MCN, the algorithm is finished and will output the best solution x_{best}. Otherwise, l increases by one and the best solution x_{best} will randomly walk its position according to Eq. (13). It will then return to Step 2 and repeat the process, as described by:

$$x_{best,k} = x_{best,k} + u_k,\ k = 1, 2, \ldots, 5 \quad (27)$$

where u_k is a random number ranged from 0 to 1.

Without loss of generality, the PSO and ABC algorithms are also used to search for optimal parameters in a similar manner as the FA.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 May 2021 doi:10.20944/preprints202105.0312.v1

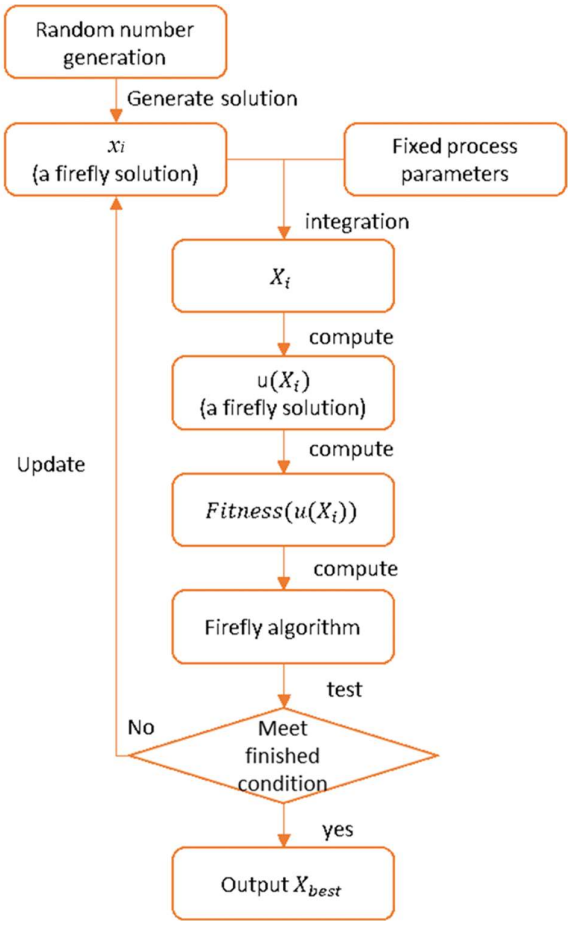

Figure 4: Flowchart of the FA used for optimizing the process parameters.

4. Results and Discussion

4.1 Training Mechanism of Uused ML methods

In experiments, the training parameters and strategy of the simplified VGG model are listed in Table 2, where each entry is optimal for the model. An initial learning rate of 0.001 and 250 training epochs were chosen in order to achieve adequate convergence. The Adam algorithm [25] was used as the optimizer, and the mean square error as the loss function. A batch size of 50 was selected, and the average training time was 232.47 s.

Table 2. Parameters Assigned in the Simplified VGG Model

Method	Batch Size	Training Epochs	Trainable Variables	Initial Learning Rate	Loss Function	Optimizer	Average Training Time (s)
Simplified VGG Model	50	250	360,661	0.001	MSE	Adam	232.47

In order to evaluate the performance of a trained simplified VGG model, we computed the mean absolute error (MAE) (see Eq. (26)) of the predicted and actual product attributes. The MAEs of Y_1, Y_2, and Y_3 are shown in Table 3, which reveals that the simplified VGG is the best, i.e., it is capable of producing strong correlations between

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 May 2021 doi:10.20944/preprints202105.0312.v1

the process parameters and the product attributes.

$$MAE_i = \frac{1}{299}\sum_{l=1}^{299}|Y_{il} - \varphi(X_l)| \quad i = 1,2,3 \tag{28}$$

where Y_{il} is i-th product attributes of l-th test sample.

Table 3. Mean Absolute Errors of the Five Surrogate Model Building Methods.

	Linear Regression	**Random Forests**	**Support Vector Regression**	**Multilayer Perception**	**Simplified VGG Network**
Yield Stress, Y_1.	5.052±0.09	3.890±0.11	4.057±0.09	4.156±0.09	3.781±0.08
Tensile Stress,Y_2	4.094±0.10	3.647±0.09	3.761±0.10	3.798±0.10	3.621±0.08
Plastic Strain Ratio, Y_3	1.032±0.02	0.954±0.02	0.993±0.02	0.948±0.02	0.946±0.02

(mean±standard derivation)

4.2 Experimental Results of the Process Parameter Optimizations

The FA was applied in order to search for the optimal adjustable process parameters of each of the test 299 samples, with the parameter settings of the algorithm given in Table 4. The FA is an iterative method, and a preassigned end condition is therefore needed. In this paper, we set the maximum number of iterations to 50 and the number of initial firefly solutions to 10. The average time spent searching for the optimal process parameters of each test sample was about 0.62 s. Additional details concerning the FA are shown in Fig. 4.

Table 4: Parameter Settings of the FA.

Parameter	**Value**
Attractiveness, β_0	1.0
Light Absorption Coefficient, γ	1.0
Number of Initial Firefly Solutions	10
MCL	50
σ	0.1

Table 5. Success Rates of Different Combinations When Evaluating the 299 TestSsamples

RF+PSO	**RF+ABC**	**RF+Firefly**	**SVGG+PSO**	**SVGG+ABC**	**SVGG+Firefly**
96.89%	97.99%	98.99%	97.32%	97.99%	100%

In our experiments, the 299 test samples were used to evaluate the performance of each method when searching for the optimal process parameters. The basis of evaluation is the success rate: how often the final product attributes fall in their reasonable quality ranges. Our experiments used the following combinations of methods: RF+PSO, RF+ABC, RF+Firefly, SVGG+PSO, SVGG+ABC, and SVGG+Firefly, the results of which are shown in Table 5. There we see that all the resulting test sample product attributes met their product quality requirements when the SVGG+Firefly combination was used, with its success rate of 100% clearly outperforming the other combinations.

4. Conclusions

The optimization of process parameters is an important problem in steelmaking. In general, surrogate models can be applied to simulate the manufacturing process, although there are many different methods available to build such models. In this paper, we propose a simplified VGG model for this purpose and then compare it with other machine-learning methods. Within the model, different algorithms (PSO, ABC, and FA) were applied to obtain the optimal process parameters. In our experiments, we evaluated different combinations of trained surrogate models and searching methods. Our proposed the simplified VGG model was demonstrated to be the best method compared with the linear regression, random forests, support vector regression and multi-layer perceptron. The firefly algorithm is an effective search mechanism to optimize the process optimization in the simplified VGG surrogate model. It achieved a success rate of 100% in the 299 test samples, and these perfect results revealed that it has the potential to be effectively applied in different areas of process parameters optimization of product manufacture. Furthermore, it will be also an interesting thing how to develop the more complex and effective CNN model to build the surrogate model for other applications of process parameter optimization.

Supplementary Materials: The following are available online at www.mdpi.com/xxx/s1, Figure S1: title, Table S1: title, Video S1: title.

Author Contributions: For research articles with several authors, a short paragraph specifying their individual contributions must be provided. The following statements should be used "Conceptualization: M.-H. H., Y.-N. S., and Y.-C. L.; methodology: M.-H. H.; software: Y.-C. L., M.-H. H., Y.-H. T., and Y.-C. H.; validation: Y.-N. S.; formal analysis: Y.-N. S. and J.-H. H.; investigation: Y.-N. S.; resources: Y.-Y. Y; data curation: Y.-T. C.; writing—original draft preparation: M.-H. H.; writing—review and editing: Y.-N. S.; visualization: M.-H. H.; supervision: Y.-N. S. All authors have read and agreed to the published version of the manuscript. All authors have read and agreed to the published version of the manuscript." Please turn to the CRediT taxonomy for the term explanation. Authorship must be limited to those who have contributed substantially to the work reported.

Funding: The authors thank the Ministry of Science and Technology, ROC (Project numbers MOST 109-2634-F-006-012 and MOST-108-2622-8-006-014) for their support of this work.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Acknowledgments: In this section, you can acknowledge any support given which is not covered by the author contribution or funding sections. This may include administrative and technical support, or donations in kind (e.g., materials used for experiments).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Eric, M., Stefanovic, M., Djordjevic, A., Stefanovic, N., Misic, M., Abadic, N., & Popović, P. (2016). Production process parameter optimization with a new model based on a genetic algorithm and ABC classification method. Advances in Mechanical Engineering, 8(8), 1–18.
2. Patil, V. D. & Sali, S. P. (2017). Process parameter optimization for computer numerical control turning on En36 alloy steel. 2017 International Conference on Nascent Technologies in Engineering (ICNTE), 1–5.
3. Hoole, J., Sartor, P, Booker, E.J., Cooper, J.E., Gogouvitis X., & Schmidt, R.K. (2020). Comparison of Surrogate Modeling Methods for Finite Element Analysis of Landing Gear Loads. Session: Surrogate Modeling for Uncertainty Quantification. AIAA 2020-0681.
4. Salah, U.H. & Raman, P. S. (2017). Taguchi-based design of experiments in training POD-RBF surrogate model for inverse material modelling using nanoindentation. Inverse Problems in Science and Engineering, 5(3), 363-381.
5. Pfrommer, J., Zimmerling, C., Liu, J., Kärger, L., Henning, F., & Beyerer, J. (2018). Optimisation of manufacturing process parameters using deep neural networks as surrogate models. Procedia CiRP, 72, 426–431.
6. Zhao, P., Zhou, H., Li, Y., & Li, D. (2010). Process parameters optimization of injection molding using a fast strip analysis as a surrogate model. The International Journal of Advanced Manufacturing Technology, 49, 949–959.
7. Forrester, A. I. & Keane, A. J. (2009). Recent advances in surrogate-based optimization. Progress in aerospace sciences, 45, 50–79.
8. Han, Z. H. & Zhang, K. S. (2012). Surrogate-based optimization. Real-world applications of genetic algorithms, 343–362.

9. Koziel, S., Ciaurri, D. E., & Leifsson, L. (2011). Surrogate-based methods. In Computational optimization, methods and algorithms (pp. 33–59). Springer, Berlin/Heidelberg.
10. Simpson, T. W., Poplinski, J. D., Koch, P. N., & Allen, J. K. (2001). Metamodels for computer-based engineering design: survey and recommendations. Engineering with computers, 17(2), 129–150.
11. Cook, D. F., Ragsdale, C. T., & Major, R. L. (2000). Combining a neural network with a genetic algorithm for process parameter optimization. Engineering applications of artificial intelligence, 13(4), 391–396.
12. Thombansen, U., Schuttler, J., Auerbach, T., Beckers, M., Buchholz, G., Eppelt, U., & Molitor, T. (2011). Model-based self-optimization for manufacturing systems. In 2011 17th International Conference on Concurrent Enterprising (pp. 1–9). IEEE.
13. Kiranyaz, S., Avci, O., Abdeljaber, O., Ince, T., Gabbouj, M., & Inman, D. J. (2019). 1D convolutional neural networks and applications: A survey. arXiv preprint:1905.03554.
14. Simonyan, K. & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint:1409.1556.
15. Łukasik, S. & Żak, S. (2009). Firefly algorithm for continuous constrained optimization tasks. In International conference on computational collective intelligence (pp. 97–106). Springer, Berlin/Heidelberg.
16. Yang, X. S. (2009). Firefly algorithms for multimodal optimization. In International symposium on stochastic algorithms (pp. 169–178). Springer, Berlin/Heidelberg.
17. Yang, X. S. (2010). Nature-inspired metaheuristic algorithms. Luniver press.
18. Breiman, Leo. (2001). "Random Forests". Machine Learning 45 (1), 5-32
19. Yan, C., Yin, Z., Shen, X., Mi, D., Guo, F., & Long D. (2020). Surrogate-based optimization with improved support vector regression for non-circular vent hole on aero-engine turbine disk. Aerospace Science and Technology, 96, 105332.
20. Nauyen, T. H., Nang D., & Paustian K., (2019). Surrogate-based multi-objective optimization of management options for agricultural landscapes using artificial neural networks, Ecological Modeling, 4, 1–13.
21. Xue, J., Xiang, Z., & Ou, G. (2021). Predicting single freestanding transmission tower time history response during complex wind input through a convolutional neural network based surrogate model. Engineering Structures, 233, 111859.
22. Jia, X. J., Liang, L., Yang, Z. L., & Yu, M. Y. (2021) Muti-parameters optimization for electromagnetic acoustic transduces using surrogated-assisted particle swarm optimizer. Mechanical Systems and Signal Processing 152 107337.
23. Sun, S., Sun, W., Liang, X., He, M., & Chen H. (2020). A modified surrogate-assisted multi-swarm artificial bee colony for complex numerical optimization. Microprocessors and Microsystems, 78, 103050.
24. Ewees, A. A., Al-qaness, M. A. A., & Abd Elaziz, M. (2021). Enhanced salp swarm algorithm based on firefly algorithm for unrelated parallel machine scheduling with set times. Applied Mathematical Modelling, 94, 285–305.
25. Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint:1412.6980.
26. Zhang, M., Long, D., Qin, T., & Yang, J. (2020). A Chaotic Hybrid Butterfly Optimization Algorithm with Particle Swarm Optimization for High-Dimensional Optimization Problems. Symmetry, 12(11), 1800.
27. Chao, C.-F., Horng, M.-H., & Chen, Y.-C. (2015) Motion Estimation Using the Firefly Algorithm in Ultrasonic Image Sequence of Soft Tissue. Computational and Mathematical Methods in Medicine, 343217.
28. Liu, W., Huang, Y., Ye, Z., Cai, W., Yang, S., Cheng, X., & Frank, I. (2020). Appl. Sci. 10, 3225.
29. Horng, M. H. (2011). Multilevel thresholding selection based on the artificial bee colony algorithm for image segmentation. Expert Systems with Applications, 38, 13785–13791.
30. Kaminski, M. (2019). Neural Network Training Using Particle Swarm Optimization – a Case Study. 24th International Conference on Methods and Models in Automation and Robotics (MMAR).
31. Lovrić, M., Meister, R., Steck, T., Fadljević, L., Gerdenitsch, J., Schuster, S., Schiefermüller, L., Lindstaedt, S., & Kern, R. (2020). Parasitic resistance as a predictor of faulty anodes in electro galvanizing: a comparison of machine learning, physical and hybrid models. Advanced Modeling and Simulation in Engineering Sciences (7), Article number: 46.
32. Cemernek, D., Cemernek, S., Gursch, H., Pandeshwar, A., Leitner, T., Berger, M., Klösch, G., & Roman Kern, R. (2021). Machine learning in continuous casting of steel: a state-of-the-art survey. Journal of Intelligent Manufacturing, 193.
33. Guo, S., Yu, J., Liu, X., Wang, C., & Jianga, Q. (2019). A predicting model for properties of steel using the industrial big data based on machine learning, Computational Materials Science, (160), 95-104.
34. Lalbkhsh, A., & Esselle, K.P. (2016). Directivity Improvement of a Fabry-Perot Cavity Antenna by enhancing Near Field Characteristic, 17th International Symposium on Antenna Technology and Applied Electromagnetics, ANTEM 2016.
35. Lalbakhsh; A., Afzal, M.U., & Esselle, K. (2016). Simulation-driven particle swarm optimization of spatial phase shifters, 2016 International Conference on Electromagnetics in Advanced Applications (ICEAA)
36. Lalbakhsh A., Afzal, M.U., & Esselle., K. P., (2016) Multi-objective Particle Swarm Optimization to Design a Time-Delay Equalizer Metasurface for an Electromagnetic Band-Gap Resonator Antenna, IEEE Antennas and Wireless Propagation Letters, (16), 912 – 915.
37. Jamshidi, M. B., Lalbakhsh, A., Alibeigi, N., Soheyli, M. R., Oryani, B., & Rabbani N. (2018), Socialization of Industrial Robots: An Innovative Solution to improve Productivity, 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON).
38. Jamshidi M. B., Alibeigi, N., Rabbani, N., Oryani, B., & Lalbakhsh. A. (2018). Artificial Neural Networks: A Powerful Tool for Cognitive Science, 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference.
39. Jamshidi, M. B., Lalbakhsh, A., Talla, J.,.., et, al. (2021), Deep Learning Techniques and COVID-19 Drug Discovery: Fundamentals, State-of-the-Art and Future Directions, Emerging Technologies During the Era of COVID-19 Pandemic, 9–31.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 May 2021 doi:10.20944/preprints202105.0312.v1

40. Lalbakhsh, P., Zaeri, B., & Lalbakhsh, A. (2013) An Improved Model of Ant Colony Optimization Using a Novel Pheromone Update Strategy. IEICE TRANSACTIONS on Information and Systems, 96(11),2309-2318

Table 1: Example data from the original training samples. The X and Y is the sets of process parameters and product quality attributes. The X contains the parameters of temperature, steeling time, iron composition and so on. The product attribute Y includes three quality attributes, i.e., yield stress, tensile stress, and plastic strain ratio.

Y1	Y2	Y3	FALSE	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15	X16	X17	X18	X19	X20	X21	X22	X23	X24	X25	X26	X27	X28	X29	X30	Y1_MIN	Y1_MAX	Y2_MIN	Y2_MAX	Y3_MIN	Y3_MAX
148	294	49	758	752	582	411	363	318	292	134	83	923	640	4.002723	0.00021	0.15	0.013	0.006	0.01	0.004	0.01	0.02	0	0.001	0.0027	0.003	0.039	0.033	0.032	0.0001	0.001	0.0001	0	185	270	0	38	0
146	294	50	748	747	574	403	361	318	288	150	124	915	643	3.99594	0.0014	0.12	0.017	0.005	0.01	0.005	0.01	0.01	0	0.001	0.0018	0.002	0.04	0.03	0.029	0.0001	0.001	0.0001	0	185	270	0	38	0
176	309	47	751	757	575	397	351	318	285	195	80	924	640	3.19883	0.0024	0.15	0.015	0.003	0	0.005	0.01	0.02	0	0.001	0.0027	0.003	0.044	0.036	0.035	0.0001	0	0.0001	0	185	270	0	37	0
139	288	51	754	754	575	397	351	313	291	195	80	928	647	3.198287	0.0012	0.14	0.014	0.003	0	0.006	0.01	0.02	0	0	0.0014	0	0.046	0.037	0.034	0.0001	0	0.0001	0	185	270	0	37	0
145	299	48	777	772	564	404	367	324	296	216	80	920	641	3.198613	0.0017	0.15	0.015	0.003	0	0.006	0.01	0.02	0	0.001	0.0016	0.003	0.042	0.041	0.04	0.0001	0.001	0.0001	0	185	270	0	37	0
150	302	49	768	757	580	411	358	320	292	194	107	922	648	3.500751	0.0014	0.16	0.015	0.005	0.01	0.005	0.01	0.02	0	0.001	0.0015	0.003	0.042	0.035	0.034	0.0001	0.001	0.0001	0	185	270	0	38	0
166	310	47	749	752	572	405	364	326	293	200	100	925	645	3.198607	0.0017	0.15	0.015	0.003	0	0.006	0.01	0.02	0	0.001	0.0016	0.003	0.042	0.041	0.04	0.0001	0.001	0.0001	0	185	270	0	37	0
153	303	47	748	752	572	402	360	325	294	206	80	923	647	3.198935	0.0017	0.15	0.015	0.003	0	0.006	0.01	0.02	0	0.001	0.0016	0.003	0.042	0.041	0.04	0.0001	0.001	0.0001	0	185	270	0	37	0
144	291	51	745	758	571	406	364	316	290	220	98	928	648	3.199251	0.0014	0.13	0.013	0.003	0	0.004	0.01	0.02	0	0.001	0.0016	0.003	0.045	0.037	0.036	0.0001	0.001	0.0001	0	185	270	0	37	0
148	295	49	758	752	575	401	361	315	288	220	88	927	641	2.600165	0.0014	0.13	0.012	0.004	0	0.005	0.01	0.01	0	0.001	0.0019	0.001	0.047	0.049	0.047	0.0002	0.001	0.0001	0	185	270	0	37	0
150	295	49	755	750	574	403	363	324	288	220	94	919	638	2.596573	0.0015	0.13	0.009	0.006	0	0.005	0.01	0.01	0	0.001	0.0026	0.001	0.047	0.04	0.039	0.0001	0.001	0.0001	0	185	270	0	37	0
154	304	47	762	764	575	406	365	320	293	204	82	921	645	3.000278	0.0014	0.17	0.018	0.004	0.01	0.004	0.01	0.02	0	0.001	0.0032	0.002	0.039	0.033	0.032	0.0001	0.001	0.0001	0	185	270	0	37	0
159	305	47	751	746	574	407	366	325	293	170	100	917	638	3.497113	0.0021	0.15	0.013	0.006	0.01	0.004	0.01	0.02	0	0.001	0.0027	0.003	0.039	0.033	0.032	0.0001	0.001	0.0001	0	185	270	0	37	0
162	308	47	756	749	574	407	365	323	289	170	96	914	648	3.50113	0.0014	0.16	0.014	0.005	0.01	0.005	0.01	0.02	0	0.001	0.0027	0.004	0.044	0.036	0.036	0.0002	0.001	0.0001	0	185	270	0	38	0
154	307	46	774	764	575	406	358	321	285	200	90	923	646	3.000014	0.0018	0.12	0.015	0.005	0	0.004	0.01	0.02	0	0	0.0016	0.001	0.043	0.038	0.035	0.0001	0.001	0.0001	0	185	270	0	37	0
150	297	49	762	756	575	406	365	315	290	200	90	921	641	3.199693	0.0017	0.13	0.012	0.006	0	0.004	0.01	0.02	0	0.001	0.0019	0	0.04	0.026	0.025	0.0002	0.001	0.0001	0	185	270	0	37	0
155	298	48	757	756	574	405	360	323	292	200	90	920	637	3.198027	0.0018	0.16	0.006	0.007	0	0.006	0.01	0.01	0	0.001	0.0043	0.002	0.048	0.051	0.05	0.0001	0.001	0.0001	0	185	270	0	37	0
149	300	49	764	762	575	405	357	314	290	200	90	922	645	3.200491	0.0017	0.14	0.014	0.005	0.01	0.005	0.01	0.02	0	0.001	0.0028	0.003	0.046	0.049	0.047	0.0002	0.001	0.0001	0	185	270	0	37	0
159	308	47	759	755	574	406	365	318	293	200	90	921	644	2.999876	0.0016	0.15	0.013	0.003	0	0.005	0.01	0.02	0	0.001	0.0039	0.002	0.044	0.031	0.03	0.0001	0	0.0001	0	185	270	0	37	0
152	302	49	764	762	575	407	364	323	291	175	80	915	647	3.498092	0.0021	0.15	0.013	0.006	0.01	0.004	0.01	0.02	0	0.001	0.0027	0.003	0.039	0.033	0.032	0.0001	0.001	0.0001	0	185	270	0	37	0
150	303	48	760	761	575	406	363	319	290	175	80	913	649	3.495116	0.0021	0.15	0.013	0.006	0.01	0.004	0.01	0.02	0	0.001	0.0027	0.003	0.039	0.033	0.032	0.0001	0.001	0.0001	0	185	270	0	37	0
146	300	49	759	756	574	406	356	315	290	175	129	922	648	3.498267	0.0017	0.14	0.014	0.005	0.01	0.005	0.01	0.02	0	0.001	0.0028	0.003	0.046	0.049	0.047	0.0002	0.001	0.0001	0	185	270	0	37	0
149	282	54	767	783	575	402	367	324	291	115	80	911	638	4.997924	0.0013	0.13	0.012	0.004	0	0.004	0.01	0.02	0	0.001	0.0024	0.004	0.042	0.039	0.038	0.0001	0.001	0.0001	0	185	270	0	39	0
156	291	52	776	763	580	412	358	318	294	111	121	908	636	5.000941	0.0013	0.13	0.012	0.004	0	0.004	0.01	0.02	0	0.001	0.0024	0.004	0.042	0.039	0.038	0.0001	0.001	0.0001	0	185	270	0	39	0
165	297	50	742	737	575	405	356	324	294	85	80	913	649	5.40558	0.0017	0.15	0.015	0.003	0	0.006	0.01	0.02	0	0.001	0.0016	0.003	0.042	0.041	0.04	0.0001	0.001	0.0001	0	185	270	0	39	0
153	279	55	735	737	575	407	354	318	292	85	80	910	639	5.400406	0.0009	0.17	0.012	0.003	0.01	0.005	0.01	0.02	0	0.001	0.0021	0.002	0.042	0.026	0.026	0.0001	0.001	0.0001	0	185	270	0	39	0
151	279	55	740	739	574	404	367	320	298	84	88	905	649	5.395242	0.0009	0.17	0.012	0.003	0.01	0.005	0.01	0.02	0	0.001	0.0021	0.002	0.042	0.026	0.026	0.0001	0.001	0.0001	0	185	270	0	39	0
160	292	52	745	736	575	403	363	327	293	85	80	915	653	5.400981	0.0022	0.15	0.014	0.005	0	0.005	0.01	0.02	0	0.001	0.0012	0.001	0.045	0.045	0.043	0.0002	0.001	0.0001	0	185	270	0	39	0
159	289	53	748	739	575	404	357	314	292	88	80	913	634	5.403232	0.0013	0.14	0.015	0.005	0	0.004	0.01	0.02	0	0.001	0.0017	0.002	0.044	0.039	0.038	0.0001	0.001	0.0001	0	185	270	0	39	0
160	292	52	742	742	573	404	354	313	297	88	80	915	641	5.4021	0.0022	0.15	0.014	0.005	0	0.005	0.01	0.02	0	0.001	0.0012	0.001	0.045	0.045	0.043	0.0002	0.001	0.0001	0	185	270	0	39	0
160	288	53	752	749	574	404	360	321	299	88	81	915	639	5.400731	0.0009	0.12	0.016	0.002	0	0.005	0.01	0.02	0	0	0.0015	0.002	0.043	0.031	0.03	0.0001	0.001	0.0001	0	185	270	0	39	0
156	288	52	750	748	574	408	371	321	295	88	80	908	640	5.39677	0.0018	0.16	0.006	0.007	0	0.006	0.01	0.01	0	0.001	0.0043	0.002	0.048	0.051	0.05	0.0001	0.001	0.0001	0	185	270	0	39	0
164	295	52	740	751	574	403	368	319	299	88	80	916	640	5.406957	0.0017	0.16	0.014	0.003	0	0.004	0.01	0.02	0	0	0.0011	0.002	0.043	0.033	0.032	0.0001	0.001	0.0001	0	185	270	0	39	0
159	290	53	741	750	574	395	366	332	289	88	79	912	647	5.395891	0.0017	0.13	0.014	0.005	0	0.004	0.01	0.02	0	0.001	0.0027	0.001	0.046	0.038	0.037	0.0001	0.001	0.0001	0	185	270	0	39	0
152	288	51	748	742	572	398	352	320	276	85	94	910	641	4.797426	0.0014	0.11	0.013	0.004	0.01	0.004	0.01	0.02	0	0.001	0.0026	0.001	0.045	0.041	0.039	0.0001	0.001	0.0001	0	185	270	0	38	0
145	295	51	793	736	575	397	349	319	277	175	81	921	648	3.495929	0.0017	0.13	0.012	0.006	0	0.005	0.01	0.02	0	0.001	0.0015	0.001	0.051	0.043	0.041	0.0001	0.001	0.0001	145	195	270	0	43	52
145	295	51	756	757	574	400	340	311	274	170	80	919	648	4.001006	0.0014	0.14	0.013	0.004	0.01	0.004	0.01	0.02	0	0.001	0.0017	0.001	0.05	0.037	0.036	0.0001	0.001	0.0001	0	185	270	0	38	0
158	294	51	779	772	580	401	337	317	274	140	80	916	650	3.999133	0.0021	0.12	0.011	0.003	0	0.004	0.01	0.02	0	0	0.002	0.001	0.041	0.029	0.026	0.0001	0.001	0.0001	0	185	270	0	38	0
164	293	53	792	786	548	384	373	320	277	136	81	910	652	5.400156	0.0014	0.17	0.018	0.004	0.01	0.004	0.01	0.02	0	0.001	0.0032	0.002	0.039	0.033	0.032	0.0001	0.001	0.0001	0	185	270	0	39	0
158	298	50	770	764	590	426	352	312	287	134	80	910	630	4.801073	0.0014	0.17	0.018	0.004	0.01	0.004	0.01	0.02	0	0.001	0.0032	0.002	0.039	0.033	0.032	0.0001	0.001	0.0001	0	185	270	0	38	0
146	285	53	742	748	575	401	362	324	297	140	80	913	632	4.80204	0.0016	0.12	0.013	0.004	0	0.004	0.01	0.01	0	0.001	0.0016	0.001	0.042	0.038	0.037	0.0001	0.001	0.0001	0	185	270	0	38	0
157	293	50	746	755	574	397	359	318	286	131	80	925	637	4.200436	0.0024	0.15	0.015	0.003	0	0.005	0.01	0.02	0	0.001	0.0027	0.003	0.044	0.036	0.035	0.0001	0	0.0001	0	185	270	0	38	0
148	293	52	751	743	576	399	365	318	294	126	80	916	644	4.803132	0.0009	0.13	0.016	0.003	0	0.004	0.01	0.02	0	0.001	0.0017	0.001	0.045	0.029	0.027	0.0002	0.001	0.0001	0	185	270	0	38	0
154	296	51	740	743	574	397	364	322	293	118	80	912	626	4.799883	0.0014	0.16	0.013	0.002	0.01	0.007	0.01	0.02	0	0.001	0.0009	0.003	0.045	0.035	0.035	0.0001	0.001	0.0001	0	185	270	0	38	0
149	291	53	746	744	574	398	359	322	288	118	80	914	638	4.7971	0.0009	0.13	0.016	0.003	0	0.004	0.01	0.02	0	0.001	0.0017	0.001	0.045	0.029	0.027	0.0002	0.001	0.0001	0	185	270	0	38	0
146	287	52	744	747	574	398	354	318	284	118	80	916	638	4.800219	0.0009	0.13	0.016	0.003	0	0.004	0.01	0.02	0	0.001	0.0017	0.001	0.045	0.029	0.027	0.0002	0.001	0.0001	0	185	270	0	38	0
151	293	51	745	746	574	397	355	316	286	118	80	916	645	4.799561	0.0009	0.13	0.016	0.003	0	0.004	0.01	0.02	0	0.001	0.0017	0.001	0.045	0.029	0.027	0.0002	0.001	0.0001	0	185	270	0	38	0
154	293	51	742	745	574	397	365	320	294	118	80	912	644	4.79961	0.0017	0.12	0.012	0.007	0.01	0.004	0.01	0.01	0	0.001	0.0017	0.003	0.042	0.04	0.039	0.0001	0.001	0.0001	0	185	270	0	38	0
152	292	50	743	743	575	397	364	323	293	118	80	911	633	4.798247	0.0017	0.12	0.012	0.007	0.01	0.004	0.01	0.01	0	0.001	0.0017	0.003	0.042	0.04	0.039	0.0001	0.001	0.0001	0	185	270	0	38	0
156	298	53	743	742	575	398	359	320	290	118	80	911	631	4.798938	0.0016	0.12	0.013	0.004	0	0.004	0.01	0.01	0	0.001	0.0016	0.001	0.042	0.038	0.037	0.0001	0.001	0.0001	0	185	270	0	38	0
166	306	48	747	744	575	396	364	321	293	110	80	914	639	4.797888	0.0015	0.14	0.019	0.007	0.01	0.004	0.01	0.02	0	0.001	0.0021	0.002	0.039	0.04	0.04	0.0001	0.001	0.0001	0	185	270	0	38	0
158	297	50	744	744	575	397	362	318	292	110	80	910	639	4.803582	0.0017	0.15	0.015	0.003	0	0.006	0.01	0.02	0	0.001	0.0016	0.003	0.042	0.041	0.04	0.0001	0.001	0.0001	0	185	270	0	38	0
161	302	49	742	745	574	397	357	318	291	110	80	916	635	4.796422	0.0015	0.14	0.019	0.007	0.01	0.004	0.01	0.02	0	0.001	0.0021	0.002	0.039	0.04	0.04	0.0001	0.001	0.0001	0	185	270	0	38	0
156	294	50	748	746	575	397	354	321	286	110	80	907	645	4.796799	0.0018	0.16	0.006	0.007	0	0.006	0.01	0.01	0	0.001	0.0043	0.002	0.048	0.051	0.05	0.0001	0.001	0.0001	0	185	270	0	38	0
156	295	49	741	745	574	397	361	320	284	110	80	917	635	4.797822	0.0021	0.14	0.012	0.005	0	0.01	0.01	0.02	0	0.001	0.0022	0.003	0.043	0.033	0.03	0.0001	0	0.0001	0	185	270	0	38	0
154	294	50	748	744	575	397	365	318	294	110	80	910	644	4.796692	0.0018	0.16	0.006	0.007	0	0.006	0.01	0.01	0	0.001	0.0043	0.002	0.048	0.051	0.05	0.0001	0.001	0.0001	0	185	270	0	38	0
156	291	49	741	742	575	397	364	324	294	110	80	913	633	4.803288	0.0008	0.13	0.013	0.006	0	0.004	0.01	0.02	0	0	0.0021	0.001	0.045	0.033	0.032	0.0001	0	0.0001	0	185	270	0	38	0
153	292	51	746	744	575	397	360	322	290	110	80	907	638	4.797641	0.0013	0.14	0.017	0.006	0	0.004	0.01	0.02	0	0	0.0022	0	0.042	0.03	0.028	0.0001	0.001	0.0001	0	185	270	0	38	0
164	306	48	747	746	575	398	352	315	287	110	80	915	643	4.798532	0.0015	0.14	0.019	0.007	0.01	0.004	0.01	0.02	0	0.001	0.0021	0.002	0.039	0.04	0.04	0.0001	0.001	0.0001	0	185	270	0	38	0
154	297	49	743	745	572	396	356	322	286	115	80	908	645	4.801856	0.0021	0.15	0.013	0.006	0.01	0.004	0.01	0.02	0	0.001	0.0027	0.003	0.039	0.033	0.032	0.0001	0.001	0.0001	0	185	270	0	38	0
154	295	51	742	743	574	398	364	319	295	120	80	912	642	4.801859	0.0008	0.13	0.013	0.006	0	0.004	0.01	0.02	0	0	0.0021	0.001	0.045	0.033	0.032	0.0001	0	0.0001	0	185	270	0	38	0
161	296	49	744	744	574	398	355	320	285	120	80	910	634	4.794713	0.0021	0.14	0.012	0.006	0.01	0.004	0.01	0.01	0	0	0.0035	0.001	0.045	0.04	0.038	0.0001	0.001	0.0001	0	185	270	0	38	0