sensors

Article

Hyper-Parameter Optimization of Stacked Asymmetric Auto-Encoders for Automatic Personality Traits Perception

Effat Jalaeian Zaferani [1], Mohammad Teshnehlab [1], Amirreza Khodadadian [2,*], Clemens Heitzinger [3,4], Mansour Vali [1], Nima Noii [5] and Thomas Wick [2]

[1] Electrical & Computer Engineering Faculty, K. N. Toosi University of Technology, Tehran 19967-15433, Iran
[2] Institute of Applied Mathematics, Leibniz University of Hannover, 30167 Hannover, Germany
[3] Institute of Analysis and Scientific Computing, TU Wien, 1040 Vienna, Austria
[4] Center for Artificial Intelligence and Machine Learning (CAIML), TU Wien, 1040 Vienna, Austria
[5] Institute of Continuum Mechanics, Leibniz University of Hannover, 30823 Garbsen, Germany
* Correspondence: khodadadian@ifam.uni-hannover.de

Abstract: In this work, a method for automatic hyper-parameter tuning of the stacked asymmetric auto-encoder is proposed. In previous work, the deep learning ability to extract personality perception from speech was shown, but hyper-parameter tuning was attained by trial-and-error, which is time-consuming and requires machine learning knowledge. Therefore, obtaining hyper-parameter values is challenging and places limits on deep learning usage. To address this challenge, researchers have applied optimization methods. Although there were successes, the search space is very large due to the large number of deep learning hyper-parameters, which increases the probability of getting stuck in local optima. Researchers have also focused on improving global optimization methods. In this regard, we suggest a novel global optimization method based on the cultural algorithm, multi-island and the concept of parallelism to search this large space smartly. At first, we evaluated our method on three well-known optimization benchmarks and compared the results with recently published papers. Results indicate that the convergence of the proposed method speeds up due to the ability to escape from local optima, and the precision of the results improves dramatically. Afterward, we applied our method to optimize five hyper-parameters of an asymmetric auto-encoder for automatic personality perception. Since inappropriate hyper-parameters lead the network to over-fitting and under-fitting, we used a novel cost function to prevent over-fitting and under-fitting. As observed, the unweighted average recall (accuracy) was improved by 6.52% (9.54%) compared to our previous work and had remarkable outcomes compared to other published personality perception works.

Keywords: big five personality traits; cultural algorithm; deep learning; hyper-parameter optimization; personality perception

check for updates

Citation: Jalaeian Zaferani, E.; Teshnehlab, M.; Khodadadian, A.; Heitzinger, C.; Vali, M.; Noii, N.; Wick, T. Hyper-Parameter Optimization of Stacked Asymmetric Auto-Encoders for Automatic Personality Traits Perception. *Sensors* **2022**, *22*, 6206. https://doi.org/10.3390/s22166206

Academic Editor: Jing Tian

Received: 17 July 2022
Accepted: 16 August 2022
Published: 18 August 2022

1. Introduction

Whether deep or shallow, the operation of artificial neural networks (ANNs) depends on their hyper-parameters and parameters [1–3]. Certain variables of ANNs are called hyper-parameters, such as the number of layers [2], or control the training process, such as the learning rate [4]. In contrast, the trainable variables pertaining to layer connections and tuned during the training process, which are weights and biases, are called parameters [5–7]. Although parameter tuning may yield good results, it does not yield notable results without hyper-parameter tuning (HPT).

The importance of HPT became more manifest than before with the development of deep learning algorithms. Deep learning is a type of machine learning (ML) technique with diverse hyper-parameters that severely affect its performance [8–10]. Since HPT is an arduous task and requires data and network knowledge [11,12], it is often acquired by empirical methods (trial-and-error), which is time-consuming and does not guarantee significant results in terms of efficient algorithms and overall cost complexity. Therefore, studies based on applying optimization methods to ANNs have gained attention.

Accordingly, the usage of optimization algorithms is divided into three groups, as follows:

1. HPT with the classical method and parameter optimization [13–16]: The fine-tuning of weights and biases (parameters) can provide useful information about the problem, but their size and initial value rely on HPT. Moreover, the number of parameters in deep neural networks (DNNs) and high dimensional datasets is enormous, and calculating the optimum value of these parameters is complicated, not easily implemented, and requires computational systems with remarkable capabilities.
2. Hyper-parameter and parameter optimization [17–19]: Adaptive hyper-parameters are obtained by parameter training. The critical disadvantage is that with each possible vector of hyper-parameters, the parameters must be optimized, which causes runtime errors in the computational system and requires expensive training and large storage capacity to save the best parameters value over epochs. Additionally, all possible combinations of hyper-parameters are computationally infeasible. Hence, this method is not applicable in a large model such as deep learning [20,21].
3. Hyper-parameter optimization (HPO) and parameter tuning with back-propagation [4,11,22]: The main drawback is that although optimization methods are efficient in finding global optima, the gradient may vanish when back-propagating. As a result, not all network parameters are tuned well, which impacts results [23]. To tackle the poor-tuning process of deep neural network parameters, an asymmetric auto-encoder (Asy_{AE}) was presented in our previous work for automatic personality perception (APP) from speech [24]. We showed that Asy_{AE} could improve the model outcome results compared with conventional auto-encoders by semi-supervised training of parameters, and it can be effectively employed in deep learning. However, the stacked asymmetric auto-encoder (SA_{AE}) hyper-parameters were chosen by trial-and-error, which was time-consuming, and two personality traits achieved lower accuracy than other prior research [24].

Thus, the aims of the present work were to (1) propose a novel optimization method based on cultural evolution and parallel computing, (2) obtain the near-optimal values of hyper-parameters of SA_{AE}, and (3) classify five personality traits.

The rest of the article is organized as follows. In Section 2, some related works of HPO in deep learning and APP are explained. In Section 3, the dataset is introduced, and the summary of the feature extraction method is presented in Section 4. The new optimization method is proposed in Section 5. The simulation results of the new method, which is applied to three benchmark functions of finding global optima, are presented in Section 6. In addition, this section discusses the outcomes of applying the proposed method to SA_{AE} for automatic personality perception classification.

2. Related Works

Given that this article examines HPO methods in order to find a proper one to optimize the hyper-parameters of SA_{AE} for automatic personality trait perception, the related works section is divided into two parts. The focus of the first part is on recently published methods of neural network hyper-parameter tuning, regardless of the application in which it is used. Thus, the works related to the investigation of HPO in ML are summarized in the first part. Since the aim of our research was HPT of SA_{AE} to classify five personality traits from speech, the second part is related to studying HPO in machine learning methods applied in the field of personality trait perception.

2.1. Hyper-Parameter Tuning in ML

Deep learning hyper-parameter types are vast and can be divided into three groups: integer, real, and categorical. The integer group consists of variables such as the number of layers (whether hidden or convolutional) [25], the number of neurons [8], the size of the kernel [26], the number of kernels [27], batch size, pooling size, and number of maximum epochs [9]. The real group includes the learning rate [25], dropout rate [25], regularization factor [25], network weight initialization [5], and momentum [4]. The categorical group comprises activation function type [8] and optimization method [8].

Considering that a change in the value of each hyper-parameter changes the values of the neural network parameters that affect the output of the network, and also that examination of any possible combination of hyper-parameters is time-consuming, expensive and practically impossible, studies have investigated the effect of adjusting and optimizing some of the most important hyper-parameters.

In this regard, the article in [4] employed the HPO method for bearing fault diagnosis in mechanical equipment. Parallel computing was used to find hyper-parameters of the deep belief network (DBN). The learning rate and momentum were optimized, while other hyper-parameters were predefined and kept constant. Additionally, Wu Deng et al. used quantum-inspired differential evolution (DE) to optimize DBN parameters. Results showed an improvement in global search and avoiding premature convergence for fault classification [28].

The numbers of hidden neurons as a hyper-parameter and of the weights/biases as parameters were optimized in a feed-forward ANN by Gray wolf optimizer in [18]. Feed-forward ANNs (not back-propagation) were used because adjusted parameters were achieved by the optimization method.

Y. Peng et al. proposed an HPO method based on a fuzzy system in [8]. They optimized the number of hidden layers and the number of neurons in each layer of a DNN. The activation function type and optimization method, including Genetic Algorithms (GA), Bayesian search, grid search, random search, and quasi-random search, were selected automatically during HPO. For preventing over-fitting, the dropout technique was used. The proposed method was tested in three rainfall prediction datasets.

The authors of [29] suggested a distributed particle swarm optimization (PSO) for the HPO of a convolution neural network (CNN). They were concerned about the time-consuming population search based on distributed PSO, and parallel computing was employed to speed up the algorithm. They optimized the number and size of the kernels, the type of pooling (max or average) for two convolutional layers, the activation function type in convolutional layers, the number of neurons, learning rate, and the dropout rate of the fully connected layers.

Time-series prediction of congestion in highway systems based on long short-term memory (LSTM) was investigated in [9]. To obtain the proper model and structure, the authors recommended an HPO method by applying the Bayesian optimization (BO) method. Five hyper-parameters were automatically obtained, including learning rate, the number of hidden layers, the number of neurons in each layer, batch size, and dropout rate.

The intention of [25] was to examine the robustness of one HPO method over six benchmarks, contrary to other works that designed an algorithm that fit one problem. In other work, the authors used BO as an old HPO method in CNN [1] and applied four strategies to alleviate the drawbacks of BO. They tuned the hyper-parameters of two convolutional layers and two fully connected layers in this way.

In [26], an intuitive architecture design using GA was proposed for CNN. The obtained model was evaluated on a CNN with a single convolutional layer and a fully connected layer. Additionally, some hyper-parameters, including maximum epochs, batch size, initial learning rate, regularization, and momentum were optimized by PSO to prepare a CNN for expression recognition in [30].

Since the success of neural networks depends on their structure, the article in [31] proposed a micro-canonical optimization algorithm for overcoming large parameter spaces and optimizing hyper-parameters of a CNN. Hyper-parameters were the number of convolution layers, activation function type, batch size, pooling type, and dropout rate. The method was evaluated by six image recognition datasets and exhibited accuracy improvement.

State-of-health estimation and remaining usable life prediction in battery prognosis were examined in [32] by a deep convolution neural network. The authors addressed hyper-parameter tuning that affected DNN performance. They improved the algorithm by using the BO method.

Anjir A. Chowdhury et al. concentrated on the role of hyper-parameter optimization in the performance and reliability of deep learning outcomes [33]. They compared several HPO algorithms to obtain better validation accuracy in DNNs and concluded that most of them are computationally expensive. Finally, a greedy approach-based HPO algorithm was proposed for enabling faster computing on edge devices for on-the-fly learning applications. The VGG and ResNet architectures were used, and their hyper-parameters such as epochs, number of hidden layers, number of units per layer, activation function, dropout rate, batch size, and learning rate were optimized.

The Gray wolf optimization was employed to optimize the parameters of the kernel extreme learning machine to realize a hyperspectral image classification method in [34].

2.2. Automatic Personality Perception

In psychology, the big five inventory (BFI) is a well-known theory of personality with five traits, including openness to experience (Ope.), conscientiousness (Con.), extraversion (Ext.), agreeableness (Agr.), and neuroticism (Neu.). These traits are in an individual simultaneously by different scores and can be measured by a BFI questionnaire in general [35,36].

Due to the importance of personality in daily life, computer science researchers have investigated personality trait identification by multimodal media (audio, text, video, image) recently. Here, we focus on studies structured by deep learning methods.

A multimodal approach for perceiving personality traits was proposed by employing well-known deep structures (ResNet-v2-101 and VGGish) [37]. The LSTM network for using temporal information was added at the end. The authors optimized only the learning rate, while other hyper-parameters were configured manually. It is clear that the structure of the mentioned deep methods is fixed, and the weights and biases are pre-trained. Therefore, HPO or HPT does not tune according to each dataset in these networks.

Given the fact that personality traits can influence appearance, MobileNetv2 and ResNeSt50 networks were employed in [38] to extract facial features and classification. Results specified that one pre-trained network such as MobileNetv2 is inappropriate for classifying all five personality traits. It indicated that each trait must classify by a specific model, which means different hyper-parameters are necessary. However, the authors did not mention it directly and applied a combination of two pre-trained deep networks to build a complex deep model.

Onno Kampman et al. examined feature extraction and the classification of five personality traits by applying a one-dimensional CNN to a raw audio dataset. The HPT of the deep network containing regularization factors and kernel size was performed manually [39].

One of the personality detection applications is discovering interpersonal communication skills. Article [40] investigated this aspect from a video interview using a semisupervised CNN in which HPT was performed by trial-and-error. The authors concentrated on video processing, and a fixed hyper-parameter set to utilize for all traits.

The study in [41] analyzed the acoustic and lexical features of a speech signal that were affected by BFI traits. Additionally, it designed six models based on recurrent neural networks for classifying those traits. Hyper-parameters such as hidden size, learning rate, batch size, and dropout percentage were defined, but tuning them was not discussed.

3. Dataset

The SSPNet speaker personality corpus (SPC) is a well-known automatic personality perception dataset introduced in 2010. This dataset originally contained 640 recorded speech signals of 322 native French speakers. There is one speaker in each clip recorded for 10 s. Due to the studies on the effect of mental factors on speech signals [42], the collected clips were emotionally neutral, and to confirm that lexical content did not affect the personality scores, evaluators who were foreign to the French language were selected. Therefore, eleven assessors who did not understand French evaluated each clip based on the BFI questionnaire. The average score of these assessors was considered as the final score for each clip. Hence, five scores were obtained for each clip [43].

Although the SPC dataset has been applied in several works and is a proper dataset for comparison with the new methods, the number of samples is uses is low to train the enormous number of parameters of a DNN. This important challenge was addressed in our previous work [24], and we proved that the sample size of speech signals could be enhanced with data augmentation methods based on a spectrogram so that the prosodic content of speech could be preserved. Data augmentation is a popular technique to expand the size of the dataset artificially and is widely used in image processing. However, using this technique in speech is not as easy as using an image. In other words, we needed to choose transformations that maintain the speaker's personality, and we had to be confident that such manipulations in the spectrogram do not interfere with the extracted features related to personality traits. In this regard, frequency masking and time warping were selected as data augmentation methods, and the number of clips increased up to 640,000. For more details, please see [24].

4. Feature Extraction

Despite DNN's ability to perform automatic feature extraction from raw speech signals, deep learning methods have been generally applied to manually extracted hand-crafted audio features. This is mainly because of the large volume of data required for deep learning methods to outperform. Nevertheless, building a dataset with large available labeled samples is costly, time-consuming, and laborious work in the automatic personality perception field, which restricts various methods. Therefore, previous studies have used handcrafted features for the DNN input [44].

These handcrafted features contain 6373 statistical features extracted from 130 low-level descriptions (LLD) [45]. Table 1 contains 65 LLD features and 65 first derivatives of LLD (ΔLLD), for a total of 130 LLD features.

For the LLD feature extraction process, each clip was divided into 60 ms frames with a 20 ms overlap in the time domain and 20 ms frames with a 10 ms overlap in the frequency domain by the Opensmile2.3 toolkit.

Table 1. The 130 LLD features, including 65 LLD and 65 ΔLLD features [46].

4 Energy Related LLD	**Group**
Sum of Auditory Spectrum (Loudness)	Prosodic
Sum of RASTA-Style Filtered Auditory Spectrum	Prosodic
RMS Energy, Zero-Crossing Rate	Prosodic
55 Spectral LLD	**Group**
RASTA-Style Auditory Spectrum, Bands 1–26 (0–8 kHz)	Spectral
MFCC 1-14	Cepstral
Spectral Energy 250–650 Hz, 1 k–4 kHz	Spectral
Spectral Roll Off Point 0.25, 0.50, 0.75, 0.90	Spectral
Spectral Flux, Centroid, Entropy, Slope, Harmonicity	Spectral
Spectral Psychoacoustic Sharpness	Spectral
Spectral Variance, Skewness, Kurtosis	Spectral
6 Voicing Related LLD	**Group**
F0 (SHS & Viterbi Smoothing)	Prosodic
Probability of Voicing	Sound Quality
Log. HNR, Jitter (Local, Delta), Shimmer (Local)	Sound Quality

Table 1. *Cont.*

4 Energy Related LLD	**Group**
Mean Values	
Arithmetic Mean A^{Δ}, B, Arithmetic Mean of Positive Values $^{A\Delta, B}$,	
Root-Quadratic Mean, Flatness	
Moments: Standard Deviation, Skewness, Kurtosis	
Temporal Centroid A^{Δ}, B	
Percentiles	
Quartiles 1–3, Inter-Quartile Ranges 1–2, 2–3, 1–3,	
1%—tile, 99%—tile, Range 1–99%	
Extrema	
Relative Position of Maximum and Minimum, Full Range (Maximum–Minimum)	
Peaks and Valleys ^{A}	
Mean of Peak Amplitudes,	
Difference of Mean of Peak Amplitudes to Arithmetic Mean,	
Peak to Peak Distances: Mean and Standard Deviation,	
Peak Range Relative to Arithmetic Mean,	
Range of Peak Amplitude Values,	
Range of Valley Amplitude Values Relative to Arithmetic Mean,	
Valley-Peak (Rising) Slopes: Mean and Standard Deviation,	
Peak-Valley (Falling) Slopes: Mean and Standard Deviation	
Up-Level Times: 25%, 50%, 75%, 90%	
Rise and Curvature Time	
Relative Time in which Signal is Rising,	
Relative Time in which Signal has Left Curvative	
Segment Lengths ^{A}	
Mean, Standard Deviation, Minimum, Maximum	
Regression A^{Δ}, B	
Linear Regression: Slope, Offset, Quadratic Error,	
Quadratic Regression: Coefficients a and b, Offset c, Quadratic Error	
Linear Prediction LP Analysis Gain (Amplitude Error), LP Coefficients 1–5	
^{A} Functionals applied only to energy related and spectral LLDs (group A)	
^{B} Functionals applied only to voicing related LLDs (group B)	
$^{\Delta}$ Functionals applied only to ΔLLDs	
$^{\Delta}$ Functionals not applied only to ΔLLDs	

5. Proposed Method

This section is divided into two parts. In the first part, we thoroughly describe the new optimization method mathematically. In order to apply our optimization method to the SA_{AE}, we had to address several problems. The second part deals with this issue and its solution.

5.1. The Proposed Optimization Method

HPO of deep learning is a time-consuming task in practice that depends on the network depth, the size of parameters, processor system, and optimization algorithm speed [5]. Applying HPO to deep learning is challenging. It can be (1) the unsupervised learning of most deep learning methods that causes trouble for optimization and imperfect tuning of parameters [47], (2) a large model with enormous trainable parameters that lead the processing system to runtime errors [5,8], and (3) an intricate search space created by different types of hyper-parameter domains (categorical, continuous, and integer value), causing inherent computational complexity [5]. A larger search space gives rise to a longer search time.

Parallel evaluation can partly reduce optimization time [48], and culture speeds up the population's evolution more than chromosomes (each chromosome represents a solution in the population space) [49]. Accumulated experience that is potentially accessible to all individuals is called culture, which is used in problem-solving activities [50]. The knowledge extracted by identifying patterns in the population's problem-solving experiences influences the generation of new solutions [51]. Therefore, the combination of CA and parallel computing can facilitate the discovery of the search space [52]. In this regard, researchers are interested in combining CA with other optimization algorithms. Sun et al. combined a cultural algorithm and two PSO populations and shared their belief space. It indicated that sharing knowledge of belief space can improve performance by avoiding local optima [53]. A single population and multi-population based on CA was proposed in [54]. A PSO population-based method with interactive belief space was introduced by [49]. A hybrid evolutionary optimization method coupling CA with GAs was defined in [55]. Fuzzy operations were employed to exchange individuals between belief space and population space in [56].

From this perspective, we proposed a four-island approach based on the parallel evaluation and CA.

Although CA and parallel computing can perform better than the basic optimization algorithms [57], they do not provide enough convergence speed alone for deep learning. Thus, three driving force factors were applied to population space for creating interactive space between four island population spaces. Creating interactive population space causes interactive belief space, which can determine the direction and step size faster than traditional optimization methods. In this regard, our proposed method is called the multi-island interactive cultural (MIC) algorithm.

The MIC method is illustrated in Figure 1. In this method, control parameters are configured firstly. The initial population X[m, D] is generated randomly in the feasible space. The variable m indicates the population size (the number of chromosomes or individuals), and D is chromosome dimension (the number of genes).

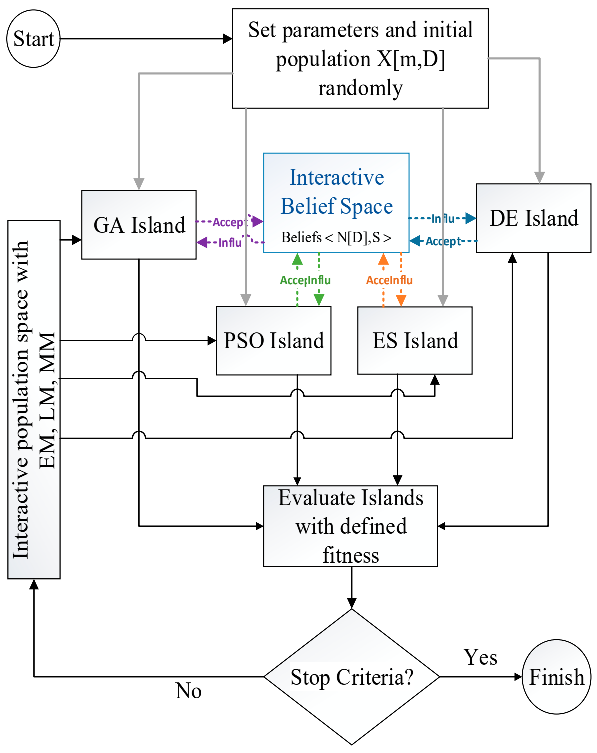

Figure 1. Flowchart of the MIC algorithm.

After preparing the random initial population, it transfers into the four islands in parallel (gray lines): GA, PSO, DE, and evaluation strategy (ES). The GA and PSO are the optimization algorithms widely applied to HPO studies in deep learning [1,8]. GA is far more successful in complex networks such as CNNs, but eliminates previous information by changing the population every iteration [50]. PSO shares information between the particles and is popular on the smaller networks [29]. The DE algorithm is utilized in optimization problems due to the high convergence speed and low control parameters when searching global optima. It is suitable for nonlinear search spaces [28]. The ES is less popular among the global optimization algorithms because it is a simple mutation-selection method, but it is helpful in making small changes [48]. It should be noticed that in the first iteration, the population of the four islands is the same.

The four islands were evaluated individually and in parallel. Then, some individuals of each island were randomly selected to transfer into an interactive belief space (InBS) through an acceptance function (colored arrows). Here, the acceptance function was 25% of the best individuals of each island. So, the belief space size was y[m, D].

The InBS consisted of normative (N[D]) and situational knowledge (S) of all islands. Knowledge of different islands in the belief space causes the chromosomes to move away from unwanted regions and get closer to the optimal points by using different experiences faster than previously published works. InBS can be used effectively to prune the population space.

Normative knowledge represents the range of the best solutions by determining the upper and lower bands of each gene of a chromosome and is used to influence the direction of the search efforts within the promising ranges. In other words, it computes the range of each gene that leads the individual to a good solution.

The offspring affected by normative knowledge are generated by Equation (1) as

$$y_{p+i,j}^{t+1} = \begin{cases} y_{i,j}^{t} + \left|(u_j^t - l_j^t) * N(0,1)\right| & \text{if } \; y_{i,j}^t < l_j^t, \\ y_{i,j}^{t} - \left|(u_j^t - l_j^t) * N(0,1)\right| & \text{if } \; y_{i,j}^t > u_j^t, \\ y_{i,j}^t + \beta\left|(u_j^t - l_j^t)\right| * N(-1,1) & \text{otherwise,} \end{cases} \tag{1}$$

where u_j is the upper and l_j is the lower band of InBS for jth gene, respectively, β is a constant value, t is the current iteration, and $N(0,1)$ is the normal distribution.

For each gene, the structure contains the upper band (u_j^t), the lower bound (l_j^t), the upper band value (U_j^t), and the lower bound value (L_j^t), which are obtained by Equations (2)–(5), respectively.

$$L_j^{t+1} = \begin{cases} f(y_i) & \text{if } \; y_{i,j} \le l_j^t \;\text{ Or }\; f(y_i) < L_j^t, \\ L_j^t & \text{otherwise,} \end{cases} \tag{2}$$

$$l_j^{t+1} = \begin{cases} y_{i,j}^t & \text{if } \; y_{i,j}^t \le l_j^t \;\text{ Or }\; f(y_i^t) < L_j^t, \\ l_j^t & \text{otherwise,} \end{cases} \tag{3}$$

$$U_j^{t+1} = \begin{cases} f(y_i) & \text{if } \; y_{i,j} \ge u_j^t \;\text{ Or }\; f(y_i) < U_j^t, \\ U_j^t & \text{Otherwise,} \end{cases} \tag{4}$$

$$u_j^{t+1} = \begin{cases} y_{i,j}^t & \text{if } \; y_{i,j}^t \ge u_j^t \;\text{ Or }\; f(y_i^t) < U_j^t, \\ u_j^t & \text{otherwise,} \end{cases} \tag{5}$$

where $y_{i,j}$ is the jth gene in the ith individual of InBS, and the $f(y_i)$ is the value of the individual y_i calculated by the fitness function. A fitness function (loss function) evaluated individuals of each island separately. The problem description determines the fitness function.

The situational knowledge, as seen in Equation (6), adjusts the mutation step size relative to the distance between the current best individual and the other individuals. The greater the distance between ith individual, y_i, and the current best individual, the greater the step size and vice versa.

Updating the situational knowledge adds the InBS's best individual to the situational knowledge if it outperforms the current best individual, as described in Equation (6).

Here, y^t_{best} is the best individual in the InBS at iteration t. The influence rule can be represented with Equation (7) (for i = 1, . . . , m and j = 1, . . . , D).

$$< E_1^{t+1}, E_2^{t+1}, \ldots, E_e^{t+1} >= \begin{cases} < y^t_{best}, E_2^t, \ldots, E_e^t > & \text{if } f(y^t_{best}) > f(E_1^t), \\ < y^t_{best} > & \text{if change detected}, \\ < E_1^t, E_2^t \ldots, E_e^t > & \text{otherwise}, \end{cases} \quad (6)$$

$$y^{t+1}_{p+i,j} = \begin{cases} y^t_{i,j} + \left| (y^t_{i,j} - E^t_{i,j}) \cdot N_{i,j}(0,1) \right| & \text{if } y^t_{i,j} < E^t_j, \\ y^t_{i,j} - \left| (y^t_{i,j} - E^t_{i,j}) \cdot N_{i,j}(0,1) \right| & \text{if } y^t_{i,j} > E^t_j, \\ y^t_{i,j} + \beta \left| (y^t_{i,j} - E^t_{i,j}) \right| \cdot N_{i,j}(0,1) & \text{otherwise}, \end{cases} \quad (7)$$

where E_j is the jth gene in the best individual, β is a constant factor, N(0, 1) is the normal distribution, and $y_{p+i,j}$ is the offspring of the individual $y_{i,j}$.

After updating InBS with new generations, some individuals are transferred into each island population space by influence function. There is no doubt that the individuals of InBS contain the knowledge of all of the islands. This is the ability of the proposed method. Various studies have shown that the efficiency of optimization methods is altered for different problems. In other words, choosing an optimization method for a problem is a challenge that some researchers consider as a kind of hyper-parameter that needs to be tuned. Hence, 25% of the best individuals of InBS were replaced with 25% of the worst population on each island. Offspring generation processing is started in each island separately and evaluated through fitness function.

If the algorithm reaches the stopping criterion, the process will be stopped. Otherwise, interactive population space is created by three driving forces in order to promote cooperation among the islands and increase diversity.

The three driving-force methods are named the elitism method (EM), merge method (MM), and lambda method (LM).

In interactive population space, all individuals of each island are considered. In EM, the best individuals with size *m* are preserved and replaced with the old population on each island. As we use this method, the populations of the next generation for each island are the same. This driving force method forces the four basic algorithms to create interactive space only by the best individuals of four islands.

In MM, after considering all individuals of each island, a random number a, $a \in (0, 1)$, is produced. The $a \times m$ $(a * sizeofpopulation)$ of the best individuals are merged with $(a - 1) \times m$ of the old population on each island. It is clear that each island has a unique new population in this interactive space.

In LM, two of the islands are selected randomly, according to two random numbers μ, μ ∈ (0, 1), and λ, λ ∈ (0, 1), representing emigration and immigration, respectively. The random numbers of individuals based on μ and λ of each island indicate which individuals can immigrate to and emigrate from another random island. This method forces islands to cooperate with the best individual and the worst one to create interactive space.

Due to the interaction and sharing of individuals among the four islands, if one algorithm traps in local optima, others can lead MIC into global optima because the result is not dependent on a single algorithm. This feature allows the MIC to be used for various global optimization problems to escape local optima efficiently.

The MIC strategy is presented step by step below (Algorithm 1).

Algorithm 1: Implementation of MIC

```
Step 1: Set the MIC parameters randomly.
Step 2: Generate the initial population randomly.
Step 3: Transfer 25% of the best individuals of each island into InBS (Accept).
Step 4: Update Belief space whith Equations (1)–(7).
Step 5: Transfer 25% of offspring into each island (Influ).
Step 6: If stop criterion < $\zeta$
            Stop algoriyhm.
        Else
            Go to Step 7.
Step 7: Create Interactive population space by using the following three methods:
            EM: $m$ of the best individuals of four islands are selected and replaced with an old
population.
            MM: The $a \times m$ $(a * sizeof population)$ of the best individuals are selected and merged
with $(a - 1) \times m$, which is obtained from the old population in islands.
            LM: According to two random numbers, $\mu$ and $\lambda$, some individuals of a random island
can immigrate to and emigrate from another random island.
Step 8: Go to Step 3.
```

5.2. Stacked Asymmetric Auto-Encoder HPO Using MIC

Since our work aimed to obtain the SA_{AE} near-optimal structure, a brief overview of this method is presented below.

(1) **Stacked asymmetric auto-encoder**

The Asy_{AE} is a semi-supervised DNN that poses the curse of dimensionality. The schematic of the Asy_{AE} is illustrated in Figure 2.

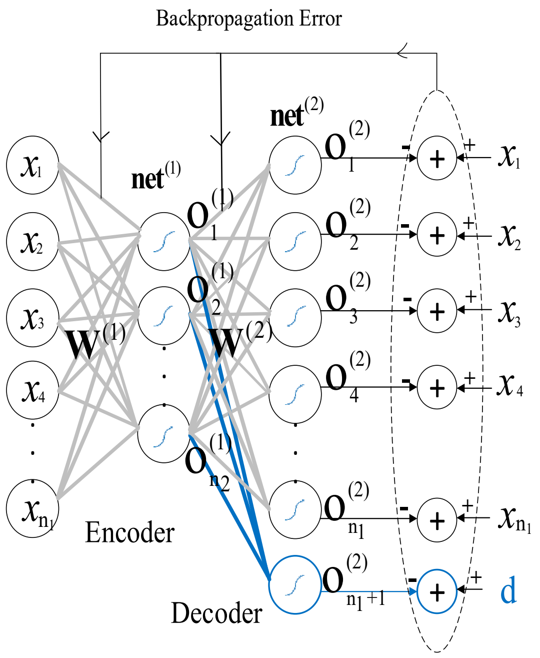

Figure 2. Schematic of the asymmetric auto-encoder [24].

In this type, one neuron is added in the decoder part of the conventional auto-encoder with the desired value of the problem, which is the studied personality score in our field. The symmetry of the encoder and decoder parts is disrupted by this single neuron and made asymmetric.

The feed-forward equations of the $\mathrm{Asy_{AE}}$ are similar to the conventional one as follows. For representing encoder and decoder layers, superscripts of 1 and 2 were used, respectively.

$$\mathbf{net}^{(1)} = \mathbf{W}^{(1)}\mathbf{X}, \tag{8}$$

$$\mathbf{O}^{(1)} = \mathrm{f}\left(\mathbf{net}^{(1)}\right), \tag{9}$$

where $\mathbf{W}^{(1)}$ indicates the encoder weight matrix, $\mathbf{X}$ displays the input matrix, $\mathbf{O}^{(1)}$ is the encoder output matrix, and f is the activation function.

$$\mathbf{net}^{(2)}\mathbf{=}\mathbf{W}^{(2)}\mathbf{O}^{(1)}, \tag{10}$$

$$\mathbf{O}^{(2)} = \mathrm{f}\left(\mathbf{net}^{(2)}\right), \tag{11}$$

where $\mathbf{W}^{(2)}$ and $\mathbf{O}^{(2)}$ are the weight and output matrixes of the decoder layer, respectively.

The error back-propagation related to the encoder and decoder weights matrixes is calculated by Equation (12).

$$\mathrm{E} := \frac{1}{\mathrm{k}}\sum_{\mathrm{i}=1}^{\mathrm{k}} \log(\cosh(\mathrm{e_t})), \tag{12}$$

where $\mathrm{e_t}$ is the error vector of $\mathrm{Asy_{AE}}$ at time t, which is described by Equation (13), and k is the neuron size of decoder layer output.

$$\mathbf{e}_\mathrm{t} : \mathbf{=}\mathbf{d}_\mathrm{t} - \mathbf{o}_\mathrm{t}{}^{(2)}. \tag{13}$$

The desired output vector at time t is presented by $\mathbf{d}_\mathrm{t}$, which belongs to the matrix $\mathbf{D}$. It is the desired output matrix of $\mathrm{Asy_{AE}}$, which is produced by the combination of desired labels and $\mathrm{Asy_{AE}}$ input.

$$\mathbf{D} = \begin{bmatrix} \mathrm{x_{11}} & \mathrm{x_{12}} & \cdots & \mathrm{x_{1n_0}} & \mathcal{L} \\ \mathrm{x_{21}} & \mathrm{x_{22}} & \cdots & \mathrm{x_{2n_0}} & \mathcal{L} \\ \vdots & \vdots & \vdots & \vdots & \\ \mathrm{x_{m1}} & \mathrm{x_{m2}} & \cdots & \mathrm{x_{mn_0}} & \mathcal{L} \end{bmatrix}.$$

Here, $\mathrm{x_{ij}}$ is the $\mathrm{Asy_{AE}}$ input matrix element, and $\mathcal{L}$ is the desired label of the problem. A stacked asymmetric auto-encoder is a result of putting several $\mathrm{Asy_{AE}}$s together.

(2) **Optimizing some hyper-parameters of a stacked asymmetric auto-encoder**

Given the fact that the number of DNN hyper-parameters is significantly large, the simultaneous optimization of all of them complicates the computation and requires high-performance computing systems. Hence, we compromised between MIC and expertise for calculating the six critical DNN hyper-parameters as follows:

1. number of neurons in each hidden layer
2. learning rate value
3. initial parameters
4. number of hidden layers
5. maximum epoch of network training
6. preventing over-fitting and under-fitting

For HPO of $\mathrm{SA_{AE}}$, the following principles come after. Figure 3 illustrates the flowchart of the proposed method in detail.

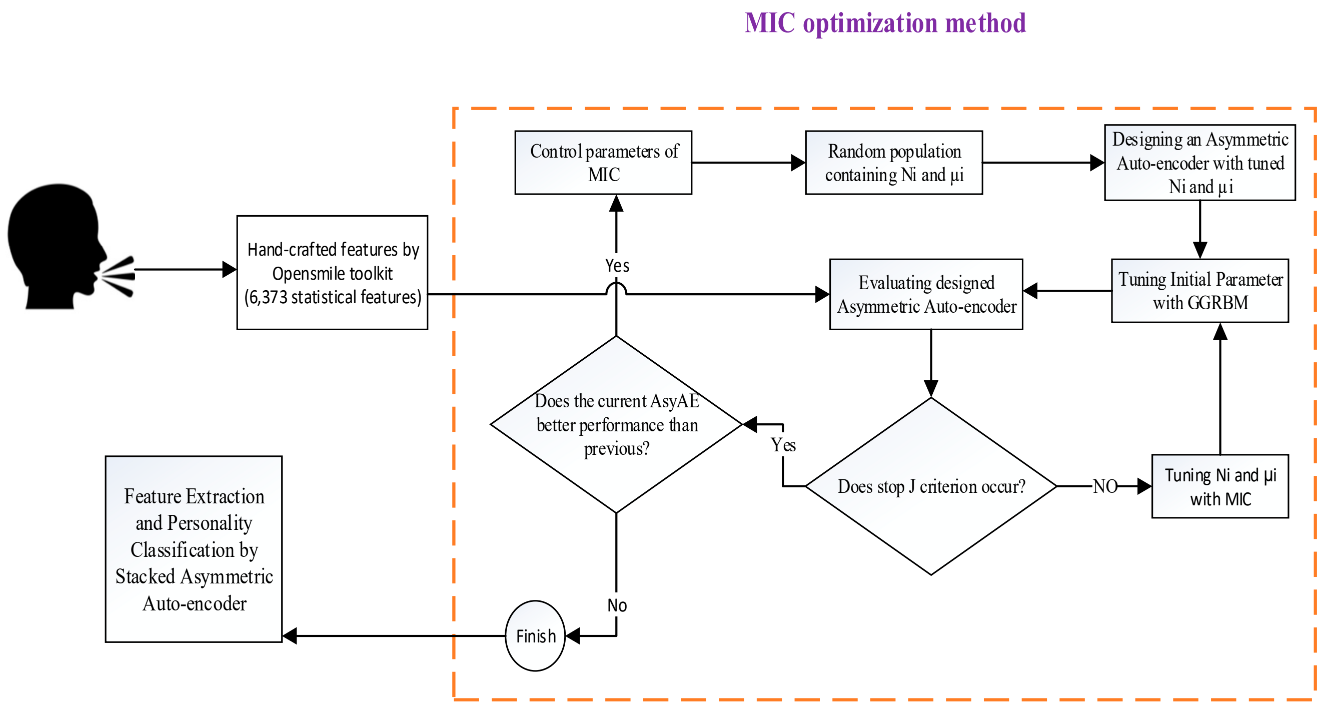
MIC optimization method
Control parameters of MIC
Random population containing Ni and µi
Designing an Asymmetric Auto-encoder with tuned Ni and µi
Hand-crafted features by Opensmile toolkit (6,373 statistical features)
Yes
Evaluating designed Asymmetric Auto-encoder
Tuning Initial Parameter with GGRBM
Does the current AsyAE better performance than previous?
Yes
Does stop J criterion occur?
NO
Tuning Ni and µi with MIC
Feature Extraction and Personality Classification by Stacked Asymmetric Auto-encoder
No
Finish

Figure 3. Flowchart of SA_{AE} hyper-parameter optimization.

Determining the number of neurons in each hidden layer: In our work, N_i indicates the number of neurons in the i^{th} hidden layer that will be optimized by the MIC method. So, the first variable of MIC is N_i, which is an integer value, $N_i \in [1, m]$ where m value is equal to the input size of Asy_{AE}. It forces the Asy_{AE} to be an incomplete network. It means the encoder layer has fewer neurons than the input layer.

Determining the learning rate in each hidden layer: μ_i specifies the learning rate in the i^{th} hidden layer, which will be optimized by the MIC method. Therefore, the second variable of the MIC population is a real value between zero and one, $\mu_i \in (0, 1)$. It should be mentioned that we set the decimal digit of μ_i equal to 5 to examine its effect on SA_{AE} performance.

Initial value of trainable parameters: Although deep learning methods have good performance in various problems, they are complicated tasks. Because there are huge factors that strongly influence them, one of the critical factors is initialization.

The DNN parameters need a starting point in the feasible area to be trained. The proper initial parameters can accelerate the convergence. Contrarily, random initialization can trap the network in the local optima.

Optimization algorithms such as GA and PSO can be used in this field. However, the number of DNN parameters (weights and biases) is vast, e.g., 10^{15}, and producing the chromosomes with these dimensions causes a memory error in the processor system and is not efficient in practice. Another method, suggested by Hinton et al., applies the restricted Boltzman machine (RBM) network to tune the auto-encoder's initial parameters [58,59].

According to the ANN-base of an Asy_{AE} and RBM, the Asy_{AE} can be interpreted as two consecutive RBMs illustrated in Figure 4. The input layer is the visible unit, and the encoder layer is the hidden unit for the RBM_1. In the RBM_2, the encoder layer is the visible unit, and the decoder layer is the hidden unit.

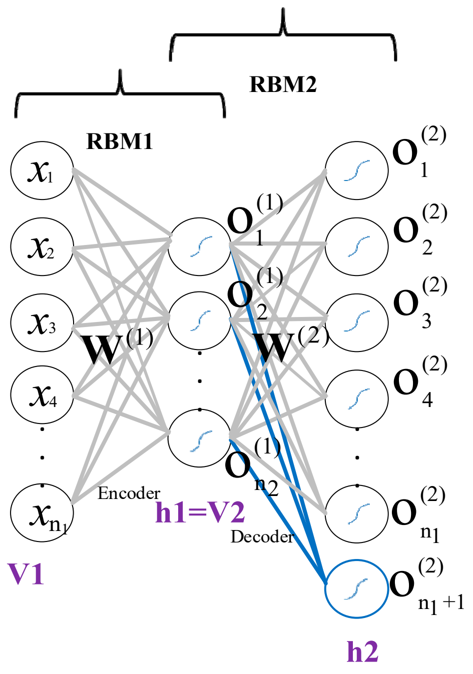

Figure 4. Converting auto-encoder to two RBMs for tuning the initial weights of the encoder and decoder layers.

The conventional RBM is based on binary visible and hidden units, called Bernoulli-Bernoulli RBM (BBRBM). If both visible and hidden units have a Gaussian distribution, the Gaussian-Gaussian RBM (GGRBM) is employed [60]. Since the Asy_{AE} input and parameters are real values, we used the GGRBM equations.

The energy function of the GGRBM is defined as Equation (14), where v presents visible units and h shows hidden units. It should be noted that the Asy_{AE} input and the encoder output are the visible units of RBM_1 and RBM_2, respectively.

$$E(v,h) = -\sum_{i=1}^{g_v}\sum_{j=1}^{g_h} W_{i,j}\frac{v_i h_j}{\sigma_i\sigma_j} - \sum_{i=1}^{g_v}\frac{(v_i - a_i)^2}{2\sigma_i^2} - \sum_{j=1}^{g_h}\frac{(h_j - b_j)^2}{2\sigma_j^2}, \tag{14}$$

where a_i and b_j are visible and hidden units biases, respectively, σ_i and σ_j are their standard deviations. $W_{i,j}$ is the weight between the visible and hidden units. A probability value is assigned to each possible visible and hidden unit by Equation (15),

$$P(v,h) = \frac{1}{Z}\exp(-E(v,h)). \tag{15}$$

Here, Z is the normalization constant calculated by Equation (16).

$$Z = \sum_v\sum_h \exp(-E(v,h)). \tag{16}$$

Equation (17) shows the loss function, which must be maximized,

$$\text{maximize}_{\{W_{i,j},a_i,b_j\}}\frac{1}{c}\sum_{L=1}^{c}\log\left(P\left(v^L,h^L\right)\right), \tag{17}$$

The updating functions are

$$\Delta W_{i,j} = \zeta(< v_i h_j >_{\text{data}} - < v_i h_j >_{\text{model}}), \tag{18}$$

$$\Delta a_i = \zeta(< v_i >_{\text{data}} - < v_i >_{\text{model}}), \tag{19}$$

$$\Delta b_j = \zeta\big(< h_j >_{data} - < h_j >_{model}\big), \tag{20}$$

where $< \bullet >_{data}$ and $< \bullet >_{model}$ are expanded values of sample data and model probabilistic distribution, and ζ is the learning rate.

We described GGRBM briefly, and this is the time to use it. For a traditional autoencoder, first, the initial parameters of the encoder layer are randomly selected and then trained by the GGRBM method. The trained parameters are considered the encoder layer's initial parameters, and its transposition is employed for the decoder layer. However, in the Asy_{AE}, the encoder and decoder parameters are not symmetric and have to be obtained individually. So, the above principle is applied to the decoder layer to obtain the initial parameters.

The number of hidden layers: The value of this hyper-parameter is dependent on the performance of Asy_{AE}s. The classification performance of each Asy_{AE} is examined in MIC for each pair of (N_i, μ_i). For the next Asy_{AE}, the performance has to be better than that of the previous one. If the performance of $Asy_{AE(i+1)}$ is better than that of $Asy_{AE(i)}$, the MIC algorithm is continued.

The performance criterion is different from one problem to another. The Unweighted Average (UA) recall criterion frequently used in personality perception studies is calculated by Equation (21),

$$\text{UA recall} = \frac{1}{2}\Big(\text{recall}_{Low} + \text{recall}_{High}\Big), \tag{21}$$

The recall_{Low} means the recall of detecting the low degree of studied personality, and the recall_{High} indicates the recall of detecting its high degree.

The maximum epoch of network training: Generally, the DNN training process proceeds to reach maximum epoch (updating time) [40]. As discussed in [24], proper data separation does not occur in the maximum epoch. Thus, a J variation is employed as a stopping criterion to finish the training process in the epoch in which the maximum separation is achieved.

J is calculated as follows,

$$J = \frac{\det(\mathbf{S_B})}{\det(\mathbf{S_W})}, \tag{22}$$

where S_W is a within-class scattering matrix, and S_B is a between-class scattering matrix [61]. **det** represents the determinant of a matrix.

$$\mathbf{S_w} = \sum_{i=1}^{c} \sum_{x \in c_i} (\mathbf{X} - \boldsymbol{\mu}_\mathbf{i})(\mathbf{X} - \boldsymbol{\mu}_\mathbf{i})^T, \tag{23}$$

$$\mathbf{S_B} = \sum_{i=1}^{c} n_i (\boldsymbol{\mu}_\mathbf{i} - \boldsymbol{\mu})(\boldsymbol{\mu}_\mathbf{i} - \boldsymbol{\mu})^T. \tag{24}$$

Here, n_i is the instance number of ith class, **X** is the encoder output matrix, and c is the number of classes, μ is the matrix for average all instances, and μ_i is the class average matrix of ith class.

Preventing over-fitting and under-fitting problems: The over-fitting problem happens when a model trains properly on the training dataset but performs poorly on the testing dataset. The under-fitting problem occurs when a model performs poorly on both the training and testing samples.

The number of layers and the neurons in each layer can excessively lead a model into over-fitting or under-fitting. This can be easily changed by changing the structure. More neurons and layers complicate the model, but fewer cannot pursue the data pattern. Therefore, this is one of the problems that has to be dealt with in designing an optimum structure. So, a new loss function is defined to guide the model toward good fitting.

$$\text{Loss} = \frac{UA_{train}}{a} * \frac{UA_{val}}{b}, \tag{25}$$

where a is the training threshold, and b is the validation threshold. We already discussed the UA recall criterion used chiefly in personality perception. We applied the loss function defined in Equation (25) instead of Equation (21). The aim is the maximization of Equation (25). We set a = 0.8 and b = 0.6 because a UA_{train} of more than 80% and UA_{val} of more than 60% are acceptable. The loss value can be in the range of [2.08, 0]. So, the set of (N_i, μ_i) is acceptable to be maximized in Equation (25).

Final algorithm: The pseudo-code of optimizing SA_{AE} hyper-parameters is described in Algorithm 2.

Algorithm 2: Optimizing SAAE hyperparameters

```
Set the initial parameters Old_max = 0, G_max = 2.08 (upper band of Loss), OldEv_Asy = 0 (the
first $Asy_{AE}$ performance) and the other randomly.
Set the input matrix of $Asy_{AE}$.
Set i = 1 (i indicates the number of hidden layer)
Set NewEv_Asy = 1 (the $(i+1)$th $Asy_{AE}$ performance)
While NewEv_Asy> OldEv_Asy
    OldEv_Asy = NewEv_Asy
    While (G_max-Old_max) > 0.1
        Optimize $(N_i, \mu_i)$ with MIC.
        Initialize the $Asy_{AE}$ parameter randomly.
        Tune $Asy_{AE}$ initial parameters with GGRBM.
        Train $Asy_{AE}$ while J increases.
        Evaluate Equation (25).
        If the value of Equation (25) $\geq$ Old_max
          Old_max = the current value of Equation (25),
          NewEv_Asy = the value of Equation (25).
        End if
        Set the encoder layer output of ith $Asy_{AE}$ as the input of $(i+1)$th $Asy_{AE}$.
        i = i + 1.
    End while
End while
```

6. Simulations and Results

In this section, firstly, the results of the MIC method on three benchmarks and comparison with other published methods will be discussed. Then, the MIC will be used to design the structure of five individual DNNs for classifying five personality traits. A final comparison can be found at the end of this section.

6.1. The Results of the MIC on Three Optimization Benchmarks

Three well-known, multimodal, continuous, and non-separable benchmark functions that have a global minimum value of zero, called Rastrigin [52], Ackley [62], and Griewang [62], are used to validate the MIC method.

The multimodal property means having many local optima or peaks in the function, which can test the ability of an algorithm to avoid being stuck in a local minimum. Non-separable refers to the independence of obtained solution variables. If all variables are independent, they can be optimized independently, and the function will be optimized [62]. Therefore, these three functions are complex problems in evaluating the performance of any new optimization algorithm.

The formula, feasible range of variables, and the global optima points of three functions are summarized in Table 2.

Table 2. Description of Three Benchmark Functions.

Name	Formula	Range	Optimal $f(x)$
Rastrigin	$f(x) = 10n + \sum_{i=1}^{n} x_i^2 - 10\cos(2\pi x_i)$	$-5.12 < x_i < 5.12$	0
Ackley	$f(x) = -20\exp\left(-0.02\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + \exp(1)$	$-32 \leq x_i \leq 32$	0
Griewang	$f(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$-600 < x_i < 600$	0

Here, n indicates the dimension of the function, which is $n \geq 2$ for all mentioned functions.

Figure 5 shows the shape of the functions described in Table 2. As can be seen, all three functions have many local optima and are suitable to show the ability of optimization methods to escape from being stuck in local optima.

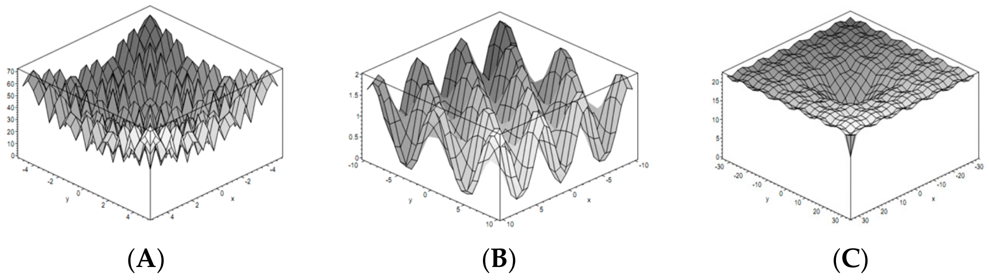

Figure 5. Benchmark functions (**A**) Rastrigin, (**B**) Ackley, and (**C**) Griewang.

In order to show the performance of MIC against the conventional optimization methods, the comparison results of the mentioned four islands and MIC are reported in Table 3.

Given the fact that the problem complexity increases with increasing dimensionality, increasing the number of the variables (dimension) grows the search space, which makes exploring the best solution difficult [62]. To investigate the effect of dimension on searching quality in MIC, we compared our results with 30D and 10D in Table 3.

For a fair comparison, all parameters and initial populations for the basic algorithms and MIC were set to the same values.

The following six criteria were utilized for a more reliable analysis. It should be mentioned that these criteria are common in optimization problems.

- The average of iterations where the stop criterion is reached for examining convergence speed (AvI).
- The average of obtained best optima point (AvP).
- The smallest iteration at which the stop criterion occurs (SI).
- The best-obtained optima point (BOP).
- Calculating the standard deviation (SD) for proving the efficiency and robustness of the algorithm.
- The number of successful runs divided by the total number of runs called success rate (SR).

It was concluded by AvI numerical results that MIC can reach more accurate solutions with a faster convergence speed than traditional algorithms in n = 10. Although for the n = 30, the MM performance diminishes, LM and EM preserve their performance with increasing complexity. It is demonstrated that LM and EM improve solutions steadily for a long time without getting stuck in local minima. It is clear that MIC is more powerful than the four basic algorithms alone when it comes to solving global optimization problems.

Table 3. The Results of MIC Compared with Traditional GA, DE, PSO, and ES in Three Benchmark Functions (10D and 30D).

Benchmark Functions	**Optimization Algorithm**	**AvI**	**AvP**	**SI**	**BOP**	**SD**	**SR (%)**	**AvI**	**AvP**	**SI**	**BOP**	**SD**	**SR (%)**
		n=10						**n**=30					
Rastrigin	MIC by LM	83.4	6.7×10^{-5}	61	7.2×10^{-5}	4.7×10^{-5}	100	307.1	7.8×10^{-4}	254	4.8×10^{-4}	2.4×10^{-4}	100
	MIC by EM	120.5	7.1×10^{-5}	101	8.8×10^{-6}	5.8×10^{-5}	100	321.5	7.4×10^{-4}	187	1.8×10^{-4}	5.6×10^{-4}	100
	MIC by MM	572.5	9.9×10^{-3}	131	4.9×10^{-3}	4.3×10^{-3}	100	2000	0.46	2000	6.1×10^{-4}	1.32	60
	GA	617.1	1.2×10^{-5}	324	6.8×10^{-4}	4.6×10^{-3}	40	1178.4	1.34	926	6.8×10^{-4}	3.20	50
	DE	1000	0.22	1000	0.14	0.47	0	2000	4.72	2000	0.10	2.55	0
	ES	1000	3.48	1000	1.94	2.39	0	2000	37.4	2000	24.7	14.4	0
	PSO	985.7	0.82	857	6.8×10^{-4}	0.71	20	2000	10.9	2000	3.13	5.60	0
Ackley	MIC by LM	467.2	4.4×10^{-15}	355	4.4×10^{-15}	0	100	1039.6	4.4×10^{-15}	956	4.4×10^{-15}	0	100
	MIC by EM	788.4	4.4×10^{-15}	462	4.4×10^{-15}	0	100	1154.8	3.1×10^{-14}	937	4.4×10^{-15}	1.9×10^{-15}	100
	MIC by MM	725.2	1.4×10^{-9}	324	3.5×10^{-10}	9.2×10^{-10}	100	2000	2.48	2000	2.24	0.20	0
	GA	957.5	7.3×10^{-2}	565	7.2×10^{-3}	1.58	70	1895.1	2.86	951	0.01	1.53	10
	DE	1000	1.69	1000	1.24	4.47	0	2000	5.35	2000	4.34	0.67	0
	ES	1000	4.96	1000	3.20	3.09	0	2000	5.43	2000	5.23	1.9×10^{-1}	0
	PSO	557.5	8.6×10^{-4}	344	6.2×10^{-4}	4.1×10^{-4}	100	839.5	4.5×10^{-3}	162	8.8×10^{-4}	7.9×10^{-3}	100
Griewang	MIC by LM	154.7	6.2×10^{-14}	38	1.2×10^{-14}	2.8×10^{-14}	100	106	8.4×10^{-14}	92	8.7×10^{-14}	4.4×10^{-14}	100
	MIC by EM	171.2	8.4×10^{-14}	43	6.4×10^{-14}	1.5×10^{-14}	100	489	1.1×10^{-13}	94	9.1×10^{-14}	2.5×10^{-14}	100
	MIC by MM	775.4	3.3×10^{-13}	146	9.6×10^{-14}	3.2×10^{-13}	100	2000	0.27	2000	9.1×10^{-13}	0.16	20
	GA	909.8	0.09	84	0.8×10^{-3}	0.18	10	2000	0.21	2000	0.09	1.1×10^{-1}	0
	DE	337.4	9.1×10^{-3}	44	7.3×10^{-3}	1.1×10^{-3}	100	993	0.01	588	7.9×10^{-3}	8.8×10^{-1}	70
	ES	1000	0.36	1000	1.2×10^{-1}	0.20	0	2000	0.81	2000	0.76	0.19	0
	PSO	555.2	0.02	258	6.6×10^{-2}	2.8×10^{-2}	30	2000	0.77	2000	0.37	3.1×10^{-2}	0

Table 3 shows the simulation outcomes of MIC and four basic optimization algorithms.

According to the AvP values in n = 10 and n = 30, traditional algorithms are often unsuccessful in finding favorable solutions in comparison to MIC, especially EM. Additionally, it can be concluded from AvP that the MIC speeds up the convergence to the global optima. The AvP values in n = 30 in comparison to n = 10 decreased about 0.1 in Rastrigin and remained constant for the other two functions in LM and EM. The change in the AvP values in MM is meaningful, which indicates getting stuck in the local optimum with the increase in the complexity of the problem, like the traditional methods.

Our SI outcomes show that the MIC method, especially EM and LM, reaches the stop criterion in a few iterations. It means the MIC method speeds up convergence. Moreover, the SI criterion shows that although the MM method performs better than the basic optimization methods in simpler functions (n = 10), its performance drops in complex functions (n = 30). LM and EM not only show their effectiveness in simple functions, but also perform well in complex problems compared to other methods.

The evaluation results of criterion BOP show that the LM and EM methods achieve the global optimal value more accurately than the basic methods in n = 10 and n = 30. However, MM implementation results decrease with increasing complexity.

It can be seen that the SD values of MIC, except for MM, are very small in comparison to those of the four basic algorithms in n = 10 and n = 30, which means the repeatability and robustness of the new algorithm are due to pruning search space.

The SR results prove that the MIC is very promising in bringing higher reliability than traditional algorithms because the number of times that LM, EM, and MM reached the desired value of the function was 100% in n = 10. As can be seen, as the complexity of the function increases (n = 30), the LM and EM methods are still successful in reaching the desired value.

From Table 3, it is concluded that despite increasing dimensions, the implementation outcomes of all algorithms decrease, except EM and LM.

Our study indicates that the quality of the solutions found using our proposed method for widespread global optima functions is higher than that of the solutions provided by traditional algorithms. This is due to a more appropriate tradeoff between exploring new individuals and exploiting highly fit individuals found at the parallelism level. By means of three widespread test functions, it is demonstrated that the new method has great potential for substantial improvement in search performance.

Due to the wide usage of these benchmarks, a comparison with other published works is presented in Table 4. It can be observed that LM and EM achieved the best solution in Ackley and Griewang functions (30D).

Table 4. Comparison with Other Published Methods in 30D. N/A means not available.

Methods	Benchmarks								
	Rastrigin			Ackley			Griewang		
	AvP	SD	SR%	AvP	SD	SR%	AvP	SD	SR%
Xin Zhao et al., 2022 [55]	2.1×10^{-13}	4.1×10^{-14}	100	8.2×10^{-15}	1.3×10^{-15}	100	3.78×10^{-13}	1.7×10^{-13}	100
Chentoufi et al., 2021 [49]	0.99	1.31	100	1.0×10^{-15}	6.4×10^{-16}	43	8.3×10^{-4}	5.4×10^{-4}	67
MIC_LM	7.8×10^{-4}	2.4×10^{-4}	100	4.4×10^{-15}	0	100	8.4×10^{-14}	4.4×10^{-14}	100
MIC_EM	7.4×10^{-4}	5.6×10^{-4}	100	3.1×10^{-14}	1.9×10^{-15}	100	1.1×10^{-13}	2.5×10^{-14}	100
MIC_MM	0.46	1.32	60	2.48	0.20	0	0.27	0.16	20

6.2. The Results of Personality Perception with The MIC Method

After the successful outcomes with the MIC method to find the global optima of three complex benchmark functions, we applied our novel method to find the near-optimal values of hyper-parameters for classifying five personality traits. We used “near-optimal” instead of “optimal” structure because tuning of MIC hyper-parameters such as mutation and crossover rating is chosen randomly.

Taking into account that different personality traits have different effects on speech characteristics [24,42], using the same DNN structure for all traits to extract features is not recommended. Assuming the five personality traits were independent, five separate neural networks were designed and trained to classify the five personality traits.

Hence, the network's depth was determined by classifying the output of each Asy_{AE} encoder layer by the SVM with radial basis function kernel. The Asy_{AE} with higher classification results is considered as the output layer of the S_{AE}.

Table 5 shows the comparison results of our proposed method with other works in the SPC dataset in terms of UA recall and accuracy. In our previous work, the structure of SA_{AE} was chosen by trial-and-error, which was time-consuming, and two traits (extraversion and openness) achieved lower accuracy than reported by other research [24]. N/A means not available.

Table 5. Comparison Results of Our Proposed Method with Other Works in the SPC Dataset in Terms of UA Recall % (Accuracy %).

Methods	Traits				
	Neu.	**Ext.**	**Ope.**	**Agr.**	**Con.**
Mohammadi et al., 2010 [43]	N/A (63)	N/A (76.3)	N/A (57.9)	N/A (63)	N/A (72)
Mohammadi et al., 2012 [63]	N/A (65.9)	N/A (73.5)	N/A (60.1)	N/A (63.1)	N/A (71.3)
Chastagnol et al., 2012 [64]	58 (N/A)	75.5 (N/A)	73.4 (N/A)	65 (N/A)	62.2 (N/A)
Mohammadi et al., 2015 [65]	N/A (66.1)	N/A (71.4)	N/A (58.6)	N/A (58.8)	N/A (72.5)
Solera-Urena et al., 2017 [66]	65.1 (64.7)	75 (75.1)	59.1 (58.2)	60.3 (60.2)	75.7 (75.6)
Carbonneau et al., 2017 [67]	70.8 (N/A)	75.2 (N/A)	56.3 (N/A)	64.9 (N/A)	63.8 (N/A)
Zhen-Tao Liu et al., 2020 [68]	N/A (69.2)	N/A (76.3)	N/A (74.7)	N/A (65.3)	N/A (73.3)
Our privuse work 2021 [24]	77.1 (76.9)	76.6 (72.9)	81.2 (70.4)	80.7 (68.7)	78.5 (69.5)
Proposed method	89.8 (80.5)	82.2 (83.4)	87.1 (84.7)	85.8 (76.2)	81.8 (72.6)

In the present study, not only were the accuracy of extraversion and openness improved, but UA recalls were also increased more than before. This evidences that the performance and robustness of trained models are highly dependent on their hyper-parameter settings.

7. Conclusions

Since HPT is the most challenging aspect of ANN studies, it is mostly obtained by trial-and-error, affecting its performance. This article proposed a new approach based on cultural evolution and parallel computing to achieve a near-optimal structure of SA_{AE} in a reasonable time for automatic personality perception. We used the concept of parallelism and information on different regions of the search space to improve the search spaces in MIC and exchanged them between islands to provide greater population diversity. The proposed approach was implemented on three complex benchmarks, and six criteria evaluated our method's performance in comparison with four basic optimization methods. The results showed that our approach outperforms other traditional optimization and newly published algorithms in four aspects: (1) convergence speed, (2) precision, (3) escaping from entrapment in local optima, and (4) repeatability. As an indication of our method's performance, we increased the problem complexity by increasing the number of variables up to 30. The outcomes demonstrated the reliability of the MIC method, especially LM and EM. Subsequently, five hyper-parameters of SA_{AE} were optimized. Since the tuning of hyper-parameters affects over-fitting and under-fitting, we introduced a new cost function to control them during the optimization process.

In comparison with the results of our previous published work, the outcomes of applying MIC to SA_{AE} indicated 3.3% (3.1%) for consciousness, 5.1% (7.5%) for agreeableness, 5.9% (14.3%) for openness, 5.6% (10.1%) for extraversion, and 12.7% (3.6%) for neuroticism.

Author Contributions: Writing—review & editing, E.J.Z., M.T., A.K., C.H., M.V., N.N. and T.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Bischl, B.; Binder, M.; Lang, M.; Pielok, T.; Richter, J.; Coors, S.; Thomas, J.; Ullmann, T.; Becker, M.; Boulesteix, A.-L. Hyperparameter optimization: Foundations, algorithms, best practices and open challenges. *arXiv* **2021**, arXiv:107.05847.
2. Szepannek, G.; Lübke, K. Explaining Artificial Intelligence with Care. In *KI-Künstliche Intell.*; 2022; Volume 16, pp. 1–10. [CrossRef]
3. Khodadadian, A.; Parvizi, M.; Teshnehlab, M.; Heitzinger, C. Rational Design of Field-Effect Sensors Using Partial Differential Equations, Bayesian Inversion, and Artificial Neural Networks. *Sensors* **2022**, *22*, 4785. [CrossRef]
4. Guo, C.; Li, L.; Hu, Y.; Yan, J. A Deep Learning Based Fault Diagnosis Method With hyperparameter Optimization by Using Parallel Computing. *IEEE Access* **2020**, *8*, 131248–131256. [CrossRef]
5. Feurer, M.; Hutter, F. Hyperparameter optimization. In *Automated Machine Learning*; Springer: Cham, Switzerland, 2019; pp. 3–33.
6. Yang, L.; Shami, A. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing* **2020**, *415*, 295–316. [CrossRef]
7. Wu, D.; Wu, C. Research on the Time-Dependent Split Delivery Green Vehicle Routing Problem for Fresh Agricultural Products with Multiple Time Windows. *Agriculture* **2022**, *12*, 793. [CrossRef]
8. Peng, Y.; Gong, D.; Deng, C.; Li, H.; Cai, H.; Zhang, H. An automatic hyperparameter optimization DNN model for precipitation prediction. *Appl. Intell.* **2021**, *52*, 2703–2719. [CrossRef]
9. Yi, H.; Bui, K.-H.N. An automated hyperparameter search-based deep learning model for highway traffic prediction. *IEEE Trans. Intell. Transp. Syst.* **2020**, *22*, 5486–5495. [CrossRef]
10. Kinnewig, S.; Kolditz, L.; Roth, J.; Wick, T. *Numerical Methods for Algorithmic Systems and Neural Networks*; Institut für Angewandte Mathematik, Leibniz Universität Hannover: Hannover, Germany, 2022. [CrossRef]
11. Han, J.-H.; Choi, D.-J.; Park, S.-U.; Hong, S.-K. Hyperparameter optimization using a genetic algorithm considering verification time in a convolutional neural network. *J. Electr. Eng. Technol.* **2020**, *15*, 721–726. [CrossRef]
12. Yao, R.; Guo, C.; Deng, W.; Zhao, H. A novel mathematical morphology spectrum entropy based on scale-adaptive techniques. *ISA Trans.* **2022**, *126*, 691–702. [CrossRef] [PubMed]
13. Raji, I.D.; Bello-Salau, H.; Umoh, I.J.; Onumanyi, A.J.; Adegboye, M.A.; Salawudeen, A.T. Simple deterministic selection-based genetic algorithm for hyperparameter tuning of machine learning models. *Appl. Sci.* **2022**, *12*, 1186. [CrossRef]
14. Harichandana, B.; Kumar, S. LEAPMood: Light and Efficient Architecture to Predict Mood with Genetic Algorithm driven Hyperparameter Tuning. In Proceedings of the 2022 IEEE 16th International Conference on Semantic Computing (ICSC), Laguna Hills, CA, USA, 26–28 January 2022; pp. 1–8.
15. Guido, R.; Groccia, M.C.; Conforti, D. Hyper-Parameter Optimization in Support Vector Machine on Unbalanced Datasets Using Genetic Algorithms. In *Optimization in Artificial Intelligence and Data Sciences*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 37–47.
16. Thavasimani, K.; Srinath, N.K. Hyperparameter optimization using custom genetic algorithm for classification of benign and malicious traffic on internet of things-23 dataset. *Int. J. Electr. Comput. Eng.* **2022**, *12*, 4031–4041. [CrossRef]
17. Awad, M. Optimizing the Topology and Learning Parameters of Hierarchical RBF Neural Networks Using Genetic Algorithms. *Int. J. Appl. Eng. Res.* **2018**, *13*, 8278–8285.
18. Faris, H.; Mirjalili, S.; Aljarah, I. Automatic selection of hidden neurons and weights in neural networks using grey wolf optimizer based on a hybrid encoding scheme. *Int. J. Mach. Learn. Cybern.* **2019**, *10*, 2901–2920. [CrossRef]
19. Li, A.; Spyra, O.; Perel, S.; Dalibard, V.; Jaderberg, M.; Gu, C.; Budden, D.; Harley, T.; Gupta, P. A generalized framework for population based training. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 1791–1799.
20. Luo, G. A review of automatic selection methods for machine learning algorithms and hyper-parameter values. *Netw. Model. Anal. Health Inform. Bioinform.* **2016**, *5*, 18. [CrossRef]
21. An, Z.; Wang, X.; Li, B.; Xiang, Z.; Zhang, B. Robust visual tracking for UAVs with dynamic feature weight selection. *Appl. Intell.* **2022**. [CrossRef]
22. Bhandare, A.; Kaur, D. Designing convolutional neural network architecture using genetic algorithms. *Int. J. Adv. Netw. Monit. Control* **2021**, *6*, 26–35. [CrossRef]
23. Tan, H.H.; Lim, K.H. Vanishing gradient mitigation with deep learning neural network optimization. In Proceedings of the 2019 7th International Conference on Smart Computing & Communications (ICSCC), Sarawak, Malaysia, 28–30 June 2019; pp. 1–4.
24. Zaferani, E.J.; Teshnehlab, M.; Vali, M. Automatic Personality Traits Perception Using Asymmetric Auto-Encoder. *IEEE Access* **2021**, *9*, 68595–68608. [CrossRef]
25. Cho, H.; Kim, Y.; Lee, E.; Choi, D.; Lee, Y.; Rhee, W. Basic enhancement strategies when using bayesian optimization for hyperparameter tuning of deep neural networks. *IEEE Access* **2020**, *8*, 52588–52608. [CrossRef]

26. Sun, Y.; Xue, B.; Zhang, M.; Yen, G.G.; Lv, J. Automatically designing CNN architectures using the genetic algorithm for image classification. *IEEE Trans. Cybern.* **2020**, *50*, 3840–3854. [CrossRef]
27. Cabada, R.Z.; Rangel, H.R.; Estrada, M.L.B.; Lopez, H.M.C. Hyperparameter optimization in CNN for learning-centered emotion recognition for intelligent tutoring systems. *Soft Comput.* **2020**, *24*, 7593–7602. [CrossRef]
28. Deng, W.; Liu, H.; Xu, J.; Zhao, H.; Song, Y. An improved quantum-inspired differential evolution algorithm for deep belief network. *IEEE Trans. Instrum. Meas.* **2020**, *69*, 7319–7327. [CrossRef]
29. Guo, Y.; Li, J.-Y.; Zhan, Z.-H. Efficient hyperparameter optimization for convolution neural networks in deep learning: A distributed particle swarm optimization approach. *Cybern. Syst.* **2020**, *52*, 36–57. [CrossRef]
30. Ozcan, T.; Basturk, A. Static facial expression recognition using convolutional neural networks based on transfer learning and hyperparameter optimization. *Multimed. Tools Appl.* **2020**, *79*, 26587–26604. [CrossRef]
31. Gülcü, A.; Kuş, Z. Hyper-parameter selection in convolutional neural networks using microcanonical optimization algorithm. *IEEE Access* **2020**, *8*, 52528–52540. [CrossRef]
32. Kong, D.; Wang, S.; Ping, P. State-of-health estimation and remaining useful life for lithium-ion battery based on deep learning with Bayesian hyperparameter optimization. *Int. J. Energy Res.* **2022**, *46*, 6081–6098. [CrossRef]
33. Chowdhury, A.A.; Hossen, M.A.; Azam, M.A.; Rahman, M.H. Deepqgho: Quantized greedy hyperparameter optimization in deep neural networks for on-the-fly learning. *IEEE Access* **2022**, *10*, 6407–6416. [CrossRef]
34. Chen, H.; Miao, F.; Chen, Y.; Xiong, Y.; Chen, T. A hyperspectral image classification method using multifeature vectors and optimized KELM. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2021**, *14*, 2781–2795. [CrossRef]
35. Phan, L.V.; Rauthmann, J.F. Personality computing: New frontiers in personality assessment. *Soc. Personal. Psychol. Compass* **2021**, *15*, e12624. [CrossRef]
36. Koutsombogera, M.; Sarthy, P.; Vogel, C. Acoustic Features in Dialogue Dominate Accurate Personality Trait Classification. In Proceedings of the 2020 IEEE International Conference on Human-Machine Systems (ICHMS), Rome, Italy, 7–9 September 2020; pp. 1–3.
37. Aslan, S.; Güdükbay, U.; Dibeklioğlu, H. Multimodal assessment of apparent personality using feature attention and error consistency constraint. *Image Vis. Comput.* **2021**, *110*, 104163. [CrossRef]
38. Xu, J.; Tian, W.; Lv, G.; Liu, S.; Fan, Y. Prediction of the Big Five Personality Traits Using Static Facial Images of College Students With Different Academic Backgrounds. *IEEE Access* **2021**, *9*, 76822–76832. [CrossRef]
39. Kampman, O.; Siddique, F.B.; Yang, Y.; Fung, P. Adapting a virtual agent to user personality. In *Advanced Social Interaction with Agents*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 111–118.
40. Suen, H.-Y.; Hung, K.-E.; Lin, C.-L. Intelligent video interview agent used to predict communication skill and perceived personality traits. *Hum.-Cent. Comput. Inf. Sci.* **2020**, *10*, 1–12. [CrossRef]
41. Liam Kinney, A.W.; Zhao, J. Detecting Personality Traits in Conversational Speech. Stanford University: Stanford, CA, USA, 2017; Available online: https://web.stanford.edu/class/cs224s/project/reports_2017/Liam_Kinney.pdf (accessed on 15 June 2022).
42. Jalaeian Zaferani, E.; Teshnehlab, M.; Vali, M. Automatic personality recognition and perception using deep learning and supervised evaluation method. *J. Appl. Res. Ind. Eng.* **2022**, *9*, 197–211.
43. Mohammadi, G.; Vinciarelli, A.; Mortillaro, M. The voice of personality: Mapping nonverbal vocal behavior into trait attributions. In Proceedings of the 2nd international workshop on Social signal processing, Firenze, Italy, 29 October 2010; pp. 17–20.
44. Rosenberg, A. Speech, Prosody, and Machines: Nine Challenges for Prosody Research. In Proceedings of the 9th International Conference on Speech Prosody 2018, Poznań, Poland, 13–16 June 2018; pp. 784–793.
45. Junior, J.C.S.J.; Güçlütürk, Y.; Pérez, M.; Güçlü, U.; Andujar, C.; Baró, X.; Escalante, H.; Guyon, I.; van Gerven, M.; van Lier, R. First impressions: A survey on computer vision-based apparent personality trait analysis. *arXiv* **2019**, arXiv:1804.08046v1.
46. Schuller, B.; Weninger, F.; Zhang, Y.; Ringeval, F.; Batliner, A.; Steidl, S.; Eyben, F.; Marchi, E.; Vinciarelli, A.; Scherer, K. Affective and behavioural computing: Lessons learnt from the first computational paralinguistics challenge. *Comput. Speech Lang.* **2019**, *53*, 156–180. [CrossRef]
47. Hutter, F.; Kotthoff, L.; Vanschoren, J. *Automated Machine Learning: Methods, Systems, Challenges*; Springer: Berlin/Heidelberg, Germany, 2019.
48. Harada, T.; Alba, E. Parallel genetic algorithms: A useful survey. *ACM Comput. Surv.* **2020**, *53*, 1–39. [CrossRef]
49. Chentoufi, M.A.; Ellaia, R. A novel multi-population passing vehicle search algorithm based co-evolutionary cultural algorithm. *Comput. Sci.* **2021**, *16*, 357–377.
50. Liu, Z.-H.; Tian, S.-L.; Zeng, Q.-L.; Gao, K.-D.; Cui, X.-L.; Wang, C.-L. Optimization design of curved outrigger structure based on buckling analysis and multi-island genetic algorithm. *Sci. Prog.* **2021**, *104*, 368504211023277. [CrossRef]
51. Shah, P.; Kobti, Z. Multimodal fake news detection using a Cultural Algorithm with situational and normative knowledge. In Proceedings of the 2020 IEEE Congress on Evolutionary Computation (CEC), Glasgow, UK, 19–24 July 2020; pp. 1–7.
52. Al-Betar, M.A.; Awadallah, M.A.; Doush, I.A.; Hammouri, A.I.; Mafarja, M.; Alyasseri, Z.A.A. Island flower pollination algorithm for global optimization. *J. Supercomput.* **2019**, *75*, 5280–5323. [CrossRef]
53. Sun, Y.; Zhang, L.; Gu, X. A hybrid co-evolutionary cultural algorithm based on particle swarm optimization for solving global optimization problems. *Neurocomputing* **2012**, *98*, 76–89. [CrossRef]
54. da Silva, D.J.A.; Teixeira, O.N.; de Oliveira, R.C.L. *Performance Study of Cultural Algorithms Based on Genetic Algorithm with Single and Multi Population for the MKP. Bio-Inspired Computational Algorithms and Their Applitions*; IntechOpen: London, UK, 2012; pp. 385–404.

55. Zhao, X.; Tang, Z.; Cao, F.; Zhu, C.; Periaux, J. An Efficient Hybrid Evolutionary Optimization Method Coupling Cultural Algorithm with Genetic Algorithms and Its Application to Aerodynamic Shape Design. *Appl. Sci.* **2022**, *12*, 3482. [CrossRef]
56. Muhamediyeva, D. Fuzzy cultural algorithm for solving optimization problems. *J. Phys. Conf. Ser.* **2020**, *1441*, 012152. [CrossRef]
57. Xu, W.; Wang, R.; Zhang, L.; Gu, X. A multi-population cultural algorithm with adaptive diversity preservation and its application in ammonia synthesis process. *Neural Comput. Appl.* **2012**, *21*, 1129–1140. [CrossRef]
58. Hinton, G.E.; Salakhutdinov, R.R. Reducing the dimensionality of data with neural networks. *Science* **2006**, *313*, 504–507. [CrossRef] [PubMed]
59. Cho, K.H.; Raiko, T.; Ilin, A. Gaussian-bernoulli deep boltzmann machine. In Proceedings of the 2013 International Joint Conference on Neural Networks (IJCNN), Dallas, TX, USA, 4–9 August 2013; pp. 1–7.
60. Ogawa, S.; Mori, H. A gaussian-gaussian-restricted-boltzmann-machine-based deep neural network technique for photovoltaic system generation forecasting. *IFAC-Pap.* **2019**, *52*, 87–92. [CrossRef]
61. Tharwat, A.; Gaber, T.; Ibrahim, A.; Hassanien, A.E. Linear discriminant analysis: A detailed tutorial. *AI Commun.* **2017**, *30*, 169–190. [CrossRef]
62. Jamil, M.; Yang, X.-S. A literature survey of benchmark functions for global optimisation problems. *Int. J. Math. Model. Numer. Optim.* **2013**, *4*, 150–194. [CrossRef]
63. Mohammadi, G.; Vinciarelli, A. Automatic personality perception: Prediction of trait attribution Based Prosodic Features. *IEEE Trans. Affect. Comput.* **2012**, *3*, 273–284. [CrossRef]
64. Chastagnol, C.; Devillers, L. Personality traits detection using a parallelized modified SFFS algorithm. *Computing* **2012**, *15*, 16.
65. Mohammadi, G.; Vinciarelli, A. Automatic personality perception: Prediction of trait attribution based on prosodic features extended abstract. In Proceedings of the 2015 International Conference on Affective Computing and Intelligent Interaction (ACII), Xi'an, China, 21–24 September 2015; pp. 484–490.
66. Solera-Ureña, R.; Moniz, H.; Batista, F.; Cabarrão, R.; Pompili, A.; Astudillo, R.; Campos, J.; Paiva, A.; Trancoso, I. A semi-supervised learning approach for acoustic-prosodic personality perception in under-resourced domains. In Proceedings of the 18th Annual Conference of the International Speech Communication Association, INTERSPEECH 2017, Stockholm, Sweden, 20–24 August 2017; pp. 929–933.
67. Carbonneau, M.-A.; Granger, E.; Attabi, Y.; Gagnon, G. Feature learning from spectrograms for assessment of personality traits. *IEEE Trans. Affect. Comput.* **2017**, *11*, 25–31. [CrossRef]
68. Liu, Z.-T.; Rehman, A.; Wu, M.; Cao, W.; Hao, M. Speech personality recognition based on annotation classification using log-likelihood distance and extraction of essential audio features. *IEEE Trans. Multimed.* **2020**, *23*, 3414–3426. [CrossRef]