micromachines

Article

Cyborg Moth Flight Control Based on Fuzzy Deep Learning

Xiao Yang [1], Xun-Lin Jiang [2], Zheng-Lian Su [3] and Ben Wang [1,*]

[1] School of Information Science and Technology, Hangzhou Normal University, Hangzhou 311121, China; 20170056@hznu.edu.cn
[2] Department of Engineering Technology and Application, Army Infantry College, Nanchang 330100, China; xulinjiang@163.com
[3] College of Field Engineering, Army Engineering University, Nanjing 210007, China; suzhenglian@compintell.cn
* Correspondence: drwang@compintell.cn

Abstract: Cyborg insect control methods can be divided into invasive methods and noninvasive methods. Compared to invasive methods, noninvasive methods are much easier to implement, but they are sensitive to complex and highly uncertain environments, for which classical control methods often have low control accuracy. In this paper, we present a noninvasive approach for cyborg moths stimulated by noninvasive ultraviolet (UV) rays. We propose a fuzzy deep learning method for cyborg moth flight control, which consists of a *Behavior Learner* and a *Control Learner*. The *Behavior Learner* is further divided into three hierarchies for learning the species' common behaviors, group-specific behaviors, and individual-specific behaviors step by step to produce the expected flight parameters. The *Control Learner* learns how to set UV ray stimulation to make a moth exhibit the expected flight behaviors. Both the *Control Learner* and *Behavior Learner* (including its sub-learners) are constructed using a Pythagorean fuzzy denoising autoencoder model. Experimental results demonstrate that the proposed approach achieves significant performance advantages over the state-of-the-art approaches and obtains a high control success rate of over 83% for flight parameter control.

Keywords: intelligent cyborgs; cyborg moth; deep learning; fuzzy systems; flight control

Citation: Yang, X.; Jiang, X.-L.; Su, Z.-L.; Wang, B. Cyborg Moth Flight Control Based on Fuzzy Deep Learning. *Micromachines* **2022**, *13*, 611. https://doi.org/10.3390/mi13040611

Academic Editor: Zhangguo Yu

Received: 14 March 2022
Accepted: 11 April 2022
Published: 13 April 2022

1. Introduction

Cyborg insects, i.e., insect–computer hybrid robots that combine living insects and miniature machines, have numerous potential applications including search and rescue, target detection and surveillance, network fault location and maintenance, animal population control in agriculture, and endangered species protection, to name just a few [1–4]. This new field introduces the possibility, beyond traditional bio-inspiration, of the merging of natural and artificial worlds in synergistic systems [5]. The study of cyborg insects, in its modern form, started with the pioneering work of Holzer and Shimoyama [6], who built a line-tracing electronic backpack that obtains input from two photosensors and uses an on-board algorithm to control cockroaches (*Periplaneta Americana*) via electric stimulation to walk along a black line. The locomotion control of cockroaches has also been realized by stimulation of ganglia [7], antennae, and cerci [8,9]. Recently, more studies have been devoted to flying cyborg insects such as beetles [10–13], moths [14–16], locusts [17,18], and bees [19,20], which are mainly achieved by electrical stimulation of nervous systems, especially the efferent nerves related to the flying muscles [21].

All the above studies develop cyborg insects based on invasive stimulation methods, which require delicate microsurgery skills to accurately place electronics in insect tissues, where the resultant injury has the potential to affect flight performance [14]. Another important consideration is the public acceptance of cyborg insects, in particular, perceptions of whether insects experience pain during implantation surgeries and stimulations [22]. Therefore, noninvasive stimulation represents the best path for developing highly efficient systems as well as encouraging public acceptance from an ethical point of view [5].

However, current studies on noninvasive cyborg insects are still few. Zheng et al. [23] proposed a noninvasive method that uses an online light-emitting diode (LED) display system to present visual stimulus within ultra-low latency to induce bumblebee flight behaviors, which employs reinforcement learning coupled with sequential *K*-means clustering to generate an optimal control sequence. In [24], Zheng et al. modeled tethered bumblebee fight control as a finite and deterministic Markov decision process, and they employed Sarsa with a transformed reward function to learn the optimal control policy. The results demonstrated that the noninvasive methods can also ensure satisfactory control performance while avoiding implantation to insect tissues and reducing physical injury. Nevertheless, compared to implanted methods, noninvasive methods not only have higher sensitivity to noise and lower control accuracy but also require more extensive experimental data for training. Consequently, there are few reports of cyborg insects with noninvasive stimulation outside of the laboratory with highly controlled conditions.

To address the above difficulties, this paper proposes a fuzzy deep learning approach to the flight control of moths *Fusarium camelliae* stimulated by ultraviolet (UV) rays. The control model consists of a *Behavior Learner* and a *Control Learner*. The *Behavior Learner* is further divided into three hierarchies: (1) the lower hierarchy of layers for learning the species' common behaviors in response to external stimuli; (2) the middle hierarchy of layers for learning the specific behaviors of different groups of moths, where the grouping is performed by a fuzzy clustering layer; and (3) the upper hierarchy of layers for learning the specific behaviors of each individual moth. The *Control Learner* learns how to set UV ray stimulation to make a moth exhibit the expected flight behaviors. The *Control Learner* and the sub-learners of the *Behavior Learner* are all constructed using a fuzzy deep learning model. Experimental results demonstrate that the proposed approach achieves significant performance advantages over other popular methods. The main contributions of this paper are as follows:

- We present a noninvasive cyborg moth design approach based on fuzzy deep learning for flight control;
- We propose a novel hierarchical fuzzy deep learning model that effectively learns the species common behaviors, group-specific behaviors, and individual-specific behaviors to achieve a high control success rate.
- We propose a new fuzzy clustering method based on Pythagorean-type fuzzy sets for moth grouping.
- The proposed approach can be easily extended for behavior learning of other cyborg animals and, therefore, contributes to the development of biobots.

In the rest of this paper, Section 2 presents the fuzzy deep learning model, Sections 3 and 4 describe the *Behavior Learner* and *Control Learner* in detail, respectively, Section 5 presents the experimental results, and Section 6 concludes with a discussion.

2. Overview of the Model for Cyborg Flight Control

2.1. Model Architecture

In our study, a cyborg moth is equipped with a wireless backpack chip, which has four micro UV LED lamps, including a UVA radiation lamp and a UVB radiation lamp in each of the left and right sides, as illustrated in Figure 1. The cyborg is designed to use UV ray stimulation to control the flight of the moth in a three-dimensional (3D) space. The control model outputs seven flight parameters, including the horizontal deflection angle θ_h, the horizontal angular velocity ω_h, the vertical deflection angle θ_v, the vertical angular velocity ω_v, and the accelerations in the x-, y-, and z-axes, denoted by a_x, a_y, and a_z, respectively.

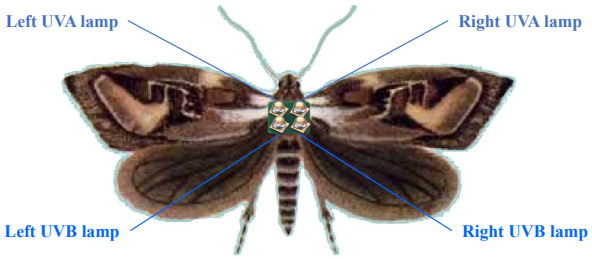

Figure 1. Illustration of a cyborg moth equipped with a backpack chip.

The proposed control model consists of a *Behavior Learner* and a *Control Learner*. As shown in Figure 2, the *Behavior Learner* is used to learn, under a given environment, how UV ray stimulation will affect the flight behaviors of moths. The *Control Learner* is used to learn, in order to make a moth to exhibit specific flight behaviors, which UV ray stimulation should be performed. The output stimuli of the *Control Learner* are input to the *Behavior Learner* to generate expected flight parameters or input to real cyborgs to generate actual flight parameters, whose differences from the required flight parameters are used as feedback to tune the *Control Learner*.

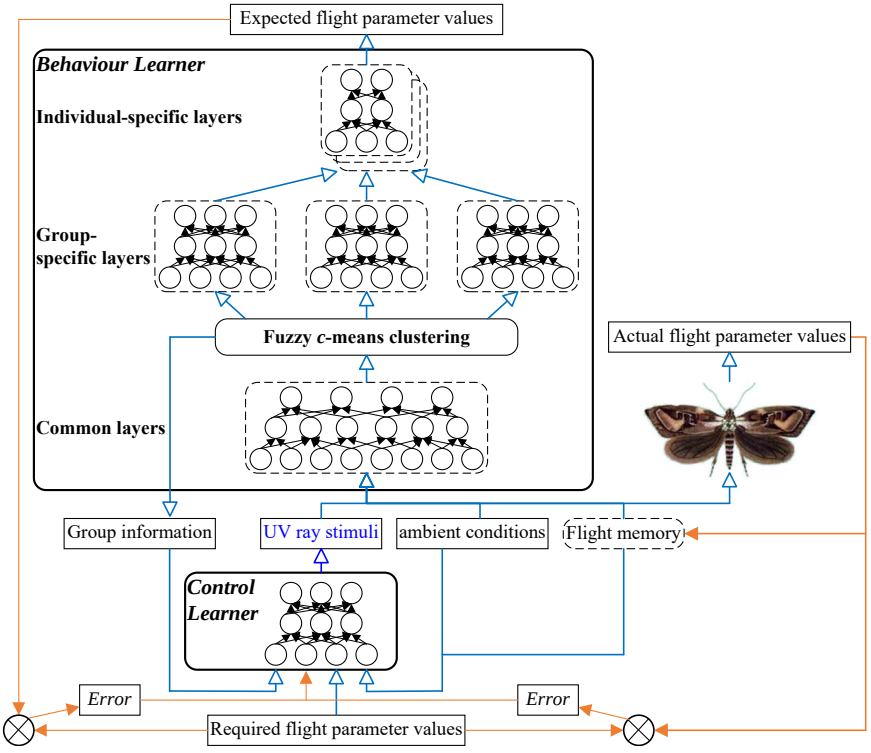

Figure 2. The framework of the fuzzy deep learning model for cyborg control. The *Control Learner* and the three sub-learners of the *Behavior Learner* are all constructed using the proposed PFDDAE model.

The inputs to the *Behavior Learner* have the following three parts:

- The UV ray stimulation, which is described by 32 variables, i.e., the light intensities, exposure durations, pattern moving velocities (in the x-, y-, and z-axes), and pattern moving distances (in the x-, y-, and z-axes) of the four lamps.
- The ambient conditions, which are described by 34 variables, i.e., temperature, humidity, atmospheric pressure, oxygen concentration, carbon dioxide concentration, horizontal and vertical wind speeds, and the light duration, intensity, and illuminance of nine different wavelengths/colors (UVA, UVB, UVC, violet, indigo, blue, yellow, green, orange) to which the insect is sensitive (note that our study assumes that the

ambient wind speed does not exceed 1.5 m/s; if the wind is too strong, then it is impossible to control the flight path of a moth).

- The previous flight parameter values of the moth, including seven values at the previous time step, and seven values accumulated over the previous three time steps using a memory neural network [25,26]. In this study, we set the interval between two time steps to 200 ms.

Therefore, the *Behavior Learner* takes 80 input variables to predict seven output flight parameter values. To address the complexity, instability, and uncertainty of moth flight behaviors, we divide the learner into three hierarchies from bottom to top: (1) the lower hierarchy of common layers, (2) the middle hierarchy of group-specific layers, and (3) the upper hierarchy of individual-specific layers.

The *Control Learner* takes inputs including (1) the user-specified (required) flight parameter values, (2) the ambient conditions, (3) the previous flight parameter values of the moth, and (4) the grouping information about the moth to output the UV ray stimulus values that are expected to make the moth perform the required flight behaviors (specified by the required flight parameter values).

The *Behavior Learner* aims to minimize the difference between the model output values and the actual flight parameter values of the cyborg moths. The *Control Learner* aims to minimize the difference between the required flight parameter values and the actual flight parameter values of the moths (or the output parameter values of the *Behavior Learner*) under the output stimuli of the *Control Learner*. We propose a Pythagorean fuzzy deep denoising autoencoder (PFDDAE) model to implement both the *Behavior Learner* and *Control Learner*.

2.2. Pythagorean Fuzzy Deep Denoising Autoencoder

In our approach, both the *Behavior Learner* and *Control Learner* use a denoising autoencoder (DAE) [27] as the basic building block. DAE is an autoencoder (AE) variant [28] that consists of an encoder and a decoder. The encoder transforms a D-dimensional input vector $\mathbf{x} \in [0,1]^D$ into a hidden representation $\mathbf{y} \in [0,1]^{D'}$ through an affine mapping:

$$f(\mathbf{x}) = s(\mathbf{W}\mathbf{x} + \mathbf{b}) \tag{1}$$

where s is an activation function (typically an exponential or sigmoid function), $\mathbf{W}$ is a $D' \times D$ weight matrix, and $\mathbf{b}$ is a D'-dimensional bias vector (typically, we have $D' < D$).

The decoder maps the hidden representation $\mathbf{y}$ back to a reconstructed vector $\mathbf{x}'$ in the input space with appropriately sized parameters $\mathbf{W}^{\mathrm{T}}$ and $\mathbf{b}'$:

$$f'(\mathbf{y}) = s(\mathbf{W}^{\mathrm{T}}\mathbf{y} + \mathbf{b}') \tag{2}$$

AE learning determines appropriate parameters $\theta = \{\mathbf{W}, \mathbf{b}, \mathbf{b}'\}$ to minimize the reconstruction error (e.g., the squared error) over the training set $\mathcal{T}$:

$$\min J(\theta) = \frac{1}{|\mathcal{T}|} \sum_{\mathbf{x} \in \mathcal{T}} \mathcal{L}(\mathbf{x}, \mathbf{x}') \tag{3}$$

To improve the robustness to noise, DAE enhances the basic AE by corrupting any initial input $\mathbf{x}$ into $\widetilde{\mathbf{x}}$ by means of a stochastic mapping $\mathbf{x}$ into $q_{\mathcal{T}}(\widetilde{\mathbf{x}}|\mathbf{x})$ and then maps the corrupted input $\widetilde{\mathbf{x}}$, as with the basic AE, to a hidden representation $\mathbf{y}$ from which we reconstruct an $\mathbf{x}' = f'(\mathbf{y})$.

To capture the highly uncertain relationships between external stimuli and the species behaviors, we further enhance DAE by expressing the model parameters as fuzzy numbers, such that the model can effectively learn the fuzzy probability distribution over cross-layer units [29]. In particular, we employ Pythagorean-type fuzzy numbers (PFN) characterized by both a membership degree and a non-membership degree whose square sum is less than or equal to 1 [30,31] so as to not only allow for a larger body of membership grades than regular and intuitionistic fuzzy numbers but also enable each neuron to learn both how

an input *contributes to* and how it *does not contribute to* the production of the output [32,33]. In this study, we use interval-valued PFN and use $\exp(\cdot)$ as the activation function in Equation (1), where the exponential operation on fuzzy numbers is defined as in [34,35].

Nevertheless, using fuzzy model parameters makes the problem with the objective of minimizing Equation (3) a fuzzy optimization problem. Here, we utilize a centroid method [29,36] to defuzzify the problem. Given a PFN $\beta = P(\mu_\beta, \nu_\beta)$, its centroid point can be calculated as

$$c(\beta) = (c_\mu(\beta), c_\nu(\beta)) = \left(\frac{\int x\mu_\beta(x)dx}{\int \mu_\beta(x)dx}, \frac{\int x\nu_\beta(x)dx}{\int \nu_\beta(x)dx} \right) \tag{4}$$

Then, the distance between a crisp number α and a PFN β is calculated as

$$|\alpha, \beta| = \sqrt{(c_\mu(\beta) - \frac{\alpha}{2})^2 + c_\nu^2(\beta)} \tag{5}$$

Accordingly, for a Pythagorean-type fuzzy DAE, the reconstruction error of a fuzzy vector $\widetilde{\mathbf{x}}'$ from a crisp vector $\mathbf{x}$ (with the same dimension D) is calculated as

$$\mathcal{L}(\mathbf{x}, \widetilde{\mathbf{x}}') = \sqrt{\frac{\sum_{d=1}^{D} |x_d, \widetilde{x}'_d|^2}{D}} \tag{6}$$

which is used in the objective function (3).

As a deep DAE is constructed by stacking layers of DAE [37], we construct a PFDDAE by stacking layers of Pythagorean-type fuzzy DAE, where each layer captures the hidden representation of the layer below as inputs, so as to effectively learn higher-order abstract representations from original input features.

Using fuzzy parameters can effectively improve the representation ability and robustness of the deep learning model [32,38], but it also increases the dimensionality of the problem, for which traditional gradient-based algorithms easily become trapped in local optima [39,40]. To improve the learning performance, we employ an evolutionary algorithm, water wave optimization (WWO) [41], to train the PFDDAE by evolving a population of solutions to explore the search space, making it more capable of jumping out of local optima. The algorithm first randomly initializes a population of solutions, each of which is a vector of network parameters and is evaluated by the reconstruction error of the corresponding model instance. The algorithm then continually evolves the solutions using WWO operators including propagation, refraction, and breaking [41]. The performance of WWO in training deep neural networks has been demonstrated by comparison with other state-of-the-art algorithms [42].

3. Behavior Leaner

3.1. Hierarchical Learning of Cyborg Flight Behaviors

To reduce the complexity of learning cyborg flight behaviors, we divide the *Behavior Learner* into three hierarchies, which are all implemented with PFDDAE.

3.1.1. Learning the Species Common Behaviors

The lower hierarchy of layers is a PFDDAE for learning the common behaviors of the species, using unsupervised pretraining layer-by-layer to minimize the reconstruction error (3) of the PFDDAE.

3.1.2. Learning Group-Specific Behaviors

A moth often exhibits stress behaviors similar to some other moths under the same environment. Grouping moths with similar behaviors and then learning group-specific behaviors can significantly improve the efficiency of model learning. However, there is no effective biological classification method for this purpose [43]. Here, we propose

an improved fuzzy clustering method (described in the next subsection), which groups all moths into c groups and calculates a membership degree u_{ij} of each jth moth to the ith group.

The middle group-specific hierarchy consists of a set of PFDDAE instances, each for learning the specific behaviors of a group of moths. Each group-specific learner takes the topmost representation of the lower common hierarchy as inputs and uses unsupervised pretraining to minimize the reconstruction error. However, here, the reconstruction error is weighted by fuzzy memberships, and the objective function for pretraining the ith group-specific learner is:

$$\min J^{(i)}(\theta) = \frac{1}{|\mathcal{T}|} \sum_{\mathbf{x}_j \in \mathcal{T}} u_{ij} \mathcal{L}(\mathbf{x}_j, \mathbf{x}'_j) \tag{7}$$

Consequently, the larger the membership degree of a moth (input vector) $\mathbf{x}_j$ to a group is, the higher the contribution of its reconstruction error to the objective function.

Ideally, whenever $u_{ij} > 0$, $\mathbf{x}_j$ participates in the training of the ith group-specific learner. However, in practice, to improve the computational efficiency, we set a lower limit u^{L} and only use $\mathcal{T}_i = \{\mathbf{x}_j | u_{ij} > u^{\mathrm{L}}\}$ as the training set for the ith group:

$$\min J^{(i)}(\theta) = \frac{1}{|\mathcal{T}_i|} \sum_{\mathbf{x}_j \in \mathcal{T}_i} u_{ij} \mathcal{L}(\mathbf{x}_j, \mathbf{x}'_j) \tag{8}$$

3.1.3. Learning Individual-Specific Behaviors

For each individual moth, we use a PFDDAE to learn its specific behaviors and utilize a multivariable linear regression (MLR) on the topmost individual-specific layer to produce the output flight parameter values. However, an individual-specific learner takes the outputs from multiple group-specific learners as its inputs. Let D'' be the output dimension of each group-specific learner; then, the input dimension of each individual-specific learner is also D'', and each input component $x_d^{(j)}$ of the jth individual-specific learner is calculated from multiple group-specific learners as follows ($1 \le d \le D''$):

$$x_d^{(j)} = \sum_{i=1}^{c} u_{ij} y_d'^{(i)} \tag{9}$$

where $y_d'^{(i)}$ denotes the dth component of the output vector of the ith group-specific learner. Consequently, the larger the membership degree u_{ij} is, the higher the contribution of the output of the ith group-specific learner to the input of the jth individual-specific learner. Similar to the training set selection for a group-specific learner, in practice, we can only use the outputs from group-specific learners that satisfy $u_{ij} > u^{\mathrm{L}}$ for training the jth individual-specific learner.

After pretraining an individual-specific learner according to the objective function (3) in an unsupervised manner, we train the learner in a supervised manner to minimize the regression error on the training set of the individual moth. However, since different flight parameters have different importance in determining the flight path, we use a weighted loss to evaluate the regression error:

$$\min J_{\mathrm{R}}^{(j)}(\theta) = \frac{1}{|\mathcal{T}|^{(j)}} \sum_{\mathbf{x} \in \mathcal{T}^{(j)}} \sqrt{\sum_{d=1}^{7} w_d (y''_d - \widehat{y}''_d)^2} \tag{10}$$

where $\mathcal{T}^{(j)}$ denotes the set of labeled samples for the jth moth, y''_d is the actual dth output component, $\widehat{y}''_d$ is the expected dth output component, and w_d is the weight importance of the dth output component. According to the statistics of the occurrences of different flight actions and contributions of different flight parameters to the actions (e.g., horizontal

deflection occurs most frequently, and deflection angle is the most important parameter to the horizontal deflection action), we set the weight of the horizontal deflection angle to 0.24, the weight of the horizontal angular velocity to 0.2, the weight of the vertical deflection angle to 0.2, the weight of the vertical angular velocity to 0.15, and the weights of accelerations in the x-, y-, and z-axes to 0.07.

3.2. Pythagorean Fuzzy c-Means Clustering for Moth Grouping

We propose an improved fuzzy clustering method that uses moth shape information together with the outputs of the topmost common layer under three typical environmental settings combined with 12 typical UA ray stimulation settings to group moths.

As we know, fuzzy c-means clustering (FCM) [44] is a method for minimizing the overall fuzzy-membership-weighted distance of the data points from cluster centroids:

$$\min J(U,V) = \frac{1}{cn}\sum_{i=1}^{c}\sum_{j=1}^{n} u_{ij}^{m} d_{ij}^{2} \tag{11}$$

where n is the number of data points, c is the number of clusters, u_{ij} is the membership degree of the jth data point to the ith cluster subject to $\sum_i u_{ij} = 1$, d_{ij} is the distance between the jth data point and the ith cluster centroid, m is a control parameter larger than 1, $U = (u_{ij})_{c\times n}$ is the weight matrix, and $V = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c]$ is the vector of cluster centroids. Xu and Wu [45] extended the standard FCM to an intuitionistic FCM (IFCM) algorithm based on some new definitions of distance measures between intuitionistic fuzzy sets, which can capture more uncertainty information to improve clustering results.

We further improve the clustering method by using Pythagorean-type fuzzy sets to represent clusters. To group D-dimensional PFN data points, each cluster centroid is a D-dimensional PFN vector. The distance between two PFN $\beta_1 = P(\mu_{\beta_1}, \nu_{\beta_1})$ and $\beta_2 = P(\mu_{\beta_2}, \nu_{\beta_2})$ is defined as follows [46]:

$$|\beta_1, \beta_2| = \sqrt{\frac{(\mu_{\beta_1}^2 - \mu_{\beta_2}^2)^2 + (\nu_{\beta_1}^2 - \nu_{\beta_2}^2)^2 + (\pi_{\beta_1}^2 - \pi_{\beta_2}^2)^2}{2}} \tag{12}$$

where $\pi_\beta = \sqrt{1-\mu_\beta^2-\nu_\beta^2}$ is the hesitant degree of β.

For any data point $\mathbf{x}_j$ and cluster centroid $\mathbf{v}_i$, we use the following distance measure to replace d_{ij} in Equation (11):

$$\|\mathbf{x}_j, \mathbf{v}_i\| = \sqrt{\frac{\sum_{d=1}^{D} |x_{jd}, v_{id}|^2}{D}} \tag{13}$$

Based on this new distance measure, we develop a Pythagorean FCM (PFCM) algorithm to cluster Pythagorean fuzzy sets. The pseudocode of the algorithm is shown in Algorithm 1 (where ϵ is a user-defined small positive value for controlling the stopping condition).

The performance of FCM clustering heavily depends on the quality of initial cluster centers [47,48]. Instead of randomly selecting the initial cluster centers, we also employ the WWO metaheuristic to search for optimal or high-quality initial cluster centers. Given the number c of clusters, WWO first randomly initializes a population of solutions, each of which represents a set $V^{(0)}$ of c initial cluster centroids and is evaluated by the objective function (11) of the clustering results derived from $V^{(0)}$. The algorithm then continually evolves the solutions until the stopping criterion is met.

Algorithm 1: Pythagorean fuzzy c-means clustering algorithm.

Initialize a $c \times n$ matrix U and a set $V^{(0)}$ of c cluster centroids;
Let $k = 0$;
while $\|V^{(k+1)}, V^{(k)}\| > \epsilon$ **do**
  for $j = 1$ *to* n **do**
    if $\exists i' : 1 \le i' \le c : \|\mathbf{x}_j, \mathbf{v}_{i'}\| = 0$ **then**
      for $i = 1$ *to* c **do**
        if $i = i'$ **then** $u_{ij}^{(k)} \leftarrow 1$;
        ;
        else $u_{ij}^{(k)} \leftarrow 0$;
        ;
    else
      for $i = 1$ *to* c **do**
        $u_{ij}^{(k)} \leftarrow \frac{1}{\sum_{i'=1}^{c} \left(\frac{\|\mathbf{x}_j, \mathbf{v}_i\|}{\|\mathbf{x}_j, \mathbf{v}_{i'}\|} \right)^{\frac{2}{m-1}}}$;
  for $i = 1$ *to* c **do**
    $\mathbf{v}_i^{(k)} \leftarrow \left\{ \left\langle \beta_d, \frac{\sum_{j=1}^{n} u_{ij}^{(k)} \mu_{\beta_d}^2}{\sum_{j=1}^{n} u_{ij}^{(k)}}, \frac{\sum_{j=1}^{n} u_{ij}^{(k)} \nu_{\beta_d}^2}{\sum_{j=1}^{n} u_{ij}^{(k)}} \right\rangle | 1 \le d \le D \right\}$;
  $k \leftarrow k+1$;
return (U, V);

4. Control Leaner

The *Control Learner* learns which UV ray stimulus values can produce the required flight parameter values of a moth. It uses a PFDDAE to learn a high-order representation of the control mechanism, and it utilizes an MLR on top of the PFDDAE to output the recommended UV ray stimulus values. For a part of the training samples, we perform the output UV ray stimulus on physical moths and compare the actual flight parameter values exhibited by the moths with the required flight parameter values to evaluate the loss. However, because such physical experiments are costly, for a majority of training samples, we send the output UV ray stimulus to the *Behavior Learner* and compare the flight parameter values output by the *Behavior Learner* with the required flight parameter values to evaluate the loss. The two parts have different contributions to the final regression error. Let $\mathcal{T}_c$ be the training set, $\mathcal{T}_p$ be the subset whose outputs are sent to physical moths for comparison, and $\mathcal{T}_m$ be the subset whose outputs are sent to the *Behavior Learner* for comparison. The regression error of the *Control Learner* is evaluated as

$$\min J_c(\theta) = \frac{\sum\limits_{\mathbf{x} \in \mathcal{T}_p} \|\mathbf{x}, o_p\big(o_c(\mathbf{x})\big)\| + w_m \sum\limits_{\mathbf{x} \in \mathcal{T}_m} \|\mathbf{x}, o_m\big(o_c(\mathbf{x})\big)\|}{|\mathcal{T}_c|} \tag{14}$$

where $o_c(\mathbf{x})$ is the *Control Learner*'s output stimulus vector according to input $\mathbf{x}$, $o_p\big(o_c(\mathbf{x})\big)$ is the vector of flight parameter values of the physical moth under given stimuli $o_c(\mathbf{x})$, $o_m\big(o_c(\mathbf{x})\big)$ is the *Behavior Learner*'s output flight parameter values under given stimuli $o_c(\mathbf{x})$, and w_m is a weight to emphasize the importance of physical verification. We set w_m as inversely proportional to the average regression error of the *Behavior Learner* as follows (where each output component of the model is normalized into [0,1] in Equation (10)):

$$w_m = 1 - \frac{\sum_j J_{\mathrm{R}}^{(j)}(\theta)}{|\mathcal{T}_m|} \tag{15}$$

5. Experiments

We conduct experiments to test (1) the performance of PFDDAE pretraining; (2) the performance of fuzzy clustering; (3) the performance of the PFDDAE-based *Behavior Learner*; and (4) the performance of PFDDAE-based *Control Learner*. The data set consists of 10,932 flight records of 36 moths, which were collected in a laboratory environment under different ambient conditions. Each data tuple consists of input UV ray stimuli and output flight parameters (as introduced in Section 2.1) for training the *Behavior Learner*; after training the *Behavior Learner*, the flight parameters are used as inputs to the *Control Learner*, and its output stimuli are input to the *Behavior Learner* or real cyborgs to produce expected/actual flight parameters in order to minimize their differences from the input flight parameters so as to train the *Control Learner*. The algorithms are executed on a workstation with an i7-6500 2.5 GH CPU, 8 GB DDR4 RAM, and an NVIDIA Quadro M500M card (Leadtek Research, Inc., Taiwan, China).

5.1. Experiments on Model Pretraining

We compare the following different algorithms, including gradient-based methods and evolutionary algorithms, for PFDDAE unsupervised pretraining:

- The basic gradient-based layer-wise (GLW) algorithm [28].
- An adaptive gradient (AdaGrad) algorithm [49].
- A non-revisiting genetic algorithm with adaptive mutation (NrGA) [50].
- A comprehensive learning PSO (CLPSO) algorithm [51] where each solution learns from different exemplars at different dimensions.
- A self-adaptive differential evolution (SaDE) algorithm [52], which adaptively chooses more prospective evolution strategies among a set of candidate strategies.
- A biogeography-based optimization (BBO) algorithm [53], which evolves solutions by continuously migrating features from high-fitness individuals to low-fitness ones based on a biogeographical migration model.
- An improved BBO algorithm called ecogeography-based optimization (EBO) [54], which defines two migration operators, namely global migration and local migration, that are adaptively applied according to the maturity of the population.
- The WWO algorithm [42].

On the data set, we tune the number of layers of each PFDDAE-based learner between two and five, and we find that it is sufficiently good to use three layers for all sub-learners of the *Behavior Learner* and use four layers for the *Control Learner*. We then tune the number of neurons of each layer between the square root and one-half of that of the previous layer. As a result, we set the model structure as follows:

- The three layers of the lower common learner have 80, 46, and 26 neurons, respectively;
- The three layers of each group-specific learner have 26, 15, and 9 neurons, respectively;
- The three layers of each individual-specific learner have 9, 6, and 7 neurons, respectively;
- The four layers of the *Control Learner* have 75, 49, 35, and 32 neurons, respectively.

To avoid overfitting, we conduct a five-fold cross-validation, i.e., the dataset is partitioned into five equal-sized pieces, and the validation is run five times, each using four pieces as the training set and using the remaining piece as the test set. A validation runs each algorithm 20 times with different random seeds. For the six evolutionary algorithms, the stopping criterion is set to that the number of objective function evaluations reaches 100,000 to ensure a fair comparison.

Figure 3 presents the average reconstruction errors and standard deviations of the comparative algorithms for pretraining the lower common learner of the *Behavior Learner*. The basic GLW algorithm exhibits the worst performance. All seven evolutionary learning algorithms achieve significant performance improvement over GLW, which demonstrates that population-based evolutionary algorithms can efficiently explore the high-dimensional solution space and therefore effectively overcome premature convergence. Nevertheless, not all evolutionary algorithms can outperform AdaGrad, which is another gradient-based

algorithm. This indicates that to achieve promising training performance, we need to carefully design or adapt the evolutionary algorithms for this high-dimensional optimization problem. Among all eight algorithms, the proposed WWO algorithm obtains the lowest average reconstruction error, which evidences the efficiency of the nature-inspired metaheuristic for this complex optimization problem.

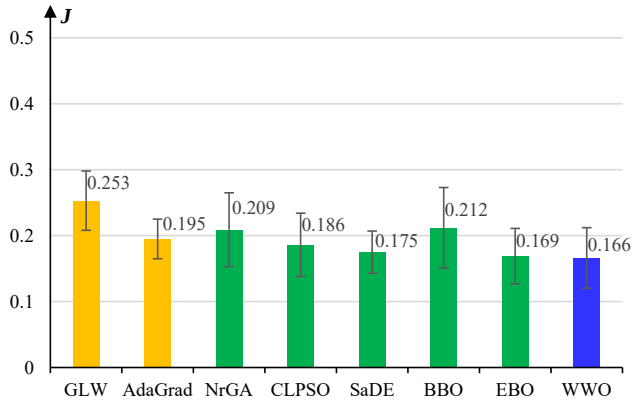

Figure 3. Average reconstruction errors and standard deviations of the comparative algorithms for pretraining the lower *common* layers of the *Behavior Learner*.

Figure 4 presents the experimental results of the comparative algorithms for pretraining the middle group-specific learners. Similar to the experiments on the lower learner, GLW exhibits the worst performance and WWO exhibits the best performance. However, the performance advantages of WWO over the other evolutionary algorithms become less significant, because the number of neurons of a group-specific learner is nearly one-third that of the lower common learner, and hence, the problem dimension becomes much smaller.

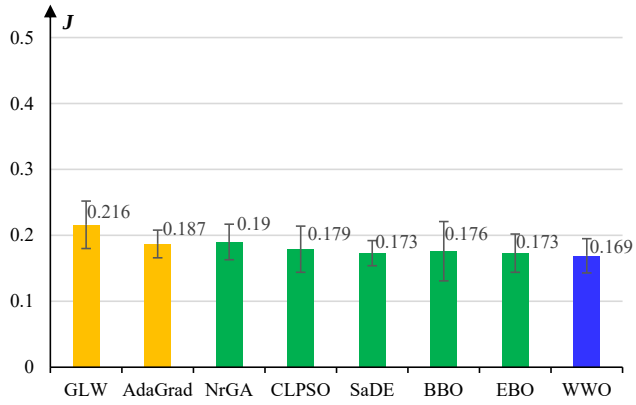

Figure 4. Average reconstruction errors and standard deviations of the comparative algorithms for pretraining the middle *group-specific* layers of the *Behavior Learner*.

Figure 5 presents the experimental results of the comparative algorithms for pretraining the upper individual-specific learners. The number of neurons of an individual-specific learner is less than half that of a group-specific learner, and the problem dimension becomes even smaller. In this experimental section, except for the GLW that still exhibits the worst performance, the performance differences among the other algorithms are not very obvious. Consequently, for the *Behavior Learner*, we use WWO to pretrain the lower common layers and the middle group-specific layers, but employ AdaGrad to pretrain upper individual-specific layers—although most of the evolutionary learning algorithms exhibit slight performance advantages, AdaGrad consumes considerably fewer computational resources, especially when the number of individual moths is large.

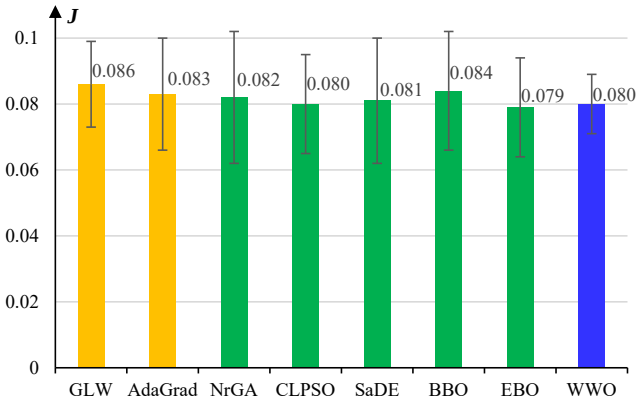

Figure 5. Average reconstruction errors and standard deviations of the comparative algorithms for pretraining the upper *individual-specific* layers of the *Behavior Learner*.

Figure 6 presents the experimental results of the comparative algorithms for pretraining the *Control Learner*. For this high-dimensional optimization problem, we observe that the performances of some algorithms, including GLW, NrGA, and BBO, decrease significantly, while SaDE, EBO, and WWO still achieve high learning performances, and WWO obtains the lowest average reconstruction error among all the comparative algorithms. In general, the experimental results on model pretraining show that the larger the problem dimension is, the more significant the performance advantages of the well-designed evolutionary learning algorithms (especially the proposed WWO) over the classical learning algorithms.

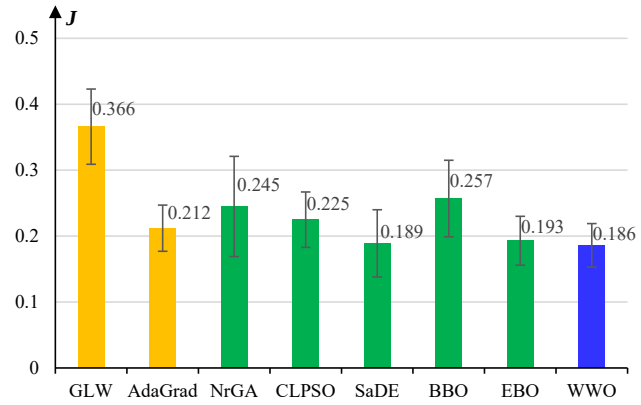

Figure 6. Average reconstruction errors and standard deviations of the comparative algorithms for pretraining the *Control Learner*.

5.2. Experiments on Fuzzy c-Means Clustering

We compare different clustering algorithms, including the basic FCM, IFCM, PFCM, and PFCM enhanced by evolutionary algorithms including NrGA [50], CLPSO [51], SaDE [52], BBO [53], EBO [54], and WWO [41]. After tuning the control parameters of the model on the data set, we set the number of clusters to five (too many clusters would consume a lot of computational resources, while fewer clusters cannot effectively discriminate individual moths and appropriately define group behaviors) and set $m = 2$. We run each algorithm 20 times on the clustering problem.

Figure 7 presents the average $J(U, V)$ value of the clustering results of each comparative algorithm. By using intuitionistic fuzzy sets that have more expression ability than basic fuzzy sets, IFCM yields significantly higher clustering cohesion than the basic FCM. Pythagorean fuzzy sets further enlarger the body of membership grades, and PFCM exhibits higher clustering performance than IFCM.

FCM, IFCM, and PFCM all exhibit large standard deviations, i.e., their clustering results are quite unstable, because they use random cluster centroids. In comparison, evolutionary algorithms significantly improve the performance of PFCM by evolutionary searching for the optimal setting of cluster centroids. Among the six evolutionary algorithms, EBO obtains the best average $J(U, V)$ value and WWO ranks second, but the standard deviation of WWO is much less than that of EBO. According to paired *t*-tests, there are no statistically significant differences among SaDE, BBO, EBO, and WWO. We select WWO because of its relatively high accuracy and robustness in clustering.

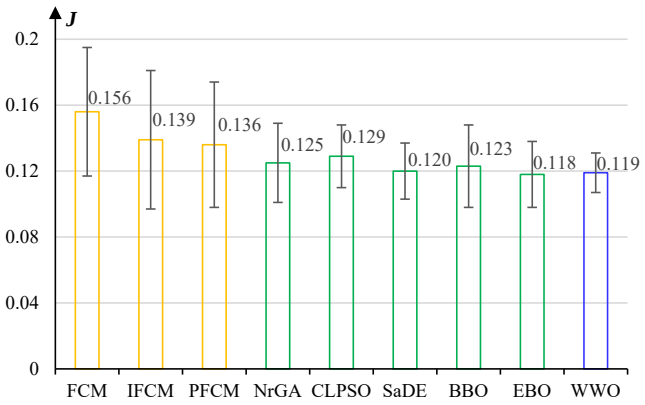

Figure 7. Average $J(U, V)$ values and standard deviations of the clustering results of the comparative algorithms for moth grouping.

5.3. Experiments on Cyborg Flight Behavior Learning

After validating the performance of model pretraining and fuzzy clustering, we test the performance of the entire *Behavior Learner* for cyborg flight parameter prediction. The following models are implemented for comparison with the proposed PFDDAE model.

- A standard three-layer back-propagation artificial neural network (ANN). The numbers of neurons in the three layers are tuned to 80, 25, and 7, respectively.
- A basic self-adaptive neuro-fuzzy inference system (SANFIS) trained by an agglomerative clustering algorithm and a recursive least-squares algorithm [55].
- An evolving interval type-2 neuro-fuzzy inference system (IT2FIS) trained by a metacognitive sequential learning algorithm [56].
- A basic and integrated deep AE model (denoted by D-AE) [57], which employs the basic AE without a denoising mechanism as the building block and does not hierarchically divide the model into common learner, group-specific learners and individual-specific learners. After fine-tuning, the number of layers of D-AE is set to five, and the numbers of neurons in the three hidden layers are set to 48, 27, and 13, respectively.
- A basic and integrated deep DAE model (denoted by DDAE) [37], which uses the same structure as D-AE but employs DAE as the building block.
- A basic hierarchical deep DAE (denoted by HDDAE) that uses three hierarchies as described in Section 2 but does not employ fuzzy model parameters.
- A hierarchical fuzzy deep DAE (denoted by FDDAE) that uses three hierarchies as described in Section 2 but employs regular fuzzy numbers to represent model parameters [29].
- Three variants of the proposed model, which employs the *k*-means clustering (denoted by PFDDAE-kc), basic FCM [44] (denoted by PFDDAE-fc), and IFCM [45] (denoted by PFDDAE-ifc) instead of PFCM for moth grouping, respectively.

Figure 8 presents the box plots of the test results in terms of the objective function (10) obtained by the comparative models on the data set. The ANN has the highest error rate (46% on average) and instability, indicating that the shallow learning model is very ineffective for the complex flight parameter learning problem. The error rates of other models

are much lower than that of ANN, which demonstrates that deep learning models can significantly improve learning performance by using multiple layers to discover intermediate representations. The average error rates of SANFIS, IT2FIS, D-AE, and DDAE are 37.2%, 33.9%, 35.6%, and 33.7%, respectively. Paired *t*-test results show that DDAE has significant performance improvement over SANFIS and D-AE, because inputs to the model typically contain much noise, and the denoising mechanism of DAE can reduce the effect of background noise. However, their error rates are still high and are unacceptable for engineering applications.

By dividing the deep learning model into three hierarchies for gradually learning the species' common behaviors, group-specific behaviors, and individual-specific behaviors, our hierarchical deep learning model can effectively divide and conquer the model complexity and thus achieve significant performance improvement over not only the shallow ANN but also the monolithic neuro-fuzzy inference models and deep DAE models.

HDDAE, FDDAE, and PFDDAE-fc use the same hierarchical structure and FCM clustering method. According to paired *t*-tests, the results of FDDAE are significantly better than those of HDDAE, and the results of PFDDAE-fc are significantly better than those of FDDAE. This demonstrates that compared with using crisp model parameters, using fuzzy parameters can effectively improve the model's representation ability and robustness; moreover, compared with using regular fuzzy parameters, using Pythagorean-type fuzzy parameters can further improve the learning performance by enabling each neuron to learn the contribution of input features to the output from both the positive and negative sides.

Comparing the results of the last four PFDDAE models, we also observe that the PFDDAE-fc model using the standard FCM clustering outperforms the PFDDAE-kc model using *k*-means clustering, and using intuitionistic and Pythagorean fuzzy sets can generate more valuable information for improving moth grouping and hence achieve better learning performance. Our PFDDAE model achieves an average error rate of 19.3%, which is the lowest among the 11 comparative models. The results demonstrate the performance advantages of the proposed PFDDAE using PFN parameters combined with PFCM clustering in learning cyborg behaviors.

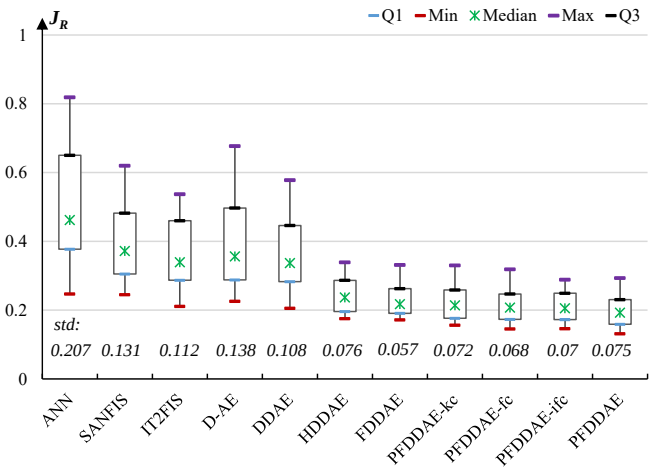

Figure 8. The performance (in terms of regression error) of the comparative models for cyborg flight behavior learning.

5.4. Experiments on Cyborg Flight Control

Finally, we compare the performance of the PFDDAE-based *Control Learner* with other models including a fuzzy proportional–integral–derivative (F-PID) controller optimized by GA [58], ANN, an ANN with a filter (denoted by I-ANN) [59], a fuzzy nonlinear internal model control (FNIMC) with a robustness filter [60], SANFIS [55], IT2FIS [56], D-AE [57], DDAE [37], and FDDAE for cyborg flight control. The experiments use 20 moths and test 200 instructions for each moth. The performance is evaluated in two aspects. The first is the mean value of the objective function (14) obtained by each algorithm. The second is the

success rate (SR) of flight instructions produced by each algorithm in actual moth flight control. We consider an instruction to be successful if, for each relevant flight parameter, the deviation of the actual output value from the expected value is less than 15%.

Figure 9 presents the box plots of the test results in terms of the objective function (14) and SR obtained by the comparative models. The traditional fuzzy PID and shallow ANN obtain high average error rates (approximately 40%) and low success rates (approximately 50%), indicating that they are not suitable for this difficult control problem. Compared to the simple ANN, ANN with a filter obtains a significantly lower error rate and higher success rate, but its performance is still much worse than the adaptive neuro-fuzzy models and deep learning models. The error rates of FNIMC, SANFIS, IT2FIS, and D-AE are around 27–30%, and their success rates are approximately 70%. By equipping D-AE with the denoising mechanism, DDAE decreases the average error rate to 25% and increases the success rate to 77%, which again demonstrates the importance of denoising in this sensitive control problem under complex environments. The error rates of FDDAE and PFDDAE are approximately 22% and 20%, and their success rates are over 79% and 83%, respectively, demonstrating that expressing model parameters with fuzzy numbers (in particular PFN) effectively enhances the model's representation ability and robustness to improve control performance.

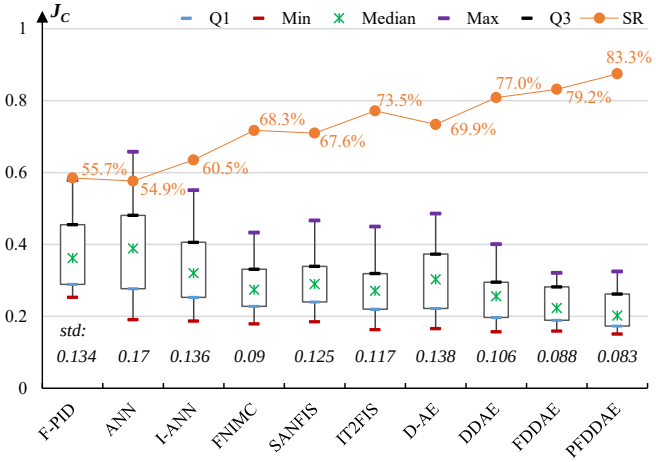

Figure 9. The performance (in terms of regression error and success rate) of the comparative models for cyborg flight control.

From the test results, we can also observe that for each model, the average error rate is inversely proportional to the success rate, which demonstrates the reasonability and practicability of the objective function of the control problem. In summary, the experiments show that the proposed PFDDAE exhibits both the best behavior learning performance and the best control performance among all comparative models, and its high success rate of over 83% indicates that it is suitable for this difficult cyborg flight control problem.

6. Conclusions and Discussion

In this paper, we present a fuzzy deep learning approach to the flight control of cyborg moths stimulated by UV rays. We propose a PFDDAE model to capture the highly uncertain relationships between external stimuli and the species behaviors. The PFDDAE model is employed for both the *Behavior Learner* for learning the flight behaviors of moths and the *Control Learner* for learning which UV ray stimulation can cause moths to exhibit required flight behaviors. To reduce complexity, the *Behavior Learner* is further divided into three hierarchies for learning the species' common behaviors, group-specific behaviors, and individual-specific behaviors. The independent unsupervised pretraining of these parts significantly simplifies the task of model learning. We also propose the PFCM clustering method to group moths for learning group-specific behaviors. Experimental results demonstrate the performance advantages of the proposed approach, which obtains

the lowest average error rate of 19.3% among the 11 comparative models for training the *Behavior Learner*, the lowest average error rate of 20% among the ten comparative models for training the *Control Learner*, and a high control success rate of over 83% that can provide sufficiently accurate control of the flight parameters of moths.

Cyborg insects have numerous potential applications. Our ongoing work includes studying solution methods for optimally controlling a cyborg moth as well as a swarm of moths to perform specific tasks such as path planning, surveillance and detection, and search and rescue [61,62]. For such complex control optimization problems, nature-inspired evolutionary algorithms are good alternatives to classical mathematical methods. We also plan to extend our cyborg moth control learning model to other cyborg insects, for which domain adaption and transfer learning methods [63] are expected to be useful.

Author Contributions: Conceptualization, X.-L.J. and B.W.; methodology, X.Y.; validation, X.-L.J. and Z.-L.S.; investigation, Z.-L.S.; writing—original draft preparation, X.Y.; writing—review and editing, Z.-L.S.; project administration, B.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: All animal experiments were conducted in accordance with the guidelines for animal experiments of the Guiding Principles of Animal Experiments by the Chinese Society of Laboratory Animal Sciences, and permission was obtained from the local Ethics Committee of Hangzhou Normal University.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare that there is no conflict of interests.

Abbreviations

The following abbreviations are used in this manuscript:

AdaGrad	Adaptive gradient
AE	Autoencoder
ANN	Artificial neural network
BBO	Biogeography-based optimization
DAE	Denoising autoencoder
DDAE	Deep denoising autoencoder
EBO	Ecogeography-based optimization
FCM	Fuzzy *c*-means clustering
GLW	Gradient-based layer-wise
IT2FIS	Interval type-2 neuro-fuzzy inference system
LED	Light-emitting diode
MLR	Multivariable linear regression
PFDDAE	Pythagorean fuzzy deep denoising autoencoder
PFN	Pythagorean-type fuzzy number
PSO	Particle swam optimization
SANFIS	Self-adaptive neuro-fuzzy inference system
UV	Ultraviolet
WWO	Water wave optimization

References

1. Dutta, A. Cyborgs: Neuromuscular Control of Insects. In Proceedings of the 9th International IEEE/EMBS Conference on Neural Engineering, San Francisco, CA, USA, 20–23 March 2019; pp. 682–685. [CrossRef]
2. Fu, Y.; Yu, H.; Zhang, X.; Malgaretti, P.; Kishore, V.; Wang, W. Microscopic Swarms: From Active Matter Physics to Biomedical and Environmental Applications. *Micromachines* **2022**, *13*, 295. [CrossRef] [PubMed]
3. Zheng, Y.J.; Wang, Y.; Ling, H.F.; Xue, Y.; Chen, S.Y. Integrated civilian-military pre-positioning of emergency supplies: A multiobjective optimization approach. *Appl. Soft Comput.* **2017**, *58*, 732–741. [CrossRef]
4. Zheng, Y.J.; Du, Y.C.; Su, Z.L.; Ling, H.F.; Zhang, M.X.; Chen, S.Y. Evolutionary human-UAV cooperation for transmission network restoration. *IEEE Trans. Ind. Inf.* **2021**, *17*, 1648–1657. [CrossRef]

5. Romano, D.; Donati, E.; Benelli, G.; Stefanini, C. A review on animal–robot interaction: From bio-hybrid organisms to mixed societies. *Biol. Cybern.* **2019**, *113*, 201–225. [CrossRef]
6. Holzer, R.; Shimoyama, I. Locomotion control of a bio-robotic system via electric stimulation. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robot and Systems (IROS'97), Grenoble, France, 11 September 1997; Volume 3, pp. 1514–1519. [CrossRef]
7. Sanchez, C.J.; Chiu, C.W.; Zhou, Y.; González, J.M.; Vinson, S.B.; Liang, H. Locomotion control of hybrid cockroach robots. *J. R. Soc. Interface* **2015**, *12*, 20141363. [CrossRef]
8. Erickson, J.C.; Herrera, M.; Bustamante, M.; Shingiro, A.; Bowen, T. Effective Stimulus Parameters for Directed Locomotion in Madagascar Hissing Cockroach Biobot. *PLoS ONE* **2015**, *10*, e0134348. [CrossRef]
9. Li, G.; Zhang, D. Brain-Computer Interface Controlling Cyborg: A Functional Brain-to-Brain Interface Between Human and Cockroach. In *Brain-Computer Interface Research*; Springer: Cham, Switzerland, 2017; pp. 71–79. [CrossRef]
10. Sato, H.; Berry, C.W.; Casey, B.E.; Lavella, G.; Yao, Y.; VandenBrooks, J.M.; Maharbiz, M.M. A cyborg beetle: Insect flight control through an implantable, tetherless microsystem. In Proceedings of the IEEE 21st International Conference on Micro Electro Mechanical System, Tucson, AZ, USA, 13–17 January 2008; pp. 164–167. [CrossRef]
11. Vo Doan, T.T.; Li, Y.; Cao, F.; Sato, H. Cyborg beetle: Thrust control of free flying beetle via a miniature wireless neuromuscular stimulator. In Proceedings of the 28th IEEE International Conference on Micro Electro Mechanical Systems (MEMS), Estoril, Portugal, 18–22 January 2015; pp. 1048–1050. [CrossRef]
12. Cao, F.; Zhang, C.; Choo, H.Y.; Sato, H. Insect-computer hybrid legged robot with user-adjustable speed, step length and walking gait. *J. R. Soc. Interface* **2016**, *13*, 20160060. [CrossRef]
13. Nguyen, H.D.; Tan, P.Z.; Sato, H.; Vo-Doan, T.T. Sideways Walking Control of a Cyborg Beetle. *IEEE Trans. Med. Robot. Bionics* **2020**, *2*, 331–337. [CrossRef]
14. Bozkurt, A.; Gilmour, R.F., Jr.; Lal, A. Balloon-Assisted Flight of Radio-Controlled Insect Biobots. *IEEE Trans. Bio-Med. Eng.* **2009**, *56*, 2304–2307. [CrossRef]
15. Tsang, W.M.; Stone, A.; Aldworth, Z.; Otten, D.; Akinwande, A.I.; Daniel, T.; Hildebrand, J.G.; Levine, R.B.; Voldman, J. Remote control of a cyborg moth using carbon nanotube-enhanced flexible neuroprosthetic probe. In Proceedings of the IEEE 23rd International Conference on Micro Electro Mechanical Systems (MEMS 2010), Wanchai, Hong Kong, 24–28 January 2010; pp. 39–42. [CrossRef]
16. Schwefel, J.; Ritzmann, R.E.; Lee, I.N.; Pollack, A.; Weeman, W.; Garverick, S.; Willis, M.; Rasmussen, M.; Scherson, D. Wireless Communication by an Autonomous Self-Powered Cyborg Insect. *J. Electrochem. Soc.* **2015**, *161*, 3113–3116. [CrossRef]
17. Giampalmo, S.L.; Absher, B.F.; Bourne, W.T.; Steves, L.E.; Vodenski, V.V.; O'Donnell, P.M.; Erickson, J.C. Generation of complex motor patterns in american grasshopper via current-controlled thoracic electrical interfacing. In Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Boston, MA, USA, 30 August–3 September 2011; pp. 1275–1278. [CrossRef]
18. Mehta, D.; Altan, E.; Chandak, R.; Raman, B.; Chakrabartty, S. Behaving cyborg locusts for standoff chemical sensing. In Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), Baltimore, MD, USA, 28–31 May 2017; pp. 1–4. [CrossRef]
19. Bao, L.; Zheng, N.; Zhao, H.; Hao, Y.; Zheng, H.; Hu, F.; Zheng, X. Flight control of tethered honeybees using neural electrical stimulation. In Proceedings of the 5th International IEEE/EMBS Conference on Neural Engineering, Cancun, Mexico, 27 April–1 May 2011; pp. 558–561. [CrossRef]
20. Wang, S.; Shen, L.; Liu, X.; Liao, H. A wearable backpack chip for honeybee biorobot. In Proceedings of the China Semiconductor Technology International Conference (CSTIC), Shanghai, China, 13–14 March 2016; pp. 1–3. [CrossRef]
21. Jamali, M.; Jamali, Y.; Golshani, M. Theory of cyborg: A new approach to fish locomotion control. *arXiv* **2019**, arXiv:1904.12460.
22. Montrose, V.; Carroll, G.; Smith, R.; Oxley, J. *Cyborg Insects: Use or Abuse?* Department of Animal and Agriculture, University Centre Hartpury: Hartpury, UK, 2017.
23. Zheng, N.; Jin, M.; Hong, H.; Huang, L.; Gu, Z.; Li, H. Real-time and precise insect flight control system based on virtual reality. *Electr. Lett.* **2017**, *53*, 387–389. [CrossRef]
24. Zheng, N.; Ma, Q.; Jin, M.; Zhang, S.; Guan, N.; Yang, Q.; Dai, J. Abdominal-Waving Control of Tethered Bumblebees Based on Sarsa with Transformed Reward. *IEEE Trans. Cybern.* **2019**, *49*, 3064–3073. [CrossRef]
25. Wu, G.D.; Zhu, Z.W.; Huang, P.H. A TS-Type Maximizing-Discriminability-Based Recurrent Fuzzy Network for Classification Problems. *IEEE Trans. Fuzzy Syst.* **2011**, *19*, 339–352. [CrossRef]
26. Zheng, Y.J.; Ling, H.F.; Chen, S.Y.; Xue, J.Y. A hybrid neuro-fuzzy network based on differential biogeography-based optimization for online population classification in earthquakes. *IEEE Trans. Fuzzy Syst.* **2015**, *23*, 1070–1083. [CrossRef]
27. Vincent, P.; Larochelle, H.; Bengio, Y.; Manzagol, P.A. Extracting and Composing Robust Features with Denoising Autoencoders. In Proceedings of the International Conference on Machine Learning, Helsinki, Finland, 5–9 July 2008; ACM: New York, NY, USA, 2008; pp. 1096–1103. [CrossRef]
28. Bengio, Y.; Lamblin, P.; Popovici, D.; Larochelle, H. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems (NIPS'06)*; Bernhard Schölkopf, J.P., Hoffman, T., Eds.; MIT Press: Cambridge, MA, USA, 2007; Volume 19, pp. 153–160.

29. Chen, C.; Zhang, C.Y.; Chen, L.; Gan, M. Fuzzy Restricted Boltzmann Machine for the Enhancement of Deep Learning. *IEEE Trans. Fuzzy Syst.* **2015**, *23*, 2163–2173. [CrossRef]
30. Yager, R. Pythagorean fuzzy subsets. In Proceedings of the Joint IFSA World Congress and NAFIPS Annual Meeting IFSA/NAFIPS, Edmonton, AB, Canada, 24–28 June 2013; pp. 57–61. [CrossRef]
31. Zhang, X.; Xu, Z. Extension of TOPSIS to multiple criteria decision making with pythagorean fuzzy sets. *Int. J. Intell. Syst.* **2014**, *29*, 1061–1078. [CrossRef]
32. Zheng, Y.J.; Chen, S.Y.; Xue, Y.; Xue, J.Y. A Pythagorean-type fuzzy deep denoising autoencoder for industrial accident early warning. *IEEE Trans. Fuzzy Syst.* **2017**, *25*, 1561–1575. [CrossRef]
33. Zheng, Y.J.; Sheng, W.G.; Sun, X.M.; Chen, S.Y. Airline passenger profiling based on fuzzy deep machine learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2017**, *28*, 2911–2923. [CrossRef]
34. Gou, X.; Xu, Z.; Liao, H. Exponential operations of interval-valued intuitionistic fuzzy numbers. *Int. J. Mach. Learn. Cybern.* **2016**, *7*, 501–518. [CrossRef]
35. Garg, H. New exponential operational laws and their aggregation operators for interval-valued Pythagorean fuzzy multicriteria decision-making. *Int. J. Intell. Syst.* **2018**, *33*, 653–683. [CrossRef]
36. Hung, W.L.; Wu, J.W. Correlation of intuitionistic fuzzy sets by centroid method. *Inform. Sci.* **2002**, *144*, 219–225. [CrossRef]
37. Vincent, P.; Larochelle, H.; Lajoie, I.; Bengio, Y.; Manzagol, P.A. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.* **2010**, *11*, 3371–3408.
38. Sharkawy, A.N. Principle of neural network and its main types. *J. Adv. Appl. Comput. Math.* **2020**, *7*, 8–19. [CrossRef]
39. Yao, X. A review of evolutionary artificial neural networks. *Int. J. Intell. Syst.* **1993**, *8*, 539–567. [CrossRef]
40. Wojtowytsch, S.; E, W. Can Shallow Neural Networks Beat the Curse of Dimensionality? A mean field training perspective. *IEEE Trans. Artif. Intell.* **2021**, *1*, 121–129. [CrossRef]
41. Zheng, Y.J. Water wave optimization: A new nature-inspired metaheuristic. *Comput. Oper. Res.* **2015**, *55*, 1–11. [CrossRef]
42. Zhou, X.H.; Zhang, M.X.; Xu, Z.G.; Cai, C.Y.; Huang, Y.J.; Zheng, Y.J. Shallow and deep neural network training by water wave optimization. *Swarm Evol. Comput.* **2019**, *50*, 1–13. [CrossRef]
43. Sarto i Monteys, V.; Acín, P.; Rosell, G.; Quero, C.; Jiménez, M.A.; Guerrero, A. Moths behaving like butterflies. Evolutionary loss of long range attractant pheromones in castniid moths: A Paysandisia archon model. *PLoS ONE* **2012**, *7*, e29282. [CrossRef] [PubMed]
44. Bezdek, J. *Pattern Recognition with Fuzzy Objective Function Algorithms*; Plenum Press: Logan, UT, USA, 1981.
45. Xu, Z.; Wu, J. Intuitionistic fuzzy C-means clustering algorithms. *J. Syst. Eng. Electron.* **2010**, *20*, 580–590. [CrossRef]
46. Ren, P.; Xu, Z.; Gou, X. Pythagorean fuzzy TODIM approach to multi-criteria decision making. *Appl. Soft Comput.* **2016**, *42*, 246–259. [CrossRef]
47. Gong, M.; Liang, Y.; Shi, J.; Ma, W.; Ma, J. Fuzzy C-Means Clustering with Local Information and Kernel Metric for Image Segmentation. *IEEE Trans. Image Process.* **2013**, *22*, 573–584. [CrossRef]
48. Zhang, M.; Jiang, W.; Zhou, X.; Xue, Y.; Chen, S. A hybrid biogeography-based optimization and fuzzy C-means algorithm for image segmentation. *Soft Comput.* **2019**, *23*, 2033–2046. [CrossRef]
49. Duchi, J.; Hazan, E.; Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.* **2011**, *12*, 2121–2159.
50. Yuen, S.Y.; Chow, C.K. A Genetic Algorithm That Adaptively Mutates and Never Revisits. *IEEE Trans. Evol. Comput.* **2009**, *13*, 454–472. [CrossRef]
51. Liang, J.J.; Qin, A.K.; Suganthan, P.; Baskar, S. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Trans. Evol. Comput.* **2006**, *10*, 281–295. [CrossRef]
52. Qin, A.K.; Huang, V.L.; Suganthan, P. Differential Evolution Algorithm with Strategy Adaptation for Global Numerical Optimization. *IEEE Trans. Evol. Comput.* **2009**, *13*, 398–417. [CrossRef]
53. Simon, D. Biogeography-Based Optimization. *IEEE Trans. Evol. Comput.* **2008**, *12*, 702–713. [CrossRef]
54. Zheng, Y.J.; Ling, H.F.; Xue, J.Y. Ecogeography-Based Optimization: Enhancing Biogeography-Based Optimization with Ecogeographic Barriers and Differentiations. *Comput. Oper. Res.* **2014**, *50*, 115–127. [CrossRef]
55. Wang, J.S.; Lee, C.S.G. Self-adaptive neuro-fuzzy inference systems for classification applications. *IEEE Trans. Fuzzy Syst.* **2002**, *10*, 790–802. [CrossRef]
56. Das, A.K.; Subramanian, K.; Sundaram, S. An Evolving Interval Type-2 Neurofuzzy Inference System and Its Metacognitive Sequential Learning Algorithm. *IEEE Trans. Fuzzy Syst.* **2015**, *23*, 2080–2093. [CrossRef]
57. Hinton, G.E.; Salakhutdinov, R.R. Reducing the dimensionality of data with neural networks. *Science* **2006**, *313*, 504–507. [CrossRef] [PubMed]
58. Tang, K.S.; Man, K.F.; Chen, G.; Kwong, S. An optimal fuzzy PID controller. *IEEE Trans. Ind. Electron.* **2001**, *48*, 757–765. [CrossRef]
59. Hunt, K.; Sbarbaro, D. Neural networks for nonlinear internal model control. *IEE Proc. D Control Theory Appl.* **1991**, *138*, 431–438. [CrossRef]
60. Boukezzoula, R.; Galichet, S.; Foulloy, L. Nonlinear internal model control: Application of inverse model based fuzzy control. *IEEE Trans. Fuzzy Syst.* **2003**, *11*, 814–829. [CrossRef]
61. Le Thi Thuy, N.; Nguyen Trong, T. The Multitasking System of Swarm Robot based on Null-Space-Behavioral Control Combined with Fuzzy Logic. *Micromachines* **2017**, *8*, 357. [CrossRef]

62. Zheng, Y.; Du, Y.; Ling, H.; Sheng, W.; Chen, S. Evolutionary collaborative human-UAV search for escaped criminals. *IEEE Trans. Evol. Comput.* **2020**, *24*, 217–231. [CrossRef]
63. Song, Q.; Zheng, Y.J.; Sheng, W.G.; Yang, J. Tridirectional transfer learning for predicting gastric cancer morbidity. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 561–574. [CrossRef]