2023 International Symposium on Nonlinear Theory and Its Applications
NOLTA2023, September 26-29, 2023, Catania and Online

Deep-Learning-Based Time-Series Analysis of Insect Behavior

Keigo Tsutsui[†], Phuoc Thanh Tran-Ngoc[‡], Hirotaka Sato[‡], Takashi Matsubara[†]

†Graduate School of Engineering Science, Osaka University,
1-3 Machikaneyama, Toyonaka, Osaka, 560-8531 Japan
‡School of Mechanical and Aerospace Engineering, Nanyang Technological University,
Fifty Nanyang Avenue, Singapore 639798

Email: tsutsui@hopf.sys.es.osaka-u.ac.jp, TRANNGOC001@e.ntu.edu.sg,
hirosato@ntu.edu.sg, matsubara@sys.es.osaka-u.ac.jp

Abstract— Self-organization in lives and other organisms is a phenomenon wherein an entire group forms a complex mechanism through simple interactions among individuals. Utilizing this characteristic, we consider controlling a group of insects by manipulating only a few individuals in the group. To achieve this, it is necessary to model how the behavior of an entire group is affected by individual-scale interactions. Beginning with the analysis of a single insect's behavior, we used deep-learning models to model the paths of insects. As the purpose of each model is to control an insect, it is sufficient to predict future coordinates relative to the current position, rather than in an absolute sense. Accordingly, we normalized the data to unify the absolute positions and directions and reduced the complexity of the dataset to facilitate model learning. Consequently, the models achieved higher scores following normalization.

1. Introduction

Self-organization is a phenomenon in which a set of simple rules generates complex movements and patterns [1]. In lives such as insects, an entire group forms a complex mechanism through the simple interactions of individuals. If this characteristic can be utilized, insect groups can become exemplary for working in small spaces, such as rubble gaps during disasters. We assumed that it is possible to control the behavior of an entire group by manipulating only a few individuals. To control group behavior, it is necessary to model the group's behavioral patterns. Although physical methods have been developed for this purpose [2], manual analyses may incur small errors. Therefore, we utilized several models to extract potential relationships to predict the movement paths of Madagascar hissing cockroaches. As the first step in modeling group behavior, we began with a single insect model.

In the present study, we examined the vector autoregression (VAR) [3], long short-term memory (LSTM) [4], and multilayer perceptron (MLP) [5] models. Although VAR is not a deep-learning model, simpler models are often used to gradually explore more complex ones [6]. Insect path data, obtained by attaching sensors to insects and allowing them to walk across a disc, were represented using two-dimensional coordinates. From each time series, the consecutive coordinates of multiple steps were extracted for use as input to predict the next position of the insect. The inputs were represented in absolute coordinates with varying positions and directions. Because the purpose of the model was to control an insect's path, it was sufficient to predict future coordinates relative to the current position, rather than in absolute terms. To simplify the data and facilitate the learning process, we performed rotational-normalization. Specifically, the data were rotated such that the current position was designated as the origin and the direction of the previous step was aligned with the coordinate axes. Consequently, normalization enabled models to predict the paths of insects more accurately.

ORCID iDs Keigo Tsutsui: 0009-0003-0082-3840, Phuoc Thanh Tran-Ngoc: 0000-0002-5298-1846, Hirotaka Sato: 0000-0003-4634-1639, Takashi Matsubara: 0000-0003-0642-4800

2. Methods

2.1. Rotational-normalization

Each model's input was a set of five consecutive points extracted from a single insect path. All sets of points were situated at various coordinates and faced different directions. To accurately predict and control insect behaviors, we normalized the data via shifting and rotation, ensuring that all points shared the same last-step coordinates and faced the same direction. The normalization process simplified the dataset and enabled positional and rotational invariance, wherein consistent relationships were maintained between points regardless of position or orientation. We expect that normalization makes it easier for models to learn input features. Figure 1 presents examples of rotational-normalization. The entire set of points was shifted and rotated such that the last point moved to position (0,0), and the immediate previous point moved to the negative side of the y-axis. We denote the t−th time step by $\boldsymbol{x}_t \in \mathbb{R}^2$. We now describe the procedure for normalizing a set of points $\boldsymbol{x}_t, \ldots, \boldsymbol{x}_{t-4}$. First, the entire set is shifted such that $\boldsymbol{x}_t$ moves to $(0, 0)$. Next, if we consider the angle between the line connecting $\boldsymbol{x}_t$, $\boldsymbol{x}_{t-1}$ and the x-axis as θ, the rotation matrix $\boldsymbol{A} \in \mathbb{R}^2$ that rotates $\boldsymbol{x}_{n-1}$ across the y-axis can be defined as

$$\boldsymbol{A} = \begin{pmatrix} \cos(270^\circ - \theta) & -\sin(270^\circ - \theta) \\ \sin(270^\circ - \theta) & \cos(270^\circ - \theta) \end{pmatrix}$$

Therefore, the normalization equation is expressed as follows:

$$(\boldsymbol{x}'_t, \ldots, \boldsymbol{x}'_{t-4}) = \boldsymbol{A} \cdot (\boldsymbol{x}_t - \boldsymbol{x}_t, \ldots, \boldsymbol{x}_{t-4} - \boldsymbol{x}_t) \qquad (1)$$

2.2. Models

Three models were compared in the present study: VAR, LSTM, and MLP.

VAR is a model that assigns weights to several previous points and uses their sum to generate predictions [7]. The VAR model is represented by the following equation:

$$\boldsymbol{x}_t = \boldsymbol{\Phi}_{t-1}\boldsymbol{x}_{t-1} + \cdots + \boldsymbol{\Phi}_{t-l-1}\boldsymbol{x}_{t-l-1} + \boldsymbol{\epsilon}, \qquad (2)$$

where $\boldsymbol{x}_t \in \mathbb{R}^n$ denotes the input at time t, $\boldsymbol{\Phi} \in \mathbb{R}^n$ denotes the weight of the input, and $\boldsymbol{\epsilon} \in \mathbb{R}^n$ denotes the bias. Typically, VAR is optimized using methods such as maximum likelihood estimation, ridge regression, or LASSO. However, in this study, a stochastic gradient descent was applied to integrate the optimization method with the one used for deep learning models.

MLP is a structure comprising multiple interconnected perceptrons as shown in Figure. 2. Each perceptron is a linear transformation represented as

$$\boldsymbol{y} = \phi \cdot (\sum_j \boldsymbol{w}_j \boldsymbol{x}_j + \boldsymbol{b}), \qquad (3)$$

where $\boldsymbol{x}_j$ is the input to the unit, $\boldsymbol{w}_j \in \mathbb{R}^n$ is the corresponding weight, $\boldsymbol{b} \in \mathbb{R}^n$ denotes the bias, and ϕ denotes the activation function. In this study, the number of units per layer was set to 64 and a Rectified Linear Unit (ReLU) was used as the activation function [8]. The ReLU function returns zero for negative inputs. For non-negative inputs, it returns a value obtained by the following equation:

$$\mathrm{ReLU(x)} = \begin{cases} x & (x \geqq 0) \\ 0 & (x < 0) \end{cases}. \qquad (4)$$

LSTM is a deep-learning model capable of learning long-term dependencies, particularly in sequence prediction problems [9]. Recurrent neural network (RNNs) have been traditionally used to analyze sequence data, wherein input features are preserved and passed to each subsequent step. However, as the amount of input data increases, an RNN becomes increasingly vulnerable to the problem of vanishing or exploding gradients. LSTM was proposed to solve this problem. Figure 3 illustrates the structure of LSTM. Here, $\boldsymbol{h}_t$ is the hidden state at time t or initial hidden state at time 0, $\boldsymbol{c}_t$ is the cell state at t, and $\boldsymbol{x}_t$ is the input at t. $\boldsymbol{i}_t$, $\boldsymbol{f}_t$, $\boldsymbol{g}_t$, and $\boldsymbol{o}_t$ are the input, forget, cellular, and output gates, respectively. σ is a sigmoid function and tanh is the hyperbolic tangent. Both functions are activation functions. Each element computes the following function:

$$\begin{aligned}
\boldsymbol{i}_t &= \sigma(\boldsymbol{W}_{ii}\boldsymbol{x}_t + \boldsymbol{b}_{ii} + \boldsymbol{W}_{hi}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{hi}), && (5)\\
\boldsymbol{f}_t &= \sigma(\boldsymbol{W}_{if}\boldsymbol{x}_t + \boldsymbol{b}_{if} + \boldsymbol{W}_{hf}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{hf}), && (6)\\
\boldsymbol{g}_t &= \tanh(\boldsymbol{W}_{\mathrm{ig}}\boldsymbol{x}_{\mathrm{t}} + \boldsymbol{b}_{\mathrm{ig}} + \boldsymbol{W}_{\mathrm{hg}}\boldsymbol{h}_{\mathrm{t-1}} + \boldsymbol{b}_{\mathrm{hg}}), && (7)\\
\boldsymbol{o}_t &= \sigma(\boldsymbol{W}_{io}\boldsymbol{x}_t + \boldsymbol{b}_{io} + \boldsymbol{W}_{ho}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{ho}), && (8)\\
\boldsymbol{c}_t &= \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \boldsymbol{g}_t, && (9)\\
\boldsymbol{h}_t &= \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_{\mathrm{t}}). && (10)
\end{aligned}$$

Based on preliminary experimental results, the number of hidden layers was set to 200.

3. Experimental setting

3.1. Dataset

We obtained data representing the movement paths of Madagascar hissing cockroaches by using VICON's motion capture camera systems. We let them walk for several seconds on a disc having a radius of 600 mm. The data represented the horizontal positions of insects, with the disc centered at $(0, 0)$. All coordinate units were in millimeters (mm). To facilitate the model learning process, we normalized the entire dataset by dividing each point by 600. A frequency of the sensor was 100Hz, resulting in only a slight difference between consecutive frames. We, therefore, discarded nine out of every ten steps. As the beginning of each time-series was affected by human's hands, we also discarded the first 10 steps. Each movement path from the center represented a single time series, with a length of approximately 80 to 300 steps. Among the 112 time series obtained, 88 were used for training and 24 were used for testing. From the beginning of each time series, we extracted consecutive sets of five points as inputs, and the following steps were designated as targets. By shifting through the entire time series, we obtained 5874 pairs of inputs and targets for training and 1337 pairs for testing.

3.2. K-Fold cross-validation

K-fold cross-validation is an evaluation method that uses an entire dataset as validation data [10]. The training dataset was segmented into k equal parts. In each training procedure, one part was used for validation, and the rest were used for training. The procedure was repeated k times to ensure that each part is used for validation once. A major advantage of cross-validation is that it prevents the model from overfitting the validation data. Because the data are randomly allocated training and validation data, it may occur that the training data contains only sets which are hard

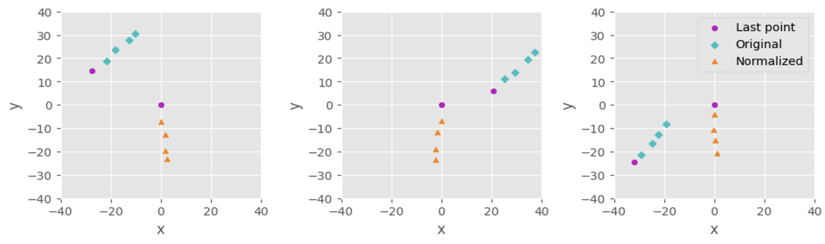

Figure 1: Examples of rotational-normalization

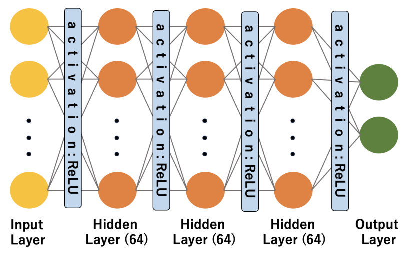

Figure 2: Structure of MLP. Each layer has 64 neurons. ReLU is used for activations.

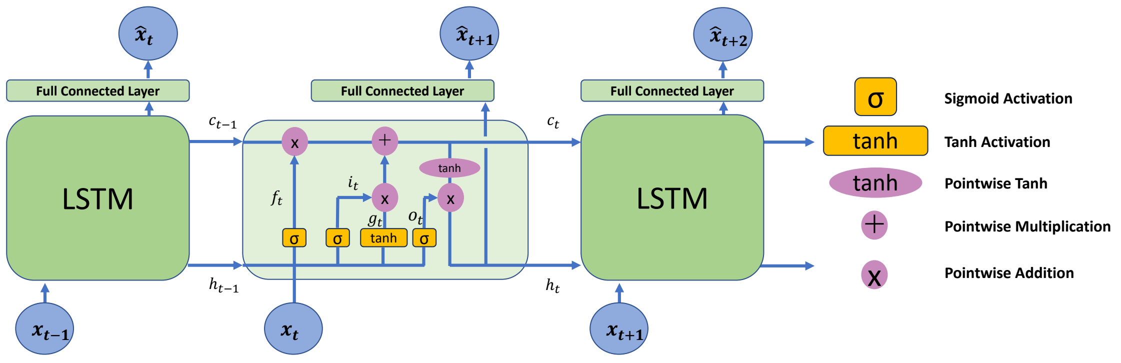

Figure 3: The structure of LSTM

for model to predict and the validation data contains only sets which are easy for model to predict.

In this study, we utilized 4-fold cross-validation, which divided 5874 training data among four equal parts. We then evaluated the four models by average score, with weights obtained for each fold of the test data.

3.3. Absolute Distance Error

The model predicts a single subsequent step using a set of five consecutive points as inputs. We evaluated each prediction using the absolute distance error, where $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ denote the i-th observed and predicted values, respectively:

$$\text{Error} = |\boldsymbol{y}_{\text{i}} - \hat{\boldsymbol{y}}_{\text{i}}|. \tag{11}$$

3.4. Cosine Annealing

Model training can be formulated as the problem of minimizing the function $f : \mathbb{R}^n \to \mathbb{R}$, where n represents the number of free-model parameters. We denote the parameter vector at time step t as $\boldsymbol{p}_t \in \mathbb{R}^n$ and the learning rate as η. The following equation was employed for the optimization procedure:

$$\boldsymbol{p}_t = \boldsymbol{p}_{t-1} - \eta \nabla f_t(\boldsymbol{p}_t), \tag{12}$$

where ∇f_t denotes the gradient information obtained in the t-th iteration.

A fixed learning rate may cause optimization problems. An excessively high learning rate ensures that the model parameters are learned quickly, but may also cause these parameters to oscillate around or even jump over the minima. In contrast, given an excessively low learning rate, the optimizer may take too long to converge or become trapped in local minima. These problems can be avoided by adjusting the learning rate throughout the training epochs. Cosine annealing is a technique that gradually adjusts the learning rate along the typical shape of a cosine function for each epoch [11]. The learning rate is initially set to a maximum value, and gradually decreases to a minimum value over a certain number of iterations. When it reaches its minimum value, it is reset to its maximum value. This procedure is repeated several times. Using cosine annealing, the learning rate is represented by the following equation:

$$\eta_t = \eta_{\text{min}} + \frac{1}{2}(\eta_{\text{max}} - \eta_{\text{min}})(1 + \cos(\frac{\text{T}_{\text{cur}}}{\text{T}}\pi)), \tag{13}$$

where η^i_{min} and η^i_{max} are the minimum and maximum learning rates, respectively, T_{cur} denotes the number of epochs in the final reset, and T_i denotes the number of epochs required to reset.

In this study, the learning rate was decreased from 1×10^{-4} to 1×10^{-2} without reset. However, for the VAR

Table 1: Results

Methods	Norm.	Error (mm)
baseline		8.88
VAR		1.75±0.02
VAR	✓	1.65±0.03
LSTM		1.76±0.03
LSTM	✓	1.55±0.03
MLP		2.61±0.03
MLP	✓	1.38±0.08

model, the learning rate was set to a range of 1×10^{-1} to 1×10^{-3} owing to the learning speed. In addition, we used the Adam optimizer to optimize the model parameters [12].

4. Results and Discussion

Models were evaluated against the test dataset using the four weights obtained during cross-validation. Table 1 lists the averages and standard deviations of the four model's scores, with the absolute distance error used as a metric. Each model's score is the average absolute distance error of all its predictions. We conducted experiments using three different models and evaluated each model using two approaches: with or without rotational-normalization. The baseline is the score between the observed values in the prediction and previous steps. Compared to the baseline, all methods exhibited improvements in accuracy. Without rotational-normalization, the VAR model demonstrated the best performance. Because VAR is a linear transformation of the previous input, it effectively captures the symmetries and patterns of data allowing for predictions, even when the directions and initial coordinates of the input exhibit variance. With rotational-normalization, the scores improved for all methods. Because the VAR model is inherently capable of learning symmetry, the impact of normalization was not particularly significant. In contrast, the LSTM and MLP models are nonlinear in nature, and their accuracy significantly improved owing to rotatinal-normalization, which increased the presence of similar data. These results indicate that rotational-normalization is effective in improving the prediction accuracy of all models considered in this study.

5. Conclusion

In this study, we verified multiple mathematical models for the prediction of insect paths. The experimental results indicate that all models achieved higher scores than the baseline. Moreover, rotational-normalization enabled the models to generate more precise predictions.

Acknowledgments

This study was supported by JST PRESTO (JPMJPR21C7). JST [Moonshot R&D][Grant Number JPMJMS223A], and JSPS KAKENHI (21H03515)

References

[1] Teuvo Kohonen. *Self-organizing maps*. Springer, 2001.

[2] Ofer Feinerman, Itai Pinkoviezky, Aviram Gelblum, Ehud Fonio, and Nir Gov. The physics of cooperative transport in groups of ants. *Nature Physics*, 2018.

[3] Eric Zivot and Jiahui Wang. *Vector Autoregressive Models for Multivariate Time Series*. Springer New York, 2003.

[4] Felix A. Gers, Douglas Eck, and Jürgen Schmidhuber. Applying lstm to time series predictable through time-window approaches. In Roberto Tagliaferri and Maria Marinaro, editors, *Neural Nets WIRN Vietri-01*. Springer London, 2002.

[5] Md. Shiblee, P. K. Kalra, and B. Chandra. Time series prediction with multilayer perceptron (mlp): A new generalized error based approach. In *Advances in Neuro-Information Processing*. Springer Berlin Heidelberg, 2009.

[6] Carl Rasmussen and Zoubin Ghahramani. Occam's razor. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*. MIT Press, 2000.

[7] Kevin P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[9] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997.

[10] Ron Kohavi. A study of cross-validation and Bootstrap for accuracy estimation and model selection. Morgan Kaufmann, 1995.

[11] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017.

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.