Title Classification of Swarm Behaviors

Student Name Deyi Zeng

Student No. 2019051712

Major Information Computing Science

Supervisor Xiaohong Zhu

Date 22/03/2023

暨 南 大 学

本科生毕业论文

论文题目　生物群体行为分类

学　　院　暨南大学伯明翰大学联合学院

学　　系

专　　业　信息与计算科学

姓　　名　曾德懿

学　　号　2019051712

指导教师　朱小红

2023 年 3 月 22 日

Statement of Originality

I hereby declare that the thesis presented is the result of research performed by me personally, under guidance from my supervisor. This thesis does not contain any content (other than those cited with references) that has been previously published or written by others, nor does it contain any material previously presented to other educational institutions for degree or certificate purpose to the best of my knowledge. I promise that all facts presented in this thesis are true and creditable.

Signed: _____ Date: _____

Classification of Swarm Behaviors

Abstract: Swarm behavior refers to the creatures in the same species moving and behaving as a swarm. Studying swarm behaviors help biologists to analyze the mechanism of animals, and enlighten managers to manage sophisticated systems. Generally, a swarm has three behaviors: flocking, aligning, and grouping. Individuals in the swarm have an influence on each other, and each individual is subject to alignment, separation, and cohesion forces. According to the Swarm Behavior Dataset in UCI Repository, we perform classification tasks of three swarm behaviors based on the coordinates, velocities, alignment, separation, and cohesion forces of 200 individuals in the swarm. The experiments indicate that the accuracy of classification based on unprocessed features is poor, with 82.31%, 69.28%, and 99.25% accuracy of classification based on the coordinate and velocity. Therefore, two feature processing methods are proposed: the Centroid method that takes the mean values of 200 boids of a swarm as features, and the 2D mapping method that maps the 200 boids of the swarm onto the 2D plane according to their coordinates. Incorporating the centroid method with K-Nearest Neighbors, Random Forest, Logistic Regression, Multi-layer Perceptron, and 2D mapping method with Convolutional neural network, completing classification task based on coordinate and velocity, and task based on three forces. The combination of 2D mapping method and Convolutional neural network has the highest performance in the classification task based on coordinate and velocity, with the accuracy improved to 99.84%, 96.24%, and 100.00% respectively. Finally, we analyze the permutation importance of variables in the classification task based on coordinate and velocity and conclude that the partial velocity on the y-axis has the highest importance in flocking and aligning behaviors, and the coordinate has the highest importance in grouping behaviors. And analyze the Gini importance in the classification task based on three forces, conclude that the alignment force is the most important in flocking and aligning behaviors and the cohesion force is the most important in grouping behaviors.

Key Words: Swarm Behaviors, Classification, Convolutional Neural Network, Permutation Importance, Gini Importance

生物群体行为分类

摘要：群体行为是指多个同种生物以群体为单位进行某种活动。研究群行为可以帮助生物学家分析生物行为的机制，可以为人类管理复杂的系统提供启发。通常，群体有三种行为：群聚、对齐和分组。群体中的每个个体之间又会相互影响，受到对齐力，分离力，内聚力的影响。本文以 UCI 数据库中的 Swarm Behavior 数据集为研究对象，根据生物群体内 200 个个体的坐标，速度，以及受到的对齐力，分离力和内聚力等特征对生物群体的三种行为进行分类。经实验发现直接基于未处理的特征进行分类效果较差，在基于坐标和速度的分类任务中对群聚，分组和对齐群体行为的分类正确率分别为 82.31%，69.28%，99.25%。因此我们提出了两种特征处理方法：质心法和 2D 映射法。质心法将一个群体中的 200 个个体取平均处理，2D 映射法将 200 个个体分别映射到平面对应位置的像素点上。通过将质心法分别与 K-邻近算法，随机森林，逻辑回归，多层感知器结合，以及将 2D 映射法与卷积神经网络结合，一共采用五种方法完成了基于坐标和速度的群体行为分类以及基于三个力的群体行为分类。我们发现 2D 映射法与卷积神经网络的组合在基于坐标和速度的分类任务中具有最高的性能，对群聚，分组和对齐的预测准确率提升至 99.84%，96.24%，100.00%。最后我们分别分析了基于坐标和速度的分类任务中的变量置换重要性和基于三个力的分类任务中的变量基尼重要性。基于坐标和速度的分类任务中，y 轴分速度在群聚与对齐行为中占有最高的的重要性，坐标在分组行为中占有最高的重要性。基于三种力的分类任务中，对齐力在群聚与对齐行为中占有最高的重要性，内聚力在分组行为中占有最高的重要性。

关键字：生物群体行为，分类，卷积神经网络，置换重要性，基尼重要性

Contents

1 Introduction

1.1 Background

A swarm of creatures exhibits some swarm behaviors in nature. For example, birds fly in the same direction and change their direction without colliding with each other. The schooling fish could swim in a meandering path in order. Ants seem to have a predefined rule such that they could forage and transport the food efficiently as a group. Wild geese migrate from north to south as a group, while seldom choose to fly to the east or west as a group during the migration seasons. Craig Reynolds proposed three inner forces which influence the swarm behaviors of these flocks: alignment force, cohesion force, and separation forces. According to the view of Reynolds (Craig, 1987), alignment force implies that the individual tends to match the direction of their neighboring peers, while cohesion force means that the individual flies toward the center of their neighboring peers. Conversely, the separation mechanism force prevents the individual from getting too close to each other. The individual steers in the converse direction of its neighboring peers as well under the influence of separation force.

In nature, there are three prevalent swarm behaviors: Flocking, Aligning, and Grouping. Individuals in the aligning swarm move in the same direction, and individuals in the grouping swarm cluster with each other, forming several clusters. Individuals in the flocking swarm are characterized by tending to move in the same direction, cluster with each other, or another simultaneous movement. These three behaviors are inclusive, which means that a swarm of creatures may exhibit the flocking, aligning, and grouping behaviors simultaneously.

Although it is easy to make a computer simulation of a swarm based on the three forces and also easy for human beings to recognize the swarm behaviors, it is hard for computers to recognize the behaviors like human beings. Therefore, we want to help computers to recognize swarm behaviors. Also, we want to study the influence and significance of three forces: alignment, cohesion, and separation on swarm behaviors.

1.2 Related literature

In 1952, the flocking behavior of birds was purposed by JOHN T. EMLEN, JR, with the suggestion of the existence of centripetal forces corresponding to social attraction and repulsion (Emlen & John T, 1952). In 1987, Craig W. Reynolds propose the first swarm behavioral model

depicting the motion of schooling fish, flocking birds, and wandering herds (Craig, 1987). In 2000, it was mentioned that the stability of artificial robots in swarms is an important metric of a system. The study of swam behaviors contributes to providing new ideas for optimization algorithms and designing new traffic patterns in transportation systems (Liu & Kevin, 2000). In 2006, Christopher Hartman and Bedrich Benes observe that some bird flocking does not obey the conventional framework proposed by Reynold. To tackle this problem, they introduce the leadership mechanism that the boid in the front line tends to escape from the flocking (Christopher & Bedrich, 2006). In 2011, the boids algorithm is extended to be multi-group, focusing on the interaction behaviors between different species (Alwyn & Ken, 2011). In 2014, the boids algorithm is furtherly optimized by the genetic algorithm and particle swarm algorithm (Saleh *et al.*, 2014). In 2018, a survey about the extension of 2-D ground swarm robotics to 3-D aerial one was published (Soon-Jo *et al.*, 2018). In 2019, swarm intelligence was applied to the Particle Swarm Algorithm, for avoiding collisions between particles (Lauren *et al.*, 2019).

In 1967, the K-Nearest Neighbor algorithm was proposed. This was a non-parametric classification algorithm that predicts the sample based on the nearest neighbor (Thomas & Peter, 1967). In 1986, decision tree ID3 was suggested, using the bisection principle to separate the variable space into several square-like subspaces (J. Ross, 1986). In 2014, the improvement of previous work C4.5 was introduced (J. Ross, 2014). A new sort of decision tree using Gini impurity as a criterion was proposed, and it could conduct both regression and classification tasks (Roger, 2000). In 1995, the supported vector machine was proposed, to use a hyperplane and kernel function to separate the samples (Corinna & Vladimir, 1995). In 2001, an ensemble algorithm ----- the random forest algorithm, which was robust to noise (Leo, 2001). When studying the growth of the population, the logistic function was raised (Acot *et al.*, 2021). The logistic regression was therefore applied to binary classification problems. The logistic regression is generalized to multi-categorical classification models (Wright, 1995). In 1958, a perceptron model that could approximate the function was proposed (Rosenblatt, 1958). To improve the capability of approximating function, the multilayer perceptron, also known as an artificial neural network, was introduced (Anil *et al.*, 1996). But the previous models and algorithms mentioned could not consider the positional information. Under this circumstance,

the convolutional neural network was introduced which was sensitive to the positional information and could process 3-dimensional variables (Keiron & Ryan, 2015) (Saad *et al.*, 2017).

1.3 Problem statement

The team at New South Wales University simulates the motion of a certain creature by setting different alignment, cohesion, and separation forces on the boids. Under different settings of alignment, cohesion, and separation forces, swarms of boids may exhibit aligning, flocking, and grouping behaviors, or move unorderly. The positional coordinate, velocity, and three forces of 200 boids participating in the simulation are recorded. With three given datasets, we need to complete two tasks as follows:

Behaviors Recognition: Classification Based on Positional Coordinate and Velocity.

Given only the Positional Coordinate and Velocity of 200 boids, we need to predict the swarm behaviors of this swarm. That is, we need to judge whether this swarm is flocking, grouping, aligning, two of them, all of them, or none of them. Only x, y, $xVel$ and $yVel$ can be used. This technique could be applied to the real case, to identify the swarm behavior based on the positional coordinate and velocity indicator attached to some individuals of the swarm which assists biologists to identify the swarm behaviors of other real-life-creatures. For example, biologists tied the GPS to birds to study the migration behaviors of birds.

Forces Importance Analysis: Classification Based on Three Forces.

Since the boids in the simulation are controlled by three forces: Cohesion, Alignment, and Separation Forces, we want to study the importance of three forces on swarm behavior. That is, which force contributes the most to the flocking, aligning, and grouping behaviors respectively. In this task, only $xA,\ yA, xS, yS, xC, yC, nAC, nS$ could be used. This technique could be applied to study the significance of three forces on the occurrence of swarm behaviors. Then the biologists and computing scientists could allocate more resources and effort to studying and tuning the corresponding parameters of forces in both the simulation and in reality.

2 Dataset

2.1 Overview

The swarm behaviors dataset comes from the UCI machine learning repository (Dheeru & Casey). There are three datasets: Flocking, Aligned, and Grouped. The basic structure of each dataset is the same. Each dataset records whether a sample is flocking or non-flocking, aligning or non-aligning, and grouping or non-grouping.

Each dataset is a 24016×2401 matrix stored in .csv format as in Figure 2-1. The row represents the index of a sample and column represents the features, except the last column stands for the label of the sample. The distribution of the labels is given in Figure 2-2. Each sample is a capture of the instantaneous state of 200 individuals under a specific system. Each individual, or said boid, is characterized with 12 features, including the positional coordinate on x and y axis (x_i, y_i), the partial velocity with respect to x and y axis$(xVel_i, yVel_i)$, the inner force by alignment, cohesion and separation principles with respect to x and y axis(xA_i, yA_i, xS_i, yS_i, xC_i, yC_i), and the number of affected peers of each boid in the radius of alignment, cohesion force (nAC_i) and separation force (nS_i).

	x1	y1	xVel1	yVel1	xA1	yA1	xS1	yS1	xC1	yC1	...	yVel200	xA200	yA200	xS200	yS200	xC200	yC200	nAC200	nS200	Class
0	-1414.14	-535.22	-17.88	-7.23	0.00	0.00	0.00	0.00	0.00	0.00	...	-16.85	0.0	0.00	0.00	0.00	0.00	0.00	29	0	0
1	-1412.93	597.54	-13.55	-5.48	0.00	0.00	0.00	0.00	0.00	0.00	...	-12.09	0.0	0.00	0.00	0.00	0.00	0.00	44	0	0
2	-1407.38	70.72	-14.37	-5.81	0.00	0.00	0.00	0.00	0.00	0.00	...	-16.20	0.0	0.00	0.00	0.00	0.00	0.00	40	0	0
3	-1407.00	-759.80	-7.59	-1.27	-0.98	-0.20	0.00	0.00	0.91	0.41	...	2.99	-1.0	-0.07	0.00	0.00	-0.52	0.86	3	0	1
4	-1406.36	698.39	-16.54	-6.95	-1.00	0.00	-944.07	-396.62	0.00	0.00	...	-12.61	0.0	-1.00	0.00	0.00	0.00	0.00	13	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
24011	1403.71	948.55	4.54	-6.29	0.00	0.00	0.00	0.00	-0.13	-0.29	...	-4.87	0.0	0.00	0.00	0.00	-0.26	-0.18	11	0	0
24012	1403.72	133.09	9.46	14.33	0.00	1.00	0.00	0.00	0.00	0.00	...	5.20	0.0	1.00	-0.10	-3.24	0.00	0.00	29	0	0
24013	1404.38	144.31	6.98	3.89	0.00	0.00	0.00	0.00	0.00	0.00	...	9.50	0.0	0.00	-0.85	-0.52	0.00	0.00	5	1	0
24014	1404.61	-315.55	6.50	4.27	0.00	0.00	0.00	0.00	0.00	0.00	...	-0.75	0.0	0.00	0.00	0.00	0.00	0.00	1	0	0
24015	1406.08	-354.52	10.00	1.18	0.00	-0.01	0.00	0.00	0.31	0.07	...	-1.13	0.0	0.00	0.00	0.00	-0.31	-0.08	0	0	0

24016 rows × 2401 columns

Figure 2-1 Display of Samples

The class distribution does not suffer from severe unbalance, so we do not consider using some balancing techniques in this problem. But since the last sample has a missing value, we need to drop this sample. For each dataset, the remaining 16000 samples are distributed to the training dataset and the remaining 8015 samples are distributed to the test dataset.

The meaning and notation of variables are displayed in Table 2-1.

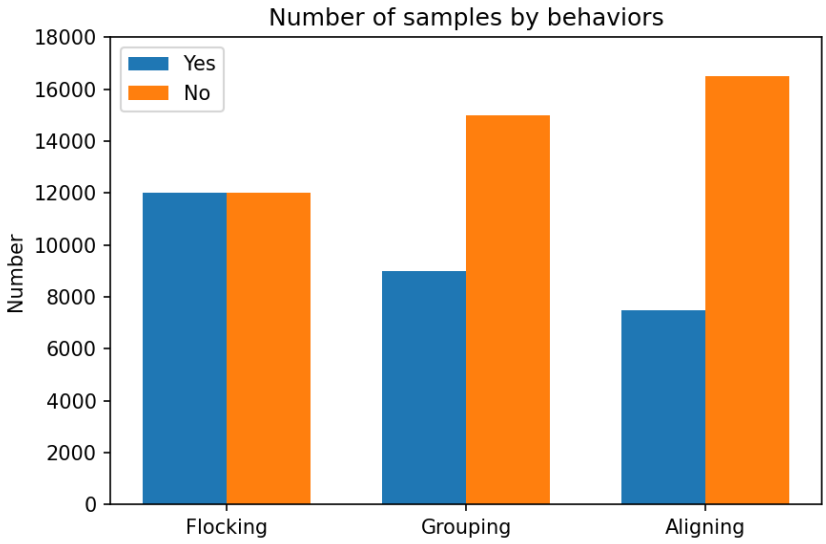


Figure 2-2 Number of Samples by Behaviors

Table 2-1 Description of Raw Features

Features	Symbols	Min	Max
Position on x axis	x_i	-1418.25	1417.68
Position on y axis	y_i	-1018.92	1018.16
Partial Velocity on x axis	$xVel_i$	-19.95	19.83
Partial Velocity on y axis	$yVel_i$	-19.99	19.78
Partial Alignment Force on x axis	xA_i	-1.07	1.1
Partial Alignment Force on y axis	yA_i	-1.12	1.1
Partial Separation Force on x axis	xS_i	-1037197.43	3692911.2
Partial Separation Force on y axis	yS_i	-1048459.42	4191195.27
Partial Cohesion Force on x axis	xC_i	-2.68	2.68
Partial Cohesion Force on y axis	yC_i	-2.68	2.68
Number of Boids in the radius of Alignment and Cohesion Force	nAC_i	0	171
Number of Boids in the radius of Separation Force	nS_i	0	108

2.2 Potential risk embedded in original features

We could not apply the classification techniques directly to the samples, as the samples are unordered: For example, suppose that we have three boids on the plane which behave as a flocking swarm. The positional coordinate of the first boid is (1,9), and positional coordinate of the second boid is (2,5), and the positional coordinate of the third boid is (4,6). Then the feature variable is written as [1, 9, 2, 5, 4, 6]. But if the positional coordinate of the first boid is (4,6), the positional coordinate of the second boid is (1,9), and the positional coordinate of the third boid is (2,5). Then the feature variable is written as [4, 6, 1, 9, 2, 5]. But indeed, they should all behave as radically the same flocking swarms, although having different feature variables.

If we apply the classifiers directly onto the original normalized data, with the indices of 200 boids reversed, the results are unsatisfying:

Table 2-2 Metrics of Classification based on Position and Velocity (Unprocessed Features)

Swarm Behaviors	Classifier	Precision	Recall	F1-score	Accuracy
Flocking	K-Nearest Neighbors	0.9719	0.5158	0.6740	0.5160
	Random Forest	**0.9966**	0.5901	0.7413	0.6419
	Logistic Regression	0.7716	0.8481	0.8081	0.8114
	Multi-layer Perceptron	0.7535	**0.8857**	**0.8143**	**0.8231**
Grouping	K-Nearest Neighbors	0.338	0.2474	0.2856	0.2243
	Random Forest	0.3883	0.5662	0.4606	0.5828
	Logistic Regression	**0.394**	0.5939	0.4737	0.5983
	Multi-layer Perceptron	0.3529	**0.9406**	**0.5132**	**0.6928**
Aligning	K-Nearest Neighbors	0.9607	0.4020	0.5668	0.4598
	Random Forest	**0.9797**	**1.0000**	**0.9897**	**0.9925**
	Logistic Regression	0.8701	0.9764	0.9202	0.9445
	Multi-layer Perceptron	0.7877	0.9906	0.8776	0.9192

According to Table 2-2, the accuracy of recognizing the flocking, grouping, and aligning behaviors based on coordinate and velocity are at most 82.31%, 69.28%, and 99.25% respectively, based on the unprocessed features. Especially, the accuracy and F1-score of classifiers in flocking behaviors and grouping behaviors are unsatisfying. This result indicates that some special feature processing techniques should be applied to eliminate the order in features.

Table 2-3 Metrics of Classification based on Three Forces (Unprocessed Features)

Swarm Behaviors	Classifier	Precision	Recall	F1-score	Accuracy
Flocking	K-Nearest Neighbors	0.7947	0.7510	0.7722	0.7587
	Random Forest	**0.8832**	0.9819	0.9299	**0.9315**
	Logistic Regression	0.6507	0.9541	0.7737	0.8041
	Multi-layer Perceptron	0.6955	**0.9832**	**0.8147**	0.8372
Grouping	K-Nearest Neighbors	0.3230	0.9347	0.4801	0.6790
	Random Forest	**0.9810**	**0.9232**	**0.9512**	**0.9538**
	Logistic Regression	0.1504	0.8316	0.2547	0.5961
	Multi-layer Perceptron	0.0595	0.9125	0.1118	0.5658
Aligning	K-Nearest Neighbors	**1.0000**	0.8814	0.9369	0.4598
	Random Forest	0.9522	**0.9901**	**0.9708**	**0.9789**
	Logistic Regression	0.9722	0.9440	0.9579	0.9686
	Multi-layer Perceptron	0.8396	0.9875	0.9068	0.9365

According to Table 2-3, the accuracy of recognizing the flocking, grouping, and aligning behaviors based on three forces at most 93.15%, 95.38%, and 97.89% respectively, based on the unprocessed features.

In the next two subsections, we propose to eliminate the order in the features by using two different feature processing methods independently. The Centroid method treats the swarm as a centroid, by taking the mean value of all features with respect to 200 boids, and the 2D mapping method maps each boid to a pixel of the plane. Both methods could transform the ordered features into unordered features.

2.3 Feature processing method 1: centroid method

The centroid method is a feature fusion method, we may assume that there is a centroid of each swarm. The centroids may exhibit special behavior as the swarms. For example, in the migration behavior of birds, we can clearly see that the swarm of birds exhibits the northern velocity. Therefore, the centroid of the birds also has a velocity directing to the north. But for a swarm of rabbits wandering aimlessly on the grasslands, the centroid of birds exhibits no clear direction and velocity. In a nutshell, the centroid of a swarm sometimes exhibits the behavior of the swarm.

Therefore, we compute the centroid of each swarm:

$$\overline{x} = \frac{1}{200}\sum_{i=1}^{200} x_i \tag{2-1}$$

$$\bar{y} = \frac{1}{200}\sum_{i=1}^{200} y_i \tag{2-2}$$

$$\overline{xVel} = \frac{1}{200}\sum_{i=1}^{200} xVel_i \tag{2-3}$$

$$\overline{yVel} = \frac{1}{200}\sum_{i=1}^{200} yVel_i \tag{2-4}$$

$$\overline{xA} = \frac{1}{200}\sum_{i=1}^{200} xA_i \tag{2-5}$$

$$\overline{yA} = \frac{1}{200}\sum_{i=1}^{200} yA_i \tag{2-6}$$

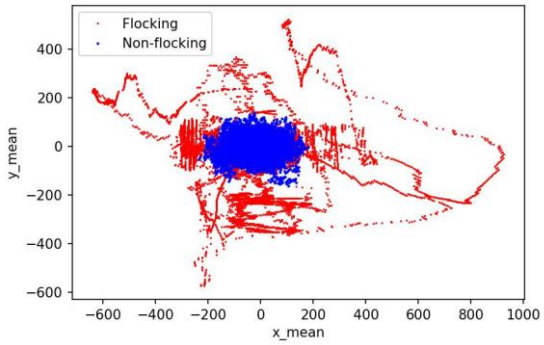
Similar for $\overline{xS}, \overline{yS}, \overline{xC}, \overline{yC}, \overline{nAC}, \overline{nS}$.

The meaning and notation of variables in centroid method are given in Table 2-3:

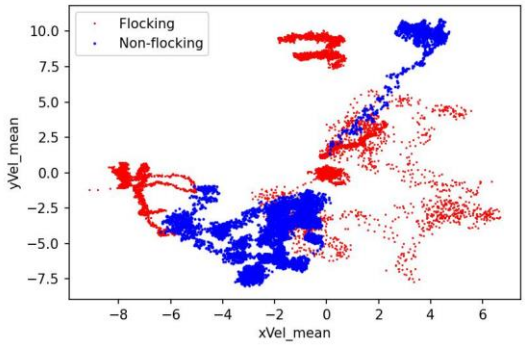
Table 2-3 Description of Features Processed by Centroid Method

Features	Symbols	Min	Max
Position on x axis of Centroid	$\bar{x}$	-639.50	925.49
Position on y axis of Centroid	$\bar{y}$	-573.79	525.03
Partial Velocity on x axis of Centroid	$\overline{xVel}$	-9.13	6.67
Partial Velocity on y axis of Centroid	$\overline{yVel}$	-8.01	10.86
Partial Alignment Force on x axis of Centroid	$\overline{xA}$	-0.98	0.90
Partial Alignment Force on y axis of Centroid	$\overline{yA}$	-0.98	1.04
Partial Separation Force on x axis of Centroid	$\overline{xS}$	-5191.43	18488.61
Partial Separation Force on y axis of Centroid	$\overline{yS}$	-5188.35	21251.03
Partial Cohesion Force on x axis of Centroid	$\overline{xC}$	-0.71	0.26
Partial Cohesion Force on y axis of Centroid	$\overline{yC}$	-0.41	0.95
Number of Boids in the radius of Alignment and Cohesion Force of Centroid	$\overline{nAC}$	0.41	121.68
Number of Boids in the radius of Separation Force of Centroid	$\overline{nS}$	0	50.07

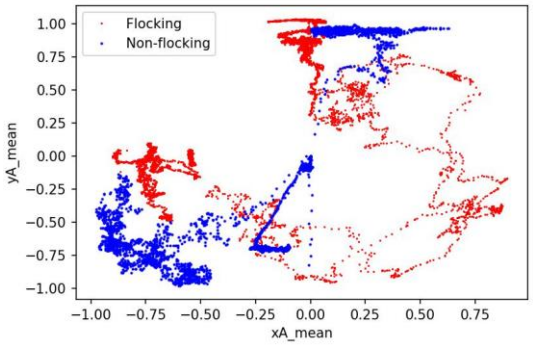
Min-max normalization is applied when training and testing classifiers.



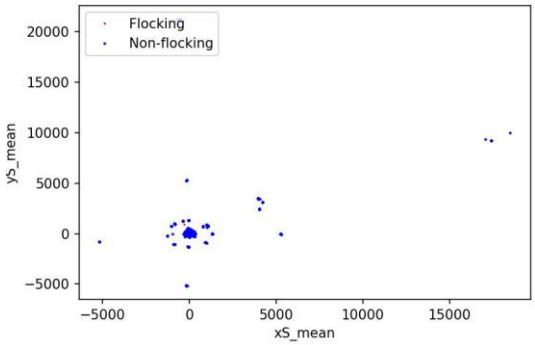
(a) Positional Coordiante of Centroids



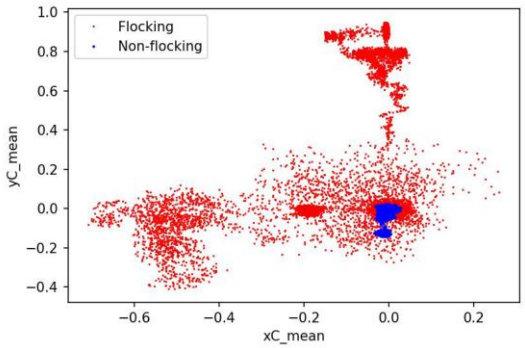
(b) Velocity of Centroids



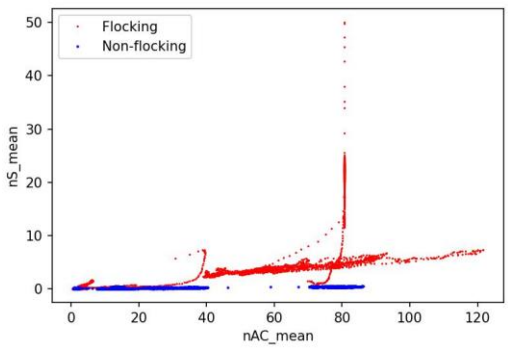
(c) Alignment Force of Centroids



(d) Separation Force of Centroids



(e) Cohesion Force of Centroids

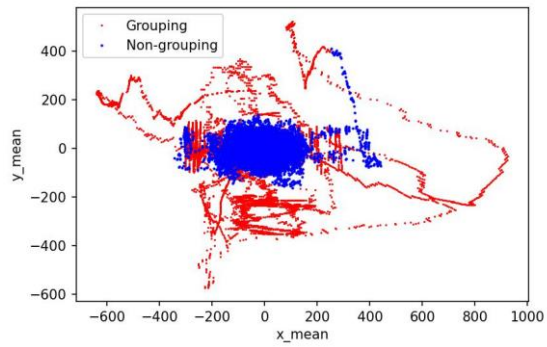


(f) Number of Boids in Radius of Centroids

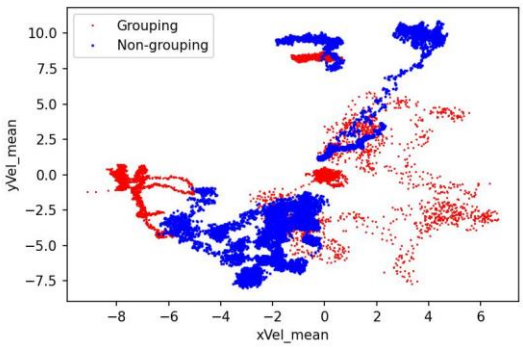
Figure 2-3 Comparison of Features between Flocking and Non-flocking Swarms

According to Figure 2-3, the centroid of non-flocking swarms is centered around (0,0), as the boids of non-flocking swarms are scattered on the map in disorder. We could also observe that the velocity and alignment force of the centroid of the non-flocking swarm is mainly directed to the northeast and southwest, while that of the flocking swarm is unclear. Both the centroids of the flocking and non-flocking swarm are not largely affected by separation force. Also, the cohesion force of centroids of non-flocking swarms is almost zero. The flocking behavior has

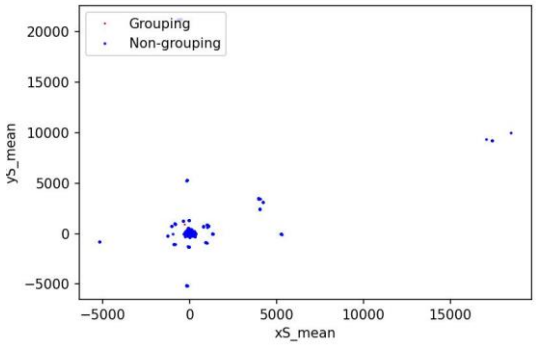
a relatively high number of affecting boids by separation force.



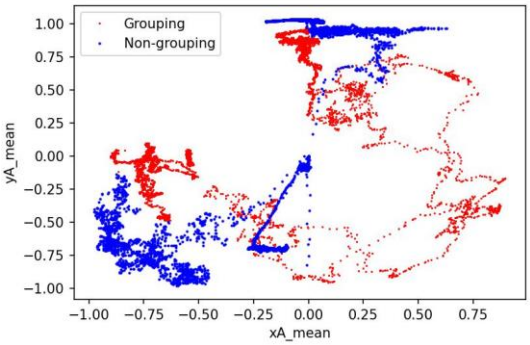
(a) Positional Coordiante of Centroids



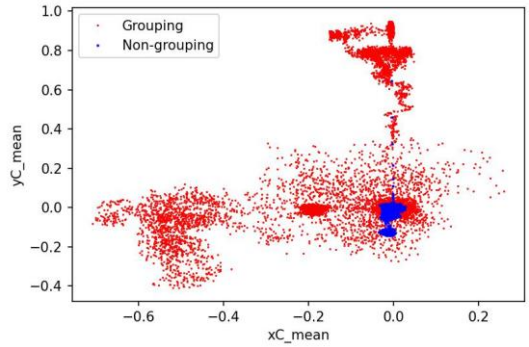
(b) Velocity of Centroids



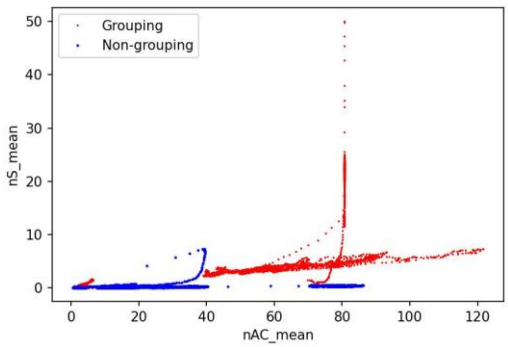
(c) Alignment Force of Centroids



(d) Separation Force of Centroids



(e) Cohesion Force of Centroids

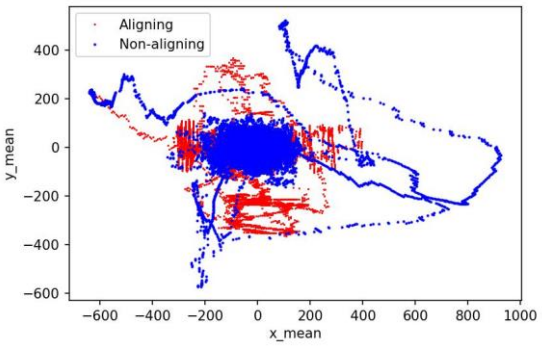


(f) Number of Boids in Radius of Centroids

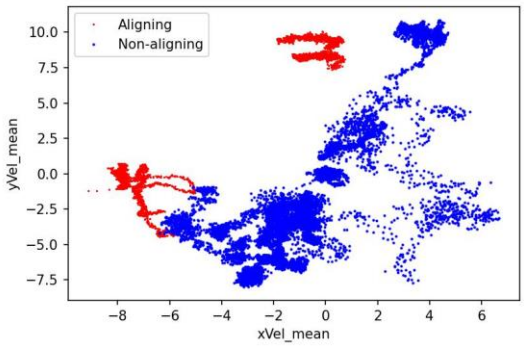
Figure 2-4 Comparison of Features between Grouping and Non-grouping Swarms

According to Figure 2-4 depicting the positional coordinate and velocity of grouping and non-grouping behaviors, we discover that there exists no obvious line or curve that could perfectly separate the grouping and non-grouping swarms. Also, the alignment forces and separation forces do not provide adequate evidence for classification. We could notice that the grouping that cohesion forces of the centroid are almost 0 when the swarm is not grouping and

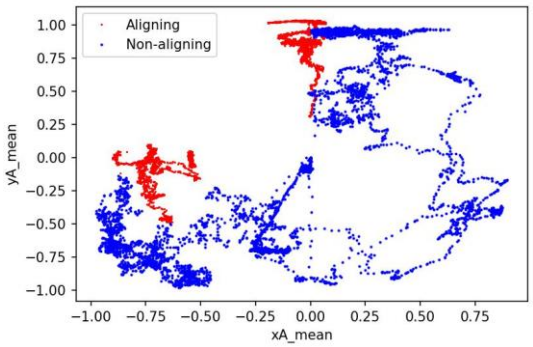
directing to other directions when the swarm is grouping. But for the number of boids in the radius of separation, alignment, and cohesion, it may be harder for the researcher to draw a line or curve such that the positive and negative swarms could be perfectly separated.



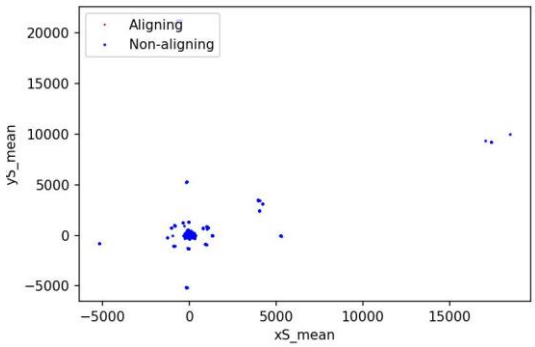
(a) Positional Coordiante of Centroids



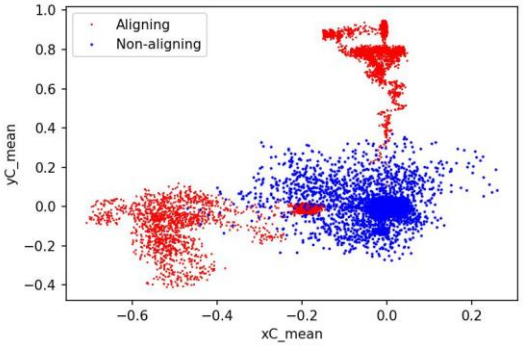
(b) Velocity of Centroids



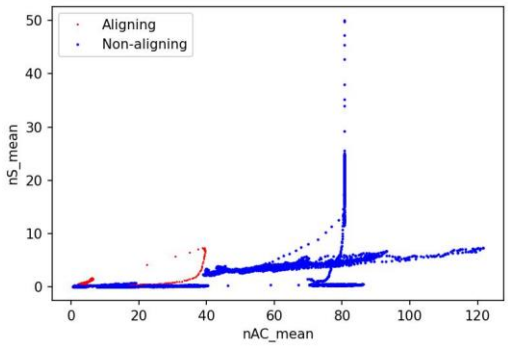
(c) Alignment Force of Centroids



(d) Separation Force of Centroids



(e) Cohesion Force of Centroids



(f) Number of Boids in Radius of Centroids

Figure 2-5 Centroids of Aligning and Non-aligning Swarms

In Figure 2-5 comparing the positional coordinate of aligning and non-aligning swarms, we discover that this feature is not that useful in prediction. But the velocity of the centroid is a perfect indicator of aligning behaviors. The aligning swarms tend to have north and west

alignment forces. We may conclude that the alignment force and cohesion force are significant features in the prediction of the aligning behaviors of this creature.

But the centroid method eliminates the information of individuals as it only concerns the mean value of the features. It relies on the tendency of a swarm for classification, which may suffer from worse performance under the circumstance that the tendency of a swarm is not adequately clear. These are the reasons why we introduce another method: 2D mapping as another method of eliminating the ordered parts of 200 boids.

2.4 Feature processing method 2: 2D-mapping method

In this section, a method for transforming the ordered tuples into unordered tuples is proposed.

Since each individual has a unique positional coordinate, this can be used to transform the unordered tuples into ordered ones. Firstly, we should tackle the positional variables: x and y. We choose one swarm from the 24015 swarms and visualize the positional coordinates of 200 boids in a swarm in Figure 2-6.

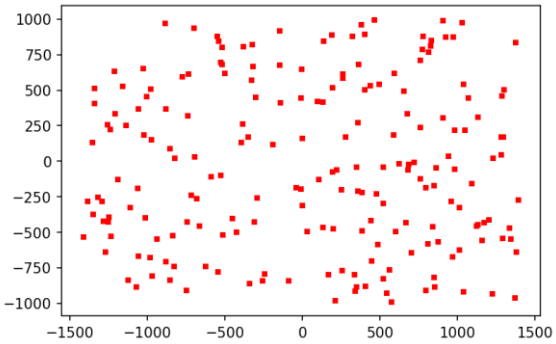


Figure 2-6 Visualization of a Swarm

Step 1: Apply the linear transformation to the x- and y- axis, such that they can be controlled to an affordably small range. For example, in default setting of this experiment, we could apply $x\_new = \frac{x + 1420}{60}$ and $y\_new = \frac{y + 1020}{60}$, then we rescale the x range from [-1500, 1500] to [0, 50], and y range from [-1000, 1000] to [0,35]. (Figure 2-7)

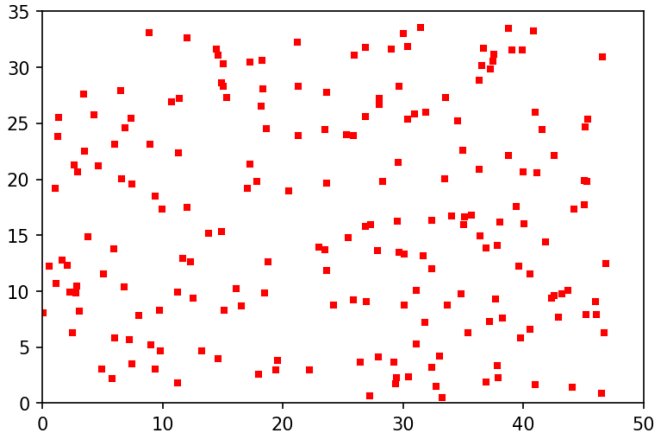


Figure 2-7 Visualization of a Swarm after Linear Transformation

Step 2: Evenly separate the region using mesh grids (Figure 2-8), and count the number or existence of boids in each region. Two statistical methods can be used here: A: Count the number of bodies in each square. B: Count whether there exist boids in each square. If there is, it should be recorded as 1, and if there is not, it will be recorded as 0. We can obtain a 50×35 matrix. Then we visualize it on the graph.

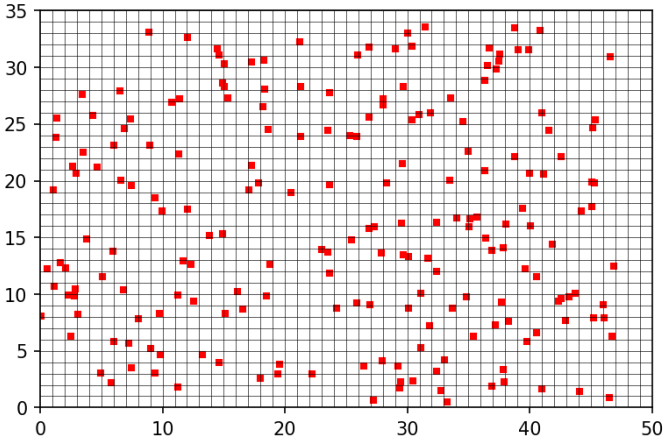


Figure 2-8 Mesh grid of swarm

Step 3: Visualize the Indicator matrix or Intension Matrix as an image, and compare it with the original graph (Figure 2-9). In the figure below, we use the indicator matrix with a resolution of 50×35, which can basically retain the fidelity of the original distribution of 200 bodies. To increase the fidelity user can choose to increase the resolution, with a higher cost of memory and running time. In the Indicator matrix, the unordered tuple is converted into the ordered matrix.

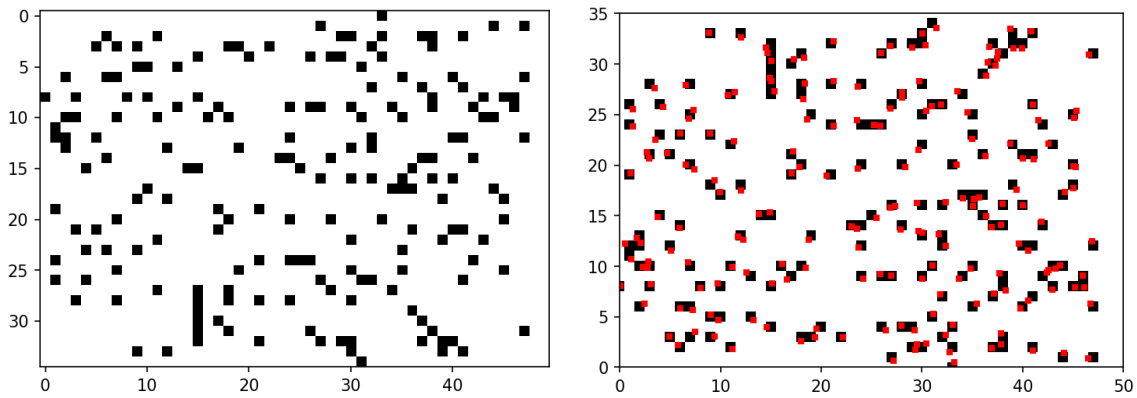


Figure 2-9 Comparison between Indictor Variable and Original Figure

Step 4: For other features, we can build the velocity field, alignment force field, cohesion force field, and separation force field based on the positional coordinate.

The features in this 2D mapping method become: $Indicator, xVel, yVel, xA, yA, xS, yS, xC, yC, nAC, nS$. The positional coordinate x, y is replaced by $Indicator$. The shape of each variable is $(50,35)$. If there is no boid at a specific pixel, then the other variables of that pixel would be 0 as well. The $Indicator, xVel, yVel$ would be used in classification task based on coordinate and velocity and the remaining variables are used in classification task based on three forces.

3 Classifiers

We introduce five binary classification models based on current available variables and labels. The K-Nearest neighbor, random forest, logistic regression and multi-layer perceptron are applied to the features processed by centroid method, and the convolutional neural network is applied to the features processed by 2D mapping method.

Suppose that the training dataset has m samples, and test dataset has n samples. And there are p features variables for each sample.

Denote the value of feature of training sample by: $x_d^{(ri)}, (d = 1,2,3, \dots, p)(i = 1,2,3, \dots, m)$, where (i) denotes the index of this sample in the training dataset, and r is the symbol of training dataset. The d is the index of feature. For example, $x_3^{(r6)}$ is the third feature of the sixth sample in training dataset. Denote $y^{(ri)}, (i = 1,2,3, \dots, m)$ to be the label of the sample i in training dataset. Similarly, test sample is denoted by: $x_d^{(ej)}, (d = 1,2,3, \dots, p)(j = 1,2,3, \dots, n)$. e represents the test dataset. Also, denote $y^{(ej)}, (j = 1,2,3, \dots, n)$ to be the label of the $j - th$ sample in training dataset.

3.1 K-nearest neighbor

This is a lazy model which has no parameter and training process. In the K-nearest neighbor classification model, the machine stores the whole training dataset. When a new sample $x_e^{(j)}, (j = 1,2,3, \dots, n)$ is input and asked for prediction, the classifier would compute the distance between all training samples and $x_e^{(j)}$. The distance is chosen to be Euclidean distance in this experiment.

The distance between two samples is:

$$d(x^{(ri)}, x^{(ej)}) = \sqrt{\sum_{d=1}^{p} (x_d^{(ri)} - x_d^{(ej)})^2} \tag{3-1}$$

After the computation of m distances. The classifier selects K samples with lowest distance $d\left(x^{(ri)}, x^{(ej)}\right)$.

Denote $N(1)$ to be the number of positive samples in the selected K samples, and $N(0)$ to be the number of negative samples in the selected K samples. Then the prediction of test sample j is:

$$y^{(ej)} = \begin{cases} 1 & , N(1) \geq N(0) \\ 0 & , N(1) < N(0) \end{cases} \tag{3-2}$$

In the experimental process, we choose different hyperparameters: $K = 1,2,3,4,5,6,7,8.$

3.2 Random forest

The CART decision tree classifier is introduced in this random forest classification model.

CART decision tree has a binary-tree-like architecture constructed from top to down. In each step, the CART classifier creates a node and uses binary split to separate the training dataset into two parts using the rule: $x_{rd} \geq t$ and $x_{rd} < t$, where d and t are selected by classifier itself to greedily minimize the weighted summation of Gini impurity of two subspaces (left node and right node).

Suppose that there is a group p containing n samples and their labels are $y^{(1)}, y^{(2)} \ldots \ldots, y^{(n)}. (y^{(j)} \in \{0,1\})$

Then the Gini impurity of this group is:

$$Gini(p) = p_0(1-p_0) + p_1(1-p_1) = 2p_1(1-p_1) = 2\frac{\sum_{j=1}^{n} y^{(j)}}{n}(1-\frac{\sum_{j=1}^{n} y^{(j)}}{n}) \tag{3-3}$$

Specially, if all of the samples in this group have the same labels, then the Gini impurity is 0, which is the best case.

For instance, there is a dataset:

$$D = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), (x^{(3)}, y^{(3)}), (x^{(4)}, y^{(4)}), (x^{(5)}, y^{(5)})\} = \{(2,1), (1,0), (2,0), (4,0), (5,1)\}$$

The Gini impurity is: $Gini(p) = 2 \times \frac{2}{5} \times \left(1 - \frac{2}{5}\right) = \frac{12}{25}.$

If we use splitting rule: $x < 3$. Then the original dataset is split to 2 sub-datasets. The left sub-dataset contains samples: $\{(2,1), (1,0), (2,0)\}$ and the right sub-dataset contains samples: $\{(4,0), (5,1)\}$. Then the Gini impurity of left sub-dataset is: $Gini\ left(p) = 2 \times \frac{1}{3} \times \left(1 - \frac{1}{3}\right) = \frac{4}{9}$. And the Gini impurity of the right sub-dataset is: $Gini\ right(p) = 2 \times \frac{1}{2} \times \left(1 - \frac{1}{2}\right) = \frac{1}{2}$. The weighted summation of Gini index of two nodes are: $Gini\ sum(p) = \frac{3}{5} \cdot Gini\ left(p) + \frac{2}{5} \cdot Gini\ right(p).$

The branch keeps growing until in a certain step, the number of existing samples in a

specific leaf is less than a pre-defined value. In this experiment, when the number of existing samples in a specific leaf is less than 2, CART performs no more splitting at this node. When a new sample is input, the CART decision tree predicts according to the well-trained binary trees.

The random forest is an ensemble model by collecting the results of decision trees. In each random forest classifier, there are N independent decision trees.

For each decision tree, they complete the following procedures independently:

Step 1: Bootstrapping: Randomly select m samples from the training dataset with replacements. Since replacement is permitted, there may be some samples repeatedly selected while others being ignored.

Step 2: Randomly select several variables from the whole feature variables and drop out others.

Step 3: CART decision tree performs the training process.

When a new test sample is input, N CART decision trees provide their predictions. Denote $N(1)$ to be the number of positive samples in the selected K samples, and $N(0)$ to be the number of negative samples in the selected K samples. Then the prediction of test sample j is:

$$y^{(ej)} = \begin{cases} 1 & , N(1) \geq N(0) \\ 0 & , N(1) < N(0) \end{cases} \tag{3-4}$$

We choose different number of estimators $N = 20,40,60,80,100,120,140,180,200$.

3.3 Logistic regression

Logistic Regression Model is a basic model in supervised classification model. The logistic regression is a linear-like model which is widely applied because of its interpretability.

$$p(x_i) = Sigmoid(\theta_0 + \sum_{d=1}^{p} \theta_d x_d^{(i)}) \tag{3-5}$$

$$Sigmoid(x) = \frac{1}{1+e^{-x}} \tag{3-6}$$

$$y^{(i)} = \begin{cases} 1 & , p(x_i) \geq 0.5 \\ 0 & , p(x_i) < 0.5 \end{cases} \tag{3-7}$$

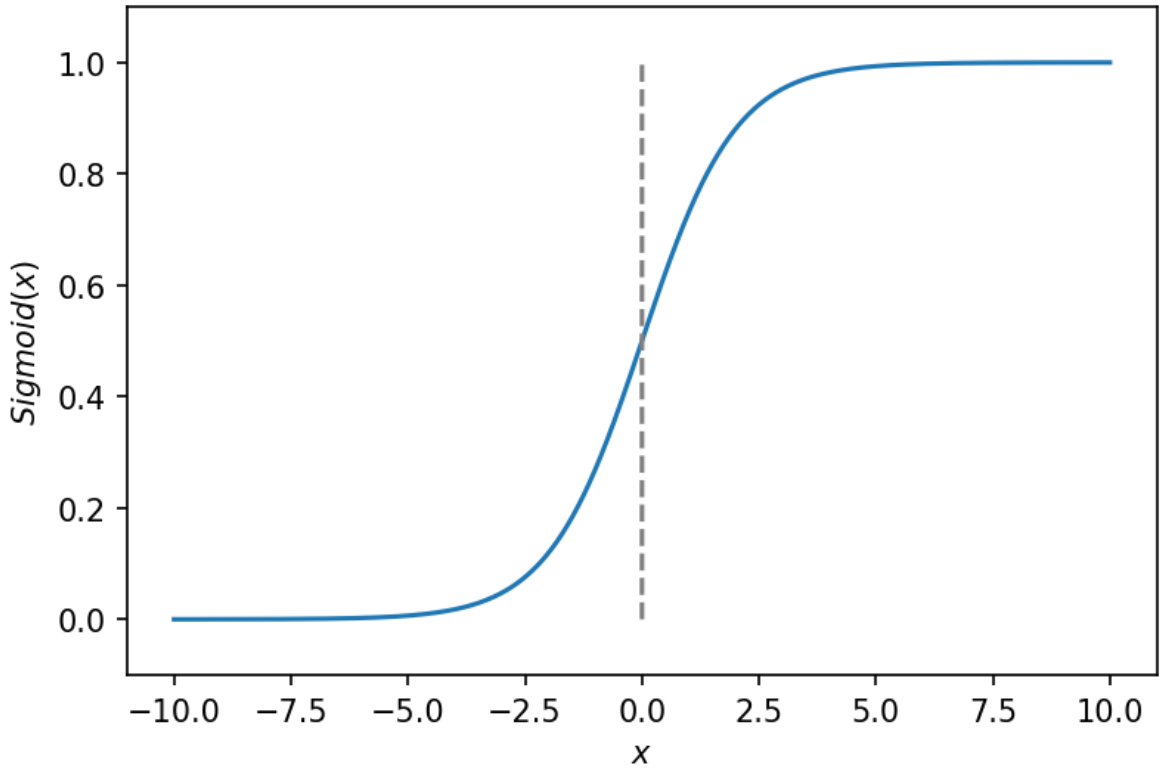


Figure 3-1 Sigmoid Function

Plot of sigmoid is displayed as Figure 3-1.

In this section, we need to introduce a loss function called binary cross-entropy:

$$BCE = \frac{1}{m}\sum_{i=1}^{m} BCE_i = -\frac{1}{m}\sum_{i=1}^{m}(y^{(i)}\log(p(\boldsymbol{x}^{(i)})) + (1-y^{(i)})\log(1-p(\boldsymbol{x}^{(i)}))) \tag{3-8}$$

The plot of binary cross-entropy function is displayed in Figure 3-2.

$$Loss_1 = C \cdot BCE + \sum_{d=1}^{p}|\theta_d| \tag{3-9}$$

$$Loss_2 = C \cdot BCE + \sqrt{\sum_{d=1}^{p}(\theta_d)^2} \tag{3-10}$$

m is the number of samples, and $p(\boldsymbol{x}^{(i)})$ represents prediction on probability that sample i is positive. And C is the hyperparameter controlling the regularization. In the training process, the machine uses gradient descent method for minimizing the loss function.

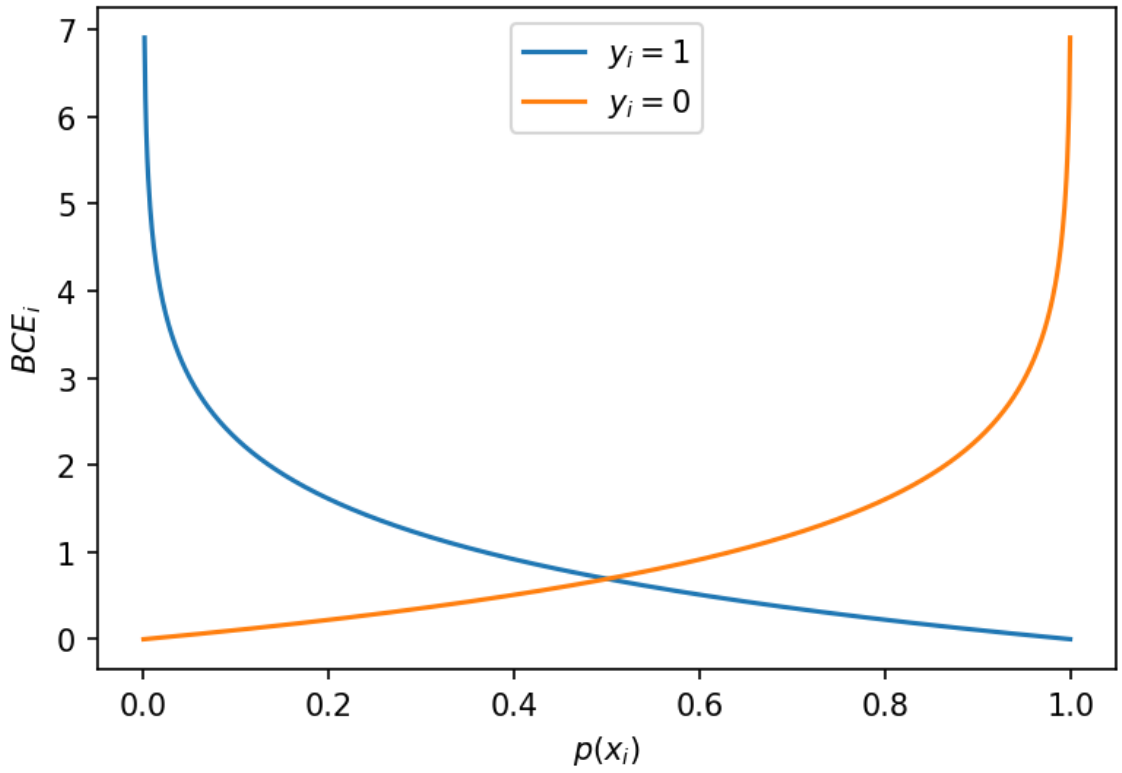


Figure 3-2 Binary Cross Entropy Function

In the experiment, we compare the performance of regularized logistic regression under 1-norm and 2-norm regularization method, with $C = 0.01,0.1,1,10,100,1000,10000$.

3.4 Multilayer perceptron

Multilayer Perceptron is a deep machine learning method. The structure of the multilayer perceptron is shown in Figure 3-3. In this problem, we use a two-layer perceptron for prediction. The neurons of each layer are hyperparameters. The hyperparameters are the neurons of each layer. In the experiment, neurons number on second layer is designed to be $(N2 = 8, 16, 32)$, and the neurons number on third layer is designed to be $(N3 = 4, 8, 16)$. In the classification task based on coordinate and velocity, since the feature variables available are $\bar{x}, \overline{xVel}, \bar{y}, \overline{yVel}$, therefore $N1 = 4$. In the classification task based on three forces, the feature variables are $\overline{xA}, \overline{yA}, \overline{xS}, \overline{yS}, \overline{xC}, \overline{yC}, \overline{nAC}, \overline{nS}$, therefore $N2 = 8$.

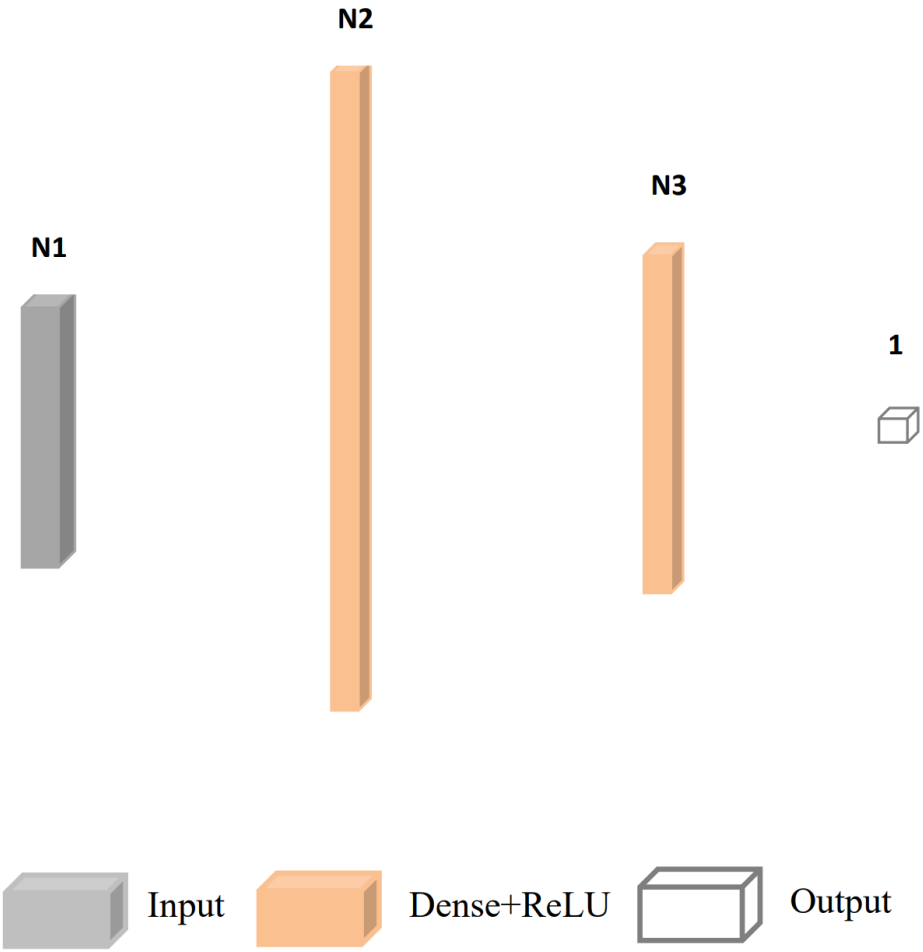


Figure 3-3 Architecture of Muti-layer Perceptron

Suppose that the input of the multi-layer perceptron is $x_1, x_2, \ldots, x_p$. Then the forward propagation of the muti-layer perceptron is as follows:

$$a_{2,k} = \mathrm{Re}\,LU(\sum_{d=1}^{p} w_{d,k}^{(1,2)} x_d + b_k^{(1,2)}), k = 1, 2, \ldots\ldots, N2 \tag{3-11}$$

$$a_{3,k} = \mathrm{Re}\,LU(\sum_{d=1}^{N2} w_{d,k}^{(2,3)} a_{2,d} + b_k^{(2,3)}), k = 1, 2, \ldots\ldots, N3 \tag{3-12}$$

$$p(\boldsymbol{x}) = Sigmoid(\sum_{d=1}^{N3} w_{d,1}^{(3,4)} a_{3,d} + b_1^{(3,4)}) \tag{3-13}$$

$$\mathrm{Re}\,LU(x) = \begin{cases} 0 & , x < 0 \\ x & , x \geq 0 \end{cases} \tag{3-14}$$

Explanation of the notation above: $\boldsymbol{x}$ is the sample, x_d is the $d-th$ feature of sample $\boldsymbol{x}$, also, it is the $d-th$ neuron on layer 1. $w_{d,k}^{(1,2)}$ and $b^{(1,2)}$ are the weight and bias of dense

layer connecting the neuron k on second layer and the neuron d on first layer. $a_{2,k}$ is the $k-th$ neuron on layer 2. ReLU and Sigmoid are known as activation functions, which could transform the linear function to be non-linear function. Another multi-layer perceptron is provided as Figure 3-4.

Binary cross-entropy, which is commonly used in classification model and the same as that of Regularized Logistic Regression, is the loss function in MLP as well.

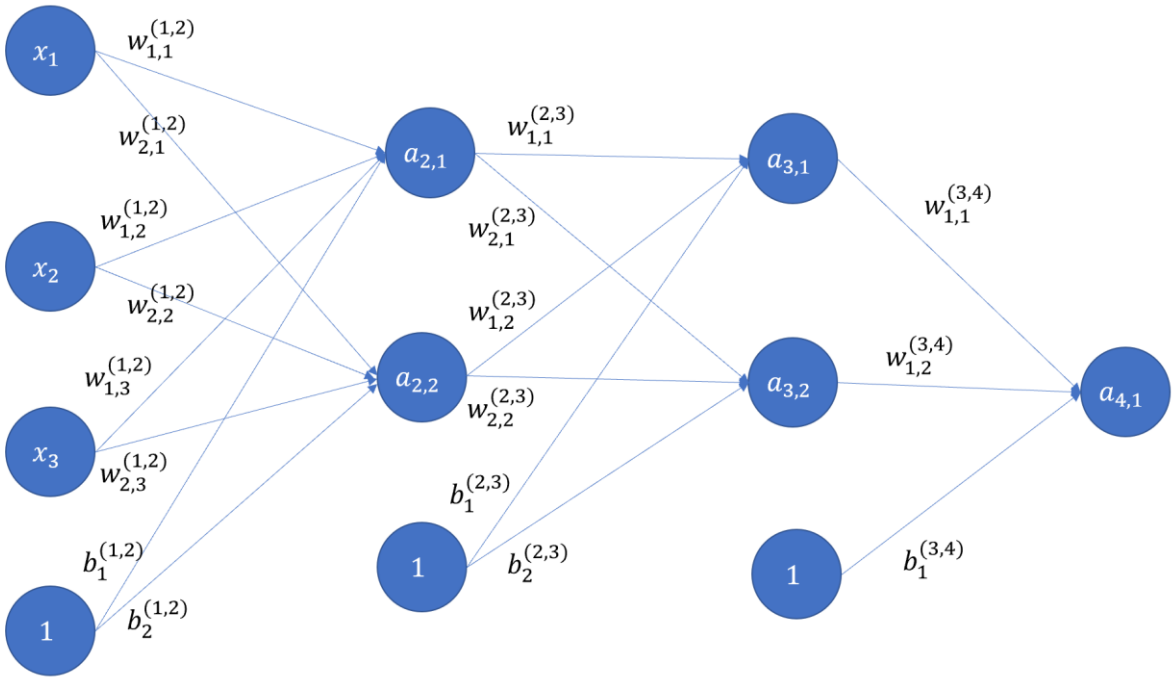


Figure 3-4 Notations of Muti-layer Perceptron

3.5 Convolutional neural network

The convolutional neural network takes the 2-D spatial information into consideration. In the multilayer perceptron, the 2-D spatial information is filtered out. Since we want to preserve the spatial information for training, the convolutional neural network which is versed in monitoring the spatial information is introduced to help with the 2D mapping method.

In this problem, the convolutional neural network is set to have two convolutional layers and a max-pooling layer for extracting some features from the input layers. And the two dense layers are classification layers. The architecture of CNN is displayed in Figure 3-5.

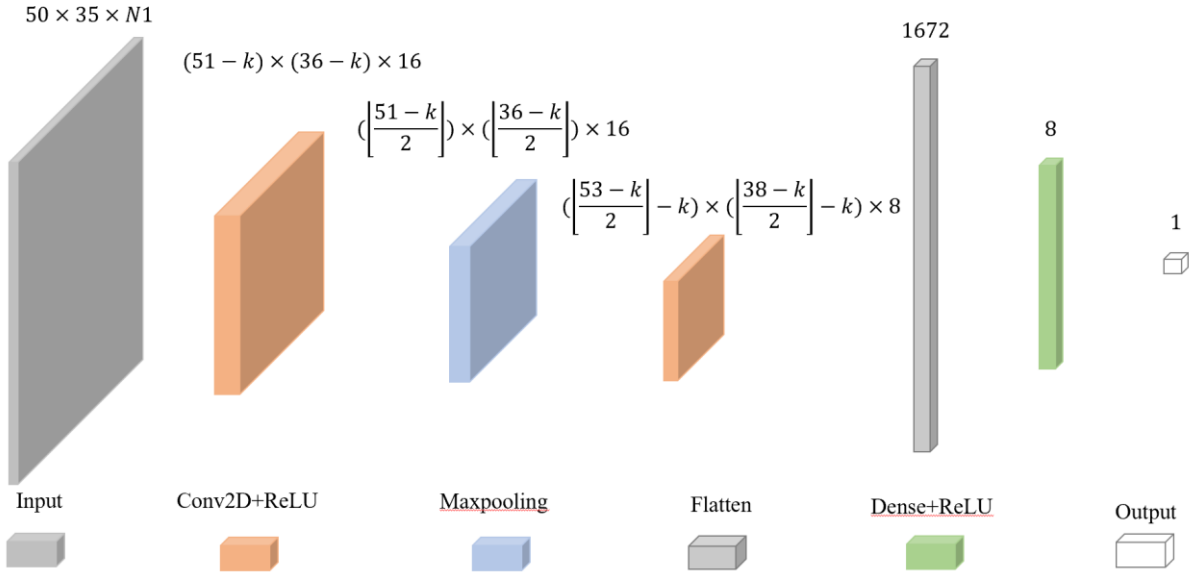


Figure 3-5 Architecture of Convolutional Neural Network

In the experiment, the kernel size is hyperparameter set to be $(k = 2,3,4,5,6,7)$, and $N1$ is the number of feature variables, which varies according to different tasks. In the classification task based on coordinate and velocity, $N1 = 3$ $(Indicator, xVel, yVel)$ and in the classification task based on three forces $N2 = 8$ $(xA, yA, xS, yS, xC, yC, nAC, nS)$. Binary cross-entropy is the loss function of CNN in this experiment.

Convolutional-2D layer: in the convolutional layer, the kernel (also known as filter) is an operator computing the weighted summation of the pixels in the window of kernel. If there are more than one channel of the image, the output is the summation of the output of each channel. For example, in Figure 3-6, if we have the following three-channel-image and kernel (kernel size = 2):

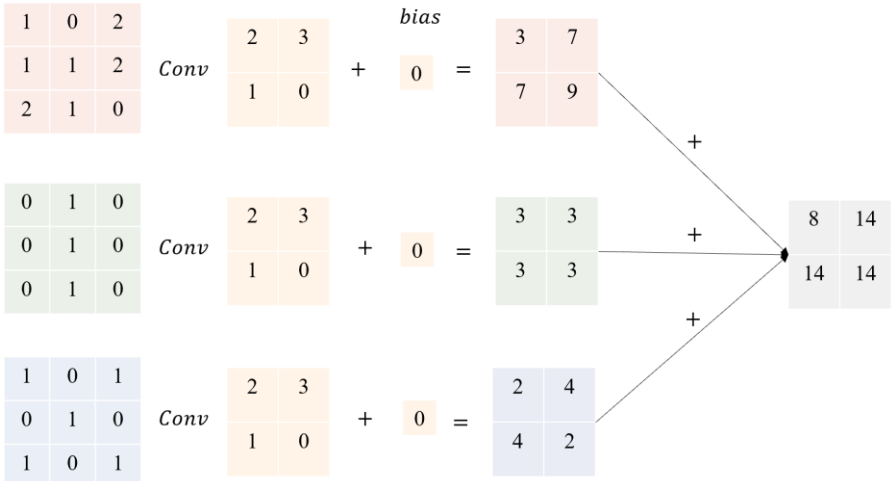


Figure 3-6 Computation of Convolutional Layer

In the red channel,

$$1 \times 2 + 0 \times 3 + 1 \times 1 + 1 \times 0 + 0 = 3$$

$$0 \times 2 + 2 \times 3 + 1 \times 1 + 2 \times 0 + 0 = 7$$

$$1 \times 2 + 1 \times 3 + 2 \times 1 + 1 \times 0 + 0 = 7$$

$$1 \times 2 + 2 \times 3 + 1 \times 1 + 0 \times 0 + 0 = 9$$

In the green channel,

$$0 \times 2 + 1 \times 3 + 0 \times 1 + 1 \times 0 + 0 = 3$$

$$1 \times 2 + 0 \times 3 + 1 \times 1 + 0 \times 0 + 0 = 3$$

$$0 \times 2 + 1 \times 3 + 0 \times 1 + 1 \times 0 + 0 = 3$$

$$1 \times 2 + 0 \times 3 + 1 \times 1 + 0 \times 0 + 0 = 3$$

In the blue channel,

$$1 \times 2 + 0 \times 3 + 0 \times 1 + 1 \times 0 + 0 = 2$$

$$0 \times 2 + 1 \times 3 + 1 \times 1 + 0 \times 0 + 0 = 4$$

$$0 \times 2 + 1 \times 3 + 1 \times 1 + 0 \times 0 + 0 = 4$$

$$1 \times 2 + 0 \times 3 + 0 \times 1 + 1 \times 0 + 0 = 2$$

The output is the summation of three channels of convoluted images:

$$3 + 3 + 2 = 8$$

$$7 + 3 + 4 = 14$$

$$7 + 3 + 4 = 14$$

$$9 + 3 + 2 = 14$$

Max-pooling Layer: in the max-pooling layer, there is a compresser which takes the maximum per 2×2 block. If the size of image is not divisible by 2, then the remaining part is dropped out. For example, in Figure 3-7:

1	3	4	2	5
2	9	7	2	6
1	3	9	0	0
5	4	3	6	0

$Maxpooling\ (2,2)\ =$

9	7
5	9

Figure 3-7 Computation of Max-pooling Layer

$$\max(1,3,2,9) = 9$$

$$\max(4,2,7,2) = 7$$

$$\max(1,3,5,4) = 5$$

$$\max(9,0,3,6) = 9$$

Since the shape is not divisible by the shape of max-pooling window, column $[5,6,0,0]$ is dropped out.

Flatten Layer: The flatten layer compile the $n-D$ tensor to a $1-D$ tensor. For example, in Figure 3-8:

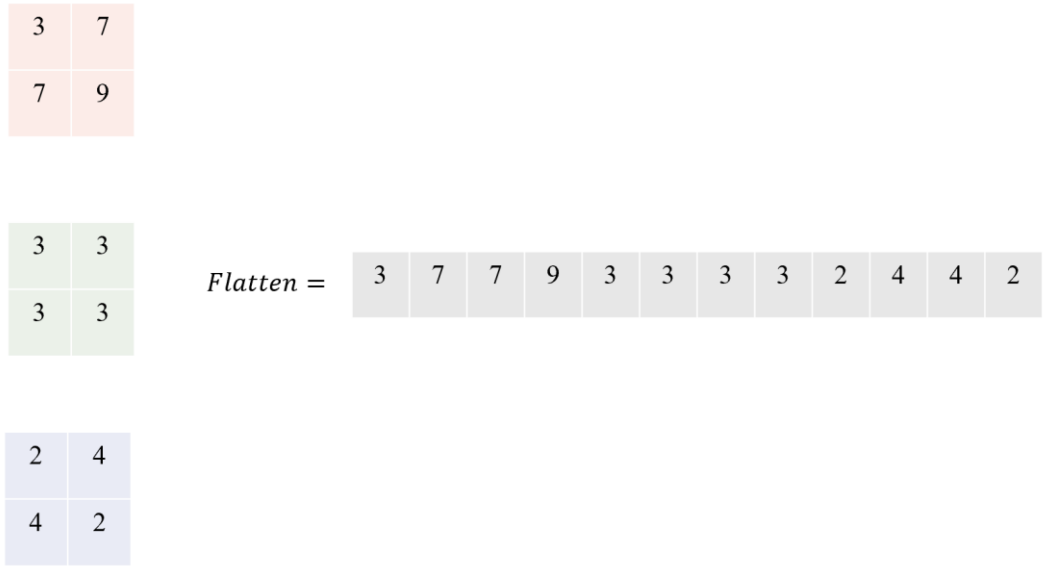


Figure 3-8 Computation of Flatten Layer

4 Implementation

The experimental environment is provided in Table 4-1.

Table 4-1 Experimental Environment

Operating System	Windows 10 Family 64-bit (10.0)
Memory	8192MB RAM
Processor	Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz (8 CPUs), ~1.8GHz
Platform	Anaconda – Python 3.8
Main Packages	Pandas 1.4.2, Scipy 1.7.3, Tensorflow 2.10.0, Keras 2.10.0

The two features processing part is mainly completed by the build-in packages and pandas. In terms of classifiers, the KNN, random forest and logistic regression are implemented by Scipy and the two deep learning models are implemented by Tensorflow and Keras. Codes are attached in the appendix.

5 Results of centroid method

In this section, the centroid method is applied to classification task based on coordinate and velocity and classification task based on three forces, the results are as follows:

5.1 Classification task based on coordinate and velocity

5.1.1 Classification of Flocking Behaviors

Table 5-1 Centroid Method + KNN Classifier

K	Precision	Recall	F1-score	Accuracy
1	0.9733	0.9980	0.9855	0.9853
2	0.9716	0.9980	0.9846	0.9844
3	0.9728	0.9980	0.9853	0.9850
4	0.9726	0.9980	0.9851	0.9849
5	0.9733	0.9980	0.9855	0.9853
6	0.9724	0.9990	0.9855	0.9853
7	**0.9733**	**0.9990**	**0.9860**	**0.9858**
8	0.9724	0.9990	0.9855	0.9853

Table 5-2 Centroid Method + Random Forest Classifier

Estimators	Precision	Recall	F1-score	Accuracy
20	0.9556	0.9674	0.9615	0.9606
40	**0.9852**	0.9672	**0.9761**	**0.9752**
60	0.9813	0.9663	0.9738	0.9728
80	0.9838	0.9664	0.9750	0.9740
100	0.9801	0.9665	0.9733	0.9723
120	0.9825	0.9650	0.9737	0.9727
140	0.9823	0.9666	0.9744	0.9734
160	0.9823	**0.9675**	0.9749	0.9739
180	0.9818	0.9675	0.9746	0.9737
200	0.9833	0.9673	0.9752	0.9743

Table 5-3 Centroid Method + Logistic Regression Classifier

Penalty	C	Precision	Recall	F1-score	Accuracy
l1	0.01	0.7147	0.8658	0.7830	0.7961
l1	0.1	0.7164	0.8735	0.7872	0.8006
l1	1	**0.7166**	**0.8735**	**0.7873**	**0.8007**
l1	10	0.7166	0.8735	0.7873	0.8007
l1	100	0.7166	0.8735	0.7873	0.8007
l1	1000	0.7166	0.8735	0.7873	0.8007
l1	10000	0.7166	0.8735	0.7873	0.8007
l2	0.01	0.7149	0.8694	0.7846	0.7980
l2	0.1	0.7156	0.8721	0.7862	0.7996
l2	1	0.7164	0.8735	0.7872	0.8006
l2	10	0.7166	0.8735	0.7873	0.8007
l2	100	0.7166	0.8735	0.7873	0.8007
l2	1000	0.7166	0.8735	0.7873	0.8007
l2	10000	0.7166	0.8735	0.7873	0.8007

Table 5-4 Centroid Method + Muti-layer Perceptron Classifier

N2	N3	Precision	Recall	F1-score	Accuracy
8	4	0.8790	0.9552	0.9155	0.9165
8	8	0.9450	0.9489	0.9469	0.9455
8	16	0.9522	0.9564	0.9543	0.9531
16	4	0.8160	0.9332	0.8707	0.8752
16	8	0.9479	0.9602	0.9540	0.9530
16	16	0.9568	0.9617	0.9593	0.9582
32	4	0.9488	0.9619	0.9553	0.9543
32	8	0.9561	0.9636	0.9598	0.9588
32	16	**0.9615**	**0.9680**	**0.9647**	**0.9638**

According to Table 5-1, Table 5-2, Table 5-3, and Table 5-4, the KNN classifier in the

classification of flocking behavior based on the coordinate and velocity has the highest performance, while the random forest classifier has the second highest performance and the multi-layer perceptron ranks the third. However, the logistic regression model has poor performance, as two classes are not perfectly linearly separable.

5.1.2 Classification of Grouping Behaviors

Table 5-5 Centroid Method + KNN Classifier

K	Precision	Recall	F1-score	Accuracy
1	0.6017	0.9964	0.7503	0.8162
2	0.5881	0.9963	0.7396	0.8100
3	0.6126	0.9965	0.7587	0.8212
4	0.5992	0.9964	0.7484	0.8151
5	0.6194	0.9965	0.7639	0.8243
6	0.6052	0.9982	0.7536	0.8183
7	**0.6229**	**0.9983**	**0.7671**	**0.8265**
8	0.6142	0.9982	0.7605	0.8225

Table 5-6 Centroid Method + Random Forest Classifier

Estimators	Precision	Recall	F1-score	Accuracy
20	0.7936	**0.9952**	0.8831	0.9036
40	0.8643	0.9800	0.9185	0.9296
60	0.8203	0.9895	0.8970	0.9135
80	0.8646	0.9885	0.9224	0.9333
100	0.7868	0.9850	0.8748	0.8967
120	**0.8896**	0.9897	**0.9370**	**0.9451**
140	0.8627	0.9885	0.9213	0.9324
160	0.8635	0.9772	0.9169	0.9281
180	0.8755	0.9911	0.9297	0.9392
200	0.8575	0.9859	0.9173	0.9290

Table 5-7 Centroid Method + Logistic Regression Classifier

Penalty	C	Precision	Recall	F1-score	Accuracy
l1	0.01	0.5595	0.9167	**0.6949**	**0.7745**
l1	0.1	0.5617	0.9030	0.6926	0.7712
l1	1	**0.5644**	0.9018	0.6943	0.7719
l1	10	0.5639	0.9017	0.6939	0.7717
l1	100	0.5639	0.9017	0.6939	0.7717
l1	1000	0.5639	0.9017	0.6939	0.7717
l1	10000	0.5639	0.9013	0.6938	0.7716
l2	0.01	0.4976	**0.9596**	0.6553	0.7598
l2	0.1	0.5566	0.9066	0.6897	0.7702
l2	1	0.5628	0.9024	0.6932	0.7714
l2	10	0.5642	0.9018	0.6941	0.7718
l2	100	0.5639	0.9013	0.6938	0.7716
l2	1000	0.5639	0.9013	0.6938	0.7716
l2	10000	0.5639	0.9013	0.6938	0.7716

Table 5-8 Centroid Method + Muti-layer Perceptron Classifier

N2	N3	Precision	Recall	F1-score	Accuracy
8	4	0.5938	**0.9767**	0.7386	0.8071
8	8	0.6215	0.9259	0.7438	0.8035
8	16	0.6748	0.9331	0.7832	0.8286
16	4	0.6028	0.9402	0.7346	0.8001
16	8	0.6014	0.9672	0.7417	0.8077
16	16	0.6335	0.9545	0.7616	0.8180
32	4	0.6892	0.9676	**0.8050**	**0.8468**
32	8	**0.6955**	0.9516	0.8036	0.8440
32	16	0.6460	0.9429	0.7667	0.8196

According to Table 5-5, Table 5-6, Table 5-7, and Table 5-8, the random forest classifier in

the classification of flocking behavior based on the coordinate and velocity has the highest performance among all, with an accuracy of 94.51%. The multi-layer perceptron has the third highest performance plunging to 84.68%, while the KNN and logistic regression classifier is not well-applicable in the classification of the grouping behaviors.

5.1.3 Classification of aligning behaviors

Table 5-9 Centroid Method + KNN Classifier

K	Precision	Recall	F1-score	Accuracy
1	0.9993	1.0000	0.9997	0.9998
2	0.9983	1.0000	0.9992	0.9994
3	0.9997	1.0000	0.9998	0.9999
4	0.9997	1.0000	0.9998	0.9999
5	**1.0000**	**1.0000**	**1.0000**	**1.0000**
6	0.9997	1.0000	0.9998	0.9999
7	1.0000	1.0000	1.0000	1.0000
8	0.9997	1.0000	0.9998	0.9999

Table 5-10 Centroid Method + Random Forest Classifier

Estimators	Precision	Recall	F1-score	Accuracy
20	0.9993	0.9980	0.9986	0.9990
40	0.9997	0.9973	0.9985	0.9989
60	0.9997	0.9973	0.9985	0.9989
80	0.9997	0.9980	0.9988	0.9991
100	0.9997	0.9980	0.9988	0.9991
120	0.9997	0.9980	0.9988	0.9991
140	**1.0000**	**0.9980**	**0.9990**	**0.9993**
160	1.0000	0.9976	0.9988	0.9991
180	0.9997	0.9980	0.9988	0.9991
200	0.9997	0.9980	0.9988	0.9991

Table 5-11 Centroid Method + Logistic Regression Classifier

Penalty	C	Precision	Recall	F1-score	Accuracy
l1	0.01	**1.0000**	**0.9913**	**0.9956**	**0.9968**
l1	0.1	1.0000	0.9778	0.9888	0.9916
l1	1	1.0000	0.9733	0.9865	0.9899
l1	10	1.0000	0.9729	0.9863	0.9898
l1	100	1.0000	0.9736	0.9866	0.9900
l1	1000	1.0000	0.9729	0.9863	0.9898
l1	10000	1.0000	0.9736	0.9866	0.9900
l2	0.01	0.8311	1.0000	0.9078	0.9379
l2	0.1	1.0000	0.9797	0.9898	0.9924
l2	1	1.0000	0.9691	0.9843	0.9883
l2	10	1.0000	0.9713	0.9855	0.9891
l2	100	1.0000	0.9723	0.9860	0.9895
l2	1000	1.0000	0.9729	0.9863	0.9898
l2	10000	1.0000	0.9739	0.9868	0.9901

Table 5-12 Centroid Method + Muti-layer Perceptron Classifier

N2	N3	Precision	Recall	F1-score	Accuracy
8	4	1.0000	0.9791	0.9894	0.9921
8	8	1.0000	0.9733	0.9865	0.9899
8	16	**1.0000**	**0.9827**	**0.9913**	**0.9935**
16	4	1.0000	0.9807	0.9903	0.9928
16	8	1.0000	0.9807	0.9903	0.9928
16	16	1.0000	0.9755	0.9876	0.9908
32	4	1.0000	0.9762	0.9879	0.9910
32	8	1.0000	0.9765	0.9881	0.9911
32	16	1.0000	0.9710	0.9853	0.9890

According to Table 5-9, Table 5-10, Table 5-11, and Table 5-12, the KNN classifier has the highest performance among all, with an accuracy of 100%. But this is not a huge improvement compared to the unprocessed feature method, with an accuracy of 99.25%. The main reason for such stunning accuracy is that there exists a hyperplane that perfectly separates the aligning and non-aligning, which could be also verified by the high performance of the logistic regression.

5.2 Classification task based on three forces

5.2.1 Classification of flocking behaviors

Table 5-13 Centroid Method + KNN Classifier

K	Precision	Recall	F1-score	Accuracy
1	0.9874	0.9855	0.9864	0.9860
2	0.9855	0.9857	0.9856	0.9852
3	0.9893	0.9848	0.9871	0.9867
4	0.9879	0.9927	0.9903	0.9900
5	**0.9898**	**0.9927**	**0.9913**	**0.9910**
6	0.9886	0.9927	0.9906	0.9904
7	0.9896	0.9927	0.9911	0.9909
8	0.9886	0.9927	0.9906	0.9904

Table 5-14 Centroid Method + Random Forest Classifier

Estimators	Precision	Recall	F1-score	Accuracy
20	**1.0000**	**0.9993**	**0.9996**	**0.9996**
40	1.0000	0.9990	0.9995	0.9995
60	1.0000	0.9990	0.9995	0.9995
80	1.0000	0.9990	0.9995	0.9995
100	1.0000	0.9993	0.9996	0.9996
120	1.0000	0.9990	0.9995	0.9995
140	1.0000	0.9993	0.9996	0.9996
160	1.0000	0.9990	0.9995	0.9995
180	1.0000	0.9990	0.9995	0.9995
200	1.0000	0.9993	0.9996	0.9996

Table 5-15 Centroid Method + Logistic Regression Classifier

Penalty	C	Precision	Recall	F1-score	Accuracy
l1	0.01	0.8199	0.9205	0.8673	0.8709
l1	0.1	0.8662	0.9870	0.9227	0.9253
l1	1	0.8928	0.9639	0.9270	0.9276
l1	10	0.8977	0.9255	0.9114	0.9102
l1	100	**0.8979**	**0.9246**	**0.9111**	**0.9098**
l1	1000	0.8979	0.9246	0.9111	0.9098
l1	10000	0.8979	0.9246	0.9111	0.9098
l2	0.01	0.8075	0.9199	0.8601	0.8648
l2	0.1	0.8116	0.9347	0.8688	0.8739
l2	1	0.8344	0.9435	0.8856	0.8891
l2	10	0.8645	0.9978	0.9264	0.9293
l2	100	0.8844	0.9905	0.9344	0.9361
l2	1000	0.8953	0.9333	0.9139	0.9132
l2	10000	0.8979	0.9246	0.9111	0.9098

Table 5-16 Centroid Method + Muti-layer Perceptron Classifier

N2	N3	Precision	Recall	F1-score	Accuracy
8	4	0.9183	0.9228	0.9205	0.9184
8	8	1.0000	0.9937	0.9969	0.9968
8	16	0.8955	0.9880	0.9395	0.9406
16	4	0.9016	0.9917	0.9445	0.9455
16	8	0.9018	0.8947	0.8982	0.8948
16	16	0.8868	0.9956	0.9381	0.9397
32	4	1.0000	0.9925	0.9963	0.9961
32	8	**1.0000**	**0.9947**	**0.9973**	**0.9973**
32	16	1.0000	0.9928	0.9964	0.9963

According to Table 5-13, Table 5-14, Table 5-15, Table 5-16, in the classification task of

the flocking behaviors based on three forces, the centroid method combined with random forest classifier seems to achieve the most satisfying result, and the multi-layer perceptron and KNN classifier has a relatively lower accuracy. For logistic regression classifier, we could see that the logistic regression only achieves 90.98% accuracy in the prediction, which means that the hyperplane could not perfectly separate the flocking and non-flocking swarms.

5.2.2 Classification of grouping behaviors

Table 5-17 Centroid Method + KNN Classifier

K	Precision	Recall	F1-score	Accuracy
1	0.9937	0.9838	0.9888	0.9896
2	0.9932	0.9841	0.9886	0.9895
3	0.9976	0.9836	0.9906	0.9913
4	0.9967	0.9922	0.9944	0.9949
5	0.9989	0.9922	0.9955	0.9959
6	0.9981	0.9922	0.9951	0.9955
7	**0.9992**	**0.9922**	**0.9957**	**0.9960**
8	0.9984	0.9922	0.9953	0.9956

Table 5-18 Centroid Method + Random Forest Classifier

Estimators	Precision	Recall	F1-score	Accuracy
20	1.0000	0.9895	0.9947	0.9951
40	1.0000	0.9911	0.9955	0.9959
60	**1.0000**	**0.9954**	**0.9977**	**0.9979**
80	1.0000	0.9911	0.9955	0.9959
100	1.0000	0.9946	0.9973	0.9975
120	1.0000	0.9949	0.9974	0.9976
140	1.0000	0.9946	0.9973	0.9975
160	1.0000	0.9943	0.9972	0.9974
180	1.0000	0.9943	0.9972	0.9974
200	1.0000	0.9949	0.9974	0.9976

Table 5-19 Centroid Method + Logistic Regression Classifier

Penalty	C	Precision	Recall	F1-score	Accuracy
l1	0.01	0.8798	0.9926	0.9328	0.9419
l1	0.1	0.9576	0.9924	0.9747	0.9772
l1	1	0.9992	0.9906	**0.9949**	**0.9953**
l1	10	**1.0000**	0.9890	0.9945	0.9949
l1	100	1.0000	0.9884	0.9942	0.9946
l1	1000	1.0000	0.9884	0.9942	0.9946
l1	10000	1.0000	0.9884	0.9942	0.9946
l2	0.01	0.7580	0.9078	0.8262	0.8536
l2	0.1	0.9029	0.9849	0.9421	0.9491
l2	1	0.9323	**0.9939**	0.9621	0.9663
l2	10	0.9671	0.9916	0.9792	0.9812
l2	100	0.9962	0.9905	0.9934	0.9939
l2	1000	1.0000	0.9895	0.9947	0.9951
l2	10000	1.0000	0.9884	0.9942	0.9946

Table 5-20 Centroid Method + Muti-layer Perceptron Classifier

N2	N3	Precision	Recall	F1-score	Accuracy
8	4	1.0000	0.9839	0.9919	0.9925
8	8	1.0000	0.9832	0.9915	0.9921
8	16	0.9989	0.9837	0.9912	0.9919
16	4	**1.0000**	**0.9895**	**0.9947**	**0.9951**
16	8	0.9981	0.9776	0.9878	0.9886
16	16	0.9959	0.9855	0.9907	0.9914
32	4	1.0000	0.9753	0.9875	0.9884
32	8	0.9995	0.9816	0.9904	0.9911
32	16	0.9995	0.9826	0.9910	0.9916

According to Table 5-17, Table 5-18, Table 5-19, Table 5-20, in the classification task of the grouping behaviors based on three forces, random forest classifier has the highest performance among all, with F1-score reaches 0.9977 and accuracy reaches 99.79%. The KNN, logistic regression and muti-layer perceptron remain accuracy more than 99%, which means that the grouping and non-grouping swarms are distinguishable in the hyper-space.

5.2.3 Classification of aligning behaviors

Table 5-21 Centroid Method + KNN Classifier

K	Precision	Recall	F1-score	Accuracy
1	0.9902	0.9908	0.9905	0.9930
2	0.9881	0.9912	0.9896	0.9924
3	0.9881	0.9912	0.9896	0.9924
4	0.9871	1.0000	0.9935	0.9953
5	**0.9871**	**1.0000**	**0.9935**	**0.9953**
6	0.9864	1.0000	0.9932	0.9950
7	0.9864	1.0000	0.9932	0.9950
8	0.9861	1.0000	0.9930	0.9949

Table 5-22 Centroid Method + Random Forest Classifier

Estimators	Precision	Recall	F1-score	Accuracy
20	**0.9990**	**1.0000**	**0.9995**	**0.9996**
40	0.9983	1.0000	0.9992	0.9994
60	0.9902	1.0000	0.9951	0.9964
80	0.9932	1.0000	0.9966	0.9975
100	0.9946	1.0000	0.9973	0.9980
120	0.9925	1.0000	0.9963	0.9973
140	0.9969	1.0000	0.9985	0.9989
160	0.9929	1.0000	0.9964	0.9974
180	0.9942	1.0000	0.9971	0.9979
200	0.9908	1.0000	0.9954	0.9966

Table 5-23 Centroid Method + Logistic Regression Classifier

Penalty	C	Precision	Recall	F1-score	Accuracy
l1	0.01	1.0000	0.9374	0.9677	0.9754
l1	0.1	0.9871	0.9915	0.9893	0.9921
l1	1	0.9861	0.9925	0.9893	0.9921
l1	10	0.9861	0.9908	0.9884	0.9915
l1	100	0.9868	0.9895	0.9881	0.9913
l1	1000	0.9875	0.9891	0.9883	0.9914
l1	10000	0.9875	0.9891	0.9883	0.9914
l2	0.01	0.8538	**1.0000**	0.9212	0.9462
l2	0.1	**1.0000**	0.9827	**0.9913**	**0.9935**
l2	1	0.9871	0.9854	0.9863	0.9899
l2	10	0.9864	0.9922	0.9893	0.9921
l2	100	0.9861	0.9918	0.9889	0.9919
l2	1000	0.9861	0.9895	0.9878	0.9910
l2	10000	0.9868	0.9878	0.9873	0.9906

Table 5-24 Centroid Method + Muti-layer Perceptron Classifier

N2	N3	Precision	Recall	F1-score	Accuracy
8	4	0.9871	0.9905	0.9888	0.9918
8	8	0.9915	0.9915	0.9915	0.9938
8	16	0.9939	0.9919	0.9929	0.9948
16	4	0.9868	0.9922	0.9895	0.9923
16	8	0.9912	0.9919	0.9915	0.9938
16	16	**0.9973**	0.9912	**0.9943**	**0.9958**
32	4	0.9905	0.9946	0.9925	0.9945
32	8	0.9956	0.9922	0.9939	0.9955
32	16	0.9919	**0.9949**	0.9934	0.9951

According to Table 5-21, Table 5-22, Table 5-23, and Table 5-24, in the classification task of the aligning behaviors based on three forces, the random forest classifier still has the highest performance, followed by the multi-layer perceptron and KNN. Still, the logistic regression classifier is the worst classifier of all.

6 Results of 2D mapping method

In this section, the 2D mapping method is applied to classification task based on coordinate and velocity and classification task based on three forces, the results are as follows:

6.1 Classification task based on coordinate and velocity

Table 6-1 2D Mapping Method + Convolutional Neural Network Classifier (Flocking)

Kernel Size	Precision	Recall	F1-score	Accuracy
2	0.9501	0.9985	0.9737	0.9735
3	0.9908	0.9985	0.9946	0.9945
4	**0.9968**	**1.0000**	**0.9984**	**0.9984**
5	0.9918	0.9993	0.9955	0.9954
6	0.9952	0.9985	0.9968	0.9968
7	0.9978	0.9932	0.9955	0.9954

Table 6-1 depicts the results of the classification of flocking behaviors based on coordinate and velocity, the combination of the 2D mapping method and convolutional neural network classifier has the highest performance among all five classifiers including the four methods in the last section. When the kernel size is 4, the convolutional neural network has the highest performance compared to other settings of kernel size, with a 99.84% accuracy.

Table 6-2 2D Mapping Method + Convolutional Neural Network Classifier (Grouping)

Kernel Size	Precision	Recall	F1-score	Accuracy
2	0.8124	0.9861	0.8909	0.9087
3	0.8961	0.9871	0.9394	0.9470
4	0.9135	0.9874	0.9490	0.9550
5	0.9095	0.9873	0.9468	0.9531
5	0.9016	0.9872	0.9424	0.9495
6	**0.9296**	**0.9879**	**0.9578**	**0.9624**
7	0.9258	0.9878	0.9558	0.9607

Table 6-2 depicts the results of the classification task of grouping behaviors based on coordinate and velocity, the combination of 2D mapping method and convolutional neural network classifier also has the highest performance among all. When the kernel size is 6, the CNN does the best, with a 96.24% accuracy on the test dataset. This means that the CNN classifier requires a larger window of filter for recognizing the grouping behavior.

Table 6-3 2D Mapping Method + Convolutional Neural Network Classifier (Aligning)

Kernel Size	Precision	Recall	F1-score	Accuracy
2	0.8121	1.0000	0.8963	0.9309
3	0.9793	1.0000	0.9895	0.9924
4	**1.0000**	**1.0000**	**1.0000**	**1.0000**
5	1.0000	1.0000	1.0000	1.0000
6	1.0000	1.0000	1.0000	1.0000
7	0.9997	1.0000	0.9998	0.9999

Table 6-3 depicts the results of the classification task of aligning behaviors based on coordinate and velocity, both the CNN and KNN classifiers could reach 100% accuracy in the experiment. But compared to the KNN, the CNN is relatively stable, as it is hard for us to determine the hyperparameter of KNN.

To summarize, the 2D mapping feature processing method and CNN classifier has the highest performance in three binary classification tasks based on coordinate and velocity. This makes sense as the centroid method eliminates the features of individuals, while the 2D mapping method preserves the features of individuals in the swarm.

6.2 Classification task based on three forces

Table 6-4 2D Mapping Method + Convolutional Neural Network Classifier (Flocking)

Kernel Size	Precision	Recall	F1-score	Accuracy
2	0.9093	0.9960	0.9507	0.9515
3	0.9920	1.0000	0.9960	0.9959
4	0.9156	1.0000	0.9560	0.9566
5	0.9653	1.0000	0.9824	0.9822
6	**0.9995**	**1.0000**	**0.9998**	**0.9998**
7	0.9617	1.0000	0.9805	0.9803

Table 6-4 depicts the results of the classification of the flocking behaviors based on three forces. *T*he convolutional neural network has the highest performance in the classification of the flocking behaviors based on three forces, but we notice that the in this problem the performance of CNN is not comparable to the random forest on average. The setting of the hyperparameter has a huge influence on the performance of the classifier. For example, when the kernel size is too small (kernel size = 2), the accuracy plunges to 95.15%.

Table 6-5 2D Mapping Method + Convolutional Neural Network Classifier (Grouping)

Kernel Size	Precision	Recall	F1-score	Accuracy
2	0.5212	0.9933	0.6837	0.7787
3	0.7999	0.9997	0.8887	0.9080
4	0.8562	0.9915	0.9189	0.9306
5	**0.9391**	**0.9988**	**0.9680**	**0.9716**
6	0.8785	0.9988	0.9348	0.9437
7	0.9241	0.9988	0.9600	0.9647

Table 6-5 depicts the results of the classification of the grouping behaviors based on three forces. The convolutional neural network has the lowest performance in the classification of the grouping behaviors based on three forces among all five combinations of feature processing

methods and classifiers. The setting of the hyperparameter also has a huge influence on the performance of the classifier. CNN with a small kernel size is likely to underfit the data while CNN with a large kernel size is likely to overfit the data.

Table 6-6 2D Mapping Method + Convolutional Neural Network Classifier (Aligning)

Kernel Size	Precision	Recall	F1-score	Accuracy
2	0.8945	0.9876	0.9388	0.9571
3	0.9888	0.9915	0.9902	0.9928
4	0.9895	0.9997	0.9945	0.9960
5	0.9888	1.0000	0.9944	0.9959
6	0.9885	1.0000	0.9942	0.9958
7	**0.9895**	**1.0000**	**0.9947**	**0.9961**

Table 6-6 depicts the results of the classification of the aligning behaviors based on three forces. The performance of the convolutional neural network is also not comparable to the combination of the centroid method and CNN classifier in the classification of the aligning behaviors based on three forces. In recognizing the aligning behaviors, a larger kernel size helps with improving the accuracy.

In the task of classification based on three forces, we see that the overall performance of 2D mapping method combined with the convolutional neural network is worse than that of the centroid method combined with random forest. There are two reasons for this phenomenon: (1) The centroid method is a great representation of the behaviors of swarms, using the centroid as features would be better than using the sparse 2D mapping. (2) The input size of the combination of 2D mapping method and CNN classifier is much larger than the centroid method, it would be quite hard for the convolutional neural network to train. The CNN is always trapped by overfitting and underfitting problems and sometimes falls into the unsatisfying local minimum in our experiments.

7 Importance of features

We want to investigate the importance of variables in both tasks. In the classification task based on coordinate and velocity, we know that the convolutional neural network performs the best, and the random forest on average ranks first in the classification task based on three forces. Therefore, we would choose these two models for analyzing the importance of variables. The convolutional neural network suffers from unsatisfactory interpretability. The importance of a feature could not be predicted from its coefficient, or hypothesis test. Therefore, the permutation importance is introduced to quantify the importance of features in the convolutional neural network. The importance of variables of random forest is qualified by Gini Importance.

7.1 Permutation importance

Suppose that the variables are now $[x_1, x_2, \dots\dots, x_p]$. Given a classifier M, and Dataset: D. Suppose that the F1-score under model M with respect to dataset D is F_0. Then, F_1 is defined to be the F1-score under model M with respect to D_1, where D_1 is a random permutation of x_1, when remaining other features x_i s of D.

Then, the importance of feature variable x_1 is defined to be:

$$I_1 = F_1 - F_0 \tag{7-1}$$

Apply the normalization to permutation importance:

$$\tilde{I}_1 = \frac{I_1}{\sum_{d=1}^{p} I_d} \tag{7-2}$$

7.2 Importance of features in classification task based on coordinate and velocity

The convolutional neural network with kernel size = (4,4) is used for analyzing the importance of variables in flocking and aligning behaviors, and the convolutional neural network with kernel size = (6,6) is used for analyzing the importance of variables in grouping behavior.

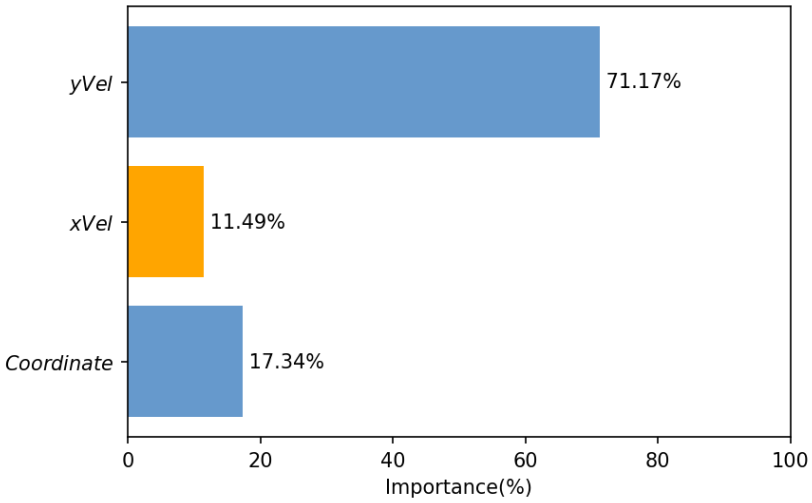


Figure 7-1 Importance of Features (Flocking)

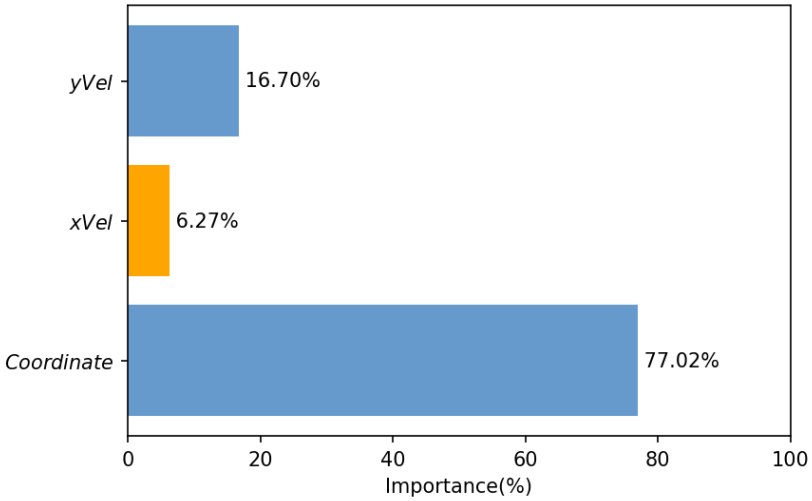


Figure 7-2 Importance of Features (Grouping)

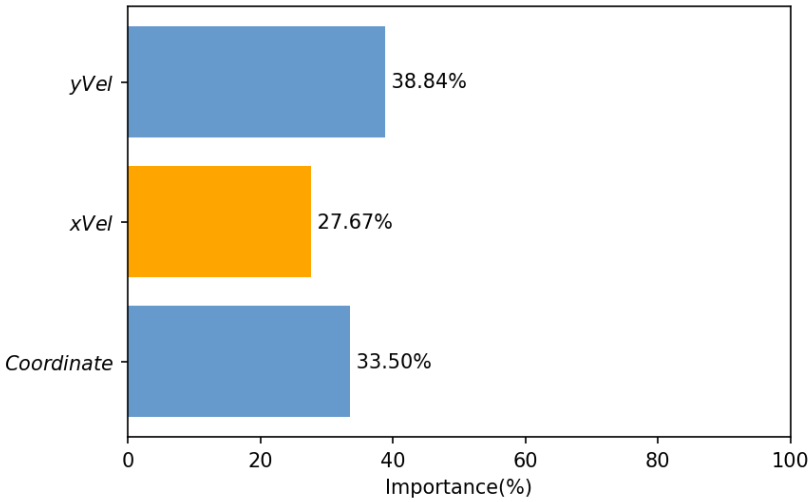


Figure 7-3 Importance of Features (Aligning)

In classification task based on coordinate and velocity, the partial velocity on y axis ($yvel$) takes the dominant place among all three variables in predicting the flocking behaviors (Figure 7-1), while the partial velocity on x axis ($xvel$) and Indictor variable (stands for coordinate) contributes less to prediction. However, in the classification of grouping behavior, indicator variable shows its dominance (Figure 7-2). All three features are broadly the same important in the recognition of aligning behavior (Figure 7-3).

7.3 Gini importance

The random forest has a special measure of the importance of feature variables which is called Gini Importance. According to its name, Gini importance means the level of decrease in Gini impurity after the introduction of a specific variable. For example, in a step the CART introduces a rule $x_1 > 2$, and the Gini impurity before splitting is 0.5, and the Gini impurity of the weighted average of right and left nodes is 0.3, then the Gini importance of this node is 0.2. This simple example may help with understanding.

Suppose that there is a well-trained CART, the computing process of Gini importance is as follows: Traverse all splitting nodes and compute their Gini importance, then the Gini importance of a specific feature is the weighted average of splitting nodes using this feature as rule. The weight equals the sample reaches this node. The random forest consists of several CART and the Gini importance of a specific feature is the mean average of Gini importance of all CARTs that use this feature in classification.

Figure 7-4 is an example of computing the Gini importance of a well-trained CART. On the first node, the Gini impurity is 0.8, and Gini impurity of weighted average of the left node and right node of the first node is $0.5 \times \frac{10}{30} + 0.2 \times \frac{20}{30} = 0.3$. Then the Gini importance at this node is $0.8 - 0.3 = 0.5$. On the leftmost node on the second layer, the Gini impurity is 0.5, and the Gini impurity of weighted average of the left node and right node of this node is $0.2 \times \frac{3}{10} + 0.1 \times \frac{7}{10} = 0.13$. Then the Gini importance at this node is $0.5 - 0.13 = 0.37$. On the rightmost node on the third layer, the Gini impurity is 0.1, and the Gini impurity of weighted average of the left node and right node of this node is $0 \times \frac{4}{7} + 0 \times \frac{3}{7} = 0$. Then the Gini importance at this node is $0.1 - 0 = 0.1$. The Gini importance of feature x_1: $GI(x_1) =$

$0.5 \times \frac{30}{100} + 0.1 \times \frac{7}{100} = 0.157$, and Gini importance of feature x_2: $GI(x_2) = 0.5 \times \frac{10}{100} = 0.05$. Apply normalization on the Gini importance: $\widetilde{GI}(x_1) = \frac{0.157}{0.157+0.05} = 0.7585$ and $\widetilde{GI}(x_2) = \frac{0.05}{0.157+0.05} = 0.2415$.

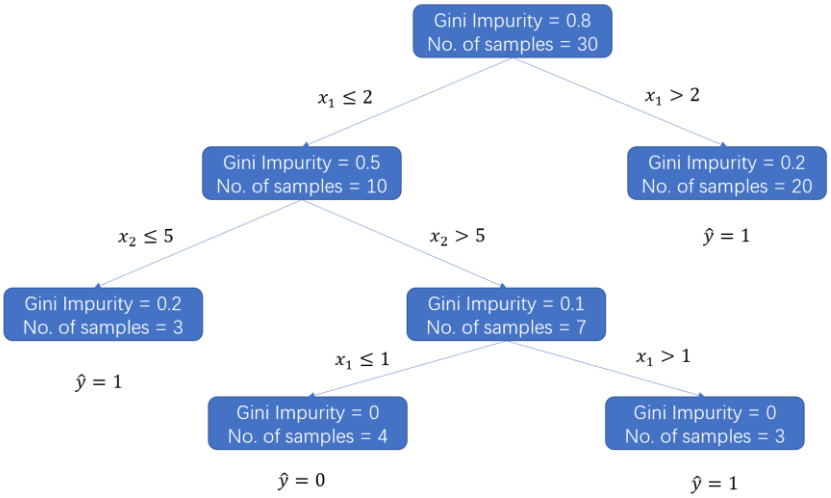


Figure 7-4 Computation of Gini Importance

7.4 Importance of features in classification task based on three forces

We choose the random forest classifiers and compute the Gini importance of feature variables. The random forest with 20 estimators is used in the classification of flocking behaviors, the random forest with 60 estimators is used in the classification of grouping behaviors, and the random forest with 40 estimators is used in the classification of aligning behaviors.

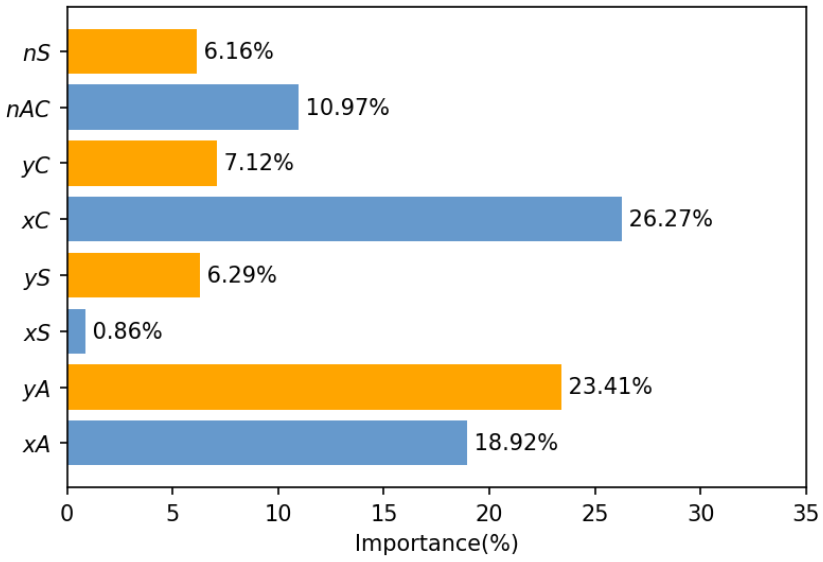


Figure 7-5 Importance of Features (Flocking)

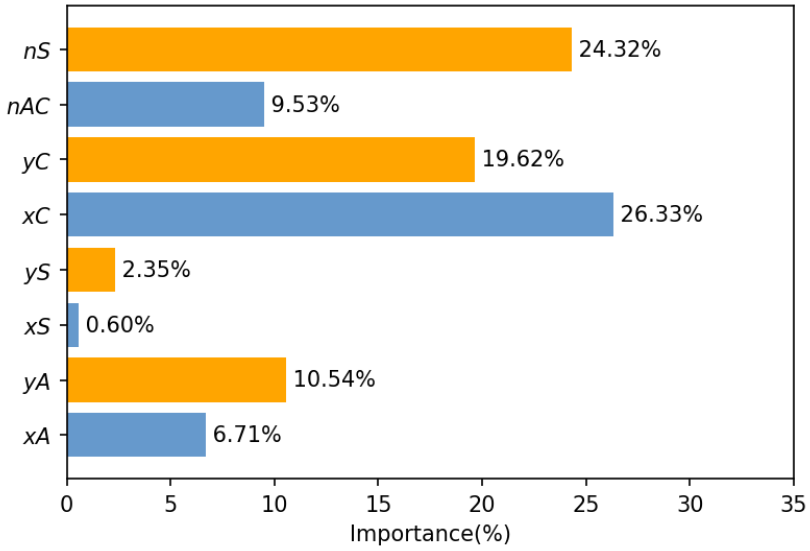


Figure 7-6 Importance of Features (Grouping)

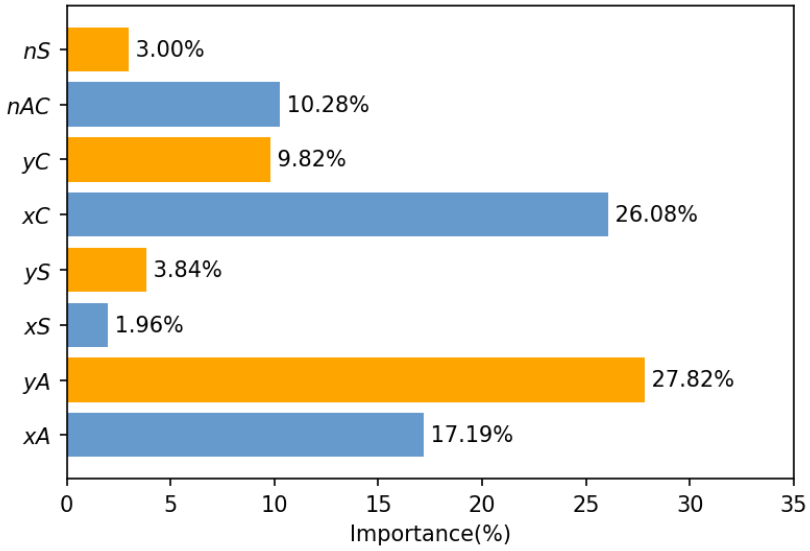


Figure 7-7 Importance of Features (Aligning)

In the classification of flocking behaviors in classification task based on three forces (Figure 7-5), the partial cohesion force on x axis (xC) contributes the most among all features, followed by partial alignment force on y and x axis (yA, xA). We may infer from this graph that the flocking behavior of this creature is closely related to both cohesion and alignment forces, and less related to the separation force.

In the classification of grouping behaviors in classification task based on three forces (Figure 7-6), the partial alignment force on x axis (xC) contributes the most among all features. It is also noteworthy that the number of boids in the radius of separation force has the

second highest importance, followed by partial alignment force on y axis (yC). But strangely, the contributions of partial separation forces on both axis are too small to be noticed. We may infer from this graph that the grouping behavior of this creature is closely related to both cohesion and separation forces, and less related to the alignment force.

In the classification of aligning behaviors in classification task based on three forces (Figure 7-7), the partial alignment force on y axis (yA) takes the first place in importance. This seems to make sense as the alignment force contributes the aligning behavior literally. The Gini importance of partial cohesion and alignment forces on x axis (xA, xC) also have an unignorable contribution to the tasks. But the separation force is almost ignorable. We may infer from this graph that the aligning behavior of this creature is closely related to alignment force, and secondly the cohesion force, and almost bears no relation to separation force.

8 Conclusion

In the classification task based on coordinate and velocity, we complete three binary classification tasks of the swarm behaviors based on the velocity and coordinate. The combination of the centroid method and K-nearest neighbors, the combination of centroid method and random forest classifier could achieve an accuracy of at most 98.58% in flocking behavior, 94.51% in grouping behavior, and 100% in aligning behavior, while the combination of 2D mapping method and convolutional neural network classifier could achieve higher accuracy of 99.84% in flocking behavior, 96.25% in grouping behavior, and 100% in aligning behavior. The 2D mapping method is better than the centroid method in the binary classification based on positional coordinates and velocity. The centroid method eliminates the relative positional information between boids, while the 2D mapping method preserves this important information. For human beings, we could recognize the swarm behaviors based on the shape and motion of 200 boids, this is also the rationale for us to use the 2D mapping method, and the reason for better performance of the 2D mapping method compared to the centroid method. The partial velocity on y axis is the most important feature of flocking behavior, and the coordinate is the most important feature of grouping behavior. Velocity and coordinate are almost the same important features of the aligning behaviors. Both the centroid feature processing method and 2D mapping feature processing method have better performance compared to directly using the unprocessed features for prediction.

In the classification task based on three forces, we recognize the swarm behavior based on the alignment, cohesion, and separation forces. We discover that the combination of centroid method and random forest has relatively great performance in the binary classification of all three behaviors. Although the 2D mapping method combined with the convolutional neural network is overall not the best compared to the centroid method combined with random forest classifier, it still remains a satisfying accuracy. We could infer that the positional information of boids is not as important as that in the classification task based on coordinate and velocity, and the tendency of the swarm centroid is a more significant feature. The classifier could easily separate the positive and negative samples by focusing on the tendency of the swarm centroid rather than the whole 200 boids. In the importance analysis, we discover that the flocking behavior is closely related to cohesion and alignment

forces, grouping behavior is closely related to separation and cohesion forces, and aligning behavior is more related to alignment and cohesion forces.

Acknowledgements

This is the end of this thesis. At this time, I would like to appreciate the patience of all readers of this thesis. Also, I would like to express my sincerely grateful feelings to the people who help and guide me during my four-year undergraduate time.

Thanks to Prof. Xiaohong Zhu, the tutor of me. Xiaohong was patient in giving guidance and correcting me in choosing the research topic and conducting the experiments. Also, Xiaohong encouraged me in my hardest time in finishing the thesis. She spent hours of time discussing my thesis and experiments with me per week.

Thanks to all professors, researchers, and tutors at Jinan University and the University of Birmingham for their assistance and guidance in academic research and competitions, mathematical modeling contests, imparting knowledge in courses, providing reference letters for me to the foreign universities, and caring in daily lives.

Thanks to my parents who financially and mentally support me in my hard time, without whose accompany I may not successfully finish my four-year studies. Also, I am grateful to my classmates and roommates for the happiness they bring me.

Thanks to the assistance of the UCI machine learning repository and the researchers Dr. Shadi Abpeikar, A/Prof Kathryn Kasmarik, A/Prof Michael Barlow, and Md Mohiuddin Khan in the University of New South Wales, Australia for providing the simulation dataset.

Appendix

Examples of Codes for Implementation:

```
#Importing Data
data = pd.read_csv(r'../Aligned.csv')
a = list(range(0,24015,1))
random.shuffle(a)
index_train = a[0:16000]
index_test = a[16000:]

#Centroid Method
x_train = np.zeros([len(index_train),4])
x_test = np.zeros([len(index_test),4])
for i in range(0,4,1):
    minimum = data.iloc[index_train,i:2400:12].mean(axis = 1).min()
    maximum = data.iloc[index_train,i:2400:12].mean(axis = 1).max()
    x_train[0:len(index_train),i] = (data.iloc[index_train,i:2400:12].mean(axis = 1) - minimum)
/(maximum - minimum)
    x_test[0:len(index_test),i] = (data.iloc[index_test,i:2400:12].mean(axis = 1) - minimum)/(m
aximum - minimum)

y_train = data.iloc[index_train,-1]
y_test= data.iloc[index_test,-1]

#K-Nearest neighbors
task1_score_KNN=pd.DataFrame(data=None,columns=['K','Precision','Recall','F1-
score','Accuracy'])
number=0
for k in range(1,9,1):
```

```
        clf=KNeighborsClassifier(n_neighbors=k)
        clf=clf.fit(x_train,y_train)
        y_predicted=clf.predict(x_test)
        task1_score_KNN.loc[number,'K']=k
        task1_score_KNN.loc[number,'Precision']='%.4f'%precision_score(np.round(y_predicted),y_test)
        task1_score_KNN.loc[number,'Recall']='%.4f'%recall_score(np.round(y_predicted),y_test)
        task1_score_KNN.loc[number,'F1-score']='%.4f'%f1_score(np.round(y_predicted),y_test)
        task1_score_KNN.loc[number,'Accuracy']='%.4f'%accuracy_score(np.round(y_predicted),y_test)
        number+=1

#Random Forest
task1_score_RDF=pd.DataFrame(data=None,columns=['Estimators','Precision','Recall','F1-score','Accuracy'])
number=0
for n in range(20,30,20):
        clf=RandomForestClassifier(n_estimators=n)
        clf.fit(x_train,y_train)
        y_predicted=clf.predict(x_test)
        task1_score_RDF.loc[number,'Estimators']=n
        task1_score_RDF.loc[number,'Precision']='%.4f'%precision_score(np.round(y_predicted),y_test)
        task1_score_RDF.loc[number,'Recall']='%.4f'%recall_score(np.round(y_predicted),y_test)
        task1_score_RDF.loc[number,'F1-score']='%.4f'%f1_score(np.round(y_predicted),y_test)
```

```
        task1_score_RDF.loc[number,'Accuracy']='%.4f'%accuracy_score(np.round(y_predicted
),y_test)
        number+=1

#Gini importance
tick_label = ['$x$','$y$','$xVel$','$yVel$']
plt.figure(dpi = 150)
importance = clf.feature_importances_ * 100
fig, ax = plt.subplots(dpi = 150)
chart = ax.barh(range(len(tick_label)), importance, tick_label= tick_label,color=['#6699CC','
Orange'])
for ite in chart:
    w = ite.get_width()
    ax.text(w, ite.get_y()+ite.get_height()/2, ' %.2f' %(w) + '%', ha='left', va='center')
plt.xlim([0,55])
plt.xlabel('Importance(%)')

#Logistic Regression
task1_score_LR =  pd.DataFrame(data=None,columns=['Penalty','C','Precision','Recall','F1-
score','Accuracy'])
number = 0
for penalty in ['l1','l2']:
    for C in [0.01,0.1,1,10,100,1000,10000]:
        if penalty == 'l2':
            clf = LogisticRegression(C = C,penalty = penalty,max_iter=200,verbose = 0)
        else:
            clf = LogisticRegression(C = C,penalty = 'l1',solver = 'liblinear',max_iter=200,verbose
 = 0)
        clf.fit(x_train,y_train)
        y_predicted = clf.predict(x_test)
```

```
        task1_score_LR.loc[number,'Penalty'] = penalty
        task1_score_LR.loc[number,'C'] = C
        task1_score_LR.loc[number,'Precision'] = '%.4f'%precision_score(np.round(y_predicted)
,y_test)
        task1_score_LR.loc[number,'Recall'] = '%.4f'%recall_score(np.round(y_predicted),y_tes
t)
        task1_score_LR.loc[number,'F1-
score'] = '%.4f'%f1_score(np.round(y_predicted),y_test)
        task1_score_LR.loc[number,'Accuracy'] = '%.4f'%accuracy_score(np.round(y_predicted)
,y_test)
        number += 1

#Multi-layer perceptron
task1_score_MLP =  pd.DataFrame(data=None,columns=['Layer 1','Layer 2','Precision','Recal
l','F1-score','Accuracy'])
number = 0
for i in [8,16,32]:
   for j in [4,8,16]:
        model = tf.keras.Sequential([tf.keras.layers.Dense(i,input_shape = (4 ,), activation = 'relu
'),tf.keras.layers.Dense(j,activation= 'relu'),tf.keras.layers.Dense(1,activation= 'sigmoid')])
        model.compile(optimizer = tf.keras.optimizers.Adam(0.0001),loss = 'binary_crossentrop
y',metrics= ['accuracy'])
        model.fit(x_train,y_train,batch_size = 64,epochs = 100,verbose = 1,validation_data = (x_
test,y_test))
        y_predicted = (model.predict(x_test))
        task1_score_MLP.loc[number,'Layer 1'] = i
        task1_score_MLP.loc[number,'Layer 2'] = j
        task1_score_MLP.loc[number,'Precision'] = '%.4f'%precision_score(np.round(y_predicte
d),y_test)
```

```
        task1_score_MLP.loc[number,'Recall'] = '%.4f'%recall_score(np.round(y_predicted),y_test)
        task1_score_MLP.loc[number,'F1-score'] = '%.4f'%f1_score(np.round(y_predicted),y_test)
        task1_score_MLP.loc[number,'Accuracy'] = '%.4f'%accuracy_score(np.round(y_predicted),y_test)
        number += 1


#2D mapping Method
x_axis = round((data.iloc[:,0:2400:12]+1420)/60)
y_axis = round((data.iloc[:,1:2400:12]+1020)/60)
x_velocity = ((data.iloc[:,2:2600:12])/20)
y_velocity = ((data.iloc[:,3:2600:12])/20)
image = np.zeros([data.shape[0],50,35])
image_x_velocity = np.zeros([data.shape[0],50,35])
image_y_velocity = np.zeros([data.shape[0],50,35])
for number in tqdm(range(0,(data.shape[0]),1)):
    for i in range(0,200,1):
        image[number,int(x_axis.iloc[number,i]), int(y_axis.iloc[number,i]) ] = 1
        image_x_velocity[number,int(x_axis.iloc[number,i]), int(y_axis.iloc[number,i])] = x_velocity.iloc[number,i]
        image_y_velocity[number,int(x_axis.iloc[number,i]), int(y_axis.iloc[number,i])] = y_velocity.iloc[number,i]


x_train = np.zeros([len(index_train),50,35,3])
x_test = np.zeros([len(index_test),50,35,3])
x_train[:,:,:,0] = image[list(index_train),:]
x_train[:,:,:,1] = image_x_velocity[list(index_train),:,:]
x_train[:,:,:,2] = image_y_velocity[list(index_train),:,:]
x_test[:,:,:,0] = image[list(index_test),:]
```

```
x_test[:,:,:,1] = image_x_velocity[list(index_test),:]
x_test[:,:,:,2] = image_y_velocity[list(index_test),:]
y_train = data.iloc[index_train,-1]
y_test= data.iloc[index_test,-1]

#Convolutional Neural Network
task1_score_CNN =  pd.DataFrame(data=None)
number = 0
for kernal_size in [2,3,4,5,6,7]:
        model = tf.keras.models.Sequential()
        model.add(tf.keras.layers.Conv2D(16, (kernal_size, kernal_size), activation='relu', inp
ut_shape=(50,35,3)))
        model.add(tf.keras.layers.MaxPooling2D((2, 2)))
        model.add(tf.keras.layers.Conv2D(8, (kernal_size, kernal_size), activation='relu'))
        model.add(tf.keras.layers.Flatten())
        model.add(tf.keras.layers.Dense(8, activation='relu'))
        model.add(tf.keras.layers.Dense(1, activation='sigmoid'))
        model.compile(optimizer=tf.keras.optimizers.Adam(0.0001),
                      loss='binary_crossentropy',
                     metrics=['accuracy'])
        model.fit(x_train, y_train, epochs=5, batch_size=64,validation_data  = (x_test,y_test),
verbose = 1)
        y_predicted = (model.predict(x_test))
        task1_score_CNN.loc[number,'Kernal Size'] = kernal_size
        task1_score_CNN.loc[number,'Precision'] = '%.4f'%precision_score(np.round(y_predi
cted),y_test)
        task1_score_CNN.loc[number,'Recall'] = '%.4f'%recall_score(np.round(y_predicted),y
_test)
        task1_score_CNN.loc[number,'F1-
score'] = '%.4f'%f1_score(np.round(y_predicted),y_test)
```

```
        task1_score_CNN.loc[number,'Accuracy'] = '%.4f'%accuracy_score(np.round(y_predicted),y_test)
        number += 1

#Permutation importance
x_test = np.zeros([len(index_test),50, 35, 3])
x_test[:,:,:,0] = image[list(index_test),:]
x_test[:,:,:,1] = image_x_velocity[list(index_test),:]
x_test[:,:,:,2] = image_y_velocity[list(index_test),:]
y_test = y_test= data.iloc[index_test,-1]
y_predicted = (model.predict(x_test))
precision_benchmark = precision_score(np.round(y_predicted),y_test)
recall_benchmark = recall_score(np.round(y_predicted),y_test)
f1_benchmark = f1_score(np.round(y_predicted),y_test)
accuracy_benchmark = accuracy_score(np.round(y_predicted),y_test)
task1_importance = []
a = list(range(0,6015,1))
for epoch in tqdm(range(0,1,1)):
    number = 0
    for variable_index in range(0,len(list(data.keys())[3:-1]),1):
        np.random.shuffle(a)
        x_test = np.zeros([len(index_test),50, 35, 3])
        x_test[:,:,:,0] = image[list(index_test),:]
        x_test[:,:,:,1] = image_x_velocity[list(index_test),:]
        x_test[:,:,:,2] = image_y_velocity[list(index_test),:]
        y_test = y_test= data.iloc[index_test,-1]
        for i in range(0,6015,1):
            x_test[i,:,:,variable_index] = data[list(data.keys())[variable_index + 3]][18000+a[i],:]
        y_predicted = (model.predict(x_test))
        task1_importance.append((f1_benchmark-f1_score(np.round(y_predicted),y_test)))
```

```
        number += 1
task1_importance_normalized = np.array(task1_importance)/sum(task1_importance)
tick_label = ['$Coordinate$','$xVel$','$yVel$']
plt.figure(dpi = 150)
fig, ax = plt.subplots(dpi = 150)
chart = ax.barh(range(len(tick_label)), task1_importance_normalized * 100, tick_label= tick_l
abel,color=['#6699CC','Orange'])
for ite in chart:
    w = ite.get_width()
    print(w)
    ax.text(w, ite.get_y()+ite.get_height()/2, ' %.2f' %(w) + '%', ha='left', va='center')
plt.xlim([0,100])
plt.xlabel('Importance(%)')
```

References

[1] Acot Pascal, Blandin Patrick, Hamm BP, Drouin Jean-Marc, Blondel-Megrelis Marika, Matagne Patrick, Bergandi Donato, Muller Gerhard, & Pierre-Francois Verhulst, 2021, Recherches mathematiques sur la loi d'accroissement de la population, Nouveaux Memoires de l'Academie Royale des Sciences et Belles-Lettres de Bruxelles, Brussels, The European Origins of Scientific Ecology, Routledge, Volume 18, pp. 245-283.

[2] Alwyn V Husselmann & Ken Hawick, 2011, Simulating species interactions and complex emergence in multiple flocks of boids with gpus, Proc. IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 2011), pp. 100-107.

[3] Anil K. Jain, Jianchang Mao, & K.M. Mohiuddin, 1996, Artificial neural networks: A tutorial, Computer, IEEE, Volume 29, pp. 31-44.

[4] Christopher Hartman & Bedrich Benes,2006, Autonomous boids, Computer Animation and Virtual Worlds, Wiley Online Library, Volume 17, pp. 199-206.

[5] Corinna Cortes & Vladimir Vapnik,1995, Support-vector networks, Machine learning, Springer, Vol. 20, pp. 273-297.

[6] Craig W. Reynolds, 1987, Flocks, herds and schools: A distributed behavioral model, Proceedings of the 14th annual conference on Computer graphics and interactive techniques, pp. 25-34.

[7] Dheeru Dua & Casey Graff, 2017, {UCI} Machine Learning Repository, University of California, Irvine, School of Information and Computer Sciences, url: https://archive.ics.uci.edu/ml/datasets/Swarm+Behaviour.

[8] Emlen & John T,1952, Flocking behavior in birds, The Auk, JSTOR, Volume 69, pp. 160-170.

[9] J. Ross Quinlan, 1986, Induction of decision trees. Machine learning, Springer, Volume 1, pp.81-106.

[10] J. Ross Quinlan, 2014, C4. 5: programs for machine learning, Elsevier.

[11] Keiron O'Shea & Ryan Nash, 2015, An introduction to convolutional neural networks, arXiv preprint arXiv:1511.08458.

[12] Lauren Parker, James Butterworth & Shan Luo, 2019, Fly safe: Aerial swarm robotics using force field particle swarm optimization, arXiv preprint arXiv:1907.07647.

[13] Leo Breiman, 2001, Random forests, Machine learning, Springer, Volume 45, pp.5-32.

[14] Liu Yang, & Kevin M. Passino,2000, Swarm intelligence: Literature overview, Department of electrical engineering, the Ohio State University.

[15] Roger J. Lewis, 2000, An introduction to classification and regression tree (CART) analysis, Annual meeting of the society for academic emergency medicine in San Francisco, California, Citeseer, Volume 14.

[16] Rosenblatt Frank, 1958, The perceptron: a probabilistic model for information storage and organization in the brain, Psychological review, American Psychological Association, Volume 65, pp. 386.

[17] Saad Albawi, Oguz Bayat, Saad Al-Azawi, & Osman N. Ucan, 2017, Understanding of a convolutional neural network, 2017 international conference on engineering and technology (ICET), IEEE, pp.1-6.

[18] Saleh Alaliyat, Harald Yndestad, & Filippo Sanfilippo, 2014, Optimisation Of Boids Swarm Model Based on Genetic Algorithm And Particle Swarm Optimisation Algorithm (Comparative Study), ECMS, pp. 643-650.

[19] Soon-Jo Chung, Aditya Paranjape, Philip Dames, Shaojie Shen, & Vijay Kumar,2018, A survey on aerial swarm robotics, IEEE Transactions on Robotics, IEEE, Volume 34, pp.837-855.

[20] Thomas Cover & Peter Hart, 1967, Nearest neighbor pattern classification, IEEE transactions on information theory, IEEE, Vol.13, pp. 21-27.

[21] Wright Raymond E, 1995, Logistic regression, American Psychological Association.

Graduation Thesis Evaluation for Undergraduate Students

Supervisor's Comments： Marks：_____/100 Signature： Date (dd/mm/yyyy):
Evaluator's Comments: Marks：_____/100 Signature： Date (dd/mm/yyyy):
Dean's Comments: Marks: _____/100 Signature： Date (dd/mm/yyyy):