Engineering Surrogate Models for Boid Systems

Jan von Pichowski and Sebastian von Mammen

Julius-Maximilians-Universität, Würzburg, BY 97070, GER

Abstract

We examine the learnability of emergent flocking behavior in boid simulations. To this end, we present (1) a detailed definition of the boid model, (2) a formulation such that model instances can be simulated efficiently, (3) metrics for training surrogate models, (4) and an evaluation of early training results. For this proof of concept, we focus on simple architectures like multi-layer perceptrons and graph neural networks. The performance of these models is comparable to simulations with an absolute error in the boid state of 5% in varying scenarios with varying interaction patterns and even surpasses the erroneous simulations for the prediction of formed flocks. By splitting the prediction task into a boid adjacency detection and a rule-application task, we observe that wrong interactions between boids only have a minor impact on the prediction results. Besides evaluating more complex models, we suggest focusing on either the detection of stable emergent states to predict them separately or on the understanding of dynamic transitions of groups that show emergent behavior.

Introduction

Reynolds (1987) introduced boid flocks to retrace the complex collective movement patterns of flocks of birds. Boid agents move in accordance with their neighbors, and thereby continuously change their neighborhoods. As a result, boid flocks represent dynamic systems with a dynamic structure (Giavitto (2003)) and are, therefore, not only hard to predict but also challenging to compute. For the computation, the repeated identification of potentially ever-changing neighbors poses the bottleneck. Lee et al. (2009) presented a variant of the algorithm that adjusts the neighborhood topology only when required. Davison et al. (2019) integrated various approaches to arrive at a fast solution, including keeping transform data in the CPU cache, laying out the agents sequentially in memory, and distributed neighbor search on the GPU. Yet, the prediction of dynamics and the efficient computation of behaviors remain major scientific challenges.

In this paper, we present tooling and methods to study the direct prediction of boid flocking behavior without the need for calculating all individual interactions. Instead, we train surrogate models, as surveyed by van der Hoog (2019), of Reynolds' boid model. In the next section, we briefly outline some related works. Next, we provide a concise boid model, closely inspired by Reynolds' original publication but adapted to support our research goal. We justify the choice of parameters that leads us to demanding scenarios. We detail the model in matrix notation to harness the computational power of matrix multiplication units (MMUs) as, for instance, provided in graphical processing units (GPUs). For a sound analysis, we introduce new metrics to assess the quality of the learned flocking behavior. We not only focus on the global deviations of a single boid but also measure its local state in relation to its neighbors. To gain some measure for comparison, we run these metrics for a baseline simulation, and test the results against runs with errors systematically introduced at the integration steps. Next, we present our approach to learning stepwise predictions of the simulation state. We investigate the impact of the boids' interactions compared to the importance of the flocking rules by splitting the prediction into a stage that only predicts the boids' adjacency and another stage that only learns the movement behavior. Lastly, we compare the learnability of a simulation with randomly initialized boids to a simulation with coherently configured boids that follow a clustered flight formation dynamics. This comparison show, that the models perform better in learning the flocking behavior than the potentially more chaotic behavior from a random initialization.

Related Work

Learning agent behavior in various forms is a common topic in the scientific community. For example, Jiahao et al. (2022) present an approach to learn a controller for a single boid agent based on observations of a flock. They do so by training an artificial neural network that directs the individual considering the local neighbors as well as obstacles to avoid. Similarly, Powers et al. (2022) deduce the symbolic rules for individual boids. Both works focus on learning agent-centric descriptions that can be included in the simulation. In contrast, we focus on learning a surrogate model that can control the whole flock.

Angione et al. (2022) follow an approach similar to ours by comparing different machine learning models deployed to learn the 'Linked Lives' model of social care provision in the UK (Noble et al., 2012). It is similar since it also aims at a surrogate model learned from a complex system. However, it is different with respect to the model domain, the characteristics of the concrete agent-based model, as well as the authors' focus on testing and comparing different machine learning techniques. We focus on boid flocks due to their potentially great dynamics arising from self-referential interactions (von Mammen and Jacob, 2008) and explore whether and to which extent it is feasible to capture these dynamics by means of a learned surrogate model.

While we consider the contributions of this paper only an early step in the quest for learning surrogate models of complex boid flocks, they might pave the way for reaching a greater goal. von Mammen et al. (2011), for instance, propose an approach to replace subsets of agents in large model spaces whose group behaviors can be well predicted. Similar to our motivation outlined above, they argue that such locally deployed surrogate models could reduce the computational complexity.

Another way to reduce the algorithmic complexity is by reusing information. Reynolds (1987) suggest to sort nearby boids into constant sized bins. This dynamic spatial ordering and subsequent partitioning approach is implemented by Klein and Spector (2009). It reduces the complexity to O(n) if the boids do not change their bins. Lee et al. (2009) improved this approach by using the kNN algorithm to determine a more stable bin assignments. At a later point, similar techniques of reusing information are worth to be considered for integrated into our approach.

Boids System Definition

The implemented simulation contains a finite set B of $n = |B|$ boids, which are located on a finite xy-plane of size $s = 1$. We obtain continuous infinite movement by wrapping the space and teleporting agents from one side to another. Likewise, the neighborhood of the boids expands over the border to the other side of the plane. This reduction of space imposes bounds on the inputs for the learned models.

Each boid $b_i \in B$ has a position on the plane given by $x_i \in [0, 1)$ and $y_i \in [0, 1)$. The velocity of the boids is limited by a maximum value m_v. Reynolds (1987) considers this a simplified implementation of body-drag forces which stop a bird from becoming infinitely fast. This variable is chosen freely by Reynolds (1987) to obtain a realistically looking animation. We choose the value $m_v = s/n$ carefully such that some boids cluster in the restricted space and give, at the same time, enough space to other boids to fly independently. The separation radius, also referred to as the protection radius, is defined as $r_p = m_v$. It is the maximum distance another boid can travel in one time step and, thus, to hit the boid trying to avoid the collision. Boids further away cannot hit the given boid, and are therefore not considered a threat. The alignment and cohesion radius, generally referred to as the view radius, is chosen as $r_v = 3 \cdot r_p$ because it results in a view area with space for roughly one order of magnitude more boids than in the protection area. Reynolds (1987) suggest to limit the view and protection area to model the at around 300 degrees limited birds field of perception. We follow his example and define the two-sided view angle with $\alpha_v = 0.8\pi$. The outlined parametric relationships render it sufficiently likely that boid agents meet but that the whole space does not automatically coincide into one cluster. Since scaling, especially in combination with different integration schemes, can easily lead to quite vast differences in observable emergent effects, the outlined deliberations should also help to consistently lead to traceable local interactions and consistent global results.

Rules

In the following, we describe the three boid rules formulated by Reynolds (1987). These rules form the core of the simulation. At each time step $t \in \mathbb{N}$, each rule generates an influence unit vector for a given boid. The influences, sometimes also referred to as urges, are combined into a weighted sum that is interpreted as an accelerating force

$$\vec{f}_t(b_i) = w_s \vec{s}_t(b_i) + w_a \vec{a}_t(b_i) + w_c \vec{c}_t(b_i) \qquad (1)$$

acting on the given boid $b_i \in B$. Reynolds chooses the weights $w_s \in \mathbb{R}^+, w_a \in \mathbb{R}^+$ and $w_c \in \mathbb{R}^+$ freely to obtain realistically looking results and afterward clamps the value by a maximal force value m_f. In Reynolds' implementation, the maximum force is defined to be one order of magnitude lower than the maximum speed, i.e. $m_f = \frac{1}{10} m_v$. We adopted this specification in our implementation accordingly. This is a reasonable choice since it prevents the boid from abruptly changing its direction and but it retains the impact of the momentum of inertia. The dynamics are implemented using explicit Euler integration with a delta of 1.

The rules consistently implement the the following structure. For each influence, a vector is calculated between the given boid and each of its neighbors and totaled across the whole neighborhood. The totaled vector is then normalized and weighted in accordance with the influence. The alignment rule calculates the mean forward direction of all boids in the view area of the given boid, defined by r_v and α_v. The given boid's forward direction is subtracted, the resulting vector normalized and used to calculate the resulting force as outlined in eqn. 1. The cohesion rule calculates the mean position of all boids in the view area and subtracts the position of the given boid. The separation rule calculates the mean weighted offset of all boids in the protection area of the given boid, defined by r_p and α_v. The offsets are weighted by their negative inverse squared length which leads to high weights for small offsets with inverted direction. The resulting, repelling vector is normalized and fed

into eqn. 1. The outlined rules form the core of the simulation. In the next section, we express these rules in terms of matrix operations to obtain a mathematical description well-suited for efficient processing.

Boid Model in Matrix Notation

Data availability or generation is a crucial requirement for any inductive learning task. As a result, it is important to have the boid simulations run efficiently. In this way, we can ensure the creation of sufficiently large training sets, and even generate training data and efficiently calculate the learning model's efficiency during an active learning phase. Overall, an efficient simulation improves the overall iteration time, and thus the outcome of the training cycles.

Instead of relying on algorithmic enhancements to the computations, as in the originally proposed spatial partitioning or kNN-based methods (Lee et al., 2009),

we propose a matrix formulation of the boid model so it can be executed on efficient MMUs alongside the learned model.

Foundation

In the following, element-wise operations on matrices and vectors are required. We use the Hadamard product (Styan, 1973) and division defined as

$$(A \odot B)_{ij} := (A)_{ij}(B)_{ij} \qquad (A \oslash B)_{ij} := \frac{(A)_{ij}}{(B)_{ij}} \tag{2}$$

and expand the definition from matrices to vectors with the following notation

$$\vec{a} \odot \vec{b} := \mathrm{diag}(\vec{a})\vec{b} \tag{3}$$

$$\vec{a} \oslash \vec{b} := \mathrm{diag}(\vec{a})\vec{b}^{-1}, \text{ with } \vec{b}^{-1} = \left(\frac{1}{b_1}, \frac{1}{b_2}, \ldots, \frac{1}{b_n}\right)^T . \tag{4}$$

Additionally, we introduce the row sum mapping $\Sigma_r : \mathbb{R}^{n\times m} \to \mathbb{R}^n$ as

$$\Sigma_r(M) : = \sum_j^n ((M)_{0,j}, (M)_{1,j}, \ldots, (M)_{n,j})^T \tag{5}$$

to sum up the entries of each matrix row in a vector. Next, we define the vector to matrix expansion $M_e : \mathbb{R}^n \to \mathbb{R}^{n\times n}$ as

$$M_e(\vec{v}) := (\underbrace{\vec{v}, \vec{v}, \ldots, \vec{v}}_{n}), \quad \text{with } \vec{v} \in \mathbb{R}^n . \tag{6}$$

State Description

First, we define four vectors containing the positions and velocities of all boids $b_1, b_2, \ldots, b_n \in B$ with

$$\vec{x} = (x_{b_1}, x_{b_2}, \ldots, x_{b_n})^T \tag{7}$$

$$\vec{y} = (y_{b_1}, y_{b_2}, \ldots, y_{b_n})^T \tag{8}$$

$$d\vec{x} = (dx_{b_1}, dx_{b_2}, \ldots, dx_{b_n})^T \tag{9}$$

$$d\vec{y} = (dy_{b_1}, dy_{b_2}, \ldots, dy_{b_n})^T . \tag{10}$$

Those vectors are arranged in position and velocity matrices with

$$X = M_e(\vec{x}) \qquad Y = M_e(\vec{y}) \tag{11}$$

$$dX = M_e(d\vec{x}) \qquad dY = M_e(d\vec{y}) . \tag{12}$$

The offset between to boids b_i and b_j is calculated as $(\bar{X})_{b_i b_j} = (X)_{b_j b_i} - (X)_{b_i b_j}$ with

$$\bar{X} = X^T - X \qquad \bar{Y} = Y^T - Y . \tag{13}$$

Additional care needs to be taken in the implementation of near-border situations. When one boid is on one side of the area and the other boid is on the opposite side then the offset needs to be calculated around the border and not through the area. This can be achieved with a shift by the size of the area and a masking operation that selects the variant where the two boids are nearest to each other. We omit this implementation detail in the following equations (especially in the equations for the adjacency matrices) for simplicity and because it does not add anything substantial to the general definitions.

The distance matrix D and the matrix $\hat{D}$ containing the squared distances between the boids are defined as

$$\hat{D} = D \odot D = \bar{X} \odot \bar{X} + \bar{Y} \odot \bar{Y} . \tag{14}$$

The total velocity $\vec{v}$ of the boids is defined as

$$\vec{v} \odot \vec{v} = d\vec{x} \odot d\vec{x} + d\vec{y} \odot d\vec{y} . \tag{15}$$

It is used to obtain the normalized velocity along the coordinate axes that is called the forward vector

$$f\vec{x} = d\vec{x} \oslash \vec{v} \qquad f\vec{y} = d\vec{y} \oslash \vec{v} . \tag{16}$$

This leads to the forward matrices

$$fX = M_e(f\vec{x}) \qquad fY = M_e(f\vec{y}) . \tag{17}$$

The case of zero velocity needs to be handled carefully when implementing these operations. Here, the forward vector is then best set to zero, too.

Modeling Interactions as Directed Adjacency Matrices

After these basic definitions, we transfer the concept of directed adjacency matrices to the boids model. We differentiate between the adjacencies for the protected area and the adjacencies for the full view area. For the protected area, a boid $b_j \in B$ is adjacent to $b_i \in B$, if the distance between b_j and b_i is smaller or equal to the separation radius r_p. The same holds for the view area, but the limiting distance is the view radius r_v.

To formalize the given description, we introduce the adjacency matrix $A_p \in \{0,1\}^{n\times n}$ for the protected area as

$$(A_p)_{b_i b_j} = \begin{cases} 1 & \text{if } b_i \neq b_j \wedge (\hat{D})_{b_i b_j} \leq r_p^2 \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

Note that in the implementation we included the limited field of perception by calculating the difference in the view directions following the same approach that leads to the offset matrices $\bar{X}$ and $\bar{Y}$. We omitted this detail in the equations for simplicity. Additionally, the adjacency $A_v \in \{0,1\}^{n\times n}$ describes the interactions in the view area with

$$(A_v)_{b_i b_j} = \begin{cases} 1 & \text{if } b_i \neq b_j \wedge (\hat{D})_{b_i b_j} \leq r_v^2 \\ 0 & \text{otherwise} \end{cases} . \tag{19}$$

The vector $\vec{a}_v$ counting the numbers of interactions in the view area is obtained by using the row sum mapping

$$\vec{a}_v = \Sigma_r(A_v)^T . \tag{20}$$

Boid Rules Reformulation

Using the obtained matrices we reformulate the three boids rules. The matrices S_x and S_y contain the separation values for each boid with respect to every other one:

$$S_x = \bar{X} \oslash (-\hat{D}) \odot A_p \quad S_y = \bar{Y} \oslash (-\hat{D}) \odot A_p \tag{21}$$

Using the row sum mapping leads to the unnormalized separation value for every boid:

$$\vec{s}'_x = \Sigma_r(S_x) \qquad \vec{s}'_y = \Sigma_r(S_y) \tag{22}$$

Using the length vector $\vec{s}$ of the separation vectors, we obtain the separation value for every boid for each dimension:

$$\vec{s} \odot \vec{s} = \vec{s}'_x \odot \vec{s}'_x + \vec{s}'_y \odot \vec{s}'_y \tag{23}$$

$$\vec{s}_x = \vec{s}'_x \oslash \vec{s} \qquad \vec{s}_y = \vec{s}'_y \oslash \vec{s} \tag{24}$$

The alignment rule is similar to the separation rule reformulated as

$$\vec{a}_x = \Sigma_r(fX^T \odot A_v) \oslash \vec{a}_v - f\vec{x} \tag{25}$$

$$\vec{a}_y = \Sigma_r(fY^T \odot A_v) \oslash \vec{a}_v - f\vec{y}. \tag{26}$$

Note that we calculate the mean value by dividing through the view sum $\vec{a}_v$. In the implementation, it is a valid choice to set the values to zero, if a division by zero occurs for special cases like when no boid is in the view area.

Lastly, the cohesion rule is based on the position matrices

$$\vec{c}'_x = \Sigma_r(X^T \odot A_v) \oslash \vec{a}_v - \vec{x} \tag{27}$$

$$\vec{c}'_y = \Sigma_r(Y^T \odot A_v) \oslash \vec{a}_v - \vec{y} \tag{28}$$

that is again normalized to

$$\vec{c} \odot \vec{c} = \vec{c}'_x \odot \vec{c}'_x + \vec{c}'_y \odot \vec{c}'_y \tag{29}$$

$$\vec{c}_x = \vec{c}'_x \oslash \vec{c} \qquad \vec{c}_y = \vec{c}'_y \oslash \vec{c}. \tag{30}$$

Error Measures for Evaluating Interactive Agents

In general, a surrogate simulation model receives the description of a given simulation state to yield a subsequent one. Comparing the obtained state with the results of the baseline simulation allow one to assess the performance of the learned model. In this section, we provide an overview of commonly used approaches and we introduce an improved metric specifically tailored to capture the relevant properties of boid flocks.

The most immediate notion to assess the surrogate's performance is by comparing the agents' absolute positions from predicted and baseline data. The velocity can be ignored, since it is an internal parameter of the boid and not considered part of the simulation state. Accordingly, Angione et al. (2022) use the mean square error (MSE) for evaluating the prediction of agents. There is a discussion by Willmott and Matsuura (2005) whether the mean-absolute error (MAE) is more appropriate. Similarly, a combination of multiple measures might be used as proposed by Chai and Draxler (2014). All these measures are similar as they compare the absolute positions of the boids and only differ in their sensitivity to outliers in the data set.

Regarding the latter, we decided to build on the root-mean-square error RMSE as it has been widely adopted specifically as it exhibits greater sensitivity to outliers and considering the approximating nature of the surrogate model we are aiming at, we prefer many small errors over a few large ones. Also, the loss function used to train the model can be designed to optimize the model with respect to this error measure.

We adapt the RMSE measure to make it comparable to the metric we will introduce shortly to capture the quality of flock predictions. By normalizing the RMSE error by the maximal possible distance between two boids, we obtain the global error

$$e_g = \sqrt{\sum_{b_i \in B} \frac{e_g(b_i, \hat{b}_i)^2}{|B|}}, \tag{31}$$

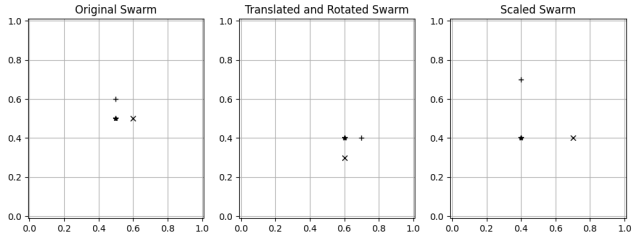

Figure 1: Three boid configurations for comparing the metrics. In the middle figure, the boids are moved on each axis by one unit and rotated by one quarter. In the right figure, the boids are separated by one unit from their original positions.

with

$$e_g(b_i, \hat{b}_i) = \frac{||\vec{x}(b_i) - \vec{x}(\hat{b}_i)||}{\sqrt{2\left(\frac{s}{2}\right)^2}}. \tag{32}$$

where $\vec{x}(b_i)$ refers to the predicted position of boid b_i whereas $\vec{x}(\hat{b}_i)$ refers to the position of the same boid in the simulated state.

We refer to this error measure as *global* because it compares the global position of the predicted boids to their simulated counterparts. In the following, we develop a *local* error measure e_l to estimate the error of the boid position with respect to its local environment. Fig. 1 shows a set of three boids that is firstly translated and rotated, and secondly scaled. From the figure, we can infer that the rotation and translation are operations that yield invariant results with respect to the local organization of the flock, i.e. the relative positions and orientations remain the same and the boids' interactions are not affected. Scaling, however, results in a different structure because some boids might not interact any longer due to their limited view area. Consequently, a local error measure should be insensitive to the invariant translation and rotation operations.

We construct the local error e_l in eqn. 34 from the distances to the local neighbors of the given boid. The relevant neighbors are restricted to those boids that are present in the view area. Additionally, we scale the error similarly to the global error measure to obtain comparable error measures. The local error is defined as

$$e_l = \sqrt{\sum_{b_i \in B} \frac{e_l(b_i, \hat{b}_i)^2}{|B|}} \tag{33}$$

with

$$e_l(b_i, \hat{b}_i) = \sum_{b_j \in B} \frac{(\hat{A}_v)_{b_i b_j} \left| ||\vec{x}(b_j) - \vec{x}(b_i)|| - ||\vec{x}(\hat{b}_j) - \vec{x}(\hat{b}_i)|| \right|}{\sum_{b_k \in B} (\hat{A}_v)_{b_i b_k} \sqrt{2\left(\frac{s}{2}\right)^2}} \tag{34}$$

Error	Translated & Rotated Flock	Scaled Flock
e_c	0.08363	0.52290
β	0.16095	0.16095
e_l	0.00000	0.32190
e_g	0.29428	0.20000
RMSE	0.04667	0.02000

Table 1: Error measures applied to the examples in fig. 1. The local error is only sensitive to scaling whereas the global error also considers translation and rotation. RMSE behaves similarly to the global error.

e_g and e_l capture the global, absolute positions as well as the local, relative positions of the boids, respectively. We weight and total these measures to combine them in eqn. 36. The blending factor β describes which error measure is more relevant for a given flocking scenario. For example, if there are no interactions between boids then the global error is more relevant. In contrast, if all boids interact as a single flock, then the local error is more relevant. The β value is the normalized average distance between a given boid and all other ones.

$$\beta(\hat{b}_i) = \frac{1}{|\hat{B}| - 1} \sum_{\hat{b}_j \in \hat{B}} \frac{||\vec{x}(\hat{b}_j) - \vec{x}(\hat{b}_i)||}{\sqrt{2\left(\frac{s}{2}\right)^2}} \tag{35}$$

A high β value decodes that the given boid is far away from other boids whereas a low β value means that it is surrounded by other boids.

The local and global error and the β value lead to the combined error measure

$$e_c(b_i, \hat{b}_i) = \beta(\hat{b}_i) e_g(b_i, \hat{b}_i) + (1 - \beta(\hat{b}_i)) e_l(b_i, \hat{b}_i). \tag{36}$$

The RMSE is used to obtain a single error measurement for the whole predicted state that is sensitive to outliers.

$$e_c = \sqrt{\sum_{b_i \in B} \frac{e_c(b_i, \hat{b}_i)^2}{|B|}} \tag{37}$$

Table 1 provides an overview of the impact of the types of transformations on boids as shown in fig. 1. It serves as a reference for comparing the obtained results.

Learnable Boids Models

In order to gain an insight in the learning success and the effectiveness of the surrogate models, we compare the predictions of different surrogate models with erroneous simulation runs. We arrive at systematically erroneous simulation runs by introducing a small error to the velocities and the positions of the boids at each simulation step. Additionally, entries in the adjacency matrix are flipped randomly. The errors are normally distributed and the standard deviation σ of

the error refers to the requested fraction e of the maximum value for the given property. For example, an error value of $e = 0.01$ means that 68% of all velocity errors are smaller than $0.01 \cdot m_v$. For reference, we use three models *RND-0.01*, *RND-0.05*, *RND-0.10* with $e = 0.01$, $e = 0.05$ and $e = 0.10$, respectively.

With respect to the learning task itself, we implement and compare four different approaches. First, we directly learn to predict a future state from a given one by means of a multi-layer perceptron (MLP) artificial neural network. We refer to this end-to-end surrogate model by means of MLP as *E2E-MLP*. It consists of two hidden layers with 300 features. Second, we fully calculate the adjacency matrix (following eqns. 18 to 20) and only replace the integration of the boids' movement, i.e. the application of the boid rules, by means of a graph neural network (GNN). We refer to this rule-integration model as *RULES-GNN*. The GNN architecture based on convolutions by Kipf and Welling (2016a) is a natural choice since it uses the inherent graph structure of the boid flocks and the message-passing approach leads to a similar information flow compared to the application of the boid rules. The *RULES-GNN* model consists of two hidden graph convolution layers with 300 features. Third, we train another MLP to only determine the adjacency matrix and integrate their movement as described in eqns. 21 to 30. We refer to this surrogate model as *ADJ-MLP*. Again, it consists of two hidden layers with 300 nodes to keep the complexity of the models comparable. Kipf and Welling (2016b) present an auto-encoder architecture for predicting links in graphs. We evaluated this as an improved approach for detecting interactions between boids but were not able to obtain non-degeneration, converging results therefore excluded it from our evaluation. Nevertheless, it is a reasonable choice since it allows to design a model which is more tailored towards the task of detecting boid interactions and additionally could lead to a sparse approach that reuses the matrix of the previous step. Fourth, we combine the ADJ-MLP and the RULES-GNN model to arrive at a second end-to-end surrogate model that is independent of any priors or predefined simulation logic. We refer to this surrogate model as *MLP-GNN*. It is trained by combining both loss functions.

Experiments

We conduct three experiments to evaluate the complexity of the learning task, the learning capability of the described models with respect to a broad sampling of flock configurations, and, reducing the learning task's complexity, their capability with respect to already formed flocks.

The first experiment provides an insight in how strongly the boid simulation diverges based on cumulative errors, also in comparison with a completely randomly predicted state. To this end, we sample 100 initial conditions with random boid state (velocity and position). Next, we obtain predictions for 10 steps from the erroneous simulations RND-0.01, RND-0.05, and RND-0.10. For each step, we determine the cumulative error by comparing the predictions to a correct simulation. The boid prediction errors are measured in terms of the average combined error over all samples to include local and global influences. Lastly, we measure how much a random state would deviate from the simulation, to understand at what point a prediction cannot be distinguished from a random guess anymore.

The second experiment is about evaluating the proposed surrogate models with respect to learning the boids' behavior from random boid states. The number of boids is fixed to $n = 20$ and the influences are weighted equally with $w_c = w_a = w_s = m_f$. Due to the initial random distribution of boids with different positions, speeds, and directions, we expect varying interaction patterns of the boids during simulation. This imposes a strong challenge on training the models. The dataset consist of 800 samples of an initial simulation state, where the positions and velocities of the boids are randomly generated. For the 800 samples 10 steps are simulated and included into the dataset. This leads to a dataset with 8000 samples consisting of an input state and the next state as the response. We use a 60-20-20 split and repeat the experiment 10 times with different seeds to obtain sound results.

In the last experiment, we evaluate the proposed models with respect to learning the behavior of boids that have already formed a specific type of flocking behavior. With a strong tendency to collectively move in one direction and not split, this specific learning task drastically reduces the space of potentially emerging interaction patterns. At the same time, investigating the prediction of flocks that have already transitioned into a specific flocking pattern may provide a starting point to divide the learning task into easier sub-tasks. A flock consists of $n = 21$ boids that are initially located near each other are heading towards a common random direction. We randomly varied the initial positions of the boids by at most $2r_v$ units. As a consequence, one randomly located and randomly shaped cluster emerges at the beginning of the simulation. Despite the different initial conditions, the dataset is prepared and the experiment conducted in the same way as for the second experiment. This includes the weights of the boids' influences such that we can compare the results to the second experiment.

Results and Discussion

Fig. 2 shows the results of the first experiment, i.e. the mean combined error of the three erroneous simulation runs RND-0.01, RND-0.05, and RND-0.10 compared with the correct simulation over 10 steps. After ten steps with $e = 0.10$, the simulation result cannot be distinguished from a completely random state. The curve gets steeper with a higher error and shows the chaotic nature of the error propagation.

Tables 2 and 3 contain the results of the second and third

Model	e_c	e_l	e_g	RMSE
RND-0.01	0.16453 ± 0.00170	0.01766 ± 0.00010	0.05334 ± 0.00072	0.02120 ± 0.00052
RND-0.05	0.67418 ± 0.00621	0.07772 ± 0.00032	0.21375 ± 0.00252	0.08698 ± 0.00186
RND-0.10	1.16898 ± 0.00571	0.15310 ± 0.00071	0.35406 ± 0.00247	0.14705 ± 0.00196
E2E-MLP	0.86254 ± 0.05090	0.14890 ± 0.01069	0.22896 ± 0.01228	0.04430 ± 0.00304
RULES-GNN	2.04911 ± 0.20352	0.10315 ± 0.00062	0.76927 ± 0.08515	0.36403 ± 0.07696
ADJ-MLP	0.04753 ± 0.00075	0.00546 ± 0.00003	0.01512 ± 0.00035	0.00614 ± 0.00024
MLP-GNN	2.78383 ± 0.04443	0.10381 ± 0.00164	1.07776 ± 0.01842	0.66324 ± 0.01916

Table 2: The table contains the results of the second experiment about predicting the state of boids with varying interaction patterns. It present the commonly used RMSE value for the models and the newly introduced error measures. A better e_l measure compared to the e_g measure implies that the model is able to learn the inter-flock relations between boids whereas a lower e_g indicates that the global positions of the individual boids are more likely correctly predicted. The e_c measure compares the overall performance of the different models, considering both local relations and global positions. The table shows in which areas the surrogate models perform better than the baselines RND-0.01, RND-0.05, or RND-0.10.

Model	e_c	e_l	e_g	RMSE
RND-0.01	0.09087 ± 0.00033	0.01944 ± 0.00008	0.02230 ± 0.00008	0.00040 ± 0.00003
RND-0.05	0.43370 ± 0.00143	0.09302 ± 0.00033	0.10309 ± 0.00028	0.00712 ± 0.00012
RND-0.10	0.88238 ± 0.00268	0.19168 ± 0.00063	0.19277 ± 0.00054	0.02514 ± 0.00033
E2E-MLP	0.33003 ± 0.02485	0.06303 ± 0.00573	0.12270 ± 0.00492	0.00972 ± 0.00076
RULES-GNN	0.80868 ± 0.21963	0.09253 ± 0.01722	0.74013 ± 0.28762	0.32824 ± 0.18751
ADJ-MLP	0.01233 ± 0.00022	0.00253 ± 0.00005	0.00364 ± 0.00007	0.00005 ± 0.00004
MLP-GNN	0.87224 ± 0.21722	0.09797 ± 0.00910	0.80032 ± 0.30478	0.37660 ± 0.20394

Table 3: The table contains the results of the third experiment about predicting the state of boids in a stable flock. The mean error rates of the step-wise predictions by the models are stated in comparison to the erroneous simulations. RMSE and e_c provide an overview about the overall performance whereas e_l and e_g give an insight in the prediction capabilities of the individual boids or the boids in relation to their flocks.

experiment, respectively. For these experiments, in addition to the step-wise errors, we measure the accuracy of the entries in the predicted adjacency matrices at each step and calculate the mean and standard deviation over the 10 experiment runs. A low accuracy either implies that boids outside a given view area influence the respective agent, or that boids inside its view area do not. The accuracy of the adjacency prediction in the second vs. the third experiment is for ADJ-MLP 0.69871 ± 0.01664 vs. 0.77998 ± 0.00134 and for MLP-GNN 0.66007 ± 0.03552 vs. 0.77703 ± 0.00665, respectively. Accordingly, better adjacency predictions are achieved in the more specific, third experiment, consistently by both surrogate models. The ADJ-MLP that only learns the adjacency prediction achieves better result in this domain than the MLP-GNN, which additionally learns the rule prediction and, thus, is tailored towards a twofold objective.

Considering the values presented in the tables, the performances reported on a specific flocking formation (table 3) are better than the ones for the generic case (table 2). We expected this due to the reduced complexity of the learning task, which is confirmed also by the generally lower error rates introduced by the RND models. This observation is only crossed by e_l which suggests that the local relations exhibited in the investigated flocking type is greater than for the generic case which is sampled across 800 different random initializations.

The RULES-GNN presents the ability of a GNN to learn the boids rules, with the correct adjacencies as prior. The results show that it's not able to learn the global properties. Jiahao et al. (2022) state the inherent problem of GNN-based models is that a given boid can only access the diffused state information of other boids because the information is repeatedly multiplied by the convolution operator. In contrast, the MLP-based model handles all boids at the same time even though they might not interact. The E2E-MLP performs better than the MLP-GNN combination which could be attributed to the weaker performance of the GNN with respect to global property predictions combined with the errors in the predicted adjacencies.

All in all, the E2E-MLP is slightly worse in the second experiment than the RND-0.05 simulation because it produces a higher local error than the erroneous simulation. Fig. 3 confirms this observation as flocks form but partly with different members. For the prediction of the flock in the third

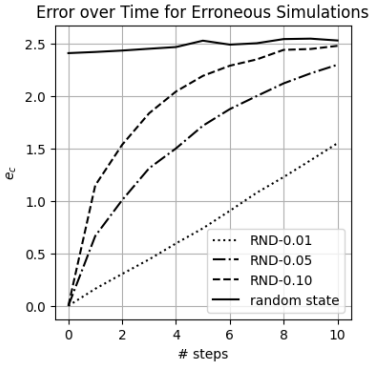

Figure 2: This figure shows the effect of cumulative errors on the simulation obtained in the first experiment. The combined prediction error for each step of the erroneous simulations RND-0.01, RND-0.05, and RND-0.10 is compared to the combined error of a randomly generated simulation state.

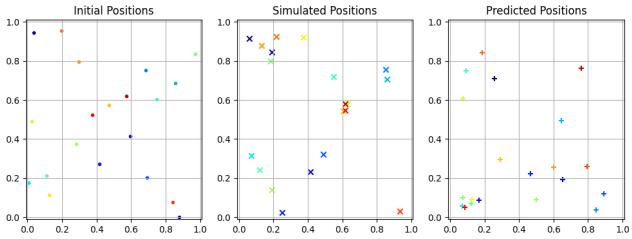

Figure 3: This figure shows an exemplary prediction from the second experiment of the E2E-MLP model after 10 steps compared to the initial state and the simulated state after 10 steps. Only those individuals within $r_v = 0.15$ units in the graph can interact and form flocks.

experiment, we observe the same patterns as in the results for experiment two, but this time the predictions are better. The E2E-MLP even surpasses the RND-0.05 model in the combined error measure. Additionally, the local error is smaller in comparison to the global error. Thus, the models are able to predict the inner state of the flock better. Fig. 4 supports this observation by showing that the flock stays together but moves slower than the simulated flock.

The ADJ-MLP performs best by far, even though it wrongly predicts 30% of the interactions between the boids. With nearly one-third of wrong interactions the predicted state deviates very little from the simulated state compared with the other models or the erroneous simulations. Therefore, we conclude that the correct simulation of the rules has a stronger impact on overall performance than determining the correct interactions.

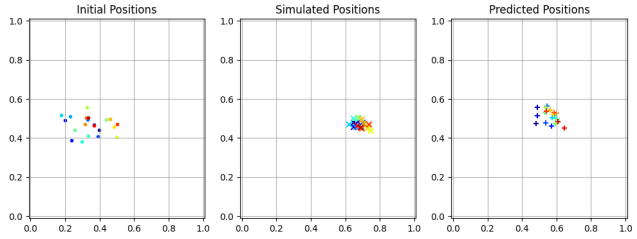

Figure 4: This figure depicts an exemplary prediction of the E2E-MLP for the third experiment. The flock has an initial movement targeting the right corner of the simulation area. In the prediction, the flock stays together and moves in the initial direction.

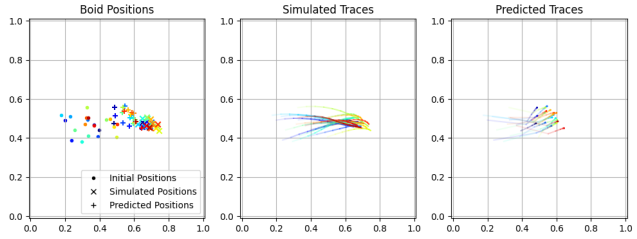

Figure 5: For an example from the the third experiment, we show the boids' traces, with weaker colors referring to earlier time steps.

Conclusion

Based on Reynolds' original boid model, we present a MMU-friendly notation that allows us to train simple surrogate models for the boids simulation. We run some baseline tests systematically introducing cumulative errors to understand how quickly the boid dynamics deviate and to understand which degree of accuracy a boid system surrogate should at least exhibit. We introduce a novel error measure that considers both the local and the global behavior of the predicted boids. We utilize this measure to train several surrogate models (E2E-MLP, ADJ-MLP, RULES-GNN, and MLP-GNN) on randomly sampled initial conditions of a boid system and, lastly, on an initial condition scenario that robustly results in a stable flocking cluster. Our analyses show that simple surrogate models are able to make predictions in the order of erroneous simulations that constantly introduce an absolute error of about 5%. For the presented proof of concept investigations, we rely on very simple model architectures that can be improved in the future. Considering the successful limitation of the complexity of the learning task in the stable flocking cluster experiment, a more systematic divide and conquer approach to learning and using surrogate models for different swarm configuration but also for simulating different phases of dynamics could be a generally fruitful direction for future research.

References

Angione, C., Silverman, E., and Yaneske, E. (2022). Using machine learning as a surrogate model for agent-based simulations. *PLOS ONE*, 17(2):1–24.

Chai, T. and Draxler, R. R. (2014). Root mean square error (rmse) or mean absolute error (mae)? – arguments against avoiding rmse in the literature. *Geoscientific Model Development*, 7(3):1247–1250.

Davison, T., Samavati, F., and Jacob, C. (2019). Lifebrush: painting, simulating, and visualizing dense biomolecular environments. *Computers & Graphics*, 82:232–242.

Giavitto, J.-L. (2003). Topological collections, transformations and their application to the modeling and the simulation of dynamical systems. *Lecture notes in computer science*, 2706:208–233.

Jiahao, T. Z., Pan, L., and Hsieh, M. A. (2022). Learning to swarm with knowledge-based neural ordinary differential equations. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 6912–6918.

Kipf, T. N. and Welling, M. (2016a). Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907.

Kipf, T. N. and Welling, M. (2016b). Variational graph auto-encoders. In *Bayesian Deep Learning Workshop at NIPS*. arXiv.

Klein, J. and Spector, L. (2009). 3d multi-agent simulations in the breve simulation environment. *Artificial life models in software*, pages 79–106.

Lee, J. M., Cho, S. H., and Calvo, R. A. (2009). A fast algorithm for simulation of flocking behavior. In *2009 International IEEE Consumer Electronics Society's Games Innovations Conference*, pages 186–190. IEEE.

Noble, J., Silverman, E., Bijak, J., Rossiter, S., Evandrou, M., Bullock, S., Vlachantoni, A., and Falkingham, J. (2012). Linked lives: The utility of an agent-based approach to modeling partnership and household formation in the context of social care. In *Proceedings of the 2012 Winter Simulation Conference (WSC)*, pages 1–12.

Powers, S., Smith, J., and Pinciroli, C. (2022). Extracting symbolic models of collective behaviors with graph neural networks and macro-micro evolution. In Dorigo, M., Hamann, H., López-Ibáñez, M., García-Nieto, J., Engelbrecht, A., Pinciroli, C., Strobel, V., and Camacho-Villalón, C., editors, *Swarm Intelligence*, pages 142–154, Cham. Springer International Publishing.

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, pages 25–34, New York, NY, USA. Association for Computing Machinery.

Styan, G. P. H. (1973). Hadamard products and multivariate statistical analysis. *Linear Algebra and its Applications*, 6:217–240.

van der Hoog, S. (2019). Surrogate modelling in (and of) agent-based models: A prospectus. *Computational Economics*, 53(3):1245–1263.

von Mammen, S. and Jacob, C. (2008). The spatiality of swarms-quantitative analysis of dynamic interaction networks. In *ALIFE*, pages 662–669.

von Mammen, S., Steghöfer, J.-P., Denzinger, J., and Jacob, C. (2011). Self-organized middle-out abstraction. In Bettstetter, C. and Gershenson, C., editors, *Self-Organizing Systems*, pages 26–31, Berlin, Heidelberg. Springer Berlin Heidelberg.

Willmott, C. J. and Matsuura, K. (2005). Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate Research*, 30(1):79–82.