INTERFACE

rsif.royalsocietypublishing.org

Research

Article submitted to journal

Subject Areas:
computational biology, ecology, computational mathematics

Keywords:
Bayesian inference, machine-learning, Gaussian Processes, collective movement, emergent properties

Author for correspondence:
Corresponding author
e-mail: colin.torney@glasgow.ac.uk

Inferring the interaction rules of complex systems with graph neural networks and approximate Bayesian computation

Jennifer Gaskell[1], Nazareno Campioni[1], Juan M. Morales[2,3], Dirk Husmeier[1], Colin J. Torney[1]

[1]School of Mathematics and Statistics, University of Glasgow, Glasgow, G12 8SQ, UK
[2]Grupo de Ecología Cuantitativa, INIBIOMA-CONICET, Universidad Nacional del Comahue, Argentina
[3]School of Biodiversity, One Health & Veterinary Medicine, University of Glasgow, Glasgow, G12 8SQ, UK

Inferring the underlying processes that drive collective behaviour in biological and social systems is a significant statistical and computational challenge. While simulation models have been successful in qualitatively capturing many of the phenomena observed in these systems in a variety of domains, formally fitting these models to data remains intractable. Recently, approximate Bayesian computation (ABC) has been shown to be an effective approach to inference if the likelihood function for a model is unavailable. However, a key difficulty in successfully implementing ABC lies with the design, selection, and weighting of appropriate summary statistics, a challenge that is especially acute when modelling high dimensional complex systems. In this work, we combine a Gaussian process accelerated ABC-method with the automatic learning of summary statistics via graph neural networks. Our approach by-passes the need to design a model-specific set of summary statistics for inference. Instead, we encode relational inductive biases into a neural network using a graph embedding and then extract summary statistics automatically from simulation data. To evaluate our framework, we use a model of collective animal movement as a test-bed and compare our method to a standard summary statistics approach and a linear regression-based algorithm.

1. Introduction

Understanding and predicting how complex interacting systems behave is a long-standing challenge that is relevant in a range of application contexts from bacterial movement [1] to political unrest [2]. A useful tool in the analysis of such systems has been individual-based modelling (IBM). However, the strength and form of the rules of interaction in such models is the key driver of the emergent phenomena that arise and is often difficult or impossible to measure in real systems. Fitting these models to data is therefore challenging and there is a trade-off between model fidelity and tractability; mechanistic, high-fidelity models are typically highly complex making it impossible to calculate the probability of an empirical observation for a given parameter set (the likelihood). When the likelihood is unavailable techniques involving simulation-based inference [3] are required.

Recently, simulation-based inference methods have focused on the use of approximate Bayesian computation (ABC) [4, 5] to estimate posterior distributions of model parameters when the likelihood is unavailable or too

THE ROYAL SOCIETY PUBLISHING

computationally intensive to evaluate. ABC methods are characterised by the use of summary statistics to compare real and simulated data via some distance metric, then employ rejection sampling based on this distance metric to sample from the posterior.

While ABC has been successfully applied in a wide range of applications, including systems biology, ecology, and climate modelling [6], a major drawback of the method is that it requires the use of informative summary statistics [7] in order to accurately capture posterior distributions. In scenarios where it is not *a priori* clear what summary statistics are appropriate or how they should be weighted, several methods have been proposed both to automatically adjust and select summary statistics [8] and to extract summary statistics from raw data [9].

In the context of interacting systems, extracting summary statistics is especially challenging. This is due to the nonlinear relationship between individual and group level behaviours [10] and the need to map high-dimensional microstates (i.e. small scale dynamics, location, orientation) to an informative reduced dimensional representation [11]. Hence, any method employed to automatically extract meaningful summary statistics for these types of systems is required to efficiently process high-dimensional data and effectively capture the nonlinear effects of varying model parameters.

Deep learning [12, 13] is a natural choice for this task as it is able to model complex patterns in data and provides a framework for the introduction of inductive biases that regularize model training. That is, they provide structural ways to constrain parameter space and avoid overfitting. The most common inductive biases used in deep learning are imposed through the use of convolutional neural networks [14] which make assumptions of locality and translation invariance. Convolutional neural networks have been tremendously successful in fields such as computer vision, however require a Euclidean, or grid-like, structure to the data. More recently, a class of models have been introduced that can be applied to data with a non-Euclidean underlying structure [15]. Specifically, graph neural networks [16] have been developed for reasoning about systems that consist of discrete entities (nodes) and the interactions between them (their edges).

Given that complex, interacting systems, by definition, consist of discrete entities that interact, it is reasonable to assume that graph neural networks (GNNs) would be an appropriate tool for the automatic extraction of summary statistics to be used in an ABC scheme. In this work, we develop such a method and apply it to synthetic data generated from a simulation of collective animal movement. To assess the performance of our method we compare its accuracy to a standard ABC scheme, and a scheme that derives summary statistics automatically using linear regression [8]. We show that a GNN-based ABC approach outperforms both methods when applied to the study of interacting systems.

Conceptually our method builds upon three recent developments in Bayesian inference and machine learning. Firstly, we employ Gaussian process (GP) emulation [17] and sequential history matching [18] to reduce the computational burden of ABC and enable relatively time-intensive neural network computations for each approximate likelihood evaluation. Secondly, we make use of the fact that the optimal choice of summary statistics (in the sense of minimizing the quadratic loss) can be obtained by using the posterior mean of the parameters as the summary statistics [8]. Even though the posterior means of the model parameters are not known, they can be approximated by training a deep neural network to predict model parameters based on simulation data [9]. Finally, we note that GNNs impose relational inductive biases that match the underlying network structure of the microscale data and are therefore a natural choice for inferring the parameters of individual-based models.

By combining and extending these previous works we are able to develop a GNN-based ABC method that accurately infers the posterior distributions of the parameters of individual-based models. We demonstrate and evaluate our method using a model of collective movement but the method may be applied to any domain in which individual-based simulation models are employed.

2. Background

(a) GP-accelerated ABC

A mathematical model of collective movement, like the one described in Section 3(a), typically depends on a parameter vector

$$\boldsymbol{\theta} = (\theta_1, \ldots, \theta_D). \tag{2.1}$$

Bayesian inference proceeds by obtaining or sampling from the posterior distribution $p(\boldsymbol{\theta}|\mathbf{y})$ given observed data $\mathbf{y}$. If the prior distribution of parameters is given by $p(\boldsymbol{\theta})$ and the likelihood function $\mathcal{L}(\boldsymbol{\theta}) = p(\mathbf{y}|\boldsymbol{\theta})$ is available in closed form then the posterior distribution may be calculated by application of Bayes' rule,

$$p(\boldsymbol{\theta}|\mathbf{y}) = \frac{p(\mathbf{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathbf{y})}. \tag{2.2}$$

In some situations Eq. 2.2 is tractable and a closed form expression for the posterior obtained. When Eq. 2.2 is available subject to a normalizing constant then stochastic sampling schemes, such as Markov chain Monte Carlo (MCMC), can be employed to sample from the posterior distribution. However, in many applications it is not possible to compute the likelihood function $\mathcal{L}(\boldsymbol{\theta})$ meaning both direct computation of the posterior and stochastic sampling are impossible. In such cases simulation-based inference [3] is a powerful tool for inferring parameters and quantifying uncertainty that expands the domain of Bayesian inference to include scenarios where the likelihood is unavailable but it's possible to draw a parameter set $\boldsymbol{\theta}'$ from the prior $p(\boldsymbol{\theta})$ and then generate a realization from the model, $\mathbf{y}' \sim p(\mathbf{y}|\boldsymbol{\theta}')$ using a stochastic simulator.

Simulation-based inference can be broadly classified into two categories: density estimation methods [5], which recently has included the use of normalizing flows [19], and approximate Bayesian computation (ABC) [6].

In its simplest form, ABC employs rejection sampling to sample from the posterior. A parameter set is drawn from the prior, then the stochastic simulator is used to generate $\mathbf{y}'$. The parameter set is then retained as a sample from the posterior distribution if the data generated from the simulator $\mathbf{y}'$ is sufficiently close to the actual empirical data $\mathbf{y}$, i.e. if $\rho(\mathbf{y}', \mathbf{y}) < \epsilon$ where ρ is a distance measure and ϵ is a tolerance. Hence, the likelihood is approximated by the acceptance probability and the value of ϵ controls the trade-off between speed and accuracy of the approximation. In the limit $\epsilon \to \infty$ the posterior is approximated by the prior. In the limit $\epsilon \to 0$ inference is exact but for continuous data the acceptance probability also approaches zero. The accuracy of standard rejection ABC is therefore dependent on selecting a tolerance value that is as small as possible given the constraints of computational resources.

In practice, this can be improved by mapping the typically high-dimensional vector of the original model outputs $\mathbf{y}$ to a lower-dimensional space of summary statistics

$$\begin{aligned} \mathbf{S}: \mathbb{R}^n &\to \mathbb{R}^k \\ \mathbf{y} &\to \mathbf{S}(\mathbf{y}) = \Big(S_1(\mathbf{y}), \ldots, S_k(\mathbf{y})\Big) \end{aligned} \tag{2.3}$$

where n and k are positive integers with typically $k \ll n$. The acceptance criterion is now modified based on $\mathbf{S}(\mathbf{y})$: a parameter set is drawn from the prior, the stochastic simulator is used to generate $\mathbf{y}'$, and the corresponding parameter set is then retained in the sample if the vector of summary statistics of the data generated from the simulator, $\mathbf{y}'$, is sufficiently close to the vector of summary statistics of the empirical data $\mathbf{y}$, i.e. if

$$\rho\Big(\mathbf{S}(\mathbf{y}'), \mathbf{S}(\mathbf{y})\Big) < \epsilon. \tag{2.4}$$

If the vector of summary statistics is *sufficient*, then we have the same convergence guarantees as before, i.e. in the limit $\epsilon \to 0$ the sample will converge in distribution to the true posterior distribution. However, *sufficiency* is a restrictive criterion that in real applications of complex models is hardly ever satisfied. In practice, the choice of summary statistics is typically heuristic, and the process is approximate even in the limiting case $\epsilon \to 0$.

Many improvements to ABC have been developed that focus on more efficiently exploring the parameter space in order to relieve the computational burden of the method and reduce the required number of expensive forward simulations of the stochastic simulator [20, 21]. In particular, improvements have aimed to develop more efficient proposal schemes for new parameters, using regression-based approaches [22], a modified Metropolis-Hastings scheme [23], or integrating a sequential Monte Carlo (SMC) sampling strategy [24] into the ABC scheme. In addition, improved scheduling schemes for adapting the decision threshold parameter ϵ have been developed [25]. Of particular interest to the present article is an approach termed Gaussian process accelerated ABC (GP-ABC) [18], which combines the ideas of ABC and statistical emulation [26] to improve the efficiency of ABC by taking advantage of the continuity and smoothness of the likelihood surface via a combination of sequential history matching and statistical emulation.

Given $\boldsymbol{\theta} \in \mathcal{R}^D$, the GP-ABC method begins by sampling N points across the D-dimensional parameter space using a space filling design. Multiple forward simulations are then performed for each of the sample points, the summary statistic(s) are calculated from the simulation output, and the approximate log-likelihood is calculated based on an acceptance kernel and a distance metric. Details will be discussed below; see (2.6). This first set of parameters make up the first ensemble of points in the training set and are associated with *wave 1* of the inference procedure. A D-dimensional GP is then fit to the log-likelihood values to create an approximation to the true log-likelihood surface.

Wave 2 proceeds by continuing to sample using the space-filling design and sampling a further N points from the parameter space. Before running forward simulations the plausibility of each point is assessed by using the GP model from the previous wave. The GP model is employed to predict the log-likelihood surface at the newly sampled points and each point is only retained if it exceeds an implausibility threshold given by,

$$m + 3\sigma < \mathcal{L}_{MAX} - T, \tag{2.5}$$

where m and σ^2 represent the mean and the variance of the prediction of the modelled log likelihood at the point under consideration, $\mathcal{L}_{MAX}$ is the maximum predicted log-likelihood at the sample locations of the previous wave, while T is a pre-specified threshold which determines how stringently points are excluded from further consideration.

Sample points that do no meet the plausibility criterion are discarded while remaining points are retained as the sample locations for *wave 2*. As before, the simulator is then employed to run multiple simulations for each sample. The approximate log-likelihood values from these points are then used as the basis for the *wave 2* GP and the process is repeated.

As each wave of the inference procedure is executed, more training points are available for the GP regression model. This leads to a more accurate GP model with lower uncertainty which is then able to effectively rule out regions of parameter space as implausible. Simulation effort is therefore focused in regions of high likelihood. Once a sufficient number of waves have been completed, the GP model from the final wave is used to emulate the approximate log-likelihood surface and becomes the objective function for a standard MCMC sampling scheme. MCMC sampling is rapid since no new forward simulations are required at this point, instead the proposals are accepted or rejected solely based on the GP model predictions.

As mentioned previously, calculating the approximate log-likelihood values from forward simulation output is

based on an acceptance kernel and a distance metric. As in [27], here we employ a multivariate Gaussian acceptance kernel and a distance metric based on derived summary statistics. Hence, the approximate log-likelihood value is given by an unbiased Monte Carlo estimate of the acceptance probability for the parameter set,

$$\mathcal{L}(\boldsymbol{\theta}') = \frac{1}{M}\sum_{i=1}^{M} \pi(\mathbf{S}(\mathbf{y}) - \mathbf{S}(\mathbf{y}_i')), \tag{2.6}$$

where $\pi(.)$ is a multivariate Gaussian acceptance kernel, $\mathbf{S}(.)$ is the projection of the full high dimensional data onto a lower-dimensional vector of summary statistics, as defined in (2.3), $\mathbf{y}$ is the empirical data, and $\mathbf{y}_i'$ is the simulation output of the ith model simulation from M total replicates at the parameter value $\boldsymbol{\theta}'$. These replicates reflect the stochastic nature of the underlying model and formally constitute a Monte Carlo approximation of the path integral over the latent (stochastic) degrees of freedom.

We employ a Gaussian acceptance kernel $\mathcal{N}(\mathbf{0}, \boldsymbol{\epsilon})$, with a diagonal covariance matrix where the width of the kernel $\boldsymbol{\epsilon}$ is set according to the standard deviation of the summary statistics within the observed empirical data [28].

As mentioned above, the choice of summary statistics $\mathbf{S}$ is mainly heuristic, and various schemes have been proposed to improve their selection and weighting. In what follows, we summarize the semi-automatic method proposed in [8] and describe how we have integrated it into our GP-ABC scheme. Starting from our initial space-filling design, using e.g. a Saltelli sampler [29]

$$\Psi = \{\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(N)}\} \tag{2.7}$$

we run, for each of the parameter vectors $\boldsymbol{\theta}^{(n)}$ in turn, $n \in \{1, \ldots, N\}$, a forward simulation from our model to obtain a sample of model outputs $\{\mathbf{y}_1', \ldots, \mathbf{y}_N'\}$. We map these model outputs into our summary statistics $\{\mathbf{S}(\mathbf{y}_1'), \ldots, \mathbf{S}(\mathbf{y}_N')\}$ and fit, for each of the parameters $\theta_1, \ldots, \theta_D$ from (2.1) in turn, the linear regression model

$$\theta_i = c_i + \boldsymbol{\beta}_i^T \mathbf{S}(\mathbf{y}) + \xi_i \tag{2.8}$$

using least-squares, where ξ_i is some zero-mean noise, $\boldsymbol{\beta}_i$ is a parameter vector, c_i is an intercept, and the superscript T denotes matrix transposition. The intercept c_i can be neglected in practice as ABC only uses the difference in summary statistics. This leads to the new vector of summary statistics

$$\tilde{\mathbf{S}} = (\tilde{S}_1, \ldots, \tilde{S}_D) = \left(\boldsymbol{\beta}_1^T \mathbf{S}(\mathbf{y}), \ldots, \boldsymbol{\beta}_D^T \mathbf{S}(\mathbf{y})\right) \tag{2.9}$$

which replaces the original summary statistics $\mathbf{S}$. Note that (2.8) is *not* used as a direct parameter estimator, as the corresponding function is not guaranteed to be injective. Moreover, directly estimating the parameters based on (2.8) would not allow the estimation uncertainty to be quantified. As a final comment, it is straightforward to replace the linear regression model of (2.8) by a more flexible non-linear regression model, like a Gaussian process. See [30] for details.

(b) Graph neural networks

Graph neural networks (GNNs) are a recent development in the field of deep learning that can be used for the analysis of data with a non-Euclidean underlying structure [16, 31]. GNNs are a generalization of other types of neural networks, such as convolutional or recurrent neural networks, that relax the assumption of a Euclidean structure to the data and can be applied to any data that can be described as a graph.

By explicitly encoding a graph structure into the neural network, a GNN imposes relational inductive biases into the learning process [16] (that is, they impose constraints on the possible relationships and interactions among the entities in the system), and as a result these models are significantly more efficient and less prone to over-fitting than other approaches for processing graph-structured data such as fully-connected networks. GNNs are therefore able to learn the behaviour of more complex systems [32–34] than would be achievable with other approaches.

Given a graph $G = (V, E)$ where $|V| = N^v$ is the number of nodes (or vertices) in the graph, and $|E| = N^e$ is the number of edges, a GNN takes as input the graph G and an associated set of node, edge and potentially graph-level features. The GNN is trained to learn a sequence of message-passing [35] steps, where information is passed between nodes via the edges and the features of the graph are updated. A layer implementing a single message-passing step updates a node's features $\mathbf{x}$ according to the update rule,

$$\mathbf{x}_i' = \gamma\left(\mathbf{x}_i, \mathcal{F}_{j\in\mathcal{N}_i}\phi\left(\mathbf{x}_i, \mathbf{x}_j, \mathbf{e}_{ji}\right)\right) \tag{2.10}$$

where γ is a differentiable update function, ϕ is a differentiable message processing function, $\mathcal{F}$ is some aggregation function that combines the messages from multiple neighbours (e.g. sum or mean), $\mathcal{N}_i$ is the neighbourhood of node i, i.e. the set of nodes that share an edge with node i, and $\mathbf{e}_{ij}$ is the feature vector associated with the edge connecting nodes i and j.

The message passing framework may also be extended to update edge features [34] and to incorporate and/or update global, graph-level features via pooling [36] or broadcasting. GNNs may be trained to perform node-level tasks, edge-level tasks, or make graph-level predictions. In this work, we focus on graph-level predictions and train a GNN to predict the underlying parameters that created a given microstate configuration.

3. Methods

Our methodology builds upon a combination of Approximate Bayesian Computation (ABC) and statistical emulation with Gaussian processes [18], using sequential history matching to increasingly focus on the parameters that achieve the best match with the data. An illustration of the methodological framework is provided in Fig. 1. Our work focuses on the key challenge of extracting summary statistics from high-dimensional data (indicated in Fig. 1 by dashed arrows). We automate the process of

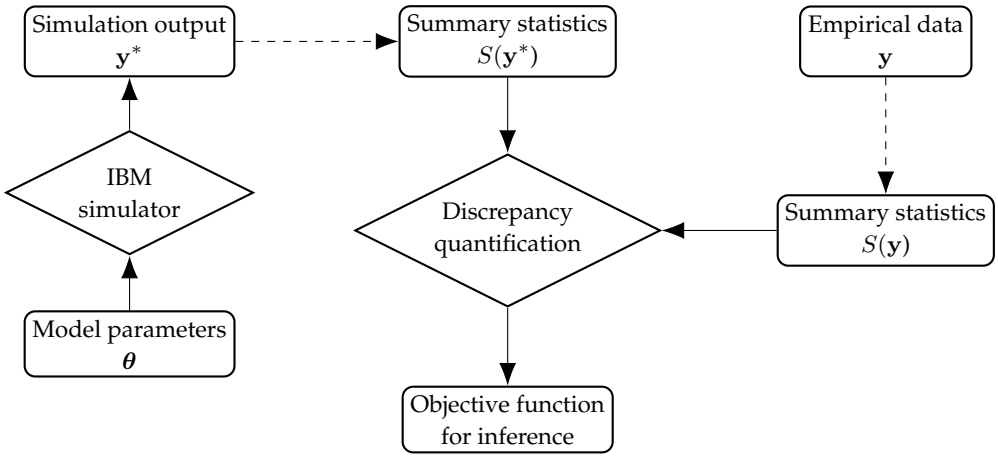

Figure 1. Conceptual illustration of the inference scheme. The collective movement model, defined in equations (3.1-3.5), and its parameters define an individual-based model (IBM) simulator for generating microscale data $\mathbf{y}^*$. We apply a pattern recognition operator to extract summary statistics from these data. The same summary statistics are extracted from empirical data. A discrepancy operator quantifies the mismatch between summary statistics from real and simulated data, which leads to the parameter dependent objective function (2.6) that is used for inference. There are various additional methodological details that are not included in the figure. For inference we use a Gaussian process emulator, which in several waves of sequential history matching increasingly focuses on the parameters that achieve the closest match. Summary statistics may be chosen in advance, based on domain knowledge or the user's intuition, and combined and weighted using linear regression. However, we propose a method for learning these summary statistics directly from simulation data using a machine learning approach based on graph neural networks.

calculating summary statistics by employing a pre-trained GNN thereby eliminating the need to handcraft reduced dimensional representations of the data based on domain knowledge or experimentation.

Our method therefore proceeds in three stages, firstly we train the GNN on simulation data to obtain an optimal encoder for ABC. We then use GP-based statistical emulation and sequential history matching to create a surrogate log-likelihood surface following the approach of [18], as briefly outlined in Section 2(a) but using the GNN to compute summary statistics. Finally, we sample from the posterior distributions using a standard MCMC sampler and the inexpensive surrogate log-likelihood. Further details of the inference method and the synthetic data generation process we use as a test bed for our method are given in the following subsections.

(a) Simulation model

To assess the accuracy of the proposed inference framework described in more detail in the next section, we use the well established zonal interaction model from the collective movement literature [37]. While our methodology is not restricted to this application domain, collective movement models are canonical examples of individual-based models [38] that consist of multiple parameters and display complex nonlinear behaviour as parameter values are varied [39]. Hence, this represents an ideal test bed for assessing our method.

The model simulates the movement of interacting individuals moving at constant speed within a periodic, two-dimensional domain. The model explicitly includes three interaction zones: repulsion, alignment, and attraction, with respective radii l_r, l_{al} and l_{at}, and $l_r < l_{al} < l_{at}$, and a visual interaction angle v_a. We define three sets of neighbours of a focal individual i,

$$\begin{aligned} n_i^r &= \{j : |\mathbf{r}_j(t) - \mathbf{r}_i(t)| < l_r, -v_a/2 < \angle_{ij} < v_a/2\} \\ n_i^{al} &= \{j : |\mathbf{r}_j(t) - \mathbf{r}_i(t)| < l_{al}, -v_a/2 < \angle_{ij} < v_a/2\} \\ n_i^{at} &= \{j : |\mathbf{r}_j(t) - \mathbf{r}_i(t)| < l_{at}, -v_a/2 < \angle_{ij} < v_a/2\} \end{aligned} \quad (3.1)$$

where $\mathbf{r}_i(t)$ is the spatial location of individual i at time t, and $\angle_{ij}$ is the relative angle of individual j calculated with respect to the heading of individual i. The sets of neighbours each give rise to a social direction vector, defined as

$$\begin{aligned} \mathbf{u}_i^r &= -\sum_{j \in n_i^r} \mathbf{r}_j(t) - \mathbf{r}_i(t) \\ \mathbf{u}_i^{al} &= \sum_{j \in n_i^{al}} \mathbf{d}_j(t) \\ \mathbf{u}_i^{at} &= \sum_{j \in n_i^{at}} \mathbf{r}_j(t) - \mathbf{r}_i(t) \end{aligned} \quad (3.2)$$

where $\mathbf{d}_j(t)$ is the direction vector of indvidual j. Next, a desired heading vector $\mathbf{u}_i$ is calculated for individual i using

$$\mathbf{u}_i = \frac{\mathbf{u}_i^r}{|\mathbf{u}_i^r|} \quad (3.3)$$

if $n_i^r \neq \emptyset$, otherwise

$$\mathbf{u}_i = \frac{\mathbf{u}_i^{al}}{|\mathbf{u}_i^{al}|} + \frac{\mathbf{u}_i^{at}}{|\mathbf{u}_i^{at}|}. \quad (3.4)$$

Once this socially-informed desired direction vector has been calculated for individual i, the following update rule

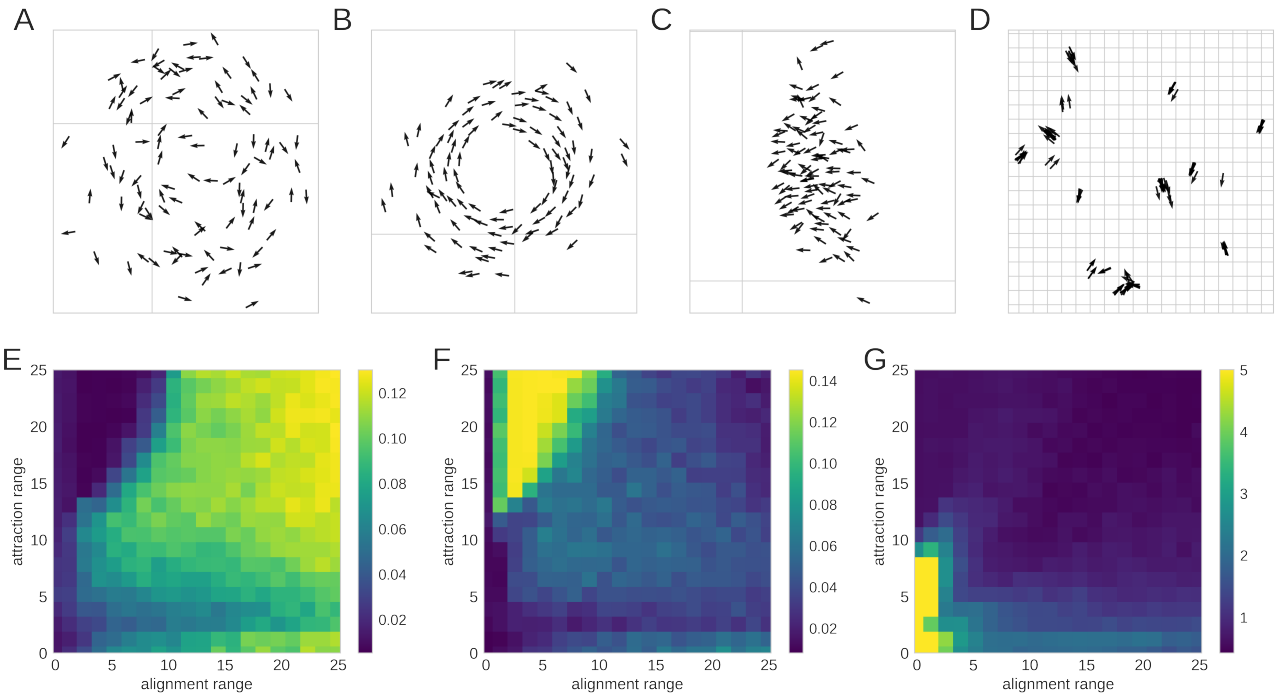

Figure 2. Panels (A)-(D): Four behaviours arising from different combinations of the zonal interaction model parameters, for different parameter combinations of the alignment range, l_{al}, and the attraction range, l_{al}. Panel (A): swarm ($l_{al}=0, l_{at}=15$). Panel (B): toroidal ($l_{al}=3, l_{at}=15$). Panel (C): dynamic parallel ($l_{al}=10, l_{at}=10$). Panel (D): concentrated parallel ($l_{al}=20, l_{at}=10$). All other parameters are fixed at $v_s=3$, $v_a=1.5\pi$, $\eta=0.9$, $l_r=1$. Panels (E)-(G): The summary statistics in the $l_{al}-l_{at}$ parameter space. Panel (E): order parameter. Panel (F): rotation parameter. Panel (G):nearest neighbour distance.

is implemented to update its position and heading,

$$\begin{aligned}\mathbf{r}_i(t+\Delta t) &= \mathbf{r}_i(t) + v_s\mathbf{d}_i(t)\Delta t \\ \mathbf{d}_i(t+\Delta t) &= (1-\eta)\frac{\mathbf{u}_i}{|\mathbf{u}_i|} + \eta\mathbf{d}_i(t)\end{aligned} \tag{3.5}$$

where v_s is the fixed speed of the individuals and η is a directional persistence parameter which determines how quickly individuals respond to social cues.

Simulations are run using $N=100$ individuals with a timestep of $\Delta t=0.1$. Sample configurations for different values of the parameters can be found in Fig. 2.

For comparison with standard ABC methods, we extract summary statistics manually from simulation output using well known macroscale quantities [10, 39]. Firstly, we use the order parameter, which is a metric of the global alignment of individuals and is defined as,

$$\frac{1}{N}\left|\sum_{i=1}^{N}\mathbf{d}_i(t)\right|. \tag{3.6}$$

The order parameter is equal to 1 for a fully-ordered system and 0 for a completely disordered system. Our second summary statistic is the group angular momentum [39, 40],

$$\frac{1}{N}\left|\sum_{i=1}^{N}\mathbf{c}_i(t)\times\mathbf{d}_i(t)\right|, \tag{3.7}$$

where $\mathbf{c}_i(t)$ is a unit vector that points in the direction of the group centroid $\mathbf{c}(t)$ given by,

$$\mathbf{c}(t)=\frac{1}{N}\sum_{i=1}^{N}\mathbf{r}_i(t). \tag{3.8}$$

As our final summary statistic, we use the average nearest neighbour distance [41] defined as,

$$\frac{1}{N}\sum_{i=1}^{N}\min_{j,j\neq i}\left(|\mathbf{r}_i(t)-\mathbf{r}_j(t)|\right). \tag{3.9}$$

Summary statistic values for a range of attraction and alignment values can be found in Fig. 2. The emergent group behaviour exhibited by the model can switch between swarm, torus, dynamic parallel and highly parallel groups, depending on the parameter values, as shown in Fig.2(a-d). These behaviours are not equally spread across the parameter space, for example, the toroidal behaviour is confined to a small region where the attraction radius is large and the alignment radius is low. This can be seen in the summary statistic scans over the $l_{at}-l_{al}$ parameter space, shown in Fig.2(e-g).

(b) Inference framework

We implement our GNN-ABC approach within the GP-accelerated inference framework proposed by [18]. For

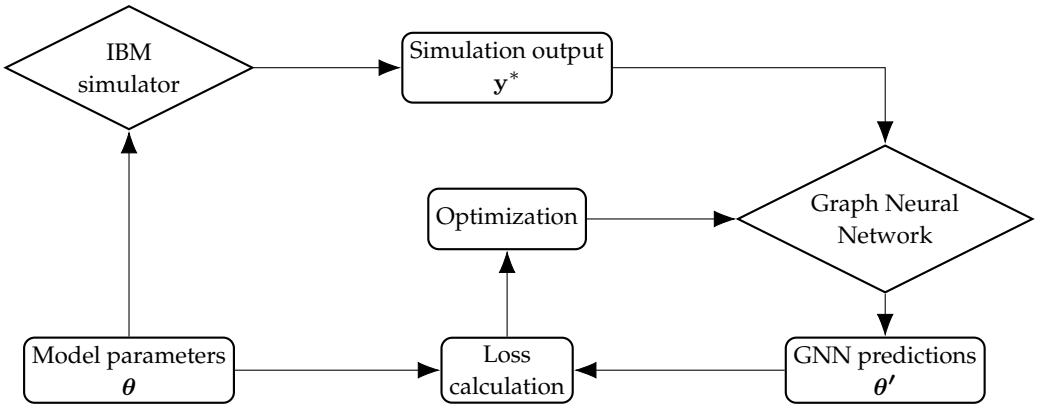

Figure 3. Training the GNN model. To create a pattern recognition operator for the automatic calculation of ABC summary statistics, we train a GNN on simulated data. The simulator generates microscale data for a given parameter set $\boldsymbol{\theta}$, the GNN then predicts the parameter $\boldsymbol{\theta}'$ based on the microscale data. The predictions and the true values are compared using a mean squared loss function, and the weights and biases of the GNN are then updated to minimize this loss using gradient descent.

comparison, we also perform inference using manually derived summary statistics and semi-automatic summary statistics [8]. An overview of the framework can be found in Algorithm 1 and we outline specific implementation details below.

As a test bed for our method, we use synthetic data generated from the zonal interaction model defined above. We simulate the zonal model for 2000 time steps and then record the positions and orientations of each individual. For each run, we have 100 individuals, each initialised with a random orientation and a random position within a 10x10 unit square in the centre of the domain (the individuals are started together to speed up reaching the equilibrium state). This is repeated 100 times to provide a set of synthetic empirical data observations from which we attempt to infer the interaction parameters that generated the microscale configuration. This simulated data-set exemplifies the case where several "snapshots" of a group of interacting individuals are available showing the relative position and heading of the individuals.

The first stage in the inference process is to determine the summary statistics to be used within the ABC framework. When employing manually derived summary statistics this process involves computing the values defined by Eqns. 3.6, 3.7 and 3.9. When using the GNN to automatically extract summary statistics this first phase is achieved via a process of simulation and model training described in detail the next section.

Once summary statistics are defined, we specify a uniform prior over the parameter values, using estimates of plausible minimum and maximum values, and then sample from the parameter space using a Halton sequence ([42], Chapter 7). For each wave we take 25 points from the sequence and then run forward simulations for each parameter set. Forward simulations are run in parallel on a GPU with 500 independent repeats performed for each parameter set. A single repeat corresponds to running the simulator for 2000 timesteps and then recording the positions and headings of all individuals, with randomised initial conditions as described above. After the simulation output has been obtained, we calculate the approximate log likelihood using Eqn. 2.6 and the relevant summary statistic mapping depending on the method being used (either manual, semi-automatic, or fully-automatic GNN). The log likelihood values for each wave are passed into a Gaussian process regression model with a radial basis function kernel [43] using the package GPy [44] in order to emulate the likelihood surface.

The next wave begins with a further 25 points sampled from the Halton sequence. New sample points are ruled as implausible based on their predicted log likelihood obtained from the GP models from previous waves and the implausibility criterion of Eqn. 2.5 with a threshold $T = 3$. To be classed a plausible, a point needs to pass the implausibility test based on the predictions from GP models from all previous waves.

Algorithm 1 GP-ABC inference

Initialise prior space, Θ_0
Create set $\mathbf{X}_1$ using $n = 25$ Halton points from Θ_0
Run simulations and calculate approx. log-likelihood per point $\mathbf{L}_1 = \mathcal{L}(\mathbf{X}_1)$
for $w \leftarrow 1$ to 10 **do**
  Fit GP model, M_w to log likelihood values $\mathbf{X}_w, \mathbf{L}_w$
  Create set $\mathbf{X}_{w+1}$ with next $n = 25$ points from Θ_0
  Remove implausible points using $M_{\leq w}$ to give $\mathbf{X}_{w+1} = \mathbf{X}_{w+1}/\mathbf{X}_{IMP}$
  Run simulations and calculate approx. log-likelihood per point $\mathbf{L}_{w+1} = \mathcal{L}(\mathbf{X}_{w+1})$
  Combine parameters and likelihood values from previous wave: $\mathbf{X}_{w+1} = \{\mathbf{X}_{w+1}, \mathbf{X}_w\}$, $\mathbf{L}_{w+1} = \{\mathbf{L}_{w+1}, \mathbf{L}_w\}$
end for

We run 10 waves of GP-ABC inference, with each wave refining the log likelihood surface and increasing the number of forward simulations in regions of high posterior probability. After the tenth wave, the final GP model acts as an emulator for the log likelihood and is used as the objective function of an adaptive Metropolis Hastings sampler [45]. The sampler bases its acceptance probabilities on samples from the GP model to account for uncertainty in the surface and will reject a proposal that is deemed implausible by any of the GP models associated with the multiple waves of inference. The proposal distribution is tuned throughout the burn-in phase to optimise sampling efficiency [45].

(c) GNN-based ABC

One of the great advantages of deep learning is its ability to learn directly from data without any manual feature design or engineering [12]. Here we apply this facet of deep learning to automatically extract summary statistics directly from simulations of an individual-based model. We employ GNNs as they are ideally suited to modelling data that can be structured as a collection of discrete entities that interact via an interaction network as is the case with most models of complex systems.

In order to connect GNNs with statistical inference, we make use of the fact that an ideal set of summary statistics for ABC corresponds to the posterior means of the parameters to be inferred [8]. Hence, we can train our neural network on simulation data to predict the underlying model parameters as an approximation to the true posterior mean [9].

To process simulation output by the GNN, we formulate the data as a graph where each node corresponds to an individual in the zonal model and each directed edge corresponds to a potential interaction with a neighbour. There are two features associated with each node that are used to encode the two-dimensional heading vector of the individual and five features associated with edges between nodes that encode the distance between the nodes, the distance in x and y coordinates, and the neighbour velocity relative to the source node for the edge. Once the simulation data is in a graphical format it is passed into the GNN. We base the design of our GNN on the XENet architecture proposed in [34]. We select this architecture as a basis for our approach as firstly it explicitly processes both node and edge attributes, and secondly it is applicable to symmetric directed graphs with asymmetric edge attributes. The need for this second property arises due to the visual interaction angle of the zonal model which creates asymmetric edge attributes between two connected nodes.

Since our architecture follows closely that used in [34] we only briefly summarise the design here and include a visualisation of the neural network in Fig S1. Our network begins with a dense layer for both node and edge features, followed by 2 XENetConv layers and an additional dense layer. As we are predicting graph-level features, i.e. the model parameters shared by all individuals, we next use a combined maximum pooling and average pooling layer to aggregate node features. Finally, aggregated features are passed through two additional dense layers, with the output layer employing a softplus activation function to ensure parameter predictions are positive. We implement our GNN using the Spektral library for graph deep learning [46].

The GNN is trained using the microscale data taken from the IBM simulator using the Adam optimizer [47]. The model loss is taken to be the mean squared error between the predicted model parameters and the true parameters. Training continues until the validation loss, as calculated using a separate validation data set (10% of the data set), stops improving.

Preparing the automatic summary statistics therefore follows an approach of IBM simulation and GNN training as illustrated in Fig. 3. Note, as we are training the GNN on simulated data we are free to train the network for as long as needed and we are not constrained by training data set size. Instead, we may continue to generate training data from the forward simulator and repeat the training loop until an accurate GNN model is obtained.

(d) Semi-automatic summary statistics

Selecting and weighting informative summary statistics is key to the success of the GP-ABC inference scheme; however, finding a good choice can be challenging and is often unique to the particular form of the simulation model. For comparison with the GNN-based method described in the previous subsection, we have adapted the semi-automatic weighting method proposed in [8], as reviewed briefly at the end of Subsection 2(a), to the sequential history matching GP-ABC scheme with its several waves. This is naturally accomplished by replacing the set of parameters from the original design Ψ, given in equation (2.7), by the set of parameters obtained from the previous wave, and then refitting the linear regression model (2.9). This scheme is iterated all the way through to the final wave. Intuitively, the initially uninformative space-filling design of the a priori plausible parameter space is thus iteratively shrunk to be increasingly focused on the posterior mode. In the present work, we start with our three raw summary statistics: the order parameter, the rotation parameter and the minimum nearest neighbour distance. These parameters are then transformed by application of (2.8), and we have a separate equation with different regression parameters for each original model parameter to be inferred. The first regression at the lowest wave is based on a standard space-filling design of the parameter space, covering the entire compact support of the prior distribution with a Saltelli sampler [29]. For each subsequent wave, the regression model of (2.8) is re-fitted, replacing the original domain in parameter space by the new domain that has not yet been ruled out. In our simulations, we have used 10 waves by default. However, when the regression coefficients in (2.8) between successive waves did not change significantly, this was taken as indication of sufficient convergence, and no further waves were considered.

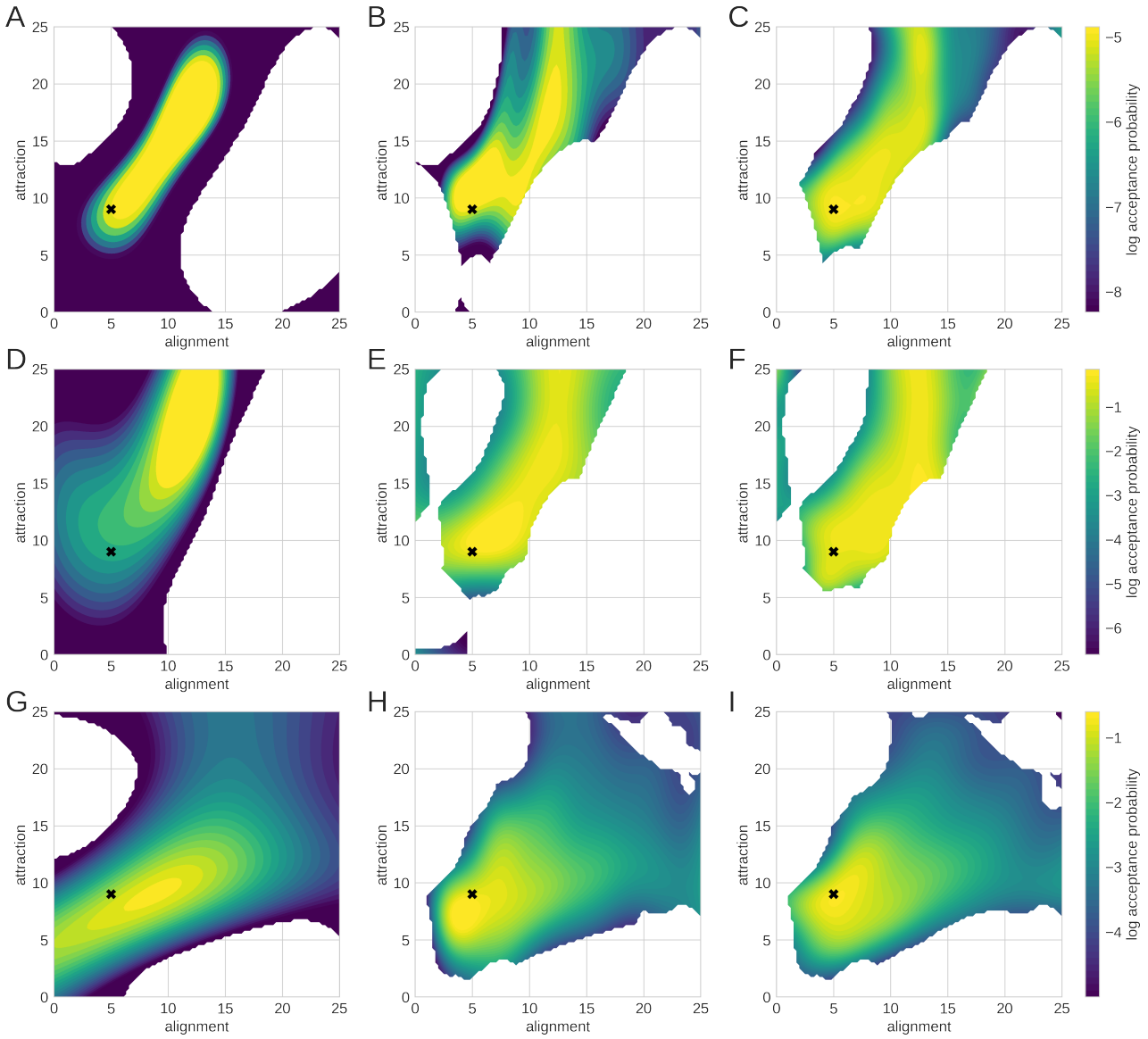

Figure 4. Emulated log-likelihood surface for each method. (A-C) Waves 1, 4, and 7 for the standard summary statistic approach. (D-F) Waves 1, 4, and 7 for the semi-automatic method. (G-I) Waves 1, 4, and 7 for the GNN method. All results generated with implausibility threshold $T = 3$, and IBM model parameters $l_{rep} = 1$, $\eta = 0.9$, $v_a = 1.5\pi$. **x** indicates the true parameter value used to create the synthetic data. Empty regions indicate regions that have been classed as implausible.

4. Results

To evaluate the performance of our framework, we test the method on synthetic data generated with an array of different parameters. This allows for the quantification of performance based on a known ground truth, which would not be available for real data, and a comparison of the different methods.

In Fig.4, we show three example waves of GP-ABC using the different approaches to calculating summary statistics. We infer two parameters, the attraction radius and the alignment radius, and assume all other parameters are known. We see an improvement throughout the waves (waves 1, 4, and 7 are pictured for each method) as more simulations are performed however the GNN method is the most effective. The emulated log-likelihood surface is peaked at the true value for the GNN method and we observe a greater ability to distinguish between interactions over longer distances, whereas both the standard approach and the semi-automatic approach result in a ridge-like structure to the likelihood surface where the likelihood effectively plateaus for a range of parameter values.

Following this qualitative comparison of the methods, we next evaluate each method for four different parameter combinations, representing different behaviours across the parameter space, with ten replicas of synthetic data for each

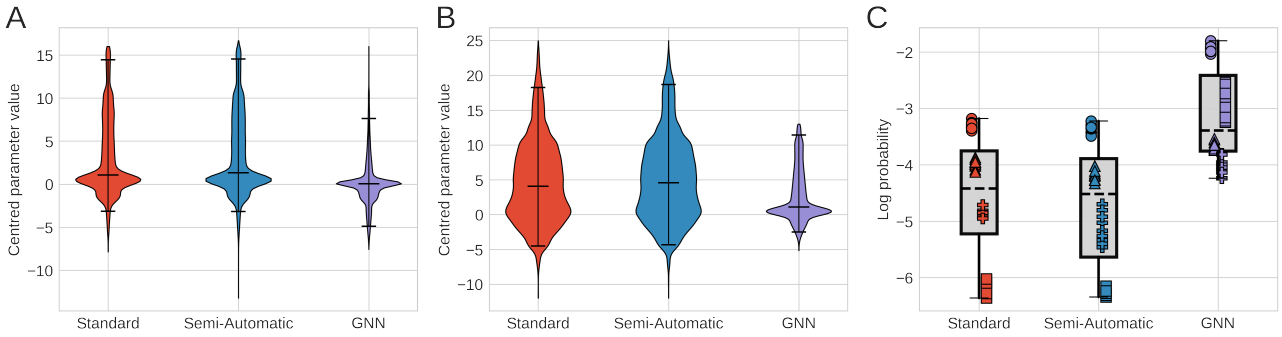

Figure 5. Comparing inferred and true values for two-parameter inference. Inference for four different parameter combinations are shown: $\{l_{al} = 0, l_{at} = 14\}$, $\{l_{al} = 2, l_{at} = 12\}$, $\{l_{al} = 9, l_{at} = 5\}$, $\{l_{al} = 14, l_{at} = 0\}$, with $l_r = 1$ and $v_a = 1.5\pi$. 10 replicas are performed for each parameter set where each replica involves creating synthetic data and inferring model parameters. (A) Combined violin plot for centred MCMC samples of the alignment radius l_{al} for each method. A value closer to zero indicates better performance, and a smaller distribution represents reduced uncertainty. (B) Combined violin plot for centred MCMC samples of the attraction radius l_{at} for each method. (C) The log probability of the true parameter value given the posterior distribution, calculated using a kernel density estimator. A value of each replica is shown with shapes indicating the parameter set, circles: $\{l_{al} = 0, l_{at} = 14\}$; triangles: $\{l_{al} = 2, l_{at} = 12\}$; +-symbols: $\{l_{al} = 9, l_{at} = 5\}$; squares: $\{l_{al} = 14, l_{at} = 0\}$. Larger values of log probability indicate the best performance.

parameter value. In this comparison, we run the GP-ABC waves and then sample from the posterior distribution of the parameter values using MCMC sampling.

Fig. 5 shows the results when we try to infer only two parameters of the model and assume other parameters are known. To illustrate the combined results across multiple parameter values and replicas, we centre the MCMC samples at the true parameter value for the replica and then combine all samples into a single violin plot for each method shown in Figs. 5A and 5B. Hence, a posterior distribution strongly peaked at zero equates to an accurate method across all parameter values and replicas.

To further quantify performance, we fit a kernel density estimator to the MCMC samples, to get an approximate posterior distribution, and then calculate the log-probability of the true parameter values given this approximate posterior probability distribution. Results from this analysis are shown in Fig. 5C. These results show that the GNN approach presents a significant improvement over both the standard ABC approach and the semi-automatic method. While it is clear that some parameter values are more difficult to accurately infer for all methods, the GNN approach gives consistently more accurate posterior probabilities for the parameters.

We next repeat this analysis in the scenario where four model parameters are unknown. As before we attempt to infer attraction and alignment radii but now also attempt to infer the visual angle and the repulsion radius of the individuals. We again run 10 waves of GP-ABC inference followed by running an MCMC sampler with the emulated log-likelihood surface as the objective function.

To evaluate the performance of the methods in the four-dimensional inference we again centre the posterior samples and produce violin plots of the posterior distributions centred on the true parameter value. Results from this analysis are shown in Fig. 6 where we observe that the GNN method is significantly more accurate at inferring model parameters for three of the four parameters, with a notable improvement observed in the visual angle and the repulsion radius parameters. As for the two-dimensional inference case, we use a kernel density estimator to calculate the log-probability of the true parameter values given the inferred posterior distributions and this analysis is shown in Fig. 6C. Again we observe that the GNN approach represents a significant improvement over the previous methods.

5. Conclusions

We have presented a novel approach to parameter inference using graph neural networks to automate the selection and weighting of summary statistics traditionally required for state-of-the-art Approximate Bayesian inference methods. We compare this method to two standard ABC approaches [8, 27] and note an improvement in performance, showing both an increased accuracy and reduced uncertainty. These findings were even more pronounced when the difficulty of inference was increased by expanding the inference to cover four parameters simultaneously.

Approximate Bayesian computation makes it possible to perform inference in models whose likelihood function is unavailable or very expensive. However, the successful implementation of ABC usually depends strongly on the selection and weighting of appropriate summary statistics. The neural network-based method proposed here bypasses these steps making ABC inference easier to implement and possible to expand to more complex problems, where formulating the correct statistics requires extensive experimentation and domain knowledge. An improvement was noted when the weighting of the summary statistics was automated through linear regression, but this method

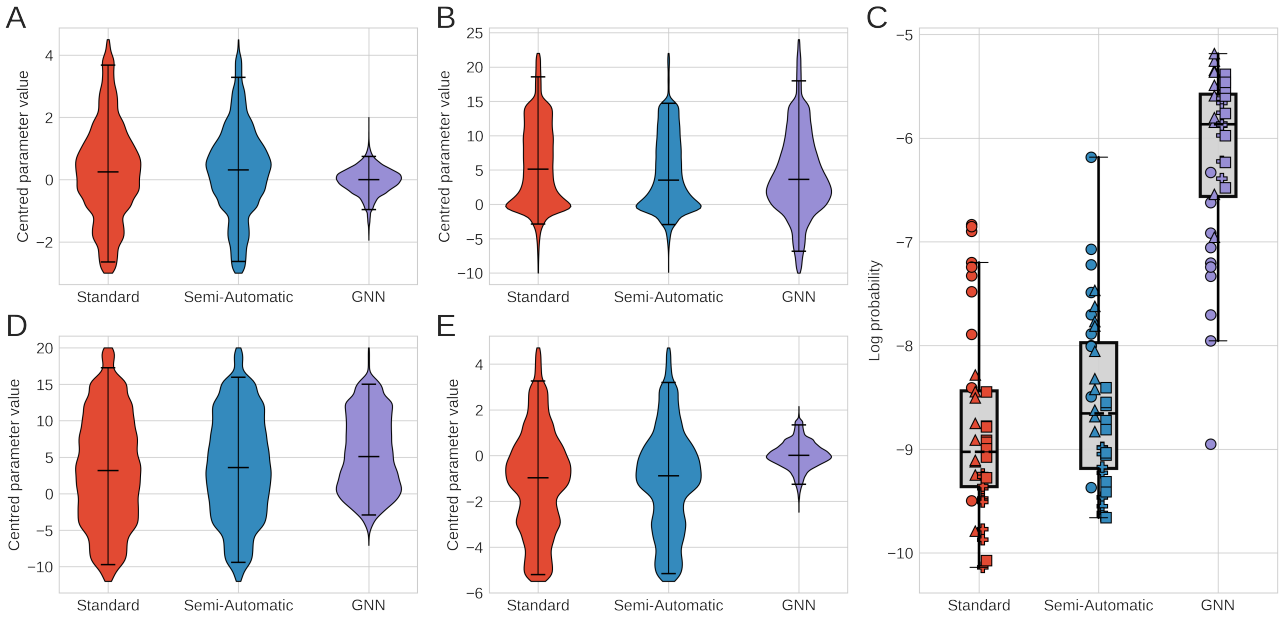

Figure 6. Comparing inferred and true values for four-parameter inference. Inference for four different parameter combinations are shown: $\{l_r = 0.5, l_{al} = 1, l_{at} = 9, v_a = 1.5\pi\}$, $\{l_r = 2, l_{al} = 3, l_{at} = 12, v_a = \pi\}$, $\{l_r = 3, l_{al} = 10, l_{at} = 10, v_a = 0.5\pi\}$, $\{l_r = 1, l_{al} = 10, l_{at} = 5, v_a = 1.75\pi\}$. 10 replicas are performed for each parameter set where each replica involves creating synthetic data and inferring model parameters. (A) Combined violin plot for centred MCMC samples of the repulsion radius l_r for each method. A value closer to zero indicates better performance, and a smaller distribution represents reduced uncertainty. (B) Combined violin plot for centred MCMC samples of the alignment radius l_{al} for each method. (C) The log probability of the true parameter value given the posterior distribution, calculated using a kernel density estimator. A value of each replica is shown with shapes indicating the parameter set, with circles, triangles, $+$-symbols, and squares corresponding to each parameter set listed above respectively. Larger values of log probability indicate the best performance. (D) Combined violin plot for centred MCMC samples of the attraction radius l_{at} for each method. (E) Combined violin plot for centred MCMC samples of the visual angle v_a for each method.

still depended upon effective potential summary statistics being designed a priori.

The neural network method performed well in areas where all other methods suffered. These areas of parameter space were difficult due to either the unidentifiability of parameters given the summary statistics or the non-linear relationship between model parameters and macroscale behaviour. The graph neural network allowed for a more flexible approach to these situations and was able to capture the non-linear effects of varying parameters as well as complex dynamics such as fission-fusion processes.

Previous work has shown how neural networks can be employed to automatically extract summary statistics. In [9] a deep fully-connected neural network was shown to outperform a linear regression approach. The use of neural networks was later extended to take advantage of data structure through the use of convolutional neural networks [48], and partially exchangeable networks [49] in the case of Markovian data, as well as incorporating model prediction uncertainty via the use of Bayesian convolutional networks [50]. However, these neural network architectures are unable to take advantage of the complex structure present in data associated with interacting systems. Graph neural networks (GNNs) are naturally an excellent tool for representing complex systems composed of multiple constituent components as they capture the interactions between individuals, treated as nodes, by encoding information into edge features [34]. Thus, our approach takes advantage of the known capacity of deep neural networks to deal with nonlinearity [51], and the ability of GNNs to faithfully represent interactions among entities. Once the GNN is trained, inference can be performed rapidly with low computational expense.

The GNN-based ABC method presented here can be applied to a wide range of systems, particularly in biology, across collective cell movement, cancer modelling, embryogenesis and morphogenesis, as well as collective animal movement. The increased performance of the GNN-based method is expected to become even more pronounced as model complexity or the number of parameters for inference increases, thus expanding the domain of applicability of simulation-based approximate inference. Finally, the use of GNNs is not restricted to the GP-accelerated ABC scheme we employ here but may be applied to any simulation-based inference scheme that employs the use of summary statistics such as ABC-SMC methods or density estimation.

Data Accessibility. Code will be available on GitHub

Authors' Contributions. J.G.: conceptualization, formal analysis, investigation, software, visualization, writing—original draft and writing—review and editing; C.T.: conceptualization, formal analysis, investigation, supervision, software, visualization, writing—original draft and writing—review and editing, N.C.: conceptualization and investigation; D.H.: conceptualization, funding acquisition, supervision, writing—review and editing, J.M.: conceptualization, funding acquisition, supervision, writing—review and editing. All authors gave final approval for publication and agreed to be held accountable for the work performed therein.

Competing Interests. We declare we have no competing interests.

Funding. This work was supported by the Leverhulme Foundation (RPG-2018-398). JMM was supported by a Leverhulme Visiting Professorship (VP2-2018-0630). CJT is supported by a James S. McDonnell Foundation Studying Complex Systems Scholar Award. DH is supported by the Engineering and Physical Sciences Research Council (EPSRC), grant reference number EP/T017899/1.

References

[1] A. Be'er and G. Ariel, "A statistical physics view of swarming bacteria", Movement ecology **7**, 1–17 (2019).
[2] A. S. Gard-Murray and Y. Bar-Yam, "Complexity and the limits of revolution: what will happen to the Arab Spring?", in *Conflict and Complexity* (Springer, 2015), pages 281–292.
[3] K. Cranmer, J. Brehmer, and G. Louppe, "The frontier of simulation-based inference", Proceedings of the National Academy of Sciences **117** (2020).
[4] D. B. Rubin, "Bayesianly justifiable and relevant frequency calculations for the applied statistician", The Annals of Statistics, 1151–1172 (1984).
[5] P. J. Diggle and R. J. Gratton, "Monte Carlo methods of inference for implicit statistical models", Journal of the Royal Statistical Society: Series B (Methodological) **46**, 193–212 (1984).
[6] S. A. Sisson, Y. Fan, and M. Beaumont, *Handbook of approximate Bayesian computation* (CRC Press, 2018).
[7] M. A. Beaumont, "Approximate Bayesian Computation in Evolution and Ecology", Annual Review of Ecology, Evolution, and Systematics **41**, 379–406 (2010).
[8] P. Fearnhead and D. Prangle, "Constructing summary statistics for approximate Bayesian computation: semi-automatic approximate Bayesian computation", Journal of the Royal Statistical Society. Series B (Statistical Methodology) **74**, 419–474 (2012).
[9] B. Jiang, T.-Y. Wu, C. Zheng, and W. H. Wong, "Learning summary statistic for Approximate Bayesian Computation via Deep Neural Network", Statistica Sinica **27**, 1595–1618 (2017).
[10] T. Vicsek, A. Czirók, E. Ben-Jacob, I. Cohen, and O. Shochet, "Novel type of phase transition in a system of self-driven particles", Physical review letters **75**, 1226 (1995).
[11] H. Atmanspacher, "On macrostates in complex multi-scale systems", Entropy **18**, 426 (2016).
[12] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning* (MIT press, 2016).
[13] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning", Nature **521**, 436–444 (2015).
[14] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition", Neural computation **1**, 541–551 (1989).
[15] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data", IEEE Signal Processing Magazine **34**, 18–42 (2017).
[16] P. Battaglia et al., "Relational inductive biases, deep learning, and graph networks", arXiv (2018).
[17] A. O'Hagan, "Bayesian analysis of computer code outputs: A tutorial", Reliability Engineering & System Safety **91**, 1290–1300 (2006).
[18] R. Wilkinson, "Accelerating ABC methods using Gaussian processes", Proceedings of the 17th International Conference on Artificial Intelligence and Statistics (AISTATS), Reykjavik, Iceland (2014).
[19] D. Rezende and S. Mohamed, "Variational inference with normalizing flows", in International conference on machine learning (PMLR, 2015), pages 1530–1538.
[20] P. Marjoram, J. Molitor, V. Plagnol, and S. Tavaré, "Markov chain Monte Carlo without likelihoods", Proceedings of the National Academy of Sciences **100**, 15324–15328 (2003).
[21] J.-M. Marin, P. Pudlo, C. P. Robert, and R. J. Ryder, "Approximate Bayesian computational methods", Statistics and Computing **22**, 1167–1180 (2012).
[22] M. Beaumont, W. Zhang, and D. Balding, "Approximate Bayesian Computation in population genetics", Genetics **162**, 2025–2035 (2002).
[23] P. Marjoram, J. Molitor, V. Plagnol, and S. Tavaré, "Markov chain Monte Carlo without likelihoods", Proc. Natl. Acad. Sci. USA **100**, 15324–15328 (2003).
[24] T. Toni, D. Welch, N. Strelkowa, A. Ipsen, and P. Stumpf, "Approximate Bayesian Computation scheme for parameter inference and model selection in dynamical systems", Interface: Focus **6**, 187–202 (2008).
[25] D. Prangle, "Adapting the ABC distance function", Bayesian Analysis **12**, 289–309 (2017).
[26] M. C. Kennedy and A. O'Hagan, "Bayesian calibration of computer models", Journal of the Royal Statistical Society, Series B **63**, 425–464 (2001).

[27]J. Gaskell, N. Campioni, J. M. Morales, D. Husmeier, and C. J. Torney, "Approximate Bayesian inference for individual-based models with emergent dynamics", Proceedings of the 2nd International Conference on Statistics: Theory and Applications (ICSTA), ISBN: 978-1-927877-68–5 (2020).
[28]D. Prangle, "Adapting the ABC Distance Function", Bayesian Analysis **12**, 289–309 (2017).
[29]A. Saltelli, "Making best use of model evaluations to compute sensitivity indices", Computer Physics Communications **145**, 280–297 (2002).
[30]A. Borowska, D. Giurghita, and D. Husmeier, "Gaussian process enhanced semi-automatic approximate Bayesian computation: parameter inference in a stochastic differential equation system for chemotaxis", Journal of Computational Physics **429** (2021).
[31]F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model", IEEE transactions on neural networks **20**, 61–80 (2008).
[32]A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. W. Battaglia, "Learning to simulate complex physics with graph networks", Proceedings of the 37 th International Conference on Machine Learning **119** (2020).
[33]C. Sun, P. Karlsson, J. Wu, J. B. Tenenbaum, and K. Murphy, "Stochastic prediction of multi-agent interactions from partial observations", arXiv preprint arXiv:1902.09641 (2019).
[34]J. B. Maguire, D. Grattarola, V. K. Mulligan, E. Klyshko, and H. Melo, "XENet: Using a new graph convolution to accelerate the timeline for protein design on quantum computers", PLoS Computational Biology **17** (2021).
[35]J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry", in International conference on machine learning (PMLR, 2017), pages 1263–1272.
[36]D. Grattarola, D. Zambon, F. M. Bianchi, and C. Alippi, "Understanding pooling in graph neural networks", arXiv preprint arXiv:2110.05292 (2021).
[37]S. Kerman, D. Brown, and M. A. Goodrich, "Supporting human interaction with robust robot swarms", in 2012 5th International Symposium on Resilient Control Systems (IEEE, 2012), pages 197–202.
[38]C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model", in Proceedings of the 14th annual conference on Computer graphics and interactive techniques (1987), pages 25–34.
[39]I. D. Couzin, J. Krause, R. James, G. D. Ruxton, and N. R. Franks, "Collective memory and spatial sorting in animal groups", Journal of theoretical biology **218**, 1–12 (2002).
[40]K. Tunstrøm, Y. Katz, C. C. Ioannou, C. Huepe, M. J. Lutz, and I. D. Couzin, "Collective States, Multistability and Transitional Behavior in Schooling Fish", PLoS Computational Biology **9** (2013).
[41]A. Kolpas, J. Moehlis, and I. G. Kevrekidis, "Coarse-grained analysis of stochasticity-induced switching between collective motion states", PNAS **104**, 5931–5935 (2007).
[42]R. G. McClarren, *Uncertainty Quantification and Predictive Computational Science* (Springer, 2018).
[43]C. E. Rasmussen and C. K. I. Williams, "Gaussian Processes for Machine Learning", MIT Press, ISBN 026218253X (2006).
[44]GPy, *GPy: A Gaussian process framework in python*, http://github.com/SheffieldML/GPy, since 2012.
[45]H. Haario, E. Saksman, and J. Tamminen, "An adaptive Metropolis algorithm", Bernoulli **7**, 223–242 (2001).
[46]D. Grattarola and C. Alippi, "Graph neural networks in TensorFlow and keras with spektral [application notes]", IEEE Computational Intelligence Magazine **16**, 99–106 (2021).
[47]D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization", arXiv preprint arXiv:1412.6980 (2014).
[48]M. Akesson, P. Singh, F. Wrede, and A. Hellander, "Convolutional Neural Networks as Summary Statistics for Approximate Bayesian Computation", IEEE/ACM Transactions on Computational Biology and Bioinformatics **10**, 1 (2021).
[49]S. Wiqvist, P.-A. Mattei, U. Picchini, and J. Frellsen, "Partially Exchangeable Networks and Architectures for Learning Summary Statistics in Approximate Bayesian Computation", Proceedings of the 36th International Conference on Machine Learning, PMLR **97**, 6798–6807 (2019).
[50]F. Wrede, R. Eriksson, R. Jiang, L. Petzold, S. Engblom, A. Hellander, and P. Singh, "Robust and integrative Bayesian neural networks for likelihood-free parameter inference", 2022 International Joint Conference on Neural Networks (IJCNN), 1–10 (2022).
[51]T. Bouwmans, S. Javed, M. Sultana, and S. K. Jung, "Deep Neural Network Concepts for Background Subtraction: A Systematic Review and Comparative Evaluation", Neural Networks **00** (2019).