PLOS COMPUTATIONAL BIOLOGY

RESEARCH ARTICLE

The use of mixture density networks in the emulation of complex epidemiological individual-based models

Christopher N. Davis[1,2], **T. Deirdre Hollingsworth**[3], **Quentin Caudron**[4], **Michael A. Irvine**[4,5]*

1 MathSys CDT, Mathematics Institute, University of Warwick, Coventry, United Kingdom, **2** Zeeman Institute (SBIDER), University of Warwick, Coventry, United Kingdom, **3** Big Data Institute, Li Ka Shing Centre for Health Information and Discovery, Nuffield Department of Medicine, University of Oxford, Oxford, United Kingdom, **4** Scai Analytics Ltd., Vancouver, Canada, **5** Institute of Applied Mathematics, University of British Columbia, Vancouver, Canada

* m.irvine@math.ubc.ca

Citation: Davis CN, Hollingsworth TD, Caudron Q, Irvine MA (2020) The use of mixture density networks in the emulation of complex epidemiological individual-based models. PLoS Comput Biol 16(3): e1006869. https://doi.org/10.1371/journal.pcbi.1006869

Editor: Derek Cummings, University of Florida, UNITED STATES

Received: February 12, 2019

Accepted: February 20, 2020

Published: March 16, 2020

Data Availability Statement: The authors confirm that all data underlying the findings are fully available without restriction. The open-access Python library for the method along with example code and notebooks to reproduce the examples given in the study can be found at the following url: https://github.com/QCaudron/pydra. All other relevant data are within the paper and its supporting information files.

Funding: The authors gratefully acknowledge funding of the Children's Investment Fund

Abstract

Complex, highly-computational, individual-based models are abundant in epidemiology. For epidemics such as macro-parasitic diseases, detailed modelling of human behaviour and pathogen life-cycle are required in order to produce accurate results. This can often lead to models that are computationally-expensive to analyse and perform model fitting, and often require many simulation runs in order to build up sufficient statistics. Emulation can provide a more computationally-efficient output of the individual-based model, by approximating it using a statistical model. Previous work has used Gaussian processes (GPs) in order to achieve this, but these can not deal with multi-modal, heavy-tailed, or discrete distributions. Here, we introduce the concept of a mixture density network (MDN) in its application in the emulation of epidemiological models. MDNs incorporate both a mixture model and a neural network to provide a flexible tool for emulating a variety of models and outputs. We develop an MDN emulation methodology and demonstrate its use on a number of simple models incorporating both normal, gamma and beta distribution outputs. We then explore its use on the stochastic SIR model to predict the final size distribution and infection dynamics. MDNs have the potential to faithfully reproduce multiple outputs of an individual-based model and allow for rapid analysis from a range of users. As such, an open-access library of the method has been released alongside this manuscript.

Author summary

Infectious disease modellers have a growing need to expose their models to a variety of stakeholders in interactive, engaging ways that allow them to explore different scenarios. This approach can come with a considerable computational cost that motivates providing a simpler representation of the complex model. We propose the use of mixture density networks as a solution to this problem. MDNs are highly flexible, deep neural network-based models that can emulate a variety of data, including counts and over-dispersion.

Foundation. CND also acknowledges funding from the EPSRC/MRC via the MAthSys Centre for Doctoral Training. The funder had no role in the study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing interests: The authors MI and QC declare that they are members of the data science consultancy Scai Analytics Ltd. All other authors declare that they have no competing interests.

We explore their use firstly through emulating a negative binomial distribution, which arises in many places in ecology and parasite epidemiology. Then, we explore the approach using a stochastic SIR model. We also provide an accompanying Python library with code for all examples given in the manuscript. We believe that the use of emulation will provide a method to package an infectious disease model such that it can be disseminated to the widest audience possible.

This is a *PLOS Computational Biology* Methods paper.

Introduction

Complex individual-based models abound in epidemiology. These can often include a mixture of different distributions leading to a high-dimensional output space with possibly multi-modal distributions. As an example, consider the relatively simple stochastic Susceptible-Infected-Recovered (SIR) model [1]. In the SIR model, individuals become infected stochastically with probability depending on the number of infected individuals in the population. Once infected, they contribute to further transmissions, until they eventually recover with immunity. For some parameter regimes, depending on initial conditions and stochastic variation, a large outbreak (epidemic) may be observed, or alternatively, the infected individuals may recover before a large-scale epidemic can occur (stochastic fade-out) [2]. Thus, we would see a multi-modal distribution of the cumulative number of infected individuals depending on whether the outbreak results in an epidemic or stochastic fade-out.

Other examples of complex, individual-based models occur in macro-parasitic diseases. These diseases will often have highly heterogeneous parasite distributions amongst its hosts, which therefore requires explicit individual hosts to be modelled in order to directly understand the consequences and implications of these distributions [3]. Compounding this, many of these diseases are controlled through mass drug administration, where coverage, adherence and demographic factors can play a role in the outcome of a program [4, 5].

Finally, there are many examples of individual-based models for sexually-transmitted infectious disease, such as HIV, HPV, gonorrhea, and syphilis [6]. Here, complexities such as heterogeneity in risk, partnership-formation, sexual contact networks, sero-sorting behaviour, and heterogeneity in interventions can lead to dramatically different outcomes that require modelling.

Coupled with the increasing number of computationally-expensive models is the move towards models that are more accessible, such that non-experts can explore the key concepts and outputs. There has been an increasing call for more models to be outward facing to be used by policy makers and other non-modellers [7]. Despite this, there remains significant technological barriers to be able to perform this in general, often requiring skill in multiple programming languages and software development [8]. In particular, one of the technological barriers is the speed at which model simulations of a given scenario can occur. Thus, this introduces the idea of using emulation in lieu of model computation [9–13], where we replace the individual-based model with a statistical model that is more computationally efficient to sample from. These statistical models can be difficult to train and often require the assumption that the model produces unimodal or normal outputs [14].

As individual-based models become more complex, the necessary computational costs increase. This can often lead to only a small number of scenarios being explored with relatively few replicates used to estimate uncertainty within the model. Inference schemes, like approximate Bayesian computation, where many simulation runs need to be performed, will have a particularly high computational cost [15]. Computational speed-up can be performed by making certain approximations within the model, such as taking the deterministic limit for a process that has relatively large numbers, or, again, through the use of emulation [4, 14].

The main concept of an emulator is to fit a statistical regression model to the inputs and outputs of the individual-based model and then evaluate the computationally much faster model instead [16]. For example, an individual-based model may have m number of inputs $(x_0, \ldots, x_{m-1})$, with a corresponding single output y. If we assume that, for a given set of inputs, the output is normally-distributed, we can then emulate the model using the following linear regression

$$y = \sum_{k=0}^{m-1} \beta_k x_k + \epsilon, \tag{1}$$

where $\epsilon \sim N(0, \sigma^2)$. However, for all but simple individual-based models, the linearity assumption may be too restrictive. The fixed $\beta_k x_k$ terms may be replaced with a Gaussian process (GP), where values can vary across the input space, with the assumption that the closer input points are together, the more correlated will be the outputs. The use of GPs for emulation of epidemiological models has seen several successes [14, 16–19].

Yet, emulating with GPs has a number of disadvantages. Firstly, they assume that the outputs of a model for a given set of inputs are normal. Therefore, they cannot take into account the multi-modality or heavy-tailedness of certain data. There may also be some restrictive assumptions on the smoothness of the correlation between two points in input space. We propose the use of a mixture density network (MDN) to overcome some of these issues. We replace the linear regression component with a neural network that is flexible enough to capture complex relationships and replace the simple normal distribution with a mixture of distributions that provides a more general family of distributions for the model output [20].

The advantage of allowing the model output to be given with a complicated distributional form, however, does mean that a large data set is required to train the model. This may introduce difficulties in the use of MDNs for the most computationally expensive models, as many simulations will be needed to initially obtain this data set, particularly when the input and output dimensions are large and the training data needs to cover this vast space. We note carefully set up GP approaches have been used for emulating models with a high input dimension [21]. Furthermore, GPs can estimate the uncertainty of the emulator, which can then be incorporated in decision support calculations. On the other hand, MDNs do not do this directly and so only by using a data set large enough that the uncertainty introduced by the emulator is small compared to other uncertainties, can MDNs be safely used. A summary of the comparison between GPs and MDNs is shown in Table 1.

The manuscript is segmented as follows: we first introduce the concept of an MDN, and how it relates to an individual-based model, and apply the concept to a number of simple examples to demonstrate its use. We then consider a stochastic SIR model and emulate the final size distribution using an MDN. Finally, we demonstrate its use on estimating the distribution of susceptible and infected individuals in a model with vaccination. All analyses presented within the manuscript were conducted within the package framework and example code is given (see https://github.com/QCaudron/pydra).

Table 1. A comparison of mixture density networks and Gaussian processes.

Mixture density network	Gaussian process
Can emulate multi-modal distributions.	Only suitable for uni-modal distributions.
Flexible output distribution mixture allows for application to different data types, such as overly-dispersed, finite domain or discrete.	Output distribution is normal.
Requires large training set to capture output.	Suitable for small data sets.
Good scaling properties.	Scales poorly with data size.
Hyperparameters need to be tuned in training process.	Hyperparameters need to be tuned in training process.
The training process is stochastic.	For given parameters, the method is optimised exactly.
The uncertainty is hard to quantify.	Directly measures uncertainty.

https://doi.org/10.1371/journal.pcbi.1006869.t001

Methods

Introduction to mixture density networks

Mixture density networks (MDNs) are built from two components: a neural network and a mixture model. We begin by introducing the concept of a mixture model—a model of probability distributions built up from a weighted sum of more simple distributions. More concretely, we consider a one-dimensional distribution with m mixture components. The probability density function (PDF) $p(x)$ is represented by the m PDFs indexed by j $p_j(x)$, with weights $\Omega = \{\omega_0, \ldots, \omega_{m-1}\}$, where $\sum_{j=0}^{m-1} \omega_j = 1$, by the following equation

$$p(x) = \sum_{j=0}^{m-1} \omega_j p_j(x). \tag{2}$$

Typically these probability distributions will be parameterised by a series of parameters that reflect the shape and location of the distribution $\Theta = \{\theta_0, \ldots, \theta_{m-1}\}$. The full parameterised model may therefore be written as

$$p(x|\Omega, \Theta) = \sum_{j=0}^{m-1} \omega_j p_j(x|\theta_j). \tag{3}$$

As an example, each p_j could be a normal distribution parameterised by a mean μ_j and a variance σ_j. The mixture model would then have the following form,

$$p(x|\Omega, \Theta) = \sum_{j=0}^{m-1} \frac{\omega_j}{\sqrt{2\pi\sigma_j^2}} \exp\left(\frac{-1}{2\sigma_j^2}(x-\mu_j)^2\right). \tag{4}$$

In general, these mixture distributions are multi-modal, including those in the form of Eq (4), and can be fitted directly to some data $\mathbf{x} = (x_0, \ldots, x_{n-1})$. Assuming independence, the corresponding likelihood is calculated as

$$l(\mathbf{x}|\Omega, \Theta) = \prod_{i=0}^{n-1} \left[\sum_{j=0}^{m-1} \omega_j p_j(x_i|\theta_j)\right]. \tag{5}$$

Fitting can then typically proceed using expectation–maximisation [20]. For our purposes, we have an individual-based model M, with some input α, that produces stochastic realisations $y \sim M(\alpha)$. We therefore wish to derive a relationship between the input parameters α, and the

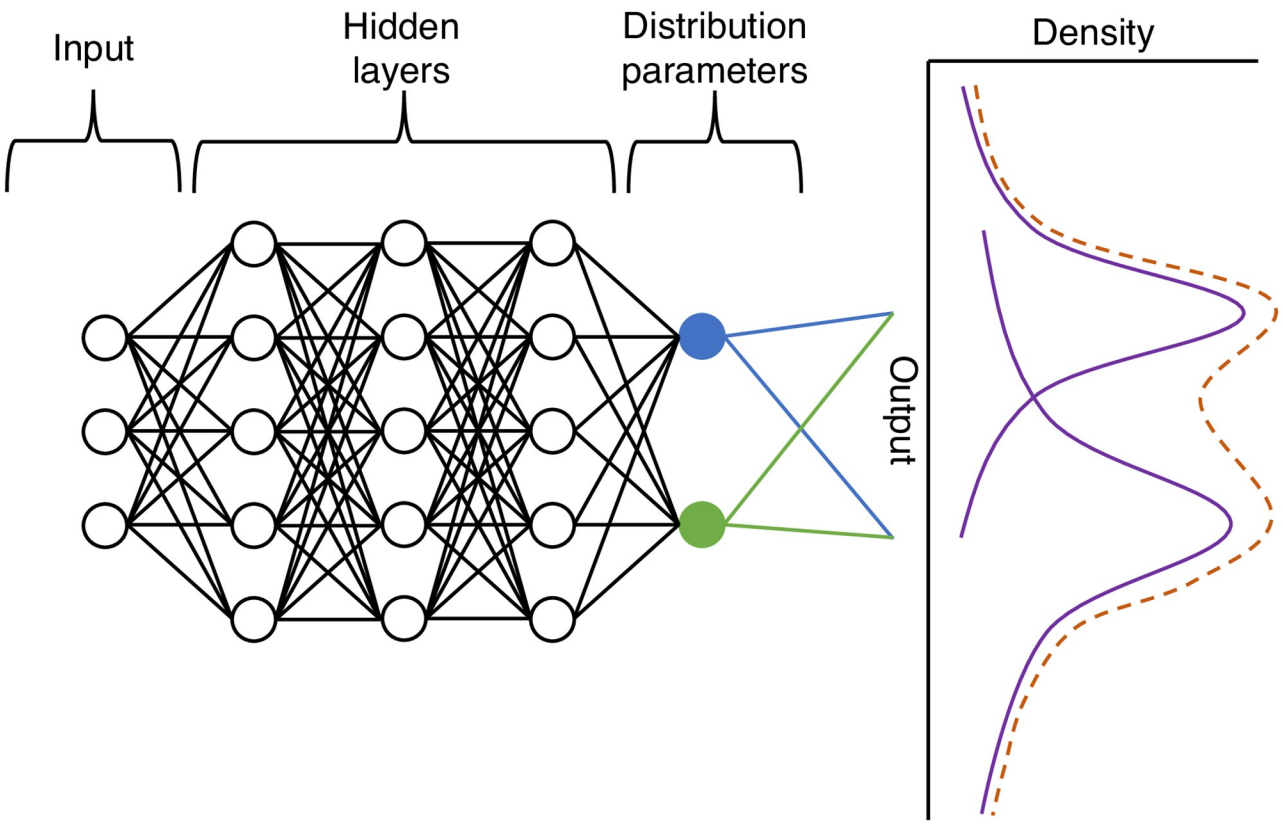

Fig 1. MDN that emulates a model with three inputs and a one-dimensional output with two mixtures. The inputs are passed through two hidden layers, which are then passed on to the normalised neurons, which represent the parameters of a distribution and its weights e.g. the mean (shown in blue) and variance (shown in green) of a normal distribution. These parameters are used to construct a mixture of distributions (represented as a dashed line).

https://doi.org/10.1371/journal.pcbi.1006869.g001

mixture density weights $\omega_j(\alpha)$ and density parameterisations $\theta_j(\alpha)$. This could potentially be done with a separate regression for each of the density parameters and weights, although this would fail to capture the corresponding relationships that would exist between each parameter and weight. We can therefore model these using a neural network, which is able to provide flexible fitting for arbitrarily complex relationships by the universal approximation theorem [22]. An MDN is therefore defined as a mixture model, where the mixture components are modelled using a neural network.

Fig 1 provides an overview of the MDN construction. The inputs of the model α are initially fed into the MDN (three such inputs in the example diagram). These are then passed through a number of hidden layers in the neural network, which provide a compact representation of the relationship between the inputs and the unnormalised inputs into the mixture model. These distribution parameters are then passed through a normalisation layer; the weights of the mixture are transformed such that they sum to one and the shape parameters are transformed so that they are positive. These parameters are used to construct the mixture model, where one can draw samples or calculate statistics, such as mean and variance, for a given input. For multiple outputs the final layer can be copied with independent parameters for the number of outputs being considered. Note that a number of aspects of the MDN need to be specified including the number of input parameters, the dimension of the output, the distributions used in the mixture density, and the number and size of the hidden layers.

The MDN can then be fit to the following objective loss function, which is equivalent to maximising the likelihood given in Eq (5),

$$f(\mathbf{x}|\Omega,\Theta) = -\sum_{i=0}^{n-1} \log\left(\sum_{j=0}^{m-1} \omega_j p_j(x_i|\theta_j)\right). \quad (6)$$

Note that provided p_j is differentiable with respect to θ_j, this loss represents a differentiable function. Standard techniques based on stochastic gradient descent can then be applied in order to optimise the weights of the network with respect to this loss [23].

Performance on a simple model

In order to examine how a fitted MDN can capture the broad statistical properties of a distribution, where the underlying mixture distributions differ significantly from the true distribution, we explored fitting to a negative binomial model. The negative binomial can be parameterised by a mean m and a shape parameter k using the following probability mass function,

$$f(x \mid m,k) = \frac{\Gamma(x+k)}{x!\Gamma(k)}(k/(m+k))^k(m/(m+k))^x. \quad (7)$$

The parameter m defines the mean of the distribution and the shape parameter k controls the heterogeneity of the distribution, where the variance is $m(1 + m/k)$. As k goes to infinity, the distribution approaches the Poisson distribution.

An MDN was fitted to the negative binomial distribution in the following way. An MDN with 20 gamma mixtures with 3 dense layers of 64 neurons was constructed. Data were sampled as the random output from 1,000 (m, k) pairs, which were uniformly taken at random from m with range 0–100 and from k with range 0.01–5. Hence, one data point is a (m, k) pair and an associated sample output from the negative binomial distribution. The fitting was performed for 150 epochs with a batch size of 50. One epoch is defined as one forward and backward pass through all data points, and the batch size is the number of data points used in a single iteration.

In order to compare the statistical properties between the true negative binomial distribution and the MDN emulator a number of tests were devised. First the mean and variance of each distribution were compared by fixing one of the parameters to the mid-point of the parameter range and varying the other parameter (Fig 2A and 2B). In order to statistically compare between a sample generated from the true process and generated from an emulator, the two-sided Kolmogorov–Smirnov (K–S) test was performed on two samples of size 100 across a range of input values for 100 replicates (Fig 2C) [24]. Although, we expect for more complex models the emulated and simulated distributions to exactly match the K–S test provides a quantifiable measure of how close these distributions align and also provides a direct statistic where regions of parameter space give poor emulator performance. The true cumulative density function (CDF) and the empirical CDFs were also compared (Fig 2D).

This experiment broadly captures how well the MDN can emulate a distribution significantly different to its underlying mixture distribution, as well as how capable it is to adequately deal with highly heterogeneous data, which can complicate model fitting [15]. Example code using the accompanying open-source library is given in Box 1.

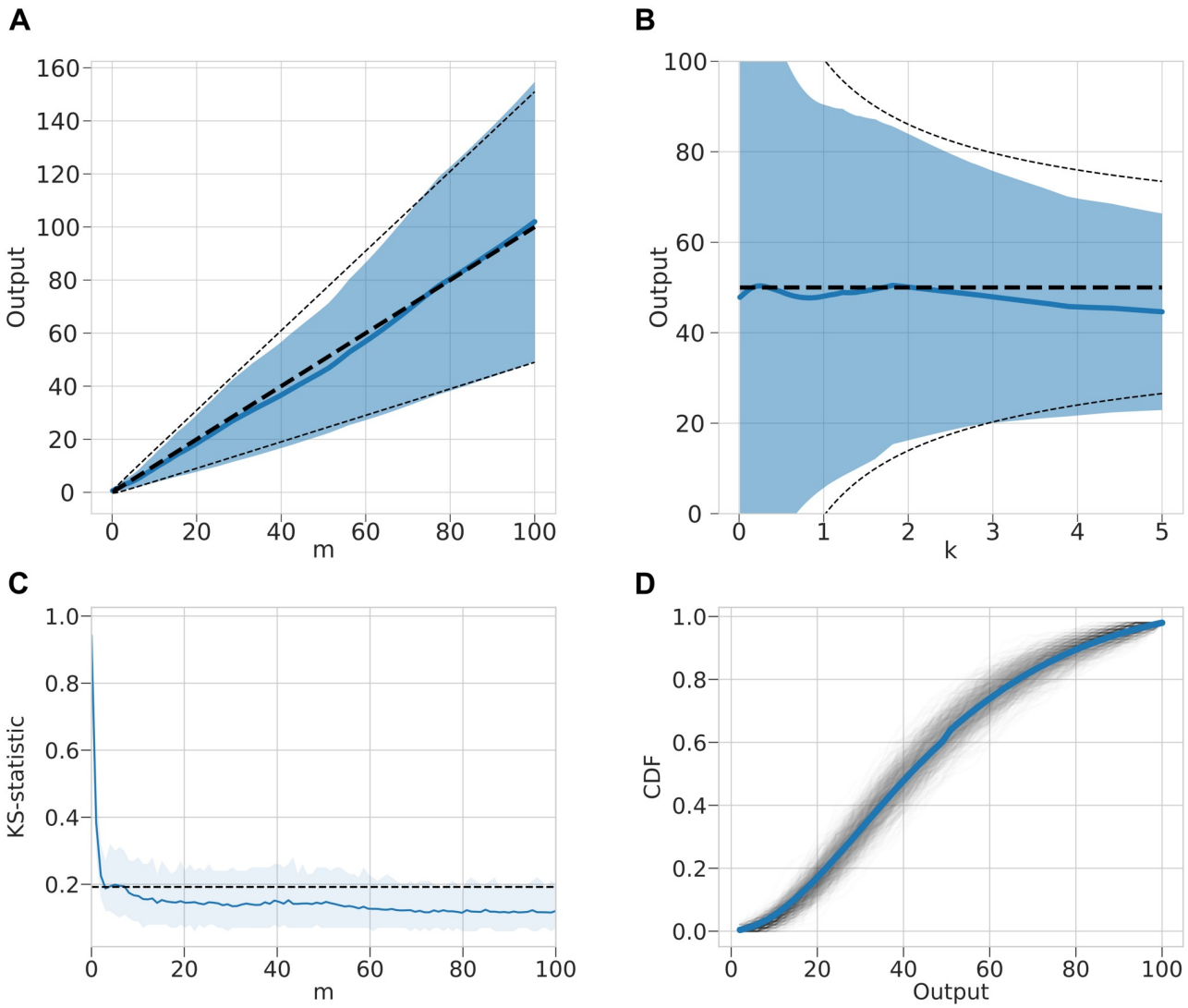

Fig 2. Gamma-MDN output emulating a negative binomial model. (A) For fixed shape parameter $k = 2.5$, the distribution of output from MDN is shown in blue (mean = solid line, variance = shaded region), the theoretical values are shown as a black dashed line (mean = bold line, variance = normal line). (B) For fixed mean parameter $m = 50$, the distribution of output from MDN over a range of k values is shown in blue (mean = solid line, variance = shaded region), the theoretical values are shown as a black dashed line (mean = bold line, variance = normal line). (C) Corresponding two-sample K–S statistic where sample of 100 points are drawn from a negative binomial and the MDN over a range of m values. 100 replicates are used to estimate a mean K–S statistic and a 95% range. The dashed line represents significance at $\alpha = 0.05$, with values less than this indicating that the two samples do not differ significantly. (D) Example empirical CDFs drawn from 100 samples of MDN with inputs $m = 50$ and $k = 2.5$. 1,000 empirical CDFs are shown as black transparent lines and true CDF is shown as a blue solid line.

https://doi.org/10.1371/journal.pcbi.1006869.g002

Stochastic SIR model

The stochastic susceptible–infected–recovered (SIR) model is a standard epidemiological model for the infection dynamics of an epidemic. It is a Markov process where each individual of a population of fixed size N can be in any one of the three states: S, I or R. Due to the stochasticity of the model, the output is multi-modal, where identical input parameters can have both quantitatively and qualitatively different outputs, depending on whether an epidemic

Box 1. Code for negative binomial distribution example

```
import pydra #import MDN emulator library
x = input_data() # import input data with shape (# data points, # inputs)
y = ouput_data() # import output data with shape (# data points,)

'''
construct an emulator with 20 mixtures, three layers with 64 neurons and
    two inputs with one output that has a gamma mixture distribution
'''
model = pydra.Pydra(cluster_size=20,output_size=1,layers=3,input_size=2,
                    dense_layer_size=64,output_distributions=['Gamma'],
    print_summary=True)

# fit for 150 epochs with batch size 50
model.fit(data, y, epochs=150, batch_size=50, verbose=1)
```

does or does not occur. The transitions for the process are defined by:

$$(S, I, R) \rightarrow (S-1, I+1, R) \text{ at rate } \beta SI/N,$$

$$(S, I, R) \rightarrow (S, I-1, R+1) \text{ at rate } \gamma I.$$

We generate multiple realisations of the stochastic SIR model and use this data to fit a series of MDNs such that we can evaluate the performance of the emulation model. We note that while there are already fast methods to draw realisations of the process (Gillespie, tau-leap method), we assess this method to understand the benefits for when we want to emulate data from a more computational intensive model [25, 26]. Here, we consider training three different MDNs for different inputs and output distributions.

Firstly, we consider the final size distribution of the epidemic model, which is given as the total number of individuals that have been infected (calculated as $N(\infty) - S(\infty)$). For our data to train an MDN, we take 10,000 realisations of the simulated process using model parameters β, sampled from $U(0, 1)$, and γ, sampled from $U(0.1, 1)$, with a population size of $N = 1,000$, which each give us a final size. To fit an MDN for these two inputs (β and γ) and one output (final size), we choose a mixture of 20 binomial distributions, where the binomial parameter $n = 1,000$ is fixed and p is learnt from the MDN. We use binomial distributions as the final size is integer valued with a maximum value of the population size. We train on a network with 3 dense layers of 64 neurons for 150 epochs with a batch size of 50. From the trained MDN, new inputs for β and γ, give us a distribution of the final size.

Due to the multi-modality, unlike for the negative binomial distribution, calculating the mean and variance of this whole distribution is not a useful concept and so we consider the similarity between the simulated and emulated distributions, by sampling from both (Fig 3A). To quantify their similarity, the two-sided Kolmogorov–Smirnov test was performed on two samples of size 100 (for simulated and emulated data) for randomly sampled β and γ values for 100 replicates (Fig 3B). Furthermore, since the distribution is bi-modal, we compare the proportion of realisations where the final size is greater than 10% of the population size, indicating that an epidemic has occurred (Fig 3C) and we also compare the empirical CDFs (Fig 3D). These tests were performed across a range of $R_0 = \beta/\gamma$ values.

Secondly, we explored the infection dynamics across time. We take 10,000 realisations of the simulated process using model parameters: β, sampled from $U(0, 1)$; γ, sampled from $U(0.1, 1)$; and time t, sampled as a random integer between 1 and 100. The population size remains fixed with $N = 1,000$. An MDN was trained on this data with two outputs, the prevalence of susceptible and infected individuals, for the three inputs of β, γ and time (scaled to be

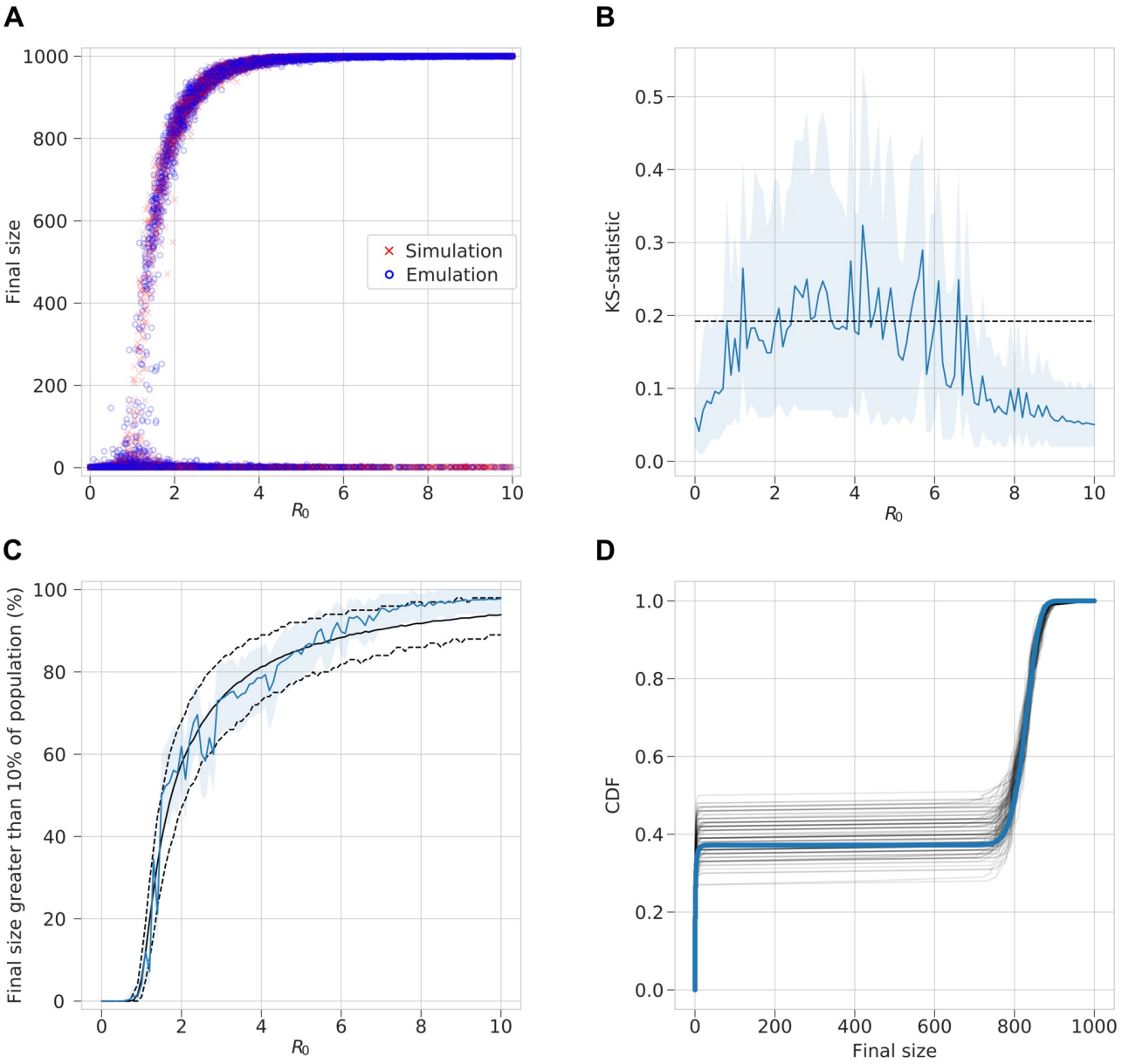

Fig 3. Binomial-MDN output emulating the final size distribution of a stochastic SIR model. (A) For random uniform sampling over β and γ a sample of the output from MDN across values for the basic reproductive number $R_0 = \beta/\gamma$ are shown in blue and the directly simulated values are shown in red. (B) Corresponding two-sample K–S statistic where sample of 100 points are drawn from a negative binomial and the MDN over a range of R_0 values. 100 replicates are used to estimate a mean K–S statistic and a 95% range. Dashed line represent significance at $\alpha = 0.05$, with values less indicating the two samples do not differ significantly. (C) The percentage of 1,000 realisations of the stochastic SIR model with final size greater than 100 is shown in black with dashed line showing a 95% range. Emulated results are shown by the blue line with a 95% range. (D) Example empirical CDFs drawn from 100 samples of MDN with inputs $\beta = 0.4$ and $\gamma = 0.2$. 1,000 empirical CDF are shown as black transparent lines and true CDF is shown as a blue solid line.

https://doi.org/10.1371/journal.pcbi.1006869.g003

between 0.01 and 1). The prevalence of recovered individuals can be inferred from this given the fixed population size. Since the aim was to learn the prevalence rates, the number in each compartment divided by the population size, we use a mixture of 20 beta distributions, as the beta distribution is defined on the interval [0,1]. The MDN has 3 dense layers of 64 neurons and is trained for 50 epochs with a batch size of 50. For given values of β, γ and t, the trained MDN gives us two output distributions for the number of susceptible and infected people. The

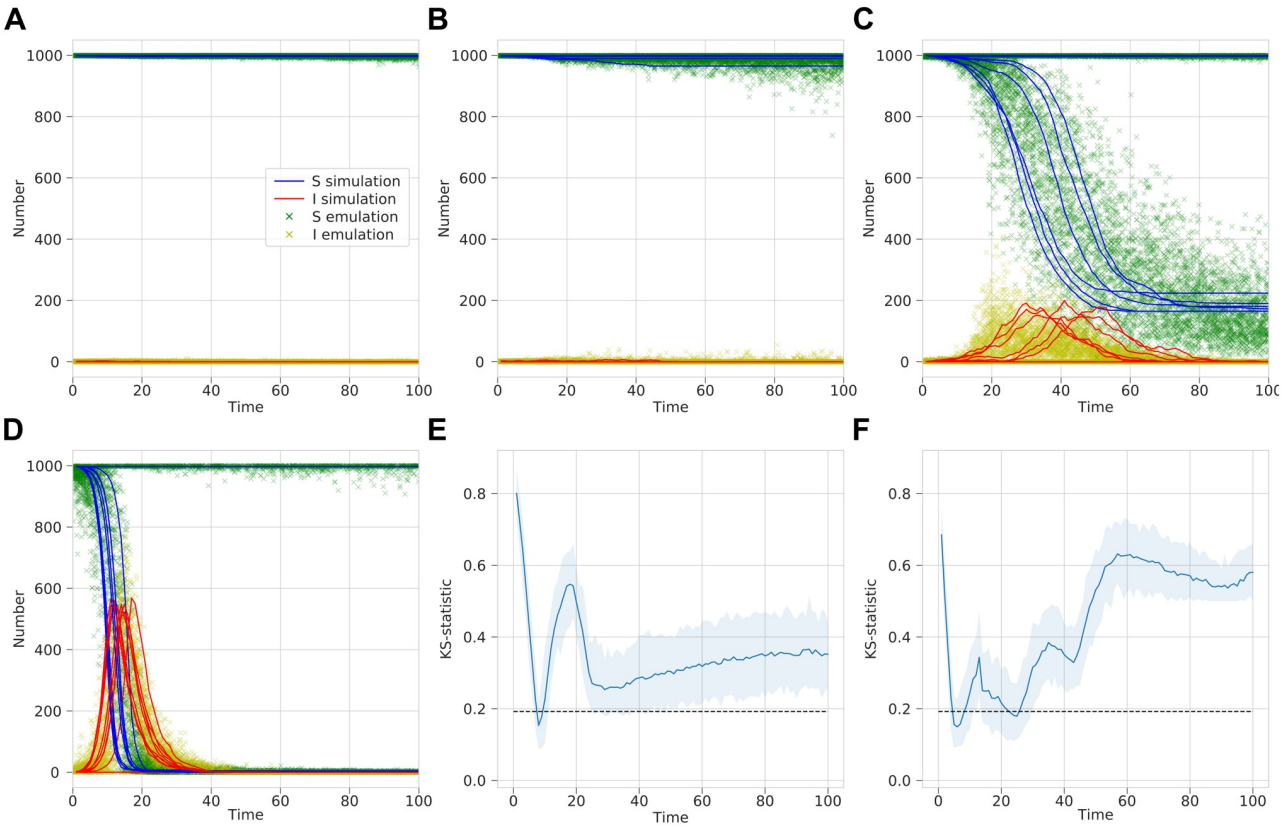

Fig 4. Beta-MDN output emulating the infection dynamics with time for a stochastic SIR model. (A–D) A comparison of simulation results with sampled MDN output for fixed $\gamma = 0.2$ and $N = 1,000$ and different β values that give the following R_0 values: (A) $R_0 = 0.5$, (B) $R_0 = 1.0$, (C) $R_0 = 2.0$, and (D) $R_0 = 5.0$. (E–F) Two-sample K–S statistic where sample of 100 points are drawn from a negative binomial and the MDN over a range of time t values. 100 replicates are used to estimate a mean K–S statistic and a 95% range. Dashed line represent significance at $\alpha = 0.05$, with values less indicating the two samples do not differ significantly. Tests are for (E) number of susceptible people and (F) number of infected people.

https://doi.org/10.1371/journal.pcbi.1006869.g004

simulated and emulated distributions are compared and a two-sample K–S test was performed on the output (Fig 4).

Finally, we added an additional transition for the process to model vaccination of susceptible individuals, which is given by

$$(S, I, R) \rightarrow (S - 1, I, R + 1) \text{at rate } \delta S.$$

We then used the same method as before, with the 2 extra inputs of the vaccination rate δ sampled from $U(0, 0.01)$ and population size sampled as a random integer between 1 and 1,000 (both linearly scaled to have a maximum of 1). Thus, using the trained MDN, given inputs β, γ, t, δ and N, we output the emulated distribution of the number of susceptible and infected people. Again, the simulation and emulation was compared and a K–S test performed (Fig 5).

Results

Simple model

The fitted gamma-MDN emulator was able to broadly capture the mean and variance of the distribution over a range of inputs parameters (Fig 2A and 2B). There were some notable deviations to these statistics however, where the parameters were near the edge of the range. In

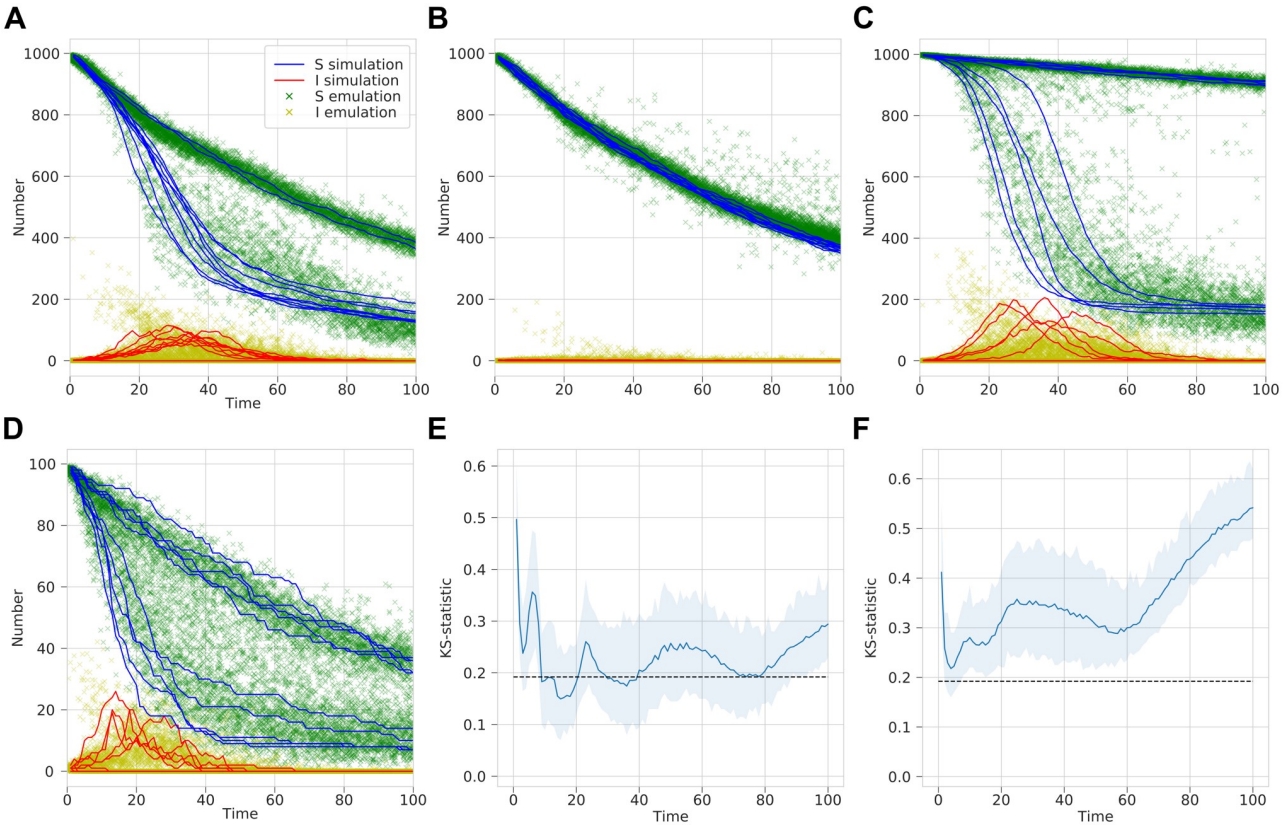

Fig 5. Beta-MDN output emulating the infection dynamics with time for a stochastic SIR model. (A–D) A comparison of simulation results with sampled MDN output for fixed $\gamma = 0.2$ and different β, δ and N values such that (A) $R_0 = 2.0$, $\delta = 0.01$ and $N = 1,000$, (B) $R_0 = 1.0$, $\delta = 0.01$ and $N = 1,000$, (C) $R_0 = 2.0$, $\delta = 0.001$ and $N = 1,000$, (D) $R_0 = 2.0$, $\delta = 0.01$ and $N = 100$. (E–F) Two-sample K–S statistic where sample of 100 points are drawn from a negative binomial and the MDN over a range of time t values. 100 replicates are used to estimate a mean K–S statistic and a 95% range. Dashed line represent significance at $\alpha = 0.05$, with values less indicating the two samples do not differ significantly. Tests are for (E) number of susceptible people and (F) number of infected people.

https://doi.org/10.1371/journal.pcbi.1006869.g005

particular, where $k < 1$, both the mean and variance begin to differ significantly from the true distribution, where the variance increases to infinity as k goes to zero (Fig 2B).

The K–S statistic is below the significance level over a broad range of m values indicating a sample drawn from the true process and the emulator are similar (Fig 2C). Only when m is close to zero, at the edge of the range of training data, does the distribution differ significantly from the true distribution according to the K–S statistic. This can also be broadly shown by plotting example empirical CDF against the true CDF (Fig 2D).

Stochastic SIR model

The three MDNs were capable of emulating the behaviour of the stochastic SIR model. When sampling from the output distribution of the final size of an epidemic, there was good agreement with the results of sampling from the actual simulation, which is true across the full range of R_0 values (Fig 3A). This is corroborated by the results of the K–S test, where the K–S statistic lies below or close to the significance level defining whether the samples could be drawn from the same distribution, for different R_0 values (Fig 3B). We note that since the distributions are, of course, not exactly the same, drawing more samples would give a weaker result. However, the result was stronger for both small and large R_0 values, with some divergence for the intermediate values, where R_0 is close to 1, where the final size takes a larger

range of values. Comparison of the proportion of samples that exhibit stochastic fade-out, as opposed to the emergence of an epidemic, was determined where the final size reaches at least 10% of the population (Fig 3C). These proportions matched closely, with the emulation only slightly deviating to a higher than expected proportion for large R_0 values. Relatively few samples were close to 10% of the population except where R_0 was close to 1, where the proportion would be more sensitive to the threshold definition. The results also matched for the CDFs of the two distributions, shown for fixed $R_0 = 2$ with $\beta = 0.4$, $\gamma = 0.2$ (Fig 3D).

The addition of emulating the actual infection dynamics against time shows that a single trained MDN captures the complexity of varying R_0 by qualitatively reproducing when an epidemic occurs in the correct timescales (Fig 4A–4D). However, statistically comparing the two output distributions for both susceptible and infected individuals, the K–S statistic increases above the significance level (Fig 4E and 4F). This is in part as the beta distribution is a continuous distribution, whereas the simulated prevalence values are all fractions of the population size $N = 1,000$. The effect of this is particularly stark for small time t values where the number of individuals will always be exactly $S_0 = 999$ and $I_0 = 1$; there will be some variation outside these values in the emulation, as it is not integer valued. Rounding the emulated values or choosing an integer-valued output distribution helped to improve the K–S test results (see S1 Appendix).

Furthermore, the emulation coped well with learning to reproduce the distributions with further model parameters, vaccination rate δ and population size N, since inputting different values for β, γ, δ and N into the trained MDN qualitatively reproduces the infection dynamics of the simulation across all time values t (Fig 5A–5D). With the added complexity of more input parameters, there are some values where the MDN emulation differed from the simulation however, particularly small times ($t < 30$) in the number of infected. Similarly to the MDN without variable vaccination and population size, the results of the K–S test suffer from a similar problem of comparing discrete and continuous distributions, but the resulting distributions were very close at some values (Fig 5E and 5F).

Discussion

Model emulation is quickly becoming an important and necessary method within infectious disease epidemiology due to the increased use of complex, computationally-intensive models, increased use of direct data fitting requiring many model queries, and increased demand for models to be made directly available to knowledge users [27]. We have explored the use of mixture density networks (MDNs) in order to provide a scalable, flexible solution to this type of emulation [28]. These are mixture models where the underlying parameters of the mixture are neural networks. This allows the significant progress in neural networks and deep learning to be incorporated into the emulation. As neural networks allow for flexible memorisation and interpolation, they provide a compact statistical representation of complex data allowing for rapid inferences to be made.

The main alternative to MDNs for the emulation of a stochastic model are Gaussian processes [29]. These represent the outputs of a model as a multivariate normal distribution. This allows for the quantification of uncertainty and for covariance between points in the model's input space. While the typical underlying assumption that the data is normal can make GPs restrictive as to the type of data they can represent, they can instead be used to mimic hyperparameters of the output distribution, rather than the output itself [30], allowing more flexibility. We have demonstrated that an MDN can be applied to both overly-dispersed count data (negative binomial example), as well as bimodal count data with a finite domain (final size distribution example). These examples would be inappropriate to apply a GP to, although a viable

solution could be to introduce a mixture distribution with parameters described with GPs, rather than the neural network.

Our presented examples expose the potential for MDNs to be used more extensively for model emulation and have scope to be extended in many different ways. For example, in emulating the infection dynamics in Fig 5, we use the input parameters of β, γ, δ, N, and t to output values of the number of susceptible and infected people. This means that we use the emulator to predict individual time points when sampled from, rather than a whole time series. A possible limitation of this approach is that the emulator may not correctly estimate where the initial epidemic occurs (see Fig 5). Despite this, there would be scope to train an emulator with the further inputs of the number of susceptible and infected people at the previous time point. This MDN could then be sampled from the model iteratively to produce a whole time series. An additional potential issue is that the presented models do not conserve total population size, with it possible for the emulated output to give a total number of people larger than the population size (although there are bounds within each class being larger than the population size). There could be several solutions if this was considered a problem. For example, the output could instead be the sum of infected and susceptible people, along with the proportion of these people in the infected class, constrained to be between zero and one.

The use of an MDN emulator for a stochastic model are two-fold. As the neural network directly learns parameters of the mixture distributions, these may be used directly in the output by, for example, estimating the mean and variance at each point. The emulator may also use the learned distributions to perform random draws from the emulator representing a realisation of the stochastic process. As the emulator approximates the distribution of the output given model input, this essentially produces a likelihood of the data point given the model parameters. Such a synthetic likelihood could then conceivably be used in a Bayesian inference scheme, such as in a approximate Bayesian computation [15]. It would be interesting to apply this approach where a model likelihood is computationally intractable.

We also note, the training of neural networks can lead to the vanishing/exploding gradient problem [31]. Techniques such as momentum and improved initialisation can help mitigate these issues [32]. Anecdotally, we found that re-initialisation with a smaller learning rate generally resolved issues encountered in training. There are also no clear rules on how many epochs training should last or what batch size to use and so the implementer needs to experiment with these parameters to achieve a good fit. In addition, training of neural networks typically involve a large amount of data. If these can be readily generated from a model then an MDN provides a feasible approach to emulation. However when data is small, either a GP or a simplified neural network based on model summary statistics may be a more appropriate approach.

When model computation is slow or there is a large number of input parameters, a more efficient sampling scheme of the parameter space may be appropriate [33]. Efficient high-dimensional sampling schemes such as entropy maximisation have been implemented previously in GPs [14, 34]. As these techniques also involve an approximated likelihood of the data, they could be readily implemented into an MDN scheme, where learning can be conducted in an online fashion.

It is also important to consider the types of appropriate distributions to emulate the model output. Whether they are discrete (e.g. binomial or Poisson) or have finite support (e.g. binomial) can impact the resulting approximation. For example, using a mixture of normal distributions to describe a finite population would lead to some probability of the population being negative. When the population is small this would be non-negligible (see S1 Appendix). It is therefore important to understand the nature of the data being approximated, for example a final size distribution can be well approximated by a Poisson distribution under certain

conditions [35]. Plotting the emulated and real data, either as their summary statistics or as a point cloud, as was done here, is an important step toward understanding the validity of the emulator approximation. We found that regular checking of these statistics was necessary to ensure the emulator had converged. We also employed the K–S statistic as a measure of similarity between emulated and real data. A limitation with this statistic is as a mixture distribution is only an approximation to the true distribution, we would expect to see worse performance as the number of samples increase. The K–S statistic can therefore be applied to give a relative measure of emulator performance and highlight regions of parameter space that may exhibit a worse approximation.

Neural networks and in particular deep learning has made enormous progress recently, rapidly improving the state-of-the-art in representation of data sets [36]. This has also led to an increase in open software for developing neural networks including Keras and Tensorflow [37, 38]. Building from these allow the rapid development of new emulation models and provide the use of established code for the testing and analysis of the trained models. In companion to this article, we provide an open access Python library to develop an MDN emulator with example notebooks demonstrating its use. The library also provides details on the exportation of a trained emulator into a web application. For more information, see the accompanying Python library: https://github.com/QCaudron/pydra.

Conclusion

Mixture density networks have the potential to be used as emulators for complex epidemiological agent-based and micro-simulation models. These techniques incorporate cutting-edge advances in machine learning that provide the possibility to leverage new software libraries in order to perform fast emulator fitting. Applications can include the building of web interfaces for models as well as in model fitting. We hope this technique will prove useful to the broad epidemiology modelling community and as such have included an accompanying open-source library with examples demonstrating its use.

Supporting information

S1 Appendix. Details on the choice of distribution in mixture density network.
(PDF)

Author Contributions

Conceptualization: T. Deirdre Hollingsworth, Michael A. Irvine.

Formal analysis: Christopher N. Davis, Michael A. Irvine.

Funding acquisition: T. Deirdre Hollingsworth.

Investigation: Christopher N. Davis, Michael A. Irvine.

Methodology: Michael A. Irvine.

Resources: T. Deirdre Hollingsworth.

Software: Quentin Caudron, Michael A. Irvine.

Supervision: Michael A. Irvine.

Validation: Michael A. Irvine.

Visualization: Christopher N. Davis, Michael A. Irvine.

Writing – original draft: Christopher N. Davis, Michael A. Irvine.

Writing – review & editing: Christopher N. Davis, T. Deirdre Hollingsworth, Michael A. Irvine.

References

1. Keeling MJ, Rohani P. Modeling infectious diseases in humans and animals. Princeton University Press; 2011.
2. Britton T, House T, Lloyd AL, Mollison D, Riley S, Trapman P. Five challenges for stochastic epidemic models involving global transmission. Epidemics. 2015; 10:54–57. https://doi.org/10.1016/j.epidem.2014.05.002 PMID: 25843384
3. May RM. Togetherness among schistosomes: its effects on the dynamics of the infection. Mathematical Biosciences. 1977; 35(3-4):301–343. https://doi.org/10.1016/0025-5564(77)90030-X
4. Irvine MA, Reimer LJ, Njenga SM, Gunawardena S, Kelly-Hope L, Bockarie M, et al. Modelling strategies to break transmission of lymphatic filariasis-aggregation, adherence and vector competence greatly alter elimination. Parasites & Vectors. 2015; 8(1):547. https://doi.org/10.1186/s13071-015-1152-3
5. Hollingsworth TD, Adams ER, Anderson RM, Atkins K, Bartsch S, Basáñez MG, et al. Quantitative analyses and modelling to support achievement of the 2020 goals for nine neglected tropical diseases. Parasites & Vectors. 2015; 8(1):630. https://doi.org/10.1186/s13071-015-1235-1
6. McCreesh N, O'Brien K, Nsubuga RN, Shafer LA, Bakker R, Seeley J, et al. Exploring the potential impact of a reduction in partnership concurrency on HIV incidence in rural Uganda: a modeling study. Sexually transmitted diseases. 2012; 39(6):407–413. https://doi.org/10.1097/OLQ.0b013e318254c84a PMID: 22592824
7. Whitty CJ. What makes an academic paper useful for health policy? BMC Medicine. 2015; 13(301).
8. Irvine MA, Hollingsworth TD. Making Transmission Models Accessible to End-Users: The Example of TRANSFIL. PLoS Neglected Tropical Diseases. 2017; 11(2):e0005206. https://doi.org/10.1371/journal.pntd.0005206 PMID: 28151997
9. Keeling MJ, Ross JV. On methods for studying stochastic disease dynamics. Journal of The Royal Society Interface. 2008; 5(19):171–181. https://doi.org/10.1098/rsif.2007.1106
10. Buckeridge DL, Jauvin C, Okhmatovskaia A, Verma AD. Simulation Analysis Platform (SnAP): a tool for evaluation of public health surveillance and disease control strategies. In: AMIA Annual Symposium Proceedings. vol. 2011. American Medical Informatics Association; 2011. p. 161.
11. Willem L, Stijven S, Vladislavleva E, Broeckhove J, Beutels P, Hens N. Active learning to understand infectious disease models and improve policy making. PLoS Computational Biology. 2014; 10(4):e1003563. https://doi.org/10.1371/journal.pcbi.1003563 PMID: 24743387
12. Kasaie P, Mathema B, Kelton WD, Azman AS, Pennington J, Dowdy DW. A novel tool improves existing estimates of recent tuberculosis transmission in settings of sparse data collection. PloS One. 2015; 10(12):e0144137. https://doi.org/10.1371/journal.pone.0144137 PMID: 26679499
13. Willem L, Verelst F, Bilcke J, Hens N, Beutels P. Lessons from a decade of individual-based models for infectious disease transmission: a systematic review (2006-2015). BMC Infectious Diseases. 2017; 17(1):612. https://doi.org/10.1186/s12879-017-2699-8 PMID: 28893198
14. Andrianakis I, Vernon IR, McCreesh N, McKinley TJ, Oakley JE, Nsubuga RN, et al. Bayesian history matching of complex infectious disease models using emulation: a tutorial and a case study on HIV in Uganda. PLoS Computational Biology. 2015; 11(1):e1003968. https://doi.org/10.1371/journal.pcbi.1003968 PMID: 25569850
15. Irvine MA, Hollingsworth TD. Kernel-density estimation and approximate Bayesian computation for flexible epidemiological model fitting in Python. Epidemics. 2018;.
16. Jandarov R, Haran M, Bjørnstad O, Grenfell B. Emulating a gravity model to infer the spatiotemporal dynamics of an infectious disease. Journal of the Royal Statistical Society: Series C (Applied Statistics). 2014; 63(3):423–444. https://doi.org/10.1111/rssc.12042
17. Bijak J, Hilton J, Silverman E, Cao VD. Reforging the wedding ring: Exploring a semi-artificial model of population for the United Kingdom with Gaussian process emulators. Demographic Research. 2013; 29:729. https://doi.org/10.4054/DemRes.2013.29.27
18. Farah M, Birrell P, Conti S, Angelis DD. Bayesian emulation and calibration of a dynamic epidemic model for A/H1N1 influenza. Journal of the American Statistical Association. 2014; 109(508):1398–1411. https://doi.org/10.1080/01621459.2014.934453

19. Cameron E, Battle KE, Bhatt S, Weiss DJ, Bisanzio D, Mappin B, et al. Defining the relationship between infection prevalence and clinical incidence of Plasmodium falciparum malaria. Nature Communications. 2015; 6:8170. https://doi.org/10.1038/ncomms9170 PMID: 26348689

20. Christopher MB. Mixture Models and Expectation-Maximization. In: Pattern recognition and machine learning. Springer-Verlag New York; 2016.

21. Andrianakis I, McCreesh N, Vernon I, McKinley T, Oakley J, Nsubuga R, et al. Efficient history matching of a high dimensional individual-based HIV transmission model. SIAM/ASA Journal on Uncertainty Quantification. 2017; 5(1):694–719. https://doi.org/10.1137/16M1093008

22. Cybenko G. Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals and Systems. 1989; 2(4):303–314. https://doi.org/10.1007/BF02551274

23. Ruder S. An overview of gradient descent optimization algorithms. arXiv preprint arXiv:160904747. 2016;.

24. Stephens MA. EDF statistics for goodness of fit and some comparisons. Journal of the American Statistical Association. 1974; 69(347):730–737. https://doi.org/10.1080/01621459.1974.10480196

25. Gillespie DT. Exact stochastic simulation of coupled chemical reactions. The Journal of Physical Chemistry. 1977; 81(25):2340–2361. https://doi.org/10.1021/j100540a008

26. Gillespie DT. Approximate accelerated stochastic simulation of chemically reacting systems. The Journal of Chemical Physics. 2001; 115(4):1716–1733. https://doi.org/10.1063/1.1378322

27. Heesterbeek H, Anderson RM, Andreasen V, Bansal S, De Angelis D, Dye C, et al. Modeling infectious disease dynamics in the complex landscape of global health. Science. 2015; 347(6227):aaa4339. https://doi.org/10.1126/science.aaa4339 PMID: 25766240

28. Bishop C, Bishop CM, et al. Neural networks for pattern recognition. Oxford University Press; 1995.

29. Kennedy MC, O'Hagan A. Predicting the output from a complex computer code when fast approximations are available. Biometrika. 2000; 87(1):1–13. https://doi.org/10.1093/biomet/87.1.1

30. Andrianakis I, Vernon I, McCreesh N, McKinley T, Oakley J, Nsubuga R, et al. History matching of a complex epidemiological model of human immunodeficiency virus transmission by using variance emulation. Journal of the Royal Statistical Society: Series C (Applied Statistics). 2017; 66(4):717–740. https://doi.org/10.1111/rssc.12198

31. Bengio Y, Simard P, Frasconi P. Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks. 1994; 5(2):157–166. https://doi.org/10.1109/72.279181 PMID: 18267787

32. Sutskever I, Martens J, Dahl G, Hinton G. On the importance of initialization and momentum in deep learning. In: International Conference on Machine Learning; 2013. p. 1139–1147.

33. Osio IG, Amon CH. An engineering design methodology with multistage Bayesian surrogates and optimal sampling. Research in Engineering Design. 1996; 8(4):189–206. https://doi.org/10.1007/BF01597226

34. MacKay DJ. Information-based objective functions for active data selection. Neural Computation. 1992; 4(4):590–604. https://doi.org/10.1162/neco.1992.4.4.590

35. Ball F, Barbour A. Poisson approximation for some epidemic models. Journal of Applied Probability. 1990; 27(3):479–490. https://doi.org/10.1017/S0021900200039048

36. LeCun Y, Bengio Y, Hinton G. Deep learning. Nature. 2015; 521(7553):436. https://doi.org/10.1038/nature14539 PMID: 26017442

37. Chollet F. keras; 2015. https://github.com/fchollet/keras.

38. Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems; 2015. Available from: https://www.tensorflow.org/.