Multitask deep learning for the emulation and calibration of an agent-based malaria transmission model

Agastya Mondal [1,*], Rushil Anirudh[2], Prashanth Selvaraj [3]

1 Divisions of Epidemiology and Biostatistics, School of Public Health, University of California, Berkeley, CA, USA
2 Amazon, Palo Alto, CA, USA
3 Institute for Disease Modeling, Gates Foundation, Seattle, WA, USA

* agastya_mondal@berkeley.edu

Abstract

Agent-based models of malaria transmission are useful tools for understanding disease dynamics and planning interventions, but they can be computationally intensive to calibrate. We present a multitask deep learning approach for emulating and calibrating a complex agent-based model of malaria transmission. Our neural network emulator was trained on a large suite of simulations from the EMOD malaria model, an agent-based model of malaria transmission dynamics, capturing relationships between immunological parameters and epidemiological outcomes such as age-stratified incidence and prevalence across eight sub-Saharan African study sites. We then use the trained emulator in conjunction with parameter estimation techniques to calibrate the underlying model to reference data. Taken together, this analysis shows the potential of machine learning-guided emulator design for complex scientific processes and their comparison to field data.

Author summary

Mathematical models can help understand and design interventions for infectious diseases such as malaria, but they can be complex and time-consuming to work with. In this study, we developed a machine learning approach to create a fast and accurate emulator for a detailed malaria transmission model. Our emulator can quickly predict various disease outcomes based on different immune system parameters. We trained it using a large set of simulations created using the EMOD malaria model and tested it against real-world data from multiple African countries. The emulator not only matches the original model's predictions closely, but can also be used to efficiently find the best parameter values to match field observations. We hope that this work can serve as a demonstration of machine learning tools to link complex models of disease transmission to field data.

Introduction

Despite progress in recent decades, the burden of malaria remains unacceptably high, especially in sub-Saharan Africa [1]. Due to numerous factors including seasonal heterogeneity, complex immune landscapes, and the spatial distribution of vectors and interventions, designing effective malaria control strategies is difficult. Mathematical

models of disease transmission can aid in the design of these programs by parameterizing the scientific processes that underlie transmission in a given setting. For novel interventions such as genetic vector control tools and updated vaccines [2, 3], for which field data do not yet exist, mathematical models can bridge the gap between expected outcomes and intervention parameters. Epidemiological models have been widely used for policy planning, prediction, and etiology [4–8]. These models generally provide a mechanistic framework by which to understand the progression of a disease from biological to sociodemographic facets.

For vector-borne diseases such as malaria, the progression of disease represents a complex relationship between vector, host, and environment. Although simple models of malaria transmission have been used for more than a century [9], in recent years more complex models have been developed. These models explicitly parameterize the development of the pathogen in the vector and can take into account heterogeneity in transmission such as age structure, immunity, species, and spatial landscape. While these models can accurately describe the processes involved in disease progression, they tend to require large amounts of computational resources to estimate parameters for a wide range of outcomes and settings. From a biological perspective, the reciprocal transmission of the malaria pathogen between vectors and humans is a detailed process involving vector feeding behavior, the development of infectious gametocytes in human liver and blood stages, and various forms of anti-parasite immunity. Modeling malaria transmission in large populations additionally requires attention to environmental conditions (as rainfall is a major ecological driver of larval carrying capacity), vector control interventions, mobility, and vector species. Clinical manifestations of malaria add another layer of complexity, whereby humans can be infected clinically, asymptomatically, or subpatently. Thus, models of malaria transmission tend to be more complex than those of other disease systems, such as respiratory viruses. Simple models of malaria transmission include numerous simplifying assumptions. For example, the Ross-Macdonald model [9] (perhaps the simplest malaria transmission model), represents humans and vectors as homogeneous, randomly-mixing populations, and considers only susceptible and infectious states. More complex models such as EMOD [10, 11] and OpenMalaria [12] explicitly describe the emergence of infection in stochastic, individual-based, formulations. Evaluating complex models for large sets of scenarios therefore requires sufficient computational resources.

Additionally, calibrating (i.e., aligning model parameters) to external field data can be challenging, as exploring multidimensional parameter spaces can quickly become intractable. These techniques therefore demand large amounts of computational resources. For complex systems such as mechanistic disease transmission models, parameter estimation techniques such as Bayesian Markov chain Monte Carlo, sequential Monte Carlo, and approximate Bayesian computation have been employed to calibrate mechanistic models [13–16]. Resource-constrained methods such as Bayesian likelihood calculations [17] and nearest neighbors [18] allow for parameter estimation with respect to complex models, but can be limited by sampling and defining explicit likelihood structures.

Recent literature has proposed the development of machine learning (ML) model "surrogates" or "emulators" to complex disease transmission models [18–20]. These models allow for the computationally efficient evaluation and prediction of disease dynamics as compared to the original model, and generally learn patterns from a large set of realizations from the underlying model. While there are many techniques to develop an emulator model [21–23], we focus here on the application of deep learning as an emulator for a complex model of malaria transmission. Deep neural networks (DNNs) represent a class of models that enable the estimation of an arbitrary function

between inputs and outputs. Unlike other emulators, DNNs make very few assumptions about the functional relationship between the input and output space, and are flexible and extensible to a large variety of use cases. Additionally, once trained, evaluation of a DNN is orders of magnitude faster than a realization of the underlying mechanistic model. Finally, DNNs were chosen for this analysis because, unlike other methods, DNNs are known to model stochastic processes that are important to consider in disease transmission, as infection events can be governed by an underlying probability distribution. Incorporating stochastic realizations into the training of the DNN emulator allows for the inclusion of stochastic uncertainty, as the emulator captures both the mean behavior and the spread across runs.

Mathematically, we are interested in training a deep neural network to infer a function f mapping a vector of parameters $\mathbf{x}$ to a set of prevalence and incidence outputs y_i, for $i \in [1, n]$ outputs, where the inputs and outputs correspond to realizations of the original mechanistic model:

$$f : \mathbf{x} \mapsto \{y_1, ..., y_n\}$$

Each input parameter set will map to multiple prevalence and incidence outputs simultaneously, leveraging methods from multitask learning. In this framework, one neural network predicts multiple outcomes, allowing the network to learn shared patterns in the simulation data. The need to train several emulators separately is additionally eliminated [24].

With respect to calibration, the trained emulator can be used to rapidly search parameter space to identify input immune parameters that result in prevalence and incidence outputs that align with reference data, as the evaluation of trained DNNs orders of magnitude faster than the underlying mechanistic model. Techniques from numerical optimization such as gradient descent allow for more flexibility in exploring parameter space and can identify out-of-distribution calibration fits. We couple these parameter estimation techniques with the trained emulator to identify parameter sets whose outputs most closely align with reference data. Finally, as we are interested in the universality of these biologically relevant model parameters across study sites, supplementary analyses were conducted to test the emulator-calibrated parameter sets against outputs from study sites not seen during training. Thus, in this work, we show that i) deep learning techniques can act as emulators to complex epidemiological models, ii) these deep learning emulators can be used to calibrate the underlying mechanistic model to field data, and iii) model parameters identified by the emulator can capture immune dynamics in study sites not previously seen during training, allowing for inference on new sites without retraining the emulator.

Methods

Malaria transmission model and parameter space

Simulations were conducted using the EMOD framework [10, 11]. EMOD is an agent-based, stochastic mechanistic model of malaria transmission that incorporates within-host parasite and immune dynamics alongside vector life cycle dynamics and human demographics. Here, we are interested in calibrating parameters associated with adaptive immunity that is stimulated by blood-stage malaria infection within the human host. Our ultimate goal in this study is to identify a universal set of immune factors whose simulated outputs closely align with collected parasite prevalence and clinical incidence data across eight sub-Saharan African study sites. As such, in order to build a suite of simulations on which the DNN was trained, we generated 2000 Latin hypercube-sampled parameter sets per site, with 10 stochastic repetitions per set,

resulting in a total of 160,000 (8 sites x 2000 parameter sets x 10 stochastic repetitions) simulations. Descriptions of the immune parameters and their biologically relevant ranges are given in Table 1. Following the methodology in Selvaraj *et al.* [25], each study site has its own set of entomological innoculation rates (EIRs), antimalarial interventions, and case management parameters, which remain unchanged across simulations for a given site. As outputs, we tracked four types of data corresponding to available reference data: annual clinical incidence by age group, annual *P. falciparum* prevalence by age group, asexual parasite density by age group by month, and gametocyte density by age group by month. Each site's reference data corresponds to a unique set of age and density bins, and the output for each simulation for a specific site is binned in alignment with the reference data.

Parameter	**Description**	**Range**
Antigen switch rate	The antigenic switching rate per infected red blood cell per asexual cycle.	$[10^{-10}, 10^{-9}]$
Falciparum MSP variants	The number of distinct merozoite surface protein variants for *P. falciparum* malaria in the overall parasite population in the simulation, not necessarily in an individual.	$[5, 50]$
Falciparum nonspecific types	The number of distinct non-specific types of *P. falciparum* malaria.	$[5, 100]$
Falciparum PfEMP1 variants	The number of distinct *Plasmodium falciparum* erythrocyte membrane protein 1 (PfEMP1) variants for *P. falciparum* malaria in the overall parasite population in the simulation.	$[600, 1700]$
Max individual infections	The limit on the number of infections that an individual can have simultaneously.	$[3, 10]$
MSP merozoite kill fraction	The fraction of merozoites inhibited from invading new erythrocytes when MSP1-specific antibody level is 1.	$[0.20, 0.80]$
Nonspecific antigenicity factor	The nonspecific antigenicity factor that adjusts antibody iRBC kill rate to account for iRBCs caused by antibody responses to antigenically weak surface proteins.	$[0.1, 0.9]$

Table 1. Overview of immune parameters used to generate Latin hypercube samples with which to simulate.

Data sources

In order to validate the ability of the model emulator to produce epidemiologically relevant outputs, we calibrated the underlying model against field data from eight sub-Saharan African sites, using the trained emulator. The Garki project [26] was a multi-year study conducted in Nigeria in the 1970s with the goal of understanding the feasibility of malaria elimination in this setting, aided by the collection of parasitology data to supplement epidemiological and entomological measurements. Additional field

data from Senegal, Tanzania, and Burkina Faso were used to evaluate the emulator [27–29]. Each reference site recorded data in a structure unique to the site's resources. The Garki sites (Matsari, Rafin Marke, and Sugungum) recorded month and age-binned asexual parasite density, the Senegal (Ndiop and Dielmo) sites recorded age-binned annual clinical malaria incidence, the Burkina Faso sites (Dapelogo and Laye) recorded month and age-binned asexual parasite and gametocyte density, and the Tanazania site (Namawala) recorded age-binned malaria prevalence. With respect to data collection, parasite density data from the Garki and Burkina Faso sites were collected via microscopy and real-time quantitative nucleic acid sequence-based amplification (QT-NASBA), respectively [26]. Giemsa stains were used to ascertain malaria prevalence in the Tanzania site [27], and active surveillance was used to ascertain clinical malaria incidence in the Senegal sites [28]. Our simulations however record all of these outcomes for each site-specific simulation. In a simulation, we are able to export fine-grained data for a range of outcomes and aggregate them to match the study site reference data. Then, despite a small reference dataset, we can train our emulator on multiple outcomes such that it can infer patterns between the input parameters and each outcome. In the calibration step, when comparing the emulator's outputs to the reference data for a site, only the relevant outputs are selected and compared. Table 2 shows an overview of the associated data for each study site.

Site name	**Country**	**Data type**
Matsari	Nigeria	Asexual parasite density by age by month
Rafin Marke	Nigeria	Asexual parasite density by age by month
Sugungum	Nigeria	Asexual parasite density by age by month
Ndiop	Senegal	Clinical incidence by age
Dielmo	Senegal	Clinical incidence by age
Dapelogo	Burkina Faso	Asexual parasite, gametocyte density by age by month
Laye	Burkina Faso	Asexual parasite, gametocyte density by age by month
Namawala	Tanzania	Malaria prevalence by age

Table 2. Overview of study sites and reference data to which the model emulator is calibrated.

Deep learning architecture

Once the simulation suite had been run, a DNN was trained on top of simulation outputs. As stated, our goal is to learn a function f that maps the input space $\mathbf{x}$ (the immune parameters and the study site) to the incidence and prevalence outputs y for each of the n outcomes:

$$f : \mathbf{x} \mapsto \{y_1, ..., y_n\}$$

We model f as a feed-forward, multitask neural network with six fully-connected shared, hidden layers and five fully-connected task-specific layers. Rectified linear units (ReLU) were used to introduce nonlinearity into the network, and dropout and batch normalization were used to prevent overfitting. Input features were standardized using min-max scaling and outputs were normalized between 0-1 via a sigmoid transformation. The neural network emulator was trained by inputting ten stochastic repetitions of EMOD model outputs for each parameter set, essentially capturing the mean response across simulation runs. While we did not model simulation uncertainty explicitly, neural network architectures such as Bayesian deep convolutional networks [18, 30, 31], allow for

the quantification of uncertainty. Explicitly modeling uncertainty may be beneficial in other scenarios, such as sensitivity analyses and the development of model-informed target product profiles [2, 32].

Model training and hyperparameter tuning

Training a neural network involves iteratively optimizing the model's "weights" (i.e., the strengths of connection between different layers in the neural network) in order to minimize the discrepancy between model output and training data. Then, the model is evaluated on a previously-unseen data subset to evaluate its performance on new values. We define the *loss function* as the numerical discrepancy between the neural network's output and the training data. While there are many loss functions for different modeling goals, a widely-used loss function is the ℓ_2 loss, corresponding to the mean squared error between model outputs $f(\mathbf{x})$ and training data $y(\mathbf{x})$ for a given set of input parameters $\mathbf{x}$ and N data samples:

$$\mathcal{L}(y(\mathbf{x}), f(\mathbf{x})) = \frac{1}{N}\sum_{i=0}^{N}(f(\mathbf{x})_i - y(\mathbf{x})_i)^2$$

In a multitask learning framework, we further segment this loss function per task, such that the overall goal of training is to minimize the joint loss across n tasks and weight λ_i for task i:

$$\mathcal{L}(y(\mathbf{x}), f(\mathbf{x})) = \sum_{i \in n} \lambda_i \mathcal{L}_i(y_i(\mathbf{x}), f_i(\mathbf{x}))$$

Because our emulator is relatively simple, we employ uniform loss weighting (i.e., $\lambda_i = 1 \forall i$), but for more complex networks, the relative importance of each task's loss becomes important [33]. Many methods have been proposed to address balancing the losses between tasks to ensure the model jointly learns each output type. We use the Adam stochastic gradient descent algorithm [34] to optimize the emulator's weights and minimize the multitask loss function.

A final component of model specification involves the selection of hyperparameters. In contrast to model weights, which are learned during the optimization process, hyperparameters are fixed attributes of the model that are selected prior to training. These include parameters associated with network architecture (number of hidden layers, number of neurons per layer), training (learning rate, batch size), and regularization (dropout probability, weight decay). In order to select hyperparameters for our model, we randomly sample 100 hyperparameter sets and use the ASHA [35] algorithm to efficiently select the sets that minimize the joint loss on a hold-out data subset after training. Once the neural network is trained, we are able to evaluate its ability to calibrate the underlying model to reference data.

Calibration workflow

To evaluate the emulator's performance in identifying the optimal input parameter set for the underlying mechanistic model, outputs from the trained emulator were compared to the reference data to determine the input parameter set whose associated emulated outputs most closely aligned with the reference data. Drawing from techniques in numerical optimization, stochastic gradient descent (SGD) was chosen as the primary parameter estimation method. In this framework, parameter sets were chosen by optimizing against a loss function comparing emulated outputs to the reference data. Iterative updates were made to the input parameters to minimize the loss function until

a stopping criterion was achieved. Fig 1 shows a schematic workflow of the emulation and parameter estimation process.

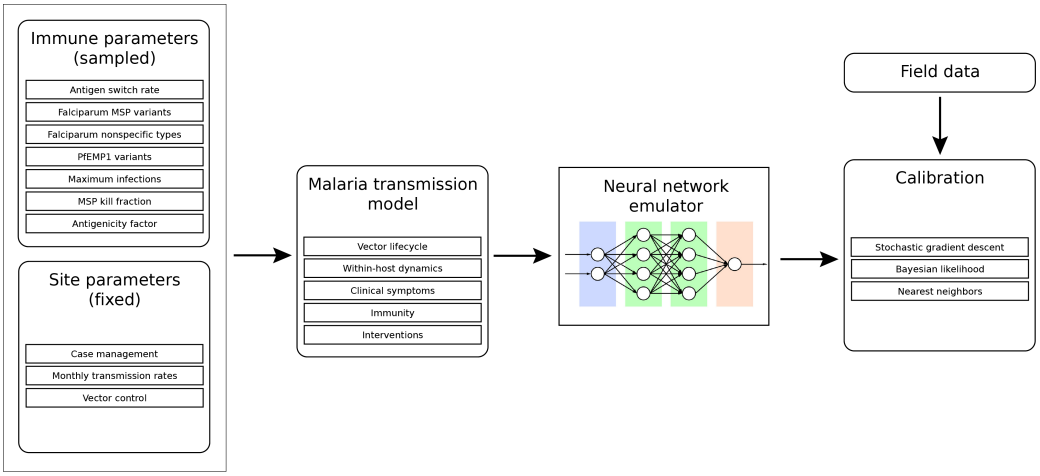


Fig 1. Emulation and calibration workflow. First, immune parameters are sampled, and site-specific parameters are specified. Then, the malaria transmission model (EMOD) is run for all sampled parameter sets. A neural network emulator is trained on the simulation data. Finally, calibration proceeds by comparing emulated output to field data.

Stochastic gradient descent

SGD allowed the emulator to explore regions of parameter space efficiently and outside the bounds on which the emulator was trained. At a high level, the algorithm proceeds as follows: 1) first, we randomly sampled 1000 initial parameter sets, 2) for each parameter set, we evaluated a loss function comparing the emulated outcomes for the parameter set against the reference, 3) we updated the parameter sets based on gradient descent (i.e., in the direction of the gradient of the loss function with respect to the parameters), 4) we iteratively continued steps 1-3 until a convergence criterion was met. Additionally, parameter values during optimization were constrained to their biologically plausible values (i.e., integer parameters are rounded, and all parameters are constrained to be positive). Finally, we selected the optimized parameter set which led to the smallest loss value. The key component of gradient descent is its iterative updating. For a parameter set $\mathbf{x}$ at iteration t, its update rule is given by:

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \gamma \nabla \mathcal{L}(\mathbf{x}_t)$$

for a learning rate γ and loss function $\mathcal{L}$. This allowed us to explore regions of parameter space that the emulator was not explicitly trained on. As before, we used the joint ℓ_2 loss function:

$$\sum_{s \in \mathcal{S}} MSE_{\mathbf{x}}(s, f(\mathbf{x}), d_s)),$$

where

$$MSE_{\mathbf{x}}(s, f(\mathbf{x}), d_s)) = \frac{1}{N_s} \sum_{i=0}^{N} (f(\mathbf{x})_i - d_{s,i})^2$$

for sites $s \in \mathcal{S}$, emulated outcomes $f(\mathbf{x})$, and reference data for site d_s. We used the Adam [34] variant of the stochastic gradient descent algorithm to efficiently search parameter space. We additionally tested different learning rates to ensure the optimization did not terminate in local extrema.

New site inference

In addition to evaluating the emulator's ability to capture epidemiological dynamics across study sites, we are interested in assessing i) the relationship between site-specific data and emulator performance, and ii) how applicable chosen parameter sets are to new study sites. Assessing emulator performance in the context of new field data will be important in understanding whether an emulator can capture patterns in simulation data without the need to retrain the entire system. In order to evaluate the inference ability of the emulator to unseen sites, we used a leave-one-out validation protocol, where one site is used as a held out test set. We additionally augmented the training data with site-specific epidemiological parameters such as monthly transmission rates and intervention coverages. Then, for each study site, we used the emulator not trained on that site to calibrate the remaining sites. Finally, we compared the simulation outputs corresponding to the calibrated parameter set on the site excluded during training. As models are reevaluated as new field data are obtained, this workflow will allow for an assessment of the emulator's ability to capture dynamics in new sites, given sufficient site-specific data are included in the emulator's training. These results are shown in the S2 Appendix.

Sensitivity analysis

An area of concern for the developers of emulator models will be the size of the training dataset. As running complex epidemiological simulations can take long periods of time and demand large amounts of computational resources, it will be important to understand the relationship between the size of the underlying simulation suite and the performance of the emulator and calibration. To test this, we ran the emulation and calibration workflow for various subsets of the full training data and assessed the performance for each subset. Fifty replicates were run to understand the variance in performance. Our goal was to learn the smallest data subset before the emulator performance degraded. For this initial exploration, we purposefully ran a very large simulation suite, but we hope that this sensitivity analysis can shed light onto the size of simulation sets required to minimize computational resources required for future analyses. We additionally analyzed the time complexity of using an emulator compared to running the underlying mechanistic model.

Results

Emulator performance

First, we show the ability of the emulator to capture simulated model dynamics. After hyperparameter tuning, the final model parameters (shown in Table 3) were used to construct the neural network. Similarly, Table 4 shows the final task-specific ℓ_2 losses on a holdout, previously-unseen data subset.

Stochastic gradient descent

Here, we show the calibrated parameter set for the stochastic gradient descent (SGD) method. This calibration methodology allows us to explore parameter sets on which the emulator was not explicitly trained. We used the SGD method to find the parameter set leading to the lowest discrepancy (loss), comparing the emulated outputs to the reference data. Then, we ran the original simulation model with the selected parameters to see the relationship between the selected parameter set, the underlying simulation, the emulated outcomes, and the reference data. Fig 2 shows the calibrated simulation,

Parameter	Description	Value
n_1	Number of neurons in first shared layer	512
n_2	Number of neurons in second shared layer	128
n_3	Number of neurons in third shared layer	4
n_4	Number of neurons in fourth shared layer	64
n_5	Number of neurons in fifth shared layer	16
n_6	Number of neurons in sixth shared layer	32
n_7	Number of neurons in first task-specific layer	16
n_8	Number of neurons in second task-specific layer	2
n_9	Number of neurons in third task-specific layer	8
n_{10}	Number of neurons in fourth task-specific layer	256
n_{11}	Number of neurons in fifth task-specific layer	8
γ	Learning rate	$1e-4$
b	Batch size	32
p	Dropout probability	0.06
λ	Weight decay	$1.1e-5$

Table 3. Overview of final neural network hyperparameters.

Task	Test set loss
Annual clinical incidence by age bin	0.007
Annual *P. falciparum* prevalence by age bin	0.006
Gametocytemia by age bin by month	0.006
Parasitemia by age bin by month (Matsari, Rafin Marke, Sugungum)	0.006
Parasitemia by age bin by month (Dapelogo, Laye)	0.008

Table 4. Overview of task-specific losses (MSE) of trained emulator.

emulator, and reference data for the Dielmo, Ndiop, and Namawala study sites, corresponding to the annual incidence and prevalence outcomes, respectively. Fig 3 shows the calibrated simulation, emulator, and reference data for the Dapelogo and Laye study sites, corresponding to density and age-binned, monthly parasitemia and gametocytemia outcomes. Fig 4 shows the calibrated simulation, emulator, and reference data for the Matsari, Rafin Marke, and Sugungum study sites, corresponding to density and age-binned, monthly parasitemia outcomes. Table 5 shows the calibrated parameter set that leads to the best-fit outcomes.

Parameter	Value
Antigen switch rate	$2.04e-10$
Falciparum MSP variants	11
Falciparum nonspecific types	59
Falciparum PfEMP1 variants	1494
Max individual infections	8
MSP merozoite kill fraction	0.59
Nonspecific antigenicity factor	0.44

Table 5. Best fit immune parameters from SGD calibration.

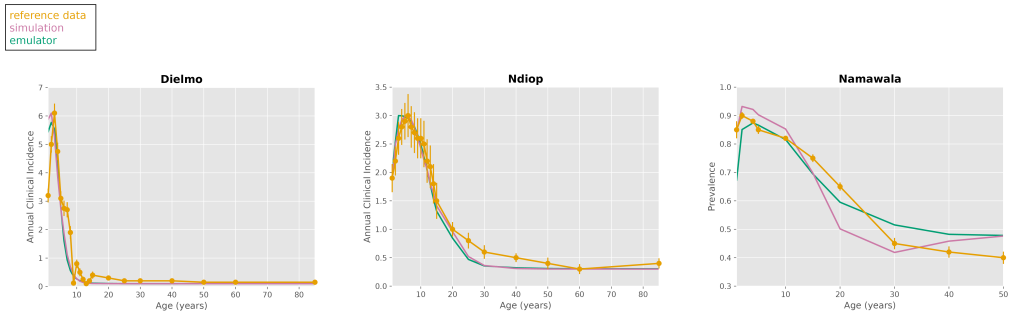


Fig 2. SGD calibration plots for Dielmo, Ndiop, and Namawala sites. These plots show the simulation, emulator, and reference data for the study sites corresponding to the annual clinical incidence and malaria prevalence outcomes, calibrated via the stochastic gradient descent method. The emulator was able to identify parameter sets whose outputs closely align with reference data for the sites reporting incidence data (Dielmo and Ndiop).

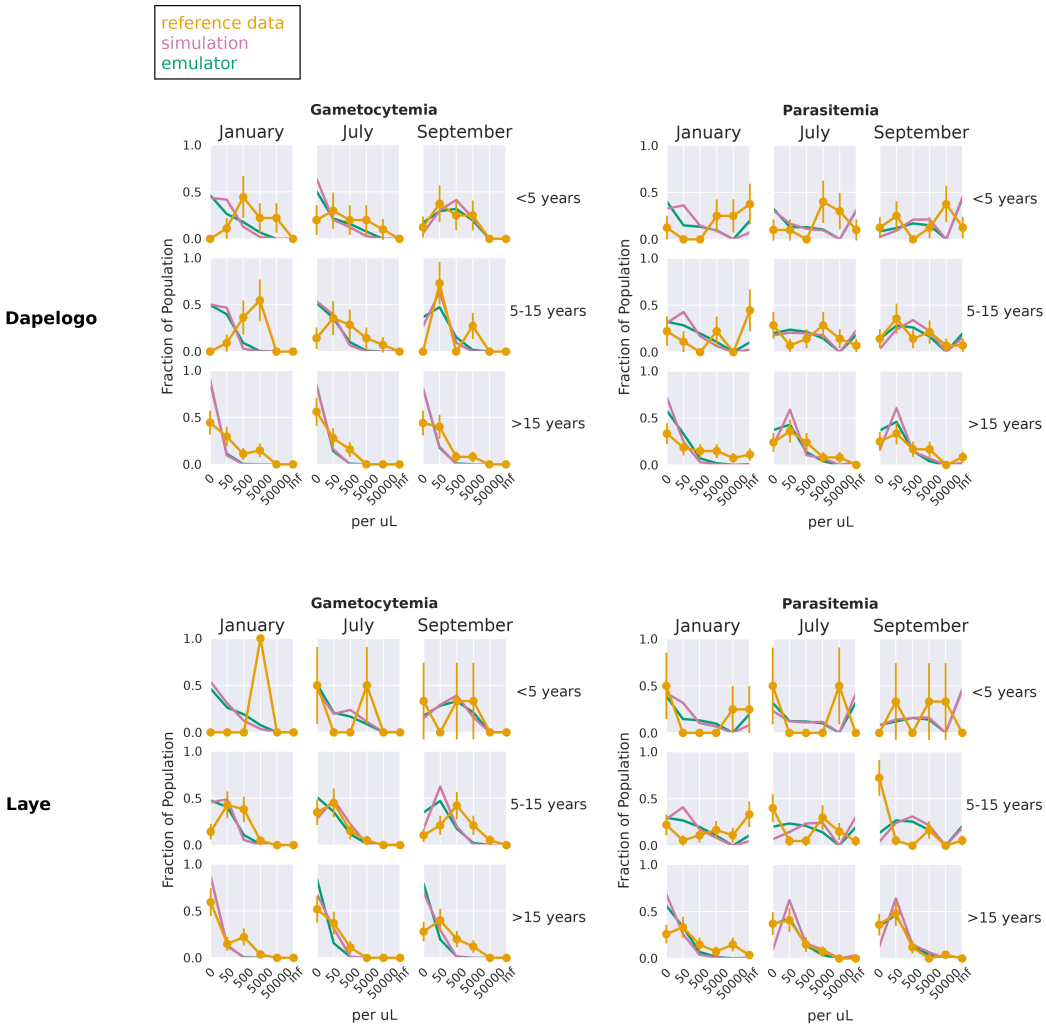


Fig 3. SGD calibration plots for Dapelogo and Laye sites. These plots show the simulation, emulator, and reference data for the study sites corresponding to the parasitemia and gametocytemia outcomes, calibrated via the stochastic gradient descent method.

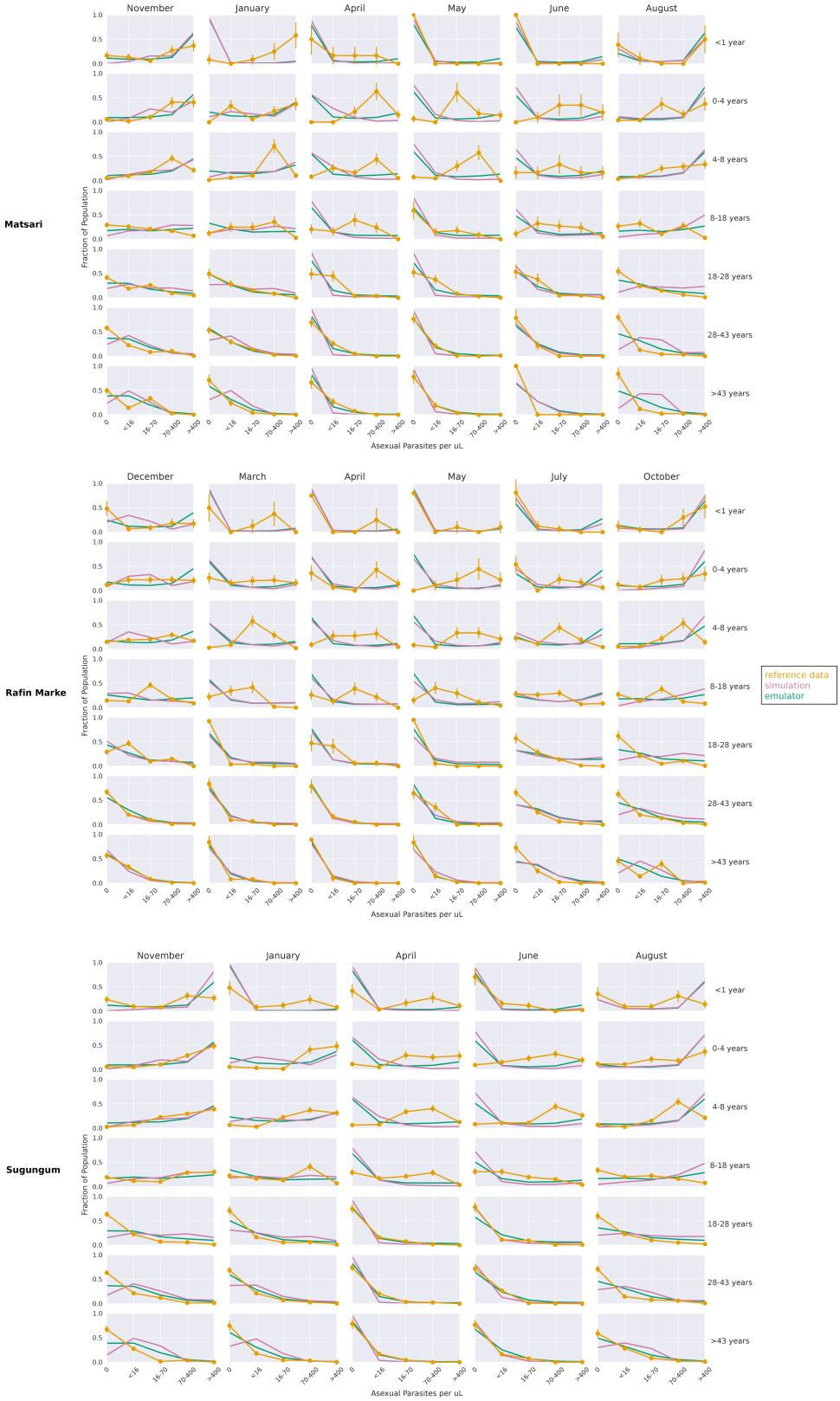


Fig 4. SGD calibration plots for Matsari, Rafin Marke and Sugungum sites. These plots show the simulation, emulator, and reference data for the study sites corresponding to the parasitemia outcomes, calibrated via the stochastic gradient descent method.

Sensitivity analysis

Fig 5 shows the relationship between the size of the dataset on which the emulator was trained versus its ability to predict outcomes in a previously-unseen simulation data subset. Training of ML models is generally the most computationally expensive step in the inference pipeline, so there is a tradeoff between training models on more data and resource usage. For simpler models, such as the feed-forward network used to emulate EMOD, we see that the emulator can learn patterns in the simulation data even when trained on smaller data subsets; these results suggest that 40-50% of the original dataset size is sufficient to maintain similar levels of performance to the full dataset. However, for more complex models such as vision or large language models [36, 37], the size of the training data is more relevant. Additionally, we see that providing the model with too much training data leads to an increase in the variance of loss for the unseen data subset, which is likely due to the smaller test set or overfitting.

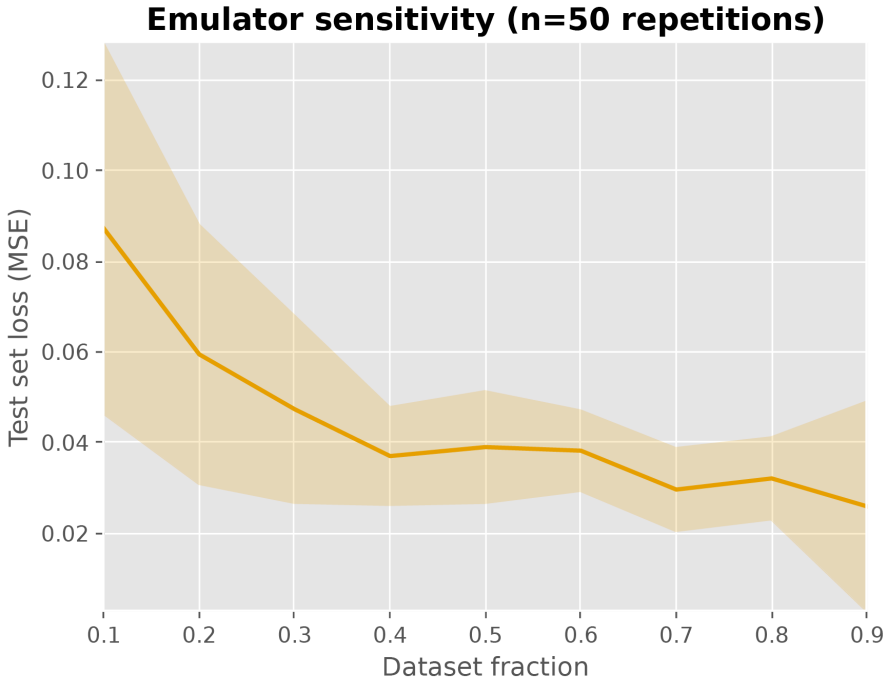


Fig 5. Sensitivity analysis for emulator training dataset size. 50 replicates were run to evaluate the relationship between the training data size and the emulator performance on a hold-out test set.

Time complexity analysis

A primary motivation of employing emulator models is their relative computational efficiency [38]. Once a neural network is trained, inference (i.e., obtaining outputs from a set of inputs) is generally orders of magnitude faster than running the underlying simulation. Training and selecting hyperparameters of DNNs generally are the rate-limiting steps in an ML pipeline; in our case, a full hyperparameter search took roughly eight hours on a single graphics processing unit (GPU). However, advances in data parallelization [39] can significantly reduce training times for DNNs and other network architectures. To demonstrate the inference speed of the emulator as compared to the underlying simulations, we show the distribution of runtimes of the same 2000

parameter sets, segmented by study site in Fig 6. We see that the trained emulator can, for the same parameter sets, predict outcomes several orders of magnitude faster than the runtimes of the underlying simulations. Additionally, simulation runtimes show more variance within and across study site, as they are highly sensitive to input parameters. For EMOD, parameters such as the maximum number of infections and parasite variants can significantly expand the runtime of a simulation, as more infection scenarios need to be considered explicitly. A trained emulator conversely shows little variance and can efficiently handle the same parameter sets.

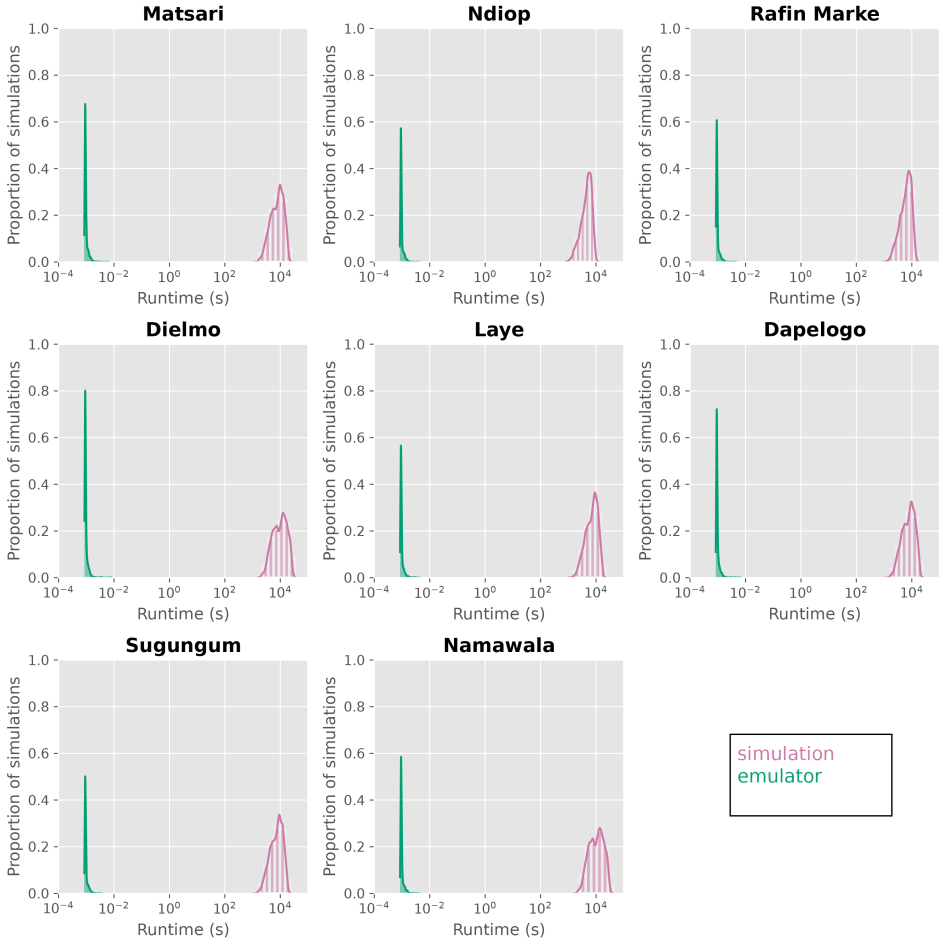


Fig 6. Comparison of runtimes between emulated and simulated outcomes, segmented by site. Trained emulators can predict outcomes several orders of magnitude faster than the runtimes of the underlying simulation.

Discussion and future directions

Here, we have described a workflow to 1) emulate a complex malaria transmission model using deep neural networks, and 2) calibrate the underlying mechanistic model to reference data using the emulator. Supplementary work showed the ability of DNN emulators to infer outputs to study sites not seen during training. Through careful sampling of the underlying parameter space, calibration methodology, and neural network architecture, we have shown that machine learning alternatives to traditional model calibration techniques (e.g., Markov chain Monte Carlo, particle filters) can be used to rapidly search parameter space and align mechanistic models to field data. There are a couple of issues that may make calibration more challenging when using an emulator for calibration. Specifically, an emulator may smooth over some detailed model dynamics, leading to potential biases or reduced accuracy in parameter estimates, particularly for parameters with subtle or complex effects. Additionally, the emulator fitting itself introduces another layer of uncertainty, which could propagate into the calibration process. However, despite these challenges, emulator-based calibration remains highly beneficial and robust. Emulators significantly reduce computational time, allowing exploration of a broader parameter space and enabling more extensive uncertainty quantification than would be possible with direct simulations alone. Furthermore, with careful validation - such as we conducted in this work - the emulator closely approximates the full model, preserving essential dynamics while enabling efficient calibration.

In terms of model architecture, we employed a multitask, feed-forward neural network mapping the input parameters (in this case, parameters associated with immunological dynamics of malaria transmission) to age and density-binned epidemiological outcomes. While more complex architectures exist (e.g., recurrent or transformer-based models) for time series inference, these models require more resources to train properly. In designing the emulator, we opted to employ a multitask strategy (i.e., one emulator jointly predicts all outcomes) for two primary reasons. First, this paradigm allows the emulator to learn shared patterns across the input and output space, as each outcome is likely not biologically independent of the others. In addition, model complexity and computational resource requirements are lower in training a single, multitask emulator as compared to several individual emulators.

We identified two primary challenges in designing a multitask emulator that model developers should consider when applying this framework to their simulation data: task weighting and specification of the output data format. With respect to task weighting, we opted to weight each task equally, as to not introduce additional complexity into the training pipeline. Prior research has found that model performance can be highly sensitive to model weights [40], and the weights themselves can be treated as a tuneable parameter. Several strategies have been proposed to address task weighting to achieve optimal network performance [24, 40]. While we were able to achieve adequate performance with a uniform loss weighting scheme, the loss weights are nonetheless another set of hyperparameters that should be considered when designing an emulator training pipeline. Specifying the output data format posed another challenge: we deliberately formatted the emulator's output to match the dimensionality of available field data. As new data become available, however, a key concern is the usability of the emulator with data formats that differ from those on which the network was trained. Recent developments in transfer learning and model finetuning [41] provide a framework by which the underlying emulator can still be used to align to new data without the need to retrain the entire system, via updating the neural network's output layers. These techniques could be leveraged here to align the emulator with new field data without needing to rerun the training pipeline.

We have additionally shown that neural network emulator models can predict outcomes for previously-unseen sites, allowing for more flexibility as more field data are obtained. Though the emulator performance is slightly degraded due to less training data and extrapolation to new study sites, the fits suggest that the emulator can still narrow down parameter space to reasonable values, given that the emulator training is augmented with suitable site-specific data such as transmission rates and intervention coverages. In our analysis, the structure of the field data matches the simulation data on which the emulator was trained, but future directions could consider updating the final layers of the neural network to output new data structures, as is done in other domains [41, 42]. This would allow even more flexibility when calibrating new field sites.

A concern with the usability of an emulator-based calibration workflow regards its ability to infer outcomes in parameter spaces not seen during emulator training. While we have shown that SGD allows for reasonable inference on parameter sets not explicitly seen during training, this method generally stays within the bounds of the training set, essentially interpolating between sampled points. To evaluate generalizability even further, we conducted supplementary analyses (S2 Appendix) to test the emulator's ability to calibrate data from study sites not seen during training. In this framework, an emulator was trained on simulation data from all study sites, excluding one. Then all sites were calibrated using the emulator. Our preliminary results show that the emulator is still able to capture relationships across all study sites, but its performance is degraded compared to emulators trained with the full dataset. Future iterations could examine training on broader site characteristics (e.g., variable EIRs, intervention coverages, etc.) to improve generalizability. Additionally, the strength of DNN-based emulators lies in their ability to leverage hierarchical learning, capturing nonlinear parameter interactions through layers of progressively abstracted features. With respect to sampling the parameter space, Rather than explicitly sampling every possible region in the parameter space uniformly, DNNs identify and generalize patterns across parameter sets, allowing them to effectively interpolate and extrapolate from limited training data [43]. This capability enables DNNs to efficiently represent high-dimensional spaces without exhaustive sampling, making them particularly suitable for scaling up from a moderate number of parameters to a higher-dimensional range.

With respect to the emulator's performance and the size of the training data, our study found that smaller parameter sets can still capture the underlying simulation's dynamics, albeit with more variance. This can inform the design of emulators in the context of limited computational resources.

Interpretability of deep neural network models

A major challenge with the acceptance of DNN models is their interpretability [44]. That is, most traditional DNNs are trained by minimizing the loss between the training data and the predicted outputs; the mechanisms on *how* they learn these patterns can be hard to discern, especially as model complexity increases. For some applications, understanding the underlying mechanisms of a DNN may be less of a concern. But for epidemiological models, the mechanisms of disease transmission are important to understanding how interventions are expected to behave on outcomes of interest. Additionally, in the calibration phase, it is important to ensure that model inputs and outputs correspond to biologically plausible parameter values (for example, by constraining values to realistic ranges, even if other values improve numerical performance).

An active area of machine learning research is improving the interpretability of models. Methods include reinforcement learning with human feedback and mixture of experts, among others [45, 46]. In the disease modeling realm, trusting inferences from ML models will require an understanding of the mechanisms by which an inference is made. Fine-tuning may also be required to ensure the model outputs are aligned with biologically plausible values. In our work, we have shown that ML emulators can rapidly search parameter space in order to align model outputs with reference data. Calibration outputs should be assessed against domain knowledge to identify which parameter sets are the most feasible given transmission setting and model dynamics [18]. Recent methods have been introduced to even incorporate domain knowledge into the training of ML models [47].

Comparison of calibration techniques

Here we have shown coupling SGD with a trained emulator is able to produce reasonable calibration fits across sites and outputs, comparing emulated outputs to reference data. In supplementary analyses (S1 Appendix), we evaluated two additional parameter selection techniques, involving Bayesian likelihoods and a nearest neighbors approach. In choosing a calibration method, attention must be paid to the functional form (if any) imposed on the outputs and how the method traverses parameter space. In the case of Bayesian likelihood and nearest neighbors, parameter sets used to train the emulator were sampled, and their associated outputs were compared against the reference data. Bayesian inference additionally requires a functional form of the conjugate distribution, which we derived from prior work [25]. Both these methods are rapid to evaluate, as they require a simple likelihood or loss calculation in comparison to reference data. For this analysis, Bayesian likelihoods were calculated on a previously-sampled parameter set. It could also be coupled with SGD as a "loss function" to provide the flexibility of evaluating new parameter sets while imposing a functional form on the data. Stochastic gradient descent with an MSE loss function, conversely, does not specify any functional form of the outputs nor does it require a sampling of the parameter space beforehand. Thus, SGD and other numerical optimization methods may be more flexible and result in better calibration fits, particularly those that are out-of-distribution (i.e., outside the emulator's training manifold). Numerical optimization methods do however require additional attention to hyperparameters, as the choice of initial values, learning rates, and momentum can greatly impact final fits [48]. Nonetheless, these methods serve as powerful and flexible tools in emulator-based calibration problems. Additional methods not evaluated here, such as iterative adaptive sampling [49], have shown promise in multitask calibration of agent based models.

Discussion of emulation frameworks

Here, we provide a brief overview of two main surrogate modeling techniques employed in infectious disease analysis and compare them to the DNN-based approach utilized in this study. Gaussian processes (GPs) represent a stochastic process (generally indexed by time or space), whose finite subsets follow a multivariate normal distribution. They tend to have fewer hyperparameters compared to other methods (and thus are easier to train) and can be used to model a wide range of scientific processes. Additionally, calibration via GPs has been shown to be robust to convergence issues and GPs provide a natural quantification of model uncertainty [50, 51]. Neural network-based approaches, on the other hand, can be more flexible in that they do not impose a normality assumption on the training data, but do require (often orders of magnitude) more parameters and attention to hyperparameters during training. Recent surrogate modeling literature has introduced Bayesian neural networks as a method to

simultaneously model the solution space and uncertainty, similar to GPs [18, 30, 31]. We opted for a traditional DNN-based approach to demonstrate its flexibility to highly non-linear simulation data and rapid inference. GP-based emulation studies [50] have additionally employed adaptive sampling to further refine parameter space and quantify uncertainty. DNN emulators can integrate effectively with adaptive sampling frameworks, employing uncertainty quantification methods such as ensembles or Monte Carlo Dropout [52, 53]. Hence, the potential for adaptive sampling is not only maintained but potentially enhanced by a DNN approach, given the computational efficiency and scalability inherent to these methods. Recent research [54, 55] has demonstrated that deep neural networks (DNNs) combined with effective methods of uncertainty estimation such as the Δ-UQ approach or Direct Epistemic Uncertainty Prediction (DEUP) often yield better performance in adaptive sampling and Bayesian optimization tasks compared to traditional GPs, Bayesian neural networks, and deep ensembles. In future work, we aim to systematically benchmark our neural-network emulator against alternative surrogate models—carefully exploring and tuning each model's hyperparameter space to ensure a fair, rigorous comparison to the same reference dataset.

Limitations

This study has several limitations. Mechanistic models of malaria transmission must be sufficiently complex to capture the seasonal and heterogeneous (by age, species, and location) nature of the disease. Even with complex models, capturing age-based heterogeneity is difficult [1] due to the nature of immune dynamics in children (i.e., children often experience more severe cases of malaria after maternal immunity has waned, but exposure immunity has not yet developed). Our calibration methodology aimed to find one set of immune parameters that best fit the reference data across age groups, but we see that the fits for parasite density are markedly worse in juvenile age groups, particularly for the Garki sites. This finding suggests that one set of immune parameters may not be fully appropriate across the age spectrum and that juvenile dynamics should be calibrated separately. Additionally, the underlying mechanistic model may not be fully describing immune dynamics in these age strata. Both of these aspects limit the ability of the calibration workflow to extend to younger age groups. From an intervention planning perspective, these dynamics should be better understood, as interventions often target children to reduce the significant morbidity and mortality they face from malaria.

A central challenge concerns the principled development of DNNs and scientific process emulators. Many architectures, hyperparameters, and training data sampling schemes must be considered. While some tools exist to help design these models, many choices must be made beforehand to ensure the emulator is capturing the dynamics of the transmission model. For other use cases where the mechanisms of the DNN are important to understand, careful attention must be paid to the development of the emulator to ensure it accurately represents the underlying scientific process [18]. Additionally, the tradeoff between simulation dataset size and model accuracy must be considered in the context of limited computational resources. We deliberately sampled a large range of parameter space as a proof-of-concept, but future studies should explore more deeply the ability of the emulator to learn simulation dynamics with more limited training datasets. In generating the training dataset for the primary analysis, we assumed that site characteristics (e.g. monthly EIRs, intervention coverages, etc.) were static across simulations. While this assumption may not be valid under all circumstances, such as interventions that may influence EIR via decreased prevalence, field data measuring EIR can be challenging to collect and validate. Our supplementary

analyses S2 Appendix show that augmenting the input data with site-specific EIRs can allow for calibration to new sites. In this transfer learning paradigm, the emulator learns relationships between site-specific characteristics like EIR and simulation outcomes. Pre-trained emulators can then be used to infer outcomes on new sites, given appropriate EIR data are supplied, without the need to retrain the emulator. Our preliminary analyses show that this approach can allow for refinement of calibration parameter space for sites not seen during training and incorporation of site-specific metadata to improve parameter estimates.

A major computational tradeoff in designing an emulator-based calibration workflow regards the generation of a robust training dataset. Fig 6 shows that once an emulator is trained, inference can be conducted rapidly. However, the training process itself requires two computationally-heavy steps: i) generating a suite of simulations and, ii) identifying optimal emulator hyperparameters. Deep learning techniques, such as fine-tuning and transfer learning, substantially mitigate issues of long training times. Extensive literature demonstrates that fine-tuning pre-trained networks can significantly reduce computational demands and improve training efficiency without sacrificing inference quality [56, 57]. Moreover, our current implementation utilized a single GPU; employing more advanced GPUs or multiple GPUs in parallel could substantially decrease training duration and further enhance computational efficiency. One example is the article by Goyal *et al.* [58], where they showed how convolutional neural networks (CNNs), a natural extension of the MLP architecture we use in this work, can essentially train large volumes of image data in an hour. Our initial experiments already suggest promising outcomes with such approaches.

A final concern relates to the validation of the calibration workflow. Biological parameters associated with malaria transmission can be difficult to quantify. Drawing from previous literature [25, 60], conservative bounds were chosen to fully capture the parameter space associated with each biological parameter. As such, simulation times varied significantly, as wider parameter ranges led to a larger variation in simulation scenarios (Fig 6 shows the distribution of simulation runtimes). With refined estimates of these parameters, either from literature or biological experiments, tighter LHS ranges could be considered for future iterations, as both simulation and calibration efficiency would increase significantly. DNN emulators can offer a very favorable trade-off between computational efficiency and reliable predictions, thereby ensuring accurate and policy-relevant calibration outcomes. Moreover, transfer learning is a technique by which neural networks could outperform other competing surrogates, where one can potentially bootstrap multiple related datasets (e.g., simulations and field data) or even simulations of varying fidelities (i.e., data structures) to effectively predict on the dataset of interest. This would not only greatly reduce the amount of data required, but also provide a meaningful way to capture implicit characteristics in datasets into the calibration process. There are numerous successful examples in literature in diverse applications. In malaria for example, transfer learning has been used to train DNNs for imaging and data analysis [61, 62], which makes a case for DNN-based calibration taking advantage of transfer learning.

Supporting information

S1 Appendix. Nearest neighbors and likelihood calibrations for all sites. Calibration results for complete (all-site) dataset.

S2 Appendix. Site-excluded calibrations. Calibration results for inferring outcomes on previously-unseen sites in emulator training.

Funding statement

AM was supported by funds from the Gates Foundation (INV-017683). This work was conducted before RA joined Amazon.

Code availability statement

Model weights, training, and calibration code are available on GitHub.

References

1. Shi D, Wei L, Liang H, Yan D, Zhang J, Wang Z. Trends of the Global, Regional and National Incidence, Mortality, and Disability-Adjusted Life Years of Malaria, 1990–2019: An Analysis of the Global Burden of Disease Study 2019. RMHP. 2023;Volume 16:1187–1201. doi:10.2147/RMHP.S419616.
2. Mondal A, Vásquez VN, Marshall JM. Target Product Profiles for Mosquito Gene Drives: Incorporating Insights From Mathematical Models. Front Trop Dis. 2022;3:828876. doi:10.3389/fitd.2022.828876.
3. Burgert L, Reiker T, Golumbeanu M, Möhrle JJ, Penny MA. Model-informed target product profiles of long-acting-injectables for use as seasonal malaria prevention. PLOS Glob Public Health. 2022;2(3):e0000211. doi:10.1371/journal.pgph.0000211.
4. Mondal A, Sánchez C HM, Marshall JM. MGDrivE 3: A decoupled vector-human framework for epidemiological simulation of mosquito genetic control tools and their surveillance. PLoS Comput Biol. 2024;20(5):e1012133. doi:10.1371/journal.pcbi.1012133.
5. Griffin JT, Hollingsworth TD, Okell LC, Churcher TS, White M, Hinsley W, et al. Reducing Plasmodium falciparum Malaria Transmission in Africa: A Model-Based Evaluation of Intervention Strategies. PLoS Med. 2010;7(8):e1000324. doi:10.1371/journal.pmed.1000324.
6. Griffin JT, Bhatt S, Sinka ME, Gething PW, Lynch M, Patouillard E, et al. Potential for reduction of burden and local elimination of malaria by reducing Plasmodium falciparum malaria transmission: a mathematical modelling study. The Lancet Infectious Diseases. 2016;16(4):465–472. doi:10.1016/S1473-3099(15)00423-5.
7. Penny MA, Verity R, Bever CA, Sauboin C, Galactionova K, Flasche S, et al. Public health impact and cost-effectiveness of the RTS,S/AS01 malaria vaccine: a systematic comparison of predictions from four mathematical models. The Lancet. 2016;387(10016):367–375. doi:10.1016/S0140-6736(15)00725-4.
8. Eckhoff P. Mathematical Models of Within-Host and Transmission Dynamics to Determine Effects of Malaria Interventions in a Variety of Transmission Settings. The American Journal of Tropical Medicine and Hygiene. 2013;88(5):817–827. doi:10.4269/ajtmh.12-0007.
9. Smith DL, Battle KE, Hay SI, Barker CM, Scott TW, McKenzie FE. Ross, macdonald, and a theory for the dynamics and control of mosquito-transmitted pathogens. PLoS Pathog. 2012;8(4):e1002588. doi:10.1371/journal.ppat.1002588.

10. Eckhoff PA. A malaria transmission-directed model of mosquito life cycle and ecology. Malar J. 2011;10(1):303. doi:10.1186/1475-2875-10-303.

11. Eckhoff PA. Malaria parasite diversity and transmission intensity affect development of parasitological immunity in a mathematical model. Malar J. 2012;11(1):419. doi:10.1186/1475-2875-11-419.

12. Smith T, Maire N, Ross A, Penny M, Chitnis N, Schapira A, et al. Towards a comprehensive simulation model of malaria epidemiology and control. Parasitology. 2008;135(13):1507–1516. doi:10.1017/S0031182008000371.

13. Verity R, Okell LC, Dorigatti I, Winskill P, Whittaker C, Imai N, et al. Estimates of the severity of coronavirus disease 2019: a model-based analysis. The Lancet Infectious Diseases. 2020;20(6):669–677. doi:10.1016/S1473-3099(20)30243-7.

14. Li M, Dushoff J, Bolker BM. Fitting mechanistic epidemic models to data: A comparison of simple Markov chain Monte Carlo approaches. Stat Methods Med Res. 2018;27(7):1956–1967. doi:10.1177/0962280217747054.

15. Hogan AB, Doohan P, Wu SL, Mesa DO, Toor J, Watson OJ, et al. Estimating long-term vaccine effectiveness against SARS-CoV-2 variants: a model-based approach. Nat Commun. 2023;14(1):4325. doi:10.1038/s41467-023-39736-3.

16. Fornace KM, Topazian HM, Routledge I, Asyraf S, Jelip J, Lindblade KA, et al. No evidence of sustained nonzoonotic Plasmodium knowlesi transmission in Malaysia from modelling malaria case data. Nat Commun. 2023;14(1):2945. doi:10.1038/s41467-023-38476-8.

17. Tohme T, Vanslette K, Youcef-Toumi K. A generalized Bayesian approach to model calibration. Reliability Engineering & System Safety. 2020;204:107141. doi:10.1016/j.ress.2020.107141.

18. Thiagarajan JJ, Venkatesh B, Anirudh R, Bremer PT, Gaffney J, Anderson G, et al. Designing accurate emulators for scientific processes using calibration-driven deep models. Nat Commun. 2020;11(1):5622. doi:10.1038/s41467-020-19448-8.

19. Golumbeanu M, Yang GJ, Camponovo F, Stuckey EM, Hamon N, Mondy M, et al. Leveraging mathematical models of disease dynamics and machine learning to improve development of novel malaria interventions. Infect Dis Poverty. 2022;11(1):61. doi:10.1186/s40249-022-00981-1.

20. Gherman IM, Abdallah ZS, Pang W, Gorochowski TE, Grierson CS, Marucci L. Bridging the gap between mechanistic biological models and machine learning surrogates. PLoS Comput Biol. 2023;19(4):e1010988. doi:10.1371/journal.pcbi.1010988.

21. Wang S, Fan K, Luo N, Cao Y, Wu F, Zhang C, et al. Massive computational acceleration by using neural networks to emulate mechanism-based biological models. Nat Commun. 2019;10(1):4354. doi:10.1038/s41467-019-12342-y.

22. Liang L, Liu M, Martin C, Sun W. A machine learning approach as a surrogate of finite element analysis–based inverse method to estimate the zero-pressure geometry of human thoracic aorta. Numer Methods Biomed Eng. 2018;34(8):e3103. doi:10.1002/cnm.3103.

23. Davies V, Noè U, Lazarus A, Gao H, MacDonald B, Berry C, et al. Fast Parameter Inference in a Biomechanical Model of the Left Ventricle by Using Statistical Emulation. Journal of the Royal Statistical Society Series C: Applied Statistics. 2019;68(5):1555–1576. doi:10.1111/rssc.12374.

24. Crawshaw M. Multi-Task Learning with Deep Neural Networks: A Survey; 2020. Available from: https://arxiv.org/abs/2009.09796.

25. Selvaraj P, Wenger EA, Gerardin J. Seasonality and heterogeneity of malaria transmission determine success of interventions in high-endemic settings: a modeling study. BMC Infect Dis. 2018;18(1):413. doi:10.1186/s12879-018-3319-y.

26. Molineaux L, Gramiccia G. The Garki project: research on the epidemiology and control of malaria in the Sudan savanna of West Africa; 1980.

27. Smith T, Charlwood JD, Kihonda J, Mwankusye S, Billingsley P, Meuwissen J, et al. Absence of seasonal variation in malaria parasitaemia in an area of intense seasonal transmission. Acta Tropica. 1993;54(1):55–72. doi:10.1016/0001-706X(93)90068-M.

28. Rogier C, Tall A, Diagne N, Fontenille D, Spiegel A, Trape JF. Plasmodium falciparum clinical malaria: lessons from longitudinal studies in Senegal. Parassitologia. 1999;41(1-3):255–259.

29. Ouédraogo AL, Gonçalves BP, Gnémé A, Wenger EA, Guelbeogo MW, Ouédraogo A, et al. Dynamics of the Human Infectious Reservoir for Malaria Determined by Mosquito Feeding Assays and Ultrasensitive Malaria Diagnosis in Burkina Faso. J Infect Dis. 2016;213(1):90–99. doi:10.1093/infdis/jiv370.

30. Zhu Y, Zabaras N. Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification. Journal of Computational Physics. 2018;366:415–447. doi:10.1016/j.jcp.2018.04.018.

31. Westermann P, Evins R. Using Bayesian deep learning approaches for uncertainty-aware building energy surrogate models. Energy and AI. 2021;3:100039. doi:10.1016/j.egyai.2020.100039.

32. Leung S, Windbichler N, Wenger EA, Bever CA, Selvaraj P. Population replacement gene drive characteristics for malaria elimination in a range of seasonal transmission settings: a modelling study. Malar J. 2022;21(1):226. doi:10.1186/s12936-022-04242-2.

33. Lin B, Ye F, Zhang Y, Tsang IW. Reasonable Effectiveness of Random Weighting: A Litmus Test for Multi-Task Learning. 2021;doi:10.48550/ARXIV.2111.10603.

34. Kingma DP, Ba J. Adam: A Method for Stochastic Optimization. 2014;doi:10.48550/ARXIV.1412.6980.

35. Li L, Jamieson K, Rostamizadeh A, Gonina E, Hardt M, Recht B, et al. A System for Massively Parallel Hyperparameter Tuning. 2018;doi:10.48550/ARXIV.1810.05934.

36. Prabhu VU, Birhane A. Large image datasets: A pyrrhic win for computer vision?; 2020. Available from: https://arxiv.org/abs/2006.16923.

37. Liu Y, Cao J, Liu C, Ding K, Jin L. Datasets for Large Language Models: A Comprehensive Survey; 2024. Available from: https://arxiv.org/abs/2402.18041.

38. Kudela J, Matousek R. Recent advances and applications of surrogate models for finite element method computations: a review. Soft Comput. 2022;26(24):13709–13733. doi:10.1007/s00500-022-07362-8.

39. Li S, Zhao Y, Varma R, Salpekar O, Noordhuis P, Li T, et al.. PyTorch Distributed: Experiences on Accelerating Data Parallel Training; 2020. Available from: https://arxiv.org/abs/2006.15704.

40. Kendall A, Gal Y, Cipolla R. Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics; 2017. Available from: https://arxiv.org/abs/1705.07115.

41. Zhou C, Li Q, Li C, Yu J, Liu Y, Wang G, et al. A Comprehensive Survey on Pretrained Foundation Models: A History from BERT to ChatGPT. 2023;doi:10.48550/ARXIV.2302.09419.

42. Yuan L, Chen D, Chen YL, Codella N, Dai X, Gao J, et al. Florence: A New Foundation Model for Computer Vision. 2021;doi:10.48550/ARXIV.2111.11432.

43. Xu XRS, Bulger EA, Gantz VM, Klanseck C, Heimler SR, Auradkar A, et al. Active Genetic Neutralizing Elements for Halting or Deleting Gene Drives. Molecular Cell. 2020;80(2):246–262.e4. doi:10.1016/j.molcel.2020.09.003.

44. Du M, Liu N, Hu X. Techniques for Interpretable Machine Learning; 2018. Available from: https://arxiv.org/abs/1808.00033.

45. Wu X, Huang S, Wang W, Wei F. Multi-Head Mixture-of-Experts; 2024. Available from: https://arxiv.org/abs/2404.15045.

46. Kaufmann T, Weng P, Bengs V, Hüllermeier E. A Survey of Reinforcement Learning from Human Feedback; 2023. Available from: https://arxiv.org/abs/2312.14925.

47. Dash T, Chitlangia S, Ahuja A, Srinivasan A. Incorporating Domain Knowledge into Deep Neural Networks; 2021. Available from: https://arxiv.org/abs/2103.00180.

48. Taylor R, Ojha V, Martino I, Nicosia G. Sensitivity Analysis for Deep Learning: Ranking Hyper-parameter Influence. In: 2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI). Washington, DC, USA: IEEE; 2021. p. 512–516. Available from: https://ieeexplore.ieee.org/document/9643336/.

49. Jain P, Shashaani S, Byon E. Simulation Model Calibration with Dynamic Stratification and Adaptive Sampling; 2024. Available from: https://arxiv.org/abs/2401.14558.

50. Reiker T, Golumbeanu M, Shattock A, Burgert L, Smith TA, Filippi S, et al. Emulator-based Bayesian optimization for efficient multi-objective calibration of an individual-based model of malaria. Nat Commun. 2021;12(1):7212. doi:10.1038/s41467-021-27486-z.

51. Liu B, Zhang Q, Gielen GGE. A Gaussian Process Surrogate Model Assisted Evolutionary Algorithm for Medium Scale Expensive Optimization Problems. IEEE Trans Evol Computat. 2014;18(2):180–192. doi:10.1109/TEVC.2013.2248012.

52. Gal Y, Ghahramani Z. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning; 2016. Available from: http://arxiv.org/abs/1506.02142.

53. Lakshminarayanan B, Pritzel A, Blundell C. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles; 2017. Available from: http://arxiv.org/abs/1612.01474.

54. Thiagarajan JJ, Anirudh R, Narayanaswamy V, Bremer PT. Single Model Uncertainty Estimation via Stochastic Data Centering; 2022. Available from: http://arxiv.org/abs/2207.07235.

55. Lahlou S, Jain M, Nekoei H, Butoi VI, Bertin P, Rector-Brooks J, et al.. DEUP: Direct Epistemic Uncertainty Prediction; 2023. Available from: http://arxiv.org/abs/2102.08501.

56. Yosinski J, Clune J, Bengio Y, Lipson H. How transferable are features in deep neural networks?; 2014. Available from: http://arxiv.org/abs/1411.1792.

57. Han K, Xiao A, Wu E, Guo J, Xu C, Wang Y. Transformer in Transformer; 2021. Available from: http://arxiv.org/abs/2103.00112.

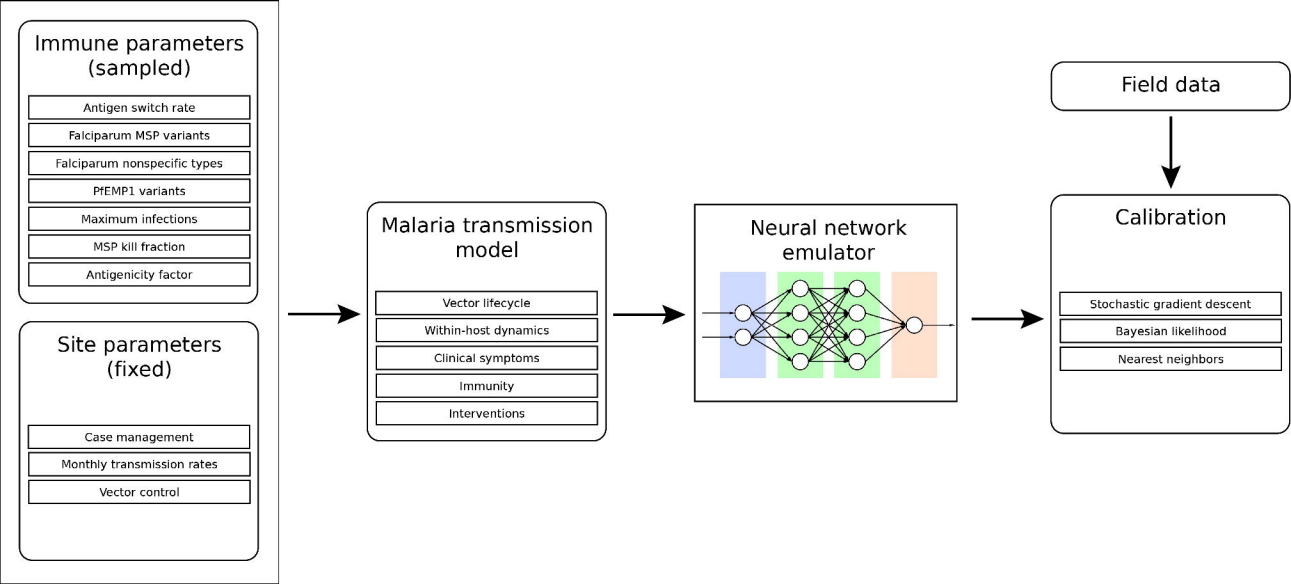
58. Goyal P, Dollár P, Girshick R, Noordhuis P, Wesolowski L, Kyrola A, et al.. Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour; 2018. Available from: http://arxiv.org/abs/1706.02677.

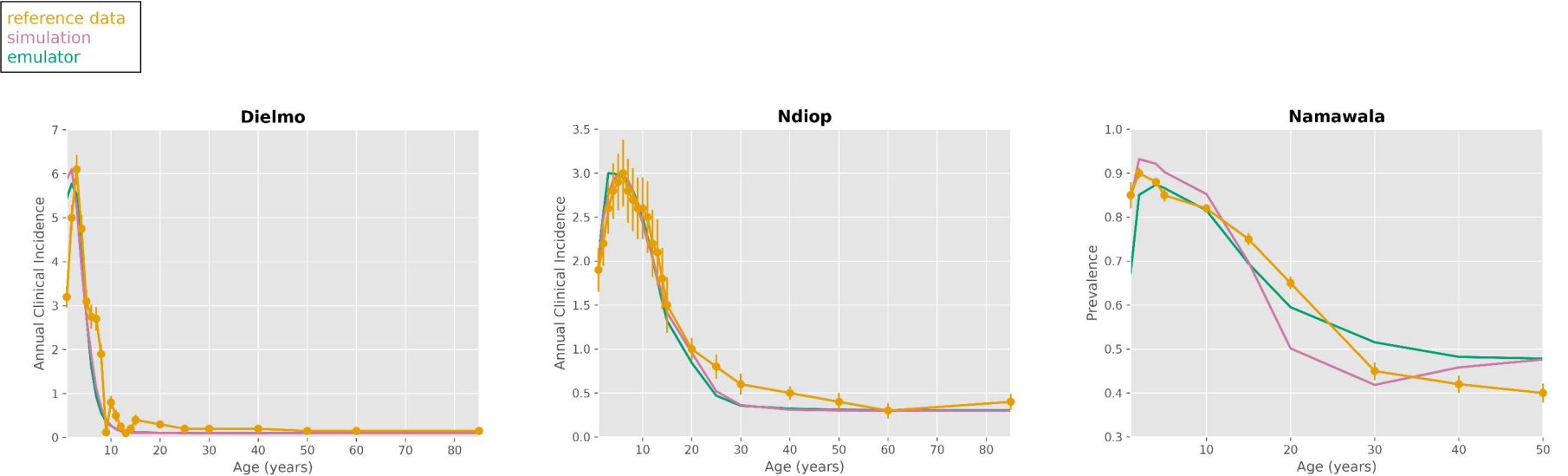
59. Zhuang F, Qi Z, Duan K, Xi D, Zhu Y, Zhu H, et al.. A Comprehensive Survey on Transfer Learning; 2019. Available from: https://arxiv.org/abs/1911.02685.

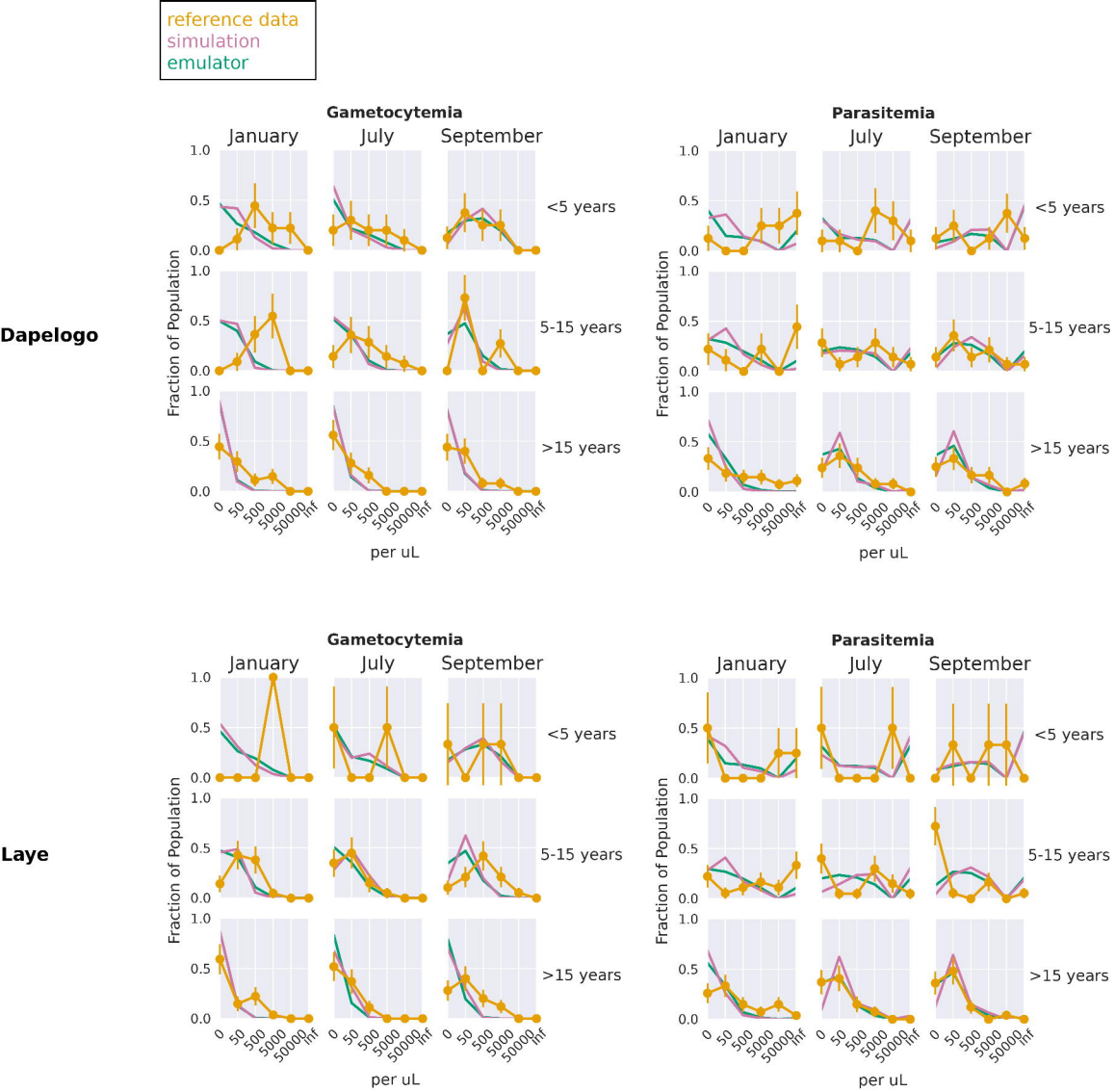
60. McCarthy KA, Wenger EA, Huynh GH, Eckhoff PA. Calibration of an intrahost malaria model and parameter ensemble evaluation of a pre-erythrocytic vaccine. Malar J. 2015;14(1):6. doi:10.1186/1475-2875-14-6.

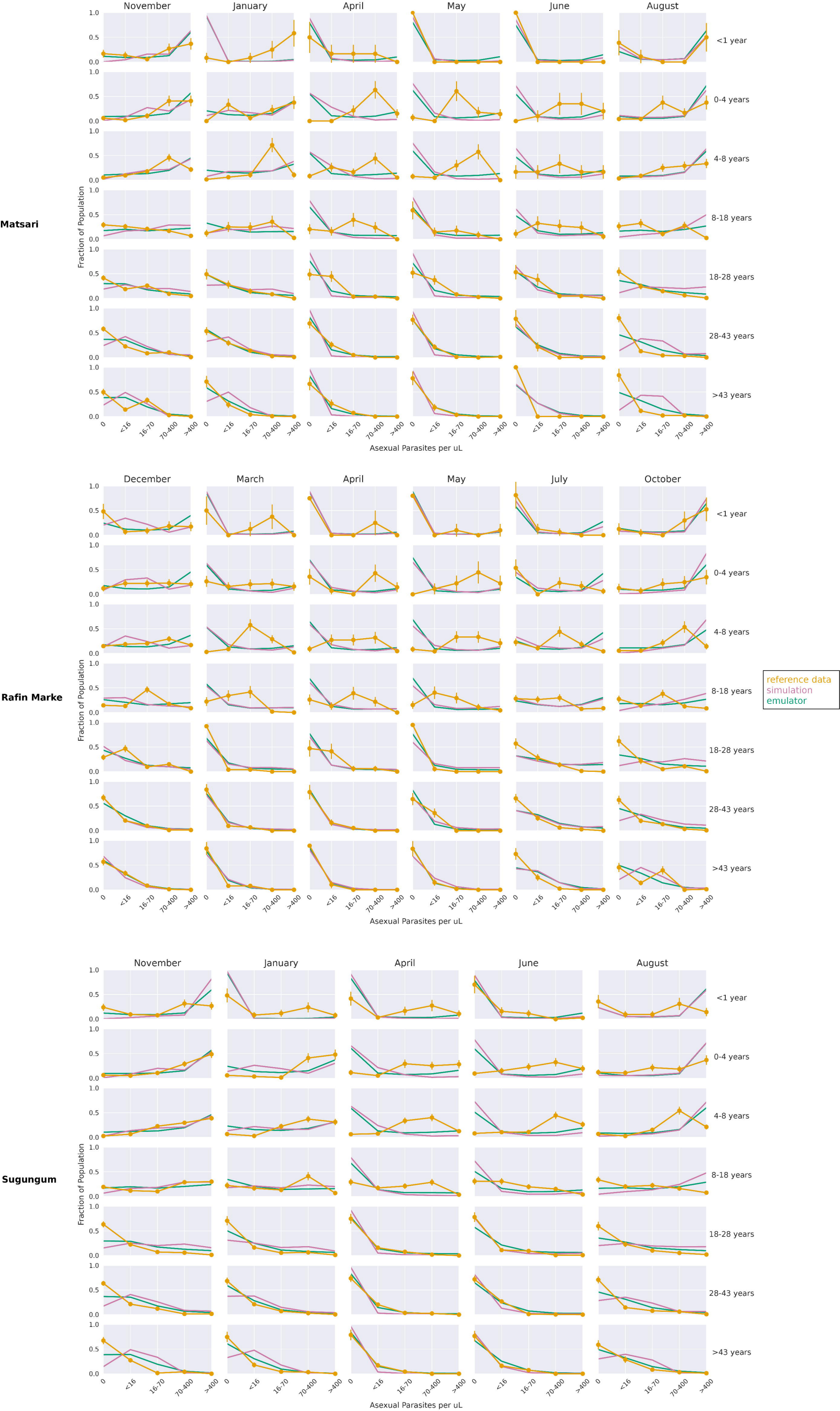
61. Mwanga EP, Siria DJ, Mshani IH, Mwinyi SH, Abbasi S, Jimenez MG, et al. Rapid classification of epidemiologically relevant age categories of the malaria vector, Anopheles funestus. Parasites Vectors. 2024;17(1):143. doi:10.1186/s13071-024-06209-5.

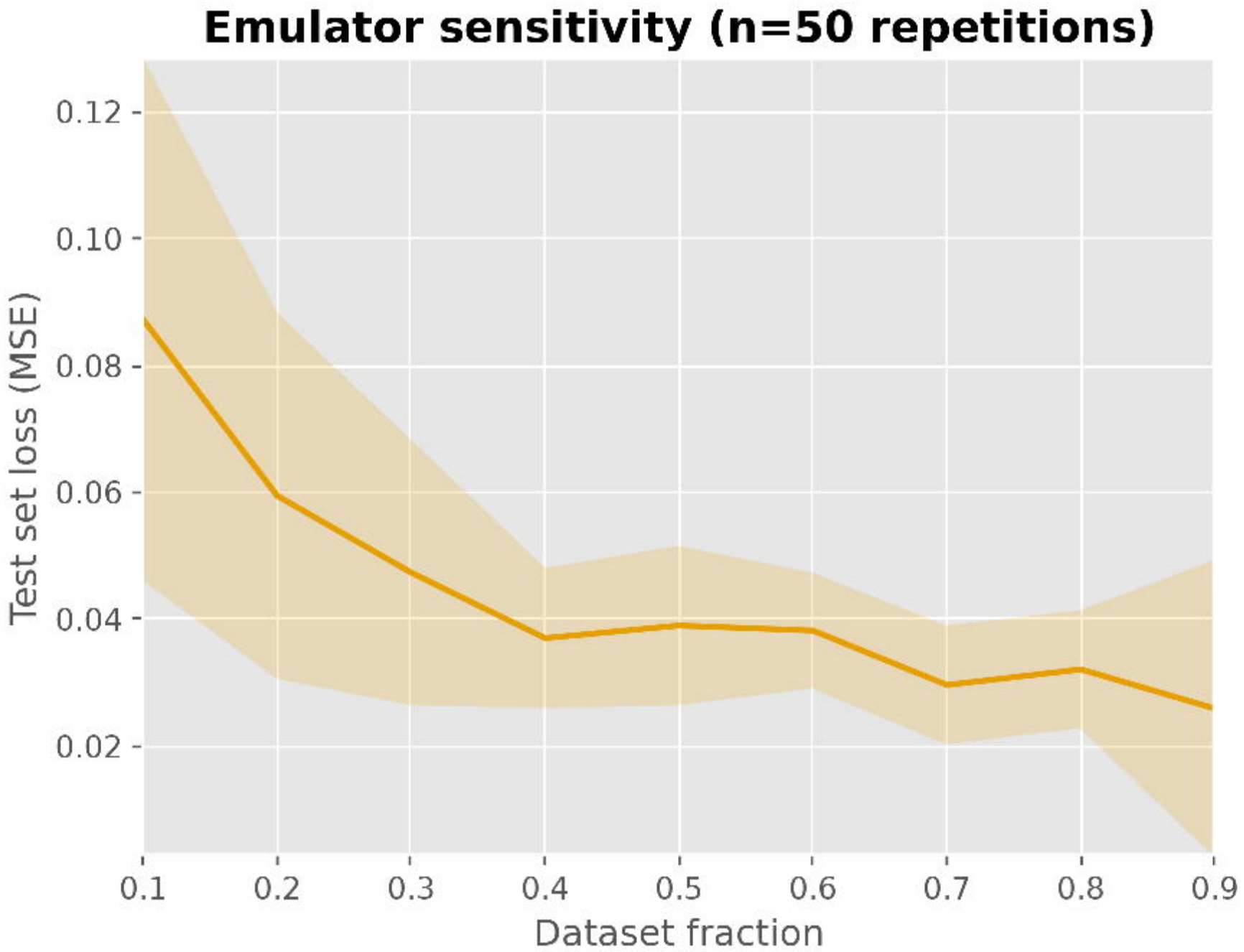
62. Li D, Hegde S, Sunil Kumar A, Zacharias A, Mehta P, Mukthineni V, et al. Towards transforming malaria vector surveillance using VectorBrain: a novel convolutional neural network for mosquito species, sex, and abdomen status identifications. Sci Rep. 2024;14(1):23647. doi:10.1038/s41598-024-71856-8.

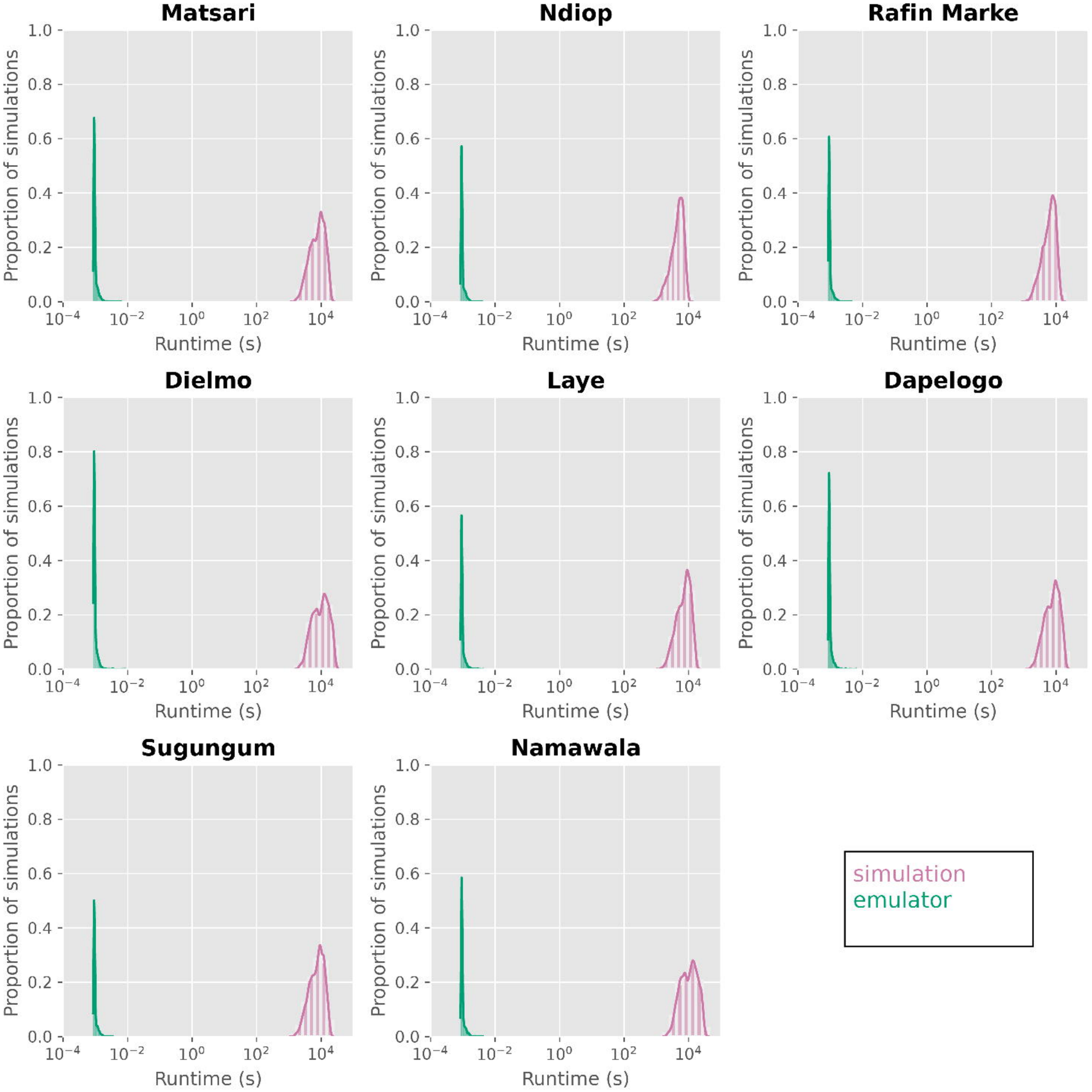

Immune parameters (sampled)
Antigen switch rate
Falciparum MSP variants
Falciparum nonspecific types
PfEMP1 variants
Maximum infections
MSP kill fraction
Antigenicity factor
Site parameters (fixed)
Case management
Monthly transmission rates
Vector control
Malaria transmission model
Vector lifecycle
Within-host dynamics
Clinical symptoms
Immunity
Interventions
Neural network emulator
Field data
Calibration
Stochastic gradient descent
Bayesian likelihood
Nearest neighbors


reference data
simulation
emulator
Dielmo
Ndiop
Namawala
Annual Clinical Incidence
Prevalence
Age (years)


reference data
simulation
emulator
Dapelogo
Laye
Gametocytemia
Parasitemia
January
July
September
<5 years
5-15 years
>15 years
Fraction of Population
per uL
0
50
500
5000
50000
Inf
0.0
0.5
1.0


Matsari
November
January
April
May
June
August
<1 year
0-4 years
4-8 years
8-18 years
18-28 years
28-43 years
>43 years
Fraction of Population
Asexual Parasites per uL
0
<16
16-70
70-400
>400
Rafin Marke
December
March
April
May
July
October
Sugungum
reference data
simulation
emulator


Emulator sensitivity (n=50 repetitions)
Test set loss (MSE)
0.12
0.10
0.08
0.06
0.04
0.02
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Dataset fraction


Matsari
Ndiop
Rafin Marke
Dielmo
Laye
Dapelogo
Sugungum
Namawala
Proportion of simulations
Runtime (s)
simulation
emulator