Deep Learning From Phylogenies To Uncover The Epidemiological Dynamics Of Outbreaks

J. Voznica, A. Zhukova, V. Boskova, E. Saulnier, F. Lemoine, M. Moslonka-Lefebvre, O. Gascuel

DEEP LEARNING FROM PHYLOGENIES TO UNCOVER THE EPIDEMIOLOGICAL DYNAMICS OF OUTBREAKS

Voznica J[1,2,3*], Zhukova A[1,4,5,6*], Boskova V[7], Saulnier E[1], Lemoine F[1,4], Moslonka-Lefebvre M[1], Gascuel O[1,8*]

AFFILIATIONS

[1] Institut Pasteur, Université Paris Cité, Unité Bioinformatique Evolutive, Paris, FRANCE

[2] Université de Paris, Paris, FRANCE

[3] Institut de Biologie de l'École Normale Supérieure, Ecole Normale Supérieure, CNRS, INSERM, Université Paris Sciences et Lettres, Paris, FRANCE

[4]Institut Pasteur, Université Paris Cité, Bioinformatics and Biostatistics Hub, Paris, FRANCE

[5] Institut Pasteur, Université Paris Cité, Epidemiology and Modelling of Antibiotic Evasion, Paris, FRANCE

[6] Université Paris-Saclay, UVSQ, Inserm, CESP, Villejuif, FRANCE

[7] Center for Integrative Bioinformatics Vienna, Max Perutz Labs, University of Vienna and Medical University of Vienna, Vienna, AUSTRIA

[8] Institut de Systématique, Evolution, Biodiversité (UMR 7205 - CNRS, Muséum National d'Histoire Naturelle, SU, EPHE, UA), Paris, FRANCE

* CO-CORRESPONDING AUTHORS

voznica.jakub@gmail.com (JV), anna.zhukova@pasteur.fr (AZ), olivier.gascuel@mnhn.fr (OG)

ORCIDs

Jakub Voznica: 0000-0002-0530-5837 --- Anna Zhukova: 0000-0003-2200-7935

Frédéric Lemoine: 0000-0001-9576-4449 --- O. Gascuel: 0000-0002-9412-9723

ABSTRACT

Widely applicable, accurate and fast inference methods in phylodynamics are needed to fully profit from the richness of genetic data in uncovering the dynamics of epidemics. Standard methods, including maximum-likelihood and Bayesian approaches, generally rely on complex mathematical formulae and approximations, and do not scale with dataset size. We develop a likelihood-free, simulation-based approach, which combines deep learning with (1) a large set of summary statistics measured on phylogenies or (2) a complete and compact representation of trees, which avoids potential limitations of summary statistics and applies to any phylodynamics model. Our method enables both model selection and estimation of epidemiological parameters from very large phylogenies. We demonstrate its speed and accuracy on simulated data, where it performs better than the state-of-the-art methods. To illustrate its applicability, we assess the dynamics induced by superspreading individuals in an HIV dataset of men-having-sex-with-men in Zurich. Our tool PhyloDeep is available on github.com/evolbioinfo/phylodeep.

KEYWORDS

Deep learning; phylodynamics; molecular epidemiology; tree simulation and representation; HIV.

INTRODUCTION

Pathogen phylodynamics is a field combining phylogenetics and epidemiology[1]. Viral or bacterial samples from patients are sequenced and used to infer a phylogeny, which describes the pathogen's spread among patients. The tips of such phylogenies represent sampled pathogens, and the internal nodes transmission events. Moreover, transmission events can be dated and thereby provide hints on transmission patterns. Such information is extracted by phylodynamic methods to estimate epidemiological and population dynamic parameters[2-4], assess the impact of population structure[2,5], and reveal the origins of epidemics[6].

Birth-death models[7] incorporate easily interpretable parameters common to standard infectious-disease epidemiology, such as basic reproduction number R_0, infectious period, *etc.* In contrast to the standard epidemiological models, the birth-death models can be applied to estimate parameters from phylogenetic trees[8]. In these models, births represent transmission events, while deaths represent removal events for example due to treatment or recovery. Upon a patient's removal, their pathogens can be sampled, producing tips in the tree.

Here we focus on three specific, well-established birth-death models (**Fig. 1)**: birth-death model (BD)[8,9], birth-death model with exposed and infectious classes (BDEI)[5,10,11], and birth-death model with superspreading (BDSS)[5,12]. These models were deployed using BEAST2[12,13] to study the phylodynamics of such diverse pathogens as Ebola virus[10], Influenza virus[12], Human Immunodeficiency Virus (HIV)[5], Zika[14] or SARS-CoV-2[15]. Using these models, we will demonstrate the reliability of our deep learning-based approach.

While a great effort has been invested in the development of new epidemiological models in phylodynamics, the field has been slowed down by the mathematical complexity inherent to these models. BD, the simplest model, has a closed form solution for the likelihood formula of a tree for a given set of parameters[8,10], but more complex models (*e.g.,* BDEI and BDSS) rely on a set of ordinary differential equations (ODEs) that cannot be solved analytically. To estimate parameter values through maximum-likelihood and Bayesian approaches, these ODEs must be approximated numerically for each tree node[5,10-12]. These calculations become difficult as the tree size increases, resulting in numerical instability and inaccuracy[12], as we will see below.

Inference issues with complex models are typically overcome by approximate Bayesian computation (ABC)[16,17]. ABC is a simulation-based technique relying on a rejection algorithm[18], where from a set of simulated phylogenies

within a given prior (values assumed for parameter values), those closest to the analysed phylogeny are retained and give the posterior distribution of the parameters. This scheme relies on the definition of a set of summary statistics aimed at representing a phylogeny and on a distance measure between trees. The ABC approach is thus sensitive to the choice of the summary statistics and distance function (*e.g.,* Euclidean distance). To address this issue Saulnier *et al.*[19] developed a large set of summary statistics. In addition, they used a regression step to select the most relevant statistics and to correct for the discrepancy between the simulations retained in the rejection step and the analysed phylogeny. They observed that the sensitivity to the rejection parameters were greatly attenuated thanks to regression (see also Blum *et al.*[20]).

Our work is a continuation of regression-based ABC, and aims at overcoming its main limitations. Using the approximation power of currently available neural network architectures, we propose a likelihood-free method relying on deep learning from millions of trees of varying size simulated within a broad range of parameter values. By doing so, we bypass the rejection step, which is both time consuming with large simulation sets, and sensitive to the choice of the distance function and summary statistics. To describe simulated trees and use them as input for the deep learner, we develop two tree representations: (1) a large set of summary statistics mostly based on Saulnier *et al.*[19], and (2) a complete and compact vectorial representation of phylogenies, including both the tree topology and branch lengths. The summary statistics are derived from our understanding and knowledge of the epidemiological processes. However, they can be incomplete and thus miss some important aspects of the studied phylogenies, which can potentially result in low accuracy during inference. Moreover, it is expected that new phylodynamic models will require design of new summary statistics, as confirmed by our results with BDSS. In contrast, our vectorial representation is a raw data representation that preserves all information contained in the phylogeny and thus should be accurate and deployable on any new model, provided the model parameters are identifiable. Our vectorial representation naturally fits with deep learning methods, especially the convolutional architectures, which have already proven their ability to extract relevant features from raw representations, for example in image analysis[21,22] or weather prediction[23].

In the following, we introduce our vectorial tree representation and the new summary statistics designed for BDSS. We then present the deep learning architectures trained on these representations and evaluate their accuracy on simulated datasets in terms of both parameter estimation and model selection. We show that our approach applies not

only to trees of the same size as the training instances, but also to very large trees with thousands of tips through the analysis of their subtrees. The results are compared to those of the gold standard method, BEAST2[12,13]. Lastly, we showcase our methods on an HIV dataset[24,25] from the men-having-sex-with-men (MSM) community from Zurich. All technical details are provided in **Methods** and **Supplementary Information**. Our methods and tools are implemented in the PhyloDeep software, which is available on GitHub (github.com/evolbioinfo/phylodeep), PyPi (pypi.org/project/phylodeep) and Docker Hub (hub.docker.com/r/evolbioinfo/phylodeep).

RESULTS

Neural networks are trained on numerical vectors from which they can learn regression and classification tasks. We trained such networks on phylogenetic trees to estimate epidemiological parameters (regression) and select phylodynamic models (classification). We undertook two strategies for representing phylogenetic trees as numerical vectors, which we describe first, before showing the results with simulated and real data.

Summary statistics (SS) representation. We used a set of 83 SS developed by Saulnier *et al.*[19]: 26 measures of branch lengths, such as median of both internal and tip branch lengths; 8 measures of tree topology, such as tree imbalance; 9 measures on the number of lineages through time, such as time and height of its maximum; and 40 coordinates representing the lineage-through-time (LTT) plot. To capture more information on the phylogenies generated by the BDSS model, we further enriched these SS with 14 new statistics on transmission chains describing the distribution of the duration between consecutive transmissions (internal tree nodes). Our SS are diverse, complementary and somewhat redundant. We used feed-forward neural networks (FFNN) with several hidden layers (**Fig. 2 b (i)**) that select and combine relevant information from the input features. In addition to SS, we provide both the tree size (*i.e.,* number of tips) and the sampling probability used to generate the tree, as input to our FFNN (**Fig. 2 a (vi)**). We will refer to this method as FFNN-SS.

Compact vectorial tree representation. While converting raw information in the form of a phylogenetic tree into a set of SS, information loss is unavoidable. This means not only that the tree cannot be fully reconstructed from its SS, but also that depending on how much useful and relevant information is contained in the SS, the neural network may fail to solve the problem at hand. As an alternative strategy to SS, and to prevent information loss in the tree representation, we developed a representation called 'Compact Bijective Ladderized Vector' (CBLV).

Several vectorial representations of trees based either on polynomial[26,27], Laplacian spectrum[28] or F matrices[29] have been developed previously. However, they represent the tree shape but not the branch lengths[26] or may lose information on trees[28]. In addition, some of these representations require vectors or matrices of quadratic size with respect to the number of tips[29], or are based on complex coordinate systems of exponential size[27].

Inspired by these approaches, we designed our concise, easily computable, compact, and bijective (i.*e.* 1-to-1) tree representation that applies to trees of variable size and is appropriate as machine learning input. To obtain this representation, we first ladderize the tree, that is, for each internal node, the descending subtree containing the most recently sampled tip is rotated to the left, **Fig. 2 a (ii)**. This ladderization step does not change the tree but facilitates learning by standardizing the input data. Moreover, it is consistent with trees observed in real epidemiological datasets, for example Influenza, where ladder-like trees reflect selection and are observed for several pathogens[1]. Then, we perform an inorder traversal[30] of the ladderized tree, during which we collect in a vector for each visited internal node its distance to the root and for each tip its distance to the previously visited internal node. In particular, the first vector entry corresponds to the tree height. This transformation of a tree into a vector is bijective, in the sense that we can unambiguously reconstruct any given tree from its vector representation (**Supplementary Fig. 1**). The vector is as compact as possible, and its size grows linearly with the number of tips. We complete this vector with zeros to reach the representation length of the largest tree contained in our simulation set, and we add the sampling probability used to generate the tree (or an estimate of it when analysing real data; **Fig. 2 a (v), b (i))**.

Bijectivity combined with ladderization facilitates the training of neural networks, which do not need to learn that different representations correspond to the same tree. However, unlike our SS, this full representation does not have any high-level features. In CBLV identical subtrees will have the same representation in the vector whenever the roots of these subtrees have the same height, while the vector representation of the tips in such subtrees will be the same no matter the height of the subtree's root. Similar subtrees will thus result in repeated patterns along the representation vector. We opted for Convolutional Neural Networks (CNN), which are designed to extract information on patterns in raw data. Our CNN architecture (**Fig. 2 b (ii)**) includes several convolutional layers that perform feature extraction, as well as maximum and average pooling layers that select relevant features and keep feature maps of reasonable dimensions. The output of the CNN is then fed into a FFNN that combines the patterns found in the input to perform predictions. In the rest of the manuscript, we refer to this method as CNN-CBLV.

Simulated datasets

For each phylodynamic model (BD, BDEI, BDSS), we simulated 4 million trees, covering a large range of values for each parameter of epidemiological interest (R_0, infectious period: $1/\gamma$, incubation period: $1/\varepsilon$, the fraction at equilibrium of superspreading individuals: f_{SS}, and the superspreading transmission ratio: X_{SS}). Of the 4 million trees, 3.99 million were used as a training set, and 10,000 as a validation set for early stopping in the training phase[31]. Additionally, we simulated another 10,000 trees, which we used as a testing set, out of which 100 were also evaluated with the gold standard methods, BEAST2 and TreePar, which are more time consuming. Another 1 million trees were used to define confidence intervals for estimated parameters. For BD and BDEI we considered two settings: one with small trees (50 to 199 tips, in **Supplementary Fig. 2**) and a second with large trees (200 to 500 tips, **Fig. 3**). For BDSS, we considered only the setting with large trees, as the superspreading individuals are at a low fraction and cannot be detected in small trees. Lastly, we investigated the applicability of our approach to very large data sets, which are increasingly common with viral pathogens. To this goal, we generated for each model 10,000 'huge' trees, with 5,000 to 10,000 tips each and with the same parameter ranges as used with the small and large trees. To estimate the parameter values of a huge tree, we extracted a nearly complete coverage of this tree by disjoint subtrees with 50 to 500 leaves. Then, we predicted the parameter values for every subtree using our NNs, and averaged subtree predictions to obtain parameter estimates for the huge tree.

To increase the generality of our approach and avoid the arbitrary choice of the time scale (one unit can be a day, a week, or a year), we rescaled all trees and corresponding epidemiological parameters, such that the average branch length in a tree was equal to 1. After inference, we rescaled the estimated parameter values back to the original time scale.

Neural networks yield more accurate parameter estimates than gold standard methods

We compared accuracy of parameter estimates yielded by our deep learning methods and those yielded by two state-of-the-art phylodynamics inference tools, BEAST2[12,13] and TreePar[5]. The comparison shows that our deep learning methods trained with SS and CBLV are either comparable (BD) or more accurate (BDEI and BDSS) than the state-of-the-art inference methods (**Fig. 3, Supplementary Tab. 1**). The simple BD model has a closed form solution for the likelihood function, and thus BEAST2 results are optimal in theory[8,9]. Our results with BD are similar to those obtained with BEAST2, and thus nearly optimal as well. For BDEI and BDSS our results are more accurate than

BEAST2, which is likely explained by numerical approximations of likelihood calculations in BEAST2[5,10,11] for these models. These approximations can lead BEAST2 to a lack of convergence (2% cases for BDEI and 15% cases for BDSS) or a convergence to local optima. We suspect BEAST2 of converging to local optima when it converged to values with high relative error (>1.0; 8% cases for BDEI and 11% cases for BDSS, **Fig. 3 b-c**). Furthermore, our deep learning approaches showed a lower bias in parameter estimation than BEAST2 (**Supplementary Tab. 2**). As expected, both approaches, FFNN-SS and CNN-CBLV, get more accurate with larger trees (**Supplementary Fig. 3**).

We tried to perform maximum likelihood estimation (MLE) implemented in the TreePar package[5] on the same trees as well. While MLE under BD model on simulations yielded as accurate results as BEAST2, for more complex models it showed overflow and underflow issues (*i.e.,* reaching infinite values of likelihood) and yielded inaccurate results, such as more complex models (BDEI, BDSS) having lower likelihood than a simpler, nested one (BD) for a part of simulations. These issues were more prominent for larger trees. TreePar developers confirmed these limitations and suggested using the latest version of BEAST2 instead.

To further explain the performance of our NNs, we computed the likelihood value of their parameter estimates. This was easy with the BD model since we have a closed form solution for the likelihood function. The results with this model (**Supplementary Tab. 3**, using TreePar) showed that the likelihoods of both FFNN-SS and CNN-CBLV estimates are similar to BEAST2's, which explains the similar accuracy of the three methods (**Fig. 3**). We also computed the likelihood of the 'true' parameter values used to simulate the trees, in order to have an independent and solid assessment. If a given method tends to produce higher likelihood than that of the true parameter values, then it performs well in terms of likelihood optimization, as optimizing further should not result in higher accuracy. The results (**Supplementary Tab. 3**) were again quite positive, as BEAST2 and our NNs achieved a higher likelihood than the true parameter values for ~70% of the trees, with a significant mean difference. With BDEI and BDSS, applying the same approach proved difficult due to convergence and numerical issues, with both BEAST2 and TreePar (see above). For the partial results we obtained, the overall pattern seems to be similar to that with BD: the NNs obtain highly likely solutions, with similar likelihood as BEAST2's (when it converges and produces reasonable estimates), and significantly higher likelihood than that of the true parameter values. All these results are remarkable, as the NNs do not explicitly optimize the likelihood function associated to the models, but use a radically different learning approach, based on simulation.

Neural networks are fast inference methods

We compared the computing time required by each of our inference methods. All computing times were estimated for a single thread of our cluster, except for the training of neural architectures where we used our GPU farm. Neural networks require heavy computing time in the learning phase; for example, with BDSS (the most complex model), simulating 4M large trees requires ~800 CPU hours, while training FFNN-SS and CNN-CBLV requires ~5 and ~150 hours, respectively. However, with NNs, inference is almost instantaneous and takes ~0.2 CPU seconds per tree on average, including encoding the tree in SS or CBLV, which is the longest part. For comparison, BEAST2 inference under the BD model with 5 million MCMC steps takes on average ~0.2 CPU hours per tree, while inference under BDEI and BDSS with 10 million MCMC steps takes ~55 CPU hours and ~80 CPU hours per tree, respectively. In fact, the convergence time of BEAST2 is usually faster (~6 CPU hours with BDEI and BDSS), but can be very long in some cases, to the point that convergence is not observed after 10 million steps (see above).

Neural networks have high generalization capabilities and apply to very large data sets

In statistical learning theory[31], generalization relates to the ability to predict new samples drawn from the same distribution as the training instances. Generalization is opposed to rote learning and overfitting, where the learned classifier or regressor predicts the training instances accurately, but new instances extracted from the same distribution or population poorly. The generalization capabilities of our NNs were demonstrated, as we used independent testing sets in all our experiments (**Fig. 3**). However, we expect poor results with trees that depart from the training distribution, for example showing very high R_0, while our NNs have been trained with R_0 in the range [1, 5]. If, for a new study, larger or different parameter ranges are required, we must retrain the NNs with *ad hoc* simulated trees. However, a strength of NNs is that thanks to their flexibility and approximation power, very large parameter ranges can be envisaged, to avoid repeating training sessions too often.

Another sensible issue is that of the size of the trees. Our NNs have been trained with trees of 50-to-199 tips (small) and 200-to-500 tips (large), that is, trees of moderate size (but already highly time consuming in a Bayesian setting, for the largest ones). Thus, we tested the ability to predict the parameters of small trees using NNs trained on large trees, and vice versa, the ability to predict large trees with NNs trained on small trees. The results (**Supplementary Fig. 4**) are surprisingly good, especially with summary statistics (FFNN-SS) which are little impacted by these changes of scale as they largely rely on means (*e.g.*, of branch lengths[19]). This shows unexpected generalization capabilities

of the approach regarding tree size. Most importantly, the approach can accurately predict huge trees (**Fig. 4**) using their subtrees and the means of the corresponding parameter estimates, in ~1 CPU minute. This extends the applicability of the approach to data sets that cannot be analysed today, unless using similar tree decomposition and very long calculations to analyse all subtrees.

Neural networks are accurate methods for model selection

We trained CNN-CBLV and FFNN-SS on simulated trees to predict the birth-death model under which they were simulated (BD or BDEI for small trees; BD, BDEI or BDSS for large trees). Note that for parameters shared between multiple models, we used identical parameter value ranges across all these models (**Supplementary Tab. 4**). Then, we assessed the accuracy of both of our approaches on 100 simulations obtained with each model and compared it with the model selection under BEAST2 based on Akaike information criterion through Markov Chain Monte Carlo (AICM)[32,33]. The AICM, similar to deviance information criterion (DIC) by Gelman *et al.*[32], does not add computational load and is based on the average and variance of posterior log-likelihoods along the Markov Chain Monte Carlo (MCMC).

FFNN-SS and CNN-CBLV have similar accuracy (**Supplementary Tab. 5**), namely 92% for large trees (BD vs BDEI vs BDSS), and accuracy of 91% and 90%, respectively, for small trees (BD vs BDEI). BEAST2 yielded an accuracy of 91% for large trees and 87% for small trees. The non-converging simulations were not considered for any of these methods (*i.e.,* 5% simulations for small trees and 24% for large trees).

The process of model selection with a neural network is as fast as the parameter inference (~0.2 CPU seconds per tree). This represents a practical, fast and accurate way to perform model selection in phylodynamics.

Neural networks are well suited to learn complex models

To assess the complexity of learned models, we explored other inference methods, namely: (1) linear regression as a baseline model trained on summary statistics (LR-SS); (2) FFNN trained directly on CBLV (FFNN-CBLV); (3) CNN trained on Compact Random Vector (CNN-CRV), for which the trees were randomly rotated, instead of being ladderized as in **Fig. 2 (ii)**; and (4) two "null models".

LR-SS yielded inaccurate results even for the BD model (**Supplementary Tab. 1**), which seems to contrast with previous findings[19], where LR approach combined with ABC performed only slightly worse than BEAST2. This can

be explained by the lack of rejection step in LR-SS, which enables to locally reduce the complexity of the relation between the representation and the inferred values to a linear one[18]. However, the rejection step requires a metric (*e.g.,* the Euclidean distance), which may or may not be appropriate depending on the model and the summary statistics. Moreover, rejection has a high computational cost with large simulation sets.

Neural networks circumvent these problems with rejection and allow for more complex, non-linear relationships between the tree representation and the inferred values to be captured. This is also reflected in our results with FFNN-CBLV and CNN-CRV, which both proved to be generally more accurate than LR-SS. However, FFNN-CBLV was substantially less accurate than CNN-CBLV (**Supplementary Tab. 1**, **Supplementary Fig. 5**). This indicates the presence of repeated patterns that may appear all along the vectorial representation of trees, such as subtrees of any size, which are better extracted by CNN than by FFNN. In its turn, CNN-CRV required larger training sets to reach an accuracy comparable to CNN-CBLV (**Supplementary Tab. 1**, **Supplementary Fig. 5**), showing that the ladderization and bijectivity of the CBLV helped the training.

To assess how much information is actually learned, we also measured the accuracy of two "null models": FFNN trained to predict randomly permuted target values; and a random predictor, where parameter values were sampled from prior distributions. Results show that the neural networks extract a considerable amount of information for most of the estimated parameters (**Supplementary Tab. 1**). The most difficult parameter to estimate was the fraction of superspreading individuals in BDSS model, with accuracy close to random predictions with small trees, but better performance as the tree size increases (**Fig. 4, Supplementary Fig. 3**).

SS is simpler, but CBLV has high potential for application to new models

FFNN-SS and CNN-CBLV show similar accuracy across all settings (**Fig. 3, Supplementary Tab. 1-2**), including when predicting huge trees from their subtrees (**Fig. 4**). The only exception is the prediction of large trees using NNs trained with small trees (**Supplementary Fig. 4**), where FFNN-SS is superior to CNN-CBLV, but this goes beyond the recommended use of the approach, as only a part of the (large) query tree is given to the (small) CNN-CBLV.

However, the use of the two representations is clearly different, and it is likely that with new models and scenarios their accuracy will differ. SS requires a simpler architecture (FFNN) and is trained faster (*e.g.*, 5 hours with large BDSS trees), with less training instances (**Supplementary Fig. 6**). However, this simplicity is obtained at cost of a

long preliminary work to design appropriate summary statistics for each new model, as was confirmed in our analyses of BDSS simulations. To estimate the parameters of this model, we added summary statistics on transmission chains on top of the SS taken from Saulnier *et al.*[19]. This improved the accuracy of superspreading fraction estimates of the FFNN-SS, so that it was comparable to the CNN-CBLV, while the accuracy for the other parameters remained similar (**Supplementary Fig. 7**). The advantage of the CBLV is its generality, meaning there is no loss of information between the tree and its representation in CBLV regardless of which model the tree was generated under. However, CBLV requires more complex architectures (CNN), more computing time in the learning phase (150 hours with large BDSS trees) and more training instances (**Supplementary Fig. 6**). Such an outcome is expected. With raw CBLV representation, the convolutional architecture is used to "discover" relevant summary statistics (or features, in machine learning terminology), which has a computational cost.

In fact, the two representations should not be opposed. An interesting direction for further research would be to combine them (*e.g.* during the FFNN phase), to possibly obtain even better results. Moreover, SS are still informative and useful (and quickly computed), in particular to perform sanity checks, both a priori and a posteriori (**Fig. 5, Supplementary Fig. 8**), or to quickly evaluate the predictability of new models and scenarios.

Showcase study of HIV in MSM subpopulation in Zurich

The Swiss HIV Cohort is densely sampled, including more than 16,000 infected individuals[24]. Datasets extracted from this cohort have often been studied in phylodynamics[8,25]. We analysed a dataset of an MSM subpopulation from Zurich, which corresponds to a cluster of 200 sequences studied previously by Rasmussen *et al.*[25], who focused on the degree of connectivity and its impact on transmission between infected individuals. Using coalescent approaches, they detected the presence of highly connected individuals at the beginning of the epidemic and estimated R_0 to be between 1.0 and 2.5. We used their tree as input for neural networks and BEAST2.

To perform analyses, one needs an estimate of the sampling probability. We considered that: (1) the cohort is expected to include around 45% of Swiss individuals infected with HIV[24]; and (2) the sequences were collected from around 56% of individuals enrolled in this cohort[34]. We used these percentages to obtain an approximation of sampling probability of 0.45*0.56 ~ 0.25 and used this value to analyse the MSM cluster. To check the robustness of our estimates, we also used sampling probabilities of 0.2 and 0.3 in our estimation procedures.

First, we performed a quick sanity check considering the resemblance of HIV phylogeny with simulations obtained with each model. Two approaches were used, both based on SS (**Supplementary Fig. 8**). Using principal component analysis (PCA), all three considered birth-death models passed the check. However, when looking at the 97 SS values in detail, namely checking whether the observed tree SS were within the [min, max] range of the corresponding simulated values, the BD and BDEI models were rejected for some of the SS (5 for both models, all related to branch lengths). Then, we performed model selection (BD vs BDEI vs BDSS) and parameter estimation using our two methods and BEAST2 (**Fig. 5 a-b**). Finally, we checked the model adequacy with a second sanity check, derived from the inferred values and SS (**Fig. 5 c, Supplementary Fig. 8)**.

Model selection with CNN-CBLV and FFNN-SS resulted in the acceptance of BDSS (probability of 1.00 versus 0.00 for BD and BDEI), and the same result was obtained with BEAST2 and AICM. These results are consistent with our detailed sanity check, and with what is known about HIV epidemiology, namely, the presence of superspreading individuals in the infected subpopulation[35] and the absence of incubation period without infectiousness such as is emulated in BDEI[36].

We then inferred parameter values under the selected BDSS model (**Fig. 5 a-b**). The values obtained with FFNN-SS and CNN-CBLV are close to each other, and the 95% CI are nearly identical. We inferred an R_0 of 1.6 and 1.7, and an infectious period of 10.2 and 9.8 years, with FFNN-SS and CNN-CBLV, respectively. Transmission by superspreading individuals was estimated to be around 9 times higher than by normal spreaders and superspreading individuals were estimated to account for around 7-8% of the population. Our R_0 estimates are consistent with the results of a previous study[8] performed on data from the Swiss cohort, and the results of Rasmussen *et al.*[25] with this dataset. The infectious period we inferred is a bit longer than that reported by *Stadler et al*, who estimated it to be 7.74 [95% CI 4.39-10.99] years[8]. The infectious period is a multifactorial parameter depending on treatment efficacy and adherence, the times from infection to detection and to the start of treatment, *etc.* In contrast to the study by *Stadler et al*, whose data were sampled in the period between 1998 and 2008, our dataset covers also the period between 2008 and 2014, during which life expectancy of patients with HIV was further extended[37]. This may explain why we found a longer infectious period (with compatible CIs). Lastly, our findings regarding superspreading are in accordance with those of Rassmussen *et al.*[25], and with a similar study in Latvia[5] based on 40 MSM sequences analysed using a likelihood approach. Although the results of the latter study may not be very accurate due to the small dataset size,

they still agree with ours, giving an estimate of a superspreading transmission ratio of 9, and 5.6% of superspreading individuals. Our estimates were quite robust to the choice of sampling probability (*e.g.,* R0 = 1.54, 1.60 and 1.66, with FFNN-SS and a sampling probability of 0.20, 0.25 and 0.30, respectively, **Fig. 5 b**).

Compared to BEAST2, the estimates of the infectious period and R_0 were similar for both approaches, but BEAST2 estimates were higher for the transmission ratio (14.5) and the superspreading fraction (10.6%). These values are in accordance with the positive bias of BEAST2 estimates that we observed in our simulation study for these two parameters, while our estimates were nearly unbiased (**Supplementary Tab. 2**).

Finally, we checked the adequacy of BDSS model by resemblance of HIV phylogeny to simulations. Using inferred 95% CI, we simulated 10,000 trees and performed PCA on SS, to which we projected the SS of our HIV phylogeny. This was close to simulations, specifically close to the densest swarm of simulations, supporting the adequacy of both the inferred values and the selected model (**Fig. 5 c**). When looking at the 97 SS in detail, some of the observed values where not in the [min, max] range of the 10,000 simulated values. However, these discordant SS were all related to the lineage-through-time plot (LTT; *e.g.*, *x* and *y* coordinates of this plot; **Supplementary Fig. 8**), consistent with the fact that the probabilistic, sampling component of the BDSS model is an oversimplification of actual sampling schemes, which depend on contact tracing, sampling campaigns and policies, etc.

DISCUSSION

In this manuscript, we presented new methods for parameter inference and model selection in phylodynamics based on deep learning from phylogenies. Through extensive simulations, we established that these methods are at least as accurate as state-of-the-art methods and capable of predicting very large trees in minutes, which cannot be achieved today by any other existing method. We also applied our deep learning methods to the Swiss HIV dataset from MSM and obtained results consistent with current knowledge of HIV epidemiology.

Using BEAST2, we obtained inaccurate results for some of the BDEI and BDSS simulations. While BEAST2 has been successfully deployed on many models and tasks, it clearly suffers from approximations in likelihood computation with these two models. However, these will likely improve in near future. In fact, we already witnessed substantial improvements done by BEAST2 developers to the BDSS model, while carrying out this research.

Both of our neural network approaches circumvent likelihood computation and thereby represent a new way of using molecular data in epidemiology, without the need to solve large systems of differential equations. This opens the door to novel phylodynamics models, which would make it possible to answer questions previously too complex to ask. This is especially true for CBLV representation, which does not require the design of new summary statistics, when applied to trees generated by new mathematical models. A direction for further research would be to explore such models, for example based on structured coalescent[38,39], or to extend the approach to macroevolution and species diversification models[40], which are closely related to epidemiological models. Other fields related to phylodynamics, such as population genetics, have been developing likelihood-free methods[41], for which our approach might serve as a source of inspiration.

A key issue in both phylodynamics and machine learning applications is scalability. Our results show that very large phylogenies can be analysed very efficiently (~1 minute for 10,000 tips), with resulting estimates more accurate than with smaller trees (**Fig. 4**), as predicted by learning theory. Again, as expected, more complex models require more training instances, especially BDSS using CBLV (**Supplementary Fig. 3**), but the ratio remains reasonable, and it is likely that complex (but identifiable) models will be handled efficiently with manageable training sets. Surprisingly, we did not observe a substantial drop of accuracy with lower sampling probabilities. To analyse very large trees, we used a decomposition into smaller, disjoint subtrees. In fact, all our NNs were trained with trees of moderate size (<500 tips). Another approach would be to learn directly from large trees. This is an interesting direction for further research, but this poses several difficulties. The first is that we need to simulate these very large trees, and a large number of them (millions or more). Then, SS is the easiest representation to learn, but at the risk of losing essential information, which means that new summary statistics will likely be needed for sufficiently complete representation of very large phylogenies. Similarly, with CBLV more complex NN architectures (*e.g.*, with additional and larger kernels in the convolutional layers) will likely be needed, imposing larger training sets. Combining both representations (*e.g.*, during the FFNN phase) is certainly an interesting direction for further research. Note, however, that the predictions of both approaches for the three models we studied are highly correlated (Pearson coefficient nearly equal to 1 for most parameters), which means that there is likely little room for improvement (at least with these models).

A key advantage of the deep learning approaches is that they yield close to immediate estimates and apply to trees of varying size. Collection of pathogen genetic data became standard in many countries, resulting in densely sampled infected populations. Examples of such datasets include HIV in Switzerland and UK[24,42], 2013 Ebola epidemics[6], several Influenza epidemics and the 2019 SARS-Cov-2 pandemic (www.gisaid.org)[43]. For many such pathogens, trees can be efficiently and accurately inferred[44-46] and dated[47-49] using standard approaches. When applied to such dated trees, our methods can perform model selection and provide accurate phylodynamic parameter estimates within a fraction of a second. Such properties are desirable for phylogeny-based real-time outbreak surveillance methods, which must be able to cope with the daily influx of new samples, and thus increasing size of phylogenies, as the epidemic unfolds, in order to study local outbreaks and clusters, and assess and compare the efficiency of healthcare policies deployed in parallel. Moreover, thanks to the subtree picking and averaging strategy, it is now possible to analyse very large phylogenies, and the approach could be used to track the evolution of parameters (*e.g.*, R_0) in different regions (sub-trees) of a global tree, as a function of dates (as in Bayesian skyline models[4]), geographical areas, viral variants etc.

METHODS

Here we describe the main methodological steps. For algorithms, technical details, software programs used and their options, and additional comments, an extended version is available in **Supplementary Information**.

Tree representation using summary statistics (SS)

We use 98 summary statistics (SS), to which we add the sampling probability, summing to a vector of 99 values. We use the 83 SS proposed by Saulnier *et al.*[19]:

- 8 SS on tree topology
- 26 SS on branch lengths
- 9 SS on Lineage-Through-Time (LTT) plot
- 40 SS providing the coordinates of the LTT plot

In addition, we designed 14 SS on transmission chains to capture information on the superspreading population. A superspreading individual transmits to more individuals within a given time period than a normal spreader. We thus expect that with superspreading individuals we would have shorter transmission chains. To have a proxy for the

transmission chain length, we look at the sum of 4 subsequent shortest times of transmission for each internal node. This gives us a distribution of time-durations of 4-transmission chains. We assume that information on the transmission dynamics of superspreading individuals is retained in the lower (*i.e.*, left) tail of this distribution, which contains relatively many transmissions with short time to next transmission, while the information on normal spreaders should be present in the rest of the distribution. From the observed distribution of 4-transmission-chain lengths, we compute 14 statistics:

- number of 4-transmission chains in the tree
- 9 deciles of 4-transmission-chain lengths distribution
- minimum and maximum values of 4-transmission-chain lengths
- mean value of 4-transmission-chain lengths
- variance of 4-transmission-chain lengths

Moreover, we provide the number of tips in the input tree, resulting in 83+14+1 = 98 SS in total.

Complete and compact tree representation (CBLV)

The representation of a tree with n tips is a vector of length $2n$-1, where one single real-valued scalar corresponds to one internal node or tip. This representation thus scales linearly with the tree size. The encoding is achieved in two steps: tree ladderization and tree traversal.

The tree ladderization consists in ordering the children of each node. Child nodes are sorted based on the sampling time of the most recently sampled tip in their subtrees: for each node, the branch supporting the most recently sampled subtree is rotated to the left, as in **Fig. 2 a (i-ii)**.

Once the tree is sorted, we perform an inorder tree traversal[30]. When visiting a tip, we add its distance to the previously visited internal node or its distance to the root, for the tip that is visited first (*i.e.*, the tree height due to ladderization). When visiting an internal node, we add its distance to the root. Examples of encoding are shown in **Fig. 2 a (ii-iii)**. This gives us the Compact Bijective Ladderized Vector (CBLV). We then separate information relative to tips and to internal nodes into two rows (**Fig. 2 a (iv)**) and complete the representation with zeros until reaching the size of the largest simulated tree for the given simulation set (**Fig. 2 a (v)**).

CBLV has favourable features for deep learning. Ladderization does not actually change the input tree, but by ordering the subtrees it standardizes the input data and facilitates the learning phase, as observed with random subtree order (**Supplementary Fig. 5,** Compact Random Vector (CRV) representation,). The inorder tree traversal procedure is a bijective transformation, as it transforms a tree into a tree-compatible vector, from which the (ordered) tree can be reconstructed unambiguously, using a simple path-agglomeration algorithm shown in **Supplementary Fig. 1**. CBLV is "as concise as possible". A rooted tree has $2n$-2 branches, and thus $2n$-2 entries are needed to represent the branch lengths. In our $2n$-1 vectorial encoding of trees, we not only represent the branch lengths, but also the tree topology using only 1 additional entry.

Tree rescaling

Before encoding, the trees are rescaled so that the average branch length is 1, that is, each branch length is divided by the average branch length of the given tree, called rescale factor. The values of the corresponding time-dependent parameters (*i.e.*, infectious period and incubation period) are divided by the rescale factor too. The NN is then trained to predict these rescaled values. After parameter prediction, the predicted parameter values are multiplied by the rescale factor and thus rescaled back to the original time scale. Rescaling thus makes a pre-trained NN more generally applicable, for example both to phylogenies of pathogen associated with an infectious period on the scale of days (e.g., Ebola virus) and years (e.g., HIV).

Reduction and centering of summary statistics

Before training our NN and after having rescaled the trees to unit average branch length (see above), we reduce and center every summary statistic by subtracting the mean and scaling to unit variance. To achieve this, we use the standard scaler from the scikit-learn package [50], which is fitted to the training set. This does not apply to CBLV representation, to avoid losing the ability to reconstruct the tree.

Parameter and model inference using neural networks

We implemented deep learning methods in Python 3.6 using Tensorflow 1.5.0[51], Keras 2.2.4[52] and scikit-learn 0.19.1[50] libraries. For each network, several variants in terms of number of layers and neurons, activation functions, regularization, loss functions and optimizer, were tested. In the end, we decided for two specific architectures that best fit our purpose: one deep FFNN trained on SS and one CNN trained on CBLV tree representation.

The FFNN for SS consists of one input layer with 99 input nodes (98 SS + the sampling probability), 4 sequential hidden layers organized in a funnel shape with 64-32-16-8 neurons and 1 output layer of size 2-4 depending on the number of parameters to be estimated. The neurons of the last hidden layer have linear activation, while others have exponential linear activation[53].

The CNN for CBLV consists of one input layer (of 400 and 1002 input nodes for trees with 50-199 and 200-500 tips, respectively). This input is then reshaped into a matrix of size of 201×2 and 501×2, for small and large trees, respectively, with entries corresponding to tips and internal nodes separated into two different rows (and one extra column with one entry in each row corresponding to the sampling probability). Then, there are two 1D convolutional layers of 50 kernels each, of size 3 and 10, respectively, followed by max pooling of size 10 and another 1D convolutional layer of 80 kernels of size 10. After the last convolutional layer, there is a GlobalPoolingAverage1D layer and a FFNN of funnel shape (64-32-16-8 neurons) with the same architecture and setting as the NN used with SS.

For both NNs, we use the Adam optimisation algorithm[54] as optimizer and the Mean Absolute Percentage Error (MAPE) as loss function. The batch size is set to 8,000. To train the network, we split the simulated dataset into 2 groups: (1) proper training set (3,990,000 examples); (2) validation set (10,000). To prevent overfitting during training, we use: (1) the early stopping algorithm evaluating MAPE on the validation set; and (2) dropout that we set to 0.5 in the feedforward part of both NNs[55] (0.4, 0.45, 0.55 and 0.6 values were tried for basic BD model without improving the accuracy).

For model selection, we use the same architecture for FFNN-SS and CNN-CBLV as those for parameter inference described above**.** The only differences are: (1) the cost function: categorical cross entropy, and (2) the activation function used for the output layer, that is, softmax function (of size 2 for small trees, selecting between BD and BDEI model, and of size 3 for large trees, selecting between BD, BDEI and BDSS). As we use the softmax function, the outputs of prediction are the estimated probabilities of each model, summing to 1.

Parameter estimation from very large trees using subtree picking and averaging

To estimate parameters from very large trees we designed the 'Subtree Picker' algorithm (see Supplementary Information for details). The goal of Subtree Picker is to extract subtrees of bounded size representing independent sub-epidemics within the global epidemic represented by the initial huge tree T, while covering most of the initial tree

branches and tips in T. The sub-epidemics should follow the same sampling scheme as the global epidemic. This means that we can stop the sampling earlier than the most recent tip in T, but we cannot omit tips sampled before the end the sampling period (this would correspond to lower sampling probability). Each picked subtree corresponds to a sub-epidemic that starts with its root individual and lasts between its root date D_{root} and some later date ($D_{last} > D_{root}$). The picked subtree corresponds to the top part of the initial tree's clade with the same root, while the tips sampled after D_{last} are pruned. The picked subtrees do not intersect with each other and cover most of the initial tree's branches: 98.5% (BD), 97.3% (BDEI) and 82.4% (BDSS) of the initial tree branches on the 'huge' tree datasets (5,000 to 10,000 tips). For the BDSS model, this percentage is lower than for BD and BDEI, because of the narrower subtree size interval (200-to-500 tips versus 50-to-500 tips) corresponding to current PhyloDeep training set settings. Subtree Picker performs a postorder tree traversal (tips-to-root) and requires $O(n^2)$ computing time in the worst case, where n is the number of tips in T. In practice, Subtree Picker takes on average 0.6 (BD), 0.8 (BDEI) and 0.8 (BDSS) seconds per 'huge' tree (5,000-to-10,000 tips), meaning that it could easily be applied to much larger trees. Once subtrees (sub-epidemics) have been extracted, they are analysed using CNN-CBLV or FFNN-SS, and the parameter estimates are averaged with weights proportional to subtree sizes.

Confidence intervals

For all NN-based parameter estimates, we compute 95% CI using a form of parametric bootstrap. To facilitate the deployment and speed-up the computation, we perform an approximation using a separate set of 1,000,000 simulations. For each simulation in the CI set, we store the true parameter values and the parameter values predicted with both of our methods. This large dataset of true/predicted values is used to avoid new simulations, as required with the standard parametric bootstrap. For a given simulated or empirical tree T, we obtain a set of predicted parameter values, $\{p\}$. The CI computation procedure searches among stored data those that are closest to T in terms of tree size, sampling probability and predicted values. We first subset:

- 10% of simulations within the CI set, which are closest to T in terms of size (number of tips).
- Amongst these, 10% of simulations that are closest to T in terms of sampling probability.

We thus obtain 10,000 CI sets of real/predicted parameter values, similar in size and sampling probability to T. For each parameter value p predicted from T, we identify the 1,000 nearest neighbouring values amongst the 10,000 true

values of the same parameter available in the CI sets, $R_{CI} = \{r_{i=1,1000}\}$, and keep the corresponding predicted values, $P_{CI} = \{p_{i=1,1000}\}$. We then measure the errors for these neighbours as $E_{CI} = \{e_i = p_i - r_i\}$. We center these errors around p using the median of errors, $m(E_{CI})$, which yields the distribution of errors for given prediction p: $D = \{p + e_i - m(E_{CI})\}$, from which we extract the 95% CI around p. Individual points in the obtained distribution that are outside of the parameter ranges covered through simulations are set to the closest boundary value of the parameter range. With very large trees and the subtree picking and averaging procedure, we use a quadratic weighted average of the individual CIs found for every subtree. To assess this fast implementation of the parametric bootstrap, we used the coverage of the true parameter values (expected to be of 95%) and the width (the lower the better) of the CIs. Results and comparisons with BEAST2 are reported in **Supplementary Tab. 7**.

Models

The models we used for tree simulations are represented in the form of flow diagrams in **Fig. 1**. We simulated dated binary trees for (1) the training of NNs and (2) accuracy assessment of parameter estimation and model selection. We used the following three individual-based phylodynamic models:

- Constant rate birth-death model with incomplete sampling: this model (BD[8,9], **Fig. 1 a**) contains three parameters and three compartments: infectious (I), removed with sampling (R) and removed unsampled (U) individuals. Infection takes place at rate β. Infectious individuals are removed with rate γ. Upon removal, an individual is sampled with probability *s*. For simulations, we re-parameterized the model in terms of: basic reproduction number, R_0; infectious period, $1/\gamma$; sampling probability, *s*; and tree size, *t*. We then sampled the values for each simulation uniformly at random in the ranges given in **Supplementary Tab. 4**.
- Birth-death model with exposed-infectious classes: This model (BDEI[10-12], **Fig. 1 b**) is a BD model extended through the presence of an exposed class. More specifically, this means that each infected individual starts as non-infectious (E) and becomes infectious (I) at incubation rate ε. BDEI model thus has four parameters (β, γ, ε and *s*) and four compartments (E, I, R and U). For simulations, we re-parameterized the model similarly as described for BD and set the ε value via the incubation ratio (= ε/γ). We sampled all parameters, including ε/γ, from a uniform distribution, just as with BD (**Supplementary Tab. 4**).

- Birth-death model with superspreading: This model (BDSS[5,10,11], **Fig. 1 c**) accounts for heterogeneous infectious classes. Infected individuals belong to one of two infectious classes (I_S for superspreading and I_N for normal spreading) and can transmit the disease by giving birth to individuals of either class, with rates $\beta_{S,S}$ and $\beta_{S,N}$ for I_S transmitting to I_S and to I_N, respectively, and $\beta_{N,S}$ and $\beta_{N,N}$ for I_N transmitting to I_S and I_N, respectively. However, there is a restriction on parameter values: $\beta_{S,S} \times \beta_{N,N} = \beta_{S,N} \times \beta_{N,S}$. There are thus superspreading transmission rates $\beta_{S,.}$ and normal transmission rates $\beta_{N,.}$ that are $X_{SS} = \beta_{S,S}/\beta_{N,S} = \beta_{S,N}/\beta_{N,N}$ times higher for superspreading. At transmission, the probability of the recipient to be superspreading is $f_{ss} = \beta_{S,S}/\left(\beta_{S,S}+\beta_{S,N}\right)$, the fraction of superspreading individuals at equilibrium. We consider that both I_S and I_N populations are otherwise indistinguishable, that is, both populations share the same infectious period ($1/\gamma$)[5,10,11]. The model thus has six parameters, but only five need to be estimated to fully define the model[5,10]. For simulations, we chose parameters of epidemiological interest for re-parameterization: basic reproduction number R_0, infectious period $1/\gamma$, f_{SS}, X_{ss} and sampling probability *s*. In our simulations, we used uniform distributions for these five parameters, just as with BD and BDEI (**Supplementary Tab. 4**).

Parameter inference with BEAST2

To assess the accuracy of our methods, we compared it with a well-established Bayesian method, as implemented in BEAST2 (version 2.6.2). We used the BDSKY package[4] (version 1.4.5) to estimate the parameter values of BD simulations and the package bdmm[12,13] (version 1.0) to infer the parameter values of BDEI and BDSS. Furthermore, for the inference on BDSS simulations, instead of BEAST 2.6.2 we used the BEAST2 code up to the commit nr2311ba7, which includes important fixes to operators critical for our analyses. We set the Markov Chain Monte Carlo (MCMC) length to 5 million steps for the BD model, and to 10 million steps for the BDEI and BDSS models.

The sampling probability was fixed during the estimation. Since the BD, BDEI and BDSS models implemented in BEAST2 do not use the same parametrizations as our methods, we needed to apply parameter conversions for setting the priors for BEAST2 inference (**Supplementary Tab. 6**), and for translating the BEAST2 results back to parameterizations used in our methods, in order to enable proper comparison of the results (see **Supplementary Information** for details).

After we obtained the parameters of interest from the original parameters estimated by BEAST2, we evaluated the Effective Sample Size (ESS) on all parameters. We reported the absolute percentage error of the median of *a posteriori* values (more stable and accurate than the maximum *a posteriori*), corresponding to all reported steps (spaced by 1,000 actual MCMC steps) past the 10% burn-in. For simulations for which BEAST2 did not converge, we considered the median of the parameter distribution used for simulations (**Fig. 3, Supplementary Tab. 1-2, Supplementary Fig. 2**) or excluded them from the comparison (**Supplementary Tab. 1-2**, values reported in brackets, **Supplementary Tab. 5**).

For the HIV application, the prior of infectious period was set to [0.1, 30] years (uniform). All the other parameters had the same prior distributions as used in simulations and shown in **Supplementary Tab. 4, 6.**

Accuracy of parameter estimation

To compare the accuracy of the different methods, we used 100 simulated trees per model. For each simulated tree, we computed the relative error and its mean over the 100 trees (**Fig. 3-4, Supplementary Tab. 1, Supplementary Fig. 2-4**):

$$MRE = \frac{1}{100}\sum_{i=1}^{100}\frac{|predicted_i - target_i|}{target_i}.$$

The mean relative bias (**Supplementary Tab. 2**) was measured in a similar manner as:

$$MRB = \frac{1}{100}\sum_{i=1}^{100}\frac{(predicted_i - target_i)}{target_i}.$$

Comparison of time efficiency

For FFNN-SS and CNN-CBLV, we reported the average CPU time of encoding a tree (average over 10,000 trees), as reported by NextFlow[56]. The inference time itself was negligible.

For BEAST2, we reported the CPU time averaged over 100 analyses with BEAST2 as reported by NextFlow. For the analyses with BDEI and BDSS models, we reported the CPU time to process 10 million MCMC steps, and for the analyses with BD, we reported the CPU time to process 5 million MCMC steps. To account for convergence, we re-calculated the average CPU time considering only those analyses for which the chain converged and an ESS of 200 was reached for all inferred parameters.

The calculations were performed on a computational cluster with CentOS machines and Slurm workload manager. The machines had the following characteristics: 28 cores, 2.4 GHz, 128 GB of RAM. Each of our jobs (simulation of one tree, tree encoding, BEAST2 run, etc.) was performed requesting one CPU core. The neural network training was performed on a GPU cluster with Nvidia Titan X GPUs.

HIV dataset

We used the original phylogenetic tree reconstructed by Rasmussen *et al.*[25] from 200 sequences corresponding to the largest cluster of HIV-infected men-having-sex-with-men (MSM) subpopulation in Zurich, collected as a part of the Swiss Cohort Study[24]. For details on tree reconstruction, please refer to their article.

PhyloDeep software

FFNN-SS and CNN-CBLV parameter inference, model selection, 95% CI computation and *a priori* checks are implemented in the PhyloDeep software, which is available on GitHub (github.com/evolbioinfo/phylodeep), PyPi (pypi.org/project/phylodeep) and Docker Hub (hub.docker.com/r/evolbioinfo/phylodeep). It can be run as a command-line program, Python3 package and a Docker container. PhyloDeep covers the parameter subspace as described in **Supplementary Tab. 4**. The input is a dated phylogenetic tree with at least 50 tips and presumed sampling probability. The output is a PCA plot for *a priori* check, a csv file with all SS, and a csv file with probabilities of each model (for model selection) and point estimates and 95% CI values (for parameter inference with selected model). The installation details and usage examples are available as well on GitHub.

DATA AVAILABILITY

The data generated in this study (simulated trees, BEAST2 logs, and the results of BEAST2 and PhyloDeep runs), as well as the HIV phylogenetic tree for Zurich epidemic (a showcase application) are provided on GitHub (github.com/evolbioinfo/phylodeep, version 0.3) and have been deposited in the Zenodo database under accession code https://doi.org/10.5281/zenodo.6646668. The simulated trees were obtained with our simulator (see Code availability), the HIV tree was previously published by Rasmussen et al[25] and is available on their GitHub: github.com/davidrasm/PairTree (all confidential information has been removed).

CODE AVAILABILITY

The PhyloDeep package (version 0.3) is under the GPL v3.0 license and uses Python (3.6) and Python libraries : ete3 (version 3.1.2 under GNU general licence); pandas (version 1.1.5); numpy (version 1.19.5); scipy (version 1.1.0); scikit-learn (version 0.19.1); tensorflow (version 1.13.1); joblib (version 0.13.2); h5py (version 2.10.0); Keras (version 2.4.3 under Apache 2.0 license); matplotlib (version 3.1.3 under PSF license).

We provide (i) the source code of PhyloDeep, (ii) the code of the tree simulators used to train the deep learners and (iii) the log files obtained with BEAST2 on GitHub (github.com/evolbioinfo/phylodeep). The code has been deposited in Zenodo[57].

We used the version 2.6.2 of BEAST2 for BD and BDEI inferences and BEAST2 compiled up to the commit nr2311ba7 for BDSS inferences, and version 3.3 of TreePar. For BEAST2 inferences, we used BEAST2 libraries bdmm (version 1.0) and BDSKY (version 1.4.5).

We used Snakemake (version 5.10.0) and Nextflow (version 21.04.3.5560) pipeline managers for simulations and analyses of simulated data.

REFERENCES

1. Grenfell, B.T. *et al.* Unifying the epidemiological and evolutionary dynamics of pathogens. *Science* **303,** 327-332 (2004).
2. Volz, E.M., Kosakovsky Pond, S.L., Ward, M.J., Leigh Brown, A.J., Frost, S.D. Phylodynamics of infectious disease epidemics. *Genetics* **183,** 1421-30 (2009).
3. Drummond, A.J., Rambaut, A., Shapiro, B., Pybus, O.G. Bayesian Coalescent Inference of Past Population Dynamics from Molecular Sequences. *Molecular Biology and Evolution*, **22,** 1185–1192 (2005).
4. Stadler, T. Birth–death skyline plot reveals temporal changes of epidemic spread in HIV and hepatitis C virus (HCV). *Proceedings of the National Academy of Sciences* **110**, 228-233 (2013)
5. Stadler, T., Bonhoeffer, S. Uncovering epidemiological dynamics in heterogeneous host populations using phylogenetic methods. *Philosophical Transactions of the Royal Society B: Biological Sciences* **368(1614)**, 20120198 (2013).

6. Gire, S.K. et al Genomic surveillance elucidates Ebola virus origin and transmission during the 2014 outbreak. *Science* **345**, 1369-72 (2014).
7. Boskova, V., Bonhoeffer, S., Stadler, T. Inference of Epidemiological Dynamics Based on Simulated Phylogenies Using Birth-Death and Coalescent Models. *PLOS Computational Biology* **10(11),** e1003913 (2014).
8. Stadler, T. *et al.* Estimating the Basic Reproductive Number from Viral Sequence Data. *Mol. Biol. Evol.* **29,** 347–57 (2012).
9. Leventhal, G.E., Günthard, H.F., Bonhoeffer, S., Stadler, T. Using an Epidemiological Model for Phylogenetic Inference Reveals Density Dependence in HIV Transmission. *Mol. Biol. Evol.* **31,** 6–17 (2014).
10. Stadler, T., Kuhnert, D., Rasmussen, D.A., du Plessis, L. Insights into the early epidemic spread of Ebola in sierra leone provided by viral sequence data. *PLoS Curr.* **6,** (2014).
11. Kühnert, D., Stadler, T., Vaughan, T.G., Drummond, A.J. Phylodynamics with Migration: A Computational Framework to Quantify Population Structure from Genomic Data. *Mol. Biol. Evol.* **33,** 2102-16 (2016).
12. Sciré, J., Barido-Sottani, J., Kühnert, D., Vaughan, T.G., Stadler, T. Improved multi-type birth-death phylodynamic inference in BEAST 2 (2020). Preprint at https://www.biorxiv.org/content/10.1101/2020.01.06.895532v1.full.pdf
13. Bouckaert, R. *et al.* BEAST 2: A Software Platform for Bayesian Evolutionary Analysis. *PLoS Computational Biology* **10(4)**, e1003537 (2014).
14. Boskova, V., Stadler, T., Magnus, C. The influence of phylodynamic model specifications on parameter estimates of the Zika virus epidemic. *Virus Evolution* **4(1),** vex044 (2018).
15. Vaughan, T.G., Sciré, J., Nadeau, S.A., Stadler, T. Estimates of outbreak-specific SARS-CoV-2 epidemiological parameters from genomic data (2020). Preprint at https://www.medrxiv.org/content/10.1101/2020.09.12.20193284v1.full.pdf
16. Rubin, D.B. Bayesianly Justifiable and Relevant Frequency Calculations for the Applies Statistician. *The Annals of Statistics* **12,** 1151-72 (1984).
17. Beaumont, M.A., Zhang, W., Balding, D.J. Approximate Bayesian Computation in Population Genetics. *Genetics* **164,** 2025-35 (2002).

18. Csilléry, K., Blum, M.G.B., Gaggiotti, O.E., François, O. Approximate Bayesian Computation (ABC) in practice. *Trends in Ecology & Evolution* **25,** 410-8 (2010).
19. Saulnier, E., Gascuel, O., Alizon, S. Inferring epidemiological parameters from phylogenies using regression-ABC: A comparative study. *PLoS Comp. Biol.* **13(3)**, e1005416 (2017).
20. Blum, M.G.B. *Handbook Of Approximate Bayesian Computation Ch. Regression approaches for ABC.* 71–85. (Chapman and Hall/CRC Press, Boca Raton, 2018).
21. LeCun, Y., Kavukcuoglu, K., Farabet, F. Convolutional networks and applications in vision. *Proc. IEEE Int. Symp. Circuits Syst.* 253-6 (2010).
22. Krizhevsky, K., Sutskever, I., Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. *Advances in neural information processing systems* 1097-105 (2012).
23. Chattopadhyay, A., Hassanzadeh, P., Pasha, S. Predicting clustered weather patterns: A test case for applications of convolutional neural networks to spatio-temporal climate data. *Sci. Rep.* **10,** 1317 (2020)
24. The Swiss HIV Cohort Study *et al.* Cohort Profile: The Swiss HIV Cohort Study. *International Journal of Epidemiology* **39,** 1179–89 (2010).
25. Rasmussen, D.A., Kouyos, R., Günthard, H.F., Stadler, T. Phylodynamics on local sexual contact networks. *PLOS* Comp. Biol. **13(3)**, e1005448 (2017).
26. Colijn, C. & Plazzotta, G. A metric on phylogenetic tree shapes. *Systematic Biology* **67,** 113–26 (2018).
27. Liu, P., Gould, M., Colijn, C. Analyzing Phylogenetic Trees with a Tree Lattice Coordinate System and a Graph Polynomial, *Systematic Biology*, in press (2022). Preprint at https://doi.org/10.1093/sysbio/syac008
28. Lewitus, E. & Morlon, H. Characterizing and Comparing Phylogenies from their Laplacian Spectrum. *Systematic Biology* **65,** 495-507 (2016).
29. Kim, J., Rosenberg, N.A., Palacios, J.A. Distance metrics for ranked evolutionary trees. *Proceedings of the National Academy of Sciences* **117,** 28876-86 (2020).
30. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C. *Introduction To Algorithms*. 286-307 (The MIT Press, Cambridge, 2009).
31. Bengio, Y. *Neural Networks: Tricks Of The Trade, Ch. Practical Recommendations for Gradient-Based Training of Deep Architectures.* (Springer, Berlin, Heidelberg 2002).

32. Gelman, A., Carlin, J.B., Stern, H.S., Rubin, D.B. *Bayesian Data Analysis: Second Edition.* (Chapman and Hall/CRC Press, Boca Raton, 2004).
33. Baele, G. *et al.* Improving the accuracy of demographic and molecular clock model comparison while accommodating phylogenetic uncertainty. *Mol. Biol. Evol.* **29,** 2157-67 (2012).
34. Kouyos, R.D. *et al.* Molecular epidemiology reveals long-term changes in HIV type 1 subtype B transmission in Switzerland. *J. Infect. Dis*. **201,** 1488-97 (2010).
35. May, R.M. & Anderson, R.M. Transmission dynamics of HIV infection. *Nature* **326,** 137–142 (1987).
36. Brenner, B.G. *et al.* Quebec Primary HIV Infection Study Group. High rates of forward transmission events after acute/early HIV-1 infection. *J. Infect. Dis.* **195,** 951-9 (2007).
37. Gueler, A. *et al.* Swiss National Cohort Life expectancy in HIV-positive persons in Switzerland. *AIDS* **31,** 427-436 (2017).
38. Rasmussen, D.A., Volz, E.M., Koelle, K. Phylodynamic Inference for Structured Epidemiological Models. *PLoS Comput. Biol*. **10(4)**, e1003570 (2014).
39. Volz, E.M. & Siveroni, I. Bayesian phylodynamic inference with complex models. *PLoS Comput. Biol.* **14(11)**, e1006546 (2018).
40. MacPherson, A., Louca, S., McLaughlin, A., Joy, J.B., Pennell, M.W. Unifying Phylogenetic Birth–Death Models in Epidemiology and Macroevolution, *Systematic Biology*, **71(1)**, 172–189 (2022).
41. Sanchez, T., Cury, J., Charpiat, G., Jay, F. Deep learning for population size history inference: Design, comparison and combination with approximate Bayesian computation. *Mol. Ecol. Resour.* **00,** 1-16. (2020).
42. Dunn, D. & Pillay, D. UK HIV drug resistance database: background and recent outputs. *J. HIV Ther.* **12,** 97–8 (2007).
43. Shu, Y. & McCauley, J. GISAID: Global initiative on sharing all influenza data - from vision to reality. *Euro Surveill.* **22,** 30494 (2017).
44. Minh, B.Q. *et al.* IQ-TREE 2: New models and efficient methods for phylogenetic inference in the genomic era. *Mol. Biol. Evol.* **37,** 1530–1534 (2020).
45. Kozlov, A.M., Darriba, D., Flouri, T., Morel, B., Stamatakis, A. RAxML-NG: a fast, scalable and user-friendly tool for maximum likelihood phylogenetic inference. *Bioinformatics* **35,** 4453–4455 (2019).

46. Guindon, S. *et al.* New Algorithms and Methods to Estimate Maximum-Likelihood Phylogenies: Assessing the Performance of PhyML 3.0. *Systematic Biology* **59,** 307-21 (2010).
47. Sagulenko, P., Puller, V., Neher, R.A. TreeTime: Maximum-likelihood phylodynamic analysis. *Virus Evol.***4(1),** vex042 (2018).
48. To, T.H., Jung, M., Lycett, S., Gascuel, O. Fast Dating Using Least-Squares Criteria and Algorithms. *Syst Biol.* **65,** 82-97 (2016).
49. Volz, E.M. & Frost, S.D.W. Scalable relaxed clock phylogenetic dating. *Virus Evol.* **3(3)**, vex025 (2017).
50. Pedregosa, F. *et al.* Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* **12,** 2825–2830 (2011).
51. Abadi, M. *et al.* TensorFlow: Large-scale machine learning on heterogeneous systems. Preprint at https://arxiv.org/abs/1603.04467 (2015).
52. Chollet, F. K. https://keras.io. (2015).
53. Clevert, D.A., Unterthiner, T., Hochreiter, S. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *ICLR* (2016).
54. Kingma, D.P. & Ba, J. Adam: A Method for Stochastic Optimization. *ICLR* (2015).
55. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* **15,**1929–1958 (2014).
56. Di Tommaso, P. *et al.* Nextflow enables reproducible computational workflows. Nat. Biotechnol. **35,** 316–319 (2017).
57. Voznica, J., Zhukova, A., Boskova, V., Saulnier, E., Lemoine, F., Moslonka-Lefebvre M., Gascuel, O. Source code of "Deep learning from phylogenies to uncover the epidemiological dynamics of outbreaks" (2022), deposited in the Zenodo database under accession number https://doi.org/10.5281/zenodo.6646668.

ACKNOWLEDGEMENTS

We would like to thank Dr Kary Ocaña and Tristan Dot for initiating experiments on machine learning and phylogenetic trees in our laboratory. We would like to thank Quang Tru Huynh for administrating the GPU farm at Institut Pasteur and the INCEPTION program (Investissement d'Avenir grant ANR-16-CONV-0005) that financed

the GPU farm. We would like to thank Dr Christophe Zimmer from Institut Pasteur, Sophia Lambert and Dr Hélène Morlon from Institut de Biologie de l'Ecole Normale Supérieure IBENS and Dr Guy Baele from Katholieke Universiteit KU Leuven for useful discussions, and Dr Isaac Overcast from IBENS and Luc Blassel from Institut Pasteur for critical reading of the manuscript. We would like to thank Dr Tanja Stadler and Jérémie Sciré for their help with BEAST2 and MLE approaches. JV is supported by Ecole Normale Supérieure Paris-Saclay and by ED Frontières de l'Innovation en Recherche et Education, Programme Bettencourt. VB would like to thank Swiss National Science Foundation for funding (Early PostDoc mobility grant P2EZP3_184543). OG is supported by PRAIRIE (ANR-19-P3IA-0001).

AUTHOR CONTRIBUTIONS

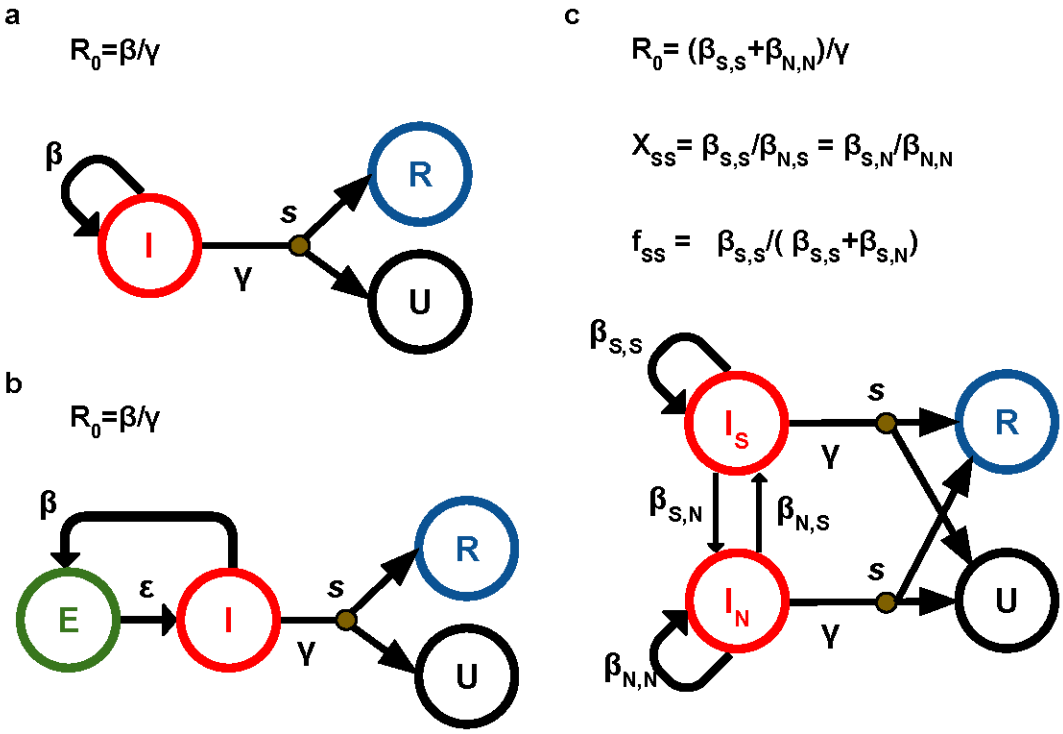
JV, AZ and OG conceived and set up the methods; JV, AZ and VB performed the experiments; JV, AZ and OG analyzed the results; JV and AZ wrote the Python package; JV and OG wrote the manuscript; JV, AZ, VB and OG edited the manuscript; all authors helped in this research, discussed the results and read the final manuscript; OG initiated and supervised the project.

COMPETING INTERESTS

The authors declare no competing interests

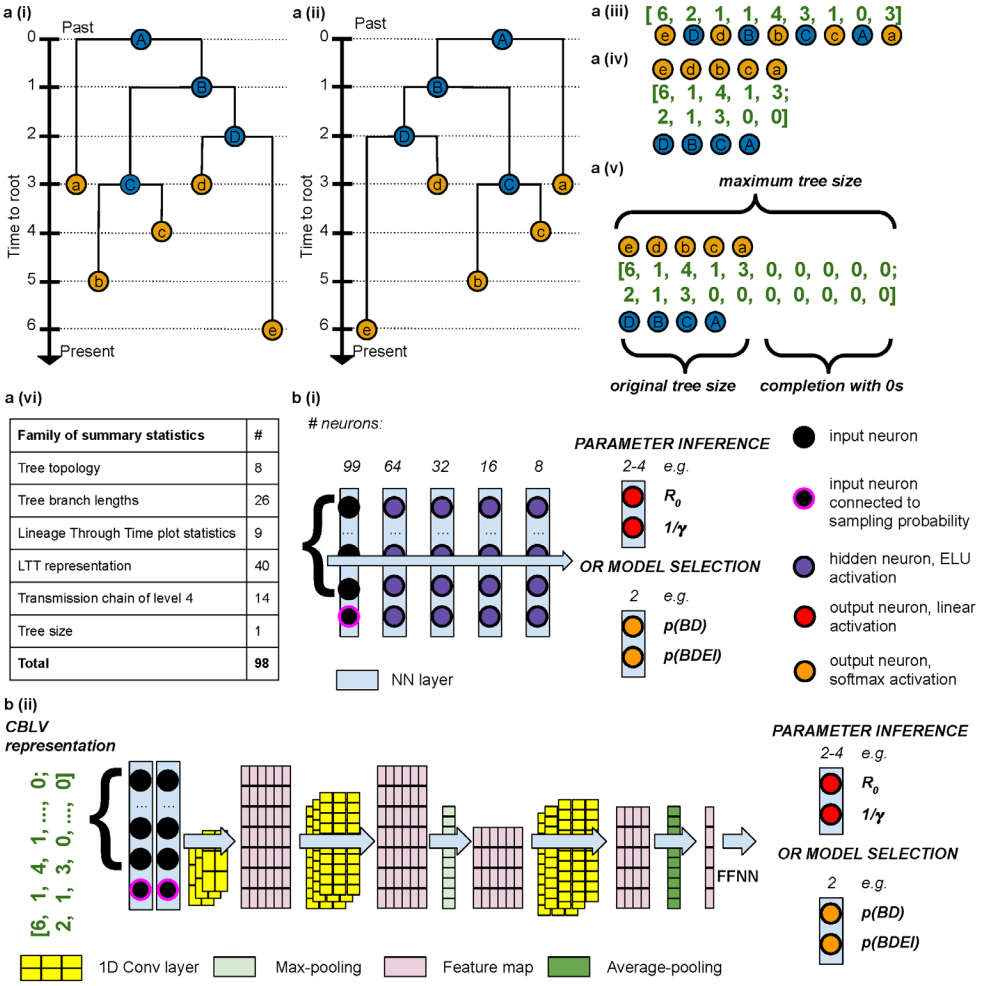
FIGURES & LEGENDS

Fig. 1: Birth-death models



Note to Fig. 1. a Birth-death model (BD)[8,9], **b,** birth-death model with Exposed-Infectious individuals (BDEI)[5,10,11] and **c,** birth-death model with SuperSpreading (BDSS)[5,12]. BD is the simplest generative model, used to estimate R_0 and the infectious period ($1/\gamma$)[8,9]. BDEI and BDSS are extended version of BD. BDEI enables to estimate latency period ($1/\varepsilon$) during which individuals of exposed class E are infected, but not infectious[5,10,11]. BDSS includes two populations with heterogeneous infectiousness: the so-called superspreading individuals (S) and normal spreaders (N). Superspreading individuals are present only at a low fraction in the population (f_{ss}) and may transmit the disease at a rate that is multiple times higher than that of normal spreaders (rate ratio = X_{ss})[5,12]. Superspreading can have various complex causes, such as the heterogeneity of immune response, disease progression, co-infection with other diseases, social contact patterns or risk behaviour, *etc*. Infectious individuals I (superspreading infectious individuals I_S and normal spreaders I_N for BDSS), transmit the disease at rate β ($\beta_{X,Y}$ for an individual of type X transmitting to an individual of type Y for BDSS), giving rise to a newly infected individual. The newly infected individual is either infectious right away in BD and BDSS or goes through an exposed state before becoming infectious at rate ε in BDEI. Infectious individuals are removed at rate γ. Upon removal, they can be sampled with probability *s*, becoming of removed sampled class R. If not sampled upon removal, they move to non-infectious unsampled class U.

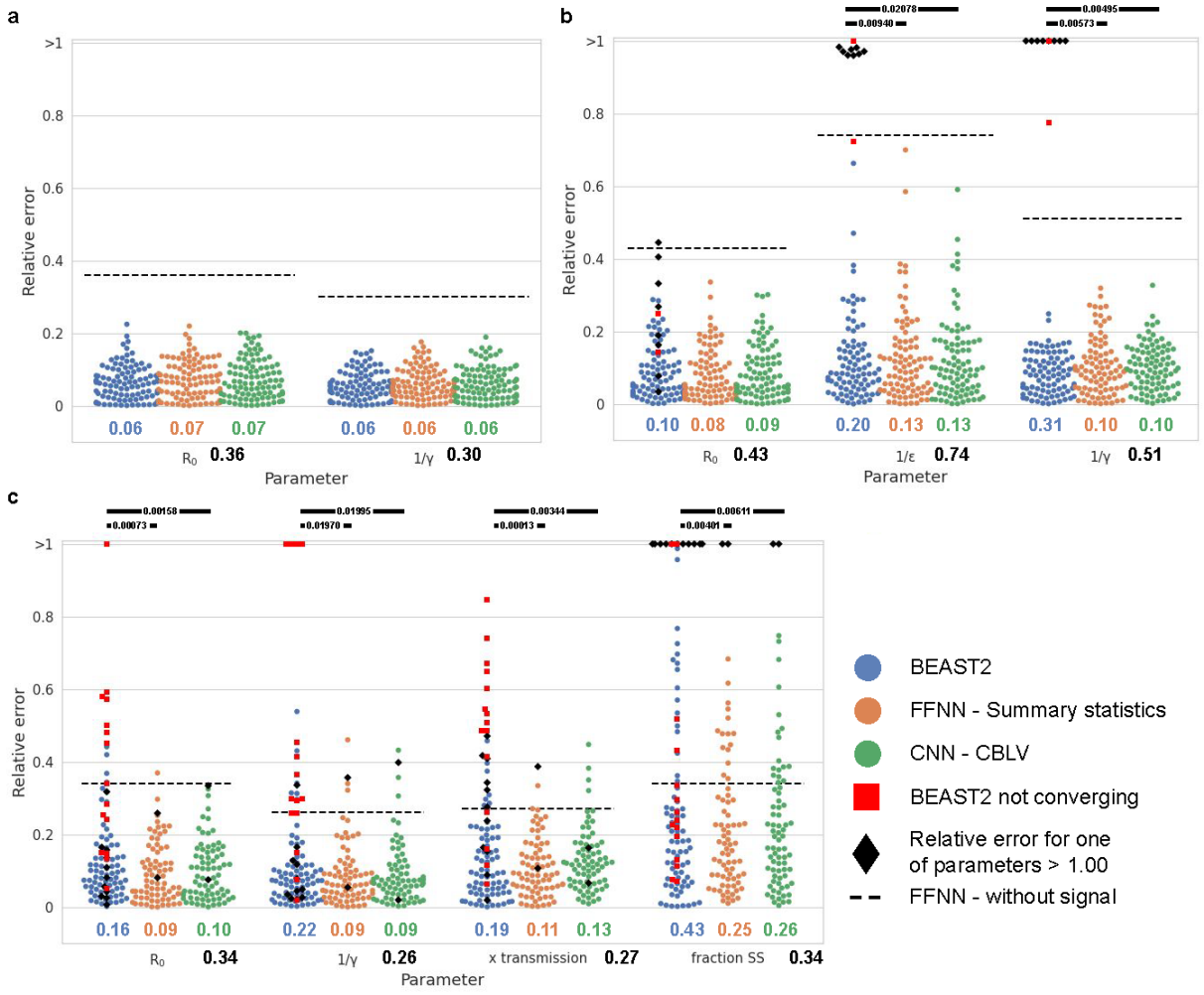
Fig. 2: Pipeline for training neural networks on phylogenies



Family of summary statistics	#
Tree topology	8
Tree branch lengths	26
Lineage Through Time plot statistics	9
LTT representation	40
Transmission chain of level 4	14
Tree size	1
Total	**98**

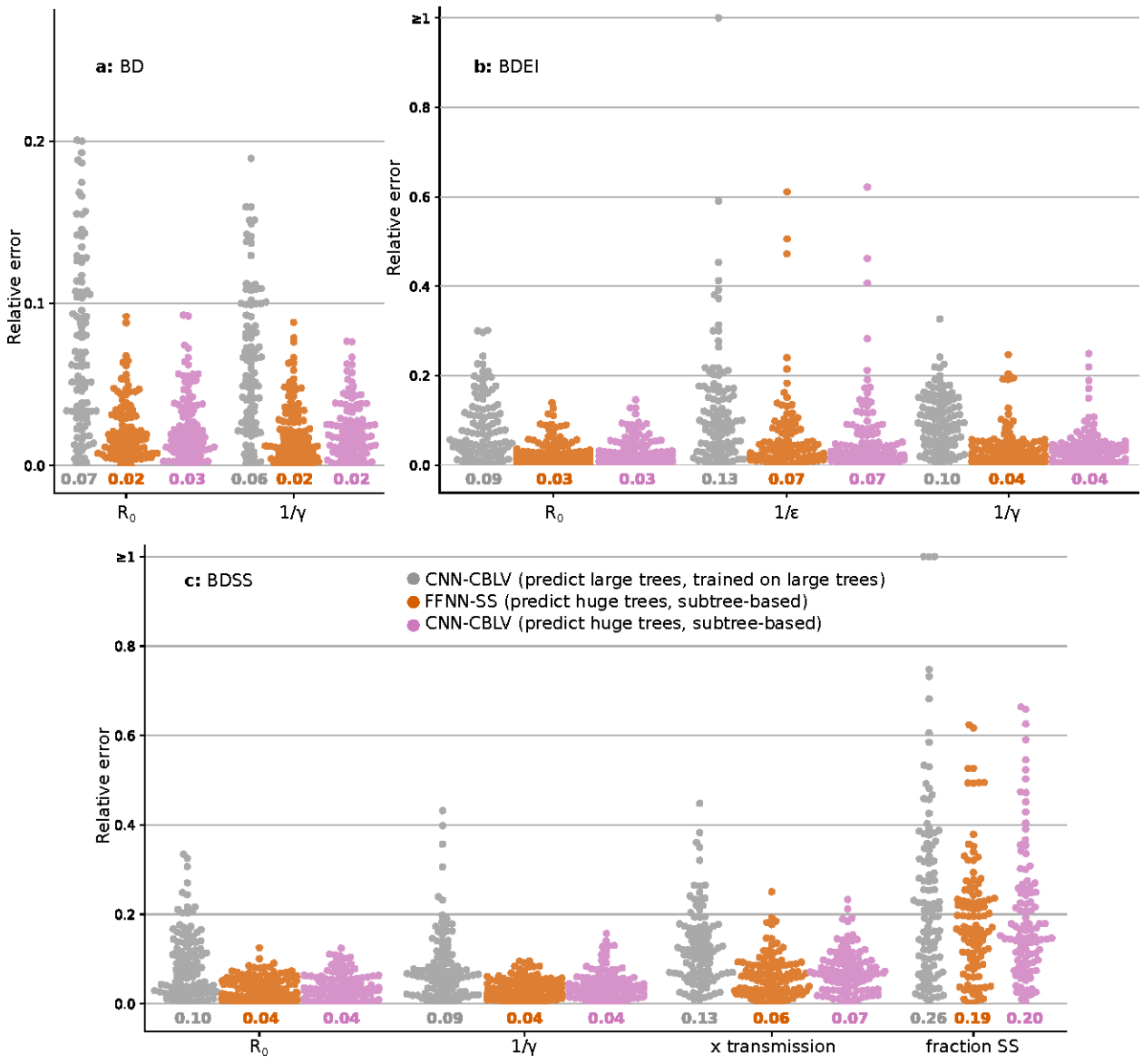
Note to Fig.2. Tree representations: **a (i),** simulated binary trees. Under each model from Fig. 1, we simulate many trees of variable size (50 to 200 tips for 'small trees' and 200 to 500 tips for 'large trees'). For illustration, we have here a tree with 5 tips. We encode the simulations into two representations, either **a (ii-v),** in a complete and compact tree representation called 'Compact Bijective Ladderized Vector' abbreviated as CBLV or a (vi) with summary statistics (SS). CBLV is obtained through **a (ii)** ladderization or sorting of internal nodes so that the branch supporting the most recent leaf is always on the left and **a (iii)** an inorder tree traversal, during which we append to a real-valued vector for each visited internal node its distance to the root and for each visited tip its distance to the previously visited internal node. We reshape this representation into **a (iv)**, an input matrix in which the information on internal nodes and leaves is separated into two rows. Finally, **a (v)**, we complete this matrix with zeros so that the matrices for all simulations have the size of largest simulation matrices. For illustration purpose, we here consider that the maximum tree size covered by simulations is 10, and the representation is thus completed with 0s accordingly. SS consists of **a (vi),** a set of 98 statistics: 83 published in *Saulnier et al*[19], 14 on transmission chains and 1 on tree size. The information on sampling probability is added to both representations. **b: Neural networks** are trained on these representations to estimate parameter values or to select the underlying model. For SS, we use, **b (i)**, a deep feed-forward neural network (FFNN) of funnel shape (we show the number of neurons above each layer). For the CBLV representation we train, **b (ii),** Convolutional Neural Networks (CNN). The CNN is added on top of the FFNN. The CNN combines convolutional, maximum pooling and global average pooling layers, as described in detail in **Methods** and **Supplementary Information**.

Fig. 3: Assessment of deep learning accuracy



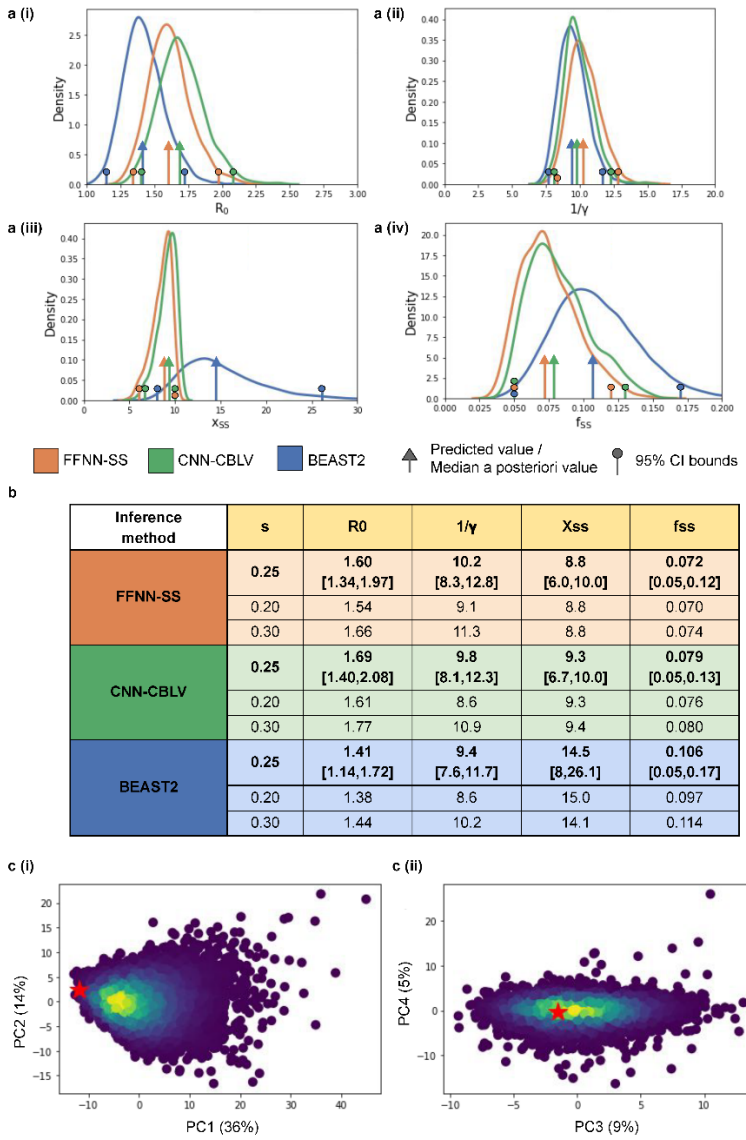
Note to Fig. 3. Comparison of inference accuracy by BEAST2 (in blue), deep neural network trained on SS (in orange) and convolutional neural network trained on the CBLV representation (in green) on 100 test trees. The size of training and testing trees was uniformly sampled between 200 and 500 tips. We show the relative error for each test tree. The error is measured as the normalized distance between the median *a posteriori* estimate by BEAST2 or point estimates by neural networks and the target value for each parameter. We highlight simulations for which BEAST2 did not converge and whose values were thus set to median of the parameter subspace used for simulations by depicting them as red squares. We further highlight the analyses with a high relative error (>1.00) for one of the estimates as black diamonds. We compare the relative errors for **a,** BD-simulated, **b,** BDEI-simulated and **c,** BDSS-simulated trees. Average relative error is displayed for each parameter and method in corresponding colour below each figure. The average error of a FFNN trained on summary statistics but with randomly permuted target is displayed as black dashed line and its value is shown in bold black below the x-axis. The accuracy of each method is compared by two-sided paired z-test; $P < 0.05$ is shown as thick full line; non-significant is not shown.

Fig. 4: Deep learning accuracy with 'huge' trees



Note to Fig. 4. Comparison of inference accuracy by neural networks trained on large trees in predicting large trees (CNN-CBLV, in grey, same as in **Fig. 3**) and huge trees (FFNN-SS, in orange, and CBLV-NN, in pink) on 100 large and 100 huge test trees. The training and testing large trees are the same as in Fig. 3 (between 200 and 500 tips each). The huge testing trees were generated for the same parameters as the large training and testing trees, but their size varied between 5,000 and 10,000 tips. We show the relative error for each test tree. The error is measured as the normalized distance between the point estimates by neural networks and the target values for each parameter. We compare the relative errors for **a**, BD-simulated, **b**, BDEI-simulated and **c**, BDSS-simulated trees. Average relative error is displayed for each parameter and method in corresponding colour below each plot.

Fig. 5: Parameter inference on HIV data sampled from MSM Zurich



Inference method	s	R0	1/γ	Xss	fss
FFNN-SS	**0.25**	**1.60 [1.34,1.97]**	**10.2 [8.3,12.8]**	**8.8 [6.0,10.0]**	**0.072 [0.05,0.12]**
	0.20	1.54	9.1	8.8	0.070
	0.30	1.66	11.3	8.8	0.074
CNN-CBLV	**0.25**	**1.69 [1.40,2.08]**	**9.8 [8.1,12.3]**	**9.3 [6.7,10.0]**	**0.079 [0.05,0.13]**
	0.20	1.61	8.6	9.3	0.076
	0.30	1.77	10.9	9.4	0.080
BEAST2	**0.25**	**1.41 [1.14,1.72]**	**9.4 [7.6,11.7]**	**14.5 [8,26.1]**	**0.106 [0.05,0.17]**
	0.20	1.38	8.6	15.0	0.097
	0.30	1.44	10.2	14.1	0.114

Note to Fig. 5. Using BDSS model with BEAST2 (in blue), FFNN-SS (in orange), and CNN-CBLV (in green) we infer, **a (i),** basic reproduction number, **a (ii),** infectious period (in years), **a (iii),** superspreading transmission ratio and, **a (iv),** superspreading fraction. For FFNN-SS and CNN-CBLV, we show the posterior distributions and the 95% CIs obtained with a fast approximation of the parametric bootstrap (**Methods**, **Supplementary Information**). For BEAST2, the posterior distributions and 95% CI were obtained considering all reported steps (9,000 in total) excluding the 10% burn-in. Arrows show the position of the original point estimates obtained with FFNN-SS and CNN-CBLV and the median *a posteriori* estimate obtained with BEAST2. Circles show lower and upper boundaries of 95% CI. **b,** these values are reported in a table, together with point estimates obtained while considering lower and higher sampling probabilities (0.20 and 0.30). **c,** 95% CI boundaries obtained with FFNN-SS are used to perform an *a posteriori* model adequacy check. We simulated 10,000 trees with BDSS while resampling each parameter from a uniform distribution, whose upper and lower bounds were defined by the 95% CI. We then encoded these trees into SS, performed PCA and projected SS obtained from the HIV MSM phylogeny (red stars) on these PCA plots. We show here the projection into **c (i),** first two components of PCA, **c (ii),** the 3rd and 4th components, together with the associated percentage of variance displayed in parentheses. Warm colours correspond to high density of simulations.

DEEP LEARNING FROM PHYLOGENIES TO UNCOVER THE EPIDEMIOLOGICAL DYNAMICS OF OUTBREAKS

Voznica J*, Zhukova A*, Boskova V, Saulnier E, Lemoine F, Moslonka-Lefebvre M, Gascuel O*

*Correspondence: voznica.jakub@gmail.com, anna.zhukova@pasteur.fr, olivier.gascuel@mnhn.fr

SUPPLEMENTARY TABLES

SUPPLEMENTARY FIGURES

Supplementary Table 1: Method comparison in terms of accuracy

Model	Parameter	Mean Relative Error							
		1. FFNN SS	2. CNN CBLV	3. BEAST2	4. FFNN CBLV	5. LR SS	6. Null model 1	7. Null model 2	z-test (<0.05)
BD 200-500 tips	R_0	**0.07**	**0.07**	**0.06**	**0.08**	0.19	0.36	0.56	1=2=3=4
	$1/\gamma$	**0.06**	**0.06**	**0.06**	**0.06**	0.22	0.30	0.82	1=2=3=4
BDEI 200-500 tips	R_0	**0.08** (0.08)	**0.09** (0.09)	**0.10** (0.10)	**0.10** (0.10)	0.20 (0.20)	0.43 (0.43)	0.56	1=2=3=4
	$1/\varepsilon$	**0.13** (0.13)	**0.13** (0.13)	0.20 (0.19)	0.29 (0.29)	0.21 (0.21)	0.74 (0.74)	2.08	1=2
	$1/\gamma$	**0.10** (0.10)	**0.10** (0.10)	0.31 (0.30)	0.20 (0.21)	0.26 (0.26)	0.51 (0.51)	0.82	1=2
BDSS 200-500 tips	R_0	**0.09** (0.09)	**0.10** (0.10)	0.16 (0.11)	**0.10** (0.10)	0.18 (0.17)	0.34 (0.35)	0.56	1=2=4
	$1/\gamma$	**0.09** (0.09)	**0.09** (0.09)	0.22 (0.10)	**0.10** (0.10)	0.18 (0.16)	0.26 (0.27)	0.84	1=2=4
	X_{SS}	**0.11** (0.11)	**0.13** (0.13)	0.19 (0.14)	0.23 (0.22)	**0.15** (0.14)	0.27 (0.28)	0.41	1=2=5
	f_{SS}	**0.25** (0.23)	**0.26** (0.23)	0.43 (0.44)	0.33 (0.33)	0.36 (0.36)	0.34 (0.34)	0.47	1=2

For each model and inferred parameter, we compared the different inference methods in terms of Mean Relative Error (MRE). We compared these measures for 1. FFNN trained on SS, 2. CNN trained on CBLV, 3. BEAST2, 4. FFNN trained on CBLV, 5. Linear regression trained on SS, 6. "Null model 1", *i.e.* FFNN trained on SS with permuted target values and 7. "Null model 2" measured on 100 targets, where for each target we sampled randomly values from the prior parameter subspace. For 1.-6. MRE values were measured on the same 100 simulations. For BEAST2, we considered median values of covered parameter subspace, when BEAST2 did not converge (2% and 15% of simulations for BDEI and BDSS, respectively). For 1.-6. we displayed in parentheses the MRE when not considering the simulations for which BEAST2 did not converge. We compared the individual methods (1.-6.) in a pairwise fashion to find out if one was more accurate in terms of MRE than the other, using two-sided paired z-test, at significance level of 0.05 and we show the most accurate methods for each parameter in bold.

In all settings, FFNN-SS and CNN-CBLV are amongst the most accurate methods, being exclusively the most accurate ones for BDEI $1/\gamma$ and $1/\varepsilon$ and for BDSS f_{SS}. Furthermore, the comparison of MRE values shows that the prediction of f_{SS} has low accuracy for all methods (0.25 and 0.26 for FFNN-SS and CNN-CBLV, respectively, vs 0.34 obtained with "Null model 1"). This might be due to low information on superspreading individuals in the trees of this size, as supported by Supplementary Fig. 3.

Supplementary Table 2: Method comparison in terms of bias

Model	Parameter	Mean Relative Bias				
		1. FFNN SS	2. CNN CBLV	3. BEAST2	4. FFNN CBLV	5. LR SS
BD 200-500 tips	R_0	-0.01	-0.01	-0.01	-0.01	0.04
	$1/\gamma$	-0.01	-0.02	-0.02	-0.01	0.04
BDEI 200-500 tips	R_0	-0.03 (-0.03)	-0.03 (-0.03)	-0.04 (-0.04)	0.04 (0.04)	-0.03 (-0.03)
	$1/\varepsilon$	-0.01 (-0.01)	-0.01 (-0.01)	**-0.10 (-0.12)**	0.02 (0.03)	0.02 (0.02)
	$1/\gamma$	-0.03 (-0.03)	-0.01 (-0.01)	**0.24 (0.23)**	-0.03 (-0.03)	-0.01 (-0.01)
BDSS 200-500 tips	R_0	-0.01 (-0.01)	-0.01 (-0.01)	**0.07** (0.04)	0.00 (-0.01)	0.02 (0.00)
	$1/\gamma$	0.00 (0.00)	0.00 (-0.01)	**0.12** (0.03)	0.00 (-0.01)	0.03 (0.00)
	X_{ss}	-0.01 (-0.02)	0.01 (0.00)	**0.05** (-0.02)	**0.07** (0.04)	**0.05** (0.03)
	f_{ss}	-0.03 (-0.01)	0.02 (0.03)	**0.32 (0.36)**	-0.02 (-0.01)	**0.19 (0.20)**

For each model and inferred parameter, we compared the different inference methods in terms of Mean Relative Bias (MRB). We compared the MRB for 1. FFNN trained on SS, 2. CNN trained on CBLV, 3. BEAST2, 4. FFNN trained on CBLV and 5. Linear regression trained on SS. These MRB values were measured on the same 100 simulations. If BEAST2 did not converge, we set the inferred value to the median of covered parameter subspace. We further display MRB when not considering simulations for which BEAST2 did not converge, shown in parentheses. The relative biases bigger than 0.05 (5%) are displayed in red bold.

This shows that FFNN-SS and CNN-CBLV have low bias, while BEAST2 suffers from an intermediate to high bias with BDEI and BDSS for most parameters. These biases may account for a large fraction of MRE for BEAST2.

Supplementary Table 3: Method comparison in terms of likelihood with BD model

Y > X	Y: True values	Y: BEAST2	Y: CNN-CBLV	Y: FFNN-SS
X: True values	-	70	66	69
X: BEAST2	30	-	43	53
X: CNN-CBLV	34	57	-	55
X: FFNN-CBLV	31	47	45	-

Median Y-X	Y: True values	Y: BEAST2	Y: CNN-CBLV	Y: FFNN-SS
X: True values	0.0	4.9	4.4	4.5
X: BEAST2	-	0.0	-0.2	0.2
X: CNN-CBLV	-	-	0.0	0.2
X: FFNN-CBLV	-	-	-	0.0

For BD model and 100 large test trees (200-500 tips), we compared parameter estimates obtained with different inference methods in terms of likelihood. We compared loglikelihood values evaluated by TreePar package for: True values with which test trees were simulated; estimates obtained with BEAST2; estimates obtained with CNN-CBLV; estimates obtained with FFNN-SS. The first table shows a simple pairwise comparison (*e.g.*, 30 means that the True value was better than BEAST2 in 30% of cases), while the second table shows the median of differences between loglikelihood values.

The two results go in the same direction. The likelihood of both FFNN-SS and CNN-CBLV estimates is similar to BEAST2's, which explains the similar accuracy of the three methods (**Fig. 3**). Regarding the comparison with 'True values', if a given method tends to produce higher likelihood than that of the true parameter values, then it performs well in terms of likelihood optimization, as optimizing further should not result in higher accuracy. The results are again quite positive, as BEAST2 as well as our NNs achieved a higher likelihood than the true parameter values for ~70% of the trees, with a significant mean difference.

Supplementary Table 4: Parameter ranges used for simulations

Parameters	Name	Range	Inferred/set	Relation to other parameters
R_0	basic reproduction number	U(1,5)	inferred	$= \beta/\gamma$ (BD & BDEI) $= (\beta_{S,S} + \beta_{N,N})/\gamma$ (BDSS)
$1/\gamma$	infectious period	U(1,10)	inferred	
t	tree size	"small" trees: U(50, 199); "large" trees: U(200, 500)	set	
s	sampling probability	U(0.01,1)	set	
f_I	incubation factor	U(0.2,5)	parameterization	$= \varepsilon/\gamma$
$1/\varepsilon$	incubation period	[0.2, 50]	inferred	$= 1/(f_I * \gamma)$
X_{SS}	superspreading infectious ratio at equilibrium	U(3,10)	inferred	$= \beta_{S,S}/\beta_{N,S} = \beta_{S,N}/\beta_{N,N}$
f_{SS}	fraction of superspreading individuals at equilibrium	U(0.05, 0.20)	inferred	$= \beta_{S,S}/(\beta_{S,S}+\beta_{S,N})$ $= 1-\beta_{N,N}/(\beta_{N,N}+\beta_{N,S})$

For each parameter, we display its full name and the parameter range covered by simulations of training and testing sets. Note that all parameters were sampled from a uniform distribution, which we denote by U(x,y), with x being the lower and y the upper bound of that uniform distribution. The table further shows whether these parameters were inferred or used as input and, where appropriate, their relation to other model parameters (displayed in Figure 1). Parameters common to all three models are coloured in yellow, BDEI-specific in purple and BDSS-specific in green. The models are parameterized by the parameters of epidemiological interest except for the incubation factor, which enables to set the incubation period to values that are reasonable with respect to the infectious period. More specifically, through our parameterization choices, the incubation factor spans from 20% to 500% of the value of the infectious period, making it both not negligible but also not too high when compared to infectious period.

Supplementary Table 5: Method comparison in terms of model selection

a (i)

FFNN-SS	Actually BD	Actually BDEI
Predicted BD	94 (9360)	13 (1837)
Predicted BDEI	6 (640)	87 (8163)

a (ii)

FFNN-SS	Actually BD	Actually BDEI	Actually BDSS
Predicted BD	93 (9488)	11 (1006)	4 (561)
Predicted BDEI	6 (304)	86 (8783)	0 (281)
Predicted BDSS	1 (208)	3 (211)	96 (9158)

b (i)

CNN-CBLV	Actually BD	Actually BDEI
Predicted BD	96 (9289)	17 (1970)
Predicted BDEI	4 (711)	83 (8030)

b (ii)

CNN-CBLV	Actually BD	Actually BDEI	Actually BDSS
Predicted BD	91 (9131)	10 (1052)	3 (514)
Predicted BDEI	7 (519)	88 (8781)	1 (137)
Predicted BDSS	2 (350)	2 (167)	96 (9349)

c (i)

BEAST2	Actually BD	Actually BDEI
Predicted BD	75	4
Predicted BDEI	21	91
ESS<200	4	5

c (ii)

BEAST2	Actually BD	Actually BDEI	Actually BDSS
Predicted BD	62	0	2
Predicted BDEI	6	72	0
Predicted BDSS	6	7	72
ESS<200	26	21	26

Confusion matrices obtained with **a (i-ii)**, FFNN-SS, **b (i-ii)**, CNN-CBLV and **c (i-ii),** BEAST2 using AICM, either with **a-c (i)**, small trees (between 50 and 199 tips) or **a-c (ii)**, large trees (between 200 and 500 tips). BDSS and BDEI are nested within BD, namely, BDEI becomes BD when incubation period is 0 and BDSS becomes BD when superspreading individuals transmit at the same rate as normal individuals. We display the results as confusion matrices, actual classes being columns and predicted ones being rows. These were obtained with a test set of 100 simulations obtained with each model (or 10,000 for FFNN-SS and CNN-CBLV, results in parentheses). For BEAST2, 76% simulations converged for large trees and 96% for small trees. We show in red the number of simulations that did not reach an ESS of 200 for at least one parameter.

Supplementary Table 6: BEAST2 parameters and priors

Parameters	Name	BEAST2 parameter	Range	Initial value	Formula
γ	become uninfectious rate	Yes	U(0.1,1.0)	0.55	
$1/\gamma$	infectious period	No	[1,10]	1.8	
s	sampling probability	Yes	fixed	true value	
R_0	basic reproduction number	Yes	U(1.0,5.0)	3.0	$= \beta/\gamma$ (BD & BDEI)
ε	incubation rate	Yes	U(0.02,5.0)	2.51	
$1/\varepsilon$	incubation period	No	[0.2,50]	0.40	
$R_{0,SS}$	partial, within deme superspreading R_0	Yes	U(0.14,4.31)	1.25	
$R_{0,NN}$	partial, within deme normal spreading R_0	Yes	U(0.14,4.31)	1.44	
$R_{0,SN}$	partial, outside deme superspreading R_0	Yes	U(0.034,32.30)	9.0	
$R_{0,NS}$	partial, outside deme normal spreading R_0	Yes	U(0.034,32.30)	0.20	
R_0	basic reproduction number	No	[0.28,8.62]	2.69	$= R_{0,SS} + R_{0,NN}$ (BDSS)
X_{SS}	superspreading infectious ratio at equilibrium	No	$[4*10^{-4},127]$	6.25	$= R_{0,SS}/R_{0,NS} = R_{0,SN}/R_{0,NN}$
f_{SS}	fraction of superspreading individuals at equilibrium	No	$[4*10^{-3},0.99]$	0.12	$= R_{0,SS}/(R_{0,SS}+R_{0,SN})$ $= 1-R_{0,NN}/(R_{0,NN}+R_{0,NS})$

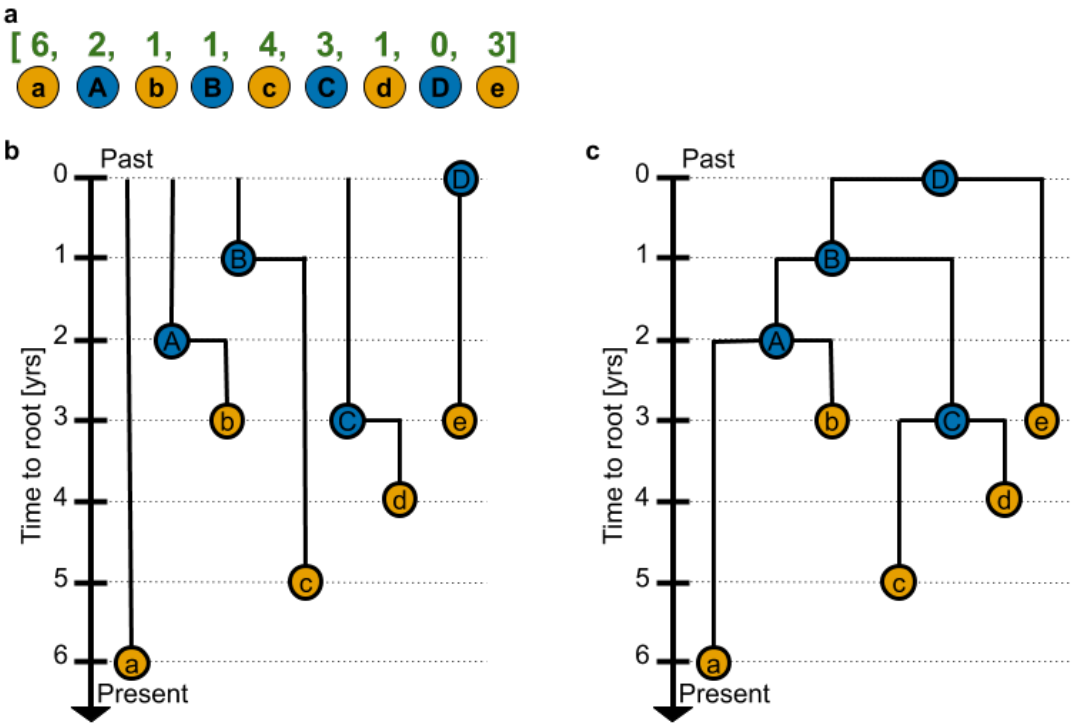
This table shows parameters and their prior distributions used during inference with BEAST2. We display the parameters, their definitions and priors in BEAST2, that are common to all models (in yellow), common to BD and BDEI (in red), BDEI-specific (in purple) and BDSS-specific (in green). From these parameters, we deduce the values and distributions of parameters of interest as shown in the table. Note that the parameters of epidemiological interest are basic reproduction number and infectious period for BD, BDEI and BDSS, incubation period for BDEI, and superspreading infectious ratio and fraction of superspreading individuals for BDSS. We check convergence (ESS) on all parameters and extract median *a posteriori* and CI values exclusively for the parameters of epidemiological interest.

Supplementary Table 7: Confidence interval assessment

Model	Parameter	Range	95% CI width and coverage					
			FFNN-SS computed on converged simulations		CNN-CBLV computed on converged simulations		BEAST2 computed on converged simulations	
			Coverage	Width	Coverage	Width	Coverage	Width
BD 200-500 tips	R_0	**U(1, 5)**	**0.99 (0.93)**	**0.99 (1.0)**	**0.98 (0.93)**	**1.0 (1.1)**	**0.99**	**0.99**
	$1/\gamma$	**U(1, 10)**	**0.97 (0.92)**	**1.6 (1.5)**	**0.97 (0.92)**	**1.6 (1.6)**	**1.00**	**1.6**
BDEI 200-500 tips	R_0	**U(1, 5)**	**0.89 (0.93)**	**1.0 (1.1)**	**0.92 (0.93)**	**1.1 (1.1)**	**0.88**	**1.0**
	$1/\varepsilon$	**[0.2, 50]**	**0.89 (0.91)**	**6.6 (6.6)**	**0.88 (0.92)**	**7.1 (7.2)**	**0.84**	**6.0**
	$1/\gamma$	**U(1, 10)**	**0.84 (0.93)**	**2.1 (2.0)**	**0.93 (0.93)**	**2.1 (2.1)**	**0.90**	**2.1**
BDSS 200-500 tips	R_0	**U(1, 5)**	**0.94 (0.93)**	**1.1 (1.2)**	**0.93 (0.93)**	**1.2 (1.2)**	**0.94**	**1.1**
	$1/\gamma$	**U(1, 10)**	**0.87 (0.92)**	**1.6 (1.7)**	**0.88 (0.91)**	**1.6 (1.7)**	**0.94**	**1.9**
	X_{SS}	**U(3, 10)**	**0.91 (0.90)**	**3.5 (3.5)**	**0.91 (0.90)**	**3.6 (3.6)**	**0.97**	**3.5**
	f_{SS}	**U(0.05, 0.20)**	**0.82 (0.78)**	**0.087 (0.086)**	**0.80 (0.78)**	**0.089 (0.089)**	**0.94**	**0.110**
BD 50-199 tips	R_0	**U(1, 5)**	**0.90 (0.91)**	**1.4 (1.5)**	**0.90 (0.90)**	**1.4 (1.5)**	**0.91**	**1.4**
	$1/\gamma$	**U(1, 10)**	**0.86 (0.87)**	**2.3 (2.2)**	**0.85 (0.87)**	**2.3 (2.2)**	**0.94**	**2.4**
BDEI 50-199 tips	R_0	**U(1, 5)**	**0.93 (0.89)**	**1.6 (1.5)**	**0.93 (0.89)**	**1.6 (1.6)**	**0.96**	**1.7**
	$1/\varepsilon$	**[0.2, 50]**	**0.91 (0.89)**	**9.7 (9.3)**	**0.88 (0.88)**	**10 (9.8)**	**0.96**	**12**
	$1/\gamma$	**U(1, 10)**	**0.95 (0.90)**	**2.8 (2.8)**	**0.93 (0.90)**	**2.9 (2.8)**	**0.99**	**3.1**

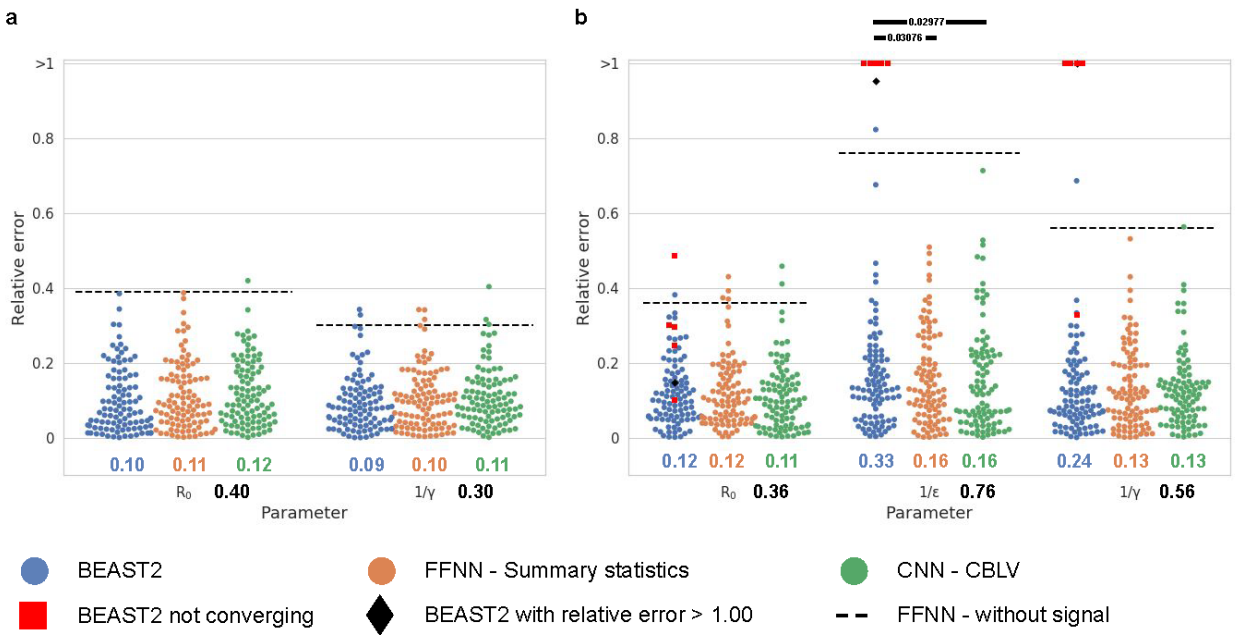
For each model and inferred parameter, we compared the different inference methods in terms of 95% confidence interval (95% CI) width and coverage, the latter being defined as the fraction of samples where the true value was within the 95% CI. We compared FFNN-SS and CNN-CBLV methods to BEAST2 on a set of 100 simulations. We did not consider the simulations for which BEAST2 did not converged (2% for BDEI large trees, 5% for BDEI small trees and 15% for BDSS large trees). We evaluated the same metrics on 10,000 simulations for FFNN-SS and CNN-CBLV (in parentheses). For FFNN-SS and CNN-CBLV we performed an approximated parametric bootstrap, while for BEAST2, we considered the entire chain with exception of the initial 10% burn-in. We highlight poor performance (coverage ≤ 0.85, width $\geq 1/3^{rd}$ of prior) in one of these metrics in red and good performance (coverage ≥ 0.95) in green. Globally, the 95% CI width and coverage of FFNN-SS and CNN-CBLV are comparable with BEAST2 (while not penalizing BEAST2 for non-converged estimations) and reflect the accuracy of parameter predictions (Fig. 3 and Supplementary Fig. 2). They are slightly better for BDEI with large trees and slightly worse for BDEI with small trees. For details on how the 95% CIs were obtained, see Methods.

Supplementary Figure 1: Bijectivity of the CBLV representation



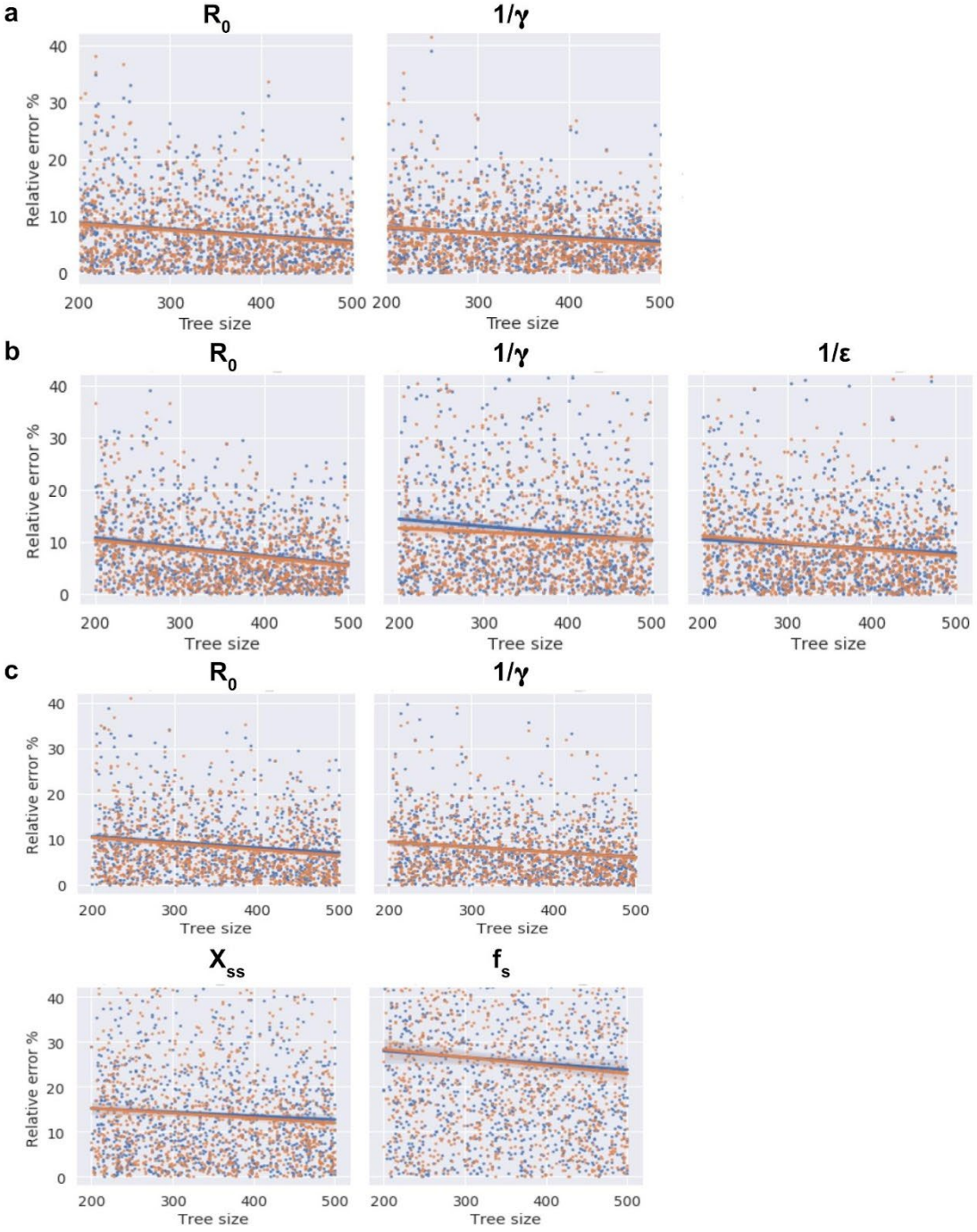
Trees are assumed to be ordered, *e.g.*, using ladderization or any other criterion. The inorder tree traversal procedure transforms an ordered tree into a unique vector. We show in this figure using a simple example, how an ordered tree is reconstructed without ambiguity from its vectorial representation, thus demonstrating the bijectivity (1-to-1 correspondence between trees and tree-compatible vectors) of the representation. **a,** shows the Compact Bijective Ladderized Vector (CBLV) representation from Figure 2 a (iii); the nodes are named alphabetically in order of appearance. Lower case letters are highlighted in yellow and represent the external nodes (or tips), upper-case letters are highlighted in blue and represent the internal nodes. **b,** depicts the creation of individual 'paths' comprising each pair of nodes: one external and one internal node, taken directly from the vector representation in the order of appearance. Note that the first external node is not paired with an internal node as it is connected to the root. **c,** shows how the tree reconstruction is achieved, by simply joining the paths from **b,** one by one from left to right. Note that not all vectors correspond to trees, as all entries must be positive or null, plus additional constraints and inequalities (*e.g.*, the second entry (A=2) must be less than (or equal to) the first entry (a=6))**.**

Supplementary Figure 2: Assessment of deep learning accuracy on small trees



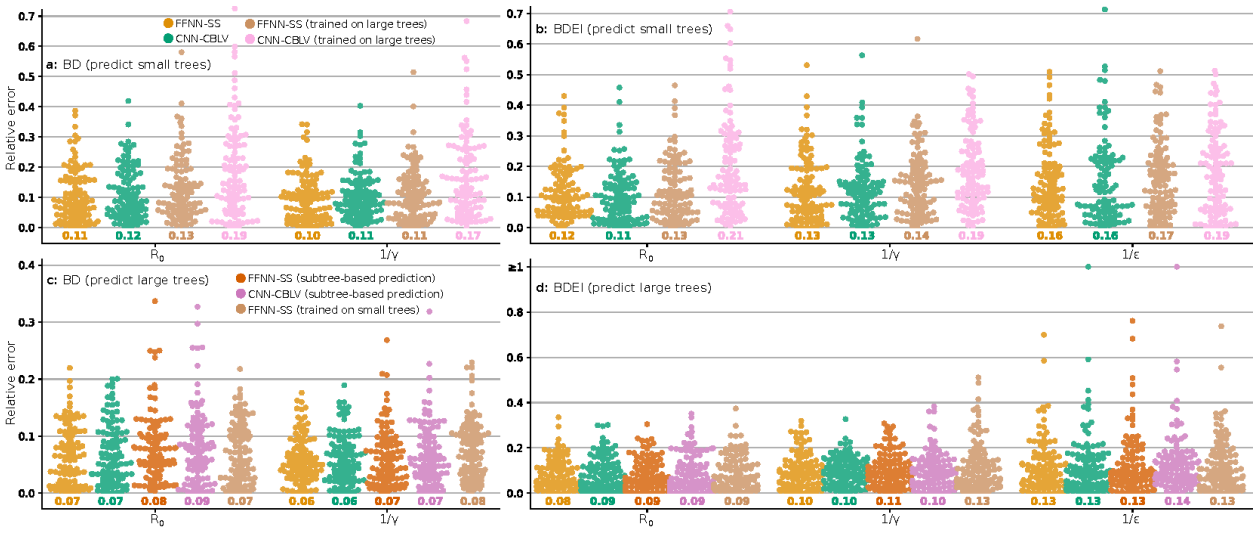
Comparison of inference accuracy by BEAST2 (in blue), FFNN-SS (in orange) and CNN-CBLV (in green) on 100 small test trees (50-199 tips). We compare the relative error for each tree, between the median *a posteriori* estimate by BEAST2 or point estimates by neural networks and the target value for each parameter. We highlight simulations for which BEAST2 did not converge and whose values were thus set to the median of the covered parameter space by depicting them as red squares. We further highlight the analyses with a high relative error (>1.00) for one of the estimates as black diamonds. We compare the relative errors for **a,** BD-simulated, and **b,** BDEI-simulated small trees. Average relative absolute error (MRE) is displayed under each distribution in the corresponding colour. The average error of an FFNN trained on summary statistics but with randomly permuted target is displayed as black dashed line and its value is shown in bold black below the x-axis. The accuracy of the output of each method is compared by two-sided paired z-test; *p-value* < 0.05 is shown as thick full line along with the corresponding *p-value*; non-significant when not shown. The accuracy is similar for the BD model, while the NNs reach better accuracy for BDEI model, while avoiding problems with convergence (5% BEAST2 inferences did not converge).

Supplementary Figure 3 Accuracy of deep learning methods increases with tree size



For each model **a,** BD, **b,** BDEI and **c,** BDSS, we display the regression on relative error for each parameter as a function of tree size. We show the error for both CNN-CBLV (blue) and FFNN-SS (orange) estimated for 1,000 trees (instead of 10,000 trees for display purposes). As expected, and consistent with statistical learning theory, for each parameter the accuracy increases with tree size. For example, for BDEI, the relative error of R_0 is on average 11% for trees of 200 tips and decreases to 6% for trees of 500 tips. Importantly, the relative error of superspreading fraction f_{SS}, a parameter that is difficult to estimate, decreases from 28% for trees of 200 tips to 23% for trees of 500 tips.

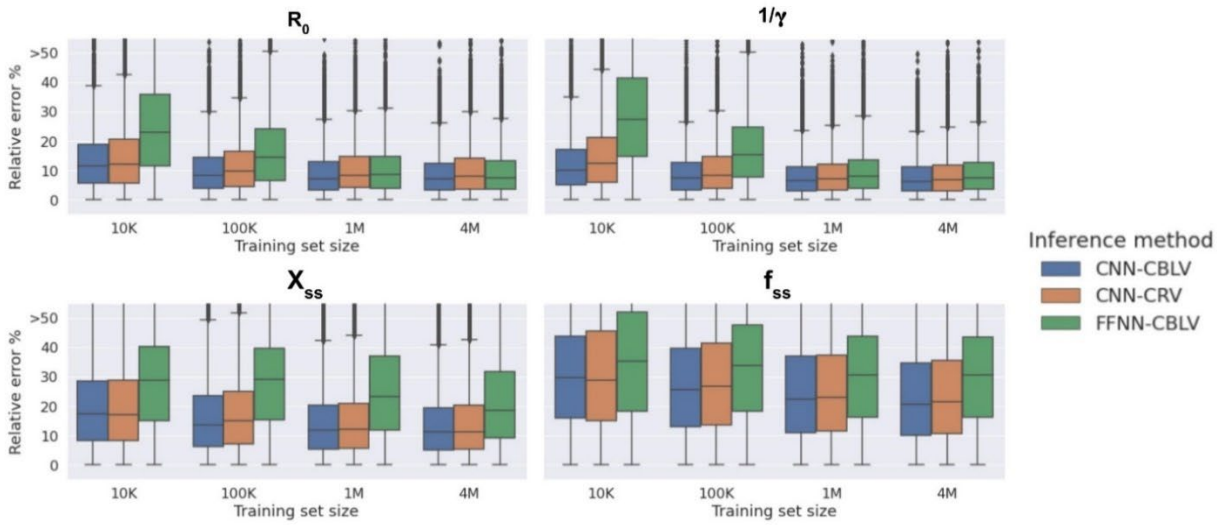
Supplementary Figure 4: Assessment of deep learning generalization capabilities



Comparison of inference accuracy by neural networks trained on trees of sizes different from those of the test trees:(top) trained on large trees and evaluated on prediction with small trees (FFNN-SS in beige, CNN-CBLV in pink); and (bottom) trained on small trees and evaluated on prediction with large trees (FFNN-SS in beige, FFNN-SS using subtree picking-and-averaging in red, CBLV-NN using subtree picking-and-averaging in magenta). For comparison, we also show FFNN-SS (orange) and CNN-CBLV (green) trained and evaluated on prediction with trees of compatible sizes (small on top, large on the bottom). The training and testing trees are the same as in **Fig. 3** (large) and **Supplementary Fig. 2** (small). We show the relative error for each test tree. The error is measured as the normalized distance between the point estimates by neural networks and the target value for each parameter. We compare the relative errors for **a, c**, BD-simulated, **b, d**, BDEI-simulated trees. Average relative error is displayed for each parameter and method in corresponding color below each figure.

The results are surprisingly good, especially with summary statistics (FFNN-SS) which are little impacted by these changes of scale as they largely rely on means. Moreover, the subtree picking-and-averaging approach performs well for both FFNN-SS and CNN-CBLV, which confirms the finding in **Fig. 4**.

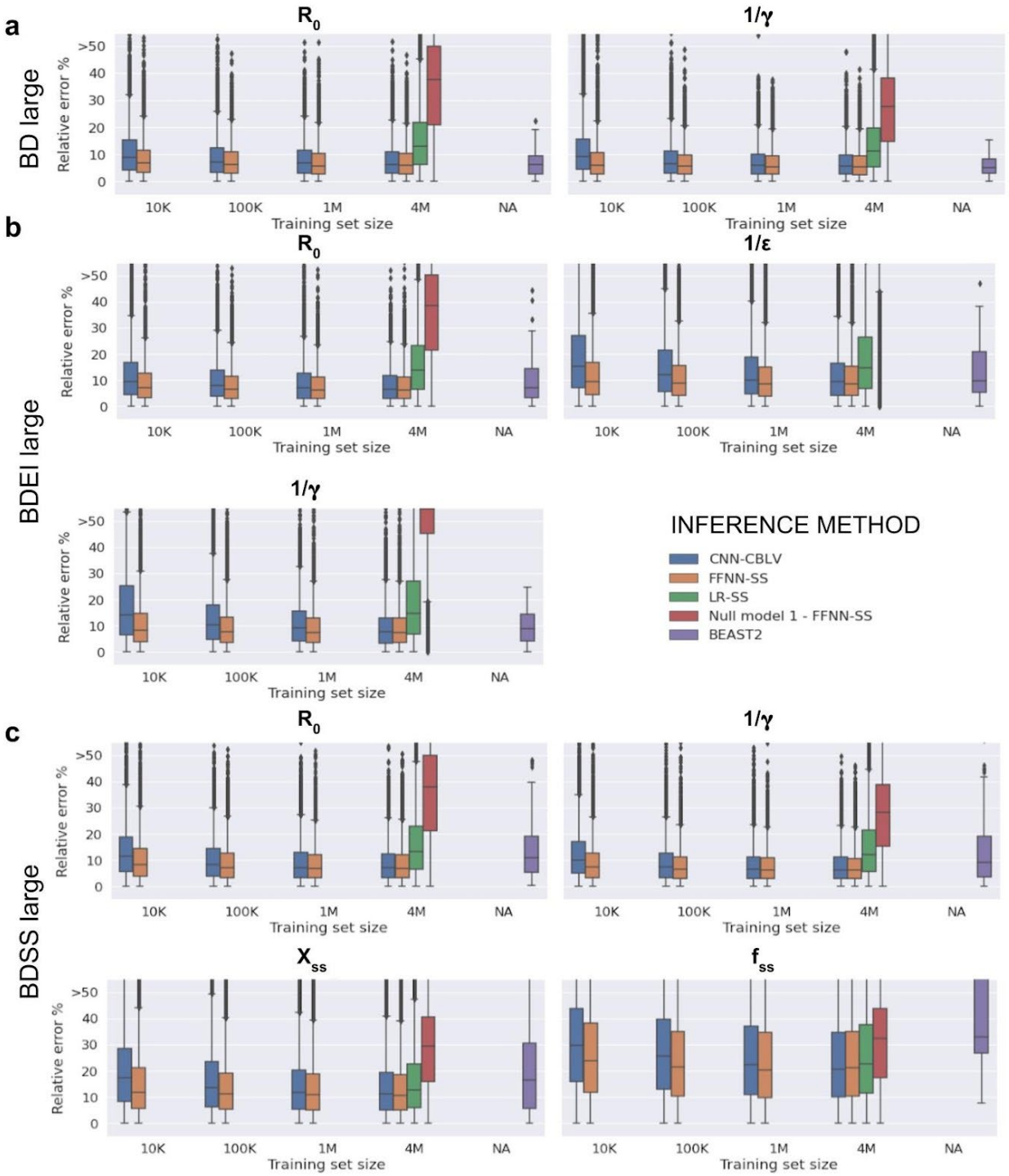
Supplementary Figure 5: CNN performs well with CBLV tree representation



We display percentage absolute error of point estimates from deep learning methods on 10,000 test trees generated with BDSS. We compare CNN-CBLV (in blue) with FFNN-CBLV (in green) and CNN-CRV (in orange), which is a CNN trained on a Compact Random Vector (CRV) representation, where all internal nodes are randomly rotated instead of being ladderized, (in orange). In addition, we show the accuracy of these models when trained on varying training set sizes (10K: 10,000; 100K: 1,000,000; 1M: 1,000,000; and 4M: 4,000,000 trees). Boxplots are 95% CIs, the middle bar shows the mean, and the middle box corresponds to the 25% and 75% percentiles of the relative errors with 10,000 test trees.

For most parameters and training set sizes, the accuracy of CNN-CRV is lower than the one of CNN-CBLV, especially with low number of training examples, while with 4M the accuracy of both methods becomes relatively close. Thanks to ladderization, CBLV is thus enabling the CNN to learn faster and more accurately than if trained on CRV. The difference in performance is even more striking for FFNN-CBLV when compared to CNN-CBLV.CNN-CBLV is more accurate across all parameters and training set sizes, especially for X_{SS} and f_{SS} parameters even after being trained on 4M examples.

Supplementary Figure 6: How much and how fast FFNN-SS and CNN-CBLV learn?



We display the distribution of relative error of individual point estimates from statistical learning methods for 10,000 trees, and median *a posteriori* values from BEAST2 for 100 large trees simulated under each of the three birth-death models **a,** BD, **b,** BDEI, **c,** BDSS. As there are many more points considered for statistical learning methods, the BEAST2 measures are shown for indicative purposes only (in purple). Boxplots are 95% CIs, the middle bar shows

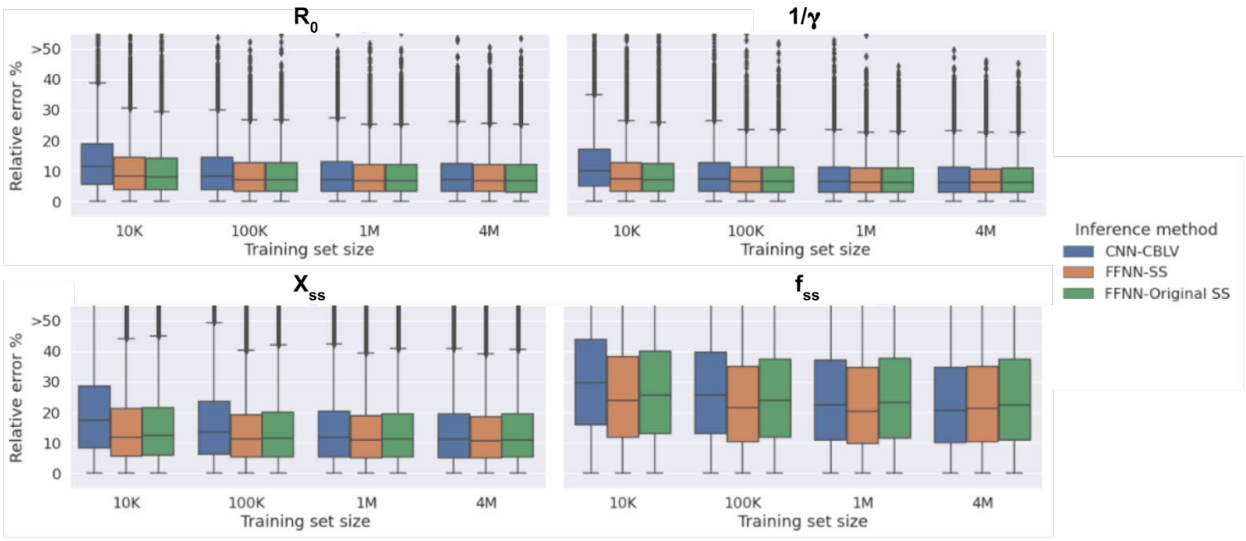
the mean, and the middle box corresponds to the 25% and 75% percentiles. We compare CNN-CBLV (in blue), FFNN-SS (in orange), LR-SS as a baseline model (in green) and "Null model 1", for which the FFNN was trained on summary statistics to predict permuted target values (in red). The Null model maintains the same cost function as other neural networks and thus minimizes the mean percentage relative error in the absence of any signal.

In addition, we show the accuracy of estimated statistical models trained on varying training set sizes (10K: 10,000; 100K: 100,000; 1M: 1,000,000; and 4M: 4,000,000 trees), to study the efficiency of the different learning approaches.

The FFNN-SS accuracy culminates at a training size of 100,000, while CNN-CBLV keeps improving at a training size of 4,000,000. This is consistent with the fact that summary statistics represent high-level information, while the CNN has to learn how to infer parameter values from raw information and thus require many more examples.

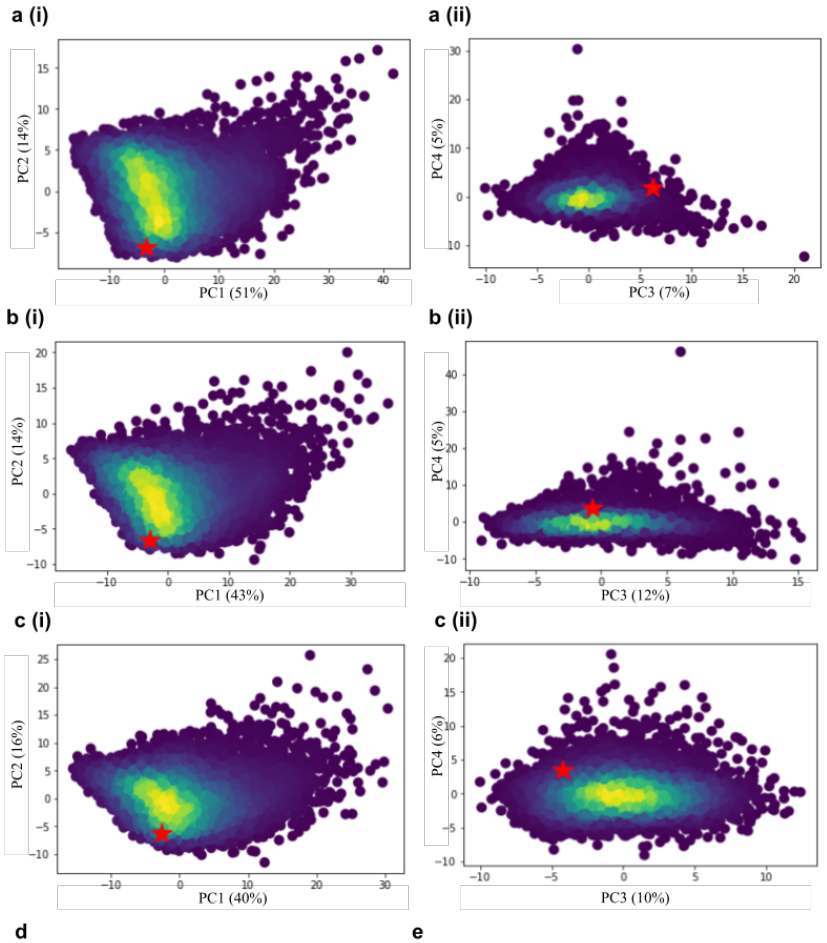
The baseline model LR-SS does not reach the same level of accuracy as FFNN-SS and CNN-CBLV for most parameters, but X_{SS} and f_{SS} parameters in BDSS. The relationship between the summary statistics and the studied model parameter values is too complex to be handled by linear regression. Finally, by comparing the accuracies to those of "Null model 1", we show how much information is extracted by the properly set-up deep-learning methods as opposed to a model trained in the absence of signal. In most cases, the accuracy gain is high. The only exception is f_{SS} parameter, where the "Null model 1" has an accuracy of 0.33, while with CNN-CBLV and FFNN-SS we reach an accuracy of around 0.25. These two approaches thus do not estimate this parameter very accurately, most likely because (1) f_{SS} is a fraction of rates and thus cumulates several errors, and (2) the information in the tree on superspreading is low, as the fraction and thus the number of superspreading individuals is low as well. One solution to this problem is to gather more data. Indeed, as shown in Supplementary Fig.7 accuracy increases substantially when learning and inferring on larger trees.

Supplementary Figure 7: Adding new SS to increase accuracy of FFNN-SS



Adding SS on transmission chains improves the accuracy of prediction of superspreading individual frequency f_{ss} in the BDSS model. We display relative absolute error (RE) of point estimates from deep learning methods for 10,000 trees simulated under the BDSS model. Boxplots are 95% CIs, the middle bar shows the mean, and the middle box corresponds to the 25% and 75% percentiles. We compare CNN trained on CBLV representation with FFNN trained on SS and on Original Saulnier's SS (*i.e.,* without SS on transmission chains), for each parameter of interest. In addition, we show the accuracy of NN models trained on varying training set sizes (10K: 10,000; 100K: 100,000; 1M: 1,000,000; and 4M: 4,000,000 trees). This shows that additional SS enable to decrease the MRE by over 2% for f_{ss} making it comparable to CNN-CBLV accuracy with large training sample (4M).

Supplementary Figure 8: *A priori* and *a posteriori* checks of model adequacy for HIV data



d

Model	# SS outside Min-Max	SS Names
BD	5	ie BL mean 2 and 3, ie BL median 2 and 3, ie BL var 3
BDEI	5	min Height, ie BL mean 2 and 3, ie BL median 3, ie BL var 3
BDSS	0	-

e

BDSS	# SS outside Min-Max	SS Names
FFNN-SS	13	max_L, slope_1, mean_s_time, x_16, x_17, x_18, y_3-y_9
CNN-CBLV	20	max_L, t_max_L, slope_1, mean_s_time, x_11-x_18, y_2-y_9

For each model **a**, BD, **b**, BDEI and **c**, BDSS, we encoded 10,000 simulations from the test set into SS and standardized them. We then performed principal component analysis (PCA) and projected the SS from HIV phylogeny (red star) on these PCA plots. Here we show the projections along **a-c (i)**, the 1st and the 2nd components (PC1 and PC2) and **a-c (ii)**, the 3rd and the 4th components (PC3 and PC4) of the PCA, together with the associated percentage variance explained in parentheses. For each model and projection, the HIV data point is surrounded by the simulations, meaning it resembles globally the simulations and thus we can apply our deep learning (and BEAST2) inference methods under each birth-death model.

Furthermore, we performed more detailed **d**, *a priori* and **e**, *a posteriori* checks using directly the values of summary statistics without a PCA. For each statistics of HIV phylogeny, we checked whether it lays between the minimum and the maximum value of this statistics covered in **d**, test set of 10,000 trees for given model (BD, BDEI or BDSS), **e**, the *a posteriori* set under BDSS (10.000 simulations, see text). The statistics that were outside the minimum and maximum values are numbered and named in **d** and **e**. All rejected SS in *a posteriori* check (**e**) correspond to the LTT plot (*e.g.*, x and y coordinates), consistent with the fact that the probabilistic, sampling component of the BDSS model is an oversimplification of actual sampling schemes, which depend on contact tracing, sampling campaigns and policies, etc. For details on each statistics, refer to https://doi.org/10.1371/journal.pcbi.1005416.

METHODS – EXTENDED VERSION

TREE REPRESENTATION USING SUMMARY STATISTICS (SS)

We use a set of 98 summary statistics (SS), to which we add the sampling probability, summing to a vector of 99 values.

Saulnier et al summary statistics

We use the 83 SS proposed by Saulnier *et al.*[19]:

- 8 SS on tree topology
- 26 SS on branch lengths
- 9 SS on Lineage-Through-Time (LTT) plot
- 40 SS providing the coordinates of the LTT plot

The computing time of these statistics grows linearly with tree size. For details, see the original paper.

Additional summary statistics

In addition to Saulnier *et al.*[19] statistics, we designed 14 SS on transmission chains. Moreover, we provide the number of tips in the tree as input resulting in 83+14+1 = 98 SS in total.

The statistics on transmission chains are designed to capture information on the superspreading population. A superspreading individual transmits to more individuals within a given time period than a normal spreader. We thus expect that with superspreading individuals we would have shorter transmission chains. To have a proxy for the transmission chain length, we look at the sum of 4 subsequent shortest times of transmission for each internal node. This gives us a distribution of time-durations of 4-transmission chains. We assume that information on the transmission dynamics of superspreading individuals is retained in the lower (*i.e.*, left) tail of 4-transmission-chain lengths distribution, which contains relatively many transmissions with short time to next transmission, while the information on normal spreaders should be present in the rest of the distribution.

The implementation of these 4-transmission-chain SS is the following. For each internal node, we sum the distances from the internal node to its closest descendant nodes, descending exactly four times, that is, we take first the distance from the given internal node to its closest child node (of level 1), then from the (level 1) child node, we take its distance to its own closest child node (of level 2), *etc.* If one of the closest descendant nodes is a tip (except for the last one in

the chain), we do not retain any value for the given internal node. Other options, like the shortest 4-edge pathway, could have been used as well and would likely give comparable results.

On the obtained distribution of 4-transmission-chain lengths, we compute 14 statistics:

- number of 4-transmission chains in the tree
- 9 deciles of 4-transmission-chain lengths distribution
- minimum and maximum values of 4-transmission-chain lengths distribution
- mean value of 4-transmission-chain lengths
- variance of 4-transmission-chain lengths

Adding the same summary statistics but on chains comprising 2, 3 and 5 consecutive transmissions had a negligible impact on parameter inference accuracy (data not shown).

COMPLETE AND COMPACT TREE REPRESENTATION (CBLV)

Simulated dated trees are encoded in the form of real-valued vectors, which are then used as input for the neural networks. The representation of a tree with n tips is a vector of length $2n$-1, where one single real-valued scalar corresponds to one internal node or tip. This representation thus scales linearly with the tree size. The encoding is achieved in two steps: tree ladderization and tree traversal.

Tree ladderization

The tree ladderization consists in ordering the children of each node. Child nodes are sorted based on the sampling time of the most recently sampled tip in their subtrees: for each node, the branch supporting the most recently sampled subtree is rotated to the left, as in **Fig. 2 a (i-ii)**.

We considered several alternatives with different criteria for child (subtree) sorting instead of ladderization: sampling time of the most anciently sampled tip, subtree length (*i.e.*, sum of all branch lengths including the rooting branch), diversification (*i.e.*, number of tips), normalized branch lengths (*i.e.*, subtree length divided by the number of tips), *etc*. These did not yield better results than CBLV. We show in **Supplementary Fig. 5** the comparison of CBLV with Compact Random Vector (CRV), for which internal nodes were sorted randomly before the tree traversal, showing that CRV yields poorer results than CBLV, as expected.

Tree traversal and encoding

Once the tree is sorted, we perform an inorder tree traversal, using a standard recursive algorithm from the depth first family[30]. When visiting a tip, we add its distance to the previously visited internal node or its distance to the root, for the tip that is visited first (*i.e.*, the tree height due to ladderization). When visiting an internal node, we add its distance to the root. Examples of encoding are shown in **Fig. 2 a (ii-iii)**. This gives us the Compact Bijective Ladderized Vector (CBLV). We then separate information relative to tips and to internal nodes into two rows (**Fig. 2 a (iv)**) and complete the representation with zeros until reaching the size of the largest simulated tree for the given simulation set (**Fig. 2 a (v)**).

Properties of CBLV

CBLV has favourable features for deep learning. Ladderization does not actually change the input tree (phylogenies are unordered trees), but by ordering the subtrees it standardizes the input data and facilitates the learning phase, as observed with CRV (**Supplementary Fig. 5**). Then, the inorder tree traversal procedure is a bijective transformation, as it transforms a tree into a tree-compatible vector, from which the (ordered) tree can be reconstructed unambiguously, using a simple path-agglomeration algorithm shown in **Supplementary Fig. 1**. Note, however, that not all vectors are tree-compatible (*e.g.,* all entries must be non-negative; the second entry must be less than the first one, *etc*.). CBLV is "as concise as possible" being composed of $2n$-1 real values (**Fig. 2 a (iii)**), where n is the number of tips. A rooted tree has $2n$-2 branches, and thus $2n$-2 entries are needed to represent the branch lengths. In our $2n$-1 vectorial encoding of trees, we not only represent the branch lengths, but also the tree topology using only 1 additional entry.

The compactness and bijectivity of tree representation reduce the number of simulations required for training the neural network (**Supplementary Fig. 5**). This is because the number of parameters to be trained remains reasonable with compact representation. Moreover, the networks do not need to learn that several different inputs correspond to the same tree.

Our neural networks are intended to apply to trees of variable sizes (*e.g.*, trees of 200 to 500 tips in our experiments with 'large' trees). Thus, they are trained on representations of different lengths (*e.g.*, a vector of length 399 for a tree of 200 tips), that we complete with zeroes to reach the length of the largest trees (*i.e.*, 999 for 500 tips). We add an additional zero to obtain a two-row matrix (500*2 for 500 tips).

Alternative tree representations

Our CBLV tree representation could likely be improved to ease the learning phase and obtain even better parameter estimates. We tested several alternative representations, some inspired by the polynomial representation of small subtrees[26,27], the Laplacian spectrum[28] and additive distance matrices that are equivalent to trees[58]. None was by far as convincing as CBLV, which is likely due to their large size (*e.g.*, n^2 for distance matrices) or numerical instabilities and potential loss of information (*e.g.*, for Laplacian spectrum). Moreover, the margin for improvement of the accuracy of CNN-CBLV for the BD model, and likely for other models, is low. This is due to the observation that the accuracy of CNN-CBLV is similar to that of likelihood-based approaches for the BD model, and the fact that we have an analytical likelihood formula for the BD model, making the likelihood-based approach itself optimal[8,9].

TREE RESCALING

Before encoding, the trees are rescaled so that the average branch length is 1, that is, each branch length is divided by the average branch length of the given tree, called rescale factor. The values of the corresponding time-dependent parameters (*i.e.*, infectious period and incubation period) are divided by the rescale factor too. The NN is then trained to predict these rescaled values. After parameter prediction, the predicted parameter values are multiplied by the rescale factor and thus rescaled back to the original time scale.

This step enables us to overcome problems of arbitrary time scales of input trees and makes a pre-trained NN more generally applicable. More specifically, an input tree with a time scale in days will be associated naturally with the same output as the same tree with a time scale in years, since both these trees will be rescaled to the same intermediate tree with average branch length of 1. Rescaling thus makes it possible to apply the same pre-trained NN to phylogenies reconstructed from sequences of a pathogen associated with an infectious period on the scale of days (*e.g.*, Ebola virus) or years (*e.g.*, HIV).

REDUCTION AND CENTERING OF SUMMARY STATISTICS REPRESENTATION

Before training our NN and after having rescaled the trees to unit average branch length (see the sub-section above), we reduce and center every summary statistic by subtracting the mean and scaling to unit variance. To achieve this, we use the standard scaler from the scikit-learn package[50], which is fitted to the training set. This does not apply to CBLV representation, to avoid losing the ability to reconstruct the tree.

PARAMETER AND MODEL INFERENCE USING NEURAL NETWORKS

We implemented deep learning methods in Python 3.6 using Tensorflow 1.5.0[51], Keras 2.2.4[52] and scikit-learn 0.19.1[50] libraries. For each network, several variants in terms of number of layers and neurons, activation functions, regularization, loss functions and optimizer, were tested. In the end, we decided for two specific architectures that best fit our purpose: one deep FFNN trained on SS and one CNN trained on CBLV tree representation.

Deep feedforward neural network architecture for SS

The network consists of one input layer (of 99 input nodes both for trees with 50-199 and 200-500 tips), 4 sequential hidden layers organized in a funnel shape with 64-32-16-8 neurons and 1 output layer of size 2-4 depending on the number of parameters to be estimated. The neurons of the last hidden layer have linear activation, while others have exponential linear activation[53].

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 64)	6400
dense_2 (Dense)	(None, 32)	2080
dense_3 (Dense)	(None, 16)	528
dense_4 (Dense)	(None, 8)	136
dense_5 (Dense)	(None, 2)	18

Total params: 9,162
Trainable params: 9,162
Non-trainable params: 0

Architecture: **Feedforward neural network architecture.** Example of FFNN trained on large trees to estimate the parameters of the BD model (R_0 and infectious period $1/\gamma$). 'Dense' layer means that for each neuron, all the inputs are multiplied by learned weights, summed together with the bias term. The activation function is then applied to the weighted sum before being output to the next layer. Dense_1 to dense_4 are layers with neurons of exponential linear activation, while dense_5 is composed either of softmax (in case of model selection) or of linear neurons (in case of parameter estimation). The number of trainable parameters in each layer is displayed (Param #): for example in the first layer, we have 99 input values and 1 bias for each of the 64 neurons, giving us in total (99+1)*64=6,400 trainable parameters. Output by Keras[52], the 'None' in the 'Output Shape' means the network can input more than one training example at the time and that there is no constraint on the batch size (hence 'None').

Deep convolutional neural network for CBLV

The CNN consists of one input layer (of 400 and 1002 input nodes for trees with 50-199 and 200-500 tips, respectively). This input is then reshaped into a matrix of size of 201*2 and 501*2, for small and large trees, respectively, with entries corresponding to tips and internal nodes separated into two different rows (and one extra column with one entry in each row corresponding to the sampling probability). Then, there are two 1D convolutional layers of 50 kernels each, of size 3 and 10, respectively, followed by max pooling of size 10 and another 1D convolutional layer of 80 kernels of size 10. After the last convolutional layer, there is a GlobalPoolingAverage1D layer and a FFNN of funnel shape (64-32-16-8 neurons) with the same architecture and setting as the NN used with SS.

Neural network setting and training

For both NNs, we use the Adam optimisation algorithm[54] as optimizer and the Mean Absolute Percentage Error (MAPE) as loss function. The batch size is set to 8,000. To train the network, we split the simulated dataset into 2 groups: (1) proper training set (3,990,000 examples); (2) validation set (10,000).

Preventing overfitting: Early stopping and Dropout

To prevent overfitting during training, we use: (1) the early stopping algorithm evaluating MAPE on a validation set; and (2) dropout that we set to 0.5 in the feed-forward part of both NNs[55] (0.4, 0.45, 0.55 and 0.6 values were tried for basic BD model without improving the accuracy).

Neural networks for model selection

For model selection, we use the same architecture for FFNN-SS and CNN-CBLV as those for parameter inference described above**.** The only differences are: (1) the cost function: categorical cross entropy and (2) the activation function used for the output layer, that is, softmax function (of size 2 for small trees, selecting between BD and BDEI model, and of size 3 for large trees, selecting between BD, BDEI and BDSS). As we use the softmax function, the outputs of prediction are the estimated probabilities of each model, summing to 1.

The FFNN-SS and CNN-CBLV are trained on $8*10^6$ trees in the small tree setting ($4*10^6$ trees per model, BD and BDEI). In the large tree setting, the FFNN-SS is trained on $12*10^6$ trees ($4*10^6$ trees per model, BD, BDEI and BDSS)

and the CNN-CBLV is trained on $9*10^6$ trees ($3*10^6$ trees per model, BD, BDEI and BDSS), instead of $12*10^6$ for GPU limitation purposes.

Parameter estimation from very large trees using subtree picking and averaging

To predict from very large trees (*e.g.*, our 'huge' trees having 5,000 to 10,000 tips, **Fig. 4**) we designed the 'Subtree Picker' algorithm. The goal of Subtree Picker is to extract subtrees of bounded size representing independent sub-epidemics within the epidemic represented by the initial huge tree T, while covering most of the initial tree branches and tips in T. The sub-epidemics should follow the same sampling scheme as the global epidemic. This means that we can stop the sampling earlier than the most recent tip in T, but we cannot omit tips sampled before the end the sampling period (this would correspond to lower sampling probability). Each picked subtree corresponds to a sub-epidemic that starts with its root individual and lasts between its root date D_{root} and some later date ($D_{last} > D_{root}$). The picked subtree corresponds to the top part of the initial tree's clade with the same root, while the tips sampled after D_{last} are pruned.

The picked subtrees do not intersect with each other and contain between m and M tips each. Together they cover most of the initial tree's branches. The initial tree T contains more than M tips. In the current PhyloDeep setting, M=500 (the largest tree size in the training set) and m=200 for BDSS and =50 for BD and BDEI (the smallest tree size in the training set).

Subtree Picker performs a postorder tree traversal (tips-to-root), where for each tree node N it calculates the maximum number of tips t_N that can be extracted from its subtrees. The algorithm is recursive and combines two basic strategies: (1) the subtree rooted with N is decomposed into two independent sub-epidemics corresponding to N's direct descendants; or (2) we decide to keep the upper part (with x oldest tips, $m \leq x \leq M$) of the subtree rooted with N, and the rest of N's descendants is decomposed into independent sub-epidemics. The recursion is as follows (size(N) is the number of tips in the subtree rooted with N):

> If size(N) < m, then t_N=0
> Else if $m \leq$ size(N) < $M + m$, then t_N=max(size(N), M) *#pick the root subtree containing* t_N *oldest tips*
> Else pick the best between the two strategies:
> (1) Left L and right R children of N lead to independent sub-epidemics (subtrees): $t_N=t_L+t_R$

(2) Pick the root subtree of x oldest tips ($m \leq x \leq M$, all possible x compared) plus the set Δ of the oldest descendant nodes of N, which represents the roots of independent sub-epidemics sampled after x^{th} tip date: $t_N = x + \sum_{D \text{ in } \Delta} t_D$

For each processed node N, its optimal subtree picking strategy is memorized. Once t_N is calculated for the root of the global tree T, the algorithm picks the root's subtrees according to the chosen strategy and, if needed, descends to the non-affected descendant nodes to pick more subtrees.

Let n be the size of T (number of tips). This tree decomposition requires one preorder tree traversal, with computing time in $O(n)$. However, the picking strategy (2) requires another $O(s)$ time to extract Δ (with appropriate data structure and pre-treatments). Thus, the whole computing time (computing (2) for each node in the tree traversal) is in $O(n^2)$, in the worst case. In practice, the subtrees extracted by Subtree Picker cover on average 98.5% (BD), 97.3% (BDEI) and 82.4% (BDSS) of the initial tree branches on the 'huge' tree datasets (5,000 to 10,000 tips). For the BDSS model this percentage is lower than for BD and BDEI, because of the narrower subtree size interval (m=200, M=500 versus m=50, M=500) corresponding to current PhyloDeep training set settings. In terms of computing time, Subtree Picker takes on average 0.6 (BD), 0.8 (BDEI) and 0.8 (BDSS) seconds per 'huge' tree, meaning that it could easily be applied to much larger trees.

Once subtrees (sub-epidemics) have been extracted, they are analysed using CNN-CBLV or FFNN-SS, and the parameter estimates are averaged with weights proportional to subtree sizes (number of tips).

CONFIDENCE INTERVALS (95% CI)

Computation of 95% CI

We compute 95% CI using parametric bootstrap. To facilitate the deployment and speed-up the computation, we perform an approximation using a separate set of 1,000,000 simulations for calculation of CI. For each simulation in the CI set, we store the true parameter values *(i.e.*, values with which we simulated the tree) and the parameter values predicted with both of our methods. This large dataset of true/predicted values is used to avoid new simulations, as required with the standard parametric bootstrap.

For a given simulated or empirical tree T, we obtain a set of predicted parameter values, $\{p\}$. The CI computation procedure searches among stored data those that are closest to T in terms of tree size, sampling probability and predicted values. We first subset:

- 10% of simulations within the CI set, which are closest to T in terms of size (number of tips), thus obtaining 100,000 CI sets of true/predicted parameter values.
- Amongst these, 10% of simulations that are closest to T in terms of sampling probability.

We thus obtain 10,000 CI sets of real/predicted parameter values, similar in size and sampling probability to T. For each parameter value *p* predicted from T, we identify the 1,000 nearest neighbouring values amongst the 10,000 true values of the same parameter available in the CI sets, $R_{CI} = \left\{ r_{i=1,1000} \right\}$, and keep the corresponding predicted values, $P_{CI} = \left\{ p_{i=1,1000} \right\}$. We then measure the errors for these neighbours as $E_{CI} = \left\{ e_i = p_i - r_i \right\}$. We center these errors around p using the median of errors, $m\left(E_{CI}\right)$, which yields the distribution of errors for given prediction p: $D = \left\{ p + e_i - m\left(E_{CI}\right) \right\}$, from which we extract the 95% CI around p. Individual points in the obtained distribution that are outside of the parameter ranges covered through simulations are set to the closest boundary value of the parameter range. For example, for f_{SS}, if for a point in the distribution we obtain a value lower than 0.05, we set the value of that point to 0.05; and if we obtain a value larger than 0.20, we set it to 0.20. We undertake this procedure for all parameters except for the time related ones, that is, infectious and incubation period as these depend on the time rescaling. The width of our 95% CIs is defined as the distance between the 2.5% and 97.5% percentile. With very large trees and the subtree picking and averaging procedure, we consistently use a quadratic weighted average of the individual CIs found for every subtree.

Assessment of 95% CI coverage and width

To assess this fast implementation of the parametric bootstrap, we used the test set of 10,000 simulations (and 100 simulations for comparison with BEAST2 95% CI). We measured the coverage being defined as the fraction of simulations where the true/target parameter values are inside the obtained 95% CI:

$$95\%\text{CIaccuracy} = \frac{\#\text{ true values inside 95\%CI}}{\#\text{ simulations}}$$

We applied the same criteria for BEAST2. For comparison of all methods, we excluded BDEI and BDSS simulations for which BEAST2 did not converge after 10 million steps. To draw BEAST2 CIs, we discarded the burn-in, that is, the first 10% of the MCMC, and calculated the CI on the remaining part of the chain. The CI width and coverage within the CIs obtained by NNs and BEAST2 are reported in **Supplementary Tab. 7**.

There exists a plethora of approaches for assessment of uncertainty and CI estimation. For example, (1) in a similar ABC context, the use of neighbouring trees (based on the Euclidean distance, not applicable to CBLV and questionable with SS) combined with a regression-based correction similar to that explained above[19,20]; (2) the (non-approximated) parametric bootstrap[59]; (3) the prediction of values from a distribution of trees reconstructed with Bayesian methods[10]; *etc*. We chose an approximation of the parametric bootstrap for its easy deployability, speed, coverage and width of produced CIs. The easy deployability comes from the fact that CIs are based on pre-calculated data stored in our CI set. The speed of the method comes from it not requiring simulations of new trees, and thus producing CIs within 2-4 seconds. The coverage and width are comparable to those of BEAST2 (**Supplementary Tab. 7**), a Bayesian method, intended to estimate the distribution of parameters and the uncertainty of inferences, with high computational cost.

MODEL ADEQUACY

A priori checks

We performed a sanity check using the SS of the test set simulations and the SS measured on the empirical HIV phylogeny. We reduced and centered the SS and performed a Principal Component Analysis (PCA) using the PCA function from the scikit-learn[50] package.

We highlighted the data point corresponding to the Zürich HIV MSM phylogeny in **Supplementary Fig. 8**, for each model (BD, BDEI and BDSS). Dissemblance between the simulations and the HIV phylogeny would be manifested by the fact that this data point lies outside the distribution corresponding to the simulations.

Furthermore, we performed an additional *a priori* check consisting in the study of all individual SS rather than dimensional reduction with PCA. For each SS, we checked whether the value for Zürich HIV MSM phylogeny lays between the minimum and maximum value of that SS in the test set of 10.000 trees. We reported the results in **Supplementary Fig. 8**.

A posteriori checks

We performed tests analogous to the *a priori* model adequacy checks. For both PCA and individual SS tests, instead of using the test set as representative of simulations, we simulated 10,000 additional simulations under the selected BDSS model. Parameter values were resampled from uniform distribution with boundaries given by the 95% CIs, and sampling probability fixed to presumed value of 0.25 (**Fig. 5, Supplementary Fig. 8**).

MODELS

The models we used for tree simulations are represented in the form of flow diagrams in **Fig. 1**. We simulated dated binary trees for (1) the training of NNs and (2) accuracy assessment of parameter estimation and model selection. We used the following three individual-based phylodynamic models:

Constant rate birth-death model with incomplete sampling

This model (BD[8,9], **Fig. 1 a**) contains three parameters and three compartments: infectious (I), removed with sampling (R) and removed unsampled (U) individuals. Infection takes place at rate β. Infectious individuals are removed with rate γ. Upon removal, an individual is sampled with probability s.

For simulations, we re-parameterized the model in terms of: basic reproduction number, R_0; infectious period, $1/\gamma$; sampling probability, s; and tree size, t. We then sampled the values for each simulation uniformly at random in the ranges given in **Supplementary Tab. 4**.

Birth-death model with exposed-infectious classes

This model (BDEI[10-12], **Fig. 1 b**) is a BD model extended through the presence of an exposed class. More specifically, this means that each infected individual starts as non-infectious (E) and becomes infectious (I) at incubation rate ε. BDEI model thus has four parameters (β, γ, ε and s) and four compartments (E, I, R and U).

For simulations, we re-parameterized the model similarly as described for BD, set the ε value via $1/\gamma$ and incubation ratio ($=\varepsilon/\gamma$). We sampled all parameters, including ε/γ, from a uniform distribution, just as with BD (**Supplementary Tab. 4**).

Birth-death model with superspreading

This model (BDSS[5,10,11], **Fig. 1 c**) accounts for heterogeneous infectious classes. Infected individuals belong to one of two infectious classes (I_S for superspreading and I_N for normal spreading) and can transmit the disease by giving birth to individuals of either class, with rates $\beta_{S,S}$ and $\beta_{S,N}$ for I_S transmitting to I_S and to I_N, respectively, and $\beta_{N,S}$ and $\beta_{N,N}$ for I_N transmitting to I_S and I_N, respectively. However, there is a restriction on parameter values: $\beta_{S,S} \times \beta_{N,N} = \beta_{S,N} \times \beta_{N,S}$. There are thus superspreading transmission rates $\beta_{S,.}$ and normal transmission rates $\beta_{N,.}$ that are $X_{SS} = \beta_{S,S} / \beta_{N,S} = \beta_{S,N} / \beta_{N,N}$ times higher for superspreading. At transmission, the probability of the recipient to be superspreading is $f_{ss} = \beta_{S,S} / \left(\beta_{S,S} + \beta_{S,N} \right)$, the fraction of superspreading individuals at equilibrium. We consider that both I_S and I_N populations are otherwise indistinguishable, that is, both populations share the same infectious period ($1/\gamma$)[5,10,11]. The model thus has six parameters, but only five need to be estimated to fully define the model[5,10]. For simulations, we chose parameters of epidemiological interest for re-parameterization: basic reproduction number R_0), infectious period $1/\gamma$, f_{SS}, X_{ss} and sampling probability *s*. In our simulations, we used uniform distributions for these 5 parameters, just as with BD and BDEI (**Supplementary Tab. 4**).

SIMULATIONS

For the parameters $R_{0,}$ $1/\gamma$, and *s*, that are common to all three birth-death models, the same value boundaries were used across all models (**Supplementary Tab. 4**). We considered two spans of tree size: 'small trees' with 50 to 199 tips and 'large trees' with 200 to 500 tips. We then sampled parameter values uniformly at random within these parameter boundaries with standard Latin-hypercube sampling[60] using PyDOE package. We created 3,990,000 parameter sets for training, 10,000 for validation and early stopping, another 10,000 for testing parameter inference and model selection (comparison with BEAST2 used a subset of 100, for computing time reasons), and 1,000,000 parameter sets for fast computation of CIs.

With these parameter sets, we simulated trees under each birth-death model using our implementation in Python of Gillespie algorithm[61], based on a standard forward simulator. Comparable accuracies (as in **Fig. 3** and **Supplementary Fig. 1**, both for BEAST2 and our methods) were reached on test simulations obtained with a well-established (but slower) simulator: TreeSim[4,5,7] (data not shown).

Each simulation started with one infectious individual (the class was chosen randomly under the BDSS model) and stopped when we obtained a tree with the given number of sampled individuals (tips). If the epidemic died away stochastically, that is, there was no more infectious tips left due to stochastic death before reaching the given tree size, we re-initialized the simulation up to 100 times. Only around 11% of simulations reached more than 2 iterations (20% for BDSS), and less than 0.5% reached more than 50 iterations for all models. If still no tree of given size was obtained after 100 iterations, we discarded the parameter set (less than 0.3% of all sets) and generated a new one to keep the desired number of simulations. This enabled us to maintain a nearly uniform coverage of parameter space, within selected parameter boundaries.

We simulated 100 'huge' trees for each model, with the same parameter values as for the 100 large trees used for testing (Fig. 3). These trees were simulated with treesimulator (https://pypi.org/project/treesimulator). The Snakemake[62] pipeline for huge trees simulation along with the simulated trees are available on GitHub.

METHOD COMPARISON

Parameter inference with BEAST2

To assess the accuracy of our methods, we compared it with a well-established Bayesian method, as implemented in BEAST2 (version 2.6.2). We used the BDSKY package[4] (version 1.4.5) to estimate the parameter values of BD simulations and the package bdmm[12,13] (version 1.0) to infer the parameter values of BDEI and BDSS. Furthermore, for the inference on BDSS simulations, instead of BEAST 2.6.2 we used the BEAST2 code up to the commit nr2311ba7, which includes important fixes to operators critical for our analyses. We set the Markov Chain Monte Carlo (MCMC) length to 5 million steps for the BD model, and to 10 million steps for the BDEI and BDSS models.

The sampling probability was fixed during the estimation. Since the BD, BDEI and BDSS models implemented in BEAST2 do not use the same parametrizations as our methods, we needed to apply parameter conversions for setting the priors for BEAST2 inference (**Supplementary Tab. 6**), and for translating the BEAST2 results back to parameterizations used in our methods, in order to enable proper comparison of the results. More specifically, the BEAST2 parameters can be converted to those used in our methods, that is, instead of infectious period and incubation period, BEAST2 uses the inverse of these, namely the infectious rate and incubation rate, respectively; instead of

superspreading transmission ratio and superspreading fraction at equilibrium, it uses individual sub-component parameters $R_{0,SS}$, $R_{0,SN}$, $R_{0,NS}$ and $R_{0,NN}$, which we will collectively refer to as "partial R_0". For BDSS, the BEAST2 prior was thus not the same as that of our simulations for BDSS (**Supplementary Tab. 4, 6**), since BEAST2 does not infer the same parameters. We used the range of all parameter values used in our simulations to set the boundaries of uniform prior distributions of parameters inferred by BEAST2. The initial values in the MCMC were set to the medians observed in the training set. During the inference, the parameter values were constrained in the same way as in the simulations, namely, we used the constraint $R_{0,SS} \times R_{0,NN} = R_{0,SN} \times R_{0,NS}$ (equivalent to $\beta_{S,S} \times \beta_{N,N} = \beta_{S,N} \times \beta_{N,S}$) in the BDSS model inference. Furthermore, the effective frequency of superspreading individuals (parameter called "geo-frequencies" in *bdmm*) was constrained to be between 5% and 20%. Due to the parameter conversions, and despite these constraints the inferred f_{ss} and X_{ss} can reach values outside the boundaries used for simulations, in which case we set them to the closest boundary for fair comparison with deep learning methods in **Fig. 3** (*e.g.*, if the median *a posteriori* f_{ss} was estimated to be larger than 0.20, it was set to 0.20 and if inferred f_{ss} was less than 0.05, it was set to 0.05). The goal of this correction was to avoid penalizing BEAST2 when it converged to local minima outside of the parameter boundaries used for simulations, which are implicitly known to NNs since they were trained on simulations with parameters within these boundaries.

After we obtained the parameters of interest from the original parameters estimated by BEAST2, we evaluated the Effective Sample Size (ESS) on all parameters. We reported the absolute percentage error of the median of *a posteriori* values, corresponding to all reported steps (reported steps being spaced by 1,000 actual MCMC steps) past the 10% burn-in. For simulations for which BEAST2 did not converge, we considered the median of the parameter distribution used for simulations (**Fig. 3, Supplementary Tab. 1-2, Supplementary Fig. 2**) or excluded them from the comparison (**Supplementary Tab. 1-2**, values reported in brackets, **Supplementary Tab. 5**).

For the HIV application, the prior of infectious period was set to [0.1, 30] years (uniform). All the other parameters had the same prior distributions as used in simulations and shown in **Supplementary Tab. 4, 6.**

Model selection with BEAST2

We performed model selection under BEAST2 using Akaike's information criterion through MCMC (AICM)[32,33]. The AICM is based on the following formula:

$$AICM = 2s_l^2 - 2l$$

where l and s_l^2 are the sample mean and variance of the posterior log-likelihoods. The AICM is an equivalent of AIC and the model with lowest AICM value is selected.

For 100 simulations obtained with each model (BD, BDEI and BDSS for large trees, BD and BDEI for small trees), we performed parameter estimation with BEAST2 under each model, computed AICM considering the whole MCMC, but excluding 10% burn-in (*i.e.*, 9,000 log-likelihood values for BDEI and BDSS considered in total, 4,500 for BD). The results of model selection are shown in **Supplementary Tab. 5**. The BDEI and BDSS simulations for which BEAST2 did not reach an ESS of 200 for all parameters were excluded from the computation of model selection accuracy for all methods.

Linear regression

For each model, linear regression was trained using reduced and centered summary statistics (using scikit-learn package, as with FFNN). Its bias and accuracy were assessed using the same criteria as for the NN approaches (**Supplementary Tab. 1-2, Supplementary Fig. 6**).

FFNN-CBLV

We trained an FFNN on CBLV representation. The FFNN architecture was close to the one described in ***Architecture*** with one extra hidden layer, so 5 layers in total, organized in a funnel shape with 128-64-32-16-8 neurons and 1 output layer of size 2-4 depending on the number of parameters to be estimated. The setting during the training and the sizes of training, validation and testing sets were the same as for the CNN-CBLV. Its bias and accuracy were assessed using the same criteria as for other NN approaches (**Supplementary Tab. 1-2, Supplementary Fig. 5**).

TreePar

We used TreePar[5] for MLE. With BD, we obtained results close to estimates under BEAST2, which is consistent with former studies. TreePar[5] uses an exact analytical formula of likelihood for BD and thus these (and BEAST2) results are theoretically optimal.

We also performed several trials to do parameter inference for the more complex models (*i.e.*, BDEI and BDSS), but in a large number of cases, we encountered numerical problems (*e.g.*, underflow or overflow issues), which resulted in infinite negative log-likelihood values, and eventually failed runs. When the calculations did not fail, we found that many estimations under BDSS and BDEI had lower likelihood than estimations performed with (nested) BD on the same input data. These numerical issues were confirmed by the authors of the TreePar package, with no solution available at the moment.

Null models

To assess how much information was learned on given problem, we compared FFNN-SS and CNN-CBLV to two null models.

The first null model was the FFNN trained for each model on 4,000,000 simulations using SS, but with randomly permuted target values (*i.e.*, the initial correspondence between the SS and underlying parameter values was lost, while the range of values was conserved). We then predicted parameters for 10,000 test simulations (100 for comparison with BEAST2) and measured the mean absolute relative error (MRE; **Supplementary Tab. 1**). In such a case, the FFNN always predicted values close to the value with the lowest value of the cost function (*e.g.*, 2.2 for parameter values uniformly sampled between 1 and 5). The MRE of this approach represents the lowest MRE that machine learning approaches can have in the absence of information, but the knowledge of the parameter distribution. This can be used to get an idea of how well the trained approaches perform and how much information regarding each parameter they can extract from the data.

The second null model was a set of random values sampled from the parameter ranges that were used for simulations (**Supplementary Tab. 4**). In this model, as opposed to the previous null model, there is no training phase and we do not learn the best compromise in the absence of information.

PERFORMANCE ASSESSMENT

Mean relative error MRE

To compare the accuracy of parameter estimation, we used 100 simulated trees per model. We computed the mean absolute relative error (MRE, **Fig. 3-4, Supplementary Tab. 1, Supplementary Fig. 2-4**) between (1) the true (or

target) parameter values and the predicted values for machine learning approaches; and (2) the true (or target) parameter values and the median *a posteriori* values obtained with BEAST2, which are more stable and accurate than maximum *a posteriori* values:

$$MRE = \frac{1}{100}\sum_{i=1}^{100}\frac{|predicted_i - target_i|}{target_i}.$$

We plotted individual absolute relative errors (RE) of predictions (**Fig. 3-4**, **Supplementary Tab. 1, Supplementary Fig. 2, 4**) for each simulation *i*, calculated as:

$$RE_i = \frac{|(predicted_i - target_i)|}{target_i}$$

Not being limited by the computational cost for machine learning approaches, we computed the same metric but on 10,000 simulations (**Supplementary Fig. 3, 5-6;** results from 1,000 simulations plotted in **Supplementary Fig. 7**).

We assessed the statistical significance of MRE differences using two-sided paired z-test. The two NN approaches were also compared using the same test, but no significant differences were found.

Mean relative bias MRB

To compare the bias in parameter estimation, we used 100 simulated trees per model. We computed the mean relative bias (MRB) between (1) the true (or target) parameter values and the predicted values for machine learning approaches; and (2) the true (or target) parameter values and the median *a posteriori* values obtained with BEAST2 (**Supplementary Tab. 2**):

$$MRB = \frac{1}{100}\sum_{i=1}^{100}\frac{(predicted_i - target_i)}{target_i}.$$

Comparison of likelihood values for sets of parameter estimates obtained with different methods

To assess the performance of different methods, we also studied the likelihood values of parameter estimates obtained with BEAST2, CNN-CBLV and FFNN-SS. For BD, we computed the likelihood using TreePar and compared it to the likelihood value of target parameter values (**Supplementary Tab. 3**).

As TreePar was problematic with BDEI and BDSS (see above), we tried to take on the same approach for BDEI and BDSS with BEAST2, but imposing a single MCMC step. Nevertheless, this did not yield sufficient results to perform

sound comparison, since for example with FFNN-SS predictions, the likelihood was obtained only for 57/100 parameter estimates for BDEI and 49/100 for BDSS. In the remaining cases, BEAST2 either failed to return consistent likelihood values, or was unable to calculate likelihood for the initial parameter values.

Model selection accuracy

We performed model selection with CNN-CBLV, FFNN-SS and BEAST2 on 100 simulations obtained with each model (10,000 for a sub-comparison of CNN-CBLV and FFNN-SS). Results are shown in **Supplementary Tab. 5** in the form of confusion matrices, where the columns represent the true/target classes, and the rows are the predicted classes. We then computed the accuracy of each method:

$$accuracy = \frac{\#\ true\ predictions}{\#\ total\ predictions}.$$

For BEAST2 model selection and large trees, the chain did not converge (displayed as "ESS<200" in **Supplementary Tab. 5**) for 24.3% simulations of large trees and 4.5% simulations of small trees. We did not consider these in accuracy measurements, for all the methods.

Comparison of time efficiency

For FFNN-SS and CNN-CBLV, we reported the average CPU time of encoding a tree (average over 10,000 trees), as reported by NextFlow workflow manager[56], a pipeline software that we used. The inference time itself was negligible.

For BEAST2, we reported the CPU time averaged over 100 analyses with BEAST2 as reported by NextFlow. For the analyses with BDEI and BDSS models, we reported the CPU time to process 10 million MCMC steps, and for the analyses with BD, we reported the CPU time to process 5 million MCMC steps. To account for convergence, we re-calculated the average CPU time considering only those analyses for which the chain converged and an ESS of 200 was reached across all inferred parameters.

The calculations were performed on a computational cluster with CentOS machines and Slurm workload manager. The machines had the following characteristics: 28 cores, 2.4Ghz, 128 GB of RAM. Each of our jobs (simulation of one tree, tree encoding, BEAST2 run, etc.) was performed requesting one CPU core. The neural network training was performed on a GPU cluster with Nvidia Titan X GPUs.

HIV DATASET

We used the original phylogenetic tree reconstructed by Rasmussen *et al.*[25] from 200 sequences corresponding to the largest cluster of HIV-infected men-having-sex-with-men (MSM) subpopulation in Zurich, collected as a part of the Swiss Cohort Study[24]. For details on tree reconstruction, please refer to their article.

PHYLODEEP SOFTWARE

FFNN-SS and CNN-CBLV parameter inference, model selection, 95% CI computation and *a priori* checks are implemented in the PhyloDeep software, which is available on GitHub (github.com/evolbioinfo/phylodeep), PyPi (pypi.org/project/phylodeep) and Docker Hub (hub.docker.com/r/evolbioinfo/phylodeep). It can be run as a command-line program, Python3 package and a Docker container. PhyloDeep covers the parameter subspace as described in **Supplementary Tab. 4**. The input is a dated phylogenetic tree with at least 50 tips and presumed sampling probability. The output is a PCA plot for *a priori* check, a csv file with all SS, and a csv file with probabilities of each model (for model selection) and point estimates and 95% CI values (for parameter inference with selected model). The installation details and usage examples are available as well on GitHub.

ADDITIONAL REFERENCES

58. Zarestkii, K. Reconstructing a tree from the distances between its leaves. (in Russian) *Uspehi Mathematicheskikh Nauk* **20,** 90-92 (1965).

59. Efron, *B. Breakthroughs In Statistics, Ch. Bootstrap Methods: Another Look at the Jackknife.* (Springer, New York, 1999).

60. McKay, M., Beckman, R., Conover, W. A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. *Technometrics* **21,** 239-245 (1979).

61. Gillespie, D.T. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry* **81,** 2340-2361 (1977).

62. Köster, J., Rahmann, S. Snakemake - a scalable bioinformatics workflow engine. Bioinformatics. **28**(19), 2520-2 (2012).