Deep Sequence Models for Emulation of Epidemic Models

Harsh Yadav[1,2] **Vishesh**[1,3] **Adway Mitra**[1]

[1]Centre of Excellence in Artificial Intelligence, IIT Kharagpur, India,
[2]Physics Dept., IIT Kharagpur, India,
[3]Mechanical Dept., IIT Kharagpur, India,
{harshyadav17,visheshkumar506}@iitkgp.ac.in, adway@cai.iitkgp.ac.in

Abstract

Epidemic models are used to simulate the spread of infectious diseases across a population. Such models are either differential equation-based compartmental models which represent the dynamics of statistics related to the population (eg. the number of Susceptible, Infected, Recovered individuals), or agent-based models which mimic the interactions among individuals. In this paper, we explore if it is possible to emulate the dynamics of these models, irrespective of their design, using deep neural networks. We study three tasks: i) given the outputs of an epidemic model till a time-step, can a neural network predict the model's output at the next time-step? ii) given the outputs of an epidemic model for the first few time-steps, can a neural network predict the outputs for the subsequent time-steps? iii) given an output sequence from an epidemic model, can we use a neural network to estimate its parameters? We focus on three epidemic models: the simple SIR model, the *Indian Covid-19 Super-Model*, and a simple agent-based version of SIR model which includes additional states. We explore and compare a variety of neural network architectures, and find that a neural network based on the LSTNet architecture is able to perform satisfactorily on all the three tasks, for all the three epidemic models mentioned above. For the parameter estimation task, we use Approximate Bayesian Computation.

1 INTRODUCTION

Many physical processes in the real world are studied using mathematical process models. These models try to replicate the physical processes under restricted and simplified settings. The aim of building process models is not to predict or forecast the physical processes, but to gain insights about how the processes may pan out under various scenarios. Such models have been developed in the domains of physics, chemistry, earth sciences, biology, epidemiology, hydrology and many more. These models are often based on a set of governing equations and parameters. The equations are either deterministic (based on differential equations) or stochastic (based on probability distributions). The equations are designed by experts, and parameter values chosen to fit the models on observations. Usually, process models are run many times with various initial conditions and parameter settings to form a robust idea about the processes under considerations.

Some problems with this approach are as follows: i) The governing equations may be hard to design if the physics of the system are not known perfectly, ii) Choosing optimal parameters can be very difficult, iii) running the models repeatedly may be time and resource intensive. Due to these problems, scientists often consider *emulator models*, which will be able to mimic the behaviors of these models without actually running them. Emulator models can usually predict the output of the process models given their initial conditions, without going through all of their internal states. This is achieved by training the emulator models with enough data generated by running the process models. The emulators are basically machine learning models that map the initial conditions to outcomes of the process models. Deep Learning, with its success in many tasks related to forecasting and new data generation, are increasingly being deployed for this purpose (such as [Kasim et al., 2020]).

Epidemiology is an area which assumed tremendous importance in 2020 during the Covid-19 global pandemic. Different research groups worked on epidemiological models to get an idea of how the pandemic may pan out under different scenarios related to pharmaceutical and non-pharmaceutical interventions (eg. physical distancing). As the best-known *compartmental models* proved to be inadequate to handle novel features of this pandemic (such as asymptomatic patients), new models were developed. Agent-based models

Accepted for the 37th Conference on Uncertainty in Artificial Intelligence (UAI 2021).

too were developed to mimic interactions between people in a society, which has a strong bearing on the spreading of the disease. Most of these models suffered from some or most of the issues mentioned above.

In this paper, we explore if deep neural models can be used as emulators of these epidemiological models - both compartmental and agent-based. We train these neural models with initial conditions of the pandemic (number of infected people in the first few days) as inputs, to predict the conditions after a fixed duration as outputs. For this purpose, we try out different neural architectures, especially those suitable for sequential data such as Recurrent Neural Networks (RNN) and Long Short Term Memory (LSTM) and their recent variants. In a related task, we use these emulator models along with *Approximate Bayesian Computations*, to estimate the parameters of the process models given the full time-series of the pandemic (i.e. number of daily infections, recoveries etc). Success of the neural models on these tasks indicates that even if appropriate governing equations or parameters of any epidemic model are not known, we can generate equivalent predictions using these neural models. To the best of our knowledge, this is the first-ever attempt to build neural emulators for epidemiological models. In doing so, we explore a couple of novel process models apart from the well-known SIR model - one compartmental model called Indian Covid-19 Supermodel and one agent-based model.

The paper is organized as follows. In Section 2 we review the literature related to epidemiological models, approximate Bayesian computation and deep models. In Section 3 we formally define the problem statement and three models which we aim to emulate. Section 4 details three neural network architectures, which are our emulators. Section 5 explains the emulation experiments and their results.

2 RELATED WORK

Epidemiological Models: Epidemiological models have long been used to study the spread of epidemics through populations. The most widely-used epidemiological models are the compartmental models as described in Brauer [2008]. These models keep track of various statistics related to the epidemic, and capture the temporal evolution of these statistics using differential equations. For the well-known SIR model, such statistics are the number of Susceptible (S), Infected (I) and Recovered (R) people. There are similar compartmental models, most notably SEIR. Such models have been used to study the recent Covid-19 pandemic for different countries (He et al. [2020]). Some researchers have also developed new versions of these models specially suited to the Covid-19 pandemic such as the *National Covid-19 Supermodel* of India [Agrawal et al., 2020] which can handle *asymptomatic* cases.

Another approach to epidemiological models are the agent-based models. These models try to simulate the behavior of individuals (agents) and their interactions, and are an useful tool in applied social science [Macal and North, 2009]. These are used in epidemiology as alternatives to, or in conjunction with compartmental model [Bobashev et al., 2007]. Their advantage is that they can capture realistic physical scenarios better than compartmental models, but are often computationally expensive and involve too many parameters. Yet, during the covid-19 crisis, many researchers have used these to study the pandemic spread and effects of intervention policies, such as Hoertel et al. [2020], Kerr et al. [2020].

Approximate Bayesian Computation: An important aspect in process-based simulation models is the selection of optimal parameters so that the results are realistic. Standard parameter estimation approaches like maximum-likelihood and expectation-maximization do not work here, since the processes are very complex and we cannot get a mathematical expression of the likelihood function. So we go for Approximate Bayesian Computation, where candidate parameter values are sampled from a prior distribution, used to carry out a simulation, and accepted if the outcome is close enough to the desired outcome [Beaumont et al., 2002]. This approach has been used in many scientific domains, including epidemiology [Minter and Retkute, 2019, Engblom et al., 2020]. However, a criticism of this approach is that it is too slow. Hence, there have been recent attempts to pose it as a classification and regression problem and solve using machine learning approaches [Lueckmann et al., 2019].

Deep Emulation Models: Beyond epidemiology, simulation models including equation-based (eg. compartmental) and agent-based models are used to study physical processes in various domains such as physics and climate sciences. However, many of these models are heavily dependent on governing equations (which may not be accurate enough), have too many parameters (which are difficult to optimize), and/or are computationally expensive. So there is a new trend of emulating these models with machine learning/deep learning models which may be agnostic to the governing equations or parameters, but can map input states to output states if trained appropriately. Such deep emulators for process simulation models have been developed for physics and climate sciences [Kerzendorf et al., 2020, Kasim et al., 2020, Manepalli et al., 2019]. Some initial steps at emulating SIR model using neural networks have also been taken [Davis et al., 2020]. But since epidemiological models generate sequential data (time-series statistics about the epidemics), it is important to consider deep models for sequential data.

Deep Sequence Models: Deep neural networks have been very successful in many domains in the past decade. They are often used on sequential data, for prediction and sequence-to-sequence mapping purposes in many domains, most notably natural language processing. The key chal-

lenge is that information from earlier parts of the sequence is often needed to predict the later parts. Recurrent Neural Networks (RNN), Long Short Term Memory (LSTM) and Gated Recurrent Units (GRU) have been used to propagate information across the sequence and update them as needed [Sutskever et al., 2014, Che et al., 2018, Chung et al., 2014, Dasgupta and Osogami, 2017, Laptev et al., 2017]. A concept that has significantly improved sequence models is *attention* [Vaswani et al., 2017], which identifies more salient parts of the sequence for more effective prediction, as shown by Song et al. [2018]. The problem becomes more difficult when the sequence is multi-dimensional, with interactions between the different dimensions. In such cases, a convolutional component is added too [Yang et al., 2015]. One recent model is the LST-Net [Lai et al., 2018], which combines convolutional and recurrent components to capture both short-term and local patterns and long-term trends, and it was subsequently augmented with attention [Shih et al., 2019]. Deep Models have been used in the domain of epidemic forecasting too, such as Wu et al. [2018]. More recently, Wu et al. [2020] used attention-based transformer architecture for influenza time-series forecasting, that first encodes the input sequence followed by decoding.

3 PROBLEM STATEMENT

Now, we discuss the settings of the epidemic, and the different epidemic models which we aim to emulate using Deep Neural Networks. Consider a society of N people. A small subset of these people is initially infected with a communicable disease, at $t = 0$. Everyone else is susceptible to the disease, i.e. they can get infected by it from people who are already infected. The total population is divided into 3 or more *compartments*, where each compartment contains people in the same condition (susceptible, infected, recovered etc) with respect to the epidemic. The compartments vary from one model to another. The number of people in each compartment in each day is tracked. The epidemic dynamics, i.e. the process by which people move from one compartment to another, is represented in different ways in different models.

3.1 SIR (COMPARTMENTAL MODEL)

The simplest and most well-known compartmental epidemic model is SIR, which has three compartments: Susceptible (S), Infected (I) and Recovered (R). It is assumed that initially everyone is susceptible, some of them may get infected subsequently after coming in contact with others who are already infected, and then recover after some days. Anyone who has recovered once will not get infected again. The process of infection is assumed to happen by physical contact, but this model does not represent it explicitly. Instead, it is assumed that everyone interacts with everyone else, and a fraction β of all susceptible-infected interactions results in spreading of infection. Similarly, every day a fraction γ of the infected persons are assumed to recover. These dynamics are represented by the following *governing equations*:

$$\frac{dS}{dt} = -\beta\frac{IS}{N};\ \frac{dI}{dt} = \beta\frac{IS}{N} - \gamma I;\ \frac{dR}{dt} = \gamma I \tag{1}$$

Here, the ratio $R_0 = \frac{\beta}{\gamma}$ is called the *basic reproductive number*, which indicates the number of new infections per recovery. If $R_0 > 1$ then the pandemic spreads through the population, but if $R_0 < 1$ then it dies down. A discrete-time equivalent of the process can be made accordingly. There are other variants of this model too, which includes more compartments, such as SEIR model with an *exposed (E)* compartment for people who have contacted the disease but not yet ill.

3.2 INDIAN COVID-19 SUPERMODEL (COMPARTMENTAL MODEL)

In view of the Covid-19 pandemic, a number of models were developed specifically for this disease with compartments developed according to its unique characteristics. One of them is the so-called *National Covid-19 Supermodel* of India. This model separately handles *sympotamtic* and *asymptomatic* persons. Accordingly, S, I, R are separated into $(S_A, S_I), (A, I), (R_A, R_I)$ respectively, and a state D to account for deaths. Their dynamics are represented by the following governing equations:

$$\begin{aligned} \frac{dS_A}{dt} &= -\beta S_A(A+I);\ \frac{dS_I}{dt} = -\beta S_I(A+I) \\ \frac{dA}{dt} &= \beta S_A(A+I) - \gamma A;\ \frac{dI}{dt} = \beta S_I(A+I) - \gamma I \\ \frac{dR_A}{dt} &= \gamma A;\ \frac{dR_I}{dt} = \gamma I;\ \frac{dD}{dt} = \eta I \end{aligned} \tag{2}$$

Clearly, the parameters of this model are the infection rate β, recovery rate γ and death rate η.

3.3 SIARQ (AGENT-BASED MODEL)

A different approach of epidemic modeling is agent-based, where each person is represented by an agent, and their movements and interactions are simulated to provide a basis for infection spreading. Unlike the deterministic models above, these models are stochastic. In the context of the Covid-19 pandemic, we suggest a simple agent-based model that includes *asymptatic infected (A)* and *quarantined (Q)* as additional states.

In this model, we consider N individuals separately, and keep track of their states. Every day, each individual interacts with a random subset ρ of individuals. During such an interaction, if one person is susceptible (S) and another is infected (I), then the first person gets infected with probability

β. But if a susceptible person interacts with a person who is asymptomatic (A), then the first person can get interact with probability $\alpha\beta$ where $0 \leq \alpha \leq 1$. On getting infected, a person is initially asymptomatic (A) for a few days (geometric distribution with parameter κ), and then becomes symptomatic (I). Every day, a random and increasing subset of people is tested, and those among this subset who are in states A or I, get quarantined (Q) who cannot infect anyone else. Those in states A, I or Q move to state R after some days (geometric distribution with parameter γ), and play no further role in the process.

3.4 EMULATION AND PARAMETER ESTIMATION

Each of the above models can be run for a number of days to generate a sequence of $\{S,I,R\}$ values. In other words, for any day t we get $(S(t),I(t),R(t))$. Each of these models provides a way to generate $(S(t+1),I(t+1),R(t+1))$ from $(S(t),I(t),R(t))$. The aim of this paper is to build *emulator models*, which can do the same without knowing the governing equations of the models. We attempt to solve the following three tasks related to this:

1. **One-step Prediction**: Given sequential data till any day t, i.e. $((S(1),I(1),R(1)),\ldots,(S(t),I(t),R(t)))$, predict $((S(t+1),I(t+1),R(t+1)))$.
2. **Sequential Prediction** Given sequential data till a fixed day τ, i.e. $((S(1),I(1),R(1)),\ldots,(S(\tau),I(\tau),R(\tau)))$, predict the rest of the sequence $((S(\tau+1),I(\tau+1),R(\tau+1)),\ldots,(S(T),I(T),R(T)))$. This can be done by solving the one-step prediction problem, and using the predicted values as part of the sequence for further predictions.
3. **Parameter Estimation** Given an entire sequence $((S(1),I(1),R(1)),\ldots,(S(T),I(T),R(T)))$ from one of the above models, predict the values of the concerned model's parameters, such as β, γ, α.

3.5 DATASET

Clearly, we need data generated by the epidemic models to train the emulator models. For this purpose, we generate 1000 sequences, each of length 100 (i.e. $T = 100$) for each of the 3 models described above. We take the population size $N = 100000$, with 10 of them initially infected. For each run, we randomly choose the parameter values (β,γ), but the basic reproductive number R_0 is maintained at 2. In case of the SIARQ agent-based model, α is chosen uniformly between 0 and 1. The geometric distribution parameters κ, γ are chosen such that a person stays in A state for 5-10 days, and in I state for 7-14 days. The number of persons tested per day starts with 5000, and increases in arithmetic progression with slope 1000. In case of the national supermodel, we take $S(t) = I_S(t) + I_A(t)$, $I(t) = I(t) + A(t)$ and $R(t) = R_A(t) + R_I(t)$. In case of the SIARQ Agent-based model, we take $I(t) = I(t) + A(t) + Q(t)$. For developing the emulator models, about 80% of these sequences are used for training, and the rest for testing.

4 DEEP EMULATOR MODELS

In this section we discuss about the deep neural networks used for the first two tasks defined above. For this purpose, we examine three models: Extreme LSTM Encoder-Decoder model ([Laptev et al., 2017]), a hybrid CNN-LSTM model based on LSTNet ([Lai et al., 2018]) and a Transformer-based model based on [Vaswani et al., 2017].

4.1 EXTREME LSTM AUTOENCODER

This network [Laptev et al., 2017] is a combination of two models, one is the encoder-decoder, which is a very common architecture for sequence-to-sequence mapping, such as neural translation of a sentence in one language to another. Another part is the forecasting model which uses LSTM layer at its core and predicts the final output. The architecture is illustrated in Figure 1).

This model is made up of two networks, encoder and decoder. The encoder extracts the long-term temporal features from the input time-series using stacked LSTM units and creates the code vector. This code vector layer is the input to the decoder network. Decoder is the exact replica of encoder layers in opposite order, i.e the first layer of LSTM in encoder is the last layer of decoder, whereas the last layer of encoder is the first layer of decoder. This model is trained with the input data and acts as a feature extraction unit for our architecture. The output of the encoder is extracted as the embedding vector and it is further concatenated with other features. This concatenated layer is the input layer to the LSTM forecaster, which is a simple model with LSTM layers stacked together followed by dense layers to predict the next time step.

For training, the input at time t contains the (S,I,R) vectors along with the static parameters, *beta* and *gamma*, for $m = 5$ previous time-steps $(t-m+1,\ldots,t)$, which is fed into the encoder network. The output is $(S(t+1),I(t+1),R(t+1))$. The input is of shape *(batch, m, f)*, where f is the total number of temporal and static features, i.e. 5 (S,I,R, β, γ) for SIR dataset and similarly for SAIRQ and Covid-19 Supermodel datasets. We have used the *Adam* optimizer along with the default activation function for LSTM layer, i.e. *tanh*. For regularization we have applied dropout layers after each layer. Mean Squared Error (*mae*) is used as a loss function while training along with the back-propagation.

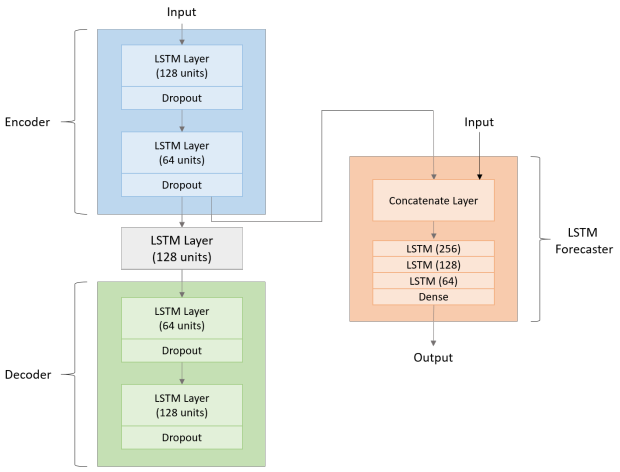

Figure 1: Architecture of Extreme LSTM encoder-decoder

4.2 TRANSFORMER-BASED ENCODER-DECODER MODEL

Our transformer-based time-series forecasting model is developed on the top of the original Transformer architecture [Vaswani et al., 2017]. This model also includes the encoder and decoder, but with the addition of self attention layer. The architecture is explained in Figure2.

Input: Input embedding is created using 1D convolutional layer. Further positional encoding with *sine* and *cosine* functions is used to encode sequential information from the input data. An element-wise addition of the input embedding and positional encoding is done and this is the layer which is connected to the first encoder layer. Similarly, another input layer is constructed for the decoder input.

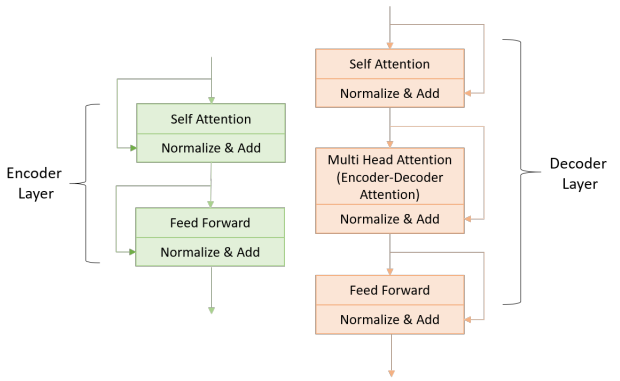

Figure 2: Encoder (left) and Decoder (right) layers of Transformer

The encoder network comprises of two encoding layers. Each layer consists of a self-attention layer followed by a normalisation layer. A feed-forward layer follows it sequentially which is further normalised and element-wise addition is done. This entire setup (Figure 2) is called encoder layer and in our model, two encoder layers are sequentially added to form the encoder. The output vector of the encoder is one of the inputs to the decoder.

The decoder network is given two inputs, one is the input layer (input embedding + positional encoding) and another is the output vector from the encoder. The decoder is composed of two decoder layers. Each decoder layer has a self-attention sublayer followed by multi-head attention sublayer and a feed-forward sublayer. Each sublayer is followed by a normalisation layer. Multi-head attention also termed as encoder-decoder attention is fed with the output vector from the encoder along with the last sub layer (output of self attention). This helps to apply the attention mechanism over the encoder output with the decoder sub-layer. The output of the decoder is finally mapped with the output layer, i.e. the feed forward layers to predict the the target sequence of the time series (Figure 3).

For training, suppose input Data is a sequence x_1, x_2,..., x_N. Input to the encoder is a sliding window of *N-m* data points, i.e. $\{x_1, x_2, \ldots, x_{N-m}\}$. Input to the Decoder is a sliding window of the next *m* data points, i.e. x_{N-m},..., x_N. We train the model to predict 1 future data point using 5 prior data points, i.e. given the encoder input (x_1, x_2,..., x_4) and decoder input (x_4,x_5) model aims to output (x_6). These inputs include the static features β and γ along with the temporal features. All the LSTM layers have *tanh* as the activation function. We have used the *Adam* optimizer along with the *mae* as the loss function. A batchsize of 128 is used for the training of the model. Dropout layers have been used as a regularisation technique.

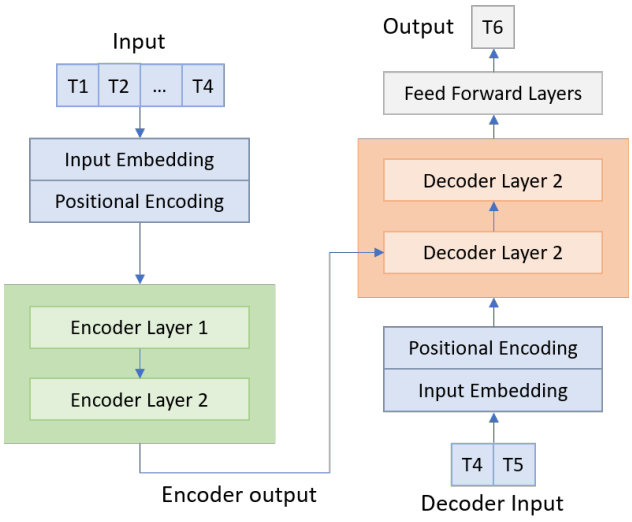

Figure 3: Architecture of Transformer

4.3 LSTNET-BASED MODEL

This network is basically divided into three building blocks, convolutional component, recurrent component, and a feed forward component (Figure 4).

The first layer of the network is the *Convolutional Component*. With the help of convolutional layer, the model aims to extract the local dependencies between the variables, which is of great importance for our dataset. Alongside, it also helps in the extraction of short-term pattern in the time-series. The output of the convolutional component is simultaneously fed into parallel layers of the *recurrent component* comprising of LSTM layers. One recurrent layer is a LSTM layer which ensures the proper mapping of the features. The other recurrent layer comprises of the LSTM layer with skip links between these layers. It is specifically designed to learn the long term dependencies along with the periodic patterns. Alternatively, we can also use the attention mechanism [Shih et al.,2019] to avoid tuning of hyper-parameters of the skip links. This approach produces better results for our experiments (shown below). The output of the last layer is the prediction. We name these two variants of LSTNet as follows: *LSTNet-S* for the skip RNN component and *LSTNet-A* for the attention mechanism instead of skip links. We use a dense layer to combine these two recurrent layers and produce the final output.

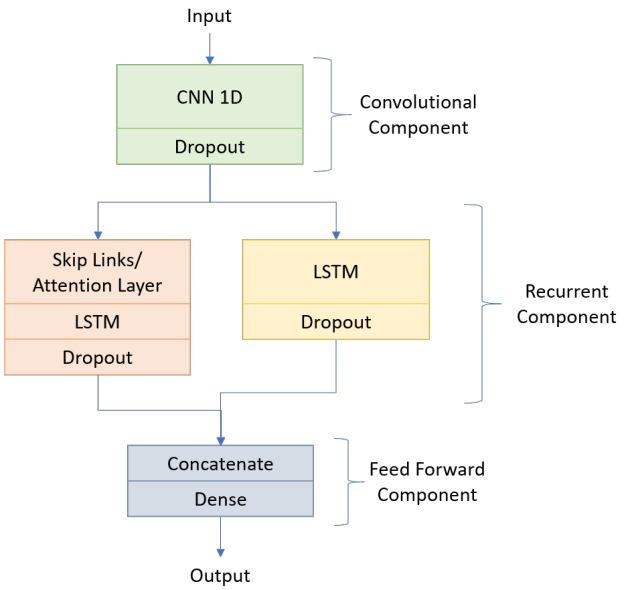

Figure 4: Architecture of LSTNet

For training, the input is a sliding window of n timesteps, while the output is a sliding window of m timesteps. In this study, we have taken $n = 5$ and $m = 1$, i.e. we are using 5 previous data points $(x_t, x_{t-1}, \ldots, x_{t-4})$ to predict next data points (x_{t+1}) where $x(t) = (S(t), I(t), R(t))$, so the input is of shape, (batch, n=5, f), where f is the features (temporal and static) and the output is of shape, (batch, m=1, f'), where f' is temporal features only, i.e. *S, I, R* in case of SIR dataset. In a typical training setup, *relu* has been used as an activation function for LSTM layers instead of the default activation function, *tanh*. We have used *Adam* optimizer along with dropout as a regularization technique. *mae* has been used as a loss function.

5 EXPERIMENTATION

In this study, we conducted three experiments: one-step prediction, sequential prediction and parameter estimation. All these experiments are explained in detail below. For each of these experiments, we compared the three models mentioned above against some baseline models: (RNN-GRU, LSTM-based sequence-to-sequence (seq2seq) [Sutskever et al., 2014], and sequence-to-sequence with attention (seq2seq-A). We are using Root Relative Squared Error as loss functions, as follows:

$$RRSE = \frac{\sqrt{\sum_{(i,t)\in\Omega_{test}} (Y_{it} - \hat{Y}_{it})^2}}{\sqrt{\sum_{(i,t)\in\Omega_{test}} (Y_{it} - mean(Y))^2}}$$

In Table 1 and Table 2, *C19* refers to *Indian COVID-19 Supermodel* and *xLSTM* refers to Extreme LSTM Autoencoder.

Table 1: One Step Prediction results; RRSE

Models	SIR	SIARQ	C19
LSTNet-S	0.0115	0.0366	0.0161
LSTNet-A	**0.0050**	**0.0317**	**0.0094**
Transformer	0.0671	0.1046	0.1278
xLSTM	0.0864	0.1617	0.0570
GRU-RNN	0.0064	0.0507	0.0094
seq2seq-A	0.3581	0.6582	0.5560
seq2seq	0.5504	1.3756	0.4193

Table 2: Sequential Prediction results; RRSE

Models	SIR	SIARQ	C19*
LSTNet-S	0.0765	0.1433	0.0728
LSTNet-A	**0.0376**	**0.0958**	**0.0485**
Transformer	0.0671	0.1452	0.2570
GRU-RNN	0.6476	1.7558	5.7702
seq2seq-A	0.4919	1.5972	0.9420
seq2seq	0.5850	1.7558	0.5929

5.1 ONE STEP PREDICTION

In this experiment, we tested whether our Transformer-based model, LSTNet Model and Extreme LSTM Autoen-

coders could predict one day ahead from five days of historical data points, for all datasets from the three models i.e SIR (Compartmental Model), SIARQ (Agent Based Models) and COVID-19 Supermodel, as discussed in Sec 3.5. For evaluation, the trained models performed one-step ahead prediction for each state using the testing data set. Root Relative Squared Error (RRSE) values were calculated for each state.

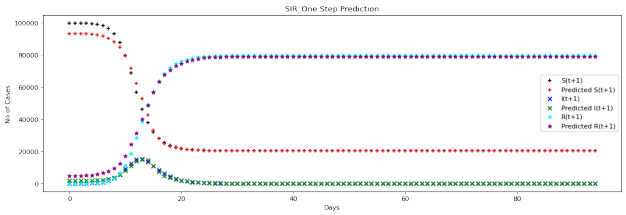

Figure 5: One Step Prediction on SIR dataset; Extreme LSTM Autoencoder (xLSTM)

Table 1 summarizes the results of One Step Prediction for all models. LSTNet-A performance is the best among all models used in this study for all 3 datasets. The strong neural baseline model, GRU-RNN share the almost same result with LSTNet-A in One-Step Prediction for all datasets. LSTNet-S also provide the state-of-the-art results for all datasets. In case of Transfomer, the results are good compared to the baseline models, seq2seq-A and seq2seq on the SIR dataset, but GRU-RNN performs better than Transformer and x-LSTM for One Step Prediction method for all datasets.

5.2 SEQUENTIAL PREDICTION

This experiment introduces a method to generate entire time series, i.e. 100 time-steps using Transformer Based Model, and LSTNet Model with only the first five days of historical data points, for all datasets i.e. SIR (Compartmental Model), SIARQ (Agent-Based Models) and COVID-19 Supermodel.

We hypothesized that given the first five days of data for each state the whole time series is predicted and in order to do so, we are using the predicted values from the one-step prediction experiment. Using the data at five time steps t_1, t_2, t_3, t_4, t_5, we can predict the data at t_6 time. Now, in order to predict t_7 time step, we use this predicted value for t_6 (from the earlier iteration), along with t_2, t_3, t_4, t_5 which we already had. We run this process sequentially, making one-step predictions and using these predicted values as inputs, to predict the entire sequence. This forecasted time series is compared with the actual time series to calculate the RRSE error.

Table 2 summarizes the performances of all models for the sequential prediction task. Once again, LSTNet-A produces the best results for all the datasets. LSTNet-S and Transformer models share the same results for SIR and SIARQ datasets but LSTNet-S performs 71.6% better than Transformer in case of Covid-19 Supermodel dataset because LSTNet-S becomes more effective as the number of static and temporal features increase. All the baseline models - GRU-RNN, seq2seq and seq2seq-A are outperformed by the models discussed in Section 4.

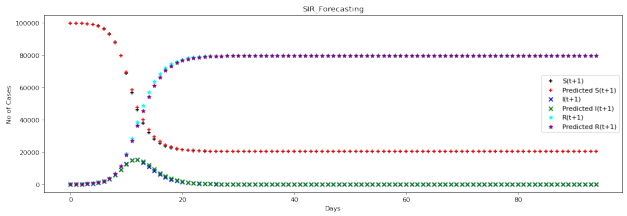

Figure 6: Sequential Prediction on SIR dataset using LSTNet-A

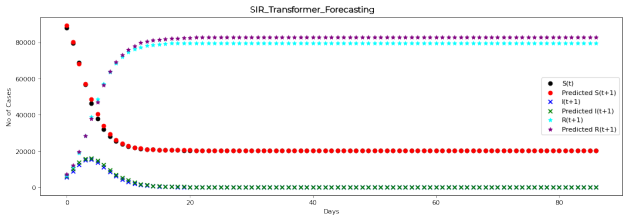

Figure 7: Sequential Prediction on SIR dataset using Transformer

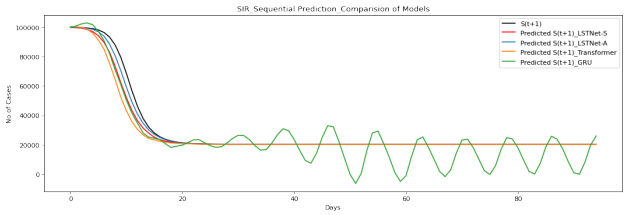

Figure 8: Sequential Prediction of variable *S* using different models

5.3 PARAMETER ESTIMATION

Given the dynamics of epidemics, we may want to fit a model, including the parameters values such as *Transmission Rate (beta)*, *Recovery Rate (gamma)* or the *Death Rate (mu)*. Doing so can enable to us to predict the epidemic in the near future.

5.3.1 Regression by Neural Networks

The parameter estimation task can be posed as an inverse regression problem, where we take a sequence as input and map it to the model parameter values which may have been

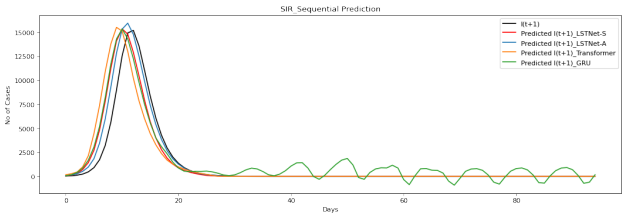

Figure 9: Sequential Prediction of variable I using different models

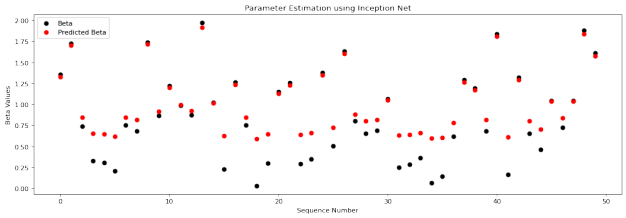

Figure 10: Parameter Estimation using Inception Net for beta

used to generate the sequence. We explored deep learning models for this regression, such as the basic LSTM model, Shared and Non-Shared Regression model (SNR) [Reyes, 2019] and Inception Net model [Ismail Fawaz, 2020] with a Dense Layer at the output. Input Vector to the Inception Net Model and LSTM Model is the multi-variate time-series of $((S(1),I(1),R(1)),\ldots,(S(T),I(T),R(T)))$. These models are trained separately for the SIR, SIARQ and C19 datasets, with the training sequences as input and corresponding parameter values as output. As shown in the Figure 10, the results are not satisfactory.

5.3.2 ABC Rejection Algorithm

Due to the failure of parameter estimation by direct regression, we now explore Approximate Bayesian Computation (ABC). We consider an iterative method, the ABC Rejection Algorithm (Algorithm 1), to determine the parameters that outline the whole dynamics of infectious disease. The aim of this method is to search the parameter space for suitable values.

We initially choose random values of the parameters from a Gaussian distribution and generate the sequence using them as inputs to the pre-trained LSTNet model. Next we compare the generated sequence with the actual sequence. If the RRSE error between the generated and actual sequence is within a user-defined threshold range, we store the parameter values along with the RRSE error for further comparison. Once we have stored sufficient number of parameter values, we continue the search around these values i.e. choose a stored parameter value from the list, and perturb them by adding some noise to get new candidate values. The probability of choosing any stored parameter value is inversely

Algorithm 1 ABC Rejection Algorithm

INPUT: A sequence X_0 of (S,I,R) of length T
Values = [], $j = (anynumber)$,$k,i = 0$
while len(Values) < j **do**
  Random selection of (β,γ) from Gaussian prior
  Generate a sequence X by trained LSTNet Model using these values
  if $threshold > RRSE(X,X_0)$ **then**
    append $(\beta,\gamma,RRSE)$ in Values list
  end if
end while

while $k < 20$ **do**
  while $i < 20$ **do**
    Choose candidate (β,γ) from *Values* list with probability inversely related to *RRSE*
    Add noise around the selected (β,γ)
    Follow Step $5,6,7$
    $i = i+1$
  end while
  $threshold = threshold - x$
  $k = k+1$
end while
OUTPUT: (β,γ) with least *RRSE* in *Values* list

proportional to the corresponding errors. Once again, we generate a sequence using the same trained model with the perturbed parameters as input. The generated sequence is compared with the actual sequence using RRSE. The threshold on RRSE for acceptance of parameter values are made progressively stricter with each iteration. After we have stored enough parameter values, we finally choose the one with least RRSE, i.e. which can generate a sequence that best resembles the input sequence.

6 CONCLUSION

This paper demonstrated that deep models can emulate epidemiological models - both compartmental and agent-based, without knowing either governing equations or parameters. In other words, deep sequential models can learn the dynamics of epidemics from observed data, and use them to make forecasts about pandemics which will be at least as accurate as different epidemic models, even without knowing governing equations. This work can be built upon to develop more advanced neural epidemic models, which may be able to represent epidemic dynamics using a more detailed representation than (S,I,R)-statistics, including spatial locations, social networks and various health attributes of individuals in the population.

References

Manindra Agrawal, Madhuri Kanitkar, and M Vidyasagar. Modeling the spread of sars-cov-2 pandemic—impact of lockdowns and interventions. *Indian Journal of Medical Research*, 2020.

Mark A Beaumont, Wenyang Zhang, and David J Balding. Approximate bayesian computation in population genetics. *Genetics*, 162(4):2025–2035, 2002.

Georgiy V Bobashev, D Michael Goedecke, Feng Yu, and Joshua M Epstein. A hybrid epidemic model: combining the advantages of agent-based and equation-based approaches. In *2007 Winter Simulation Conference*, pages 1532–1537. IEEE, 2007.

Fred Brauer. Compartmental models in epidemiology. In *Mathematical epidemiology*, pages 19–79. Springer, 2008.

Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, 8(1):1–12, 2018.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Sakyasingha Dasgupta and Takayuki Osogami. Nonlinear dynamic boltzmann machines for time-series prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Christopher N Davis, T Deirdre Hollingsworth, Quentin Caudron, and Michael A Irvine. The use of mixture density networks in the emulation of complex epidemiological individual-based models. *PLoS computational biology*, 16(3):e1006869, 2020.

Stefan Engblom, Robin Eriksson, and Stefan Widgren. Bayesian epidemiological modeling over high-resolution network data. *Epidemics*, 32:100399, 2020.

Shaobo He, Yuexi Peng, and Kehui Sun. Seir modeling of the covid-19 and its dynamics. *Nonlinear Dynamics*, 101 (3):1667–1680, 2020.

Nicolas Hoertel, Martin Blachier, Carlos Blanco, Mark Olfson, Marc Massetti, Frederic Limosin, and Henri Leleu. Facing the covid-19 epidemic in nyc: a stochastic agent-based model of various intervention strategies. *MedRxiv*, 2020.

Lucas B. Forestier G Ismail Fawaz, H. Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery volume*, 34(8):1936–1962, 2020.

MF Kasim, D Watson-Parris, L Deaconu, S Oliver, P Hatfield, DH Froula, G Gregori, M Jarvis, S Khatiwala, J Korenaga, et al. Building high accuracy emulators for scientific simulations with deep neural architecture search. *arXiv e-prints*, pages arXiv–2001, 2020.

Cliff C Kerr, Robyn M Stuart, Dina Mistry, Romesh G Abeysuriya, Gregory Hart, Katherine Rosenfeld, Prashanth Selvaraj, Rafael C Nunez, Brittany Hagedorn, Lauren George, et al. Covasim: an agent-based model of covid-19 dynamics and interventions. *medRxiv*, 2020.

Wolfgang E Kerzendorf, Christian Vogl, Johannes Buchner, Gabriella Contardo, Marc Williamson, and Patrick van der Smagt. Dalek–a deep-learning emulator for tardis. *arXiv preprint arXiv:2007.01868*, 2020.

Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 95–104, 2018.

Nikolay Laptev, Jason Yosinski, Li Erran Li, and Slawek Smyl. Time-series extreme event forecasting with neural networks at uber. In *International conference on machine learning*, volume 34, pages 1–5, 2017.

Jan-Matthis Lueckmann, Giacomo Bassetto, Theofanis Karaletsos, and Jakob H Macke. Likelihood-free inference with emulator networks. In *Symposium on Advances in Approximate Bayesian Inference*, pages 32–53. PMLR, 2019.

Charles M Macal and Michael J North. Agent-based modeling and simulation. In *Proceedings of the 2009 Winter Simulation Conference (WSC)*, pages 86–98. IEEE, 2009.

A Manepalli, A Albert, A Rhoades, D Feldman, and M Prabhat. Emulating numeric hydroclimate models with physics-informed conditional generative adversarial networks. *Environmetrics*, 2019.

Amanda Minter and Renata Retkute. Approximate bayesian computation for infectious disease modelling. *Epidemics*, 29:100368, 2019.

Ventura Reyes. Performing multi-target regression via a parameter sharing-based deep network. *International Journal of Neural Systems*, 29(9):1950014, 2019.

Shun-Yao Shih, Fan-Keng Sun, and Hung-yi Lee. Temporal pattern attention for multivariate time series forecasting. *Machine Learning*, 108(8):1421–1441, 2019.

Huan Song, Deepta Rajan, Jayaraman Thiagarajan, and Andreas Spanias. Attend and diagnose: Clinical time series analysis using attention models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *arXiv preprint arXiv:1409.3215*, 2014.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

Neo Wu, Bradley Green, Xue Ben, and Shawn O'Banion. Deep transformer models for time series forecasting: The influenza prevalence case. *arXiv preprint arXiv:2001.08317*, 2020.

Yuexin Wu, Yiming Yang, Hiroshi Nishiura, and Masaya Saitoh. Deep learning for epidemiological predictions. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1085–1088, 2018.

Jianbo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiaoli Li, and Shonali Krishnaswamy. Deep convolutional neural networks on multichannel time series for human activity recognition. In *Ijcai*, volume 15, pages 3995–4001. Buenos Aires, Argentina, 2015.