IJETST- Vol.||11||Issue||02||Pages 8064-8075||February||ISSN 2348-9480 2024

Open access Journal International Journal of Emerging Trends in Science and Technology

DOI: http://dx.doi.org/10.18535/ijetst/v2024.03

A Pandemic Predictive Model with Convolutional Neural Networks and Deep Reinforcement Learning using Simulated Partial Differential Equations Data

Authors

Sai Nethra Betgeri, Ph.D.[1], Shashank Reddy Vadyala[2]

[1]Assistant Professor, Department of Computer Science and Engineering, University of Louisville, Louisville, Kentucky, USA

[2]Department of Computational Analysis and Modeling, Louisiana Tech University, Ruston, Louisiana, United States

Abstract

Detecting the spread of pandemics will greatly reduce human and economic loss. Existing Epidemiological models used for coronavirus disease 2019 (COVID-19) prediction models are too slow and fail to capture epidemic development thoroughly. This research presents a Physics-based Machine Learning Architecture (PMLA) to improve the processing speed and accuracy of epidemic forecasting governed by susceptible–exposed–infected–recovered–deceased (SEIRD) model equations. The dynamics of the epidemic were extracted using Convolutional Neural Networks (CNN) and Deep Reinforcement Learning (Deep RL) from data simulated with Partial Differential Equations (PDEs). The PMLA accuracy is measured using mean squared error. The PMLA prediction model enhances the ability of health authorities to predict the spread of COVID-19 in real time efficiently and effectively.

Keywords: *COVID-19, Convolutional Neural Networks, Deep Reinforcement Learning, Partial Differential Equations, Machine learning, Finite Element Method.*

1.Introduction

The COVID-19 epidemic is undoubtedly the most severe public health threat since the Spanish flu outbreaks of 1918 and 1919[1]. Following a dramatic increase in COVID-19 infections, many nations, including the United States of America (USA) and India, declared states of emergency. The World Health Organization (WHO) said the outbreak has affected 25,227,970 people and killed 278,751 people in India as of May 17, 2021, causing hospitals in India to overflow. COVID-19 has created mayhem in the financial sector, resulting in the S&P 500's worst trading day since 1945(Bhadra et al., 2021). On December 8, 2019, COVID-19, caused by the Severe Acute Respiratory Syndrome COVID-19, was discovered in Wuhan, China ("Clinical characteristics of refractory COVID-19 pneumonia in Wuhan, China | Clinical Infectious Diseases | Oxford Academic," n.d.). The majority of the first patients were exposed to the nearby Huanan South China seafood market, which offers several wild animals, implying that the zoonotic coronavirus breached the animal-human boundary at this wet market. COVID-19, namely the Severe Acute Respiratory Syndrome and the Middle East

Respiratory Syndrome (MERSV), have caused a couple of major epidemics in the last twenty years[2].

COVID-19 drastic impact on our social lives and the economy has piqued scientists' interest in this novel virus. Multiple significant concerns regarding the pandemic remain unanswered at this period[3]. Aside from pathology, microbiology, and bioinformatics, the COVID-19 epidemic has sparked interest in epidemiology and statistics. Time series analysis, machine learning models, and forecasting models are of particular interest [4-12]. Critical risk assessment and coordination countermeasures should be taken to aid an accurate forecast of future events. Forecasting COVID-19 is critical for understanding the estimation of virus transmission characteristics such as the basic reproduction number, incubation period, and infectious period[13-26]. In a real scenario, those parameters are not easy to estimate. Accurate physics-based machine learning models for forecasting COVID-19 require solving complex PDEs.

PDEs have played a critical role in providing detailed and reliable models for various scientific phenomena, especially in Physics-based machine learning and engineering[27-30]. PDEs control physical phenomena such as the Navier–Stokes equation in fluid, aerodynamics, Fourier's heat conduction equation, and Schrödinger's equation in quantum mechanics. These models were first discovered using skills in philosophy, statistical modeling, and observational evidence. Equations that seem to model seemingly disparate physical processes are identical in that they are made up of specific, commonly important mathematical components. Physical behaviors placed on the observable data by various parts of the model may aid experts in identifying the model using prior theory and data information [31]. We can now view massive volumes of data from tests and simulations thanks to recent advances in data acquisition, storage, and computing tools. This makes it possible to derive information from raw data using data-driven approaches. Physics Informed Neural Networks (PINNs), as shown in Figure.1, have revolutionized many areas, such as machine vision, in recent decades [12, 32-34]. When extended to scientific evidence, they can help solve and explain physical problems.

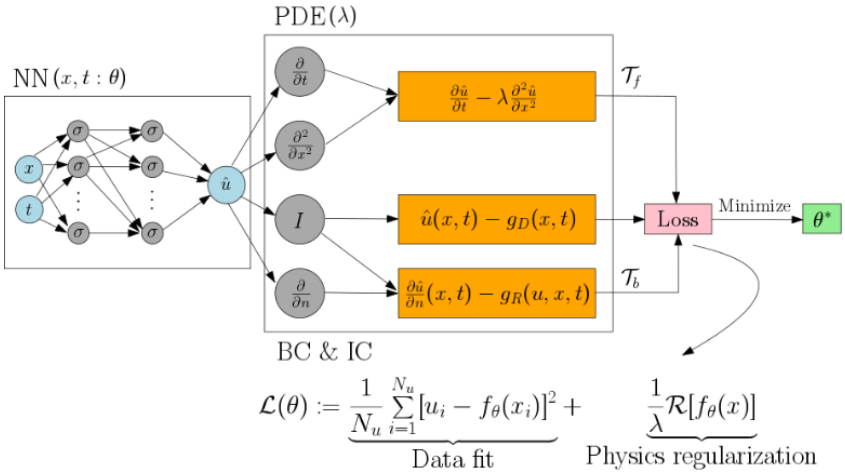

Figure. 1. Overview of PINN

PINNs include the Physics-based machine learning of the underlying problem in the loss function. The governing partial differential equation is used to directly calculate the loss function of PINNs, which is minimized during training. The residuals at collocation points, the weighted residuals obtained by the Galerkin-Method, or the energy functional of an Euler-Lagrange differential equation all contribute to the loss function [35, 36]. Thus, for training a PINNs,

there is no need to generate labeled training data in advance. In the problem domain the input data for PINNs points are sampled. Distance between the approximated solution of the partial differential equation and measured values of the solution is augmented by the loss function. Then one or more coefficients of the partial differential equation can be included as unknowns during training. By using this way, an inverse problem is solved. Solving inverse problems with PINNs may lead to a significant speedup compared to conventional methods[32]. PINNs solve eigenvalue problems when the loss function of a neural network is related to the Rayleigh-Ritz coefficient [37,38]. Physics-based machine learning models that could offer faster and more accurate solutions are essential to replace existing numerical epidemiological models.

To conduct this research, we took the following approach: We start with a SIERD model and use analytics to derive a differential equation of infection rate, which provides information on individual infection percentages in the region. The infection rate is then calculated using the heatmap snapshots for the region-infected community. This process uses Machine Learning techniques, specifically Deep Residual Recurrent Neural Networks **(**DR-RNN)[39]. The latter was pre-trained on simulated SIERD data before being trained on each region's recorded infected data. We validate the resulting SIERD model against the region's data until we know the infection rate. The following is a breakdown of the paper's structure. Section 2 talks about the methodology of the study introduce the procedure to produce data for PINN. The performance of PINN is evaluated and discussed in Section 3. Section 4 summarizes the limitation of the study, and section 5 concludes.

2. Methods

The PINN is trained on large quantities of data simulated by the SEIRD model PDEs with a heterogeneous diffusion model. The model describes the spatiotemporal spread of the COVID-19 pandemic, and aims to capture dynamics also based on human habits and geographical features[40]. Finally, the computational efficiency of the PINN is investigated. Our proposed model PINN uses the snapshots of COVID-19 of Delhi, India, from January 1 to April 30, 2021, to forecast the potential growth of COVID-19 for the next two weeks. Data was obtained from the ArcGIS. ArcGIS Online is a cloud-based mapping and analysis solution. Use it to make maps, analyze data, and share and collaborate. With a free public account, data can be created, stored, managed and shared with others.

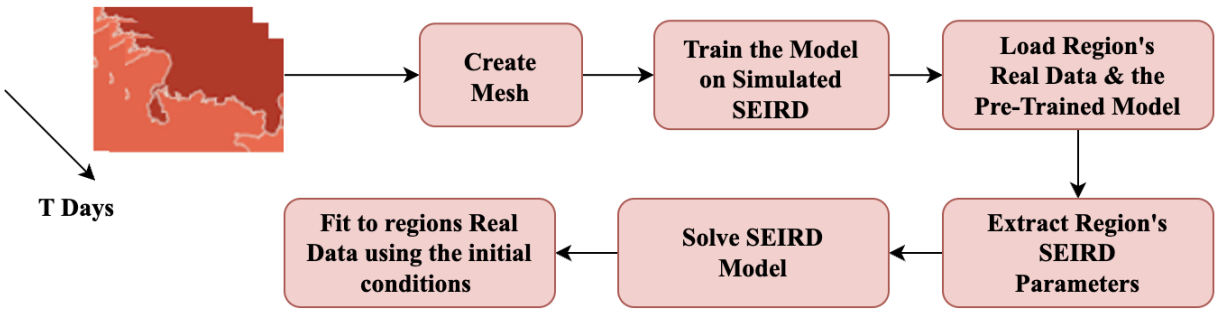

Figure 2. Overview of PINN

In this Figure.2, we describe the overview of PINN through a flowchart. We note that after the extraction of the infection rate from the data we use the SEIRD model to further investigate the parameter. Gmsh is used to build the mesh, which is uniformly optimized as the simulation

progresses. Gmsh is an open-source 3D finite element mesh generator with a built-in CAD engine and post-processor. Its design goal is to provide a fast, light, and user-friendly meshing tool with parametric input and flexible visualization capabilities. Gmsh's mesh module regroups several 1D, 2D, and 3D meshing algorithms: The 2D unstructured algorithms generate triangles and/or quadrangles (when recombination commands or options are used). The 3D unstructured algorithms generate tetrahedra, or tetrahedra and pyramids (when the boundary mesh contains quadrangles). [41,42]. After being developed in one level, the mesh has a minimum spatial resolution of around 1 kilometer.

2.1. Convolutional Neural Network

Yann André LeCun proposed CNN in 1998[43]. CNN has eight layers with weights, as shown in Figure.3, with the first five being convolutional and the last three being connected. The previous completely connected layer's output is fed into a 1000-way softmax, which generates a distribution over the 1000 class labels. Since the data representation to our neural network is an image (2D).

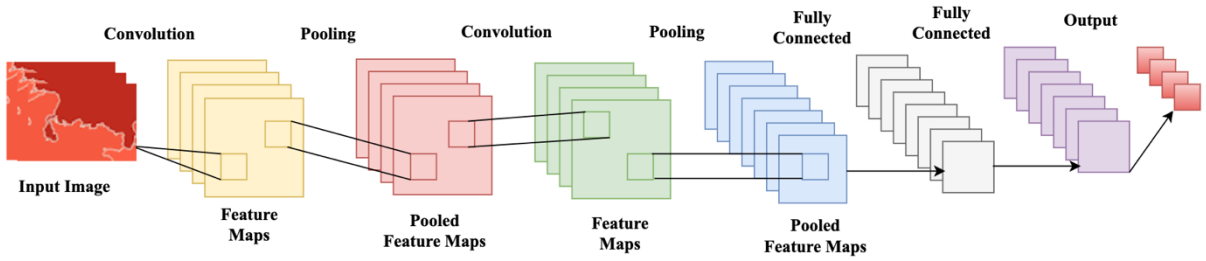

Figure. 3. CNN overview

2.2.Long Short-Term Memory

The Recurrent Neural Network (RNN) architecture can be written as:

$$h_t = tanh(W[h_{t-1}, z_t] + b) \quad (1)$$

$$x_t = Vh_t + c \quad (2)$$

$$L(\theta) = \frac{1}{TSN}\sum_{S=1}^{S}\sum_{n}^{N}\sum_{t=1}^{T}|\hat{y}_t - y_t| \quad (3)$$

Where z_t is the input at time t; h_t is the hidden state at time t; x_t is the output at time t; b and c are the bias vectors of the RNN; W and V are the weight matrices of the RNN is shown in Eq. (1) and Eq. (2). The current time step's input and the previous time step's hidden state are used in calculating the current hidden state., which acts as the RNN's memory, according to Eq. (1). For predicting the responses of dynamical systems, the loss function is shown in Eq. (3). Where $\hat{y}_t$ and y_t are the predicted and actual system states at given time t. The number of time measures, functions, and training samples is denoted by T, N, and S, respectively. Back-propagation through time is used to change the values of RNN parameters during the preparation[44]. However, it was reported that the standard RNN architecture has difficulties in learning long-term dependencies due to the vanishing or exploding gradient problem[45]. Gated RNN designs such as the Long Short Term Memory (LSTM) and the gated recurrent unit were developed to overcome this issue[46,47]. RNNs that can learn long-term dependencies are known as LSTM networks. LSTM anticipated by Hochreiter and Schmidhuber[48], has been used as an advance version of the RNN network and has overcome the limitation of RNN by use of a hidden layer unit known as memory cells. Memory cells have self-connections that store the network's temporal state and are controlled through three gates named: input gate, output gate, and forget gate [49]. The work of the input gate and output gate is used

to control the flow of memory cell input and outputs into the rest of the network. In addition, the forget gate has been added to the memory cell, which passes the output information with high weights from before the next neuron. The information that resides in memory depends upon the high activation results; if the input unit has high activation, the information is stored in the memory cell. In addition, if the output unit has high activation, then it will pass the information to the next neuron. Otherwise, input information with high weights resides in the memory cell [50]. The LSTM architecture can be expressed in the following Eq.(4-8).

$$f_t = \sigma_s(W_f[h_{t-1}, z_t] + b_f) \quad (4)$$

$$i_t = \sigma_s(W_i[h_{t-1}, z_t] + b_i) \quad (5)$$

$$o_t = \sigma_s(W_o[h_{t-1}, z_t] + b_o) \quad (6)$$

$$c_t = f_t * c_{t-1} + i_t * \sigma_h((W_c[h_{t-1}, z_t] + b_c) \quad (7)$$

$$h_t = o_t * \sigma_h(c_t) \quad (8)$$

where f_t is the forget gate vector; it is the input gate vector; o_t is the output gate vector; h_{t-1} is the hidden state vector; c_{t-1} is the cell state vector, W_c, W_i , $W_o$and W_f are the weight matrices;, b_c,b_o, b_i and b_f are the bias vectors; the operator $*$ denotes element-wise multiplication; σ_s represents the sigmoid function; and σ_h represents the hyperbolic tangent function.

2.3. Deep residual RNN (DR-RNN)

The Physics-based machine learning of dynamical systems is expressed in their governing equations, which can be written in a general form as:

$$dy/dt = f(t, y) \quad (9)$$

Where y is the dynamical system's state variable, Eq. (9) can be solved analytically or numerically to obtain the dynamical system's answer. For example, using the implicit Euler form, the device state at time instant *t + 1* can be obtained as:

$$y_{t+1} = y_t + h \times f(t+1, y_{t+1}) \quad (10)$$

where h is the time step size. From Eq. (10), a residual function can be formulated as:

$$r_{t+1} = y_{t+1} - y_t + h \times f(t+1, y_{t+1}) \quad (11)$$

By stacking I network layers as shown in Eq.(12), the DR-RNN architecture is meant to iteratively reduce the residual function provided in Eq. (11).

$$y_{t+1}^i = y_{t+1}^{i-1} - W * tanh(Ur_{t+1}^i), \text{ for i=1} \quad (12)$$

$$y_{t+1}^i = y_{t+1}^{i-1} - \frac{\eta_k}{\sqrt{G_k+\alpha}} r_{t+1}^i, \text{ for i>1} \quad (13)$$

where i is the layer number; r_{t+1}^i is the residual at time instant *t + 1* in the ith layer as show in Eq.(7-8); *W, U,* and η are the weight parameters of the DR-RNN [39];To avoid division by zero α is a used which is a small number; and H_i is determined as the residual's exponentially decreasing squared norm

$$H_i = \gamma||r_{t+1}||^2 + \beta H_{i-1} \quad (14)$$

where β and γ are the fraction factors and their values are set as 0.9 and 0.1, respectively. The training objective of the DR-RNN is to find a set of parameters that minimizes the residual function defined in Eq. (11) the DR-RNN can be thought as a numerical integrator, which is, to a great extent, like performing implicit integration, i.e. making the residual zero by solving Eq. (10),. DR-RNN is learning to perform implicit integration. The DR-RNN is explicit in time with a constant computational cost at each time step, unlike implicit integration methods.. To minimize the loss function given in DR-RNN the Adam optimizer[51] is used, which was created using the TensorFlow machine learning system in Eq.(3).

2.4. Modeling and integration of COVID-19

In this section, the COVID-19 is first briefly reviewed, and then the integration of the equations of SIERD into the DR-RNN is introduced. The development of short-term prediction models for forecasting the number of possible cases, aided by computational simulation of the virus's dynamics.

To avoid deaths and cure patients, strategic planning should be implemented in the public health industry. Using computational model's virus transmission can be forecasted. Here, we work with a spatio-temporal SEIRD model, presented in [40], and given in Eq.(15-19).

$$\frac{\partial s}{\partial t} + \varphi_i \left(1 - \frac{A_e}{n_p}\right) si + \varphi_e \left(1 - \frac{A_e}{n_p}\right) se - \nabla \cdot (n_p \upsilon_s \nabla s) = 0 \tag{15}$$

$$\frac{\partial e}{\partial t} - \varphi_i \left(1 - \frac{A_e}{n_p}\right) si - \varphi_i \left(1 - \frac{A_e}{n_p}\right) se + (\alpha + \gamma_e) e - \nabla \cdot (n_p \upsilon_e \nabla e) = 0 \tag{16}$$

$$\frac{\partial i}{\partial t} - \alpha e + (\gamma_i \mathrm{i} + \delta)\mathrm{i} - \nabla \cdot (n_p \upsilon_r \nabla i) = 0 \tag{17}$$

$$\frac{\partial i}{\partial t} - \gamma_e e - \gamma_i \mathrm{i} - \nabla \cdot (n_p \upsilon_r \nabla r) = 0 \tag{18}$$

$$\frac{\partial d}{\partial t} - \delta i = 0 \tag{19}$$

where the densities of the susceptible, exposed, infectious, recovered, and deceased populations are denoted by *s(x; t), e(x; t), i(x; t), r(x; t),* and *d(x; t)* respectively. The total living population is expressed by n_p Which is the number of all compartments except *d(x; t)*. we considered the tendency of outbreaks to cluster around large populations, φ_i and φ_e denote the transmission rates between symptomatic and susceptible individuals and asymptomatic and susceptible individuals, respectively (units days^{-1}), α denotes the incubation period (units days^{-1}), γ_e corresponds to the asymptomatic recovery rate (units days^{-1}), γ_i the symptomatic recovery rate (units days^{-1}), represents the mortality rate (units days^{-1}), and υ_s, υ_e, υ_i, υ_r are the diffusion parameters of the different population groups as denoted by the sub-scripted letters (units km^2 persons^{-1} days^{-1}). Note that all these parameters can be considered time and space dependent.

2.5. Learning virus dynamics

The snapshots can now be assembled into a snapshot matrix for the training PINN as shown in Figure 4. Once we have a PINN trained on the infected data of the region, we may use it to extract the presence and persistence of the social distancing measures typified through the function. The reference target mesh is the uniformly refined fixed mesh considered in the early stages of the simulation, presenting 56558 nodes and 78340 elements. The simulation considers a time step size of Δt = 0:25 days for the numerical integration and Δt_0= Δt = 0:25 days for the observations. We then run fixed mesh simulations. The simulation of the SIERD model, obtained using the Runge-Kutta method. The accuracy of the learned parameters by DR-RNN is validated using the mean square error with respect to the exact solution. We start by presenting the results for the daily learned parameters followed by the associated reproduction numbers. Then, predictions of infectious cases are provided. The details of the training are explained below.

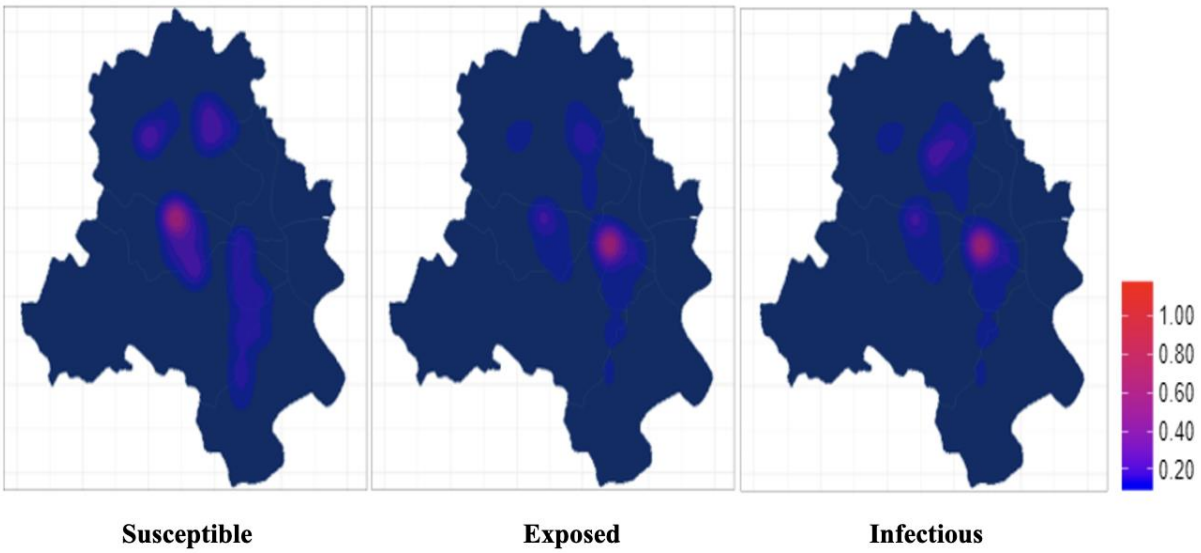

Figure.4. Initial conditions (January 1, 2021) in Delhi, India

The model is trained using a custom training loop by minimizing the mean squared error loss for data (MSE_u) as shown in Eq. (20).

$$MSE_u = \frac{1}{N}\sum_{i=1}^{N} |\, y_t - \hat{y}_t|^2 \qquad (20)$$

where $\{x_t, \hat{x}_t\}_{t=1}^{N}$denote the set of the reported, x_t real reported cases and $\hat{x}_t$ represents predicted cases and the mean squared error loss (MSE_s) represents the residuals that are obtained from the SIERD model Eq. (15-19) by subtracting the right side from the left side.

$$MSE_L = \omega_u MSE_u + \omega_s MSE_s \qquad (21)$$

The loss function of PINN is made of two terms: MSE_u and MSE_s as shown in Eq.(21).Which are defined by the last part of the above equation. The parameters $\{\omega_u, \omega_s\}$ denote the weight coefficients in the loss function that can balance the optimization effort between learning the data and satisfying the SEIRD model PDEs. We load the pre-trained model and allow all its weights to be tuned by minimizing again both the MSE_u and MSE_s on the region data. The pre-trained model is used to accelerate each region's training process.

3. Results and Discussions

In this section, we conduct extensive evaluations of the proposed method. COVID-19 heatmap snapshots for the first 120 days (120 snapshots) are collected. The snapshot matrix assembles the information regarding 106 days for training, while PINN approximates the results for 120 days. The idea is to forecast two weeks in the future, given the data observed in the past 106 days. We present MSE between the real data and the PINN prediction for the 120th day in Figure.5. Table 1 and Figure shows the overall mean square error for the compartments (susceptible, exposed, infectious, recovered, and deceased) approximations and computation time.

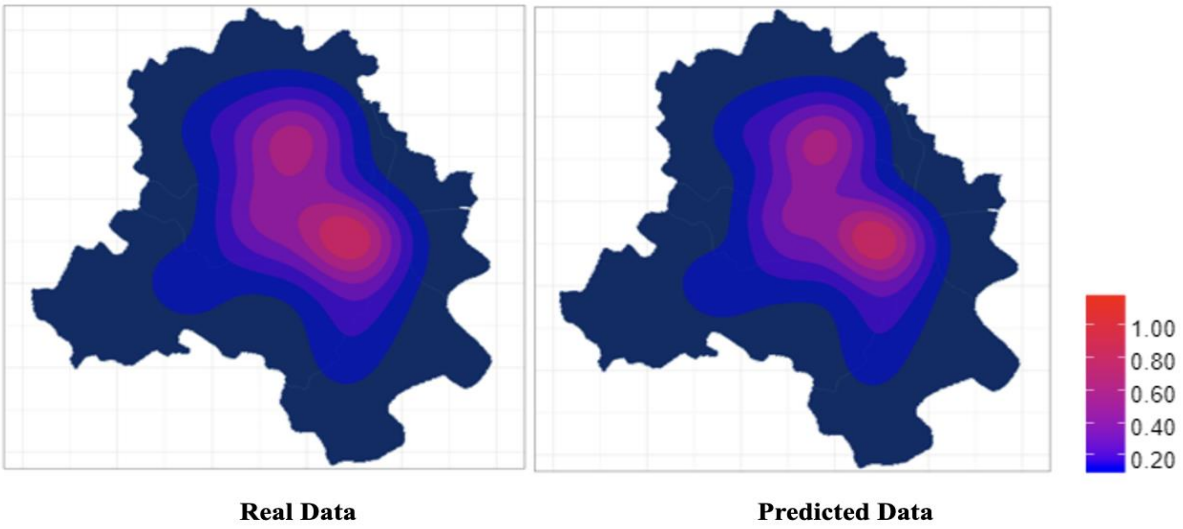

Figure 5: Comparison between a snapshot of real data and PINN prediction at t = 120 days. The PINN correctly captures the virus dynamics and accurately reproduces the solution with a mean square error $2.79e^{-3}$ in space and time.

Table.1. The mean square error for the six compartment approximations and computation time

Compartments	mean square error	**Solution Time(seconds)**
Susceptible	1.54×10^{-3}	76.4
Exposed	2.74×10^{-3}	61.9
Infectious	1.30×10^{-3}	77.3
Recovered	2.54×10^{-3}	81.9
Deceased	1.94×10^{-3}	88.5

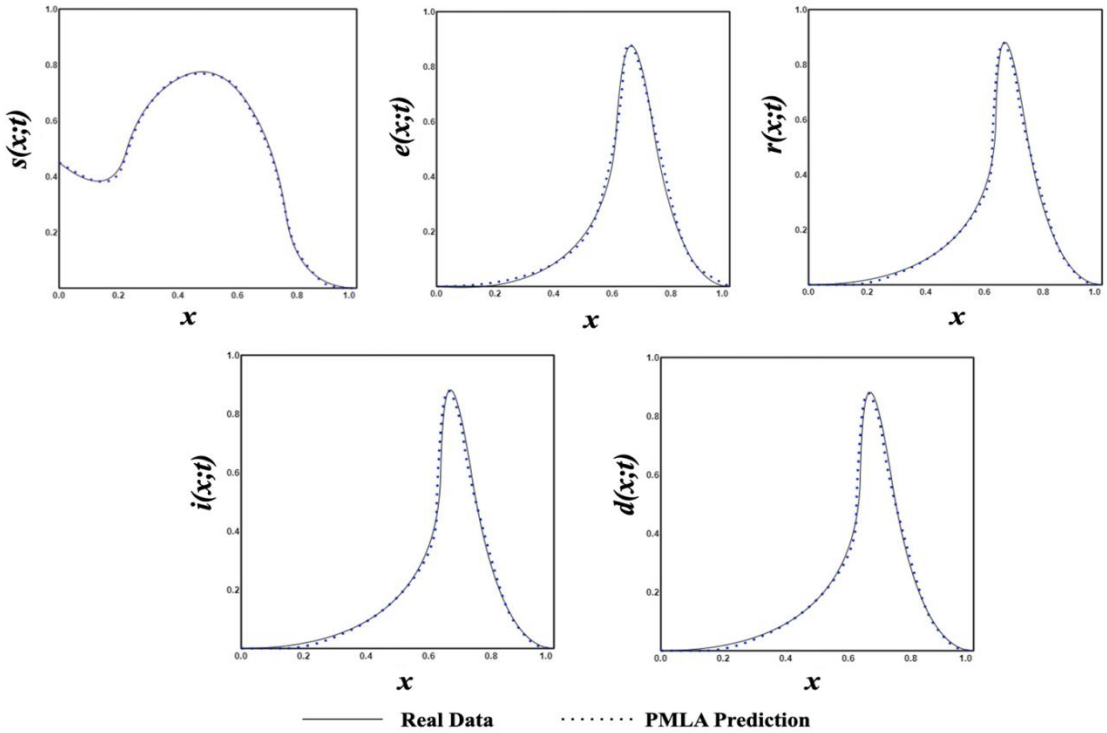

Figure.6. PINN model prediction and real data from April 16th to April 30, 2021

From Figure 6, we can note that most compartments show results in agreement with the real data, while the exposed compartment reveals more pronounced differences than the other compartments. We notice that the curves are different for each compartment. This discrepancy occurs due to the different parameters for each equation in the SEIRD model, which largely affects the system's dynamics. The dynamics for each compartment are different since each compartment presents different coupling, diffusion, and reaction parameters.

The curves for each compartment are different, which is the first thing we find. This discrepancy occurs due to the different parameters for each equation in the SEIRD model, which predominantly affects the system's dynamics. The dynamics for each compartment are different since each compartment presents different coupling, diffusion, and reaction parameters. Also, regarding this issue, since the parameters are time and space-dependent, sudden changes in their values can affect the dynamics of the system as well as PINN dynamics mapping ability. Some sudden changes in the susceptible and exposed compartments related to stricter public policies considered to reduce the transmission rates (parameters infectious and exposed) are incorporated into the model. Since the variation in the parameters is not introduced smoothly, PINN's ability to map sudden changes in the dynamics of the system is reflected by some spikes on the curves of the mean square errors in time. Comparing the real data and prediction data, we observe that the errors tend to grow as soon as the forecasting starts for the 118^{th} day. The exposed compartment, which yielded most of the oscillations due to parameter changing on the real data, presented the same behavior on the prediction phase around the 118^{th} day. We also note that the exposed compartment yields a significant mean square error for the 120th day than the other compartments. In Figure 6 we compared these results with the presented results, we can conclude that the predictions are reasonably accurate compared to the real data, especially considering the time required for calculation. In this study, the total population during the simulation is normalized by the total population modeled in the initial conditions. The total population is computed as the sum of the integral of the compartments divided by the sum of the integral elements of the mesh.

4. Limitations and Future work

The limitation of the study is that the PINN model does not consider population growth, the value must be theoretically constant for all the simulations. Secondly, variation in the parameters is not introduced smoothly so the PINN's ability to map sudden changes in the dynamics of the system is reflected by some spikes on the curves of the mean square errors in time. For further study, this developed model should be applied to various data to see the accuracy of the model, and variation in the parameters should be introduced smoothly so the PINN's ability to map sudden changes in the dynamics of the system is not reflected in the mean square errors in time.

5. Conclusion

Accurately forecasting COVID-19 is a major concern for many public health agencies to end an outbreak by effectively and timely planned hospital treatment. PINN a data-driven deep learning approach based on physics physics-informed neural network is introduced to solve the SIERD models on daily time-varying parameters. The introduced PINN is trained on large quantities of data simulated by the SEIRD model PDEs. The proposed model PINN used the snapshots of COVID-19 data obtained from ArcGIS of Delhi, India, from January 1 to April 30, 2021, to forecast the potential growth of COVID-19 for the next two weeks. The simulation of the SIERD model was obtained using the Runge-Kutta method. The accuracy of the learned parameters by DR-RNN is validated using the mean square

error concerning the exact solution. In the results, compartments showed agreement with the real data, while the exposed compartment revealed more pronounced differences than the other compartments. The study's results will aid policymakers and healthcare providers in effectively preparing and providing resources to deal with the crisis in the coming days and weeks, including nurses, beds, and intensive care units. It is shown that the predictions made by a PINN align well with the real data. The PINN can still predict the responses with sufficient accuracy, and limited training data does not have a significant effect on the prediction performance of the physics-based learning method. PINN models offer faster and more accurate solutions by replacing existing numerical epidemiological models.

References

1. Wang, C., et al., *A novel coronavirus outbreak of global health concern.* The lancet, 2020. **395**(10223): p. 470-473.
2. Petrosillo, N., et al., *COVID-19, SARS and MERS: are they closely related?* Clinical microbiology and infection, 2020. **26**(6): p. 729-734.
3. Maggi, E., G.W. Canonica, and L. Moretta, *COVID-19: Unanswered questions on immune response and pathogenesis.* Journal of Allergy and Clinical Immunology, 2020. **146**(1): p. 18-22.
4. Akhtar, A., et al., *COVID-19 detection from CBC using machine learning techniques.* International Journal of Technology, Innovation and Management (IJTIM), 2021. **1**(2): p. 65-78.
5. De Felice, F. and A. Polimeni, *Coronavirus disease (COVID-19): a machine learning bibliometric analysis.* in vivo, 2020. **34**(3 suppl): p. 1613-1617.
6. Kwekha-Rashid, A.S., H.N. Abduljabbar, and B. Alhayani, *Coronavirus disease (COVID-19) cases analysis using machine-learning applications.* Applied Nanoscience, 2021: p. 1-13.
7. Lalmuanawma, S., J. Hussain, and L. Chhakchhuak, *Applications of machine learning and artificial intelligence for Covid-19 (SARS-CoV-2) pandemic: A review.* Chaos, Solitons & Fractals, 2020. **139**: p. 110059.
8. Zoabi, Y., S. Deri-Rozov, and N. Shomron, *Machine learning-based prediction of COVID-19 diagnosis based on symptoms.* npj digital medicine, 2021. **4**(1): p. 3.
9. Heidari, A., et al., *Machine learning applications for COVID-19 outbreak management.* Neural Computing and Applications, 2022. **34**(18): p. 15313-15348.
10. Kuo, K.-M., P.C. Talley, and C.-S. Chang, *The accuracy of machine learning approaches using non-image data for the prediction of COVID-19: A meta-analysis.* International journal of medical informatics, 2022: p. 104791.
11. Alantari, H.J., et al., *An empirical comparison of machine learning methods for text-based sentiment analysis of online consumer reviews.* International Journal of Research in Marketing, 2022. **39**(1): p. 1-19.
12. Hu, H., et al., *A Modified PINN Approach for Identifiable Compartmental Models in Epidemiology with Application to COVID-19.* Viruses, 2022. **14**(11): p. 2464.
13. Vadyala, S.R., et al., *Prediction of the number of COVID-19 confirmed cases based on K-means-LSTM.* Array, 2021. **11**: p. 100085.
14. Zeroual, A., et al., *Deep learning methods for forecasting COVID-19 time-Series data: A Comparative study.* Chaos, Solitons & Fractals, 2020. **140**: p. 110121.

15. Kumar, N. and S. Susan. *COVID-19 pandemic prediction using time series forecasting models*. in *2020 11th international conference on computing, communication and networking technologies (ICCCNT)*. 2020. IEEE.
16. Qi, H., et al., *COVID-19 transmission in Mainland China is associated with temperature and humidity: A time-series analysis.* Science of the total environment, 2020. **728**: p. 138778.
17. Chimmula, V.K.R. and L. Zhang, *Time series forecasting of COVID-19 transmission in Canada using LSTM networks.* Chaos, Solitons & Fractals, 2020. **135**: p. 109864.
18. Singh, V., et al., *Prediction of COVID-19 corona virus pandemic based on time series data using Support Vector Machine.* Journal of Discrete Mathematical Sciences and Cryptography, 2020. **23**(8): p. 1583-1597.
19. Alassafi, M.O., M. Jarrah, and R. Alotaibi, *Time series predicting of COVID-19 based on deep learning.* Neurocomputing, 2022. **468**: p. 335-344.
20. Petropoulos, F., S. Makridakis, and N. Stylianou, *COVID-19: Forecasting confirmed cases and deaths with a simple time series model.* International journal of forecasting, 2022. **38**(2): p. 439-452.
21. Kumar, R., et al. *Covid-19 outbreak: An epidemic analysis using time series prediction model*. in *2021 11th international conference on cloud computing, data science & engineering (Confluence)*. 2021. IEEE.
22. Chen, X., et al., *Integration of machine learning prediction and heuristic optimization for mask delivery in COVID-19.* Swarm and Evolutionary Computation, 2023. **76**: p. 101208.
23. Solayman, S., et al., *Automatic COVID-19 Prediction Using Explainable Machine Learning Techniques.* International Journal of Cognitive Computing in Engineering, 2023.
24. Kamelesun, D., R. Saranya, and P. Kathiravan, *A Benchmark Study by using various Machine Learning Models for Predicting Covid-19 trends.* arXiv preprint arXiv:2301.11257, 2023.
25. Aslani, S. and J. Jacob, *Utilisation of deep learning for COVID-19 diagnosis.* Clinical Radiology, 2023. **78**(2): p. 150-157.
26. Hasan, M.M., et al., *Review on the Evaluation and Development of Artificial Intelligence for COVID-19 Containment.* Sensors, 2023. **23**(1): p. 527.
27. Barbu, T., *Structural inpainting techniques using equations of engineering physics.* Physica Scripta, 2020. **95**(4): p. 044001.
28. Vadyala, S.R., et al., *A review of physics-based machine learning in civil engineering.* Results in Engineering, 2021: p. 100316.
29. Willard, J., et al., *Integrating physics-based modeling with machine learning: A survey*. arXiv preprint arXiv:2003.04919, 2020. **1**(1): p. 1-34.
30. Long, Z., et al. *Pde-net: Learning pdes from data*. in *International conference on machine learning*. 2018. PMLR.
31. Carleo, G., et al., *Machine learning and the physical sciences.* Reviews of Modern Physics, 2019. **91**(4): p. 045002.
32. Vadyala, S.R., S.N. Betgeri, and N.P. Betgeri, *Physics-informed neural network method for solving one-dimensional advection equation using PyTorch.* Array, 2022. **13**: p. 100110.
33. Malinzi, J., S. Gwebu, and S. Motsa, *Determining COVID-19 dynamics using physics informed neural networks.* Axioms, 2022. **11**(3): p. 121.
34. Raissi, M., P. Perdikaris, and G.E. Karniadakis, *Physics-informed neural networks: A deep learning framework for*

solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational physics, 2019. **378**: p. 686-707.
35. Nabian, M.A. and H. Meidani, *A deep neural network surrogate for high-dimensional random partial differential equations.* arXiv preprint arXiv:1806.02957, 2018.
36. Lu, Y., T. Belytschko, and L. Gu, *A new implementation of the element free Galerkin method.* Computer methods in applied mechanics and engineering, 1994. **113**(3-4): p. 397-414.
37. Kovacs, A., et al., *Conditional physics informed neural networks.* Communications in Nonlinear Science and Numerical Simulation, 2022. **104**: p. 106041.
38. Smith, S.T., M.A. Bradford, and D.J. Oehlers, *Numerical convergence of simple and orthogonal polynomials for the unilateral plate buckling problem using the Rayleigh–Ritz method.* International Journal for Numerical Methods in Engineering, 1999. **44**(11): p. 1685-1707.
39. Kani, J.N. and A.H. Elsheikh, *DR-RNN: A deep residual recurrent neural network for model reduction.* arXiv preprint arXiv:1709.00939, 2017.
40. Viguerie, A., et al., *Simulating the spread of COVID-19 via a spatially-resolved susceptible–exposed–infected–recovered–deceased (SEIRD) model with heterogeneous diffusion.* Applied Mathematics Letters, 2021. **111**: p. 106617.
41. Avdis, A. and S. Mouradian, *A Gmsh tutorial.* Imperial College London, Applied Modelling and Computation Group (AMCG), 2012.
42. Geuzaine, C. and J.F. Remacle, *Gmsh: A 3-D finite element mesh generator with built-in pre-and post-processing facilities.* International journal for numerical methods in engineering, 2009. **79**(11): p. 1309-1331.
43. LeCun, Y. and Y. Bengio, *Convolutional networks for images, speech, and time series.* The handbook of brain theory and neural networks, 1995. **3361**(10): p. 1995.
44. Plaut, D.C., *Experiments on Learning by Back Propagation.* 1986.
45. Schaefer, A.M., S. Udluft, and H.-G. Zimmermann, *Learning long-term dependencies with recurrent neural networks.* Neurocomputing, 2008. **71**(13-15): p. 2481-2488.
46. Sherstinsky, A., *Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network.* Physica D: Nonlinear Phenomena, 2020. **404**: p. 132306.
47. Tjandra, A., et al. *Gated recurrent neural tensor network*. in *2016 International Joint Conference on Neural Networks (IJCNN)*. 2016. IEEE.
48. Hochreiter, S. and J. Schmidhuber, *Long short-term memory.* Neural computation, 1997. **9**(8): p. 1735-1780.
49. Zaremba, W., I. Sutskever, and O. Vinyals, *Recurrent neural network regularization.* arXiv preprint arXiv:1409.2329, 2014.
50. Shahid, F., A. Zameer, and M. Muneeb, *Predictions for COVID-19 with deep learning models of LSTM, GRU and Bi-LSTM.* Chaos, Solitons & Fractals, 2020. **140**: p. 110212.
51. Bock, S. and M. Weiß. *A proof of local convergence for the Adam optimizer*. in *2019 international joint conference on neural networks (IJCNN)*. 2019. IEEE.