EINNs: Epidemiologically-Informed Neural Networks

Alexander Rodríguez[1], **Jiaming Cui**[1], **Naren Ramakrishnan**[2], **Bijaya Adhikari**[3], and **B. Aditya Prakash**[1]

[1]College of Computing, Georgia Institute of Technology, Atlanta, USA
[2]Department of Computer Science, Virginia Tech, Arlington, USA
[3]Department of Computer Science, University of Iowa, Iowa City, USA
{arodriguezc, jiamingcui1997, badityap}@gatech.edu, naren@cs.vt.edu, bijaya-adhikari@uiowa.edu

Abstract

We introduce EINNs, a framework crafted for epidemic forecasting that builds upon the theoretical grounds provided by mechanistic models as well as the data-driven expressibility afforded by AI models, and their capabilities to ingest heterogeneous information. Although neural forecasting models have been successful in multiple tasks, predictions well-correlated with epidemic trends and long-term predictions remain open challenges. Epidemiological ODE models contain mechanisms that can guide us in these two tasks; however, they have limited capability of ingesting data sources and modeling composite signals. Thus, we propose to leverage work in physics-informed neural networks to learn latent epidemic dynamics and transfer relevant knowledge to another neural network which ingests multiple data sources and has more appropriate inductive bias. In contrast with previous work, we do not assume the observability of complete dynamics and do not need to numerically solve the ODE equations during training. Our thorough experiments on all US states and HHS regions for COVID-19 and influenza forecasting showcase the clear benefits of our approach in both short-term and long-term forecasting as well as in learning the mechanistic dynamics over other non-trivial alternatives.

1 Introduction

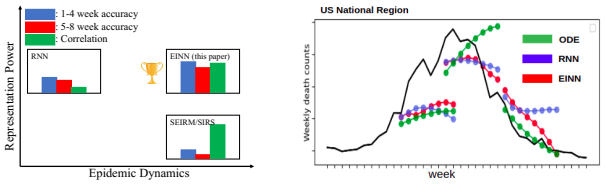
The COVID-19 pandemic has led to a maturing of methods for epidemic modeling and forecasting with the CDC establishing the first Center for Forecasting and Outbreak Analytics in 2021. A variety of forecasting innovations in machine learning and deep learning were developed–e.g., (Rodríguez et al. 2021a; Kamarthi et al. 2021)–with many lessons learned for COVID-19 and future pandemics. As the current experience has shown, predicting and preventing epidemics is one of the major challenges with far reaching impacts on health, economy and broad social well being[1].

From this perspective, active participation by several academic and industrial teams (including by coauthors) in multiple CDC-led forecasting initiatives has led to two broad themes that are important for epidemic modeling. First, modern disease surveillance has grown by leaps and bounds yielding novel data sources that can shed light into happenings real-time. *Statistical/ML epidemic models* leverage these data sources to provide dramatic improvements in short-term forecasting (usually 1-4 weeks ahead). At the same time, as these methods do not explicitly learn mechanistic dynamics, such methods do not provide understanding of how the epidemic will unfold at even longer time horizons, and do not support posing causal and counterfactual questions (e.g., design of countermeasures). Such longer-term forecasting remains the province of *mechanistic epidemic models* that can support scenario-based understanding of epidemic progression (e.g., "what will happen if schools are closed?"). However, these methods present scalability issues, their calibration is prone to noise (Hazelbag et al. 2020), and they have limited capability to ingest multimodal data sources (Viboud and Vespignani 2019). At the intersection of these two modeling approaches, we have hybrid models that make compartmental models (based on ordinary differential equations – ODEs) more informed of these data sources (Shaman et al. 2013; Arik et al. 2020). However, most of these approaches use the mechanistic model for prediction, thus, they are not flexible enough to fit the complex patterns in epidemics and have very few tunable parameters. In addition, they are not easily generalizable to new models/data sources or do not aim to incorporate ODE dynamics from first principles (e.g. predict ODE parameters instead of the ODE states).

In this paper, we develop a *general framework* for incorporating epidemic dynamics from a mechanistic model into a neural framework for forecasting, which enables seamless integration of multimodal data, greater representation power, and inclusion of composable neural modules of learned representations. Our focus is to leverage the selective superiorities of both approaches (see Figure 1) to have predictions that are *accurate* (low error) and *well-correlated* with even longer-term epidemic trends, than what is usually studied in past literature.

Recent lines of research (e.g., (Karniadakis et al. 2021)) aim to bridge scientific models (usually represented as differential equations) and ML algorithms. Specifically, the rapidly growing literature in physics-informed neural networks (PINNs) has demonstrated that integrating neural networks with ODEs can lead to large scalability improvements as forward passes (over the ODE) are no longer needed (Lu et al. 2021). In addition, the neural network flexibility and gradient-based learning enables robust solutions in spite of

[1]www.who.int/activities/preventing-epidemics-and-pandemics



(a) Modeling spectrum and performance

(b) Examples of forecasts

Figure 1: Our method, EINNs, (a) takes the best from both modeling approaches and is suitable for short- and long-term forecasting, and (b) its predictions are well-correlated with the epidemic trends.

noise (Yang et al. 2021). We propose to build upon this body of work to incorporate the dynamics of a mechanistic epidemic model into deep neural models. Our goal requires innovations to the PINN literature as many of the compartments in epi-models are latent (e.g. the actual number of people exposed to the disease) while most work in PINNs has only experimented with all states of the system dynamics being observable.

In addition, PINNs are limited to working with the variables that are described in the mechanistic equations which limits their capabilities to ingest data sources. On top of that, PINNs often use a simple multi-layer perceptron architecture whose inductive bias is often insufficient for sequential data. The main technical innovation of this paper comes from designing a transfer learning framework for transferring learned dynamics from a PINNs to a Recurrent Neural Network (RNN) that can ingest exogenous data features (data not-represented in the ODE). The goal of this is to have an RNN aware of epidemic ODE dynamics that has learned how data features (inputs to the RNN) shape/connect to the latent epidemic dynamics (outputs of the RNN).

We summarize our contributions as follows:

• **Push the boundaries of data-driven epi forecasting via integration of ODE-based mechanistic dynamics:** We introduce Epidemiologically-informed Neural Networks (EINNs), a new framework to bridge the gap between mechanistic and neural models for epidemiology. Our method incorporates epidemiological (expert) knowledge embedded in ODE models into neural networks (NNs) that ingest heterogeneous sources of data. The key idea in EINNs is utilizing a PINN to learn the latent epidemic dynamics and transfer its learned representations to another neural model with more appropriate inductive bias (RNN) and capable of ingesting heterogeneous sources of data that are exogenous to the ODE equations.

• **Transfer learning via gradient matching of dynamics:** We propose a novel method to transfer learned representations from a PINN (source model) to another neural network (target model). This is based on matching the gradients of mechanistic ODEs in the target model as we do so when training the source model. This cannot be directly done in the target model (RNN) due to its neural architecture and data inputs. Therefore, we propose approximating internal representations of source and target models to enable the target model to learn to match the ODE gradient.

• **Extensive empirical evaluation:** We evaluate our method in the challenging task of weekly COVID-19 forecasting (in 48 geographical regions and a period of 8 months) and flu (in 10 regions and 5 months). Our results showcase that our method can indeed leverage the 'best of both worlds' compared to other non-trivial ways of merging such approaches. We believe this opens new venues for exploring how AI can better complement domain knowledge in traditional epidemiology.

2 Related Work

Mechanistic and ML models for epidemic forecasting Epidemic mechanistic models (Hethcote 2000) like the popular SIR are designed using domain knowledge of the epidemic dynamics. They model causal underpinnings to explain empirically observed variables (e.g., mortality), and ODE-based models have been a workhorse of epidemiology since the late 18th century (Marathe et al. 2013). More recently, there are several successful applications of ML to short-term forecasting (Osthus et al. 2019; Brooks et al. 2018; Adhikari et al. 2019; Rodríguez et al. 2021a) which led them to be often ranked among the top performing models in these tasks (Cramer et al. 2022; Reich et al. 2019). Some of the recent deep learning innovations include advances in incorporting multi-view and multimodal data (Kamarthi et al. 2022b), spatial correlations (Deng et al. 2020; Jin et al. 2021), transfer learning for domain adaptation (Rodríguez et al. 2021b) and non-parametric approaches (Kamarthi et al. 2021; Zimmer et al. 2020).

Hybrid epidemic models They integrate mechanistic models and ML approaches (Rodríguez et al. 2022). Some lines of work use statistical techniques to estimate the mechanistic parameters (e.g. transmission rate) (Arik et al. 2020; Wang et al. 2021a), learn from simulation-generated data (Wang et al. 2019), or use the ODEs as regularization (Gao et al. 2021). However, in addition to the previously mentioned flexibility problems, these models require a forward pass over the mechanistic models, which may become very expensive for long periods and large mechanistic models. Furthermore, they often need to discretize the continuous ODE space, which is a delicate process as it has been found that sub-optimal discretization can impede the learning process (Thuerey et al. 2021). Our approach utilizes PINNs which allows skipping forward passes and discretization. Perhaps the most prominent in this line of research is the work by (Shaman et al. 2013) that integrated search volume data into an SIRS ODE model via the principle of data assimilation. However developing such data assimilation approaches requires a large amount of domain knowledge and cannot be used to incorporate many of the data features studied in our work (e.g., mobility).

Physics-informed neural networks (PINNs) PINNs are universal function approximators via neural networks that embed knowledge of ODEs and partial differential equations (PDEs) via unsupervised loss functions based on these equations. They have been used for forward and inverse problems with ODEs in a variety of domains including computational biology (Yazdani et al. 2020). PINNs have connections to implicit neural representations (Sitzmann et al. 2020) in

the sense that they both provide continuous representations breaking the discretization (grid) limitation, which is advantageous for when data samples are irregular. Previous work often use a multi-layer perceptron architecture because they are amenable for direct computation of derivatives of neural network outputs with respect to its inputs (via autograd). Indeed, incorporating inductive biases into PINNs is an active research area–e.g., convolutional layers (Wandel et al. 2022) and graph neural networks (Kumar and Chakraborty 2021)– and to our best knowledge a recurrent neural architecture for PINNs remains an open problem. Also, incorporating exogenous variables to this framework and working with partially observable systems are largely unexplored problems (Cai et al. 2021; Wang et al. 2021b). Our approach EINNs extends the capabilities of PINNs by directly addressing these limitations in the context of epidemiology.

A remotely related but popular line of work for learning dynamical systems is neural ODE (Chen et al. 2018). While PINNs learn from the system dynamics represented in an ODE or PDE (which represent domain expert knowledge from epidemiologists), neural ODEs learn an unknown ODE via neural networks and do a continuous modeling using a numeric ODE solver. Since neural ODEs cannot incorporate domain ODEs, they are not applicable to our problem.

3 Background

As mentioned earlier, we aim on merging neural models with epidemiological dynamics from a mechanistic model. Without loss of generality, here we introduce instantiations of such models which we refer to as building blocks. Their definitions help us to explain the formulation of the problem INCORPORATING EPI-DYNAMICS IN NNS and later our implementation and experiments. Additionally, we briefly introduces PINNs as formulated for Systems Biology.

3.1 ODE-based mechanistic epidemic models

Our first building block is a mechanistic epidemic model. Epidemiologists use different mechanistic models for each disease because infection dynamics and disease progression varies (Hethcote 2000). In this paper we use COVID-19 and influenza as a vehicle to demonstrate the benefits of our general framework, therefore, we use two different mechanistic epidemic models: SEIRM and SIRS (SIRS description is similar to SEIRM and can be found in our appendix).

SEIRM model for COVID-19 The SEIRM ODE model consists of five compartments: Susceptible (S), Exposed (E), Infected (I), Recovered (R), and Mortality (M). It is parameterized by four variables $\Omega = \{\beta, \alpha, \gamma, \mu\}$, where β is the infectivity rate, $1/\alpha$ is the mean latent period for the disease, $1/\gamma$ is the mean infectious period, and μ is the mortality rate. Due to COVID-19's prolonged incubation period, the SEIRM has been broadly used in modeling its progression (Wu et al. 2020; Morozova et al. 2021). It has also been used by the CDC in modeling transmission of Ebola (Gaffey et al. 2018). To capture the evolving nature of the dynamics and spread of COVID-19 (e.g. consider the multiple variant waves), we leverage the dynamic version of the SEIRM model, where the parameters governing the disease progression themselves evolve over time. In such a setting, the dynamics is governed by the set of parameters $\Omega_t = \{\beta_t, \alpha_t, \gamma_t, \mu_t\}$ at the given time-stamp t. Let $\mathbf{s}_t = [S_t, E_t, I_t, R_t, M_t]^T$ be the values taken by the states at time t. (ODE state S_t represents the number of susceptible people at time t, similar for the other states). Then, the ODEs describing the SEIRM model are given by $f_{\text{ODE}}(\mathbf{s}_t, \Omega_t)$:

$$\frac{dS_t}{dt} = -\beta_t \frac{S_t I_t}{N} \qquad \frac{dE}{dt} = \beta_t \frac{S_t I_t}{N} - \alpha_t E_t$$
$$\frac{dI_t}{dt} = \alpha_t E_t - \gamma_t I_t - \mu_t I_t \qquad \frac{dR_t}{dt} = \gamma_t I_t \qquad \frac{dM_t}{dt} = \mu_t I_t.$$

In our SEIRM model, only mortality is considered observed (Wu et al. 2020), therefore, ODE states S_t, E_t, I_t, and R_t are latent. By solving the set of ODEs, we can discover the *latent epidemic dynamics* which are described by the values of $\mathbf{s}_t$ and Ω_t for the entire period for which we have observational data, i.e., $\forall t \in \{t_0, \ldots, t_N\}$.

3.2 RNN architecture

Our second building block is a Recurrent Neural Network (RNN) with attention, which has been extensively used in neural epidemic forecasting as a central building block (Adhikari et al. 2019; Kamarthi et al. 2021; Wang et al. 2019). Here, we introduce the base architecture of this model. Informally, at prediction time $t = t_N$ we are given a multivariate time series of features/signals $\mathcal{X} = \{\mathbf{x}_t\}_{t=t_0}^{t_N}$ with $\mathbf{x}_t \in \mathbb{R}^{D_x}$, where D_x is the number of features. And we are tasked to predict K steps ahead in the future. We encode the feature time series until t_N by passing it through a Gated Recurrent Unit (GRU) (Cho et al. 2014) to obtain a condensed representation for each time step: $\{\mathbf{h}_t\}_{t=t_0}^{t_N} = \text{GRU}(\{\mathbf{x}_t\}_{t=t_0}^{t_N})$ where $\mathbf{h}_t$ is the hidden state of the GRU for time step t. To capture long-term relations and prevent over-emphasis on last terms of sequence we use a self-attention layer (Vaswani et al. 2017) which involves passing the embeddings into linear layers to extract meaningful similarities. Then, we use the attention weights to combine the latent representations and obtain a single embedding representing the time series of data features and pass it to a feedforward network to make the prediction $y_{t=N+k}$.

3.3 PINNs for Systems Biology

Recently, several works (Yazdani et al. 2020; Karniadakis et al. 2021) in Systems Biology have used PINNs for solving forward and inverse problems with ODEs. The neural network $\text{N}(t)$ is a function of single variable t and ODE system is in the form $f_{\text{ODE}}(t)$ describing the rate of change (gradient) of some function with respect to t. Gradient $\frac{d\text{N}(t)}{dt}$ can be using computed via Automatic Differentiation–autograd, which in turn makes it possible to train the neural network $\text{N}(\cdot)$ while minimizing the residual between the two gradients, e.g. loss $\left(\frac{d\text{N}(t)}{dt} - f_{\text{ODE}}(t)\right)^2$.

4 Problem formulation

As mentioned earlier, we aim on harnessing the strengths of both machine learning/deep learning approaches (which

have been very successful in short-term forecasting) and mechanistic models (which are useful for long-term trend projections). Hence, our problem is one of merging neural models with mechanistic model dynamics while maintaining benefits from both the techniques. To capture this specific intention, we modify traditional forecasting problems (Adhikari et al. 2019) in the following manner:

Problem: INCORPORATING EPI-DYNAMICS IN NNS
Given: • A base **epidemiological model** mathematically represented as a set of ODEs (for example, see the SEIRM and SIRS models in Section 3.1). • A base **RNN** (See Section 3.2). • **Data:** an observed multivariate time series of COVID/flu-related signals $\mathcal{X} = \{\mathbf{x}_t\}_{t=t_0}^{t_N}$ and corresponding values for the forecasting target (new COVID-associated deaths or ILI flu counts) $\mathcal{Y} = \{y_t\}_{t=t_0}^{t_N}$, where t_0 is the first day of the outbreak and t_N is the current date.
Predict: next K values of the forecasting target, i.e. $\{\hat{y}_{N+k}\}_{k=1}^{K}$ (here K is the size of the forecasting window/horizon), such that predictions are *accurate* and *well-correlated* with the trends of the epidemic curve.

We are also interested in learning if taking advantage of selective superiorities of both approaches can push the prediction horizon (i.e., how many steps ahead we can forecast). Typically, CDC forecasting initiatives request short-term forecasts up to 4-weeks ahead ($K = 4$) – see (Cramer et al. 2022; Jin et al. 2021). Longer forecasting horizons have not been explored much, thus, we propose the double of the current horizon (i.e., $K = 8$) in this paper.

5 Our Approach

To tackle the problem INCORPORATING EPI-DYNAMICS IN NNS, one can easily conjure 'naïve' approaches. A simple approach is to calibrate the given mechanistic model with the observed targets $\mathcal{Y}$, and train the base RNN using the generated (synthetic) curve. Similarly, one could also use the ODEs to regularize the neural predictions or could train an ensemble with neural network's and ODE-model's predictions. However, as we show later in our experiments, while these approaches often can maintain the performance of the base RNN, they do not generate well-correlated predictions.
Overview See Figure 2. We propose using an heterogeneous domain transfer learning setup (Moon and Carbonell 2017), where we transfer knowledge from a source model to a target model. Here our source model is a PINN whose purpose is discovering the latent epidemic dynamics (solving ODEs). The gradients of our ODE epi-models ($d\mathbf{s}_t/dt$) are with respect to time; therefore, as noted in Section 3.3, time is the only input to this PINN. Thus, we refer to this PINN as *time module*. The target model is an RNN which ingests data features from heterogeneous sources–thus we call it *feature module*–and incorporates appropriate inductive bias to model sequential time series data. Note that both source and target models predict the same output $\mathbf{s}_t$, which are the ODE states. Therefore, the feature module learns a mapping from a multivariate time series of COVID/flu-related signals $\mathcal{X}$ (exogenous data features, i.e., not-represented in the ODE) to the epidemic dynamics $\mathbf{s}_t$. Next, we explain each of these modules in detail.

5.1 Time module (source model): learning latent time-varying dynamics

The time module interfaces with the set of ODEs describing an epidemic mechanistic model. Via PINNs, it learns the latent epidemic dynamics given observational data. Following the introduction in Section 3.3, PINNs solve the ODEs by jointly minimizing the observational error and the residual between gradient given by the ODE and the gradient of the neural network with respect to the time inputs (computed via autograd). As in most literature–see Section 2–our time module $N_{\text{Time}}(t)$ is parametrized by a multi-layer perceptron that ingests time t as input and predicts ODE states for time t, denoted as $\mathbf{s}_t \in \mathbb{R}^{D_s}$, where D_s is the number of ODE states (e.g., 5 for SEIRM). We want this neural network to make predictions that follow epi dynamics described the set ODEs f_{ODE}. This is, we make $N_{\text{Time}}(t) = \mathbf{s}_t$; subject to $d\mathbf{s}_t/dt = f_{\text{ODE}}(\mathbf{s}_t, \Omega_t)$, where Ω_t are the learned ODE parameters for time t. We minimize the ODE loss (unsupervised loss $\mathcal{L}^{\text{ODE-T}}$) while fitting the observed data (supervised loss $\mathcal{L}^{\text{Data-T}}$):

$$\mathcal{L}^{\text{ODE-T}} = \frac{1}{N+1}\sum_{t=t_0}^{t_N}\left[\frac{d\mathbf{s}_t}{dt} - f_{\text{ODE}}(\mathbf{s}_t, \Omega_t)\right]^2 \quad (1)$$

$$\mathcal{L}^{\text{Data-T}} = \frac{1}{N+1}\sum_{t=t_0}^{t_N}\left[\hat{M}_t - M_t\right]^2 \quad (2)$$

where $\hat{M}_t$ is predicted mortality by the time module. To discover the latent dynamics, we want to obtain $\mathbf{s}_t$ and Ω_t.
Constraining the optimization via epi domain knowledge In contrast to the setting in most prior work in PINNs where most states of the system dynamics are observed, most states in our SEIRM model are latent. We found learning in such a scenario can be very challenging. We alleviate this by infusing additional epidemiological domain knowledge in the form of monotonicity constraints. In particular, we adapt monotonicity losses from (Muralidhar et al. 2018) to our setting. They proposed to penalize consecutive predictions if they are not monotonic in the required direction. Note that the difference between consecutive predictions are discrete approximation to the derivatives. Here, we generalize this loss to continuous derivatives by taking the limit $\lim_{t\to 0}$. Now, derivatives can be directly computed via autograd. To incorporate these constraints, we use domain knowledge. Using SEIRM as an example (similar can be easily derived for other epidemiological models), we know that the Susceptible state S_t monotonically decreases and the Recovered state R_t monotonically increases. Then, we add a penalty when dS_t/dt is positive and when dR_t/dt is negative as follows:

$$\mathcal{L}^{\text{Mono}} = \frac{1}{N+1}\Big(\sum_{t=t_0}^{t_N}\left[\frac{dS_t}{dt}\text{ReLU}(\frac{dS_t}{dt})\right]^2 + \sum_{t=t_0}^{t_N}\left[-1\frac{dR_t}{dt}\text{ReLU}(-\frac{dR_t}{dt})\right]^2\Big), \quad (3)$$

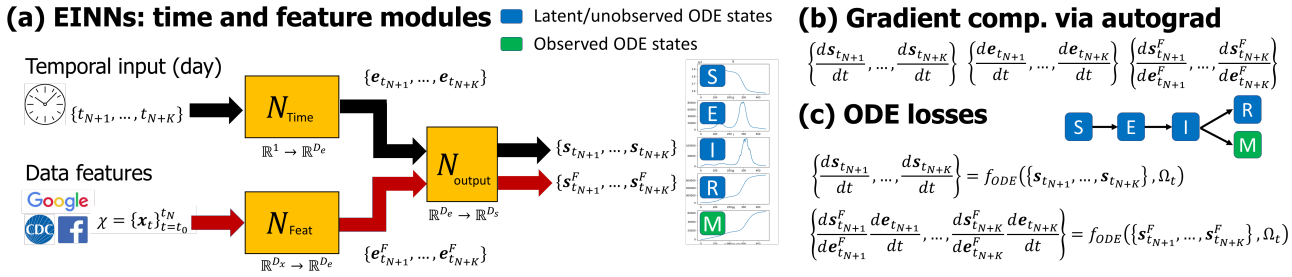


Figure 2: Depiction of our proposed framework EINNs. (a) The pipeline of EINNs has two independent computational paths: time module (source model) and feature module (target model). For simplicity, in our equations we refer to N_{Time} as a single module, but, in practice, this is implemented as two neural models: $N_{\text{output}} \circ N_{\text{Time}}$ where N_{output} is a multi-layer feedforward network. Similarly, N_{Feat} is implemented as $N_{\text{output}} \circ N_{\text{Feat}}$. During training step 2, when $\mathbf{e}_t \approx \mathbf{e}_t^F$, we will freeze layers and train only N_{output}. (b) Three gradients are computed via autograd: $d\mathbf{s}_t/dt$, $d\mathbf{e}_t/dt$, and $d\mathbf{s}_t^F/d\mathbf{e}_t^F$. Using these gradients we can compute ODE loss for the time module and approximate $d\mathbf{s}_t^F/dt$ via our gradient trick. (c) We utilize equations f_{ODE} to compute the ODE losses. Approximation of $d\mathbf{s}_t^F/dt$ is used to compute the ODE loss for the feature module $\mathcal{L}^{\text{ODE-F}}$. This loss encourages integration of ODE dynamics from the time module (source model) into the feature module (target model).

where ReLU(x) $= \max(0, x)$ is the rectified linear function. Note that S_t and R_t are part of $\mathbf{s}_t$, which is the output of the time module; thus, $\frac{dS_t}{dt}$ and $\frac{dR_t}{dt}$ are computed via autograd.

Coping with spectral bias in neural networks One of the central issues in fitting PINNs is the spectral bias of neural networks, which is the tendency of neural networks to fit low frequency signals (Wang et al. 2021b). To overcome this, usually the neural networks are given more flexibility to fit high frequency systems. Here, we adopted Gaussian Random Fourier feature mappings (Tancik et al. 2020): $\Gamma(v) = [\cos(2\pi\mathbf{B}v), \sin(2\pi\mathbf{B}v)]^T$, where each entry in $\mathbf{B} \in \mathbb{R}^{d\times 1}$ is sampled from $\mathcal{N}(0, \sigma^2)$, where σ is a hyperparameter.

Handling time-varying ODE parameters As mentioned earlier, our ODE model is time varying, therefore we have to learn mechanistic parameters for each time step, which increases the difficulty of the optimization. To make this more tractable, we propose a consistency loss between consecutive parameters.

$$\mathcal{L}^{\text{Param}} = \frac{1}{N+1}\sum_{t=t_0}^{t_N}\left[\Omega_{t+1} - \Omega_t\right]^2 \tag{4}$$

5.2 Feature module (target model): connecting features to epidemic dynamics via gradient matching

The feature module is composed of a based RNN and ingests multivariate time-series of COVID/flu-related signals $\mathcal{X}$ to predict the epidemic dynamics $\mathbf{s}_t^F$. (Note: embeddings and outputs of feature module have superscript F to avoid confusions with the ones from the time module). Here, we want to ensure that the predictions made by the feature module are consistent with the ones given by the ODE model. Hence, we want the feature module neural network $N_{\text{Feat}}(\mathcal{X})$ to follow the learned latent dynamics from the time module neural network $N_{\text{Time}}(t)$ (note the heterogeneous domains). We can formalize our goal as follows: $N_{\text{Feat}}(\mathcal{X}) = \mathbf{s}_t^F$; subject to $\frac{d\mathbf{s}_t^F}{dt} = f_{\text{ODE}}(\mathbf{s}_t^F, \Omega_t)$, where $\mathbf{s}_t^F \in \mathbb{R}^{D_s}$, are the ODE states predicted by the feature module and Ω_t are the same ODE parameters used by in the time module.

Matching the ODE gradient We cannot directly calculate $\frac{d\mathbf{s}_t^F}{dt}$ via autograd from the inputs as we did for the time module because our base RNN ingests features. We propose to use internal representations (embeddings) so that we can approximate the gradient to an expression that can be computed via autograd directly. Let $\mathbf{e}_t \in \mathbb{R}^{D_e}$ and $\mathbf{e}_t^F \in \mathbb{R}^{D_e}$ be embeddings for the time module and feature module, respectively (D_e is embedding size). Then, by using the chain rule, we propose to approximate the gradient of $\mathbf{s}_t^F$ assuming $\mathbf{e}_t \approx \mathbf{e}_t^F$ and have our *gradient trick*:

$$\frac{d\mathbf{s}_t^F}{dt} = \frac{d\mathbf{s}_t^F}{d\mathbf{e}_t^F}\frac{d\mathbf{e}_t^F}{dt} \approx \frac{d\mathbf{s}_t^F}{d\mathbf{e}_t^F}\frac{d\mathbf{e}_t}{dt} \tag{5}$$

where $\frac{d\mathbf{s}_t^F}{d\mathbf{e}_t^F}$ can be calculated in the feature module using autograd because $\mathbf{e}_t^F$ is the only variable that is needed to compute $\mathbf{s}_t^F$. Similarly, t is the only input needed for computing $\mathbf{e}_t$, thus, we can use autograd to compute $\frac{d\mathbf{e}_t}{dt}$. To make this approximation valid, we have to make these embeddings $\mathbf{e}_t$ and $\mathbf{e}_t^F$ similar. We do this with the following loss:

$$\mathcal{L}^{\text{Emb}} = \frac{1}{N+1}\sum_{t=t_0}^{t_N}\left[\mathbf{e}_t - \mathbf{e}_t^F\right]^2 \tag{6}$$

This derivation allows us to make the feature module to learn the gradients learned by the time module by minimizing an ODE loss for the feature module:

$$\mathcal{L}^{\text{ODE-F}} = \frac{1}{N+1}\sum_{t=t_0}^{t_N}\left[\frac{d\mathbf{s}_t^F}{d\mathbf{e}_t^F}\frac{d\mathbf{e}_t}{dt} - f_{\text{ODE}}(\mathbf{s}_t^F, \Omega_t)\right]^2 \tag{7}$$

Aligning with data and time module outputs Matching the ODE gradient is not enough to ensure the dynamics will be transferred. We have to make sure that the feature module outputs are aligned with data and with the ODE

dynamics as found by the time module. For fitting the data, we define data loss in a manner similar to the time module:

$$\mathcal{L}^{\text{Data-F}} = \frac{1}{N+1} \sum_{t=t_0}^{t_N} \left[\hat{M}_t^F - M_t\right]^2 \tag{8}$$

where $\hat{M}_t^F$ is the predicted mortality of the feature module. To align the time and feature modules, we use knowledge distillation (KD) (Ba et al. 2014), a popular transfer learning method. We impose our KD loss on the outputs of these two:

$$\mathcal{L}^{\text{Output}} = \frac{1}{N+1} \sum_{t=t_0}^{t_N} \left[\mathbf{s}_t - \mathbf{s}_t^F\right]^2. \tag{9}$$

Note that our time module is able to predict for any given time but our base RNN makes prediction for one target in the future. To align these two, we make our feature module to make joint prediction using a decoder GRU which takes $\mathbf{u}_{t_0:t_N}$ as the initial hidden state and roll the GRU forward for every prediction step ahead taking time as input. Thus, our decoder equations will be $\mathbf{e}_{t=t_{N+k}} = \text{GRU}(\mathbf{u}_{t_0:t_N}, t_{N+k})$ and our final predictions $y_{t=t_{N+k}} = \text{FFN}(\mathbf{e}_{t=t_{N+k}})$.

5.3 Model training, inference, and implementation

Training • Step 1: During the first training step, our goal is to make $\mathbf{e}_t \approx \mathbf{e}_t^F$ so that later we can use the gradient approximation stated in Equation (5) so that we can then match the gradient of the ODE. For this, we can train all parameters (neural networks and ODE parameters Ω_t) jointly with all the losses except for the feature ODE loss $\mathcal{L}^{ODE_F}$. • Step 2: Once $\mathcal{L}^{\text{Emb}}$ is small, we can train all losses together, however, it might be unstable and $\mathcal{L}^{\text{Emb}}$ may start to increasing which in turn invalidates our gradient matching trick and makes the minimization of $\mathcal{L}^{\text{ODE}_F}$ misleading. We found the training is more stable when freezing all previous layers to $\mathbf{e}_t$ in the time module and all previous layers to $\mathbf{e}_t^F$ in the feature module. In this case, we only focus the learning in the last layers, therefore, they should have enough representation power for this task. **Inference** At inference, although we have predictions from both time and feature modules, we solely utilize the feature module predictions as it ingests features and we want to emphasize the utility of inserting dynamics in ML models. **Implementation** As previous work notes (Yazdani et al. 2020) learning ODE dynamics with PINNs is challenging. We found it useful to bound ODE parameters Ω_t and initialize them based on the numerical solution of the ODE as output by a standard solver, e.g., Nelder-Mead. We keep Ω_t kept learnable through the training process because the neural networks may find a better solution. More details are in our appendix.

6 Experiments

6.1 Setup, metrics, and baselines

All experiments were conducted using a 4 Xeon E7-4850 CPU with 512GB of 1066 Mhz main memory and 4 GPUs Tesla V100 DGXS 32GB. Our method implemented in PyTorch (implementation details in the appendix) trains on a GPU in about 30 mins for one predictive task. Inference takes only a few seconds. Appendix, code, and other resources can be found online[2].

Evaluation All the results are for forecasting COVID-19 mortality in the US up to 8-weeks ahead in the future. For COVID-19, as per previous CDC-coauthored papers (Cramer et al. 2022), we evaluate at state and national level. Specifically, we include 47 states; we exclude 3 out of the 50 states where the SEIRM mechanistic model had difficulties fitting due to sparsity of death counts (specifically Alaska, Montana, and Wyoming). Our evaluation period is 8 months from Sept. 2020 to March 2021. which includes the complete Delta wave in the US, and we make 8-weeks ahead predictions for every two weeks in this period. We used June 2020 to Aug. 2020 to tune our models. For flu, we also follow CDC (Biggerstaff et al. 2018) and predict for all 10 HHS region[3] for a period of 5 months (Dec. 2017 to May 2018). For each forecasting week, all models are trained with historical data available until that week (i.e., they are trained on every prediction week). In total, we make 5696 predictions per model which requires training each of them 700+ times, therefore, is very computationally expensive to run all models for multiple runs.

Metrics Our focus is in predictions that are accurate and well-correlated the epidemic trends, thus we measure two aspects of our predictive performance: error and trend correlation. • **Error metrics**: As previous work in this area (Adhikari et al. 2019), we adopt metrics based on root mean squared error and absolute deviation. Because the number of deaths largely vary across regions, we use normalize versions of popular error metrics so that we can aggregate performance over multiple regions. We use two different versions of Normalized Root Mean Squared Error (NR1 and NR2) and Normal Deviation (ND) following (Roy et al. 2021; Remy et al. 2021). These metrics are described in detail in our appendix. For all of these metrics, we calculate them at short-term forecasting (1-4 weeks) and long-term forecasting (5-8 weeks) and calculate their mean value. • **Correlation metric**: Following (Deng et al. 2020), we use Pearson correlation and use their median across weeks.

Data We collected important publicly available signals from a variety of trusted sources that are relevant to COVID-19 forecasting. For COVID-19, we collected 13 features in this dataset, this include mobility from Google and Apple , social media surveys from Facebook, hospitalization data from the U.S. Depart. of Health & Human Services and CDC , and cases and mortality from Johns Hopkins Univ. For flu, we use the 14 signals from the Google symptom dataset. See appendix for more details and links.

Baselines As we are the first to pose the INCORPORATING EPI-DYNAMICS IN NNS problem, we do not have off-the-shelf baselines. Instead, our focus is on how to incorporate ODE dynamics into NNs. Hence we focus on the different ways these have been explored in literature (Dash

[2]Resources website: https://github.com/AdityaLab/EINNs

[3]hhs.gov/about/agencies/iea/regional-offices/index.html

Table 1: EINNs is the only one consistently providing accurate and well-calibrated forecasts and it is among the best performing for all metrics (lower NR1, NR2 and ND is better; higher Pearson correlation is better). Top 2 models per column are in bold (including tied models). These results are an average across 5696 predictions per model.

	Short-term (1-4 wks)			Long-term (5-8 wks)			Trend correlation
Model	**NR1**	**NR2**	**ND**	**NR1**	**NR2**	**ND**	**PC**
Task 1: COVID-19 Forecasting (US National + 47 states)							
RNN (GRU+Atten)	1.09	0.50	0.86	1.19	0.53	0.96	0.08
Mechanistic model (SEIRM)	2.35	1.13	1.36	7.14	2.99	3.11	**0.53**
GENERATION	0.79	0.35	0.60	**0.93**	**0.40**	0.74	-0.01
REGULARIZATION	1.05	0.48	0.81	1.19	0.53	0.97	0.09
ENSEMBLING	0.91	0.41	0.68	**0.93**	**0.40**	**0.69**	-0.01
EINNs (ours)	**0.54**	**0.24**	**0.38**	**0.85**	**0.37**	**0.66**	**0.46**
PINN (time module standalone)	0.84	0.38	0.64	**0.93**	**0.40**	0.72	0.24
EINNs-NoGradMatching	**0.64**	**0.29**	**0.49**	0.98	0.43	0.79	0.03
Task 2: Influenza Forecasting (10 HHS regions)							
RNN (GRU+Atten)	0.72	0.38	0.67	1.19	0.51	1.14	-0.03
Mechanistic model (SIRS)	0.72	0.38	0.51	1.16	0.55	0.81	**0.71**
GENERATION	0.76	0.4	0.71	1.21	0.52	1.15	-0.14
REGULARIZATION	1.19	0.64	1.00	1.22	0.54	0.9	-0.45
ENSEMBLING	0.89	0.47	0.77	**0.83**	**0.35**	**0.73**	-0.69
EINNs (ours)	**0.53**	**0.27**	**0.37**	**1.01**	**0.42**	**0.73**	**0.68**
PINN (time module standalone)	0.55	0.29	0.44	1.13	0.48	1.02	-0.47
EINNs-NoGradMatching	**0.53**	**0.27**	**0.38**	1.02	**0.42**	0.76	0.50

et al. 2022). • GENERATION: Similar to (Wang et al. 2019; Sanchez-Gonzalez et al. 2020), the NN learns directly from data generated by the numerical solution of SEIRM/SIRS. • REGULARIZATION: Similar to (Gao et al. 2021; Gaw et al. 2019), the NN predicts both the ODE states and the ODE parameters. Then uses the ODE parameters to regularize the states via a loss based on the SEIRM/SIRS equations. • ENSEMBLING: As per (Adiga et al. 2021; Yamana et al. 2017), combines predictions from RNN and SEIRM/SIRS via a NN that outputs final predictions.

6.2 Results in COVID-19 and influenza

Our results showcase EINN as an effective general framework for incorporating epidemic dynamics from a mechanistic model into a neural network. We first demonstrate that we can leverage advantages from both modeling paradigms resulting in consistently good forecasts across all tasks and metrics. We also compare against other non-trivial methods to incorporate ODE dynamics into neural models. To contextualize our model's performance with the broader picture of epidemic forecasting, we also have results with standard data-driven baselines, which can be found in our appendix.

Q1: Leveraging advantages of both mechanistic models and neural models. Our RNN has a lower or similar error in short- and long-term forecasting than the SEIRM and SIRS, but its predictions are much less correlated with epidemic trends (see lines 1-2 of comprehensive results in Table 1). By integrating mechanistic and neural models, EINNs is capable of taking advantage of both. Comparing EINNs with the SEIRM/SIRS, Pearson correlations are close but our predictions are much more accurate (up to 77% less error). Indeed, EINNs not only improves RNN correlation by 475% but also its accuracy up to 55% thanks to the incorporation of short and long-term dynamics. Note that our goal was not to beat the SEIRM/SIRS but have a method that has a consistently good performance across accuracy and correlation.

Q2: Benefits over other ways to incorporate epidemic dynamics into neural models. EINNs has the lowest error and best correlation in comparison with other existing ways to incorporate epidemic dynamics to neural networks. We can see that these methods may excel in one task (e.g., ENSEMBLING in long-term forecasting) but they are severely worse in other important tasks. Instead, EINNs is the only one consistently good in all tasks.

Q3: Ablation: time module PINN and gradient matching. We perform ablation studies to understand what are the contributions of the main components of our model. First, we analyze our time module trained standalone, i.e., being trained without the feature module with losses in Equations (5-9) (PINN in Table 1). We can see that, although our time module PINN directly interacts with the ODE and their behavior will be coupled during training, they have different behavior in test. In fact, this points to the need that we need features to be able to extract representations that generalize in test. Second, we assess the contribution of our gradient matching trick (EINNs-NoGradMatching), for which we train with all losses except for the ones in Equations (6) and (7). In this scenario where only $\mathcal{L}^{\text{Output}}$ helps to transfer the dynamics, we can see that it is a less effective way.

Q4: Case-study and **Q5: Sensitivity to hyperpameters.** In our appendix, we conducted a case study in US National and New York to visually analyze the advantages of our method in both accuracy and correlation (see example for US National in In Figure 1b). We found most hyperparameters are not sensitive. See appendix for more results and details.

7 Discussion and Societal Impact

The COVID-19 pandemic has impacted possibly every aspect of life and has exemplified our vulnerability to major public health threats. This underscores the importance of infectious disease detection and prediction for shaping more resilient societies. Preventing and responding to such pandemics requires trustworthy epidemic forecasts, e.g. forecasts well correlated with actual epidemic trends.

The ML community has been very active in CDC forecasting initiatives and has harnessed multiple successes. However, generating trustworthy epidemic forecasts may require more than only data. In this paper we tackle this challenge by introducing EINNs to incorporate mechanistic dynamics (via the SEIRM/SIRS model) into neural models (using a RNN style base model). We show the effectiveness of a principled method to transfer relevant knowledge via gradient matching of the ODE equations, without integrating (forward pass) the ODE model. Through extensive experiments over states/regions in the US we also show the usefulness of EINNs in COVID-19 and flu forecasting and also the importance of our various design choices. Overall, we believe this work opens up new avenues for leveraging epi domain knowledge into AI models for better decision making. Connecting complex mechanistic models to neural networks also enables us to have learned representations useful for other tasks downstream like what-if predictions, which would be worth exploring. In addition investigating more complex epidemiological models (like network based agent models) would be fruitful.

Acknowledgments

This work was supported in part by the NSF (Expeditions CCF-1918770, CAREER IIS-2028586, RAPID IIS-2027862, Medium IIS-1955883, Medium IIS-2106961, CCF-2115126), CDC MInD program, ORNL, faculty research award from Facebook and funds/computing resources from Georgia Tech. B.A. was supported by CDC-MIND U01CK000594 and start-up funds from University of Iowa. NR was supported by US NSF grants Expeditions CCF-1918770, NRT DGE-1545362, and OAC-1835660. We also would like to thank Harsha Kamarthi for his helpful suggestions which improved the paper.

References

Adhikari, B.; et al. 2019. EpiDeep: Exploiting Embeddings for Epidemic Forecasting. In *KDD*.

Adiga, A.; et al. 2021. All Models Are Useful: Bayesian Ensembling for Robust High Resolution COVID-19 Forecasting. In *KDD*.

Arik, S. O.; et al. 2020. Interpretable Sequence Learning for COVID-19 Forecasting. *NeurIPS*.

Ba, J.; et al. 2014. Do deep nets really need to be deep? *NIPS*.

Biggerstaff, M.; et al. 2018. Results from the second year of a collaborative effort to forecast influenza seasons in the United States. *Epidemics*.

Brooks, L. C.; et al. 2018. Nonmechanistic forecasts of seasonal influenza with iterative one-week-ahead distributions. *PLOS Comp. Biology*.

Cai, S.; et al. 2021. DeepM&Mnet: Inferring the electroconvection multiphysics fields based on operator approximation by neural networks. *Journal of Comp. Physics*.

Chen, R. T.; et al. 2018. Neural ordinary differential equations. In *NeurIPS*.

Cho, K.; et al. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *EMNLP*.

Cramer, E. Y.; et al. 2022. Evaluation of individual and ensemble probabilistic forecasts of COVID-19 mortality in the US. *PNAS*.

Dash, T.; et al. 2022. A review of some techniques for inclusion of domain-knowledge into deep neural networks. *Scientific Reports*.

Deng, S.; et al. 2020. Cola-GNN: Cross-location Attention based Graph Neural Networks for Long-term ILI Prediction. In *CIKM*.

for Disease Control, C.; and Prevention. 2020. U.S. Influenza Surveillance System: Purpose and Methods.

Gaffey, R. H.; et al. 2018. Application of the CDC EbolaResponse modeling tool to disease predictions. *Epidemics*.

Gao, J.; et al. 2021. STAN: spatio-temporal attention network for pandemic prediction using real-world evidence. *JAMA*.

Gaw, N.; et al. 2019. Integration of machine learning and mechanistic models accurately predicts variation in cell density of glioblastoma using multiparametric MRI. *Scientific reports*.

Hazelbag, C. M.; et al. 2020. Calibration of individual-based models to epidemiological data: A systematic review. *PLoS Comp Bio*, 16(5): e1007893.

Hethcote, H. W. 2000. The Mathematics of Infectious Diseases. *SIAM Review*.

Jin, X.; et al. 2021. Inter-Series Attention Model for COVID-19 Forecasting. In *SDM*.

Kamarthi, H.; et al. 2021. When in Doubt: Neural Non-Parametric Uncertainty Quantification for Epidemic Forecasting. In *NeurIPS*.

Kamarthi, H.; et al. 2022a. Back2Future: Leveraging Backfill Dynamics for Improving Real-time Predictions in Future. *ICLR*.

Kamarthi, H.; et al. 2022b. CAMul: Calibrated and Accurate Multi-view Time-Series Forecasting. *WWW*.

Karniadakis, G.; et al. 2021. Physics-informed machine learning. *Nature Reviews Physics*.

Kumar, Y.; and Chakraborty, S. 2021. GrADE: A graph based data-driven solver for time-dependent nonlinear partial differential equations. *arXiv*.

Liu, X.; et al. 2021. Self-supervised learning: Generative or contrastive. *TKDE*.

Lu, L.; et al. 2021. DeepXDE: A deep learning library for solving differential equations. *SIAM Review*.

Marathe, M.; et al. 2013. Comp. epidemiology. *Comm. of the ACM*.

Moon, S.; and Carbonell, J. G. 2017. Completely Heterogeneous Transfer Learning with Attention-What And What Not To Transfer. In *IJCAI*.

Morozova, O.; et al. 2021. One year of modeling and forecasting COVID-19 transmission to support policymakers in Connecticut. *Scientific reports*.

Muralidhar, N.; et al. 2018. Incorporating Prior Domain Knowledge into Deep Neural Networks. In *IEEE Big Data*.

Osthus, D.; et al. 2019. Dynamic Bayesian Influenza Forecasting in the United States with Hierarchical Discrepancy (with Discussion). *Bayesian Analysis*.

Pei, S.; and Shaman, J. 2020. Aggregating forecasts of multiple respiratory pathogens supports more accurate forecasting of influenza-like illness. *PLoS Comp Bio*, 16(10): e1008301.

Reich, N. G.; et al. 2019. A collaborative multiyear, multimodel assessment of seasonal influenza forecasting in the United States. *PNAS*.

Remy, S. L.; et al. 2021. Overcoming Digital Gravity when using AI in Public Health Decisions. *arXiv*.

Rodríguez, A.; et al. 2022. Data-centric epidemic forecasting: A survey. *arXiv preprint arXiv:2207.09370*.

Rodríguez, A.; et al. 2021a. DeepCOVID: An Operational Deep Learning-driven Framework for Explainable Real-time COVID-19 Forecasting. In *AAAI*.

Rodríguez, A.; et al. 2021b. Steering a Historical Disease Forecasting Model Under a Pandemic: Case of Flu and COVID-19. In *AAAI*.

Roy, P.; et al. 2021. Deep diffusion-based forecasting of COVID-19 by incorporating network-level mobility information. In *ASONAM*.

Sanchez-Gonzalez; et al. 2020. Learning to simulate complex physics with graph networks. In *ICLR*.

Shaman, J.; Pitzer, V. E.; Viboud, C.; Grenfell, B. T.; and Lipsitch, M. 2010. Absolute humidity and the seasonal onset of influenza in the continental United States. *PLoS biology*, 8(2): e1000316.

Shaman, J.; et al. 2013. Real-time influenza forecasts during the 2012–2013 season. *Nature Comm.*

Sitzmann, V.; et al. 2020. Implicit neural representations with periodic activation functions. In *NeurIPS*.

Tancik, M.; et al. 2020. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. In *NeurIPS*.

Thuerey, N.; et al. 2021. *Physics-based Deep Learning*. WWW.

Vaswani, A.; et al. 2017. Attention is all you need. *NIPS*.

Viboud, C.; and Vespignani, A. 2019. The future of influenza forecasts. *PNAS*.

Wandel, N.; et al. 2022. Spline-PINNs: Approaching PDEs without data using fast, physics-informed hermite-spline CNNs. In *AAAI*.

Wang, L.; et al. 2019. DEFSI: Deep Learning Based Epidemic Forecasting with Synthetic Information. *AAAI*.

Wang, R.; et al. 2021a. Bridging physics-based and data-driven modeling for learning dynamical systems. In *L4DC*.

Wang, S.; et al. 2021b. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM*.

Wu, J. T.; et al. 2020. Nowcasting and forecasting the potential domestic and international spread of the 2019-nCoV outbreak originating in Wuhan, China: a modelling study. *The Lancet*.

Yamana, T. K.; et al. 2017. Individual versus superensemble forecasts of seasonal influenza outbreaks in the United States. *PLoS Comp. Bio*.

Yang, L.; et al. 2021. B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data. *Journal of Comp. Physics*.

Yazdani, A.; et al. 2020. Systems biology informed deep learning for inferring parameters and hidden dynamics. *PLOS Comp. Bio*.

Zimmer, C.; et al. 2020. Influenza Forecasting Framework based on Gaussian Processes. In *ICML*.

A Background (continuation)

SIRS model for seasonal influenza. The SIRS model has been extensively used for modelling the seasonal influenza outbreaks. Here we use the version proposed in (Shaman et al. 2010) which other follow up papers have used to guide their analysis (e.g., (Pei and Shaman 2020)). The model consists of three compartments: Susceptible (S_t), Infected (I_t), and Recovered (R_t). It is parameterized by three variables $\Omega = \{\beta, D, L\}$, where β is the infectivity rate, D is the mean duration of immunity, and L is the mean duration of the immunity period. Note that $N - S - I$ is the number of immune individuals. As per in SEIRM, we utilize time-varying parameters $\Omega_t = \{\beta_t, D_t, L_t\}$ at the given time-stamp t and denote the ODE states as $\mathbf{s}_t = [S_t, I_t, R_t]^T$. The ODEs describing the model is as follows:

$$\frac{dS_t}{dt} = \frac{N - S_t - I_t}{L_t} - \frac{\beta_t I_t S_t}{N} \qquad (10)$$
$$\frac{dI_t}{dt} = \frac{\beta_t I_t S_t}{N} - \frac{I_t}{D_t}$$

Calibration. The target/calibration variable for COVID-19 is associated mortality, while in flu it is percentage of patients with influenza-like-illness (ILI) symptoms, which is not directly represented in the SIRS model. We provide details in Section E on how to connect ILI to the SIRS model.

B More details on data

Our datasets were collected from publicly available sources[4]. We describe them in details as follows.

- Data signals 1: Mobility signals. The signals originate from the record of people visiting points of interest (POI) in various regions. According to Google, daily changes in visits to various POI categories are collected and compared with the period January 3 - February 6, 2020. Additionally, we collected a daily change of visitors from Apple, which shows the relative volume of directions requested across different US states compared to January 13. Different non-pharmaceutical interventions (NPIs) and different policies adopted by different states are implicitly illustrated by mobility signals.
- Data signals 2: Symptomatic surveys. Every day, Facebook collects statistics on COVID-like illness (%CLI) and influenza-like illness (%ILI) across the US and different states. On the basis of symptoms reported in voluntary surveys, they estimate this percentage.
- Data signals 3: Symptom search data. Google collects records of searches related to symptoms for multiple conditions and syndromes across the US and different states. Their system provides a metric that quantifies search volume associated with specific symptoms, which undergoes a privacy-protecting mechanism before being publicized. There are 400+ symptoms available in this dataset dating back to 2017, from which we only use a subset of 14. These are the following which are symptoms associated with influenza: Fever, Low-grade fever, Cough, Sore throat, Headache, Fatigue, Vomiting, Diarrhea, Shortness of breath, Chest pain, Dizziness, Confusion, Generalized tonic–clonic seizure, and Weakness.
- Data signals 4: Number of hospitalizations. The US Department of Health & Human Services provides daily hospitalization admissions dating back to January 1, 2020. Several primary sources provide facility-level granularity reports to create this signal: (1) HHS TeleTracking, (2) reporting provided to HHS Protect by state/territorial health departments on behalf of their healthcare facilities, and (3) the National Healthcare Safety Network.
- Data signals 5: Number of new deaths. The Johns Hopkins University reports daily mortality for COVID-19. They collect and curate data from official websites of state public health departments across the US. This has been the source of data for the CDC COVID-19 forecasting initiative (Craemer et al. 2022).
- Data signals 6: weighted Influenza-like Illness (wILI). Time series data are collected by CDC from over 3,500 outpatient healthcare providers in the Outpatient Influenza-like Illness Surveillance Network (ILINet). Health care providers report voluntarily every week the percentage of patients with Influenza-like Illness (ILI) symptoms. ILI is defined as "fever (temperature of 100◦F [37.8◦C] or greater) and a cough and/or a sore throat without a known cause other than influenza." This has been the source of data for previous iterations of the CDC FluSight forecasting initiative (Reich et al. 2019).

[4]Data links: apple.com/covid19/mobility; google.com/covid19/mobility; coronavirus.jhu.edu; healthdata.gov; delphi.cmu.edu; gis.cdc.gov/grasp/COVIDNet/COVID19_3.html; goo.gle/covid19symptomdataset

C Experimental setup (extra details)

Code and data are available attached as a supplement. They both will be made public upon acceptance.

Computational setup. All experiments were conducted using a 4 Xeon E7-4850 CPU with 512GB of 1066 Mhz main memory and 4 GPUs Tesla V100 DGXS 32GB. Our method implemented in PyTorch (implementation details in the appendix) trains on GPU in about 30 mins for one predictive task. Inference is takes only a few seconds.

Real-time forecasting. We follow the literature on evaluating epidemic forecasting methodologies (Shaman et al. 2013; Kamarthi et al. 2022b; Adhikari et al. 2019) and use the *real-time forecasting* setup. We simulate real-time forecasting by making models train *only* using data available until each of the prediction weeks and make predictions for 1 to 8 weeks ahead in the future. Data revisions in public health data are large and may affect evaluation and conclusions (Kamarthi et al. 2022a; Cramer et al. 2022), therefore, we utilize fully revised data following previous papers on methodological advances (Adhikari et al. 2019; Rodríguez et al. 2021b).

Evaluation details. Some models may make daily predictions while others weekly predictions. We follow CDC evaluation papers (Cramer et al. 2022; Reich et al. 2019) and convert all forecasts to weekly. For this, we sum over 7 days for COVID-19 and take a 7-day average for flu.

D Implementation details

D.1 Data Preprocessing

Feature scaling. Time-series of exogenous features can have wide range of values (e.g., number of confirmed cases vs percentage of people with COVID-like symptoms in social media surveys) Therefore, we scale all signals per each region for which we use standard scaling (normalization).

Time series for training. Although we may have long time series, we found our RNN works better with no chunking. As during training we have variable-length input sequences, we use a mask that is utilized when calculating the attention scores. As we follow the real-time forecasting setup, at inference time we use the complete input sequence thus we do not need a mask.

D.2 Architecture and hyperparameters

We describe in detail the hyperparameters for EINNs used for our experiments. As mentioned in Section 5, we used data from June 2020 to Aug. 2020 for model design and hyperparameter tuning.

• **Time module:** It is a feedforward network with input layer layers 40x40x40x20 followed by output layers 20x40x40x5, and activation function $\tanh$. Note that the input to this module is time, which is of dimension 1, but this is immediately transformed to 40 different signals via Gaussian Random Fourier feature mapping which then enter to the neural network. Between the input and output layers, we have our embeddings $\mathbf{e}_t$, which are of smaller size (20) than the other hidden layers. We make this selection because we want to make embeddings $\mathbf{e}_t$ and $\mathbf{e}_t^F$ to be as close as possible, and this is hard to achieve when we deal with high-dimensional embeddings as it has been noted in the Contrastive Learning literature (Liu et al. 2021). Passing $\mathbf{e}_t$ to the output layer gives us $\mathbf{s}_t$. With respect to the learnable ODE parameters, we a $\tanh$ transformation following (Yazdani et al. 2020). This ensures that the actual parameter values will be within their corresponding domain (0-1). In our experience, we found that initializing the ODE parameters with the ones found by the analytical solution works best.

• **Feature module:** As encoder, we used a bi-directional GRU with 2 layers and hidden states of dimension 32. As decoder, we use another bi-directional GRU with 1 layer and hidden states of dimension 32 and an feedforward output layer of 32x20 to obtain $\mathbf{e}_t^F$. To encourage transfer learning, we utilize shared layers between the time and feature modules. Therefore, the embedding $\mathbf{e}_t^F$ is passed to the output layer of the time module to obtain $\mathbf{s}_t^F$.

• **Loss weights:** All our results use loss weight of 1 except for the following losses. For our ODE loss $\mathcal{L}^{\text{ODE}}$ $\mathcal{L}^{\text{ODE}-F}$ we use weight loss of 10, and we weight the same in our monotonicity loss $\mathcal{L}^{\text{Mono}}$. Our parameter consistency loss $\mathcal{L}^{\text{Param}}$ is weighted with 0.001. Finally, we have a helper loss for the time module that ingests data from the analytical solution of the ODE, to which we put a weight of 0.1

E Evaluation metrics

As noted in Section 5, we used two different versions of Normalized Root Mean Squared Error (NR1 and NR2) and Normal Deviation (ND) following (Roy et al. 2021; Remy et al. 2021). Given prediction $\hat{y}_{w,k}$ at prediction week w for k-weeks ahead in the future, and the corresponding ground truth value $y_{w,k}$, our error metrics are the following:

$$\text{NR1} = \frac{\sqrt{\frac{1}{WK}\sum_{w,k}(y_{w,k}-\hat{y}_{w,k})^2}}{\frac{1}{NK}\sum_{w,k}|y_{w,k}|}$$

$$\text{NR2} = \frac{\sqrt{\frac{1}{WK}\sum_{w,k}(y_{w,k}-\hat{y}_{w,k})^2}}{\max(y_{w,k})-\min(y_{w,k})}$$

$$\text{ND} = \frac{\sum_{w,k}|y_{w,k}-\hat{y}_{w,k}|}{\sum_{w,k}|y_{w,k}|}$$

where W is the number of predictions weeks and K is number of weeks ahead in the future; for our setup we have $W = 14$ and $K = 8$, which makes 112 different predictions for model for a single region and makes 5376 predictions per model over all regions. Note: In the case of COVID-19, there are several states for which there was no death in a particular week, therefore, we add 1 death to the denominator to avoid numerical issues. In flu, we do not need this.

Regarding correlation, we use Pearson correlation over the sequence of 8 predictions in the future as per (Deng et al. 2020):

$$\text{PC} = \frac{\sum_k (y_k - \overline{y_k})(\hat{y}_k - \overline{\hat{y}_k})}{\sqrt{\sum (y_k - \overline{y_k})^2 \sum (\hat{y_k} - \overline{\hat{y_k}})^2}}$$

where $\overline{y_k}$ and $\overline{\hat{y}_k}$ are the mean values of the ground truth and the model's predictions, respectively.

Calibration of ODE. The target variables for COVID-19 and flu forecasting are different. In COVID-19, we want to calibrate/predict using COVID-associated mortality– which is more reliable than confirmed cases (Cramer et al. 2022)–, while in flu we use influenza-like-illness (ILI) counts, which is collected by the CDC. ILI measures the percentage of healthcare seekers who exhibit influenza-like-illness symptoms, defined as "fever (temperature of 100°F/37.8°C or greater) and a cough and/or a sore throat without a known cause other than influenza" (for Disease Control and Prevention 2020). Data on COVID-19 mortality can be directly associate to our SEIRM. However, to connect ILI to the SIS model, we need to estimate it based on the available variables. For this, we use ILI $\% = \frac{\beta(t) IS}{N} / (N \cdot \text{OR})$, where OR is the outpatients ratio, which is the proportion of the population that are outpatients in a given day. We set OR based on CDC flu facts [5].

F Extra forecasting results (vs. standard data-driven methods)

While the selection of our baselines is correct to address the main focus of the paper, we compare our method against standard data-driven baselines to contextualize our model's performance with the bigger picture of epidemic forecasting. Following (Cramer et al. 2022), the COVID Forecast Hub

[5] cdc.gov/flu/about/keyfacts.htm

Table 2: EINNs vs standard data-driven baselines. Top model per column is in bold.

	Short-term (1-4 wks)			Long-term (5-8 wks)			Trend correlation
Model	**NR1**	**NR2**	**ND**	**NR1**	**NR2**	**ND**	**PC**
Task 1: COVID-19 Forecasting (US National + 47 states)							
EINNs (ours)	**0.54**	**0.24**	**0.38**	**0.85**	**0.37**	**0.66**	**0.46**
COVID Forecast Hub baseline	0.80	0.36	0.62	0.93	0.40	0.74	NaN
Autoregressive model	0.78	0.35	0.60	1.25	0.55	0.92	0.03
Lasso model w/ features	0.87	0.39	0.66	0.89	0.38	0.67	-0.18
Task 2: Influenza Forecasting (10 HHS regions)							
EINNs (ours)	**0.53**	**0.27**	**0.37**	1.01	**0.42**	**0.73**	**0.68**
COVID Forecast Hub baseline	0.74	0.39	0.69	1.15	0.49	1.10	NaN
Autoregressive model	0.56	0.33	0.52	0.88	0.44	0.82	-0.19
Lasso model w/ features	0.60	0.36	0.55	**0.85**	0.43	0.80	-0.82

Table 3: Hyperparameter sensitivity for EINNs over states California, Georgia, Illinois, Texas, and US National.

		Short-term (1-4 wks)			Long-term (5-8 wks)			Trend correlation
Hyperparameter	**Value**	**NR1**	**NR2**	**ND**	**NR1**	**NR2**	**ND**	**PC**
w^{ODE}	1	0.41	0.50	0.35	0.52	0.73	0.48	0.68
	10	0.38	0.46	0.32	0.51	0.70	0.47	0.72
	100	0.34	0.39	0.29	0.57	0.73	0.50	0.55
w^{Transfer}	1	0.38	0.46	0.32	0.51	0.70	0.47	0.72
	10	0.39	0.46	0.33	0.53	0.70	0.49	0.75
	100	0.39	0.46	0.34	0.53	0.71	0.49	0.76

baseline is a persistence baseline which always predicts the past. We also an autoregressive and a LASSO model which also takes the same features as input. As we can see in Table 2, our method is the only one with consistently low error (NR1, NR2, ND) and high correlation. Note that the persistence baseline (COVID Forecast Hub baseline) predicts a constant trend, thus, correlation is not defined (NaN).

G Q5: Hyperparameter sensitivity

Overall, we found that most hyperparameters are not sensitive. The most sensitive ones are the weights for ODE loss for time module $\mathcal{L}^{\text{ODE}}$ and feature module $\mathcal{L}^{\text{ODE-F}}$, and embedding $\mathcal{L}^{\text{Emb}}$ and output losses $\mathcal{L}^{\text{Output}}$. To illustrate this, we vary the loss weight w^{ODE} which is applied to both $\mathcal{L}^{\text{ODE}}$ and $\mathcal{L}^{\text{ODE-F}}$ and another loss weight w^{Transfer} which is applied to both $\mathcal{L}^{\text{Emb}}$ and $\mathcal{L}^{\text{Output}}$.

We analyze hyperparameters sensitivity on five geographical regions in the uptrend of the COVID-19 Delta wave (over 2 months, specifically epidemic weeks 202048 to 202101) which is one of the most difficult to predict due to the unprecedented infectiousness of the Delta variant and shift in human behavior. In Table 3, we can see that EINNs performance is stable across different values of hyperparameters It is worth noting that it tuning is important to have a correct balance on these losses as increasing one weight on may improve error or correlation but degrade the one facet. As noted in the literature (Wang et al. 2021b), this is common when working with theory-based constrains and how to best proceed remains an open problem.