Leveraging Loop Polarity to Reduce Underspecification in Deep Learning

Donald Martin, Jr.* and David Kinney*

*Equal Contribution

Forthcoming in System Dynamics Review

Abstract

Deep learning provides a set of techniques for detecting complex patterns in data and is a critical component of the burgeoning artificial intelligence revolution, enabling transformative advancement in a variety of fields. However, when the causal structure of the data-generating process is underspecified, deep learning models can be brittle, lacking robustness to shifts in data-generating distributions. In this paper, we demonstrate that methods and concepts familiar to system dynamics modelers can be used to address this problem of brittleness, thereby improving the efficacy of deep learning systems. Specifically, we turn to loop polarity analysis as a tool for specifying the causal structure of a data-generating process, in order to encode a more robust understanding of the relationship between system structure and system behavior within the deep learning pipeline. We use simulated epidemic data based on an SIR model to demonstrate how measuring the polarity of the accumulations of the different feedback loops that compose a system can lead to more robust inferences on the part of neural networks, improving out-of-distribution performance and infusing a system-dynamics-inspired approach into the deep learning pipeline. This case study provides one example of how to leverage an understanding of the causal structure of a data-generating process to extract low-dimensional summary statistics that in turn allow us to build more robust deep learning pipelines. Code for this paper is available at https://github.com/davidbkinney/loop_polarity_underspecification.

Introduction

The term "deep learning" refers to a subfield of machine learning that uses a set of algorithms and computational architectures based on layers of neural networks for estimating unknown functions from data. While deep learning encompasses a wide variety of different techniques, at its core, deep learning aims to use training data to estimate a vector of weights, which is then used to parameterize a function that approximates a data-generating process. In other words, deep learning aims to use data to make inferences about the *structural* features of reality that generate those data. Deep learning has proven to be an extremely useful tool for both classification and prediction in a wide range of use cases and has become a critical component of the burgeoning AI revolution, enabling transformative advancement in a variety of fields. The domains in which deep learning has been successfully applied include but are not limited to: image recognition and classification (He et al., 2015), facial recognition (Schroff et al., 2015), fraud detection in

financial data (Mienye et al., 2024), diagnostic radiology (McBee et al., 2018), and flood forecasting (Nearing et al., 2024). Perhaps most famously, deep learning techniques are at the core of large language models (Brown et al., 2020), which are extremely powerful tools for predicting and generating fluent text responses to input prompts. Thus, deep learning has emerged as a ubiquitous technique for inference across a wide range of scientific domains.

Despite their success in many domains, applications of deep learning are also notoriously *brittle*. When deployed in a setting that is sufficiently different from the setting in which their training data were generated, the accuracy of deep learning algorithms can severely, if not completely, deteriorate (Goodfellow et al., 2016; Hendrycks and Dietterich, 2019; Barbu et al., 2019). D'Amour et al. (2020) argue convincingly that this brittleness is especially liable to occur in cases where a deep learning pipeline is *underspecified*. By a "deep learning pipeline," we mean a sequence of human decisions and technical components (e.g., training datasets, test datasets, training algorithms, or hardware) that lead a real-world problem to be understood, formulated as a deep learning problem (i.e., as a problem in which our goal is approximate an unknown data-generating function), and putatively solved by inducing a model or predictor that approximates the true data-generating process. In this paper, we hypothesize that concepts of accumulation and loop polarity developed in the system dynamics literature can be deployed in a deep learning pipeline to reduce underspecification, providing opportunities for system dynamics practices and practitioners to play critical roles in influencing and improving the broader AI ecosystem. Using a simulation-based computational experiment, we find evidence in favor of this hypothesis.

To introduce the concept of underspecification, D'Amour et al. begin with a definition of an *underspecified problem*, writing that "the solution to a problem is underspecified if there are many distinct solutions that solve the problem equivalently" (2022, p. 3). To illustrate via a toy example, suppose that, according to ground truth, the following facts all hold: 1) a person spends $500.00 on hot dogs, burgers, and ears of corn; 2) hot dogs cost $5.00 each, burgers cost $6.00 each, and ears of corn cost $4.00 each; 3) the person bought 100 food items (which include only hot dogs, burgers, and ears of corn) in total; and 4) the person bought three times as many ears of corn as they did hot dogs and hamburgers combined. If our

problem is to determine how many hot dogs (*x*), hamburgers (*y*), and ears of corn (*z*) they bought, then we can just solve the system of linear equations:

$$5x + 6y + 4z = 500$$

$$x + y + z = 100$$

$$x + y - 3z = 0$$

This problem is *not* underspecified, since there is a unique solution for *x*, *y*, and *z* that solves these three equations. If, however, we only knew that: 1) a person spends $500.00 on hot dogs, burgers, and ears of corn; 2) hot dogs cost $5.00 each, burgers cost $6.00 each, and ears of corn cost $4.00 each; and 3) the person bought 100 food items in total, then we would only be able to write the first two of the three equations above, and solving for *x*, *y*, and *z* would become an underspecified problem.[1]

From this definition of an underspecified problem, D'Amour et al. build up to a definition of an underspecified *pipeline* in deep learning. An underspecified deep learning pipeline is one that can produce multiple different solutions that satisfy the same evaluation criteria; i.e., the problem of solving the pipeline's evaluation criteria is underspecified. As D'Amour et al. put it:

> In the context of ML [i.e., 'machine learning,' which we take here to be inclusive of deep learning], we say an ML pipeline is underspecified if there are many distinct ways for the pipeline to produce a predictor that satisfies the pipeline's validation criterion equivalently, even if the specification of the pipeline (e.g., model specification, training data) is held constant (2022, p. 3).

Citing the example above, if a deep learning pipeline's validation conditions were such that a valid output for *x*, *y*, and *z* had to be a solution to all three equations, then that pipeline would *not* be underspecified, since only a single predictor would be valid (namely, whatever predicts the purchase of 50 hot dogs, 25 burgers, and 25 ears of corn). However, if we suppose instead that a pipeline's validation conditions were such that a valid output for any *x*, *y*, and *z* only had to be a solution to the first *two* equations in the linear system defined above, then that pipeline *would* be underspecified, since multiple predictors would yield valid solutions. More generally, when the deep learning pipeline yields a problem specification - which

[1] An example of this form is suggested by D'Amour et al. when they write that "the solution to an underdetermined system of linear equations (i.e., more unknowns than linearly independent equations) is underspecified, with an equivalence class of solutions given by a linear subspace" (2022, p. 3).

includes the validation criteria - that lacks a unique solution, deep learning algorithms are more likely to generate a putative model of the data-generating process that is not isomorphic to or representative of the "ground truth" data-generating structure. When such a model is deployed in a setting beyond the training data, its predictions are liable to be inaccurate or misguided, because the model ultimately lacks fidelity to the true data-generating process.

The underspecification problem in deep learning is well-known, and there have been many attempts to address it in the machine learning literature. Ross et al. (2017) introduce techniques to train predictive models that are "right for the right reasons," meaning that these models provide a generalizable explanation of why they produce the (accurate) predictions that they do. Specifically, they demonstrate that manually flagging some dimensions of parameter space as relevant or irrelevant to prediction (or iteratively restricting the model to using only some dimensions in an automated fashion) produces models that only generate predictions by exploiting parameters that are intuitively relevant to prediction or classification problem at hand. This addresses underspecification by eliminating from consideration models that are right for the wrong reasons in that they produce accurate predictions about their training data but fail to instantiate generalizable, explanatory models of the underlying phenomenon being modeled. Schölkopf (2022) has argued that deep learning algorithms should be designed not just to make accurate predictions but to accurately capture the underlying causal structure of the systems they are making predictions about, thereby reducing brittleness and increasing generalizability. More recently, Tsai et al. (2024) have also aimed to exploit the underlying causal structure of the data-generating process to identify proxy variables that act as reliable predictors across all domains.

The approach that we pursue here is similar in spirit to these previous works, in that we aim to exploit explanatory, causal models of data-generating processes in an attempt to reduce underspecification. However, whereas these previous approaches have tended to draw on acyclic causal models, we turn to the inherently feedback-driven, temporally structured causal models used in system dynamics. Specifically, we use the polarity of the accumulations of the feedback loops identified in stock-and-flow models as predictors of key system characteristics (in our case, the effective reproduction number of an epidemic at a given

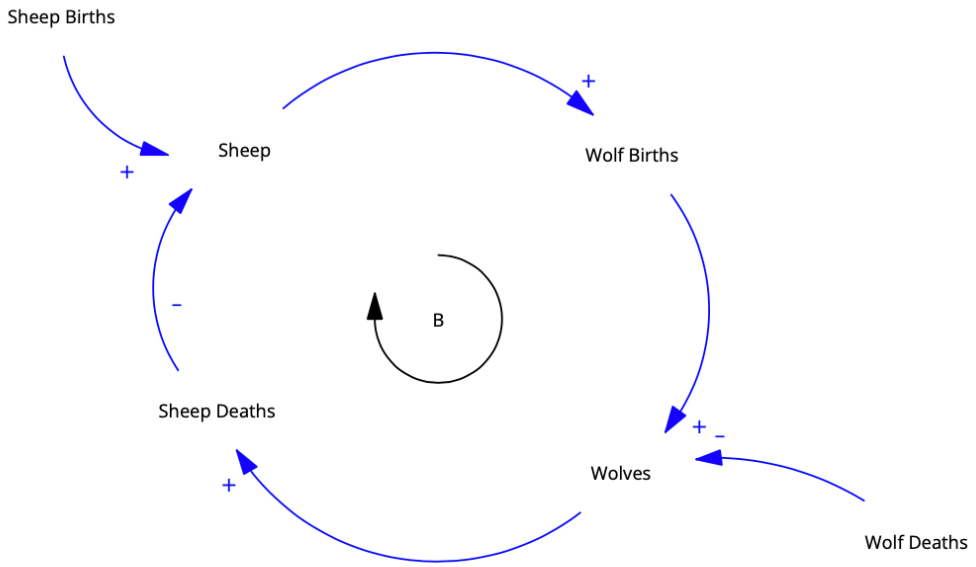

Fig. 1: Causal loop diagram due to Lin et al. (2020) representing a balancing (negative polarity) feedback loop formed by the predator-prey dynamics of an ecosystem of sheep and wolves.

period in time). We show that this system-dynamics-inspired approach improves the out-of-distribution performance of deep learning models; this is good evidence that we are reducing underspecification. To our knowledge, ours is the first study that explicitly aims to use loop polarity measurements as a way of reducing underspecification in deep learning models. While we present just this single case study here, we take our results to be illustrative of a broader point: formally representing the causal structure of a data-generating process in a stock-and-flow model can allow us to extract low-dimensional summary statistics that in turn allow us to build robust deep learning pipelines.

In contrast with deep learning, system dynamics uses *causal loop diagrams* and *stock-and-flow diagrams* to represent data-generating processes. Though the details of these models will be familiar to readers in system dynamics, we briefly present them here so as to contextualize our subsequent results. Causal loop diagrams are composed of three basic elements: nodes, connections, and feedback loops (Barbrook-Johnson and Penn 2022, p. 48). *Nodes* represent variables within the data-generating structure (i.e., any quantity in the structure that can be described as going up or down over time) and *connections* (or, to use the terminology of Sterman, 2000, *links*) represent asymmetric relations of causal influence from

one node to another, and can be either positive or negative depending on how changes in the variable at the tail of the arrow (i.e., the cause) lead to corresponding changes in the variable at the head of the arrow (i.e., the effect). The most important features of causal loop diagrams for our current purposes are *feedback loops*, which are collections of nodes and connections such that one can use the connections to trace a directed path from any node in the loop back to itself. Following Richardson (1995), we note that feedback loops can be either *balancing* or *reinforcing*. The valence of a feedback loop (i.e., whether it is balancing or reinforcing) is sometimes called the *polarity* of that feedback loop; balancing feedback loops have negative polarity, while reinforcing feedback loops have positive polarity.[2] Again following Richardson (1995, p. 68), we define the polarity of the feedback loop affecting a level x via the equation:

$$\operatorname{sgn}\left(\frac{\frac{dx}{dt}}{dx}\right),$$

where the function sgn(x) returns the value 1 if x>0, -1 if x<0, and 0 otherwise. It has become a core tenet of contemporary system dynamics that, depending on the structure of the data-generating process, certain loops in a causal loop diagram will be *dominant*, in the sense that their having a positive or negative polarity is a key driver of overall system behavior (Schoenberg et al. 2020, p. 159). This yields an overall account of system understanding in which *one understands a system when and because one understands how the feedback loops that compose the system drive overall system behavior*.

However, to fully understand the drivers of feedback loops when accumulation and delay is present, a more detailed representation of the dynamics of a system, of the kind provided by a *stock-and-flow diagram*, is required. See Fig. 2 for an example, also due to Lin et al. (2020), of a stock-and-flow diagram that provides a fuller representation of the dynamical system from which the balancing feedback loop between sheep and wolves shown in Fig. 1 emerges. Crucially, in a stock-and-flow diagram, flows can be represented mathematically as a time-dependent function (i.e., a time series). For instance, in Fig. 2, the

[2] Note that system dynamics practitioners are sometimes taught that if the number of connections or links in a feedback loop with negative sign is even, then the feedback loop will have positive polarity, with negative polarity otherwise. However, as Sterman notes, this is a heuristic rather than a rigorous definition of polarity (2000, p. 144).

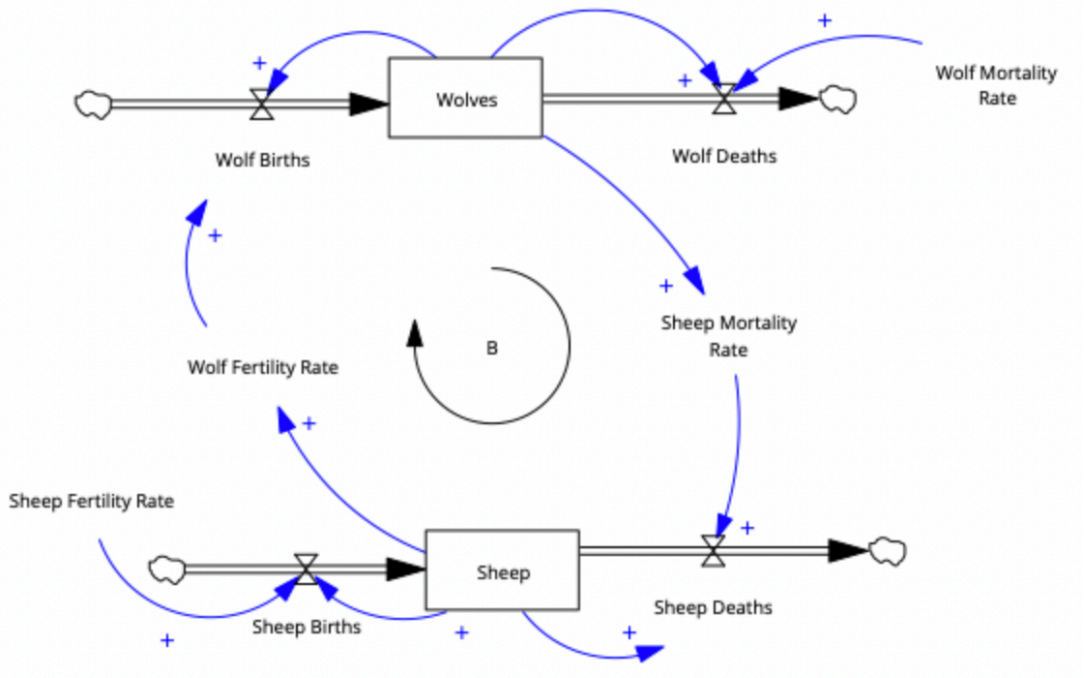

Fig. 2: Stock-and-flow diagram representing the same system as the causal loop diagram in Fig. 1.

birthing process is an inflow into the stock of wolves, and the dying process is an outflow from the stock of wolves. Letting $b_w(t)$ be the number of wolves born at a given time t and letting $d_w(t)$ be the number of wolves that die at t, the *accumulation* of the stock of wolves over a given time interval $[0,t]$ is $\int_0^t b_w(t')dt' - \int_0^t d_w(t')dt'$ (Sterman 2000, p. 194). In light of this mathematical definition, we can also calculate the polarity of an accumulation over a given time interval to determine whether, as the level of the accumulation increases or decreases, the difference between inflows and outflows moves in the same or the opposite direction. Polarities of accumulations will play a crucial role in our application of system dynamics concepts to the deep learning pipeline.

In what follows, we leverage stock-and-flow diagrams (SFDs), and the account of system understanding that they engender, to address the underspecification problem in deep learning. In particular, we will show how integrating an SFD representation of the data-generating process into the deep learning pipeline, while also selecting and preparing training data in a way that tracks the polarities of the loops that compose a system, can result in deep learning pipelines that are more completely specified and more robust to shifts in the underlying data-generating distributions (i.e., they are less brittle). Since brittleness is

understood to be a byproduct of underspecification in the deep learning pipeline, we take our results to be indicative of the capacity for systems thinking and SFDs to reduce underspecification.

The remainder of this paper proceeds as follows: in the next section, we provide more technical details on the problem of underspecification in the deep learning pipeline. After that, we introduce a case study involving an attempt to use deep learning to classify the polarity of the feedback loops that control the overall level of infections in an epidemic. Using a fairly simple version of the SIR (Susceptible, Infected, Recovered) model of epidemic dynamics, we can build an SFD and measure loop polarities in a way that improves the performance of a deep learning model on this classification task, as opposed to a more naive, baseline approach. We then discuss the significance of these results for the broader relationship between deep learning, system dynamics, and artificial intelligence. Code implementing our computational experiment is available at https://github.com/anon92189/loop_polarity_underspecification.

Underspecification in the Deep Learning Pipeline

We will use the term "deep learning" to refer to any algorithmic process for using a set of training data points to approximate a function from features to outputs. To be more precise, consider a set of training data points $\{(x_1, y_1), \ldots, (x_N, y_N)\}$, where each x_i is a feature vector in a d-dimensional real vector space and each y_i is a real number output. We assume that each y_i is computed from each x_i by finding the value of a data-generating function $f(x_i)$, perhaps with some irreducible stochasticity or noise affecting the computation. Deep learning aims to approximate this data-generating function by using training data to learn a vector of parameters θ, which defines a function f_θ that can be used to compute outputs from features. Many different architectures and algorithms exist for taking in training data and then using those data to learn weights that define a function f_θ that is a close approximation of the unknown data-generating function f, in the sense that for any pair (x_i, y_i) in the training data, the value of $f_\theta(x_i)$ is sufficiently close to $y_i = f(x_i)$. The details of these different algorithms (e.g., neural network training with gradient descent) need not detain us here. Since f_θ is an approximate representation of the

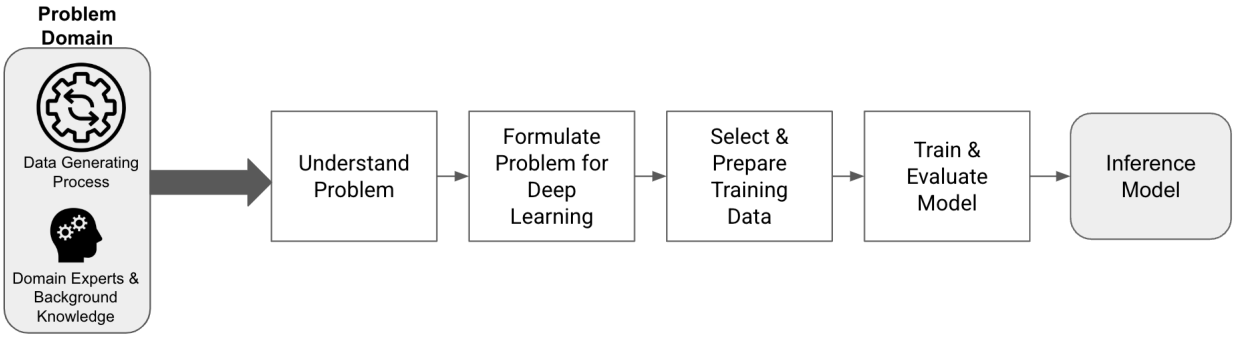

Fig. 3: A schematic representation of the deep learning pipeline.

true data-generating function f, it is sometimes referred to as a "model" of the data-generating function (also called a "data-generating process," or "DGP").

What *will* concern us are the human choices (e.g., the choice of training data, the choice of algorithm and architecture, and the choice of specific vectors to represent inputs and outputs of the DGP) that are made throughout the deep learning pipeline described in the previous paragraph. Indeed, before training data are even collected, practitioners must first leverage domain expertise and background knowledge about the problem domain in order to develop a robust qualitative, conceptual understanding of both the problem that they are trying to solve as well as the DGP their solution will estimate. From there, the problem can be formulated in terms that are tractable in the context of deep learning; that is, practitioners state the problem that they are trying to solve and establish evaluation criteria in terms of the outputs that they are trying to predict from a specific set of possible inputs/features that a DGP might instantiate. Next, practitioners select and prepare training data that are collected from the DGP, before feeding that data into any number of learning algorithms to train and evaluate a model of the DGP. Finally, once this model is sufficiently trained and evaluated (and found to have high accuracy), we end up with a model of the DGP that is capable of inferring outcomes from novel features (i.e., features not included in the training data set). See Fig. 3 for a schematic representation of this deep learning pipeline.

As described in the introduction, our aim in this paper is to use techniques from system dynamics to reduce the threat of underspecification in the deep learning pipeline. We do this using an example in which we classify the behavior of an epidemic based on the underlying data generated by that epidemic.

There are three specific techniques from the system dynamics literature that we deploy. First, at the stage of the pipeline where a problem is specifically formulated in terms amenable to deep learning, we will *also* use terms with a well-defined meaning in the system dynamics literature. Specifically, we will use the language of *the polarity of accumulations of feedback loops* during the problem understanding and formulation phases. Second, during the training data selection and preparation phase of the pipeline, we will use simulation techniques from system dynamics in order to generate synthetic training data according to an *a priori* causal theory of the data-generating process (an SIR model of an epidemic), where this causal theory is representable as a system of differential equations that are consistent with an SFD. Finally, and also during the training data selection and preparation phase, we will manually engineer the feature vectors of our training data to measure the polarity of different feedback loops in our causal model of the DGP. The result is a novel instance of the deep learning pipeline that is thoroughly imbued with the vocabulary and techniques of systems dynamics. Most importantly, we show that this reformulation of the deep learning pipeline is effective, significantly outperforming a naive, baseline approach to the same problem. We begin this demonstration in the next section, by introducing the details of our case study.

Case Study: Understanding an Epidemic

Suppose that a non-fatal epidemic is occurring within a given population. At each time period *t*, we have access to the number *S(t)* of susceptible individuals in the population (i.e., those who have not yet been infected), the number *I*(t) of currently infected individuals in the population, and the number *R(t)* of recovered individuals in the population. For any given time period, we are interested in a crucial, system-level behavior of the epidemic: is the **effective reproduction number** greater than or equal to one? In epidemiology, the effective reproduction number of an epidemic is the average number of new infections caused by an infected individual at a given time period (Cintrón-Arias 2009, p. 265). Following a standard compartmental model of an epidemic, we assume that the spread of the disease is governed by the following differential equations:

$$\frac{dS(t)}{dt} = -\frac{\beta(t)S(t)I(t)}{N} \qquad (1)$$

$$\frac{dI(t)}{dt} = \frac{\beta(t)S(t)I(t)}{N} - \gamma(t)I(t) \qquad (2)$$

$$\frac{dR(t)}{dt} = \gamma(t)I(t) \qquad (3)$$

where *N* is the total population, $\beta(t)$ is the transmission rate of the epidemic at time period *t* (i.e., the expected number of times per time period that each infected person transmits the disease to a susceptible person), and $\gamma(t)$ is the recovery rate of the epidemic at time *t* (i.e., the probability that an infected person recovers on a given day). The effective reproduction number at time period *t* is given by the equation:

$$R_e(t) = 1 + \frac{1}{\gamma(t)}\frac{1}{I(t)}\frac{dI(t)}{dt} \qquad (4)$$

.

As per the differential equations governing the system, the time derivative of the infection rate can be re-written as follows:

$$\frac{dI(t)}{dt} = I(t)\left(\frac{\beta(t)S(t)}{N} - \gamma(t)\right) \qquad (5)$$

.

Substituting the time derivative of the infection level in the equation for the effective reproduction number for the expression in the RHS of the equation immediately above and simplifying yields:

$$R_e(t) = 1 + \frac{\beta(t)S(t)}{\gamma(t)N} \qquad (6)$$

.

Thus, given a constant population, the effective reproduction number for a given time period *t* is a function of the transmission rate, recovery rate, and the size of the susceptible population at *t*. If the effective reproduction number at a given time period is greater than one, then each infected person

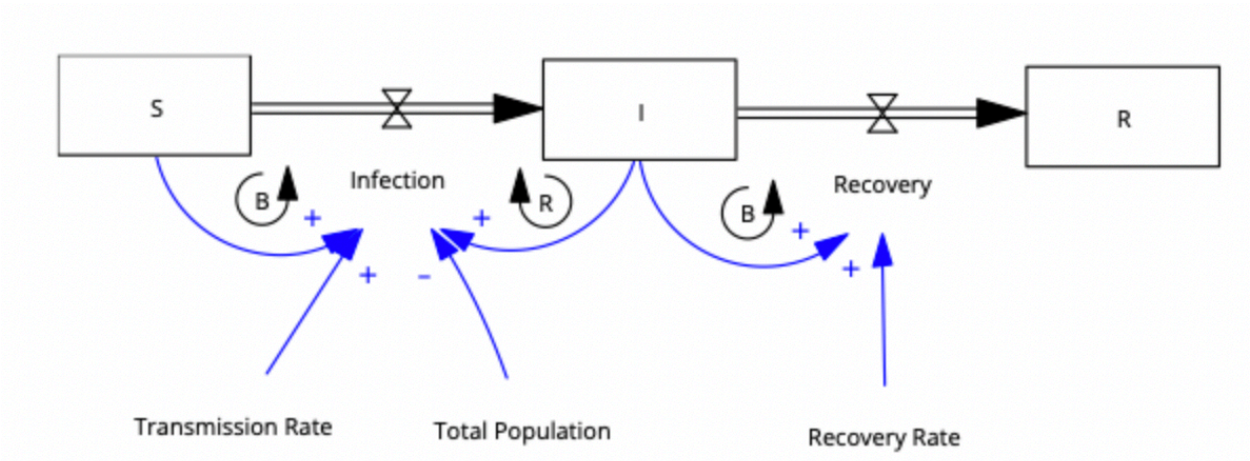

Fig. 4: SFD of the SIR model used for our computational experiments (Sterman 2000, p. 304).

produces more than one infection, and so the infection is expanding. If the effective reproduction number at a given time period is less than one, then each infected person produces more than one infection, and so the infection is contracting.

Fig. 4 shows a canonical SFD representation of an epidemic system governed by an SIR model, following the presentation in Sterman (2000, p. 304). There are three feedback loops in this model. The first is a balancing feedback loop between the susceptible population stock and the flow from the susceptible stock to the infected stock. As the susceptible population decreases, outflows to the infected population decrease, leading to a decrease in the rate of decline in the susceptible population. This loop is governed mathematically by the term $(\beta(t)S(t)I(t))/N$ in the differential equations. The second loop is a reinforcing loop between the infected population stock and the flow from the susceptible stock to the infected stock. As the infected population increases, inflows to the infected stock increase, thereby increasing the rate of change in the infected population. This loop is also governed mathematically by the term $(\beta(t)S(t)I(t))/N$ in the differential equations. Finally, there is the balancing loop between the infected population and the flow from the infected stock to the recovered stock. As the number of infected people increases, outflows to the recovered population increase, leading to a decrease in the rate of increase in the infected population. This loop is also governed mathematically by the term $\gamma(t)I(t)$in the differential

equations. Thus, we have two mathematical terms - $(\beta(t)S(t)I(t))/N$ and $\gamma(t)I(t)$- governing the loops in the SFD representation of the epidemic system governed by the SIR model. This fact will play a crucial role in the computational experiments described below.

We begin our case study by simulating 100 epidemics, each with 100 time periods, that are governed by the differential equations described above. Specifically, we use the `ODEInt` Python package to generate values for *S(t)*, *I(t)*, *R(t)* for time periods in the interval [1,100], assuming a fixed population of N = 1.0x10^6 and a single initial infection. At each period *t*, $\beta(t)$ and $\gamma(t)$ are generated by sampling from a probability distributions:

$$\beta(t) \sim \textsc{TruncNorm}\,(1, .5, [0, \infty]) \quad (\text{Mean: } 1, \text{ SD: } .5) \qquad (7)$$

$$\gamma(t) \sim \textsc{Beta}(2, \frac{2 - .2}{.1}) \quad (\text{Mean: } .1, \text{ SD: } .065) \qquad (8)$$

where $\textsc{TruncNorm}\,(1, .5, [0, \infty])$ is a truncated normal distribution with mean 1, standard deviation .5, lower bound 0 and infinite upper bound, and $\textsc{Beta}(2, (2 - .2)/.1)$ is a beta distribution with parameter $a = 2$ and $b = (2 - .2)/.1$, implying a mean of .1. At each time period *t* of each epidemic, we save the values *S(t)*, *I(t)*, and *R(t)*, as well as the value of a binary variable that takes the value 1 if the effective reproduction number is greater than or equal to 1, and zero otherwise. When calculating the effective reproduction number, we use the actual sampled values of $\beta(t)$ and $\gamma(t)$ so as to store the "ground truth" effective reproduction number (recall Equation 6 defining the effective reproduction number).

These simulations yield a data set with 10,000 rows (one for each time period for each epidemic) and columns for each value of *S(t)*, *I(t)*, *R(t)*, and the binary effective reproduction number (i.e., a binary variable representing whether the effective reproduction number is greater than or less than 1). We focus on the binary effective reproduction number because it is an example of the coarse-grained, qualitative information that many model users find most important: it tells us whether the epidemic is expanding or contracting (Cintron-Arias et al., 2011, p. 261). Here is an example of what this data set looks like:

Epidemic	*t*	*S(t)*	*I(t)*	*R(t)*	Binary Effective Reproduction Number
1	1	999,999	1	0	1
1	2	999,996	4	0	1
1	3	999,988	11	1	1
…	…	…	…	…	…
100	98	57	296	999,646	0
100	99	57	268	999,675	0
100	100	57	240	999,703	0

Table 1: Initial presentation of the data generated by our simulation.

Ultimately, our goal will be to use SIR data to predict the binary reproduction number across all time periods and epidemics. However, because we will build a classifier that predicts the binary effective reproduction number using transmission and recovery rates, which can only be estimated from data spanning multiple time periods, we will need to transform this data set slightly to turn it into a training set that can be used to train a neural network model to approximate the function mapping S, I, R data to the binary effective reproduction number for a given time period. Specifically, we produce a data set with 9,800 rows, with each row corresponding to a three-period time interval in a particular epidemic. Here is what this data set looks like for a single simulation:

Epidemic	*t*	*S(t-2)*	*I(t-2)*	*R(t-2)*	*S(t-1)*	*I(t-1)*	*R(t-1)*	*S(t)*	*I(t)*	*R(t)*	Binary Effective Reproduction Number
1	2	999,999	1	0	999,996	4	0	999,988	11	1	1
1	3	999,996	4	0	999,988	11	1	999,961	36	3	1
…	…	…	…	…	…	…	…	…	…	…	…
100	99	57	306	999,637	57	296	999,646	57	268	999,675	0
100	100	57	296	999,646	57	268	999,675	57	240	999,703	0

Table 2: Training data generated by our simulation.

It is this second data set that forms the basis of the question we ask in this case study: can we use this data set training data to build a neural network model that accurately predicts the binary effective reproduction number at time period *t* using only S, I, R data from time periods *t-2*, *t-1*, and *t*, even when parameters $\beta(t)$ and $\gamma(t)$ are sampled from very different distributions than those used to produce the training data? If we can build a model that is robust to these kinds of distribution shifts, then it will serve as evidence that we have gone some way toward mitigating the problem of underspecification.[3]

In the computational experiment presented herein, we will show that while a naive, baseline approach to learning the relationship between S, I, R data and the binary effective reproduction number fails to learn a model that is robust to a distribution shift, a system-dynamics-inspired approach *can* learn such a model. This provides evidence of the fecundity of system dynamics approaches for addressing the problem of underspecification in deep learning. In what follows, we describe the methodology of these computational experiments, before presenting the results.

Method

The Baseline Approach to Neural Network Training

In our baseline approach, we directly use the data shown in Table 2 to train a neural network to classify the binary effective reproduction rate of an epidemic at time period *t*, using the S, I, R data from periods *t-2*, *t-1*, and *t*. To clarify, recall that neural networks are trained on data of the form $\{(x_1, y_1), \ldots, (x_{9,800}, y_{9,800})\}$, where each x_i is a feature vector in a *d*-dimensional real vector space and each y_i is a real number output. Here, we let each feature vector x_i be a nine-component vector with components [*S(t-2)*, *I(t-2)*, *R(t-2)*, *S(t-1)*, *I(t-1)*, *R(t-1)*, *S(t)*, *I(t)*, *R(t)*], and let each y_i be the value of the binary variable representing whether the effective reproduction number at *t* is greater than or less than one. We train a neural network with nine input neurons, two nine-neuron hidden layers where each neuron has the rectified linear unit activation function, and a single-neuron output layer with a sigmoid activation

[3] For the sake of clarity, we emphasize that the effective reproduction number can be estimated for a single time period *t*. Indeed, each row of Tables 1 and 2 corresponds to a single time period *t*. However, in our deep learning pipeline, we use the S, I, R data for times *t-2*, *t-1*, and *t* to predict the effective reproduction number for a single time period *t*.

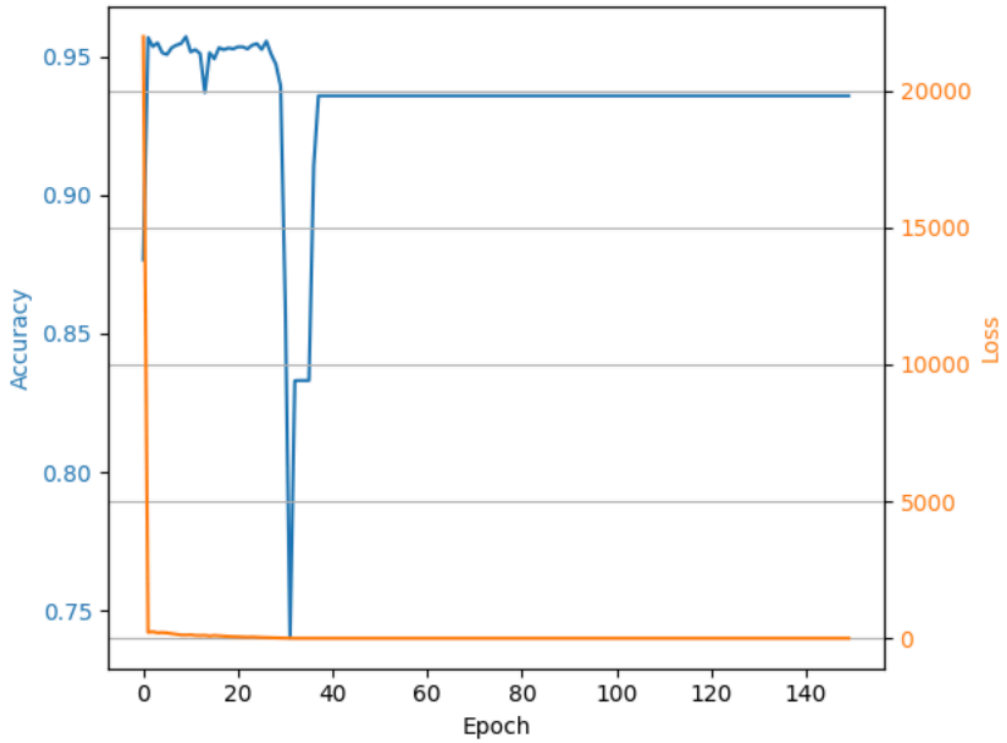

Fig. 5: Accuracy and Loss across all training epochs for the baseline approach.

function. To train this model to estimate a function from feature vectors to outputs, we initially feed it all 9,800 feature vectors and ask it to take a guess as to the actual value of the output vector for each input vector. Based on the accuracy of these initial guesses, we adjust the parameters and then repeat the process. Each repetition is called an **epoch**.

For each epoch, we measure both the **accuracy** and the **loss** of our model. To calculate accuracy, we begin by noting that the model outputs as its prediction of the binary effective reproduction number a real number between zero and one. So, for each epoch, we round the model's guess as to the binary effective reproduction number for each column of Table 2 to zero or one, and then calculate the percentage of predictions that match the ground truth to obtain the accuracy. Higher values indicate greater accuracy. To calculate loss, we measure the average value of the negative logarithm of the difference between the true value of the binary effective reproduction number and the prediction made by the model. For loss, *smaller* values are better. Fig. 5 shows the accuracy and loss across all 150 epochs when training the baseline model. As can be seen, accuracy settles around 92.5% as the model is trained, while loss bottoms out at approximately .15.

The SD-Inspired Approach to Neural Network Training

Like the baseline approach, we begin our SD-inspired approach with the training data shown in Table 2. However, we then engage in an additional data-processing stage in which new, SD-inspired variables are calculated from the S, I, R data. First, for each epidemic, we use S, I, R data from all 100 time periods to estimate the mean values of the parameters $\beta(t)$ and $\gamma(t)$ for that epidemic, using the Trust Region Reflective (Pedregosa et al., 2011) algorithm for estimating parameter values in non-linear systems, as implemented by the `curve_fit` function of the `scikit-learn` Python package. For epidemic ϵ, let $\hat{\beta}_\epsilon$ *and* $\hat{\gamma}_\epsilon$ be the estimated mean values of $\beta(t)$ and $\gamma(t)$. Next, recall that the feedback loops in the SFD for this epidemic system are governed by the mathematical terms $(\beta(t)S(t)I(t))/N$ and $\gamma(t)I(t)$. For a given time period *t* in epidemic *i*, we can estimate the values of these terms using the expressions $(\hat{\beta}_\epsilon S(t)I(t))/N$ and $\hat{\gamma}_\epsilon I(t)$. Next, we can use these estimates to estimate, for any time period *t*, the accumulation of both quantities up to *t*:

$$\mathcal{A}_1(t) := \int_0^t \frac{\hat{\beta}_\epsilon I(t')S(t')}{N}\mathrm{d}t' \qquad (9)$$

$$\mathcal{A}_2(t) := \int_0^t \hat{\gamma}_\epsilon I(t')\mathrm{d}t' \qquad (10)$$

.

Finally, for any three-time-period interval, we calculate the polarity of an accumulation $\mathcal{A}_j$ via the following equation:

$$\mathrm{Polarity}(\mathcal{A}_j[t-2:t]) = \mathrm{sgn}\left(\frac{(\mathcal{A}_j(t)-\mathcal{A}_j(t-1))-(\mathcal{A}_j(t-1)-\mathcal{A}_j(t-2))}{\mathcal{A}_j(t)-\mathcal{A}_j(t-2)}\right) \qquad (11)$$

.

At last, we are in a position to define our training data for the SD-inspired approach. As in the baseline condition, our training data has the form $\{(\tilde{x}_1, y_1), \ldots, (\tilde{x}_{9,800}, y_{9,800})\}$, where each $\tilde{x}_i$ is a feature vector in a *d*-dimensional real vector space (in the baseline approach, *d*=9; here, it will be the case that *d*=2) and each y_i is a real number output. However, in the SD-inspired condition, each feature vector $\tilde{x}_i$ is a two

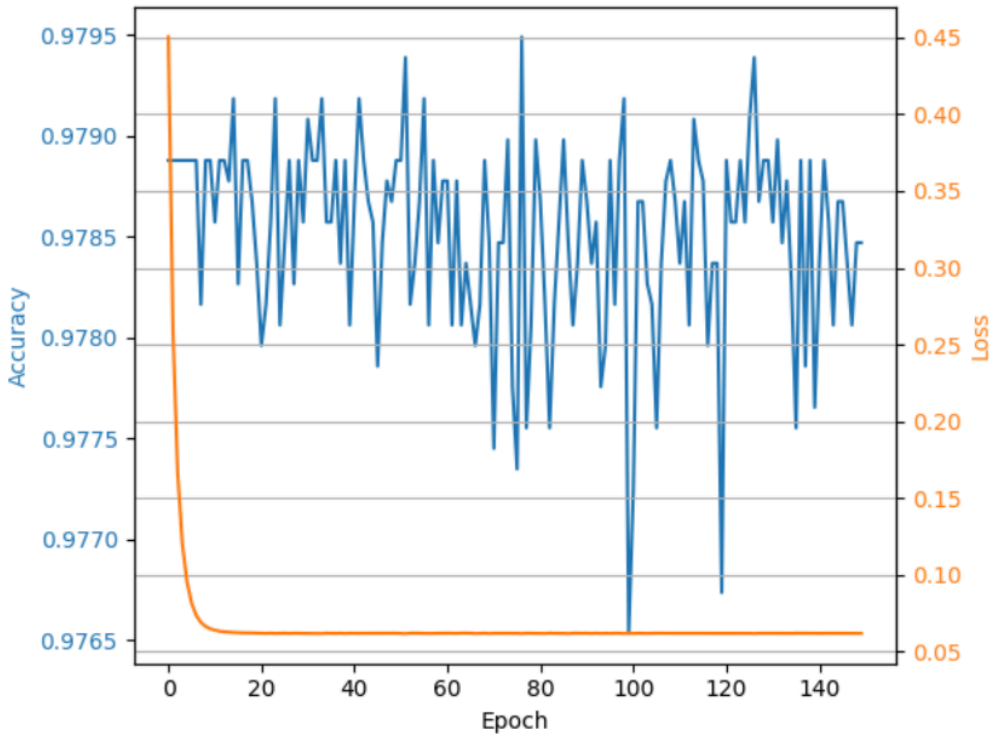

Fig. 6: Accuracy and Loss across all training epochs for the SD-inspired approach.

dimensional vector whose components are the polarities of the accumulations $\mathcal{A}_1$ and $\mathcal{A}_2$ over the interval from time period *t-2* to time period *t*, which each y_i remains the value of the binary variable representing whether the effective reproduction number at *t* is greater than or less than one.

We train a neural network with two input neurons, two two-neuron hidden layers where each neuron has the rectified linear unit activation function, and a single-neuron output layer with a sigmoid activation function. By matching the number of neurons in each hidden layer to the number of inputs, we ensure as much of a like-for-like structure as possible between this neural network and the one trained in the baseline condition. As in the baseline condition, we train this neural network for 150 epochs. The trends in accuracy and loss are shown in Fig. 6; training accuracy quickly converges to a value just below 98%, with minor fluctuations, while loss quickly approaches zero, bottoming out at approximately .07.

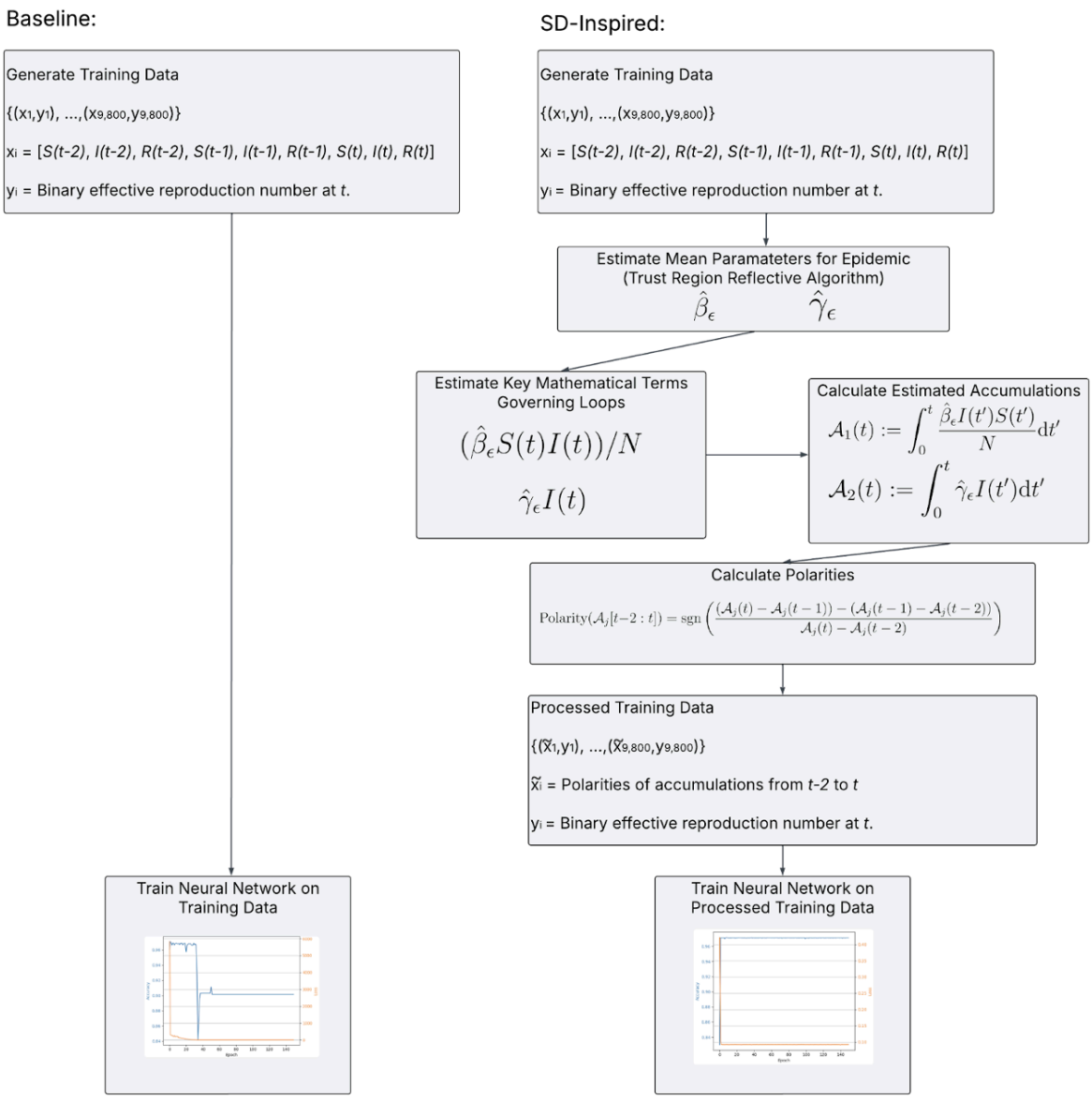

Fig. 7: Comparison of the Baseline and SD-Inspired approaches.

Comparing the Two Training Approaches

See Fig. 7 for a visual comparison of the two approaches. As should be clear from the figure, the sole difference between the baseline and SD-inspired approach is that the latter introduces several intermediate data-processing steps, with the goal of generating a training data set consisting solely of: 1) estimations of the polarities of the accumulations of key mathematical quantities governing loop behavior at each time period *t*, and 2) the ground truth of the binary effective reproduction rate at time *t*. Importantly,

all of the quantities used in these processing steps are *estimated* from data. One needs only the original S, I, R data to derive them. Thus, the two methods are eligible for apples-to-apples comparison; they begin with the same training data, and share the goal of finding an accurate algorithm for estimating the binary effective reproduction rate.

Testing the Two Approaches

Recall that our goal in this paper is to determine whether an SD-inspired approach to deep learning helps ameliorate the problem of underspecification. Recall in addition that one of the primary pieces of evidence for a model being underspecified is that it fails to perform well when tested in an out-of-distribution (OOD setting). To this end, we generated 20 simulated epidemics using the same techniques used to generate our training data, except that the transmission and recovery rates were now, at each time step, sampled from the following distributions:

$$\beta(t) \sim \textsc{TruncNorm}\,(5, .5, [0, \infty]) \quad (\text{Mean: } 5, \text{ SD: } .5) \tag{12}$$

$$\gamma(t) \sim \textsc{Beta}(2, \frac{2 - .02}{.01}) \quad (\text{Mean: } .01, \text{ SD: } .007) \tag{13}$$

Thus, the transmission rate is now sampled from a distribution with a mean of 5 (as opposed to the mean of 1 used to generate the training data) and the recovery rate is now sampled from a distribution with a mean of .01 (as opposed to the mean of .1 used to generate the training data). Thus, these 20 simulated epidemics constitute a decided distributional shift from the setting in which training data is generated. For each three-time-period interval in each of the 20 OOD epidemics, we use both the model trained using the baseline approach and the model trained using the SD-inspired approach to predict the binary effective reproduction rate from the S, I, R data. We then record the average accuracy of each model across all 20 epidemics.

Results

Fig. 8 shows the results of this test for both models across all 20 epidemics. Clearly, the SD-inspired approach generalizes well outside of distribution, with the SD-inspired model retaining its accuracy across the distribution shift, while the accuracy of baseline approach declines relative to its training accuracy in

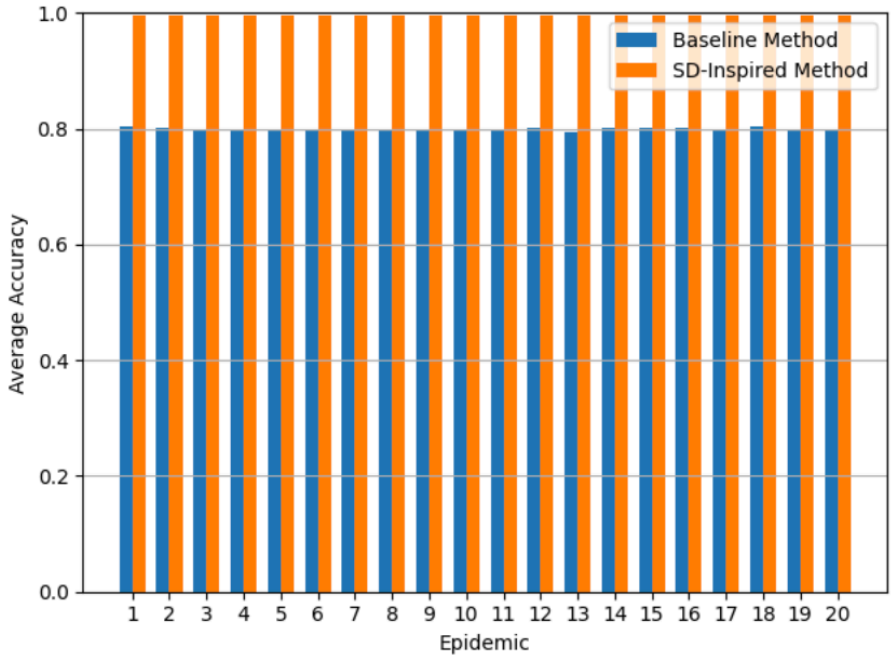

Fig. 8: Performance of the Baseline and SD-Inspired approaches on 20 out-of-distribution epidemics.

the out-of-distribution regime. Indeed, a paired *t*-test found that the difference in accuracy between the two models across all time periods in all 20 epidemics is highly statistically significant ($t = 280.70$, $p \approx 0$). Note further that the comparatively poor performance of the baseline model is *not* attributable to a paucity of training data. When we scale up the training data to include 1,000 simulated epidemics (as opposed to 100 in our original experiment) we do not see a change in results.

Discussion

Our findings from this computational experiment have broader implications for the optimal functioning of the deep learning model development pipeline. As discussed throughout this paper, at a high level of abstraction, a typical workflow in developing deep learning applications involves: 1) observing data from a system, 2) devising a mathematical representation of that data, 3) training a neural network model on that data, and 4) using that data to make OOD classifications. Our results demonstrate the importance of data representation in the success of this pipeline. Choices that we make about how data is represented as it is measured and collected influence the training of neural network classifiers, which in turn influence the success of these models in OOD testing scenarios. Importantly, implementing the polarity

scheme for representing data *requires domain knowledge of the kind that is emphasized in the practice of system dynamics*. Namely, one must know the summands of the ODEs that represent the dynamics of a system, recognize their accumulations as corresponding to the feedback loops that compose the overall data-generating system, and know that measuring their polarity is key to inferring the overall polarity of the number of infected people. This is a perspective on the data that requires some background in epidemiology and the modeling of dynamical systems; unlike the raw data representation, it cannot be measured from an entirely naive perspective on the nature of the data-generating process.

The perspective on the nature of dynamical systems that recommends decomposing those systems into feedback loops and measuring the polarity of those loops is not one that is well-established in existing deep learning practice. The results here indicate that in at least some contexts, this perspective *should* be established, as it can enable data representations that allow for more accurate classification of OOD data. To this end, we recommend that, throughout the development of the deep learning pipeline and the development of artificial intelligence applications more broadly, developers should draw on the literature in system dynamics. They should also receive input - in the form of causal theories - from system dynamics practitioners, problem domain experts, and stakeholders. This is especially true in the data representation stage of the pipeline, where the emphasis is not on designing optimal architectures for function approximation, but is instead on how to interpret and optimally represent the outputs of a data-generating process. It is at this earlier (but still deeply important) stage that domain-specific causal theories with reduced epistemic uncertainty about the nature of the data-generating can be leveraged to enable better OOD performance.

Note that a certain amount of systems dynamics knowledge was required in order to implement our enhanced, SD-inspired deep learning pipeline. To implement this pipeline, one must be able to articulate mathematically (i.e., in terms of differential equations) the structure of the data generating process. One must also have a qualitative sense of the feedback loops (as captured by CLDs and stock-and-flow models) that could be potentially driving system behavior. Finally, one must be able to merge these quantitative and qualitative approaches in order to know which parameters of the system to estimate and how to use these

estimations to calculate the polarities of the accumulations of the loops. Given the success of this approach that we have demonstrated here, and in light of both deep learning's increasing prevalence and its well-known difficulties with underspecification and brittleness, we close by encouraging system dynamics practitioners to seek opportunities to use their expertise to improve the robustness of real-world deep learning pipelines. Indeed, we hope that the example given here will empower systems dynamics practitioners to influence and improve the model-building pipeline fueling the current AI revolution.

Thus, even though our choice to focus on the polarity of the accumulations of loops may be at least somewhat idiosyncratic to the case of the SIR model, one can still draw a more general conclusion from what we have presented. By representing the causal structure of the DGP using a stock-and-flow diagram, we were able to extract summary statistics from our data that were highly informative as to overall system behavior. These summary statistics measured *structural* features of the DGP. By training a neural network using these statistics, we effectively trained a neural network to encode an understanding of the structure of the DGP, rather than merely encoding statistical patterns in the data produced by the DGP. Moreover, these summary statistics had *lower* dimensionality than the summary statistics extracted from the raw data, which meant that the neural network that we trained to use these summary statistics to classify system behavior had fewer parameters, and while still performing better than the model trained on raw SIR data. We take these results to demonstrate how taking an SD-inspired approach to deep learning can help us not only the robustness, but also the efficiency, of the deep learning pipeline.

There are several concrete avenues for future work that would build on our findings here. First, we note that the polarity framework is just one example of the many ways that experts in system dynamics have formalized the concept of understanding system behavior. In future work, we aim to explore whether other SD-inspired methods can show similar fecundity in reducing underspecification in deep learning. For example, one could extract summary statistics from the data produced by a system quantifying features other than the polarities of the accumulations of loops. These features should still be mappable to qualitative features of a stock-and-flow model. Examples include, but are not limited to, loop gain (Kampmann 1996), link scores and loop scores (Schoenberg et al., 2020), the relative delay of a path through a stock-and-flow

model (Schoenenberger et al., 2021), the pathway participation metric measuring the impact of a loop of stock behavior (Mojtahedzadeh et al., 2004), and the loop impact measure that also quantifies the impact of a loop on stock behavior (Hayward and Boswell, 2014). These summary statistics could then be subjected to a sensitivity analysis to measure their correlation with important behaviors of the overall system (Bier, 2011). Promising correlates could then be used within deep learning pipelines that produce neural networks that ultimately encode a structural understanding of a given DGP. These kinds of sensitivity analyses will not only facilitate the construction of robust, SD-inspired deep learning pipelines, but will also help SD practitioners quantify and improve the robustness of their own SD models, independently of any deep learning applications. This potential avenue for future work does raise an important limitation for our findings here. It may be that one can use SD-inspired models to identify useful summary statistics, but that the available data does not allow those statistics to be calculated or estimated. In practice, this will, in at least some contexts, be a significant constraint on the applicability of our methods.

Second, we plan to extend our results beyond the synthetic setting and demonstrate similar performance of our approach on real-world data sets. Third, while we have demonstrated the feasibility of our approach in a specific case study, we hope in the near future to both extend our approach to other dynamical systems beyond the SIR model, and to provide a theoretical guarantee or closed-form proof of the general ability of the our framework to improve out-of-distribution classification of system behavior. Fourth and finally, we note that we have focused here on system *understanding*, which we operationalize as the ability to classify the global behavior of a system at a point in time based on the behavior of its component parts at that same point in time. In future work, we plan to explore whether the techniques developed here can be extended to the predictive setting, and used to anticipate and predict the temporal evolution of a system.

References

Barbrook-Johnson, P., & Penn, A. S. (2022). Causal loop diagrams. In *Systems Mapping: How to build and use causal models of systems* (pp. 47-59). Cham: Springer International Publishing.

Barbu, A., Mayo, D., Alverio, J., Luo, W., Wang, C., Gutfreund, D., ... & Katz, B. (2019). Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. *Advances in neural information processing systems*, *32*.

Bengio, Y., Goodfellow, I., & Courville, A. (2016). *Deep learning* (Vol. 1). Cambridge, MA, USA: MIT press.

Bier, A. B. (2011). Copy of Sensitivity Analysis Techniques for System Dynamics Models of Human Behavior (No. SAND2011-1788C). Sandia National Lab.(SNL-NM), Albuquerque, NM (United States).

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020, December). Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (pp. 1877-1901).

Cintrón-Arias, A., Castillo-Chávez, C., Bettencourt, L. M., Lloyd, A. L., & Banks, H. T. (2009). The estimation of the effective reproductive number from disease outbreak data. Mathematical Biosciences & Engineering, 6(2), 261-282.

D'Amour, A., Heller, K., Moldovan, D., Adlam, B., Alipanahi, B., Beutel, A., ... & Sculley, D. (2022). Underspecification presents challenges for credibility in modern machine learning. *Journal of Machine Learning Research*, *23*(226), 1-61.

Hayward, J., & Boswell, G. P. (2014). Model behaviour and the concept of loop impact: A practical method. System Dynamics Review, 30(1-2), 29-57.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

Hendrycks, D., & Dietterich, T. (2019). Benchmarking neural network robustness to common corruptions and perturbations. arXiv preprint arXiv:1903.12261.

Kampmann, C. E. (1996/2012). Feedback loop gains and system behavior (1996). System Dynamics Review, 28(4), 370-395.

Le, Q. V. (2013, May). Building high-level features using large scale unsupervised learning. In *2013 IEEE international conference on acoustics, speech and signal processing* (pp. 8595-8598). IEEE.

Lin, G., Palopoli, M., & Dadwal, V. (2020). From causal loop diagrams to system dynamics models in a data-rich ecosystem. Leveraging data science for global health, 77-98.

McBee, M. P., Awan, O. A., Colucci, A. T., Ghobadi, C. W., Kadom, N., Kansagra, A. P., ... & Auffermann, W. F. (2018). Deep learning in radiology. Academic radiology, 25(11), 1472-1480.

Mienye, I. D., & Jere, N. (2024). Deep learning for credit card fraud detection: A review of algorithms, challenges, and solutions. IEEE Access.

Mojtahedzadeh, M., Andersen, D., & Richardson, G. P. (2004). Using Digest to implement the pathway participation method for detecting influential system structure. System Dynamics Review: The Journal of the System Dynamics Society, 20(1), 1-20.

Nearing, G., Cohen, D., Dube, V., Gauch, M., Gilon, O., Harrigan, S., ... & Matias, Y. (2024). Global prediction of extreme floods in ungauged watersheds. Nature, 627(8004), 559-563.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. the Journal of machine Learning research, 12, 2825-2830.

Richardson, G. P. (1995). Loop polarity, loop dominance, and the concept of dominant polarity (1984). *System Dynamics Review*, 11(1), 67-88.

Ross, A. S., Hughes, M. C., & Doshi-Velez, F. (2017). Right for the Right Reasons: Training Differentiable Models by Constraining their Explanations. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (pp. 2662-2670). International Joint Conferences on Artificial Intelligence Organization.

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 815-823).

Schoenberg, W., Davidsen, P., & Eberlein, R. (2020). Understanding model behavior using the Loops that Matter method. *System Dynamics Review*, *36*(2), 158-190.

Schoenenberger, L., Schmid, A., Tanase, R., Beck, M., & Schwaninger, M. (2021). Structural analysis of system dynamics models. Simulation Modelling Practice and Theory, 110, 102333.

Schölkopf, B. (2022). Causality for machine learning. In *Probabilistic and causal inference: The works of Judea Pearl* (pp. 765-804).

Sterman, J. (2000). *Business dynamics*. Irwin/McGraw-Hill.

Tsai, K., Pfohl, S. R., Salaudeen, O., Chiou, N., Kusner, M., D'Amour, A., ... & Gretton, A. (2024, April). Proxy methods for domain adaptation. In *International Conference on Artificial Intelligence and Statistics* (pp. 3961-3969). PMLR.

Verdonck, T., Baesens, B., Óskarsdóttir, M., & van den Broucke, S. (2024). Special issue on feature engineering editorial. *Machine learning*, *113*(7), 3917-3928.