Leveraging Loop Polarity to Reduce Underspecification in Deep Learning

Donald Martin, Jr.[1] and David Kinney[1,2]

[1] Google Research

[2] Washington University in St. Louis

Abstract

Deep learning provides a set of techniques for detecting complex patterns in data. 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 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 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.

Introduction

The term "deep learning" refers to a set of algorithms and computational architectures for estimating unknown functions from data. These algorithms and architectures have powered significant advances in artificial intelligence, enabling AI applications that display human-level abilities in tasks ranging from image recognition (Le et al., 2013) to predicting the next word in a natural language sentence (Brown et al., 2020). While deep learning encompasses a wide variety of different techniques, at their core, deep learning aims to use training data to estimate a vector of weights, which are 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.

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 that lead a real-world problem to be understood, articulated, and putatively solved using the techniques of deep learning. An *underspecified* deep learning pipeline is one in which the problem that is formulated and solved is one with multiple solutions. As D’Amour et al. put it:

> In general, the solution to a problem is underspecified if there are many distinct solutions that solve the problem equivalently. For example, 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. In the context of ML [‘machine learning,’ which we take here to be synonymous with ‘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 (2020, p. 3).

When the deep learning pipeline yields a problem that lacks a unique solution, deep learning algorithms will typically only identify *one* of these solutions. This, in turn, renders those algorithms liable to generate a putative model of the data-generating structure 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 field of system dynamics has developed its own vocabulary and techniques for understanding and representing data-generating structure. Here, we focus specifically on *causal loop diagrams*. Causal loop diagrams are composed of three basic elements: nodes, connections, and feedback loops (Barbrook-Johnson and Penn 2017, 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). *Connections* represent asymmetric relations of causal influence from one node to another, and can be either positive or negative depending on whether increases in the variable at the tail of the arrow (i.e., the cause) lead to decreases in the variable at the head of the arrow (i.e., the effect). Finally, *feedback loops* 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. Feedback loops can be either *balancing* or *reinforcing*, depending on whether increases in the value of the variable represented by a node in a feedback loop lead to a dampening down

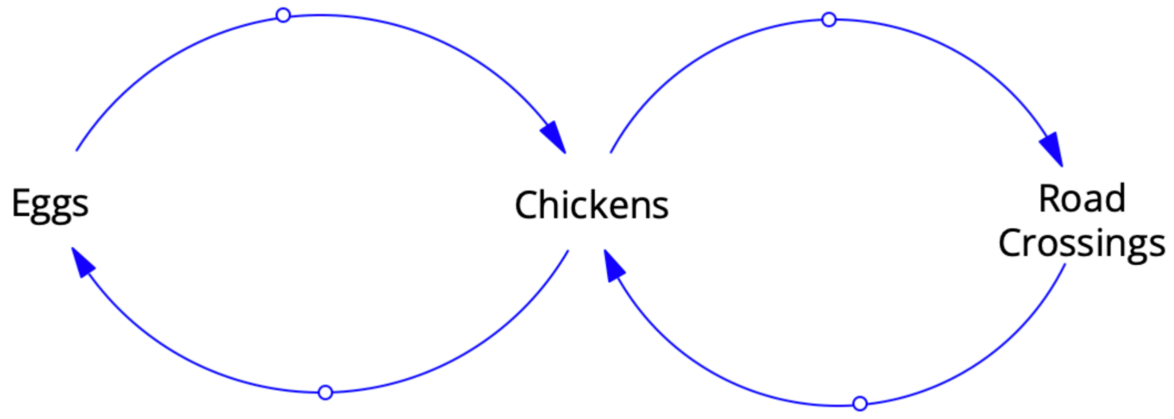

Fig. 1: Causal loop diagram from Sterman (2000, p. 14) showing a toy example of feedback loops.

in the rate of growth in the value of that variable (balancing loop) or an increase in the rate of growth in the value of that variable (reinforcing loop). 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. In what follows, we will provide a more formal definition of loop polarity.

To illustrate, consider the classic toy example of a system composed of two causal feedback loops given by Sterman (2000, p. 14), and shown in Fig. 1. In this example, the nodes ‘Eggs’ and ‘Chickens,’ and the connections between these nodes, form a positive-polarity feedback loop: as more eggs are laid, more chickens are born, which leads to a growth in the rate of increase in eggs being laid, which leads to a growth in the rate of chickens being born, and so on. By contrast, the nodes ‘Chickens’ and ‘Road Crossings,’ and the connections between these nodes, form a negative-polarity feedback loop: as more chickens are born, more chickens cross the road, where some are hit by cars, leading to a decrease in the rate at which chickens are born, and so on.

A core tenet of contemporary system dynamics is 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*. In this paper, we leverage causal loop diagrams - and the account of system understanding that they engender - to address the underspecification problem in deep learning. In particular, we will show how integrating a causal loop diagram 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 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 through causal loop diagrams 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 loop that controls 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 a causal loop diagram 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 approach. We then discuss the significance of these results for the broader relationship between deep learning, system dynamics, and artificial intelligence.

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 $\{(x_1, y_1), ..., (x_N, y_N)\}$, where each x_i is a vector in an *n*-dimensional real vector space and each y_i is a real number. We assume that each y_i is computed from each x_i by finding the value of a data-generating function $\widehat{f}(x_i)$, perhaps with some irreducible stochasticity or noise affecting the

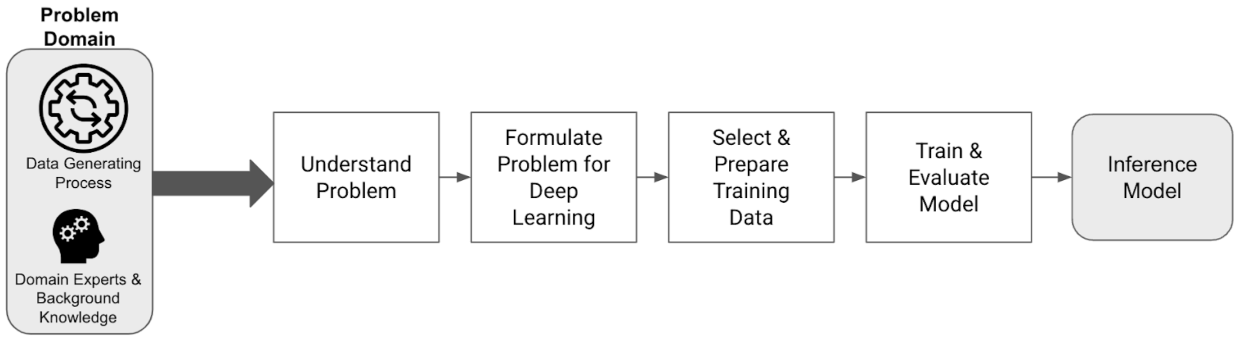
Problem Domain
Data Generating Process
Domain Experts & Background Knowledge
Understand Problem
Formulate Problem for Deep Learning
Select & Prepare Training Data
Train & Evaluate Model
Inference Model

Fig. 2: A schematic representation of the deep learning pipeline.

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 $\widehat{f}$, 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 $y_i = \widehat{f}(x_i)$. 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 true data-generating function $\widehat{f}$, it is sometimes referred to as a "model" of the data-generating function.

What *will* concern us are the human choices that are made throughout the deep learning pipeline described in the previous paragraph. Indeed, before training data are even collected, researchers must first identify the problem that they are even trying to solve via deep learning, by identifying an unknown data-generating function (also called a "data generating process") that they are trying to estimate and by leveraging domain expertise and background knowledge to better understand that problem domain. This gives rise to a qualitative, conceptual understanding of the problem that they are trying to solve. From there, the problem can be formulated in terms that are tractable in the context of deep learning; that is, researchers state the problem that they are trying to solve in terms of the outputs that they are trying to predict from a specific set of possible features that a system might instantiate. Next, researchers select and prepare training data that are collected from the system, before feeding that data into any number of learning algorithms to train and evaluate a model of the data-generating process. Finally, once this model is sufficiently trained and evaluated (and found to have high accuracy), we end up with a model of the data generating process that is capable of inferring outcomes from novel features (i.e., features not included in the training data set). See Fig. 2 for a schematic representation of this deep learning pipeline.

As a toy example of the deep learning pipeline, suppose that we want to understand the problem of how a person's physiology affects their ability to exercise vigorously. We first gain an understanding of the problem by adopting a simple operationalization of a person's physiology in terms of their height in centimeters and their weight in kilograms. Thus, each person can be represented as a two-dimensional vector, with one component representing their height and the other representing their weight. We operationalize a person's ability to engage in vigorous exercise using their VO2 max, or the maximum rate at which they can consume oxygen during physical activity. Now we are in a position to formulate the problem of understanding the relationship between physiology and exercise ability as a problem for deep learning: can we use deep learning to approximate the true data-generating function that takes as input a vector representing a person's height and weight and outputs a real number representing their VO2 max. To attempt to solve this problem, we might select and prepare training data by taking some sample of a population and measuring their height, weight and VO2 max, before using a deep learning algorithm to train and evaluate a model that eventually produces an accurate function (in the sense that it has high accuracy *when evaluated on the training data*) mapping height-weight vectors to VO2 max values.

The problem with this application of the deep learning pipeline (and, indeed, with many applications of this pipeline) is that the problem being solved is very likely to be underspecified. That is, for any given choice of algorithm and architecture for finding a vector of weights to parameterize a function that approximates the true data-generating process relating physiology to exercise ability, there are likely *many* different weight vectors that will parameterize this function equally well, in the sense that each weight vector will yield a function that is highly accurate within the context of the training data.

There are many reasons why such underspecification can occur. To begin with, given the rich diversity of ways in which physiology (even when measured as crudely as just considering height and weight) can affect VO2 max, the training data may not be representative enough to include instances of all of these interactions. Thus, it may be that multiple weight vectors parameterize a function to fit the training data equally well, but if more (and more diverse) data were collected, then the solution space for the approximation problem would be narrowed. In the absence of such a narrower solution space, the approximation problem being solved (and thus, the deep learning pipeline) remains underspecified. In addition, only two variables (height and weight) are measured, meaning that any choice of parameters will not be sensitive to other physiological variables that affect VO2 max; were more variables measured, it might be difficult to find multiple distinct parameter vectors that capture all of the interactions between these variables and their effects on VO2 max. This would also narrow the solution space and reduce underspecification.

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 aim to 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 feedback loops* during this problem-definition phase. 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 causal loop diagram. 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 data-generating process. 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 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 every time step *t*, we have access to the number (*S*) of susceptible individuals in the population (i.e., those who have not yet been infected), the number (*I*) of currently infected individuals in the population, and the number (*R*) of recovered individuals in the population. Over any given interval of time, we are interested in a crucial, system-level behavior of the epidemic: is the rate of the overall level of infected individuals behaving as though it is part of a balancing or reinforcing feedback loop? In other words, as the total number of infected individuals in a population goes up (down) during a given temporal interval, does the rate of change in the level of infections also increase (decrease) during the same interval? Assuming that we know the level of infections *I(t)* at any time step *t*, we can help ourselves to a simple calculation of this polarity value over the three-step time interval [*t* : *t+2*]:

$$\text{Polarity}(I[t:t+2]) = \text{sgn}\left(\frac{(I(t+2)-I(t+1))-(I(t+1)-I(t))}{I(t+2)-I(t)}\right),$$

where the function sgn(x) returns the value 1 if x>0, -1 if x<0, and 0 otherwise. In the toy example that follows, we will set up our simulations so that the polarity of the overall level of infections is easily calculated using the equation given above. However, in real-world cases, these polarities may not be as easily calculated, and are of considerable epidemiological importance (Centers for Disease Control and Prevention, 2021). Thus, we are interested in the ability of a deep learning algorithm to classify the level of infections in an epidemic as being of positive or negative polarity from measurable data.

Moreover, even if quantities like the polarity of the level of infections in an epidemic over time are easily calculable by humans, we remain interested in whether deep learning models can learn to perform the same calculations. That is, given only training data consisting of micro-level observations of an epidemic system over some time interval and macro-level observations of the same system over a period of time, can a deep learning architecture learn the relationship between micro-level components of a system and macro-level system behavior? If so, then the deep learning model could be said to instantiate a systems-thinking-style understanding (i.e., understanding how component parts of a system drive overall system behavior) of the data-generating process behind an epidemic. As we will see below, it turns out that on a naive approach to the selection and preparation of training data, the deep learning models we study here have, at best, a limited ability to instantiate this kind of systems-thinking approach to understanding an epidemic. In what follows, we illustrate these limitations of a naive approach, before showing how an approach that takes more seriously how the insights gleaned from a causal loop diagram can inform our selection of training data and reduce underspecification in the deep learning pipeline.

Using Deep Learning to Understand an Epidemic: The Naive Approach

As per the diagram in Fig. 2, the first step in the machine learning pipeline is to understand the problem to be solved. In our case study, this is straightforward: we want to use more fine-grained features of an epidemic that are observed over a given time interval to determine the overall polarity of the level of infections over the same time interval. Following the arrows of the diagram in Fig. 2, the next step is to formulate this more precisely as a problem for deep learning. One natural rephrasing of our problem statement for deep learning, which we will adopt here, is the following: can we train a neural network to predict, based on more fine-grained observations of an epidemic system across a time-step interval, the polarity of the level of infections across the same three-time-step interval?

Next, we enter the crucial phase of selecting and preparing training data. Following the naive approach, we will assume that our training data consist of pairs in which the first element represents the observed numbers of susceptible, infected, and recovered people in the population across three time steps, and the second element consists of a single number representing the polarity of the level of infections over the same time step interval. To actually generate this data, we first assume that numbers of susceptible, infected, and recovered people in the population are generated by the following set of differential equations, where $\Lambda(t)$ is the birth rate in the population at time *t*, $\mu(t)$ is the death rate in the population at time *t* (note that because the epidemic is assumed to be non-fatal, this death rate is the same across the populations of susceptible, infected, and recovered people), $\gamma(t)$ is the recovery rate in the population at time *t*, $\beta(t)$ is the average number of interactions between susceptible and infected people at time *t*, and N is the total size of the population:

$$\dot{S}(t) = \Lambda(t)N + \mu(t)S(t) + \frac{\beta(t)I(t)S(t)}{N}$$

$$\dot{I}(t) = \frac{\beta(t)I(t)S(t)}{N} + \gamma(t)I(t) + \mu(t)I(t)$$

$$\dot{R}(t) = \gamma(t)I(t) + \mu(t)R(t)$$

We assume further that at each time step the birth rate, death rate, recovery rate, and average number of interactions are generated by sampling from the following probability distributions:

$$\Lambda(t) \sim \text{Beta}(2, \frac{2-.0002}{.0001}) \text{ (i.e., a Beta distribution with a mean of .0001)}$$

$$\mu(t) \sim \text{Beta}(2, \frac{2-.0002}{.0001}) \text{ (i.e., a Beta distribution with a mean of .0001)}$$

$$\gamma(t) \sim \text{Beta}(2, \frac{2-.2}{.1}) \text{ (i.e., a Beta distribution with a mean of .1)}$$

$$\beta(t) \sim \text{Poisson}(5) \text{ (i.e., a Poisson distribution with a mean of 5)}$$

Equipped with these equations, we use the `ODEInt` Python package to simulate 100 epidemics, each with 100 time steps. All simulations assume a population size *N* of one million people, and are initialized at *N-1* susceptible people, one infected person, and zero recovered people. At each time step, we record the number of susceptible, infected, and recovered people in the population, and for each three-time step interval, we record the polarity of the level of infections in the population. This yields 9,800 pairs in which the first element is a nine-component vector representing the numbers of susceptible, infected, and recovered people in the population at each of the three time steps in an interval and the second element is a single number representing the polarity of the level of infections over that same interval. Note that while we use simulated data here, if one had access to real-world SIR data from an epidemic, one could use that data to prepare a similar training set.

Moving to the next phase of the deep learning pipeline, we trained a multi-layer perceptron (a common type of neural network) with one hidden layer on all 9,800 of our synthetically-generated training data points.[1] The training accuracy was 69%, suggesting that the neural network model learns to predict the overall polarity of the level of infections from more fine-grained SIR data at a rate significantly above chance, at least when confined to the context in which training data is generated (in the training data, the polarity of the rate of infections is positive 46.67% of the time and negative 53.33% of the time). Note that because the training accuracy is also far below 100%, we can also conclude that the neural network has *not* learned the mathematical formula for calculating the overall polarity of the level of infections from SIR data, but has instead learned a (relatively) reliable statistical pattern linking combinations of values of SIR data to the polarity of the level of infections, which it then exploits to classify the overall polarity of the system.

Finally, we move to the evaluation phase of the deep learning pipeline. Here, we seek to evaluate the performance of the model trained on our training data when it is given inputs generated by a structurally similar but distributionally distinct data-generating process. That is, we generate simulated data using the same differential equations as above, but with the birth rate, death rate, recovery rate, and average number of interactions sampled from the following new distributions:

$$\Lambda \sim \text{Beta}(2, \frac{2-.02}{.01})$$

$$\mu \sim \text{Beta}(2, \frac{2-.02}{.01})$$

$$\gamma \sim \text{Beta}(2, \tfrac{2-.02}{.01})$$

$$\bar{\beta} \sim \text{Poisson}(15)$$

[1] Code and results for all computation experiments can be viewed in an open repository at: https://osf.io/zb27c/?view_only=3b57e60d514a43fe9117697f334a8be8.

We simulate twenty of these out-of-distribution epidemics, and test the performance of our neural network that was trained on the in-distribution data when it comes to using out-of-distribution data to classify the polarity of the epidemic over a given interval. The average accuracy across the twenty epidemics is 51.86%, which is not significantly above chance. We take this as evidence that the deep learning pipeline described here, which uses a naive approach to the selection and representation of training data, is underspecified. Despite finding a weight vector that allows it to achieve some degree of accuracy on in-distribution data, the accuracy of a model parameterized by those weights evaporates when we move to an out-of-distribution setting. This suggests that whatever patterns the model was exploiting in order to classify the overall polarity of the level of infections based on more fine-grained SIR data, those patterns were specific to the distributions used to generate the parameters of the model, rather than to the underlying structure of the data-generating process captured by the differential equations. In what follows, we explore how an approach inspired by system dynamics allows us to achieve significantly better performance on the same fundamental problem.

Using Deep Learning to Understand an Epidemic: The Component Loop Approach

We now re-run the deep learning pipeline in a manner designed to specifically integrate a system-dynamics-inspired approach, in which a key part of understanding a system is understanding how the polarities of the system's various component loops constrain overall loop behavior. The first two stages of the pipeline are unchanged from the naive approach: the fundamental problem to be solved is still to use more fine-grained features of an epidemic that are observed over a given time interval to determine the overall polarity of the level of infections over the same time interval, and the formulation of that problem for machine learning is still to train a neural network to predict, based on more fine-grained observations of an epidemic system across a time-step interval, the polarity of the level of infections across the same three-time-step interval.

However, once we reach the stage where we select and prepare training data, we begin to incorporate novel insights from system dynamics. At a high level, our goal is to use both our knowledge of the structure of the differential equations from the SIR model and a loop-polarity based approach to improve the robustness of our pipeline. To do this, we run the same simulations as before, only now we save *both* the SIR data (i.e., the numbers of susceptible, infected and recovered people at each time step), *and* the parameters of the differential equations (i.e., the birth rate, death rate, recovery rate, and average number of interactions at each time step). Using these data, we calculate the *accumulations* in each of the six unique summands on the LHS of each of the differential equations at a given time-step, using the following equations:

$$\mathcal{A}_1(t) := \int_0^t \Lambda(t')N\mathrm{d}t' \text{ (the birth rate)}$$

$$\mathcal{A}_2(t) := \int_0^t \mu(t')S(t')\mathrm{d}t' \text{ (the number of susceptible people who die)}$$

$$\mathcal{A}_3(t) := \int_0^t \frac{\beta(t')I(t')S(t')}{N}\mathrm{d}t' \text{ (infected-susceptible interactions over } N\text{)}$$

$$\mathcal{A}_4(t) := \int_0^t \gamma(t')I(t')\mathrm{d}t' \text{ (the number of infected people who recover)}$$

$$\mathcal{A}_5(t) := \int_0^t \mu(t')I(t')\mathrm{d}t'\text{(the number of infected people who die)}$$

$$\mathcal{A}_6(t) := \int_0^t \mu(t')R(t')\mathrm{d}t' \text{ (the number of recovered people who die)}$$

Looking at a causal loop diagram of the epidemic system defined by our SIR model (Fig. 3), one can find a correspondence between these accumulations and the six main loops that compose the epidemic system being simulated. Beginning with Loop 1/Accumulation 1, we can see that as the number of susceptible people changes, so too does the accumulation of the multiple of the birthrate and the population, which in turn affects the number of susceptible people. In the case of Loop 2/Accumulation 2, we note that as the number of susceptible people increases, so too does the accumulation of the total number of people who

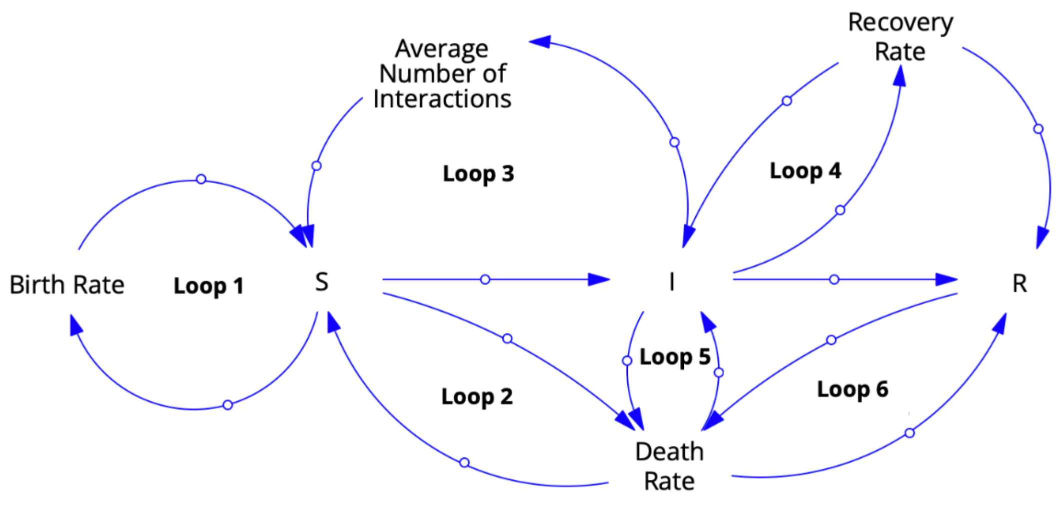

Fig. 3: Causal loop diagram representation of the SIR model used for our computational experiments.

have died during the epidemic, which in turn affects the number of susceptible people. One could give analogous reasoning for all six loops/accumulations, to demonstrate the correspondence between these loops and the accumulations in our model.

Having calculated these accumulations, we next calculate the polarity of each accumulation over the same three-time-step interval using the formula:

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

This, in turn, enables us to compile our training data. Each training datum is a pair whose first element is a six-component vector representing the polarity of each of the accumulations listed above over a three-time-step interval (which we calculate for each interval) and whose second element is a single number representing the polarity of the overall level of infections across the same three-time-step interval. Note that our calculations of the loop polarities for each component loop of the system are what, in the machine learning literature, are called "manually engineered" features, since they are calculated by human researchers *from* the data, rather than merely being read *off* of the raw data (Verdonck et al., 2024). Our goal is for a neural network to learn the systematic relationship between the polarities of these component

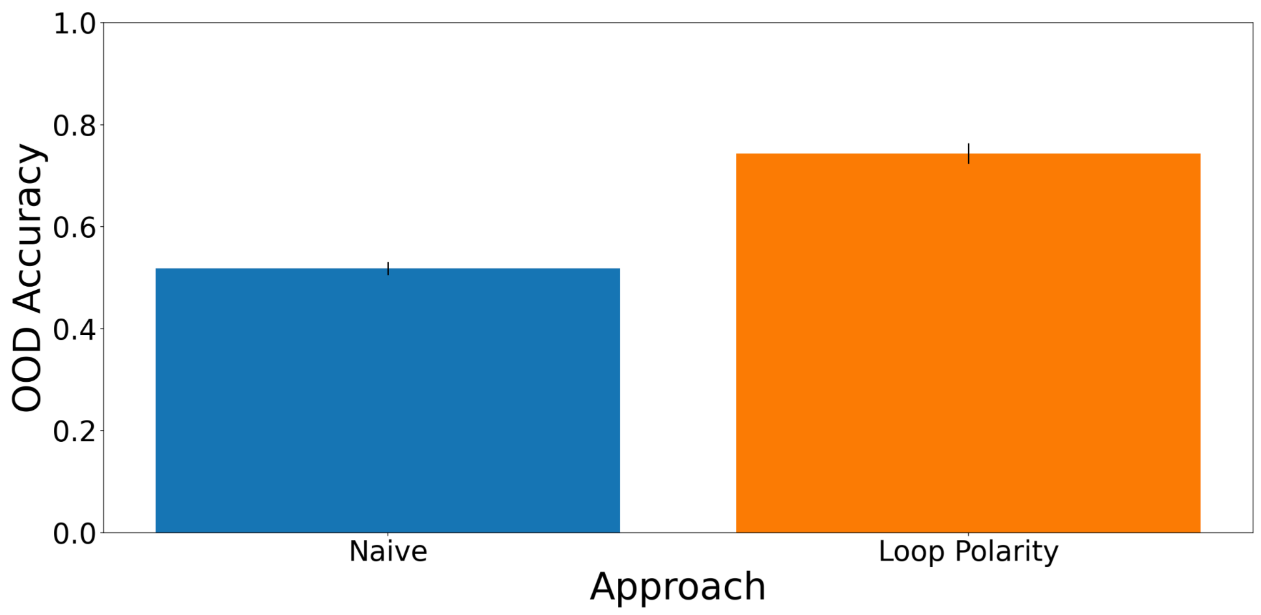

Fig. 4: Comparison of the mean accuracy of both the naive, raw-data-input approach and our loop-polarity-based approach across twenty simulated pandemics in an out-of-distribution regime. Error bars show a 95% confidence interval for the mean.

loops of the epidemic system and the overall polarity of the level of infections. If such learning is possible, then it will show that the model *is* capable of inferring macro-level behavior from the behavior of a system's component parts, provided that those components are appropriately represented in the training data.

Having prepared our training data in this way, we again train a multi-layer perceptron with one hidden layer on this new training data set. The training accuracy on this data is much improved from the naive approach, reaching 98.86%. Most importantly, when we move to the evaluation phase of the deep learning pipeline and generate twenty out-of-distribution epidemics, we find that the average accuracy of our newly-trained model when it comes to predicting the polarity of the level of infections for a given three-time-step-interval is 74.41%. This constitutes a highly significant difference in OOD accuracy between the model trained on loop polarity data and the model trained via the more naive methodology. See Fig. 4 for a comparison of the accuracy of the two approaches when applied to OOD data. Thus, we produce evidence showing that, by manually engineering our training data using a perspective rooted in causal loop dynamics, and in particular the relationship between the component loops of a system and overall system behavior, we can significantly reduce the brittleness of our models, suggesting a similarly significant reduction in the degree to which our model-production pipeline is underspecified.

Discussion

Our findings from this computational experiment have broader implications for the optimal functioning of the machine learning 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-represented in existing deep learning practice. The results here indicate that in at least some contexts, this perspective *should* be represented, 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.

Conclusion

There are several avenues for future work that 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. Second, we plan to extend our results beyond the synthetic setting, demonstrating similar performance for the results presented here on real-world data sets. Finally, 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 polarity framework to improve out-of-distribution classification of system behavior.

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.

Centers for Disease Control and Prevention. (2021, Mar) *Why Look for Feedback?* URL: https://www.cdc.gov/policy/polaris/tis/feedback/index.html.

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).

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.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In International Conference on Learning Representations, 2019.

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.

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

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

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