DEEP LEARNING FOR CAUSAL STRUCTURE LEARNING APPLIED TO CANCER PATHWAY DISCOVERY

by

Jonathan D. Young

B.A., University of Virginia, 2002

M.S., Georgetown University, 2005

M.D., Eastern Virginia Medical School, 2010

M.S., University of Pittsburgh, 2015

Submitted to the Graduate Faculty of

the Intelligent Systems Program in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2020

UNIVERSITY OF PITTSBURGH

INTELLIGENT SYSTEMS PROGRAM

This dissertation was presented

by

Jonathan D. Young

It was defended on

February 20, 2020

and approved by

Dr. Xinghua Lu, Intelligent Systems Program, University of Pittsburgh

Dr. Gregory Cooper, Intelligent Systems Program, University of Pittsburgh

Dr. Vanathi Gopalakrishnan, Intelligent Systems Program, University of Pittsburgh

Dr. Harry Hochheiser, Intelligent Systems Program, University of Pittsburgh

Dr. Songjian Lu, Department of Biomedical Informatics, University of Pittsburgh

Dissertation Director: Dr. Xinghua Lu, Intelligent Systems Program, University of Pittsburgh

DEEP LEARNING FOR CAUSAL STRUCTURE LEARNING APPLIED TO CANCER PATHWAY DISCOVERY

Jonathan D. Young, PhD

University of Pittsburgh, 2020

It is well appreciated that cancer is a disease of aberrant signaling and the state of a cancer cell can be described in terms of abnormally functioning cellular signaling pathways. Identifying all of the abnormal cellular signaling pathways causing a patient's cancer would enable more patient-specific and effective treatments. Here we interpret the cellular signaling system as a causal graphical model and apply a modified deep neural network to learn latent causal structure that represents cancer pathways. We address a problem for which it is known that a set of variables X *causes* another set of variables Y (e.g., mutations in DNA *cause* changes in gene expression), and these causal relationships are encoded by a causal network among a set of an unknown number of latent variables. We develop a deep learning model, redundant input neural network (RINN), with a modified architecture and an L_1 regularized objective function to find causal relationships between input (X), latent, and output (Y) variables. RINN allows input variables to directly interact with all latent variables in a neural network to influence the information latent variables encode. In a series of simulation experiments, we show that RINN recovers latent causal structure from various simulated datasets better than other models. We hypothesize that training RINN on omics data will enable us to map the functional impacts of genomic alterations to latent variables in a deep learning model, allowing us to discover the hierarchical causal relationships between variables perturbed by different genomic alterations. Importantly, the direct connections between input and latent variables in RINN make the latent variables partially interpretable, as they can be easily mapped to input space. We apply RINN to cancer genomic data, where it is known that genomic alterations cause changes in gene expression. We show that gene expression can be predicted from genomic alterations with reasonable AUROCs. We also show that RINN is able to discover real cancer signaling pathway relationships, especially relationships in the PI3K, Nrf2, and TGFβ pathways, including some causal relationships. However, despite high

regularization, the learned causal graphs are still somewhat too dense. We discuss promising future directions for RINN, including differential regularization, autoencoder pretrained representations, and constrained evolutionary strategies.

Table of Contents

List of Tables

List of Figures

List of Algorithms

1.0 Introduction

Understanding the cellular signal transduction pathways that drive cells to become cancerous is fundamental to developing personalized cancer therapies that decrease the morbidity and mortality of cancer. Ideally, a physician could obtain a sample of a patient's cancer tissue, extract gene expression and DNA mutation data from the sample, run these data through an algorithm, and have the algorithm model the cellular signaling pathways that are causing cancer. Having a personalized model of the cancer cellular signaling system for a patient would be a first step toward finding an effective treatment for the patient. A physician (or algorithm) could analyze the model and propose treatments to block or correct abnormal signaling pathways. This dissertation is a step towards this future. Here we interpret the cellular signaling system from within a causal framework and develop neural network-based algorithms to postulate causal structure that represents the cellular signaling system.

1.1 The Problem

In general, the cellular mechanisms leading to cancer in an individual are nuanced and not well understood. We know that cells function through cellular signaling pathways and that cancer can be described in terms of abnormally functioning pathways. If we were able to determine all of the abnormal cellular signaling pathways causing a patient's cancer, we could develop better, more patient-specific treatments—including targeting multiple abnormal pathways during a single treatment regime. However, reliably inferring all abnormal cancer cellular signaling pathways from genomics data is a very difficult and unsolved task. Most current methods attempt to infer the status of a single or a few abnormalities that are causing cancer, and do not attempt to find all abnormal pathways. In addition, we have an incomplete understanding of the pathways that cause cancer. Moreover, treatments of known pathway abnormalities remain limited.

Another reason that it is difficult to infer all abnormal cellular signaling pathways solely

based on genomic data alone is because the majority of the genomic alterations observed in a tumor are non-consequential with respect to cancer development and only a few are *drivers*. Furthermore, even if the driver genomic alterations of a tumor are known, we need to infer the activation state of other signaling proteins, which are not measured (latent variables), to understand the disease mechanisms of an individual tumor and identify drug targets. This requires us to study the causal relationships among latent (i.e., hidden or unobserved) variables, such as large protein complexes or abstractions of different biological processes within a cell, in addition to observed variables. Most causal discovery algorithms have been developed to find causal structure and parameterizations of causal structure on the observed variables of a dataset. A smaller number of causal discovery algorithms also find latent causal structure, but these methods are often highly constrained, have difficulty with high-dimensional data, or are less applicable to the problem explored here [Murray-Watters and Glymour, 2015, Frot et al., 2019, Colombo et al., 2012, Kummerfeld and Ramsey, 2016] (see Section 2.3). This suggests that new approaches to discovering latent causal variables are needed. Applying such methods to discover cellular signaling pathways could have a large impact on our understanding of biology and disease. The algorithms we develop here are intended to address some of these problems and difficulties.

1.2 The Approach

In this dissertation, we will explore the utility of deep learning models in discovering causal pathways. Deep learning models can theoretically represent the hierarchical organization of signaling molecules in a cell, with latent variables as natural representations of unobserved activation states of signaling molecules. We develop deep learning and evolution-based algorithms to learn causal relationships from genomic data in such a way that the the causal relationships can be interpreted as cellular signaling pathways. A major component of this research involves developing simulated data from a known causal ground truth directed acyclic graph (DAG), where the ground truth DAG represents a simplified cellular signaling pathway. The simulated data addresses the problem of not having gold standard ground truth

knowledge about abnormal cellular signaling pathways (i.e., gold standard causal relationships between biological entities). In addition, we develop a deep learning visualization method and a method for calculating precision and recall of causal structure. We use these methods to explore the extent to which different algorithms can learn causal structure from simulated data. We then use the best performing algorithm to find causal structure within genomic data from human cancer (TCGA) and compare with known human cancer signaling pathways.

1.2.1 Hypothesis

We hypothesize that a redundant input neural network (RINN), a type of deep learning model, can be used to discover latent causal relationships. Furthermore, we hypothesize that training RINN on omics data will enable us to map the functional impacts of genomic alterations to latent variables in a deep learning model, allowing us to discover the hierarchical causal relationships between variables perturbed by different genomic alterations.

1.2.2 Specific Aims

This research will address the following specific aims:

1. Develop a deep learning model capable of learning causal relationships among input, latent, and output variables.
2. Develop methods to evaluate the causal relationships learned by deep learning models including, a ground truth causal structure, simulated datasets inspired by biological signaling pathways, a deep learning visualization method, and a method for calculating precision and recall of causal structure.
3. Apply the new deep learning model to discover cancer pathways.

1.3 Dissertation Overview

In Background, we present a literature review necessary for understanding the topics in this dissertation and to discuss how the work presented here is different from previous

research endeavors. In Unsupervised Deep Learning Applied to Cancer Data, we present results and conclusions showing that deep learning models can learn biological relevant hidden variables. These results motivated the remaining work in the dissertation. We next describe our development of the RINN method for learning latent causal structure with a deep learning model and evaluating the correctness of the learned causal structure in Deep Learning and Causal Discovery Methods Development. In Experiment and Results Using Simulated Data, we describe the results of experiments learning latent causal structure with simulated data. We then apply the methods developed and lessons learned from the previous two chapters to human cancer genomic data from The Cancer Genome Atlas (TCGA) in Using the RINN to Study Cancer Cell Signaling. The last section of the dissertation discusses the major conclusions of the dissertation and promising future work.

2.0 Background

2.1 Deep Learning

Deep learning has experienced an explosion in popularity recently and has established state of the art results in many domains, largely due to the ability of deep learning algorithms to scale exceptionally well with large amounts of data and to parallelize across multiple GPUs [Schmidhuber, 2015, LeCun et al., 2015, Raina et al., 2009]. Deep learning represents a group of machine learning algorithms or strategies that originated from artificial neural networks (ANN). ANNs represent a framework, loosely inspired by biological neurons, for learning a function (represented as a set of parameters or weights) that maps inputs to outputs. This function consists of iteratively calculating vector-matrix multiplication (i.e., linear combination) plus adding a vector of biases (i.e., affine transformation), followed by an element-wise nonlinear function (i.e., activation functions such as the sigmoid function, softplus function, or ReLU (rectified linear unit) function). The vector in "vector-matrix multiplication" above is initially the input data, then the output of the first "vector-matrix multiplication plus nonlinearity" represents the values for the first hidden layer. These values are used as input to the next vector-matrix multiplication. The matrix in "vector-matrix multiplication" is a matrix of weights (or coefficients, parameters) that represent the parameters of the neural network function mapping input to output. An ANN, as just described with many hidden layers, is considered a Deep Neural Network (DNN) [Goodfellow et al., 2016].

In more detail, a DNN learns a function mapping inputs ($\boldsymbol{x}$) to outputs ($\boldsymbol{y}$) according to

$$f(\boldsymbol{x}) = \phi(...\phi(\phi(\boldsymbol{x}\boldsymbol{W}_1)\boldsymbol{W}_2)...\boldsymbol{W}_n) = \hat{\boldsymbol{y}} \tag{2.1}$$

where $\boldsymbol{W}_i$ represent the weight matrices between layers of a neural network, ϕ is some nonlinear function (i.e., activation function such as ReLU or sigmoid), and $\hat{\boldsymbol{y}}$ is the predicted output. This function represents our predicted value for $\boldsymbol{y}$. DNNs also have bias vectors added at each layer (i.e., at each $\boldsymbol{h_{i-1}}\boldsymbol{W_i}$, where $\boldsymbol{h_{i-1}}$ is the previous layer's output),

which have been omitted from the above equation for clarity. Next, we compare $\hat{\boldsymbol{y}}$ to $\boldsymbol{y}$ (ground truth) and measure the error, or how bad our function approximation is so far. The equation above represents repeatedly performing vector-matrix multiplication and applying a nonlinear function to the resulting vector. One performs these vector-matrix multiplications, nonlinearities for each layer until the final output values have been calculated. This process is called forward propagation through a neural network. After forward propagation, the weight matrices are updated (moved closer to values that decrease the error function) using backpropagation of errors to calculate the gradient of the error function with respect to the weights, and stochastic gradient descent optimization to move the weights closer to values that minimize the error function. This process should result in $\hat{\boldsymbol{y}}$ getting closer and closer to $\boldsymbol{y}$ as the training proceeds, thus leading to a function that accurately maps inputs to outputs. The left side of Figure 6, without the redundant input nodes and corresponding redundant input weights, represents a simple neural network. For more details about DNNs, see [Goodfellow et al., 2016] and Section 4.5.

An essential characteristic of deep learning models is their ability to learn alternate representations of the data [Lee et al., 2008, LeCun et al., 2015]. This characteristic has inspired researchers to explore imaginative uses for deep learning strategies. A deep learning strategy is composed of multiple layers of latent variables (hidden nodes or units, in the ANN jargon) [Deng et al., 2014, Goodfellow et al., 2016, LeCun et al., 2015], which learn to represent the complex statistical structures embedded in the data, such that different hidden layers capture statistical structure of different degrees of complexity. In other words, deep learning strategies learn novel representations of the statistical structure of the input data through hierarchical decomposition. For example, if one trains a convolutional neural network (a deep learning strategy often used for image recognition) with three hidden layers on a dataset of face images (in order to learn how to recognize a human face in an image), the units in these three layers capture abstract representations at different levels. The model may use the first hidden layer units (the layer that is closest to the input data) of the trained network to capture edges of different orientations present in the original image [Lee et al., 2008, Lee et al., 2011]. The second hidden layer units may learn representations of different parts of a face [Lee et al., 2011] (e.g., nose, mouth, eye, ear), by combining edges of different

orientations (represented by the units in the first hidden layer). Finally, units in the third hidden layer may represent generic faces [Le et al., 2011, Lee et al., 2011], which can be thought of, or represented as, combinations of parts of a face. In this way, deep learning finds hierarchical structure in data by finding alternate representations (often of lower dimension) that best encode the information in the original image. Once a deep learning model is trained, it can be used to detect whether a new input image contains a human face, or it can be used to generate new face images that mimic the distribution of the input images.

2.1.1 Architectures

Experimenting with different neural network architectures is a common theme in neural network and deep learning research. Various versions of fully-connected neural networks (i.e., each node is connected to every other node in the network) have been studied since 1990 [Fahlman and Lebiere, 1990, Wilamowski and Yu, 2010]. The Dense Convolutional Network (DenseNet) [Huang et al., 2017] is a more recent version of a fully-connected neural network that improves prediction and helps to alleviate the vanishing-gradient problem [Bengio et al., 1994, Glorot and Bengio, 2010] in very deep networks. DenseNets concatenate all previous feature-maps (i.e., hidden layer values) into a single tensor and use this tensor as input to the next hidden layer in the network. Related to the DenseNet, deep learning models with different types of "skip" connections have also been developed to improve prediction and help with the vanishing-gradient problem. These include Highway Networks [Srivastava et al., 2015] and Deep Residual Networks (ResNets) [He et al., 2016]. Highway Networks and ResNets allow the output of a previous hidden layer to directly influence the output of a future hidden layer. This allows important information from as far back in the network as the input to be propagated to a later hidden layer and/or the output layer. In general, the new architectures discussed here were developed to improve the prediction ability of very deep neural networks. In contrast, the RINN was developed to learn latent causal structure and specifically direct relationships between the input and latent variables. Accordingly, a major focus of the RINN architecture was on learning the parameters between the redundant inputs and various hidden layers. Conversely, some of the algorithms discussed above do not

learn parameters for the inputs to hidden layers or "skip" connections. Also, the RINN is not a fully-connected network like DenseNet; the inputs do not directly connect to the outputs and there are no hidden to hidden "skip" connections. The RINN is perhaps most closely related to a Recurrent Neural Network (RNN). It may be simplest to think of a RINN as an RNN with non-shared weights, time-invariant (i.e., static) input, and an output generated only at the last time-step [Liao and Poggio, 2016].

2.2 Deep Learning and Signaling Pathways

A major aim of this dissertation is to use deep learning to find the hidden layer representations of genomic data that possibly represent the state of the signaling systems in cells. More specifically, we hypothesize that the activation states of signaling pathways regulating transcriptomic activities in cells can be learned using deep learning algorithms (Figure 1). Consider cancer as an example to illustrate this hypothesis. Cancer is a multi-process disease, in that a cancer cell usually has multiple aberrant pathways—each pathway consists of a set of hierarchically organized signaling molecules (Figure 1A)—driving differentially expressed genes (DEGs) that are involved in multiple oncogenic processes. A transcriptomic profile (i.e., gene expression data) of a tumor is a convoluted mixture of expressed genes regulated by active pathways in tumor cells, but the information related to the hierarchical organization of pathways is "collapsed" and distinct signals from different pathways become mixed (Figure 1B). Discovering aberrant signals in cancer cells requires deconvolution (decomposition) of such signals, and deep learning is particularly well suited for such a task due to the ability to perform hierarchical decomposition.

For example, the hierarchical structure of a signaling pathway (Figure 1A) could be simulated by the hierarchical structure in a deep learning model trained on gene expression data (Figure 1C). Since transcription factor (TF) activation dictates the finest covariance structure of gene expression, which is at the bottom of the signaling pathway in Figure 1A, the first hidden layer in our deep learning model (Figure 1C) may capture the signals encoded by TFs. And just as there are different pathways being activated that regulate transcription

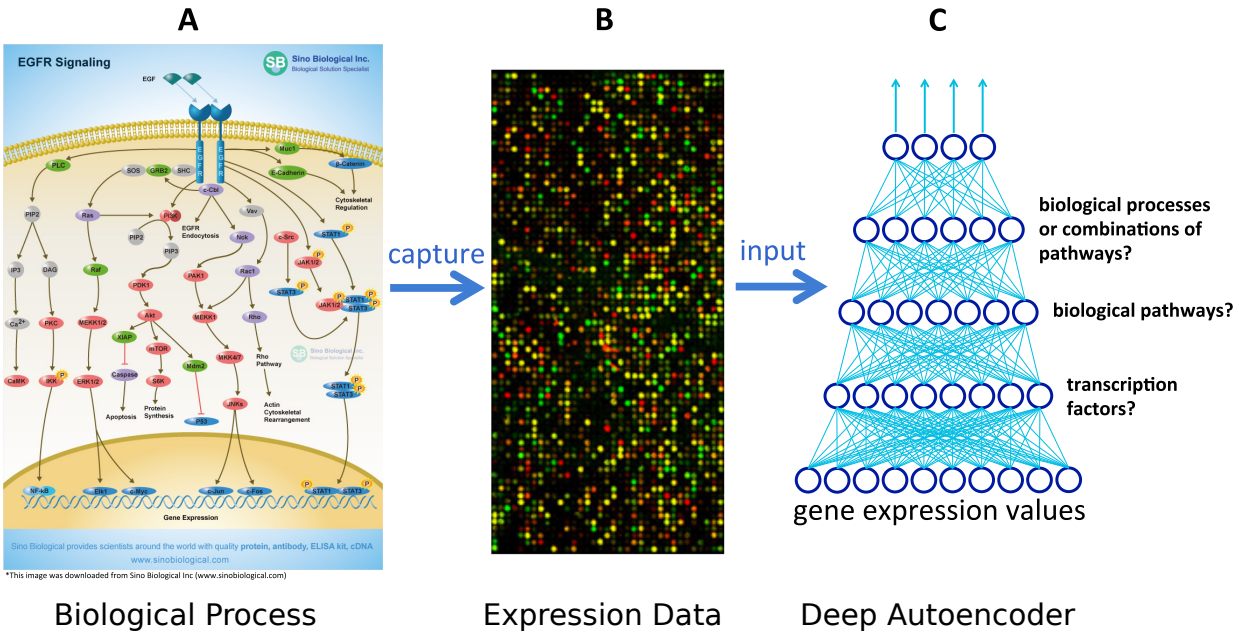


Figure 1: Inferring the Activation States of Signaling Pathways with Deep Learning

(A) Example EGFR signaling cascade. (B) A microarray representing gene expression values. (C) Representation of a Deep Belief Network (DBN), fine-tuning encoder network only, trained on gene expression values. Hypothesized biological entities whose activation states may be represented by units in the 1st, 2nd, and 3rd hidden layers are displayed on the right.

factor activation in the middle of Figure 1A, the second hidden layer of our model may represent the activation states of different biological pathways. Continuing with this analogy, units in the third hidden layer could represent biological processes (e.g., inflammation or metastasis) or combinations of pathways. In this way, we hope to learn the hierarchical representation of the signals underlying cancer gene expression data with deep learning.

Recently, our research group demonstrated that deep learning could be used to learn and represent the cellular signaling system [Chen et al., 2015, Chen et al., 2016]. In one study, our group showed that deep learning, more specifically a multi-modal deep belief network (DBN), can learn representations of the cellular signaling system shared by rat and human cells [Chen et al., 2015]. In the same study, it was also demonstrated that a trans-species deep learning model could accurately predict human cell responses to a set of unknown stimuli based on data from rat cells treated with the same stimuli. In a more recent study, it was shown that deep learning models can learn representations of the yeast transcriptomic system, and that the hidden units of these deep learning models can be mapped to real biological entities, such as transcription factors and well-known yeast signaling pathways [Chen et al., 2016]. Another group, [Liang et al., 2015] integrated multiple types of genomic cancer data with a multi-modal DBN and were able to use their model to identify cancer subtypes and genes important for each of these subtypes. However, they did not attempt to learn cellular signaling pathways as we do here.

A DBN [Hinton and Salakhutdinov, 2006] is another strategy under the deep learning umbrella and is a type of autoencoder. An autoencoder is an unsupervised algorithm and a type of ANN that learns a compressed representation of the input data by passing the input data through a bottleneck (hidden layer with low number of hidden nodes) and then attempting to regenerate the input data. A DBN consists of two parts, pre-training and fine-tuning (see Figure 2). Pre-training involves training a stack of restricted Boltzmann machines (RBM). An RBM is a two-layer undirected probabilistic network trained to learn a lower dimensional representation of the input to the RBM that is best able to reconstruct that input. RBMs are trained sequentially (i.e., layer-by-layer), using a training method known as contrastive divergence [Hinton and Salakhutdinov, 2006], then the next RBM is stacked on the previous one. Forcing the information through a lower dimensional representation creates

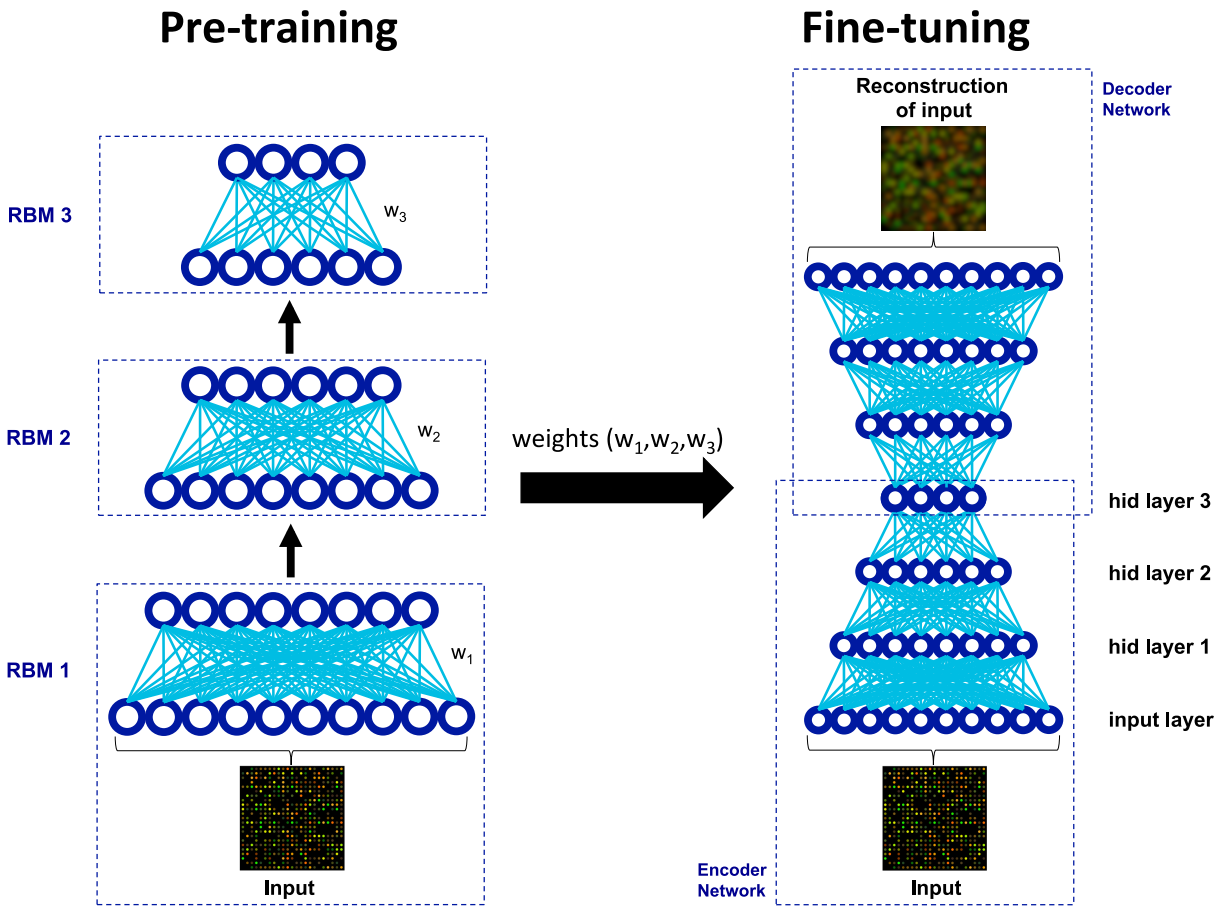


Figure 2: Deep Belief Network (DBN) with Two Phases: Pre-training and Fine-tuning

Each restricted Boltzmann machine (RBM) in the pre-training phase iteratively learns lower dimensional representations of the input (gene expression microarray) one RBM at a time. These lower dimensional representations are then used as input to the next RBM. The pre-training phase learns a set of weights $W = \{w1, w2, w3\}$ that is used to initialize the fine-tuning phase. The fine-tuning phase globally optimizes the weights by minimizing the cross-entropy error.

an information bottleneck at each layer. These lower dimensional representations must learn the most salient features of the input data in order to best be able to reconstruct the input data. After the stack of RBMs are fully trained, the weight matrices learned by the stack of RBMs are used to initialize the weights of deep autoencoder (DA), which is an unsupervised DNN (see Section 2.1) where the DA learns to reconstruct the input from lower dimensional hidden layers—so the outputs, $\boldsymbol{y}$ values, for a DA are a copy of the corresponding $\boldsymbol{x}$ values. Training the DA is referred to as fine-tuning (i.e., fine-tuning the pre-training weights). In contrast to a stack of RBMs, where each RBM is trained to completion before moving to the next RBM, the weights of a DA are updated globally, meaning all the weights are updated at the same time. For more details on the DBN used in this dissertation, see Section 4.5.2. For a more detailed explanation of a DBN and specifically its use in a biological context, see [Young et al., 2017] (Appendix).

2.3 Deep Learning and Causal Discovery

Causal Discovery is a field of research broadly interested in identifying the causal structure and parameters among variables from data and possibly background knowledge, i.e., identifying which variables in a dataset are the causes of other variables in the dataset and the strength of these causal relationships. In this context, a causal structure is represented as a graph $G = (V, E)$ containing a set of vertices V and a set of edges E, and the goal is to infer the data-generating causal structure from data. There have been numerous algorithms developed for this purpose (see [Spirtes et al., 2000, Cooper, 1999, Heinze-Deml et al., 2018, Maathuis and Nandy, 2016, Peters et al., 2017] and specifically, [Lagani et al., 2016] for causal discovery in systems biology), which will not be discussed here in detail. Please see Section 4.7 for more information and a discussion of some of the common causal assumptions necessary for causal discovery.

The work in this dissertation concentrates mostly on finding latent causal relationships when the causal direction between inputs and outputs is known. There has been some recent research in the causal discovery community that deals with this problem. Alexander

Murray-Watters and Clark Glymour developed the DM (Detect MIMIC (Multiple Indicators Multiple Input Causes)) algorithm [Murray-Watters and Glymour, 2015, Murray-Watters, 2014] for finding latent causal relationships between inputs and outputs, *when the inputs are known to cause outputs*. First, if the input and output sets are unknown, they split the data into input and output sets using the PC algorithm [Spirtes et al., 2000], which is a constraint-based causal discovery algorithm that starts with a fully connected graph and then removes edges and changes undirected edges to directed edges based on conditional independence tests. Next, they used a group of heuristics based on conditional independence and Sober's criterion [Sober, 1998, Murray-Watters, 2014] (see Section 4.6) to identify latent causal structure between the inputs and outputs. Sober's criterion is used to determine the presence of latent variables between inputs and outputs (i.e., causes and effects) when inputs are known to cause outputs. If two outputs are independent of one another when conditioned on their corresponding causal inputs, then there is no latent variable between these inputs and outputs. If not conditionally independent, then there is a latent variable present [Murray-Watters, 2014]. The DM algorithm is discussed in more detail, including the causal assumptions it makes and how it relates to the algorithms developed in this dissertation, in Section 4.6. A more constrained algorithm, relative to the DM algorithm, for finding latent structure relevant to an input/output system is the FOFC (Find One Factor Clusters) algorithm [Kummerfeld and Ramsey, 2016]. This algorithm in the input/output setting would find clusters of output variables that are causally isolated from all other output variables in the dataset. It then associates each of these clusters with one or a maximum of two latent variables. Each latent variable can only have one input variable as its cause. FOFC is unable to identify the causal direction between latent variables if an adjacency is detected between latent variables. The DM and FOFC algorithms are some of the more recent algorithms for finding latent structure that are relevant to this dissertation. In addition to the DM and FOFC algorithms there are other algorithms developed to find latent structure, including factor analysis algorithms, algorithms that use variations of the Expectation-Maximization (EM) algorithm [Friedman et al., 1997, Elidan and Friedman, 2005], and algorithms based on fast causal inference (FCI) [Spirtes et al., 1995, Colombo et al., 2012]. These algorithms are, in general, highly constrained, intractable in high-dimensional spaces, return only a subset of

the latent causal relationships, *or* don't leverage prior knowledge that input variables cause output variables. FCI, for example, only learns an abstraction of the latent variables and little about the causal structure among the latents.

Recently, however, there has been an increased interest within the deep learning community to combine deep learning with causal discovery to learn causal relationships. Some of this interest has come, in part, from reasoned skepticism regarding deep learning's ability to reach artificial general intelligence (AGI), or human-like intelligence, without some means of causal reasoning [Pearl, 2018].

A promising approach to learning causal relationships with deep learning (and related to the approach presented in this dissertation) is explored in [Kalainathan et al., 2018], where they developed an algorithm named the structural agnostic model or SAM. At a high-level, the algorithm uses a modified version of a generative adversarial network (GAN) [Goodfellow et al., 2014]—an unsupervised deep learning strategy. SAM trains multiple GANs, one GAN to generate each variable in $\boldsymbol{X}$ ($\boldsymbol{X}$ denotes the measured variables or features in a dataset). In brief, a GAN consists of a neural network representing a generator, G, that generates a sample of the data from random noisy input, and a neural network representing a discriminator, D, that is trained to learn the difference between real data (i.e., sampled from the dataset) and fake data (i.e., generated by the learned generators). G is trained to fool D, to encourage D to classify generated samples as real data instead of fake. SAM uses an ensemble of GANs to generate samples of $\boldsymbol{X}$; however, as mentioned previously, it uses a unique GAN for each variable in $\boldsymbol{X}$. This means that $|\boldsymbol{X}|$ GANs will need to be trained to learn an entire S_{out}, which describes the causal relationships for each variable in $\boldsymbol{X}$. Each GAN uses $\boldsymbol{X}_{-j}$ variables (i.e., all variables in $\boldsymbol{X}$ other than X_j) to generate X_j. Each of these GANs could be referred to as a "leave-one-out" GAN, as it is similar to leave-one-out cross-validation (leaving out an instance versus leaving out a variable for SAM).

Each of the GANs described above have two additional modifications to allow the algorithm to be used for causal discovery. First, G's objective function includes a sparsity penalty to encourage only a small subset of $\boldsymbol{X}_{-j}$ to be used to generate X_j. Second, an additional set of parameters, $\boldsymbol{a} \in \mathbb{R}^{|\boldsymbol{X}|}$, is learned by the algorithm. $\boldsymbol{a}$ is a binary vector that is multiplied by the input to each GAN at the very beginning of the generative process (Note: a_j is set to 0,

as we are not using X_j to predict itself), and can be thought of as a boolean mask applied to the input to decide which inputs to use to generate the output. A unique $\boldsymbol{a}$ is learned for each GAN, i.e., for each X_j. $\boldsymbol{a}$ represents the direct causal relationships between any variable in $\boldsymbol{X}_{-j}$ and X_j. All of the $\boldsymbol{a}$ vectors, once learned, describe the causal structure, or S_{out}, learned by the algorithm. Note that this setup allows the algorithm to learn and represent cycles in S_{out}. The sparsity penalty mentioned above is an L_1 penalty on $\boldsymbol{a}$ and the $\boldsymbol{a}$ vectors are called causal filters. SAM is trained through backpropagation and stochastic gradient descent (SGD) just like a regular GAN. In a very recent paper and somewhat similar to SAM, [Ke et al., 2019] also used boolean masks applied to inputs in an ensemble of neural networks to model the causal relationship between observed variables and the observed variable's parents. They also used an interesting meta-learning objective by parameterizing the neural networks into "fast" parameters and "slow" meta-parameters (related to the causal structure). SAM and [Ke et al., 2019] are examples of some of the creative work being done using deep learning strategies for causal discovery.

In addition to SAM and [Ke et al., 2019], other research groups have used deep learning strategies to find causal relationships. [Lopez-Paz et al., 2017] used supervised CNNs to predict the direction of the causal arrow between two variables in static images. Somewhat related, [Harradon et al., 2018] also used CNNs in part of a methodology aimed at finding causal relationships that can explain an image. However, they used autoencoder CNNs (unsupervised) to learn salient concepts (features) in images (i.e., lower dimensional representations of the image), then built a Bayesian causal model using these lower dimensional representations. They were than able to visualize the features that had the most causal influence on image classification [Harradon et al., 2018].

Some research in this space has even concentrated on counterfactual inference [Johansson et al., 2016, Hartford et al., 2017, Shalit et al., 2017, Louizos et al., 2017], i.e., What would have happened if something was changed? For example, [Johansson et al., 2016] used a deep neural network to learn hidden layer representations especially suited for counterfactual inference of the effects of alternative treatments on patients (using simulated data for the counterfactuals). [Hartford et al., 2017], using different objective functions than above, also used supervised deep neural networks to make counterfactual predictions. [Shalit et al., 2017]

and [Louizos et al., 2017] used a deep neural network and a variational autoencoder (VAE), respectively, for counterfactual or individual treatment effect (ITE) prediction.

Combining GANs with the ideas and methods from the domain of causality is a promising research direction, as we have already seen with the SAM algorithm [Kalainathan et al., 2018]. Another recent approach combining GANs and causality is the CausalGAN [Kocaoglu et al., 2018]. This work explores methods for adding causal structure to generative models, specifically GANs, of both observational and interventional distributions [Kocaoglu et al., 2018]. Once these generative models are trained, researchers can augment data with samples from the observational or interventional GAN. The major drawback of this work is that it requires a known causal structure a priori. For most data and research problems, the causal structure is unknown and often what is being searched for (i.e., causal structural discovery). However, this is an interesting method for creating a generative model from known causal structure that could lead to hybrid methods where only partial causal structure is known.

So far the approaches to combining deep learning and causal discovery have been diverse, but still relatively limited. Few papers combining deep learning and causal discovery have concentrated on using deep learning in a supervised manner to discover causal structure, or have attempted to directly interpret the weights in a deep neural network in a causal manner. Also, to our knowledge, there has not been any work using supervised deep learning to find latent causal relationships. The algorithms presented in this dissertation use supervised deep learning to learn latent causal relationships via direct interpretation of the weights.

2.4 Deep Learning and Evolutionary Algorithms

Evolutionary computation is a field of study that develops computational systems using principles of evolution to solve complex problems [Eiben and Smith, 2015, De Jong, 2006]. Evolutionary algorithms is a common umbrella term given to the algorithms in this field [Eiben and Smith, 2015, De Jong, 2006]. Some canonical paradigms of algorithms within evolutionary computation include:

1. Genetic Algorithms: using evolutionary algorithms for optimization and search that

traditionally represents individuals in a population as binary vectors

2. Evolutionary Strategies: using evolutionary algorithms to optimize real-valued functions, where individuals in the population are traditionally represented as vectors of real numbers
3. Evolutionary Programming: using evolutionary algorithms to evolve finite state machines
4. Genetic Programming: using evolutionary algorithms to evolve computer programs

In contrast to the definitions above, in practice, there is significant overlap between these paradigms, especially between genetic algorithms and evolutionary strategies.

Recently, there has been a increase in the number of studies using evolutionary algorithms with deep learning strategies [Stanley et al., 2019], which is broadly referred to as neuroevolution (i.e., evolving neural networks). Neuroevolution is not a new field (for example, see [Xin, 1993, Stanley and Miikkulainen, 2002]), but evolving high-dimensional deep neural networks is a relatively new endeavor [Stanley et al., 2019]. Deep learning and evolutionary algorithms have been combined in many creative ways, for example: using genetic algorithms to learn the weights of a deep neural network for deep reinforcement learning [Salimans et al., 2017, Such et al., 2017, Conti et al., 2018], using evolutionary algorithms to optimize the hyperparameters of a deep neural network, including the architecture of the network [Miikkulainen et al., 2019, Real et al., 2018, Liang et al., 2019], finding "safe" mutations (e.g., based on the gradient of the output with respect to the weights) in the weights of supervised and reinforcement learning tasks using deep neural nets and recurrent neural nets [Lehman et al., 2018], using neuroevolution for unsupervised feature learning [Szerlip et al., 2015], evolving neural networks (with an additional "plasticity" component) in a more diverse setting in hopes of discovering adaptive behavior related to neural adaptation/learning/memory [Soltoggio et al., 2017], evolving recurrent neural networks [Rawal and Miikkulainen, 2018], and using evolutionary algorithms to evolve generators in GANs [Depeweg et al., 2016].

This has been a brief survey of some of the current research combining deep learning and evolutionary algorithms. For more information and pseudocode for using an evolutionary strategy to optimize a neural network, see Section 4.5.3. To our knowledge, no algorithms have been developed that combine deep neural networks and evolutionary algorithms to infer causal structure. We develop a constrained evolutionary strategy to optimize the weights of a RINN in order to find latent causal structure in Section 4.5.3.

2.5 Gene Regulatory Networks and Neural Networks

In this work, gene regulatory networks (GRNs) can be thought of as a subset of cellular signaling pathways, as cellular signaling pathways include gene regulation, but also many additional protein-protein interactions beyond what is typically considered with a GRN. The reconstruction of GRNs—a graph with vertices representing biological molecules (e.g., DNA, RNA, proteins) and edges representing connections or relationships between molecules—is a research domain with many methods developed for this purpose, including: regression-based methods, information-theoretic models, probabilistic graphical models (e.g., Bayesian networks), methods based on clustering approaches, random forest-based methods, methods for inferring dynamic networks, and even some methods based on artificial neural networks (ANN) [Weaver et al., 1999, Zheng and Huang, 2018, Nguyen and Braun, 2017].

Relevant to the work presented in this dissertation, here we briefly describe some of the studies that used neural network-based approaches to discover GRNs. A study in 1999 by Weaver and Stormo [Weaver et al., 1999] modeled the relationships in gene regulatory networks as coefficients in weight matrices (using the familiar concepts of weighted-sums and an activation function in their model). However, they only tested their algorithm on simulated time series data as large gene expression datasets were not yet available and performing these calculations on large amounts of data was quite challenging at the time. Similarly, [Vohradsky, 2001] and [Keedwell et al., 2002] also interpreted the weights of a quasi recurrent neural network as relationships in gene regulatory networks. Again, time series simulated data was used in these studies with reasonable results. A more recent study from 2015 [Mandal et al., 2015], used a linear classifier with one input "layer" and one output "layer" (which they called a neural network) to infer regulatory relationships (as represented by the weights of the weight matrix in the linear classifier) between genes in lung adenocarcinoma gene expression data. However, they did not evaluate their learned regulatory network, did not use DNA mutation data (as we do in this dissertation), and did not use deep neural networks.

Interestingly, there have also been studies that used genetic algorithms (evolving a weight matrix of regulatory pathways) to infer gene regulatory networks [Ando and Iba, 2001]. Like the studies above, this work was ahead of its time and they were only able to

test their algorithm on simulated expression data. Improving upon [Ando and Iba, 2001], [Keedwell and Narayanan, 2005] used the weights representing a single-layer ANN (trained with gradient descent) to represent regulatory relationships between genes in gene expression data. Interestingly, they also used a genetic algorithm for a type of feature selection to make the task more tractable. They achieved good results on temporal simulated data and also tested on real temporal expression data (112 genes over nine time points), but did not have a ground truth to which to compare. In another study, [Narayanan et al., 2004] used single-layer ANNs to evaluate real, non-temporal gene expression data in a classification setting (i.e., they did not recover GRNs). However, the high-dimensionality of the problem was again a major limiting factor.

Many of the studies discussed above used the weights of a neural network to represent relationships in gene regulatory networks. However, it does not appear that these studies attempted to use a neural network to identify latent causal structure at different hierarchical levels (i.e., cellular signaling system) or to describe their methods within a causal framework of assumptions, as we do in this work. In general, the studies above used very basic versions of a neural network (without regularization—limiting the magnitude of the weights and thereby the complexity of the learned function) and due to computing constraints and the absence of large genomic datasets, were unable to train on high-dimensional data. Also, most of the methods above required temporal data, as opposed to the static genomic (and simulated) data that we utilize here. Overall, none of the studies discussed above, used a deep neural network to predict expression data from mutation data and then recover causal relationships in the weights of a deep neural network, as we do in this dissertation.

3.0 Unsupervised Deep Learning Applied to Cancer Data

In [Young et al., 2017] (see Appendix), we showed that an unsupervised DBN can be used to find meaningful low-dimensional representations of cancer gene expression data. Figure 3A shows that consensus hierarchical clustering of glioblastoma (GBM) tumors (gene expression data) using DBN hidden layer representations revealed more robust clustering results than clustering based on the raw gene expression data. This suggests that the DBN representations capture the information in gene expression data in a more relevant, digestible, or meaningful form. Figure 3B shows that the representations contain information that is clinically relevant, as the patients from the clusters in Figure 3A showed differential survival. We also found that the DBN-based representations discovered a DNA methylation phenotype (relevant to patient prognosis and GBM classification) despite not being trained on methylation data (see Figure 4 and [Brennan et al., 2013]), suggesting again, that the representations capture information or higher-level abstractions that are biologically and clinically relevant. Given these results and results of other members of our research group [Chen et al., 2015, Chen et al., 2016, Lu et al., 2018, Young et al., 2017, Tao et al., 2020], we hypothesize that deep neural networks trained on gene expression data can be trained in such a way as to generate representations that capture aspects of the cellular signaling systems (see Figure 1).

The work described above used a DBN to learn lower dimensional representations of gene expression data. A DBN is a type of autoencoder that reconstructs the input data from a lower dimensional representation, and a key aspect for the success of the experiments above to work was that the representations were lower dimensional. Forcing the data through an information bottleneck (a representation with decreased dimensionality) encourages the neural network to learn the most salient features of the data so that the input data can be accurately reconstructed from the bottleneck hidden layer (In general, this is the driving concept behind how autoencoders work). In [Young et al., 2017], it was found that at least some of the bottleneck hidden nodes had biological and/or clinical relevance. This bottleneck approach constrains what the hidden nodes can represent and encourages a form of simplicity—an alternative explanation of the data with fewer hidden nodes. Another technique for inducing

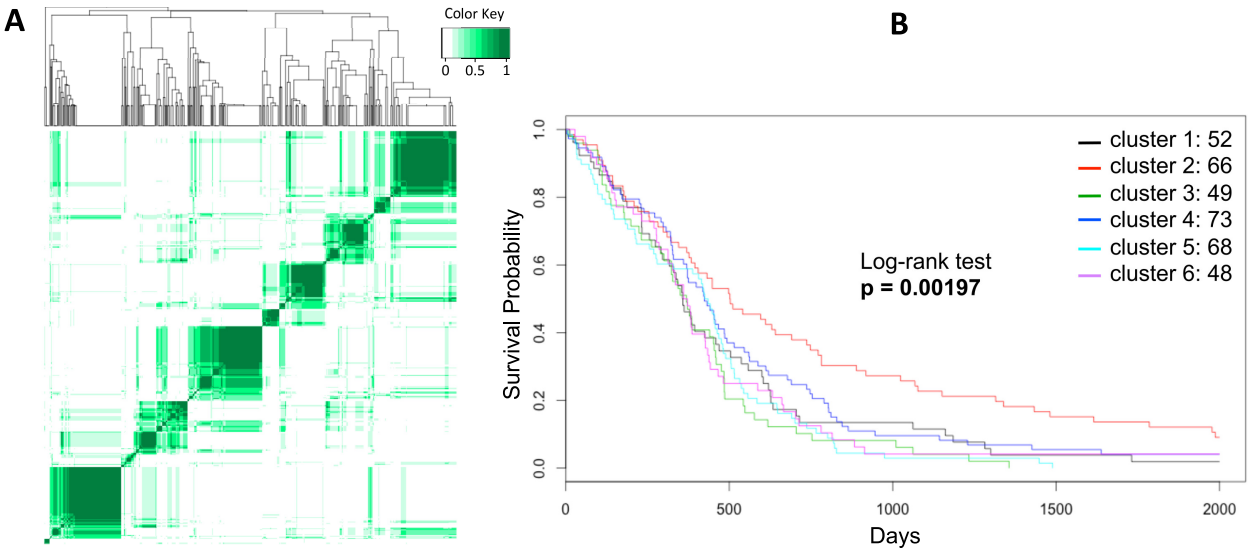


Figure 3: GBM Consensus Clustering Ensemble Results and Corresponding Kaplan-Meier Plot

(A) Heatmap and dendrogram for GBM consensus of consensus clustering results for six major clusters (k=11). Dark green corresponds to samples that always clustered together (consensus value = 1). White corresponds to samples that never clustered together (consensus value = 0). Lighter shades of green are intermediate between 1 and 0. (B) Kaplan-Meier plot for six largest clusters in part A. Each GBM cluster is represented by a colored curve in the plot. Top right of figure shows the number of samples in each cluster.

		Expression Subtype*				
			Proneural			
		Mesenchymal	Non G-CIMP	G-CIMP	Neural	Classical
Final Clusters from this Study (using deep learning and consensus clustering)	1 (black)	**44**	1	0	2	4
	2 (red)	1	**48**	**33**	1	1
	3 (green)	23	7	0	18	8
	4 (blue)	11	1	0	13	**52**
	5 (lt. blue)	12	2	0	**20**	**51**
	6 (purple)	28	18	0	6	8

*GBM sample molecular expression subtyping from Brennan et al. *Cell*. 2013

Figure 4: GBM Subtypes in each Cluster From Figure 3B Based on Molecular Subtyping from Brennan et al., 2013

simplicity, or reducing the complexity of a model is regularization of the weights of a deep learning model.

Going forward with the task of using deep neural networks to learn cellular signaling pathways, we wanted a stronger signal for learning than what unsupervised learning provided, and decided to leverage the power and success of supervised deep learning for this problem. We also decided to add a regularization term to our objective function in order to learn simpler models of genomic data instead of using a bottleneck, compression approach. We see regularization (i.e., L_1, L_2 regularization) as a possibly more robust and controllable method for inducing simplicity than relying on forcing the data through a bottleneck as described above. The remaining sections of this dissertation discuss experiments using regularized, supervised deep neural networks to learn cellular signaling pathways in genomic data. A cellular signaling pathway represents causal relationships between biological entities (i.e., proteins, protein complexes, etc.).

For more information and convenience, [Young et al., 2017] is reproduced in its entirety in the Appendix.

4.0 Deep Learning and Causal Discovery Methods Development

4.1 Summary

Most causal discovery algorithms find causal structure among a set of observed variables. In general, latent variable causal discovery methods have numerous assumptions, have difficulty with high-dimensional data, or are less relevant to the input/output setting we explore in this dissertation. Therefore, new approaches for discovering causal structures among latent causal variables would be helpful to the field. In this chapter, we address a problem for which it is known that inputs *cause* outputs, and these causal relationship are encoded by a causal network among a set of an unknown number of latent variables. We developed a deep learning model, referred to as *redundant input neural network (RINN)*, and a regularized objective function to find causal relationships between input, hidden, and output variables. More specifically, our model allows input variables to directly interact with all latent variables in a neural network to influence what information latent variables could encode in order to generate the output variables accurately. In this setting, the direct connections between input and latent variables makes the latent variables partially interpretable; furthermore, this connectivity between latent variables in the neural network reveals potential causal relationships among latent variables and their connected input variables.

4.2 Introduction

Causal discovery is a type of machine learning that focuses on learning causal relationships from data, often solely from observational data. In this context, a causal structure is represented as a graph $G = (V, E)$ containing a set of vertices V and a set of edges E, and the goal is to infer the data-generating causal structure from data. The majority of contemporary causal discovery algorithms involve searching the space of possible structures to identify the structure supported by the data, and there have been numerous algorithms developed for this

purpose [Spirtes et al., 2000, Cooper, 1999, Heinze-Deml et al., 2018, Maathuis and Nandy, 2016, Peters et al., 2017, Lagani et al., 2016].

Most causal discovery algorithms find causal structure among the observed variables of a dataset. Learning the causal structure among latent variables remains an important open problem, particularly when using high-dimensional data or attempting to leverage *a priori* knowledge that one set of observed variables causally influences a non-overlapping (disjoint) set of observed variables (i.e., the input/output setting). Latent causal structure can provide additional important insight into the causal mechanisms of a system or process. Thus, new approaches to discovering latent causal structure are needed. In this chapter, we explore the utility of using deep learning models for finding latent causal structure when the data consist of two sets of variables and it is known *a priori* that one set ("inputs") causes the other ("outputs")—and the causal path from inputs to outputs is mediated by a set of an unknown number of latent variables, among which the causal structure is also unknown.

This computational problem has important applications, including its application to cancer biology. Cancer results from somatic genomic alterations (SGAs) (e.g., mutations in DNA) that perturb the functions of signaling proteins and cause aberrant activation or inactivation of signaling pathways, which often eventually influences gene expression. Typically, an SGA directly affects the function of only one signaling protein, and its impact is transmitted through a cascade of signaling proteins to influence gene expression. Since the functional states of signaling proteins in pathways are usually not measured, the whole signaling system can be thought of as a set of hierarchically organized latent variables of which we would like to infer how changed states of some signaling proteins causally influence the state of others. In a cancer cell, when one signaling protein is perturbed by an SGA event (an observed input variable), it may causally affect the functional states of the proteins in a pathway (which are not observed) and eventually result in changed gene expression (observed output variables). Thus, our task is to learn how a set of observed input variables causally influence a set of observed output variables, through signals transmitted among a set of latent variables.

The work in this chapter concentrates on finding latent causal relationships when the causal direction between inputs and outputs is known. There has been some recent research

in the causal discovery community that deals with this problem as discussed in Section 2.3. The DM (Detect MIMIC (Multiple Indicators Multiple Input Causes)) algorithm is a causal discovery algorithm for finding latent causal relationships between inputs and outputs [Murray-Watters and Glymour, 2015, Murray-Watters, 2014]. The DM algorithm uses a series of heuristics based on conditional independence and Sober's criterion [Sober, 1998, Murray-Watters, 2014] to identify the latent causal structure. One limitation of the DM algorithm is that it constrains each latent variable to be adjacent to at least one input and one output variable. This limits the hierarchical and compositional relationships that DM can identify, which makes it possibly less accurate for cellular signalling pathway discovery.

In contrast to traditional causal discovery methods, here we develop a modified deep learning model to learn latent causal relationships. Deep learning has recently established state of the art results in many domains, largely due to the ability of deep learning algorithms to scale exceptionally well with large amounts of data and to parallelize across multiple GPUs [Schmidhuber, 2015, LeCun et al., 2015, Raina et al., 2009]. Deep learning represents a group of machine learning algorithms or strategies that originated from artificial neural networks (ANN). ANNs represent a framework, loosely inspired by biological neurons, for learning a function (represented as a set of parameters or weights) that maps inputs to outputs.

An essential characteristic of the proposed deep learning model for this work is their ability to learn compositional representations of the data [Lee et al., 2008, LeCun et al., 2015]. A deep learning model is composed of multiple layers of latent variables (hidden nodes or units) [Deng et al., 2014, Goodfellow et al., 2016, LeCun et al., 2015], which learn to represent the complex statistical structure embedded in the data, such that different hidden layers capture statistical structure of different degrees of complexity. In other words, deep learning strategies learn novel representations of the statistical structure of the input data through hierarchical decomposition (for additional discussion see [Young et al., 2017]). Researchers have found that deep learning models can represent the hierarchical organization of signaling molecules in a cell, with latent variables as natural representations of unobserved activation states of signaling molecules [Chen et al., 2016, Young et al., 2017, Chen et al., 2015, Lu et al., 2018, Tao et al., 2020].

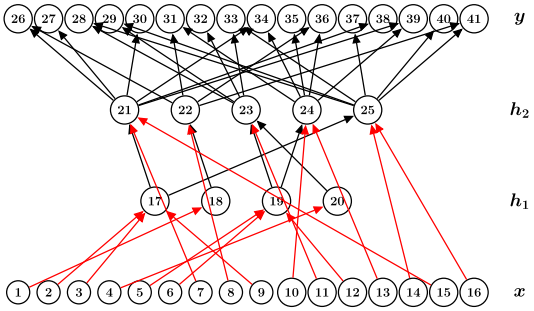
Conventional neural networks behave like black boxes in that it is difficult to interpret what

signal a hidden node in the network encodes. We hypothesize that with certain modifications inspired by the biological problem mentioned above, one may learn a partially interpretable deep neural network so that the relationships between latent variables can be interpreted as part of causal chain from input to output variable. To this end, we designed a modified deep neural network, the redundant input neural network (RINN) that allows each input variable to directly connect to each latent variable within the deep learning hierarchy. This redundant input structure allows one to learn direct causal relationships between an input variable and any latent variable. We also developed a robust pipeline for testing an algorithm's ability to capture latent causal structure, including causal simulated data inspired by cellular signaling pathways, a method for visualizing the weights of a neural network, and a method for measuring precision and recall of the causal structure.

4.3 Simulating Data From Known Causal Structure

A major goal of this work was to find hidden causal hierarchical structure between inputs and outputs. Ultimately, we are interested in biological cellular signaling systems and constructing these causal signaling pathways from mutation and expression data. However, accurate and complete ground truth cellular signaling pathway knowledge (i.e., causal structure) is not available. Ultimately, we decided that the best approach to robustly determine if a neural network can discover causal structure in its weights was to generate simulated data from an artificial causal hierarchical structure.

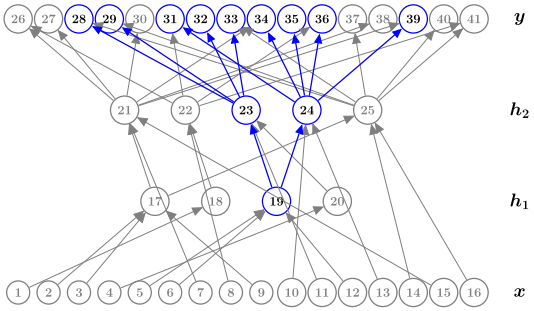
To generate simulated data, we first needed a ground truth causal structure (G_T) to generate data from. To accomplish this, we manually created a directed acyclic graph (DAG) that fit the requirements mentioned previously (i.e., hierarchical structure between a set of inputs and outputs). In the ground truth DAG (Figure 5a), the input layer is directly connected to both the first and second hidden layers to allow us to recover direct causal relationships between inputs and hidden variables. This structure is also the most biologically plausible way that mutations would affect biological entities (assuming the latent variables to be biological entities), as mutations can have an effect at any location in the signaling



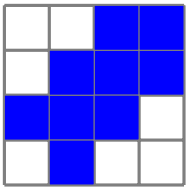
(a) Ground truth DAG

26	27	28	29
30	31	32	33
34	35	36	37
38	39	40	41

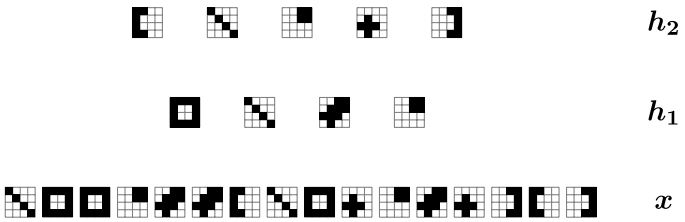
(b) $\boldsymbol{y}$ reshaped from a vector to a matrix



(c) Effect of node 19 in output space



(d) Node 19 mapped to output space. This heatmap is an alternative representation of the the causal structure in (c) from node 19 forward.



(e) Effects of all input and hidden nodes in output space. These are the ground truth patterns to look for in the weights of a neural network. Note that there are seven unique patterns, two of these seven are combinations of other patterns.

Figure 5: Ground Truth DAG for Simulated Data and Visualizing Output Space

pathway hierarchy, not just at one level (i.e., hidden layer) of the hierarchy. The ground truth DAG had 16 inputs $\boldsymbol{x}$, 16 outputs $\boldsymbol{y}$, and two hidden layers $\boldsymbol{h_1}$, $\boldsymbol{h_2}$, one with four nodes and one with five nodes (Figure 5a). In addition, we wanted an easy way to visualize the ground truth DAG, so we connected the nodes in a specific way to be able to generate visibly recognizable patterns (relative to output space) for each node in the DAG (Figure 5 and Section 4.9). Given a specific node x_i in G_T, these patterns (heatmaps or visualizations) indicate if an output node is affected by the value of x_i, i.e., Which output nodes have x_i as an ancestor? (Figure 5c, 5d).

The ground truth DAG in Figure 5a also had a biological motivation. We constructed the DAG with inputs $\boldsymbol{x}$ representing the presence or absence of DNA mutations or alterations (somatic genome alterations (SGAs)) and the outputs $\boldsymbol{y}$ representing differentially expressed genes (DEGs). Mutation and expression data are readily available from many sources, (e.g., The Cancer Genome Atlas (TCGA) [Weinstein et al., 2013]). Between these input and output layers, we envisioned a biological signaling pathway [Young et al., 2017] with the hidden layer $\boldsymbol{h}_1$ closest to the inputs representing major biological pathways and $\boldsymbol{h}_2$ representing transcription factors. Biologically, mutations (i.e., input nodes $\boldsymbol{x}$) directly target transcription factors and proteins higher up in the hierarchy (e.g., cell membrane receptors). To reflect this in our ground truth DAG, we allowed input mutations in $\boldsymbol{x}$ to directly connect to nodes in $\boldsymbol{h}_1$ or $\boldsymbol{h}_2$. Each mutation affected only one node, but each node could be affected by multiple mutations. This is similar to how biological mutations function. Our ground truth DAG, G_T, is a crude representation of a biological signaling pathway, and one obvious omission in the DAG abstraction are feedback cycles, which are prevalent in biological cellular signaling pathways.

Another goal with using simulated data in this research, was to see how well a neural network learned causal relationships in its weight matrices when the simulated data was binary versus continuous, and noisy versus non-noisy. This allowed us to make general predictions about the limitations of these algorithms as a function of the characteristics of the data. To accomplish the above, we generated six different simulated datasets. All datasets had 16 inputs, 16 outputs, and 5,000 instances.

4.3.1 Dataset 1: Matrix Multiplication with Interventions (no noise; binary input; positive integer-valued output)

For Dataset 1, we simulated data from the ground truth DAG by having the inputs *intervene* on their adjacent hidden nodes. An "active" (i.e., 1) input intervention node would set the value of an adjacent hidden node to 0 for that sample (Algorithm 1). This idea for simulating data was intended to capture the effect of a biological loss-of-function mutation on the mutation's associated transcription factor or protein complex.

We represented the ground truth causal DAG, G_T (Figure 5a), as a set of weight matrices (each matrix representing the edges between two adjacent layers of nodes in G_T, similar to how the weights of a neural network are typically represented). If an edge was present between two nodes in G_T, then the weight value between these two nodes in the corresponding weight matrix would be set to 1, otherwise the weight is set to 0. The only exception to this was for the weight matrix, $\boldsymbol{W}_x$, relative to the input layer. Notice that the input layer is directly connected to both the first and second hidden layers. We represented these edges also with a single weight matrix ($\boldsymbol{W}_x$), where the number of rows was the number of input nodes and the number of columns was the number of nodes in $\boldsymbol{h}_1$ plus the number of nodes in $\boldsymbol{h}_2$. For G_T, this means this weight matrix ($\boldsymbol{W}_x$) would be 16×9, and contain a value of 1 if there was an edge between an input (rows) and a hidden node (columns). Each input x is connected to a single hidden node. But the hidden nodes have multiple inputs connected to them.

We simulated data in this section by imagining the inputs as acting as interventions on the hidden nodes they are connected to. An "active" intervention here would set the value of a node to 0 for that sample.

To accomplish this, for each simulated instance, we sampled $\boldsymbol{x}$ (inputs) from a Binomial distribution with Bernoulli success probability (probability that a mutation is present) $p = 0.10$ and $n = 16$ (number of output values).

$$\boldsymbol{x} \sim \mathcal{B}(n, p) \tag{4.1}$$

Next, we set the values of the first hidden layer $\boldsymbol{h}_1$ to 1 (i.e., all nodes are "active"). Next, we looked to see if any of these values were intervened upon by looking at our sampled

Algorithm 1 Simulate Data using Matrix Multiplication and Interventions

p is the Bernoulli success probability ▷ probability that a mutation is present

s is the number of samples we want to generate

$\boldsymbol{Data}$ is a container for the simulated data

$\boldsymbol{W}_l$ is the weight matrix between hidden layer $l-1$ and l

$\boldsymbol{h}_l$ is the vector of values representing hidden layer l

$p = 0.10$

for $i = 1$ to s **do**

SAMPLE $\boldsymbol{x} \sim \mathcal{B}(16, p)$ ▷ Bernoulli Trials

$\boldsymbol{h}_1 = [1, 1, 1, 1]^\top$

for $j = 1$ to 2 **do**

set nodes in $\boldsymbol{h}_j$ targeted by $\boldsymbol{x}$ to 0 ▷ use $\boldsymbol{W}_x$

$\boldsymbol{h}_{j+1} = \boldsymbol{h}_j \boldsymbol{W}_{j+1}$

end for

SAVE $\boldsymbol{x}$ and $\boldsymbol{h}_{j+1}$ in $\boldsymbol{Data}$

end for

$\boldsymbol{x}$ to see if any targeted $\boldsymbol{h}_1$. If yes, we clamped the targeted nodes to 0 to get a slightly modified $\boldsymbol{h}_1$. Next, we vector-matrix multiplied $\boldsymbol{h}_1$ by $\boldsymbol{W}_2$ to get the values for $\boldsymbol{h}_2$. $\boldsymbol{W}_2$ is the weight matrix between $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$. We looked to see if any of the values in $\boldsymbol{h}_2$ were intervened upon by the current $\boldsymbol{x}$ and if so, we clamped the targeted nodes in $\boldsymbol{h}_2$ to 0. And as before, we vector-matrix multiplied $\boldsymbol{h}_2$ by $\boldsymbol{W}_3$ (weight matrix between $\boldsymbol{h}_2$ and $\boldsymbol{y}$) to get $\boldsymbol{y}$ (See Algorithm 1 for pseudocode). This method for simulating data leads to $\boldsymbol{y}$ values that are positive integer-valued.

4.3.2 Dataset 2: Linear Gaussian SEM (high noise; input and output $\in \mathbb{R}$)

$$\boldsymbol{b} \sim \mathcal{N}(0, \sigma^2) \tag{4.2}$$

$$\sigma^2 \sim \mathcal{U}(1, 3) \tag{4.3}$$

$$\boldsymbol{W} \sim \pm\mathcal{U}(0.5, 1.5) \tag{4.4}$$

Let G_T be our ground truth DAG with nodes $\boldsymbol{N}$ (input, hidden, and output), N_j^l be node j at layer l, X_i^{l-1} be the value of parent (i.e., direct cause) i in layer $l-1$. Then we simulate data using a structural equation model (SEM) to calculate the value of each node N_j^l in G_T as

$$N_j^l = \sum_{i \in parents_{G_T}(j^l)} W_{ij}^l X_i^{l-1} + b_j^l \tag{4.5}$$

4.3.3 Dataset 3: OR Logical Operator (high noise; binary)

We found simulating noisy binary data challenging. To accomplish this, we modeled the value of a node in our ground truth graph G_T as the output of a logical OR applied to the values of the parents of said node. First, we sampled a probability, p, to represent

$$P(X = 1 | parents_{G_T}(X)) \tag{4.6}$$

(a Bernoulli distribution) for all possible binary combinations of a node's parents' values. We sampled p from Beta distributions peaked close to 0.10 or 0.90 depending on the values of the parents (i.e., OR operator applied to the values of the parents) to add noise to the OR operations. This sampling from Beta distributions provided a collection of Bernoulli distributions for all possible binary combinations of parents. We then used these Bernoulli distributions to generate noisy data from OR operators. For example, to model an OR operator with two parents, we would sample four p values, each representing a Bernoulli distribution as follows,

$$P(x = 1|Pa_{1,G_T} = 1, Pa_{2,G_T} = 1) \sim Beta(90, 10) \tag{4.7}$$

$$P(x = 1|Pa_{1,G_T} = 0, Pa_{2,G_T} = 1) \sim Beta(90, 10) \tag{4.8}$$

$$P(x = 1|Pa_{1,G_T} = 1, Pa_{2,G_T} = 0) \sim Beta(90, 10) \tag{4.9}$$

$$P(x = 1|Pa_{1,G_T} = 0, Pa_{2,G_T} = 0) \sim Beta(10, 90) \tag{4.10}$$

where Pa_{1,G_T} indicates parent 1 of node x in graph G_T.

The distributions are noisy because $P(X = 1|parents_{G_T}(X))$ is not equal to 0 or 1, as it normally would be for non-noisy OR operators. Also, the Bernoulli distributions are sampled, meaning they are not all $p = 0.10$ or $p = 0.90$. This means that some operators will be considerably noisier than others. After generating all Bernoulli distributions, we sample values for each node in the ground truth DAG given the values of the parents for that node and the corresponding Bernoulli distribution (See Algorithm 2 for pseudocode, but assume all operators are OR operators).

4.3.4 Dataset 4: AND, OR, XOR Logical Operators (moderate noise; binary)

To increase the complexity of our simulated data, we added AND and XOR operators to replace some of the OR operators in Section 4.3.3 (Algorithm 2) and followed the same

procedure as Section 4.3.3. In contrast to dataset 3, we sampled p from Beta distributions peaked close to 0.05 and 0.95 for dataset 4, meaning that dataset 4 is less noisy than dataset 3.

Just as in Section 4.3.3, we sampled a probability, p, to represent

$$P(X = 1|parents_{G_T}(X)) \tag{4.11}$$

for all possible binary combinations of a node's parents' values, and considering what operator the node represented (i.e., AND, OR, or XOR). Again, this sampling from Beta distributions provided a collection of Bernoulli distributions for all possible binary combinations of parents. We then used these Bernoulli distributions to generate noisy data from AND, OR, or XOR operators. For example, to model an AND operator with two parents, we would sample four p values, each representing a Bernoulli distribution as follows,

$$P(x = 1|Pa_{1,G_T} = 1, Pa_{2,G_T} = 1) \sim Beta(95, 5) \tag{4.12}$$

$$P(x = 1|Pa_{1,G_T} = 0, Pa_{2,G_T} = 1) \sim Beta(5, 95) \tag{4.13}$$

$$P(x = 1|Pa_{1,G_T} = 1, Pa_{2,G_T} = 0) \sim Beta(5, 95) \tag{4.14}$$

$$P(x = 1|Pa_{1,G_T} = 0, Pa_{2,G_T} = 0) \sim Beta(5, 95) \tag{4.15}$$

We chose to change the α and β values for the Beta distributions (making the data less noisy) for these slightly more complex operators to evaluate the effect of noise on the algorithms and to balance the complexity of the operators with the amount of noise (See Algorithm 2 below for pseudocode for simulating this dataset).

4.3.5 Dataset 5: TCGA + OR

We wanted to test our algorithms on larger, biologically-related datasets, as we developed the algorithms in this dissertation for eventual use on biological datasets. However, biological

Algorithm 2 Simulate Data from AND, OR, XOR Logical Operators

$G_T = (V, E)$ ▷ ground truth DAG with vertices V and edges E.

PARENTS(*node*) returns the binary values of the parents of *node*

$\boldsymbol{X}$ is a $n \times m$ matrix

$\boldsymbol{Data}$ is a $n \times m$ matrix

$\boldsymbol{D}$ is a container of Bernoulli distributions

s is the number of samples we want to generate

for *node* in V **do**
  op = RANDOM_CHOICE(AND, OR, XOR)
  for each possible binary combination, b, of the parents of *node* **do**
    if $op(b)$ is $True$ **then**
      SAMPLE $D_{node,b} \sim Beta(95, 5)$
    else if $op(b)$ is $False$ **then**
      SAMPLE $D_{node,b} \sim Beta(5, 95)$
    end if
  end for
end for

SAMPLE $\boldsymbol{X} \sim \mathcal{U}(0, 1)$
for $i = 1$ to s **do**
  for *node* in V **do** ▷ start from input nodes and go to output nodes
    b = PARENTS(*node*)
    $Data_{i,node} = D_{node,b} < X_{i,node}$
  end for
end for
return $\boldsymbol{Data}$

datasets with mutation and expression data; as well as, complete and accurate ground truth causal structure, to our knowledge, do not exist. So our solution was to embed simulated data within a larger biological dataset and see if the known ground truth causal structure could still be discovered within the much larger dataset, and to also evaluate how the different algorithms performed simply on the prediction task for this biologically related dataset (i.e., which algorithm had the lowest prediction error).

To accomplish this, we obtained mutation and expression data from The Cancer Genome Atlas (TCGA) [Weinstein et al., 2013] and used an algorithm from [Cooper et al., 2018, Cai et al., 2019] called TCI (Tumor-specific Causal Inference) to binarize the data and perform feature selection on both the mutation and expression data. Please see [Cooper et al., 2018, Cai et al., 2019] for a detailed explanation of how TCI works.

A benefit of this approach was that we could use the results on the embedded simulated data to tune the regularization rate parameter for the entire dataset based on the precision and recall for the embedded simulated data only. Determining the regularization rate is nontrivial and may be the most important component of whether or not an algorithm can learn the correct causal structure, so any guidance we can get to set this hyperparameter could be of great benefit. As the purely biological data has no ground truth causal structure, we had no way to calculate precision and recall for the biological data. Assuming the biological causal structure is somewhat similar to the simulated data causal structure, tuning the regularization rate hyperparameter based on the precision and recall of the embedded simulated data may help us to more quickly identify the correct causal structure in biological data.

Briefly, after running the TCGA data through TCI, we obtained the following data for each of the 5,000 cancer tissue samples: 1. the presence or absence of 634 gene mutations and 2. the differential expression for 10,120 genes. The 634 mutations were selected by TCI as cancer driver (i.e., causing) mutations. And the 10,120 differentially expressed genes (DEGs) represented the DEGs regulated by the 634 drivers. This dataset was still too large for our purposes, so we performed feature selection by variance on the 10,120 DEGs to reduce the number of DEGs down to 5,168. This was accomplished by removing any DEGs with a variance (measured across the samples) less than

$$v = p(1 - p) \tag{4.16}$$

with $p = 0.8$.

We hypothesized that the DEGs that are most relevant for this task are the DEGs that vary the most across the dataset. Next, we ran a neural network classifier using the 634 mutations as input to predict the values of the 5,168 DEGs. We obtained Area Under the ROC Curve (AUROC) values for each DEG and then ranked the DEGs based on the AUROCs from highest to lowest. We selected the DEGs with AUROCs greater than 0.85. This ultimately led to final dataset with 634 mutations and 68 DEGs for 5,000 tissue samples.

Next, we embedded simulated Dataset 3 (OR operator) into the feature-reduced TCGA dataset described above, by simple concatenation. This generated a final dataset with 650 inputs and 84 outputs for 5,000 samples.

4.3.6 Dataset 6: TCGA + OR, XOR, AND

Following the same procedure detailed in Section 4.3.5 and using the same feature-reduced TCGA data, we embedded simulated Dataset 4 (AND, OR, XOR operators) into the TCGA data. Just like Dataset 5, Dataset 6 consisted of a 5000×650 input matrix ($\boldsymbol{X}$) and a 5000×84 output matrix ($\boldsymbol{Y}$).

4.4 Redundant Input Neural Network (RINN)

In this dissertation, we wanted to be able to model inputs that directly connected to any hidden layer in a neural network (again, as this is closest to how mutations in DNA would impact cellular-signaling pathways). To accomplish this, we developed the Redundant Input Neural Network (RINN) (Figure 6 and Algorithm 3). The RINN has an extra copy of the input $\boldsymbol{x}$ directly connected to all hidden layers after the first hidden layer. In contrast to the RINN, the inputs of a conventional feedforward DNN are only connected to the first hidden layer. Just like a DNN, the RINN is trained through backpropagation and stochastic gradient descent (see Sections 2.1, 4.5.1), it just has many more weights in the network.

Each hidden layer of a RINN with redundant inputs is calculated according to

$$\boldsymbol{h}^{(i)} = \phi([\boldsymbol{h}^{(i-1)}, \boldsymbol{x}] \cdot \boldsymbol{W}_i) + \boldsymbol{b_i} \tag{4.17}$$

where $\boldsymbol{h}^{(i-1)}$ represents the previous layer's output vector, $\boldsymbol{x}$ is the vector input to the neural network, $[\boldsymbol{h}^{(i-1)}, \boldsymbol{x}]$ represents concatenation into a single vector, $\boldsymbol{W}_i$ represent the weight matrix between hidden and redundant nodes in layer $i-1$ and hidden layer i, ϕ is a nonlinear function (e.g., ReLU), $\cdot$ represents vector-matrix multiplication, and $\boldsymbol{b_i}$ represents the bias vector for layer i. In contrast to a RINN, a plain DNN calculates each hidden layer as

$$\boldsymbol{h}^{(i)} = \phi(\boldsymbol{h}^{(i-1)} \cdot \boldsymbol{W}_i) + \boldsymbol{b_i} \tag{4.18}$$

In order to use a neural network to capture the ground truth causal structure in the weights, the weights needed to be regularized or constrained in some way. We evaluated both L_1 and L_2 regularization and L_1 provided far superior results for finding the causal structure of our ground truth DAG (results not shown). The objective functions for all neural network-based strategies used in this dissertation included L_1 regularization of the weights. Only the fine-tuning objective function for the DBN included L_1 regularization.

Another difference between the RINN and a DNN in this research is that the RINN was always trained with eight hidden layers (i.e., the number of hidden layers in a RINN was always set to eight). This is because we hypothesize (for eventually using this in a biological setting) that most biological pathways will not have more than an 8-level hierarchy. Importantly, since copies of the input are available to use at each hidden layer (should the algorithm decide to use it), we don't need to determine the best number of hidden layers (as a model selection hyperparameter)—this means less model selection is required for RINNs. If the optimal way to learn the structure in the data with a RINN is by using a 3-hidden layer network, the RINN will learn nonzero weights starting with the copy of the input that is connected to hidden layer 6. This would create a 3-hidden layer network (hidden layers 6, 7, and 8), with all weights in the network before the redundant input connected to hidden layer 6 being equal to 0.0. This happens because of the L_1 regularization used in the objective function, which constrains the RINN to map input to output using as few weights as possible. This 8-level hierarchy constraint doesn't necessarily just apply to biological pathway data,

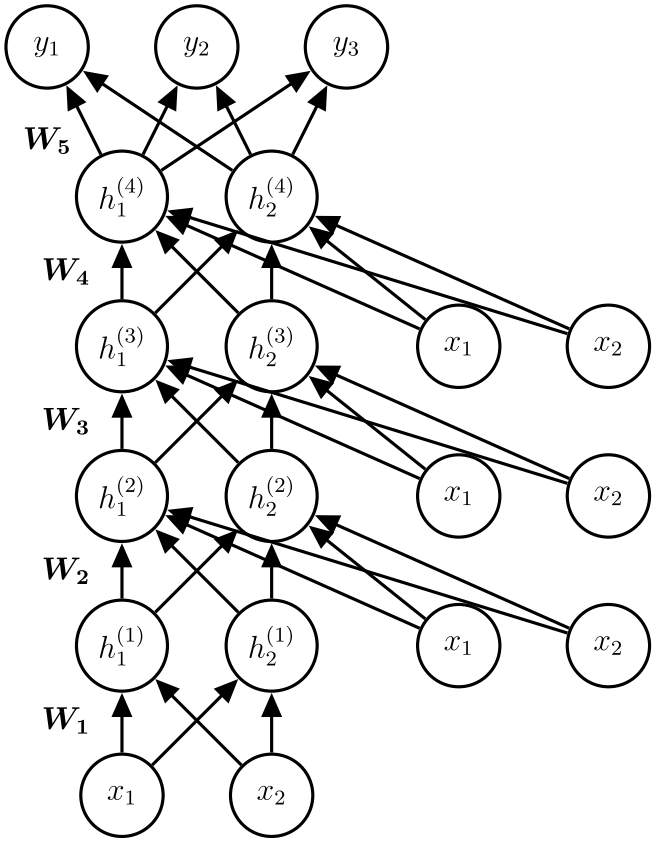


Figure 6: Simple Redundant Input Neural Network (RINN)

A RINN with four hidden layers ($h^{(1)}$, $h^{(2)}$, $h^{(3)}$, $h^{(4)}$, each with two nodes), two inputs (x_1, x_2 on bottom), three outputs (y_1, y_2, y_3), and three sets of redundant inputs (x_1, x_2 on right side). Each node represents a scalar value and each edge represents a scalar weight. The weights between layers are collected in weight matrices $\boldsymbol{W_1}$, $\boldsymbol{W_2}$, $\boldsymbol{W_3}$, $\boldsymbol{W_4}$, and $\boldsymbol{W_5}$.

but might be sufficient for most data in general. However, if there is reason to think that there may be more hierarchical levels in a particular dataset, the number of maximum hidden layers can always be increased with a RINN, or treated as a hyperparameter.

Algorithm 3 Redundant Input Neural Network (RINN)

$\boldsymbol{W}_{l,a}$ is the weight matrix between hidden layer $l-1$ and hidden layer l

$\boldsymbol{W}_{l,b}$ is the weight matrix between input $l-1$ and hidden layer l

$\boldsymbol{h}_l$ is the vector of values representing hidden layer l

$\boldsymbol{x_i}$ is the input vector for instance i

$\boldsymbol{y}_i$ is the output vector for instance i

s is the number of samples in the dataset

$ERROR$ is the objective function to be optimized. In this case, binary cross-entropy.

for $i = 1$ to s **do**

$\boldsymbol{a}_1 = ReLU(\boldsymbol{x}_i \cdot \boldsymbol{W}_{0,b})$

for $j = 1$ to num_hid_layers **do**

$\boldsymbol{a}_j = \text{CONCATENATE}(\boldsymbol{a}_j, \boldsymbol{x}_i)$

$\boldsymbol{W}_j = \text{CONCATENATE}(\boldsymbol{W}_{j,a}, \boldsymbol{W}_{j,b})$

$\boldsymbol{a}_{j+1} = ReLU(\boldsymbol{a}_j \cdot \boldsymbol{W}_j)$

end for

$\hat{\boldsymbol{y}}_i = sigmoid(\boldsymbol{a}_{j+1} \cdot \boldsymbol{W}_{j+1,a})$ ▷ for binary $\boldsymbol{y}$

Update all $\boldsymbol{W}$ by descending their gradient:

$$\nabla_{\boldsymbol{W}} ERROR(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i)$$

end for

4.5 Other Deep Learning Strategies

4.5.1 DNN

We compared the RINN with a feedforward DNN. A DNN learns a function mapping inputs ($\boldsymbol{x}$) to outputs ($\boldsymbol{y}$) according to

$$f(\boldsymbol{x}) = \phi(...\phi(\phi(\boldsymbol{x}\boldsymbol{W}_1)\boldsymbol{W}_2)...\boldsymbol{W}_n) = \hat{\boldsymbol{y}} \tag{4.19}$$

where $\boldsymbol{W}_i$ represent the weight matrices between layers of a neural network, ϕ is some nonlinear function (i.e., activation function such as ReLU, sigmoid, or tanh), and $\hat{\boldsymbol{y}}$ is the predicted output. DNNs also have bias vectors added at each layer, which have been omitted from the above equation for clarity. Each hidden layer of a DNN is calculated according to

$$\boldsymbol{h}^{(i)} = \phi(\boldsymbol{h}^{(i-1)} \cdot \boldsymbol{W}_i) + \boldsymbol{b_i} \tag{4.20}$$

where $\boldsymbol{h}^{(i-1)}$ represents the previous layer's output vector, $\boldsymbol{W}_i$ represent the weight matrix between hidden and redundant nodes in layer $i-1$ and hidden layer i, ϕ is a nonlinear function (e.g., ReLU), $\cdot$ represents vector-matrix multiplication, and $\boldsymbol{b_i}$ represents the bias vector for layer i.

For more details about DNNs, see Section 2.1 and [Goodfellow et al., 2016].

4.5.2 DBN

A deep belief network or DBN [Hinton and Salakhutdinov, 2006] is a type of autoencoder, an unsupervised DNN. This means that the DBNs used here were trained with only $\boldsymbol{y}$ data, i.e., DBNs learned to map $\boldsymbol{y}$ to a lower dimensional representation of $\boldsymbol{y}$ (in the hidden layers) and then to reconstruct $\boldsymbol{y}$ from the lower dimensional representation. Error is measured by comparing $\boldsymbol{y}$ to $\hat{\boldsymbol{y}}$. We evaluated the trained "decoder" network only (i.e., the second half of the network going from smallest hidden layer to $\boldsymbol{y}$) for evidence of G_T latent variables. We had difficulty recovering any of the causal ground truth graph (Figure 5a) with a default DBN, but adding L_1 regularization to the fine-tuning step objective function allowed us to recover at least some of the causal structure. For more details on what a DBN is, see Section

2.2. For a more detailed explanation of a DBN (including a pre-training step not mentioned here) and specifically its use in a biological context, see [Young et al., 2017].

4.5.3 Constrained Evolutionary Strategy

When using the RINN (or DNN) to find causal structure, it is necessary to look at the values of all the weights in a trained network and, based on a threshold, decide which weights are significant and which are not. This is necessary because neural networks are fully connected, meaning that each input/hidden node is connected to each hidden node in the next layer. Once a RINN is trained, we need to interpret the weights to generate a causal graph. Using L_1 regularization causes the network to set most of the values in the weight matrices to a small number, but in our experiments very few of the weights are actually zero—rather there are many weights with values in the interval $[0.0001, 0.5]$. This requires that we set a threshold, above which weights will be interpreted as a causal edge (often this threshold was near 0.1 for simulated data). This thresholding problem can be solved by using a non-differentiable optimization procedure that limits the allowed values for weights in a neural network (e.g., an extreme example would be restricting the weight values to 1 or 0 only).

Neural networks are trained using gradient descent optimization. This means that the gradient of the loss function with respect to the weights of the network must be calculated (this calculation uses the backpropagation algorithm), requiring that the loss function for gradient descent-based optimization be differentiable. Genetic algorithms can be used to optimize non-differentiable loss functions, allowing for more flexibility when choosing how to best evaluate the models of an experiment.

Recently, there has been an increased interest in the application of evolution-based optimization to deep learning and reinforcement learning with promising outcomes [Salimans et al., 2017]. In particular, genetic algorithms have been used to successfully optimize four million parameter deep neural networks with performance at or better than current SGD based methods [Such et al., 2017]. In addition to having competitive performance on reinforcement learning based tasks, evolutionary algorithms have additional benefits. I have

already mentioned the benefit of using a non-differentiable fitness or loss function above, which allow for constraints on weight values that can increase interpretability. Non-differentiable function optimization opens up many possibilities for fruitful modifications to a fitness function and neural network based-algorithm, thus dramatically increasing flexibility. Evolutionary algorithms are also highly parallelizable, as the population of individuals can be evolved mostly independently of other individuals. There is also some evidence that evolutionary algorithms can estimate better structure and converge faster than typical Bayesian network structure learning approaches [LarrañAga et al., 2013, Wong et al., 1999, van Dijk and Thierens, 2004]. It is also suggested that evolutionary algorithms "have been able to perform a more effective and diverse search of the vast solution space of different problems." [LarrañAga et al., 2013]. The benefits, flexibility, and competitive performance of evolutionary algorithms suggest that they are a reasonable alternative to gradient-based methods for causal structure search, especially for the structure search algorithms presented in this dissertation.

We developed an evolutionary strategy (ES) to optimize the weights of a RINN (instead of using gradient descent) in order to explore other optimization strategies that make it easier to interpret the weights of a neural network as causal relationships. Specifically, we developed a constrained evolutionary strategy (ES-C), where we constrained the possible values for the weights (e.g., $weight \in \{-1, -0.5, 0, 0.5, 1\}$). We represented an individual in the population to be evolved as a vector of weights. This vector of weights represents a neural network. Initially these weights were set to all zeros, then if a weight was selected for mutation (as dependent on a mutation rate), it was randomly changed to a weight in the range of legal weight values. If a RINN optimized with an evolutionary strategy in this way is able to successfully map the input to output, then interpreting the weights as causal relationships will be much simpler—a nonzero weight represents a causal edge.

We performed model selection over multiple hyperparameters including, elite ratio (percent of population saved to mate and produce the next generation), mutation rate, regularization rate, and the legal weight values. The architecture of the neural network was set to eight layers of eight hidden nodes (because we knew this would be sufficient based on previous experiments with these simulated datasets). We used a population size of 200 and evolved for up to 13,000 generations or epochs. For a fitness function, we used L_1 regularization plus

either binary cross-entropy error or squared error depending on the dataset. The *MATE* function involved choosing two individuals in the elite population and randomly combining 50% of the weights from one individual with 50% of the weights from the other individual. The evolutionary strategy used in this work was *not* setup to be parallelized across processors, as early prototyping experiments suggested that ES-C would not perform nearly as well as RINN. This, unfortunately, reduced the amount of model selection we could perform in a timely manner. To speed up ES-C, we also only trained on 20% of the data that was used to train the other algorithms. Testing 500 sets of hyperparameters on one dataset took approximately three weeks on a single desktop computer. Please see Algorithm 4 pseudocode for the evolutionary strategy we used in this work.

Algorithm 4 Genetic Algorithm for Neural Network Weight Optimization

$\boldsymbol{f}$ is a vector of fitness values

$\boldsymbol{Data}$ represents all input and output data

W_i represents all the weights of neural network i reshaped into a vector

pw represents the set of all possible weight values (e.g., $\{-1, -0.5, 0, 0.5, 1\}$)

for $i = 1$ to $num\_generations$ **do** ▷ for each generation
  for $j = 1$ to $population\_size$ **do** ▷ for each neural network
    f_j = FITNESS(W_j, $\boldsymbol{Data}$)
  end for
  $elites$ = TOP_20_PERCENT($\boldsymbol{f}$) ▷ get best performing neural nets
  for $k = 1$ to p **do**
    W_k = MATE(RANDOM_CHOICE($elites$), RANDOM_CHOICE($elites$))
    W_k = MUTATE(W_k, pw) ▷ change some weights to random value from pw
  end for
end for
RETURN W_{best} ▷ neural network with highest fitness function

4.6 DM Algorithm

The DM (Detect MIMIC—a psychological model with unmeasured latents between inputs and outputs) algorithm is a fast causal discovery algorithm (with similar assumptions as the RINN) that finds latent structure between input and output data [Murray-Watters, 2014, Murray-Watters and Glymour, 2015]. The DM algorithm needs to know which variables are input variables and which are output variables (or it can run the PC algorithm to determine which are input and which are output). Next, it performs routine statistical tests to find out which input and output variables are dependent on one another. It then partitions the input variables according to whichever inputs were found to be dependent on the same set of output variables. For each of these partitions, it adds a latent variable between the inputs and outputs, and adds edges from all inputs to the latent variable and from the latent variable to all outputs. Next, it adds a directed edge between latent variables if one of the latent variable's dependent outputs is a subset of the other latent variables dependent outputs. Finally, it prunes some of these edges based on conditional independence tests on the outputs (Sober's Criterion [Sober, 1998]—see The Steps of the DM Algorithm below).

As another comparison, we evaluated the performance of the DM (Detect MIMIC) algorithm [Murray-Watters and Glymour, 2015] on the first four simulated datasets. This will allow us to compare our neural network-based approaches for finding causal structure to a very different and more traditional causal discovery based approach. The DM algorithm requires that you specify which variables are input variables and which variables are output variables, but does not require that you specify anything about the hidden variables. The DM algorithm takes two tuning parameters: an alpha level for Fisher's Z test of conditional independence and an alpha level for Sober's criterion.

We performed an abbreviated grid search selecting values for the first alpha level $\alpha \in \{0.05, 0.01, 0.001, 0.0001\}$ motivated by [Ramsey and Andrews, 2017] and values for the second alpha level $\alpha \in \{0.1, 0.05, 0.01\}$. The best performing combination of tuning parameters for the DM algorithm on each of the first four simulated datasets are reported and compared against the other algorithms in Tables 2, 3. For a more detailed description of the DM algorithm and its assumptions, see [Murray-Watters and Glymour, 2015, Murray-Watters,

2014].

The Steps of the DM Algorithm:

(1) Separate data into input X and output Y variable sets.

(2) Routine statistical tests to determine dependence between subsets of X and subsets of Y. This step separates the observed variables into partitions $P_i = \{X_i, Y_i\}$ of the dataset with dependency between X_i and Y_i, where $X_i \subset X$ and $Y_i \subset Y$.

(3) Latent variables L are added between X and Y for each partition.

(4) Add edges from X to corresponding L.

(5) Procedure to add edges from L to Y, with only one edge into each Y.

(6) Procedure to add edges from L to L.

(7) Use Sober's criterion to remove edges. Sober's criterion:

$$Y \leftarrow X \rightarrow Z \ , \qquad Y \perp\!\!\!\perp Z|X$$

$$Y \leftarrow U \rightarrow Z \text{ and } X \rightarrow U \ , \qquad Y \not\perp\!\!\!\perp Z|X$$

(8) Return learned causal structure.

The assumptions required by the DM algorithm are somewhat similar to the assumptions required for the RINN. Overall, both have restrictive assumptions (see Section 4.7 for RINN causal assumptions). There are a few differences between the assumptions made by each algorithm. The DM algorithm requires each latent variable to have have at least one observed cause and one observed effect; however, the RINN can represent a latent variable without an observed cause through the use of a nonzero bias node. The DM algorithm also assumes that there is only one directed path between any two observed variables, meaning that each output variable has only one directed edge into it. *This is a very limiting assumption and the RINN algorithm does not assume this.* Biologically, this assumption is violated as there is often redundancy in cellular signaling pathways and many ways to activate a given protein (e.g., transcription factor). The DM algorithm is also dependent on conditional independence tests, which have additional hyperparameters that need to be set. In general, the DM algorithm has more constraints than the RINN algorithm. However, the correctness of the RINN is

dependent upon reaching the global optimum, which is unlikely when performing stochastic gradient descent to optimize neural networks. So, in practice, it is difficult to prove correctness with the RINN. *Another major difference between the RINN and the DM algorithm is that the DM algorithm only returns a causal structure (without parameterization), whereas the RINN returns a causal structure and parameterization.* The DM algorithm is available in Tetrad (http://www.phil.cmu.edu/tetrad/, https://github.com/cmu-phil/tetrad).

4.7 Causal Assumptions (Neural Network)

The following are assumed to be true regarding the true graph and data generating process. These assumptions allow the weights learned by a neural network to be interpreted as causal relationships.

A1. Causal Markov Assumption: D-separation (a criterion for determining independence between vertices in a causal graph) in the true data-generating graph G_T implies conditional independence in the data distribution.

A2. $\boldsymbol{x} \to\to \boldsymbol{y}$: We assume it is known *a priori* that $\boldsymbol{x}$ causes $\boldsymbol{y}$ and that there is at least one latent variable on the path from $\boldsymbol{x}$ to $\boldsymbol{y}$. For example, mutations in DNA cause changes in gene expression through modifications to cell signaling. The methods developed in this study can be used with any datasets where the causal direction between two sets of variables is known.

A3. Causal Sufficiency: Given $\boldsymbol{x}, \boldsymbol{y}$, the observed input and output variables (i.e., all observed variables), there are no latent variables Z, $Z \notin \boldsymbol{x}$, $Z \notin \boldsymbol{y}$, such that Z causally influences at least one variable in $\boldsymbol{x}$ and one variable in $\boldsymbol{y}$ (i.e., latent confounding). In a biological context, with $\boldsymbol{x} = \boldsymbol{SGA}$ and $\boldsymbol{y} = \boldsymbol{DEG}$, this assumption is generally true.

A4. We assume away selection bias.

A5. The true graph has no latent to latent cycles (acyclic): In a biological context, this assumption is violated by almost all signaling pathways as feedback is a common characteristic of molecular signaling pathways. Assuming acyclicity is a limitation of using the methods in this study.

A6. Faithfulness: Independence in the data distribution implies d-separation in the true data-generating graph G_T. This can be seen as an appeal to Occam's razor—preferring the most parsimonious graph over a denser graph. L_1 regularization encourages learning the most parsimonious graph.

A7. Latent Variable Identifiability: The latent variables are *identifiable*.
A latent variable is *identifiable* if, 1. the intersection of its ancestors in G_T and $\boldsymbol{x}$ is unique with respect to the other latent variables, *AND* 2. the intersection of its descendants in G_T and $\boldsymbol{y}$ is unique with respect to the other latent variables.

A8. The ground truth causal relationships can be modeled with one or multiple affine transformations plus nonlinear functions (hidden layer calculations in a neural network).

The causal ground truth DAG in Figure 5a and simulated data adhere to these assumptions. The *structure* learned by the RINN represents the underlying data generating DAG if the RINN reaches the global optimum and the assumptions above are not violated. This means that the weights of a RINN can be interpreted as edges in a causal graph, with all edges oriented in the direction that is towards $\boldsymbol{y}$, i.e., $x \rightarrow h_1 \rightarrow h_2 \rightarrow, ..., h_n \rightarrow y$, where n is the number of hidden layers. We suspect that reaching the global optimum is not necessarily required, and correct causal structure (but not necessarily correct causal parameters) could still be recovered from models close to the global optimum.

4.7.1 Causal Assumption Examples

Given that x is found to be correlated with y by a neural network (i.e., there is a nonzero weight between x and y), there are four possibilities for the true orientation of the edge between x and y according to G_T:

$$(1) \quad x \rightarrow y$$
$$(2) \quad x \leftarrow y$$
$$(3) \quad x \leftrightarrow y$$
$$(4) \quad x \perp\!\!\!\perp y$$

(2) violates our prior knowledge that x causes y, i.e., our known causal ordering. (3) represents a latent confounding variable causing x and y and therefore violates the causal sufficiency

assumption. (4) violates the faithfulness assumption. If $x \perp\!\!\!\perp y$ in G_T then a neural network with L_1 regularization and adhering to faithfulness should not learn an edge where there is no dependency. This leaves (1) as the true orientation of the edge between x and y.

The following causal graphs would not be valid interpretations of the weights of a neural network given the assumptions in this section:

$$(1) \quad x \leftarrow h_1 \leftarrow h_2 \rightarrow h_3 \rightarrow y$$

$$(2) \quad x \rightarrow h_1 \rightarrow h_2 \leftarrow h_3 \rightarrow y$$

(1) violates causal sufficiency. (2) violates faithfulness—only h_3 is necessary to predict y. If this relationship, $h_3 \rightarrow y$, were actually present in the ground truth, the RINN would learn it as a nonzero bias node.

4.8 Ranking Models Based on Balance Between Sparsity and Prediction Error

The major goal of this work was to learn neural networks that capture causal relationships within their weight matrices. In order to achieve this goal, we needed to modify our approach to model selection in order to best balance sparsity of weights and prediction error. After training (RINN, DNN, DBN, and ES-C) with various combinations of hyperparameters, we selected the models (sets of weight matrices for each trained network) that were relatively sparse and had a relatively low prediction error. This meant we needed to find models that achieved a good balance between sparsity and prediction error. We often observed prediction error decreasing as sparsity decreased and attribute this to the model having more flexibility (i.e., less sparsity) in how it learns to fit the function. This can lead to overfitting the training set if a hold-out test set is not also evaluated. Increasing regularization constrains the model, and over-regularizing leads to underfitting the data. To balance the trade-off between prediction error and sparsity, in plots of prediction error versus sparsity (Figure 7), we measured the Euclidean distance from each point (model) to the origin. In more detail, we plotted the sum of all the weights in a model (our chosen measure of sparsity as we found the regularization rate to be much more variable—i.e., the regularization rate merely encourages

sparsity, but does not enforce it) on the x-axis and the prediction error on the y-axis. We then measured the Euclidean distance from the origin to each point on the plot just described according to

$$d_x = \sqrt{\left(\sum_{i=1}^{m}\sum_{j=1}^{r_i}\sum_{k=1}^{c_i}|w_{j,k}^{(i)}|\right)^2 + L^2} \tag{4.21}$$

where L is the validation set loss for neural network x, m is the number of weight matrices in neural network x, r_i and c_i are the number of rows and columns in matrix i respectively, and w is a scalar weight. Prior to calculating the Euclidean distance (d_x), we removed large outliers by selecting values for the sum of weights and prediction error that were below the 95% quantile. We then performed min-max scaling (using sklearn's MinMaxScaler [Pedregosa et al., 2011]) on the prediction error and sum of weights. Min-max scaling scales each set of values to within the interval $[0, 1]$. Models were then ranked according to shortest Euclidean distance (smallest d_x) and the ten models for each strategy with the smallest distance were selected to be used to calculate average precision and recall. After selecting the ten best models for each strategy and dataset combination, we retrained on the training plus validation set, and tested on the hold-out test set.

4.9 Visualizing the Weights of a Neural Network

To increase our understanding of what the weights of a neural network capture after training with L_1 regularization, we developed a simple visualization technique. This technique maps every hidden node (or input node) to a representation in output space, and gives us a measure of how each node in the network changes the output values. To get the representations in output space for a layer of nodes, we matrix multiply the weight matrices from that layer going forward to output space. The representation in output space is then resized and displayed as a heatmap. This allows us to quickly assess what the weights of a neural network have captured. By carefully constructing a ground truth causal DAG (see Figure 5a) with easily identifiable patterns in the weights when mapped to output space, we created ground truth heatmaps (Figure 5e) that will be present in the weights of a neural network, if the

weights captured the causal relationships in the ground truth DAG. We simply look at the heatmaps calculated from a trained neural network to see if the weights captured any of the known causal relationships.

This visualization algorithm intuitively visualizes the weights of a trained RINN, DNN, or DBN. To create these visualizations, we need the learned weight matrices from a trained neural network. Given these weight matrices $(\boldsymbol{W}_1, \boldsymbol{W}_2, ..., \boldsymbol{W}_n)$, we start with the first weight matrix, $\boldsymbol{W}_1$ (dimensions: $i \times j$), and matrix multiply by $\boldsymbol{W}_2$, then matrix multiply by $\boldsymbol{W}_3$, and continue until we multiply by the final weight matrix, $\boldsymbol{W}_n$ (dimensions: $r \times s$). The result of this calculation is an $i \times s$ (number of first hidden layer nodes by number of output DEGs) matrix that represents the change induced by each vertex/hidden node in the first hidden layer on output space ($\boldsymbol{y}$ space), or the impact of the activation of a hidden node on the final vector of output values. Just as we performed this calculation starting with $\boldsymbol{W}_1$, we next perform the calculation starting with $\boldsymbol{W}_2$ and get a final matrix representing the change induced by each of the second hidden layer nodes on output space. We iterate through all weight matrices until we have matrix representations of output space for all nodes in a trained neural network. This algorithm is equivalent to neural network forward propagation (starting from each hidden layer individually) without using activation functions and biases. Using the activations and biases would not allow us to isolate the effects of the weights alone.

Following the calculation of the visualization matrices as above, each row of these matrices were resized to be 2-dimensional. For example, if a visualization matrix had dimensions 8×16, meaning each row was a 16-dimensional vector, we could represent each row as a 4×4 matrix simply by stacking the values, four at a time, of a single 16-D vector.

The visualizations were calculated according to

$$\boldsymbol{M}_j = \boldsymbol{W}_j \boldsymbol{W}_{j+1} \boldsymbol{W}_{j+2}, ..., \boldsymbol{W}_h \tag{4.22}$$

$$\boldsymbol{v}_j = reshape(\boldsymbol{M}_j) \tag{4.23}$$

where $\boldsymbol{M}_j$ is an $m \times n$ matrix for layer j, m is the number of nodes in layer j, n is the number of nodes in the output $\boldsymbol{y}$, $\boldsymbol{W}_j$ is the weight matrix between layer j and $j+1$, and $\boldsymbol{W}_h$ is the last weight matrix. Each row in $\boldsymbol{M}_j$ represents what a single node in layer j maps to in

output space. *reshape* means to reshape each row of $\boldsymbol{M}_j$ to a 4×4 matrix (assuming $n = 16$) (Figure 5b). Therefore, $\boldsymbol{v}_j$ is a list of 4×4 matrices (visualizations/heatmaps) for all hidden nodes in layer j. Next, the magnitude of each value in the 4×4 matrices is interpreted as a pixel intensity (or a heatmap intensity), and displayed as a given intensity of color (Figure 5d). In this way, we generated 2-D heatmaps for each hidden and input node in a trained neural network. This visualization represents the weights connecting a node to output space, or the influence of the activation of that node on the output values (Figure 5e). *The heatmap visualizations of all nodes' effects in output space is a representation of the causal structure learned by a neural network.*

5.0 Experiments and Results Using Simulated Data

5.1 Summary

In a series of simulation experiments, we show that the RINN model successfully recovered latent causal structure and relationships between input variables from various simulated datasets with different levels of noise better than other models. We compare the RINN with a conventional feedforward deep neural network (DNN), a deep belief network (DBN), a RINN optimized with a constrained evolutionary strategy, and the DM algorithm (a causal discovery algorithm). In summary, the results here show that RINN is a partially interpretable deep learning model that enhances the capability of learning complex relationships between input and output variables, and reveals causal relationships among variables.

5.2 Model Selection

Given the input and output simulated data as described in Section 4.3, we used various algorithms (see Sections 4.5, 4.6) to attempt to recover the ground truth causal structure in Figure 5a. Each simulated dataset was separated into training, validation, and test sets according to the following ratio 0.75/0.15/0.10, respectively. These ratios mean that 90% of the data was used for model selection (training and validation) and 10% was reserved as a final hold-out test set. We wanted to avoid the overfitting that often occurs if using a single validation dataset, so using the 90% split described above, we randomly selected 15% for one validation set (while training on the 75% remaining data) and a different 15% for the other validation set (while again training on the remaining 75%). We then averaged over the results for these two validation datasets to decide which models performed the best.

We used a combined random and grid search approach to select the hyperparameters (learning rate, regularization rate, size of hidden layer, etc.) for the best models [Bengio, 2012, Hinton, 2012] with the main objective of finding the optimal balance between sparsity

and prediction error to encourage the discovery of latent causal structure. See Table 1 for a breakdown of the number of sets of hyperparameters that were evaluated for each deep learning strategy.

A major goal of this work was to identify trained neural networks that had a high probability of capturing causal relationships within their weight matrices. In order to achieve this goal, we needed to modify our approach to model selection in order to best balance sparsity of weights and prediction error, in addition to performing model selection on deep learning strategies with different numbers of hyperparameters. A hyperparameter is any parameter outside the weights of a deep learning strategy that has an effect on the prediction error and needs to be "tuned" during training. Some examples are: learning rate, L_1 regularization rate, number of hidden layers, number of node in each hidden layer, batch size for updating weights, number of training iterations, optimizer, activation function, and how to initialize the weights.

Our search through hyperparameter space varied somewhat depending on the data and the deep learning strategy being evaluated. Each set of hyperparameters in Table 1 was evaluated on two different validation datasets, and the results were averaged across these two datasets. Datasets 5 and 6 were considerably larger than the other simulated datasets as they contained a large amount of real TCGA data. This increased the number of hyperparameters to be evaluated (i.e., more hidden layer sizes to search through because the input was larger), so we performed model selection on 100 more sets of hyperparameters for these larger datasets. Also, DNNs, DBNs, and ES-C have additional hyperparameters to search through relative to the RINN. For DNNs and DBNs, we need to find the optimal number of hidden layers (whereas the structure of the RINN allows one to largely avoid determining the ideal number of hidden layers—Section 4.4). For simplicity, we assumed all DNN hidden layers to be of the same size. However, DBN model selection uses the decreasing nature of the hidden layers as a driving force to learn the most salient aspects of the data. Therefore, DBN model selection included searching over different sizes of the hidden layers for each of the hidden layers. This means more hyperparameters to search through. ES-C also had additional hyperparameters, including elite ratio, mutation rate, and set of legal weight values.

For binary outputs, binary cross-entropy plus L_1 regularization was used as the objective

Table 1: Number of Sets of Hyperparameters Evaluated for each Deep Learning Strategy and Simulated Dataset

Strategy	Datasets 1,2,3,4 (non-TCGA)	Datasets 5,6 (TCGA)
RINN	440	540
DNN	732	832
DBN	940	1040
ES-C	500	NA

function. For non-binary outputs, mean squared error (MSE) plus L_1 regularization was used. Almost all of the best performing models used ReLU (Rectified Linear Unit) activation functions, although we performed model selection using sigmoid and tanh activation functions as well.

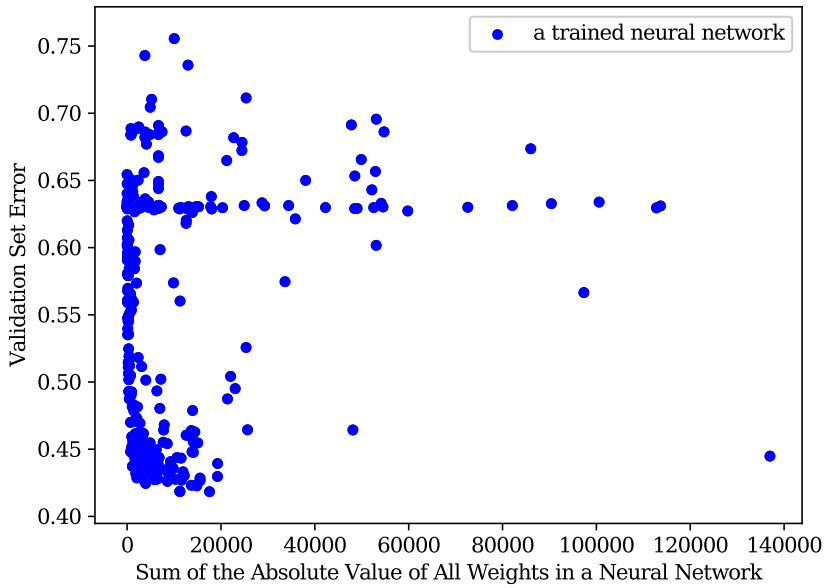
Figure 7 shows model selection results for RINNs on simulated Dataset 5. These are typical model selection results across all deep learning strategies and datasets. Figure 7a shows the validation set error and the sum of the absolute value of all weights in a network for 1,080 (540 times two validation sets) fully trained RINNs. There is a concentration of networks (shown as blue circles) in the lower left corner of this figure. It is within these models that we hypothesize one will find the models with the highest probability of capturing the causal structure of the data within their weights. Figure 7b shows a magnification of the lower left of Figure 7a, and here it becomes apparent that model selection based solely on prediction error or sparsity (sum of the absolute values of the weights) is inadequate. We also don't know, 1. Will deep learning strategies capture causal relationships in their weights? and 2. If they do capture this structure in their weights, how can we identify them during model selection (i.e., Where would these models be located in the plots in Figure 7?)? We hypothesize that the models with best chances of capturing causal relationships in their weights will be the models that optimally balance both prediction error and sparsity (i.e., the models in bottom left of Figure 7a). To quantify this hypothesis and provide a ranking

of the most optimal models, we measured the distance between each model's point location in the plot in Figure 7b and the $(0, 0)$ origin, and used this as a metric to rank the models (Section 4.8 for more details). We hypothesize that the models closer to the origin (using the distance metric described above) would better capture the ground truth causal structure in their weights.

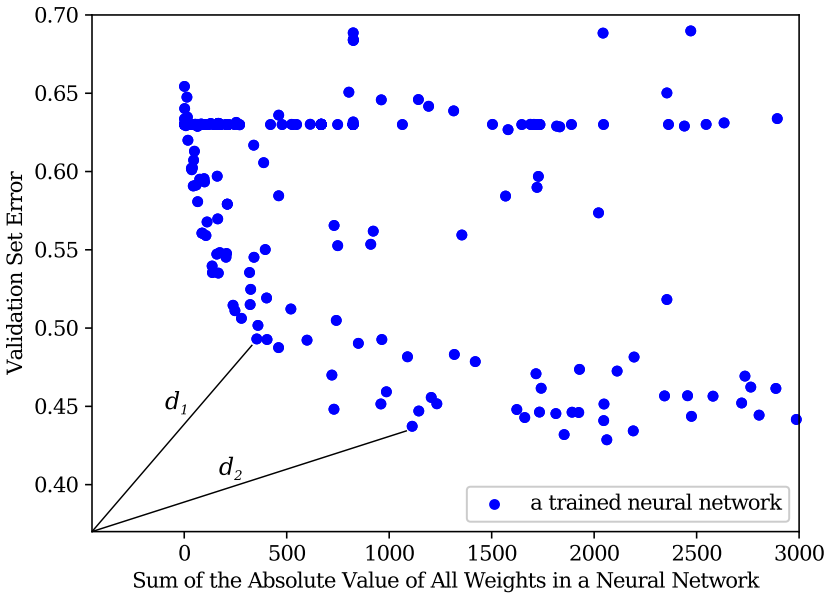
5.3 Visualizing the Weights of a Neural Network

In addition to evaluating the different deep learning strategies and distance metric (d_x) for finding causal structure, we also needed a method to quickly evaluate many networks without relying on visual inspection. The solution to this problem came from our method to visualize the weights of a neural network. Figure 8 shows visualizations of the weights for two different DNNs (Section 4.9). Each square in these visualizations represents a specific node in a neural network, and the change in the output values induced by the activation of the node represented by the square. These visualizations are calculated by iteratively matrix multiplying a vector of hidden values (or input values) by the next weight matrix, until there are no weight matrices left going forward from a layer of nodes to the output values. The 16 dimensional output is then visualized as a heatmap (where different values are represented as different colors). Figure 8a shows the ground truth patterns/visualizations based on the ground truth DAG (Figure 5a). These are the patterns we are looking for in the weights of a network to indicate the correct causal structure. The color of these heatmaps is of minimal importance, while the pattern in the heatmaps is of maximal importance.

Figure 8b shows a visualization of the weights of a 3-hidden layer DNN trained on Dataset 1. The visualizations labeled as $\boldsymbol{x}$ show what each of the input nodes for this DNN map to in output space. If we compare this vector of images to the ground truth input node patterns in Figure 8a (bottom), we see that the input nodes 100% match with the ground truth input node patterns. This example immediately suggests a possible method for evaluating how well neural networks capture causal structure—by calculating precision and recall based on the ground truth patterns that should be present within these visualizations for the causal

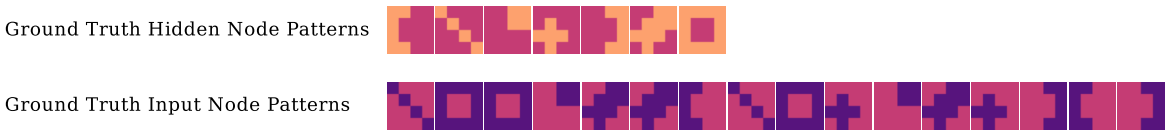


(a) Example Model Selection Results (RINN Trained on TCGA+OR Dataset). This figure represents 1080 trained neural networks.

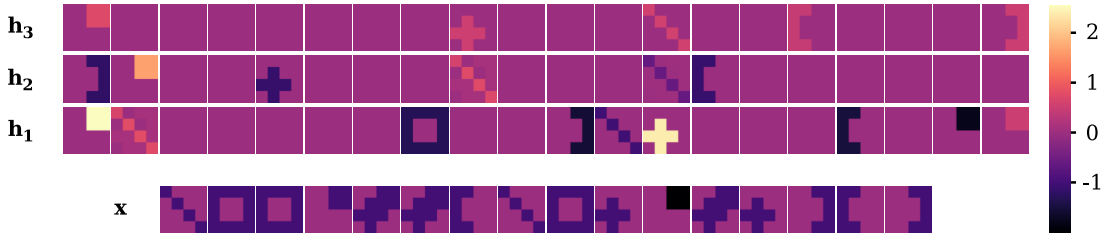


(b) Lower left of (a) magnified. Lines d_1 and d_2 represent distance from origin to a trained neural network. In practice, the distance is calculated using the point $(0, 0)$ and axes scaled to the interval $[0, 1]$ (neither of which are depicted here).

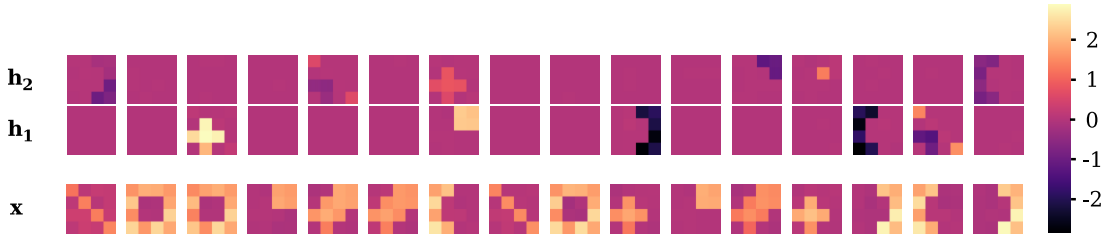
Figure 7: Measuring Euclidean Distance to Rank Model Selection Results



(a) Ground truth patterns used when calculating precision and recall.



(b) A DNN (trained on Dataset 1) that finds all input patterns and finds all hidden node patterns except one (i.e., one false negative).



(c) A different DNN (trained on Dataset 3) that finds all input patterns, but has some difficulty with the hidden patterns (3 false positives*, 4 true positives, and 3 false negatives). *False positives are nodes that don't match any ground truth ± 1 pixel and repeated false positives are only counted once.

Figure 8: Visualizing the Weights of DNNs and Calculating Precision and Recall

($\boldsymbol{x}$ represents the input layer, $\boldsymbol{h_j}$ represents hidden layer j).

structure to be correct.

$$P = \frac{TP}{TP + FP} \tag{5.1}$$

$$R = \frac{TP}{TP + FN} \tag{5.2}$$

Here we show how we measured precision and recall for known hidden node and input node patterns in the weight matrices of a trained neural network. To calculate precision and recall of the causal structure, we ran the visualization algorithm (Section 4.9) on the first input layer and each hidden layer in our trained RINN, DNN, DBN, or ES-C (referred to as a trained network from here on), obtaining a vector for each node representing the influence of that node on output space. We took the absolute value and then binarized these vectors in order to match them to the ground truth, thus being able to calculate precision and recall. This required setting a threshold, above which a value would be set to 1 and otherwise set to 0. We chose this threshold by searching through our best models for a single threshold that provided the best F_1 score. The same threshold was used for all models. We obtained precision and recall measurements for each hidden node by comparing this binary vector to a known binary ground truth vector. Because the data used in these experiments were simulated, we can easily calculate these ground truth vectors from the ground truth DAG (see Figures 5e, 8a).

To calculate the ground truth, we used the same algorithm as in Section 4.9 applied to the ground truth DAG and ground truth weight matrices. By matrix multiplying each of the ground truth weight matrices all the way to output space, we obtained a ground truth vector representing the known change in output values induced by each ground truth hidden node. Next, we compared these ground truth vectors to the predicted vectors obtained from the visualization algorithm applied to the weights of a trained network, and were able to calculate precision and recall for all ground truth hidden nodes and all ground truth input nodes.

True positives (TP) here represent ground truth hidden nodes or input nodes (remember, nodes are represented as vectors in output space) that were captured by the weights of the trained network. False positives (FP) represent predicted vectors from the trained network that did not match any of the ground truth nodes ± 1 or the zero vector ± 1. False negatives

(FN) represent ground truth hidden nodes or input nodes that were not captured by the weights of the trained network. When comparing a predicted output space vector to a ground truth vector, we considered the predicted to match the ground truth if the binary vectors were identical within ±1 value (i.e., pixel). Precision and recall for the TCGA+OR and TCGA+OR, XOR, AND datasets were calculated based only on the recovery of the simulated data ground truth—as we don't know the ground truth for the TCGA part of the data and therefore have no way to calculate precision and recall. The average test set error and AUC however was calculated based on all output values, not just the simulated data output values. The network in Figure 8b achieved 100% *input node* precision and recall.

After examining the vector of images labeled as $\boldsymbol{h}_1$, in Figure 8b, we see that this hidden layer by itself captures six of the seven ground truth hidden node patterns from Figure 8a ($\boldsymbol{h_2}$ and $\boldsymbol{h_3}$ do not find any additional ground truth hidden node patterns). The calculations for *hidden node* precision and recall for this DNN would be as follow:

$$Precision = \frac{TP}{TP + FP} = \frac{6}{6 + 0} = 1.00 \tag{5.3}$$

$$Recall = \frac{TP}{TP + FN} = \frac{6}{6 + 1} = 0.86 \tag{5.4}$$

So despite regularization encouraging the DNN to represent patterns as simply as possible, this DNN did not capture the ground truth hidden node that is a combination of the "plus sign" and "little square" (second from the right in Figure 8a top). Nodes 17 and 19 in Figure 5a and 5e are referred to as "combination" nodes.

The first hidden layer of a DNN must capture all the hidden node patterns required to completely map inputs to outputs, as it does not have another chance to capture information it may have missed in the first hidden layer later in the network. The hidden layer representations of a DNN are alternate representations of all the information in the input necessary to calculate the output. This is in contrast to a RINN, which can simply learn the patterns at a later hidden layer because a RINN has redundant inputs. So the hidden layers of a RINN are not necessarily alternate representations of the input, but they capture salient aspects of the input data in some way desirable to calculate the output.

Figure 8c shows a visualization of a different DNN (2-hidden layers). This network did not capture the causal structure as nicely as Figure 8b. The input nodes map correctly to the ground truth input node patterns in Figure 8a. However, there are at least three false positives (depending on threshold chosen) and only four true positives found within the hidden layer nodes. The calculations for hidden node precision and recall would be as follow:

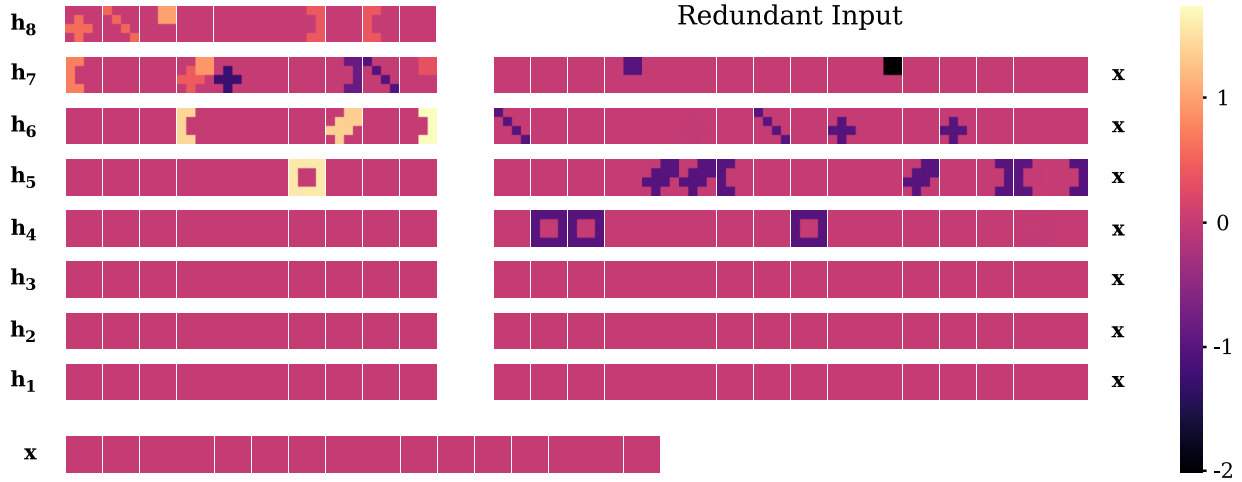
$$Precision = \frac{TP}{TP + FP} = \frac{4}{4+3} = 0.57 \tag{5.5}$$

$$Recall = \frac{TP}{TP + FN} = \frac{4}{4+3} = 0.57 \tag{5.6}$$

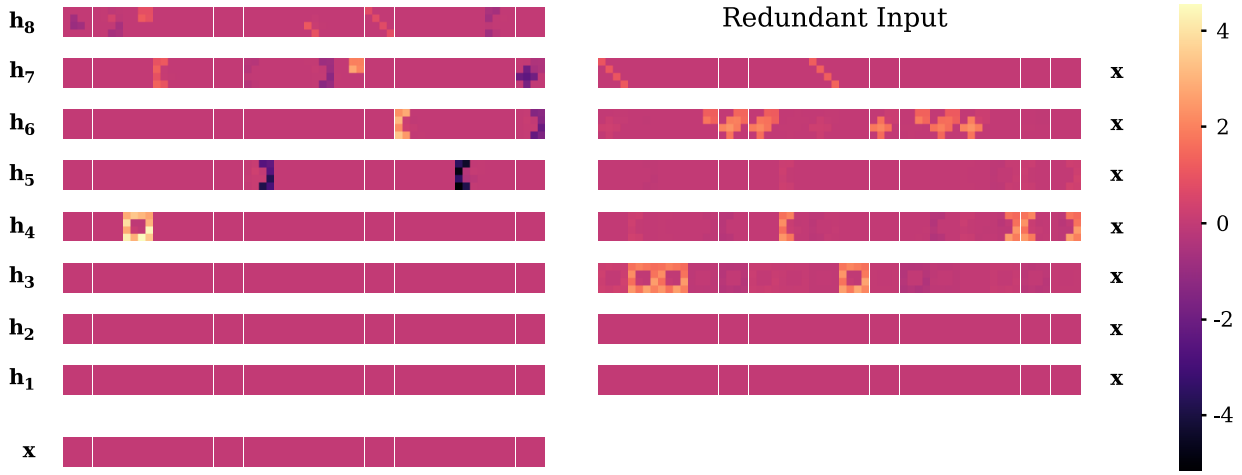
Figure 9 shows the visualizations for all weight matrices in two different trained RINNs. The RINN visualized in Figure 9a is an example of a network that scores 100% precision and recall for both input and hidden nodes. To evaluate the input nodes of a RINN, all input node heatmaps for a particular node are summed together, and this final sum of heatmaps is compared to the known ground truth for that input. We can see in Figure 9a that there is no need to sum the heatmaps because the input nodes learned the correct mapping (Figure 8a (bottom)) at only a single redundant input layer (for each input node). When we look at the heatmaps for the hidden nodes, we see that all seven ground truth patterns from Figure 8a (top) are present in either $\boldsymbol{h}_5$, $\boldsymbol{h}_6$, $\boldsymbol{h}_7$, or $\boldsymbol{h}_8$. Figure 9b shows the visualization of a different RINN that doesn't score as well as Figure 9a. The input nodes of this RINN achieve 100% precision and recall. But the hidden nodes for this RINN show four false positives, six true positives (although one of the combination nodes, i.e., nodes 17 and 19 in Figure 5a and 5e, is very close to missing the threshold), and one false negative ($7 - 6 = 1$). The final precision and recall for the hidden nodes would be:

$$Precision = \frac{6}{6+4} = 0.60 \tag{5.7}$$

$$Recall = \frac{6}{6+1} = 0.86 \tag{5.8}$$



(a) A RINN (trained on Dataset 1) that finds all input and hidden patterns (i.e., 100% precision and recall).



(b) A different RINN (trained on Dataset 3) that finds all input patterns, but has some difficulty with the hidden patterns (3 or 4 false positives*, 6 true positives, and 1 false negative). *All matching of patterns depends on setting a threshold, and anything above this threshold counts as a nonzero value in the heatmap.

Figure 9: Visualizing the Weights of RINNs and Calculating Precision and Recall

($\boldsymbol{x}$ represents the input layer, $\boldsymbol{h_j}$ represents hidden layer j).

5.4 Precision and Recall of Causal Structure

The major purpose of this work was to compare the ability of different neural network-based strategies to find causal structure in their weights. We were also interested in the limitations of these strategies depending on the characteristics of the data. For example, we were particularly interested in how well the various algorithms would perform on very noisy data. Table 2 shows the *input* node precision and recall results across all datasets and strategies used in this chapter. Table 2 also shows the average test set error and AUROC (where appropriate) for all strategies and datasets. This table is best thought of as showing how well the input mapped to output for all strategies and datasets without indicating the exact causal structure in the weights, i.e., this metric does not indicate how the latent variables interacted to achieve the output. For example, a high precision and recall for input nodes is interpreted as the network correctly mapping each of the inputs to the correct outputs. The precision and recall (as well as error and AUROC) were averaged across the ten best models (as ranked by our distance metric) for each strategy and dataset for Tables 2 and 3.

Table 2 shows that both DNN and RINN performed quite well at mapping input to output across all datasets, with the RINN performing a little better in terms of precision and recall. The RINN performed noticeably better than the DNN on the Linear Gaussian and TCGA+OR datasets. The test errors and AUROCs for DNN and RINN were almost identical. This means that if one captures more causal structure than the other, it isn't due to one strategy being able to predict better than the other, but rather due to some other difference. ES-C performed similar to DNN and RINN on the first three datasets, but markedly worse on the OR, XOR, AND dataset in terms of precision and recall. ES-C had errors and AUROCs that were similar to DNN and RINN (and on dataset 4 were better than RINN and DNN). This is a promising result for future use of parallelized and possibly more advanced evolutionary algorithms in this setting. The DM algorithm, despite getting 100% recall, performed very poorly on all datasets. This is because the DM algorithm isn't identifying any false-negatives, just a lot of false-positives and maybe 1 or 2 true-positives. What is happening is that the DM algorithm correctly finds a couple true-positives with the remaining

Table 2: Average Input Node Precision and Recall Across Ten Best Models

The best values when the five strategies are compared on a specific dataset are bolded.

		Input Mapping			Test Set	
Simulated Data	Strategy	precision	recall	F_1	error	AUROC
(1) Matrix mult. w/ interv.	DBN	NA	NA	NA	0.05 ± 0.14	NA
	DNN	$\mathbf{1.00} \pm 0.00$	$\mathbf{1.00} \pm 0.00$	$\mathbf{1.00} \pm 0.00$	0.01 ± 0.00	NA
	RINN	$\mathbf{1.00} \pm 0.00$	$\mathbf{1.00} \pm 0.00$	$\mathbf{1.00} \pm 0.00$	$\mathbf{0.00} \pm 0.00$	NA
	ES-C	0.95 ± 0.07	$\mathbf{1.00} \pm 0.00$	0.97 ± 0.04	0.03 ± 0.01	NA
	DM	0.31	**1.00**	0.48	NA	NA
(2) Linear Gaussian	DBN	NA	NA	NA	$\mathbf{1.82} \pm 0.13$	NA
	DNN	0.94 ± 0.08	0.93 ± 0.02	0.93 ± 0.05	9.31 ± 0.19	NA
	RINN	$\mathbf{0.98} \pm 0.03$	$\mathbf{1.00} \pm 0.00$	$\mathbf{0.99} \pm 0.02$	9.19 ± 0.05	NA
	ES-C	0.91 ± 0.11	0.91 ± 0.03	0.90 ± 0.05	10.12 ± 0.16	NA
	DM	0.69	**1.00**	0.81	NA	NA
(3) OR	DBN	NA	NA	NA	$\mathbf{0.55} \pm 0.16$	0.68 ± 0.21
	DNN	$\mathbf{0.99} \pm 0.02$	0.98 ± 0.05	$\mathbf{0.98} \pm 0.03$	0.56 ± 0.00	$\mathbf{0.75} \pm 0.00$
	RINN	0.96 ± 0.06	$\mathbf{1.00} \pm 0.00$	$\mathbf{0.98} \pm 0.03$	0.56 ± 0.00	$\mathbf{0.75} \pm 0.00$
	ES-C	0.81 ± 0.16	0.95 ± 0.09	0.87 ± 0.12	0.57 ± 0.00	0.74 ± 0.00
	DM	0.56	**1.00**	0.72	NA	NA
(4) OR, XOR, AND	DBN	NA	NA	NA	0.58 ± 0.13	0.57 ± 0.14
	DNN	0.78 ± 0.15	0.99 ± 0.03	0.87 ± 0.11	0.45 ± 0.00	0.84 ± 0.00
	RINN	$\mathbf{0.92} \pm 0.08$	0.91 ± 0.13	$\mathbf{0.91} \pm 0.07$	0.45 ± 0.00	0.84 ± 0.00
	ES-C	0.46 ± 0.24	0.65 ± 0.17	0.53 ± 0.22	$\mathbf{0.41} \pm 0.00$	$\mathbf{0.87} \pm 0.00$
	DM	0.44	**1.00**	0.61	NA	NA
(5) TCGA+OR	DBN	NA	NA	NA	$\mathbf{0.34} \pm 0.16$	$\mathbf{0.88} \pm 0.11$
	DNN	0.90 ± 0.13	0.86 ± 0.15	0.88 ± 0.13	0.42 ± 0.01	0.86 ± 0.00
	RINN	$\mathbf{0.95} \pm 0.08$	$\mathbf{0.98} \pm 0.04$	$\mathbf{0.96} \pm 0.05$	0.40 ± 0.00	0.87 ± 0.00
(6) TCGA+OR, XOR, AND	DBN	NA	NA	NA	0.53 ± 0.17	0.64 ± 0.20
	DNN	$\mathbf{0.79} \pm 0.29$	0.90 ± 0.32	0.84 ± 0.30	$\mathbf{0.42} \pm 0.01$	$\mathbf{0.86} \pm 0.01$
	RINN	0.78 ± 0.05	$\mathbf{0.98} \pm 0.04$	$\mathbf{0.87} \pm 0.04$	0.43 ± 0.00	$\mathbf{0.86} \pm 0.00$

input nodes mapping to false-positives, meaning that it recorded zero false-negatives (with zero false-negatives, having any true-positives yields 100% recall). The DBN doesn't use $\boldsymbol{x}$ at all, so input precision and recall could not be calculated for DBN. Also, the DBN is an unsupervised algorithm while the DNN and RINN are supervised, so comparing their average errors and AUCs doesn't seem fair, but the DBN errors and AUCs were included here for completeness.

The results in Table 3 are the main results of the work on simulated data and show how well the ground truth causal structure was captured by each of the strategies for each dataset. Again, the *hidden* node precision and recall values shown in Table 3 are the average over the ten best models for each dataset and strategy. The RINN had a higher F_1 score than all other strategies for most datasets, and consistently had the highest recall across all datasets and strategies. RINN and DNN had F_1 scores within 0.01 of each other on datasets 4 and 6. The RINN performed considerably better than any other strategy on the Linear Gaussian dataset. The Linear Gaussian dataset was also the dataset with the most noise, so it is encouraging that the RINN was able to considerably outperform all other strategies on this noisy dataset, as we suspect that most real world biological data has a high amount of noise. There didn't seem to be much of a difference between continuous and binary datasets—the strategies were still able to learn causal structure from completely binary inputs and outputs (Datasets 3 and 4). Rather, the most important characteristic of the datasets seemed to be the amount of noise. The RINN outperformed all other strategies on the TCGA+OR dataset and performed similarity to a DNN on the TCGA+OR, XOR, AND dataset. These datasets had inputs and outputs of much higher dimensionality than any of the other simulated datasets, and shows the RINNs utility even in a higher dimensionality setting.

ES-C performed moderately well, but was outperformed by DNN and RINN on all datasets except Linear Gaussian, where ES-C outperformed DNN. This, again, suggests that using a parallelized evolutionary algorithm is a direction worth pursuing for finding latent structure, as ES-C performed well considering that we performed somewhat limited model selection for ES-C as compared to the other neural network-based strategies—ES-C has more hyperparameters to be tuned than RINN due to the evolutionary strategies optimization. ES-C was also trained on significantly less data (20%) than the other strategies to decrease

the runtime of ES-C even further. ES-C (unparallelized) wasn't run on datasets 5 and 6 because of the very long running time it had on the small datasets, meaning that running on the much larger datasets would be impractical. The DBN performed the worst of the deep learning algorithms on recovering the ground truth hidden node patterns, but performed better than the DM algorithm (causal discovery) on almost all datasets and performed decently on Dataset 1. This is an interesting and encouraging result for the DBN considering that it is an unsupervised algorithm and was still able to learn some of the ground truth structure. The DM algorithm performed quite poorly across all datasets, and the worst on most datasets (note that some of the DM algorithms assumptions are violated by G_T). The DM algorithm wasn't run on datasets 5 and 6 because of its poor performance on the smaller datasets. Overall, the RINN was better able to recover the causal structure in Figure 5a than any other strategy explored in this work. This means that it is the best choice going forward, among the strategies considered here, when using neural networks to recover latent causal structure.

5.5 Discussion

The results presented here show that both a RINN and DNN (with L_1 regularized loss function) can recover latent causal structure between inputs and outputs when using various simulated datasets. Importantly, the RINN performed better than the DNN (and all other algorithms) across almost all metrics measured for recovery of the ground truth hidden nodes (Table 3), which is the task that is most relevant when attempting to recover latent causal structure and biological cellular signaling pathways. Also, the RINN required less model selection, performed better on noisy data, and its architecture allowed inputs to directly act on hidden nodes (which simplified the interpretation of the underlying causal structure). When it comes to pure prediction ability both the DNN and RINN perform similarly with no real difference between their final test set average error values (Table 2). This suggests that the reason the RINN performed better than the DNN at capturing the hidden node causal structure is because of the differences between their architectures (i.e., the RINN has

Table 3: Average Hidden Node Precision and Recall Across Ten Best Models

Do hidden nodes capture known causal structure in their weights when mapped to output space? The best values when the five strategies are compared on a specific dataset are bolded.

		Hidden Mapping		
Simulated Data	Strategy	precision	recall	F_1
(1) Matrix mult. w/ interv.	DBN	0.73 ± 0.32	0.61 ± 0.22	0.66 ± 0.26
	DNN	0.97 ± 0.07	0.80 ± 0.07	0.87 ± 0.06
	RINN	$\mathbf{1.00} \pm 0.00$	$\mathbf{0.93} \pm 0.08$	$\mathbf{0.96} \pm 0.04$
	ES-C	0.72 ± 0.08	0.80 ± 0.13	0.76 ± 0.08
	DM	0.33	0.29	0.31
(2) Linear Gaussian	DBN	0.35 ± 0.13	0.57 ± 0.17	0.43 ± 0.14
	DNN	0.34 ± 0.30	0.43 ± 0.24	0.37 ± 0.26
	RINN	$\mathbf{0.65} \pm 0.11$	$\mathbf{0.87} \pm 0.18$	$\mathbf{0.74} \pm 0.13$
	ES-C	0.41 ± 0.10	0.54 ± 0.06	0.46 ± 0.09
	DM	0.33	0.29	0.31
(3) OR	DBN	0.20 ± 0.30	0.16 ± 0.23	0.17 ± 0.25
	DNN	$\mathbf{0.88} \pm 0.17$	0.76 ± 0.15	0.81 ± 0.14
	RINN	0.87 ± 0.19	$\mathbf{0.83} \pm 0.13$	$\mathbf{0.85} \pm 0.15$
	ES-C	0.71 ± 0.22	0.57 ± 0.18	0.63 ± 0.19
	DM	0.33	0.29	0.31
(4) OR, XOR, AND	DBN	0.38 ± 0.22	0.33 ± 0.19	0.33 ± 0.17
	DNN	$\mathbf{0.87} \pm 0.18$	0.79 ± 0.17	$\mathbf{0.82} \pm 0.16$
	RINN	0.81 ± 0.17	$\mathbf{0.83} \pm 0.11$	0.81 ± 0.11
	ES-C	0.44 ± 0.09	0.74 ± 0.16	0.55 ± 0.10
	DM	0.29	0.29	0.29
(5) TCGA+OR	DBN	0.08 ± 0.17	0.06 ± 0.14	0.08 ± 0.15
	DNN	0.29 ± 0.17	0.59 ± 0.26	0.33 ± 0.12
	RINN	$\mathbf{0.37} \pm 0.25$	$\mathbf{0.63} \pm 0.24$	$\mathbf{0.41} \pm 0.19$
(6) TCGA+OR, XOR, AND	DBN	0.04 ± 0.08	0.16 ± 0.22	0.07 ± 0.12
	DNN	$\mathbf{0.57} \pm 0.26$	0.74 ± 0.16	$\mathbf{0.61} \pm 0.20$
	RINN	0.53 ± 0.24	$\mathbf{0.80} \pm 0.10$	0.60 ± 0.17

redundant inputs). These are encouraging results for using the RINN in future experiments on genomic data to discover cellular signaling pathways, and suggest that neural network-based causal discovery algorithms may be helpful in discovering latent causal structure. The constrained evolutionary strategy explored here provided promising results despite limited model selection and training on a much smaller dataset—more sophisticated versions are worth exploring in future experiments. The DM algorithm was not able to capture much of the ground truth causal structure, but may perform well on the causal ground truth used here if its constraint of requiring each latent variable to be adjacent to at least one output variable could be relaxed, as this assumption was violated by our G_T.

It has not escaped our notice that the RINN architecture simplifies the interpretability of a deep learning model. The architecture of the RINN (i.e., inputs directly connected to every hidden layer), allows hidden nodes to be easily mapped to a set of input variables. This mapping is accomplished by representing a hidden node h_{42} by the set of input variables with the largest weight values connected to h_{42} (often requiring a weight threshold). In this way, a set of input variables can be used to represent a hidden node and the hidden nodes become partially interpretable, assuming that the functions or identities of a set input variables suggests a hypothesis for the identity of the hidden variable. We did not interpret hidden nodes based on the connected input variables in this chapter using simulated data, as the input variables don't have any real meaning. However, in the next chapter we train the RINN with human cancer data and map the hidden nodes to sets of input variables with biological meaning.

In this work, we developed a neural network-based strategy (RINN) and causal framework for latent causal discovery that allowed for the interpretation of the weights in a RINN as causal relationships. We also created a robust pipeline for testing an algorithm's ability to capture latent causal structure and developed a variety of simulated data inspired by cellular signaling pathways. In addition, we created a causal ground truth DAG with simple patterns in the hidden nodes to enable quick visual verification of these patterns. Related to this, we developed a simple algorithm to visualize the weights of a small neural network, or part of a larger neural network, using the patterns that are known to be present in the simulated data. This visualization algorithm allowed us to quickly verify whether or not a

neural network-based algorithm was capturing any of the ground truth causal structure in its weights. This technique was extremely valuable for quickly prototyping algorithms and discovering modifications that encouraged a neural network to learn causal structure (e.g., this method allowed us to quickly determine that L_1 regularization was much better than L_2 at capturing causal structure). This visualization technique also inspired a method for validating the causal structure learned by any algorithm and especially neural network-based algorithms.

There are some important limitations and constraints relevant to this work. When interpreting the weights of a RINN (or DNN) as a causal graph, it is necessary to look at the values of all the weights in a trained network and, based on a threshold, decide which weights are significant (i.e., edges in a graph) and which are not. This is necessary because a layer in a neural network is fully connected to the next layer, meaning that each input/hidden node in a previous layer is connected to each hidden node in the next layer. Using L_1 regularization causes the network to set most of the values in the weight matrices to small numbers, but in our experiments very few of the weights are actually zero—rather there are many weights with values in the interval $[0.0001, 0.5]$. This requires that we set a threshold, above which weights will be interpreted as a causal edge (often this threshold was near 0.1), in order to generate a causal graph from the weight matrices. This thresholding problem can be solved by using a non-differentiable optimization procedure (e.g., evolutionary strategies) that limits the allowed values for weights in a neural network (e.g., restricting the weight values to 1 or 0 only).

Additionally, in order for the RINN to discover a direct causal connection between two hidden nodes or an input and a hidden node, the true causal relationship that is being modeled must be able to be approximated by $ReLU(wx + b)$, where all variables are scalar values (as the relationship between two hidden nodes in different layers can be isolated, i.e., without the influence of other hidden nodes). The results in this work on the simulated data support that the causal relationships in the simulated data can be adequately approximated with $ReLU(wx + b)$. This lends minimal support for the current version of the RINN being able to model causal relationships in biological data. If it turns out that direct biological causal relationships (i.e., the hidden nodes in a neural network, trained on genomic data,

all representing biological entities) cannot be approximated with $ReLU(wx + b)$ (or other activation functions), then we hypothesize that causal relationships would still be learned but actual biological entities may only be present in every other hidden layer—meaning that a hidden layer between biological entities was required to perform a more complex function between biological entities than just a $ReLU(wx + b)$. Another way to remedy this limitation (and a possible path for future work) would be to replace each weight in the RINN with a very small 1-hidden layer neural network, and then regularize the number of neural networks within the RINN that have nonzero weights. Using a small neural network for each edge would allow the RINN to learn a much more complicated function from hidden node to hidden node or input node to hidden node, and thus be able to model more complex direct causal relationships than the current RINN. An important and obvious limitation of this work is that it must be known a priori that one set of variables causes another (i.e., inputs cause outputs), which will limit the datasets where these methods are applicable.

Another limitation of this work is that calculating precision and recall did not take into account the ordering of the nodes, so it is possible for a combination node (e.g., "big square" ground truth node—node 17) to be learned in the last hidden layer of the network and still be counted as a true positive even though it doesn't branch into two brackets ("right bracket", "left bracket" heatmaps) before connecting to $\boldsymbol{y}$. In this case, the learned causal structure would be missing a latent vertex and not be completely correct. This is an example of how our method for calculating precision and recall is imperfect. However, after looking through hundreds of trained networks, we observed that this phenomenon rarely happens, and the vast majority of visualizations showed causal structure learned in the correct order. Using L_1 regularization also encourages the network to learn nodes in the correct order and hierarchy, as learning the ground truth causal structure represents the weight structure with the lowest number of edges.

The results of this chapter illustrate two main characteristics (advances) brought forward by the RINN model: the capability of learning a partially interpretable deep learning model and the capability of learning latent causal relationships in cellular signaling pathway-inspired simulated data. Considering the extreme rise in popularity of deep learning, using deep learning for causal discovery or interpreting the weights of a neural network in a causal

framework could be beneficial to both communities and lead to fruitful hybrid methods of causal discovery. Elaborating on the algorithms, especially more advanced and faster evolutionary algorithms, and causal framework developed in this dissertation could lead to improvements in the interpretability of deep neural networks and in the use of neural network-based algorithms for the discovery of latent causal structure in the future.

6.0 Using the RINN to Study Cancer Cell Signaling

6.1 Summary

Understanding cellular signaling pathways is essential to an in-depth understanding of cell biology under physiological and pathological conditions. In general, the cellular mechanisms leading to disease states in an individual are nuanced and not well understood. Reliably inferring all abnormal cellular signaling pathways from data (e.g., omics data) is a very difficult and unsolved task. Here we interpret the cellular signaling system as a causal graphical model and apply a supervised deep neural network-based algorithm to learn latent causal structure that represents the cancer cellular signaling system.

Most causal discovery algorithms have been developed to find causal structure and parameterizations of causal structure relative to the observed variables of a dataset. Only a small number of causal discovery algorithms also find latent causal structure, but these latent causal discovery methods are often highly constrained, suggesting that new methods are needed. In the previous two chapters, we developed a supervised deep neural network-based algorithm for latent causal structure discovery, called the redundant input neural network (RINN), that achieved high precision and recall of latent causal structure when trained on simulated data. The RINN allows input data to directly connect to latent variables within the deep learning hierarchy—to enhance the learning of causal relationships. For the RINN to recover latent causal structure, it must be used on data where it is known a priori that input variables cause output variables.

We hypothesize that training a RINN on multiple omics data will enable us to map the functional impacts of genomic alterations to latent variables in a deep learning model, and allow us further to discover the hierarchical causal relationships between variables perturbed by different genomic alterations. For the work in this chapter, we applied the RINN algorithm to cancer genomic data, where it is known that genomic alterations eventually cause changes in gene expression. We found that 5259 differentially expressed genes can be predicted from 372 somatic genome alterations with reasonable AUROCs by a RINN (or DNN). We also

found that a RINN trained with L_1 regularization was able to discover many real cancer signaling pathway relationships, especially relationships between genes in the PI3K, Nrf2, and TGFβ pathways, including some causal relationships. However, despite relatively large levels of regularization, the returned causal graphs were still somewhat too dense to be easily and directly interpretable as causal graphs. Future versions of the RINN, with differential regularization and other modifications, will have a good probability of capturing more easily interpretable cancer cellular signaling pathways.

6.2 Introduction

In general, the cellular mechanisms leading to cancer in an individual are heterogeneous, nuanced, and not well understood. It is well appreciated that cancer is a disease of aberrant signaling, and the state of a cancer cell can be described in terms of abnormally functioning cellular signaling pathways. Identifying all of the abnormal cellular signaling pathways causing a patient's cancer would enable more patient-specific and effective treatments—including targeting multiple abnormal pathways during a treatment regime. Aberrant signaling in cancer cells usually result from somatic genomic alterations (SGAs) that perturb the function of signaling proteins. Although large-scale cancer genomic data are available, such as those from The Cancer Genome Atlas (TCGA) and the International Cancer Genome Consortium (ICGC), it remains a very difficult and unsolved task to reliably infer how the SGAs in a cancer cell cause aberrations in cellular signaling pathways based on the genomics data of a tumor. One challenge is that the majority of the genomic alterations observed in a tumor are non-consequential (*passenger* genomic alterations) with respect to cancer development and only a few are *driver* genomic alterations, i.e., genomic alterations that are causing cancer. Furthermore, even if the driver genomic alterations of a tumor are known, it remains challenging to infer how aberrant signals of perturbed proteins affect the activation states of other signaling proteins, which are not measured (latent), in order to understand the disease mechanisms of an individual tumor and identify drug targets. This requires one to study the causal relationships among latent (i.e., hidden or unobserved) variables, such as large protein

complexes or abstractions of different biological processes within a cell, in addition to observed variables. Most causal discovery algorithms have been developed to find causal structure and parameterizations of causal structure relative to the *observed* variables of a dataset [Spirtes et al., 2000, Cooper, 1999, Heinze-Deml et al., 2018, Maathuis and Nandy, 2016, Peters et al., 2017, Lagani et al., 2016]. Only a small number of causal discovery algorithms also find latent causal structure, especially in high-dimensional data [Murray-Watters and Glymour, 2015, Frot et al., 2019].

Deep learning represents a group of machine learning strategies, based on neural networks, that learn a function mapping input to output. The signals of input variables are processed and transformed through many hidden layers of latent variables (i.e., hidden nodes) [Deng et al., 2014, Goodfellow et al., 2016, LeCun et al., 2015]. These hidden layers learn hierarchical or compositional statistical structure, meaning that different hidden layers capture structure of different degrees of complexity [Lee et al., 2008, Lee et al., 2011, Le et al., 2011]. Researchers have previously shown that deep learning models can represent the hierarchical organization of signaling molecules in a cell [Chen et al., 2015, Chen et al., 2016, Young et al., 2017, Lu et al., 2018, Tao et al., 2020], with latent variables as natural representations of unobserved activation states of signaling molecules (e.g., membrane receptors or transcription factors). In more detail, some recent work from our group used deep learning to simulate cellular signaling systems that were shared by human and rat cells [Chen et al., 2015] and to recover components of the yeast cellular signaling system, including transcription factors [Chen et al., 2016]. These studies utilized unsupervised learning methods, in contrast to the supervised methods used in this dissertation. Also, the previous studies from our group did not attempt to find causal relationships representing the cellular signaling system. In general, deep learning models have not been broadly used as a tool to infer causal relationships in the computational biology setting.

In the previous two chapters we developed a deep learning algorithm, named redundant input neural network (RINN), to learn causal relationships among latent variables from data inspired by cellular signaling pathways. RINN solves a problem where a set of input variables *cause* the change in another set of output variables, and this causal interaction is mediated by a set of an unknown number of latent variables. The constraint of inputs causing outputs

is necessary to interpret the latent structure as causal relationships (see Chapter 4.7 for more details regarding the causal assumptions of the RINN). A key innovation of the RINN model is that it is a partially transparent model allowing input variables to directly interact with all latent variables in its hierarchy, such that it can constrain an input variable to be connected to a set of latent variables that can sufficiently encode the impact of the input variable on the output variables. We showed that the RINN outperformed other algorithms, including neural network-based algorithms and a causal discovery algorithm known as DM (Detect MIMIC (Multiple Indicators Multiple Input Causes)), at identifying latent causal structure in various types of simulated data.

In this chapter, we take advantage of the partially transparent nature of the RINN model and used the model to learn a representation of the cancer cellular signaling system. In this setting, we interpret the cellular signaling system as a hierarchical causal model of interactions between the activation states of proteins or protein complexes within a cell. Based on the assumption that somatic genome alterations (SGAs) that drive the development of a cancer often influence gene expression, we train a RINN using tumor SGAs as input to predict cancer differentially expressed genes (DEGs) (outputs), and evaluated the learned latent structure in the hidden layers of the RINN—in an attempt to learn components of cancer cellular signaling system. The RINN architecture modifies a deep neural network to allow inputs to directly act on any hidden node throughout the neural network latent structure. Just as mutations can occur in the cell membrane receptor of a signaling pathway or at the level of transcription factors (further downstream in the signaling pathway), we hypothesized that the RINN modification described above will allow SGAs to act at different hidden layers corresponding to the level in the signaling pathway hierarchy that they affect in reality. More specifically, we hypothesize that since alterations in DNA cause changes in the functions of the proteins they encode and that such changes eventually cause changes in gene expression, we can train a supervised deep neural network with redundant inputs to predict gene expression values from DNA mutation data and then interpret the connections within this trained network as causal relationships. This means that by constraining the input data to be data that is known to cause our output data, under appropriate causal assumptions (Chapter 4.7), we can interpret the structure learned by the weights of a neural network as a

causal graph. We applied the RINN model to a dataset derived from TCGA, and we show that a RINN can capture cancer signaling pathway relationships within its hidden variables and weights.

6.3 Methods

6.3.1 Data

The data used for this chapter was originally downloaded from TCGA [Weinstein et al., 2013, Cai et al., 2019]. RNA Seq, mutation, and copy number variation (CNV) data over multiple cancer types were used to generate two binary datasets. A differentially expressed gene (DEG) dataset was created by comparing the expression value of a gene across tumor samples against a distribution of the expression values of the gene across normal samples from the same tissue of origin. A gene is deemed a DEG in a tumor if its value is outside the 2.5% percentile on either side of the normal sample distribution, and then that gene's value was set to 1. Otherwise, the gene's value was set to 0. A somatic genome alteration (SGA) dataset was created by using mutation and CNV data. A gene was deemed to be perturbed by an SGA event if it hosts a non-synonymous mutation, small insert deletion, or copy number alteration (deletion or amplification). If perturbed, the value in the tumor for that gene was set to 1, otherwise the value was set to 0.

We applied the tumor-specific causal inference (TCI) algorithm [Cai et al., 2019, Cooper et al., 2018] to these two matrices to identify the SGAs that causally influence gene expression in tumors and the union of their target DEGs. TCI is an algorithm that finds the SGAs that are most likely causing changes in gene expression, i.e., finding causal relationships between SGAs and DEGs in each individual tumor, without determining how the signal from the SGA is propagated in the cellular signaling system [Cai et al., 2019, Cooper et al., 2018]. We identified 372 SGAs that were deemed driver SGAs, as well as 5,259 DEGs that were deemed as target DEGs of the 372 SGAs. Overall, this led to two datasets, each with 5,097 instances (tumor samples). In this dissertation, the SGAs (inputs) are used to predict DEGs (outputs).

6.3.2 Deep Learning Strategies: RINN and DNN

For this work, we used two deep learning strategies: RINN and DNN. A DNN is a conventional supervised feed-forward deep neural network. A DNN learns a function mapping inputs ($\boldsymbol{x}$) to outputs ($\boldsymbol{y}$) according to Equation 4.19. This function represents our predicted value for $\boldsymbol{y}$ and, in words, represents vector-matrix multiplication followed by application of a nonlinear function, repeated multiple times. An iterative procedure (stochastic gradient descent) is used to slowly change the values of all $\boldsymbol{W}_i$ to bring $\hat{\boldsymbol{y}}$ closer and closer to $\boldsymbol{y}$ (hopefully). The left side of Figure 10, without the redundant input nodes and corresponding redundant input weights, represents a deep neural network. For a more detailed explanation of DNNs, please see [Goodfellow et al., 2016].

We introduced the RINN for latent causal structure discovery in Chapter 4.4. A RINN is similar to a DNN with a slight modification of the architecture. A RINN not only has the input fully connected to the first hidden layer, but also has copies of the input fully connected to each additional hidden layer (Figure 10). This structure allows a RINN to learn direct causal relationships between an input SGA and any hidden node in the RINN. Forward propagation through a RINN is performed as it is for a DNN with multiple vector-matrix multiplications and nonlinear functions, except that a RINN has hidden layers concatenated to copies of the input (Figure 10). Each hidden layer of a RINN with redundant inputs is calculated according to Equation 4.17. In contrast to a RINN, a plain DNN calculates each hidden layer according to Equation 4.20.

Backpropagation of errors and stochastic gradient descent for a RINN are the same as for a DNN, but with additional weights to optimize. In addition to the architecture modification, all RINNs in this dissertation importantly included L_1 regularization of the weights as part of the objective function. The other component of the objective function was binary cross-entropy error. All DNNs also used L_1 regularization of weights plus binary cross-entropy error as the objective function to be optimized. All RINNs used in this dissertation had eight hidden layers (with varying sizes of the hidden layers), as we hypothesized that cancer cellular signaling pathways do not have more than eight levels of hierarchy. However, this assumption can easily be updated if found to be false.

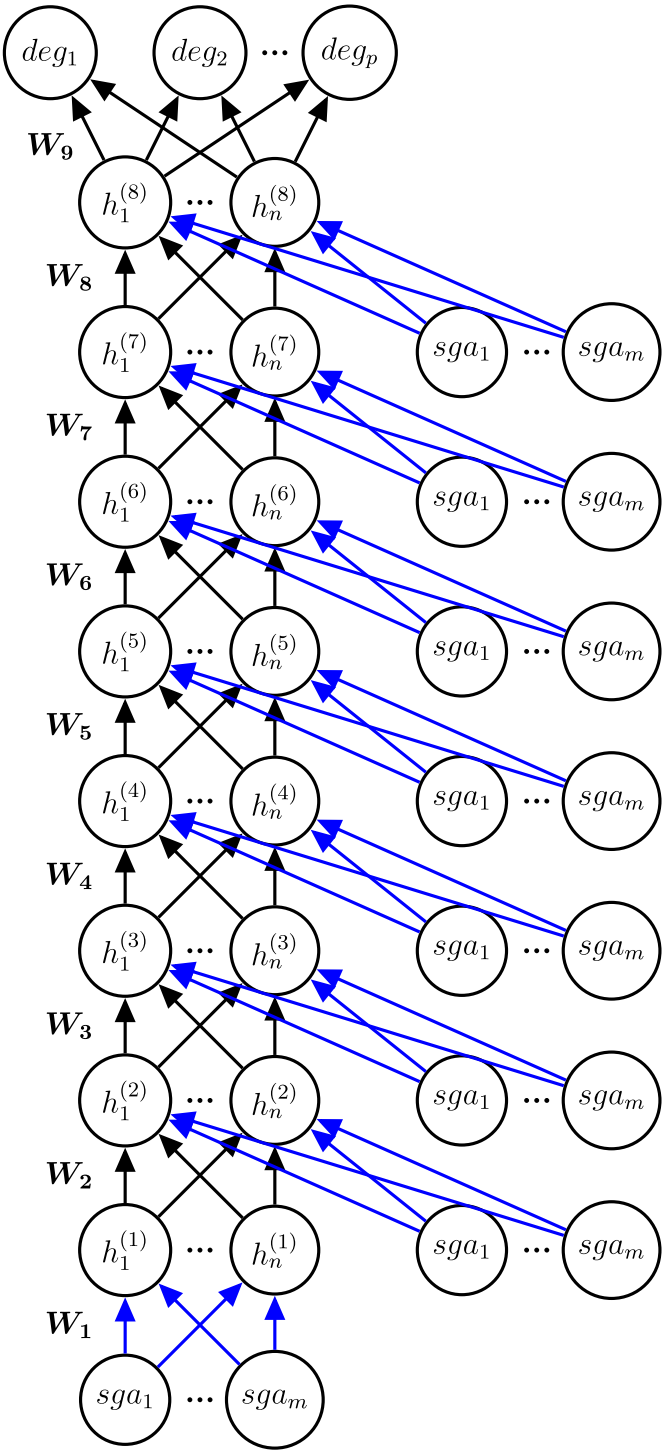


Figure 10: Redundant Input Neural Network (RINN) for TCGA Data.

A RINN with eight hidden layers ($h^{(1)}$, ..., $h^{(8)}$, each with n nodes), m inputs (sga_1, ..., sga_m on bottom), p outputs (deg_1, ..., deg_p), and seven sets of redundant inputs (sga_1, ..., sga_m on right side). Each node represents a scalar value and each edge represents a scalar weight. The weights between layers are collected in weight matrices $\boldsymbol{W_1}$, ..., $\boldsymbol{W_9}$. Blue edges represent the weights used to create SGA weight signatures.

6.3.3 Model Selection

To have any chance of finding correct causal structure with a RINN, we hypothesized that our training would need to get close to the objective function's global minimum. Therefore, we performed an extensive amount of model selection. To this end, we performed model selection over 23,000 different sets of hyperparameters (23,000 for RINN and 23,000 for DNN).

Model selection was performed using 3-folds of a 10-fold cross-validation. Using 3-folds of a 10-fold cross-validation gives us multiple validation datasets so we avoid chance overfitting that can occur with a single validation set. This setup also allows us to train on 90% of the data (and validate on 10%) for each split of the data, which is important considering the small number of instances relative to the number of output DEGs that we were trying predict. Importantly, 3-folds of a 10-fold cross-validation takes significantly less time than training on all 10-folds. Running time is especially important for this dissertation because of the large number of hyperparameter sets to be evaluated. All metrics recorded in this chapter (i.e., AUROC, cross-entropy error, etc.) represent the mean across the three validation sets.

The hyperparameters searched over for this chapter include: learning rate, regularization rate, training epochs, activation function, size of hidden layer, number of hidden layers (DNN only; RINN set to eight), and batch size. We used a combined random and grid search approach to find the best sets of hyperparameters [Bengio, 2012, Hinton, 2012] with the main objective of finding the optimal balance between sparsity and cross-entropy as explained in Section 6.3.4 and in Section 4.8.

6.3.4 Ranking Models Based on a Balance Between Sparsity and Prediction Error

Model selection in this chapter is more complex then standard DNN model selection as we needed to find a balance between the sparsity of the model (i.e., sets of weight matrices for each trained network) and prediction error. In contrast, many DNNs are trained by simply finding the model with the lowest prediction error on a hold-out dataset. As was shown in Chapter 5, RINN models trained on simulated data with relatively high sparsity and low cross-entropy error were able to recover much of the latent causal structure contained

within the data. We followed the same procedure as in Section 4.8 for selecting the best models (i.e., the models with the highest chance of containing correct causal structure). In brief, we plotted prediction error versus sparsity (see Figure 12) and measured the Euclidean distance from the origin to a set of hyperparameters, i.e., a unique trained neural network (blue circles in the figure). This distance was measured according to Equation 4.21. The sets of hyperparameters were then ranked according to shortest distance to the origin (smallest d_x) and then we retrained on all data for further analysis of the learned weights. Please see Section 4.8 for a more detailed explanation.

6.3.5 AUROC and Other Metrics

Area Under the Receiver Operating Characteristics (AUROC) were calculated for each DEG and then averaged over the three validation sets, leading to 5,259 AUROCs for each classifier in Figure 13. k-nearest neighbors (kNN) was performed using sklearn's KNeighborsClassifier [Pedregosa et al., 2011]). Both Euclidean and Jaccard distance metrics were evaluated and the best k values were 21 and 45, respectively. Other distance metrics were evaluated with results worse than those using the Jaccard distance metric. For the random control, we sampled predictions from a uniform distribution over the interval $[0, 1)$ for each of the validation sets. Then used these predictions to calculate AUROCs for each validation set. As with the other AUROC calculations, we took the mean over the three validation sets.

For Table 6, the AUROCs were calculated as described above and then the mean over all DEGs was calculated. The same procedure was followed for calculating cross-entropy error and Area Under the Precision-Recall curve (AUPR). The same random predictions described above were also used to calculate cross-entropy error and AUPR.

6.3.6 Ground Truth

Throughout this work, the results from [Sanchez-Vega et al., 2018] were used as ground truth for comparison purposes. Specifically, Figures S1 and 2 were used as ground truth causal relationships that we hypothesize the RINN may be able to find. Figure S1 is reproduced here, with a minor formatting modification, as Figure 11 for convenience. Of the 372 genes in

our SGA dataset there were 35 genes that overlapped with the genes in Figure 11. Therefore, the causal relationships that we can find are limited to the relationships between these 35 genes (Table 4).

6.3.7 SGA Weight Signature

To better understand how a RINN learns to connect SGAs to hidden nodes, we analyzed only the weights going from SGA nodes to hidden nodes. To accomplish this, we generated an "SGA Weight Signature" for each SGA, which is the concatenation, into a single vector, of all weights for a single SGA going from that SGA to all hidden nodes in all hidden layers. This can be visualized as the concatenation of the blue weights in Figure 10 into one long vector for each SGA. The end result is an SGA weight signature matrix of SGAs by hidden nodes. The weight signatures for a DNN were generated using only a single weight matrix, the weight matrix between the inputs and the first hidden layer (i.e., $\boldsymbol{W_1}$), as these are the only weights in a DNN that are specific to individual SGAs.

Cosine similarity between SGA weight signatures was measured using the sklearn function *cosine_similarity* [Pedregosa et al., 2011].

Hierarchical clustering using cosine similarity and average linkage was performed on the SGA weight signatures using the seaborn python module function *clustermap*.

6.3.8 Community Detection

Cosine similarity community detection figures were generated using Gephi [Bastian et al., 2009] and the cosine similarity between SGA weight signatures. Edges represented the highest or three highest cosine similarities for a given SGA. After generating a graph where nodes were SGAs and edges were highest cosine similarities, we ran the *Modularity* function in Gephi to perform community detection. *Modularity* runs an algorithm based on [Blondel et al., 2008, Lambiotte et al., 2008]. Next, we partitioned and colored the graph based on community.

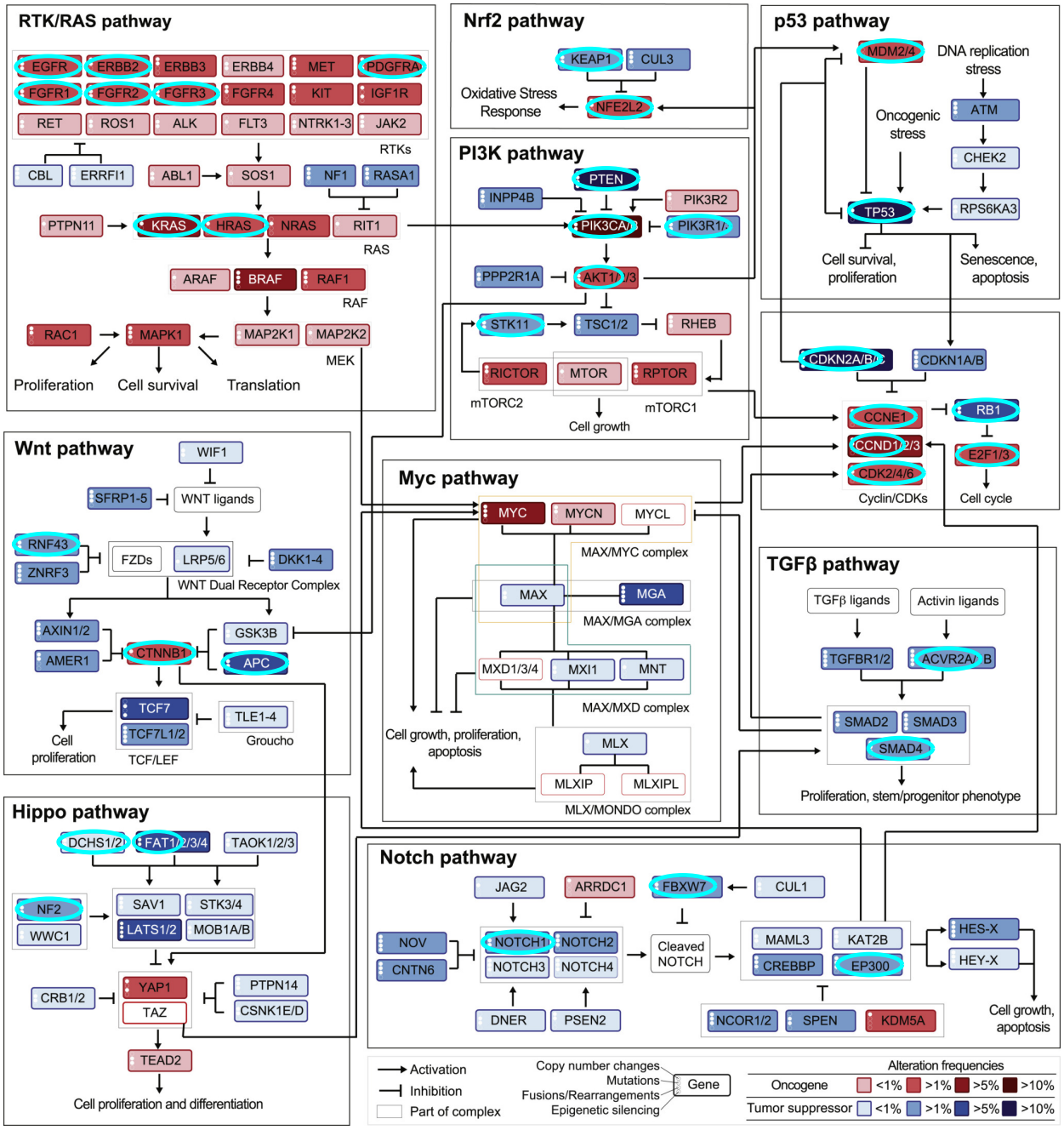


Figure 11: Genes in SGA Dataset that Overlap with Sanchez-Vega et al., 2018

This is Figure S1 from Sanchez et al., 2018: "Curated Pathways Including Cross-Cross Pathway Interactions" [Sanchez-Vega et al., 2018]. Circled in cyan are all genes from these pathways that are in our SGA dataset. We are treating these pathways as ground truth for comparison purposes. The cyan circles have been added to the original figure.

Table 4: Genes in SGA Dataset that Overlap with Sanchez-Vega et al., 2018

Pathway from Figure 11	Genes in SGA Dataset and Figure 11
RTK/RAS	EGFR, FGFR1, ERBB2, FGFR2, FGFR3, PDGFRA, KRAS, HRAS
Nrf2	KEAP1, NFE2L2
TGFβ	ACVR2A, SMAD4
PI3K	PTEN, PIK3CA, PIK3R1, AKT1, STK11
p53	MDM2, CDKN2A, TP53
Cell cycle	CDKN2A, CDKN2B, CCNE1, CCND1, CDK4, RB1, E2F3
Notch	NOTCH1, FBXW7, EP300
Hippo	DCHS2, FAT1, NF2
Wnt	RNF43, APC, CTNNB1

6.3.9 Visualizing a RINN as a Causal Graph

Let $G_{i,j} = (V, E)$, where $G_{i,j}$ is a causal directed acyclic graph with vertices V and edges E for neural network i and set of SGAs j. For this work, V represents SGAs and hidden nodes, and E represents directed weighted edges with weight values corresponding to the weights of a RINN. The edges are directed from SGAs (input) to DEGs (output), as we know from biology that SGAs cause changes in expression. Please see Section 4.7 for further explanation of interpreting a RINN in a causal framework.

If we simply interpreted any nonzero weight in a trained RINN as an edge in a $G_{i,j}$, there would be hundreds of thousands to millions of edges in the causal graph (depending on the size of the hidden layers) as L_1 regularization encourages weight values toward zero, but weights are often not actually zero. This means that some thresholding of the weights was required (Chapters 4 and 5).

Even after selecting models based on the best balance between sparsity and error (smallest d_x), our weight matrices were still very dense in terms of what can be readily interpreted visually (even after rounding weights to zero decimals). Therefore, a threshold weight value

was needed to limit weight visualizations to only the largest (and we suspect most causally important) weights. To accomplish this, we first limited the weights to be visualized to only those that are descendants (i.e., downstream) of any of the 35 SGAs from Figure 11. Next, for each of the top ten RINN models (ten shortest d_x), we found the absolute value weight threshold that led to 300 edges in total, including all SGA to hidden and hidden to hidden edges. This threshold varied slightly from model to model, ranging from 0.55 to 0.71. This threshold was chosen as it seemed to give a biologically reasonable density of edges, and allowed us to best recover some of the relationships in Figure 11.

After finding a threshold for each model, any weight whose absolute value was greater then the threshold was added as an edge to the causal graph $G_{i,j}$ for that model. The causal graphs were then plotted as modified bipartite graphs (see Figure 16) with SGAs on one side and all hidden nodes on the other. Hidden to hidden edges were included as arcing edges on the outside of the bipartite graph. We labeled hidden nodes using a recursive algorithm that found all ancestor or upstream SGAs (i.e., on a path to that hidden node) for a given hidden node and graph $G_{i,j}$.

6.3.10 Finding Shared Hidden Nodes

To determine if similarly connected hidden nodes were shared across the best models, we needed a method to map hidden nodes to some meaningful label. To accomplish this, we used the same recursive algorithm (described at the end of Section 6.3.9) to map each hidden node in a causal graph $G_{i,j}$ to the set of SGAs that were ancestors of that hidden node. For this mapping, only the SGAs in the set j could be used to label a hidden node as these are the only SGAs in $G_{i,j}$. We performed this mapping using graphs generated with j set to only the SGAs in the individual pathways from Figure 11 (e.g., $j = \{AKT1, PIK3CA, PIK3R1, PTEN, STK11\}$ for the PI3K pathway). For each model in the best ten models and each pathway in Figure 11, we mapped hidden nodes to a set of SGAs. Next, we compared the labeled hidden nodes across models and determined the number of models that shared identically labeled hidden nodes (see Table 9).

We compared the number of models that shared specific hidden nodes with random controls.

The random controls were performed the same as described above for the experimental results, except that j was not set to the SGAs in one of the pathways in Figure 11—rather j was selected randomly from the set of SGAs including all 372 SGAs minus the 35 SGAs in Figure 11. Then ten $G_{i,j}$, one for each of the top ten RINN models, were generated using the random SGAs. The number of models with shared hidden nodes was recorded. This procedure was repeated 30 times for each possible number of SGAs in j. For example, the PI3K pathway (in Figure 11) has five SGAs in it that are also in our SGA dataset. To perform the random control for PI3K, we performed 30 replicates of randomly selecting five SGAs (from the set of SGAs described above) and then recorded the mean number of models sharing an n-SGA labeled hidden node, where n is the number of SGAs a hidden node mapped to using our recursive algorithm for finding ancestors.

6.4 Results

6.4.1 Model Selection

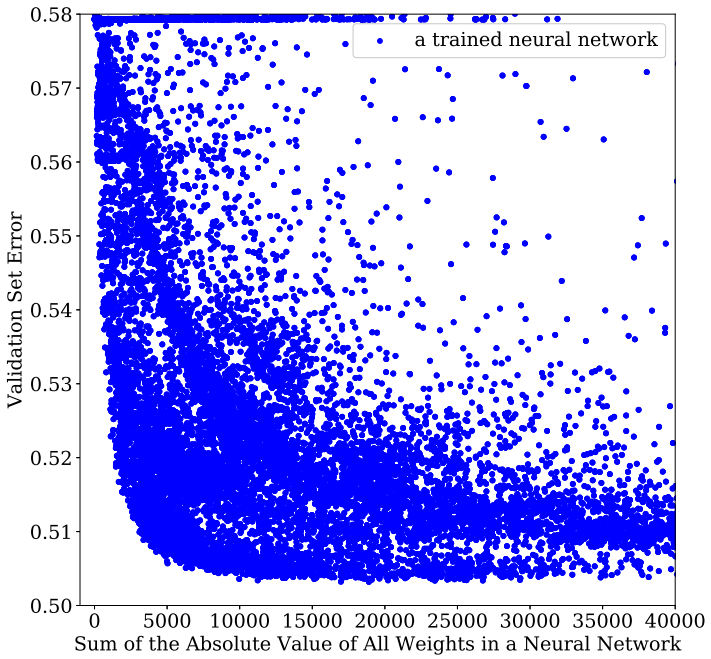
We trained approximately 23,000 RINN and 23,000 DNN models on the TCGA training dataset with distinct sets of hyperparameters (e.g., number of hidden nodes, activation function, regularization rate, learning rate, etc.). We evaluated each trained model on multiple validation datasets to evaluate how well it performed (Section 6.3.3). We hypothesized that the models with the most parsimonious weight structure, while still maintaining the ability to accurately capture the statistical relationship between SGAs and DEGs, likely have learned optimal representations of the impact of SGAs in cancer cells. To reflect the balance between these two objectives, we visualized the performance of all models on a scatter plot with validation-set error and the sum of the absolute value of all weights as axes (Figure 12). Each dot in this figure represents a neural network trained on a unique set of hyperparameters. We ranked models based on their Euclidean distance to the origin (d_x), and models with the shortest d_x were selected as the models with the best balance between sparsity and validation-set error.

The ten best RINN models (i.e., models with lowest d_x) are shown in Table 5. Among multiple activation functions studied, the softplus activation function overwhelmingly provided the best results. This was also seen with the best DNN models. Overall, the number of hidden nodes in each hidden layer for the RINN were relatively small ($\sim$100) compared to the dimensionality of output space (5,259). Despite being trained with eight hidden layers, the best RINN models utilized only three or four of the eight hidden layers (see Figures 15, 16 for examples). All top ten DNN models had two hidden layers with hidden layer sizes ranging from 50 to 644 hidden nodes.

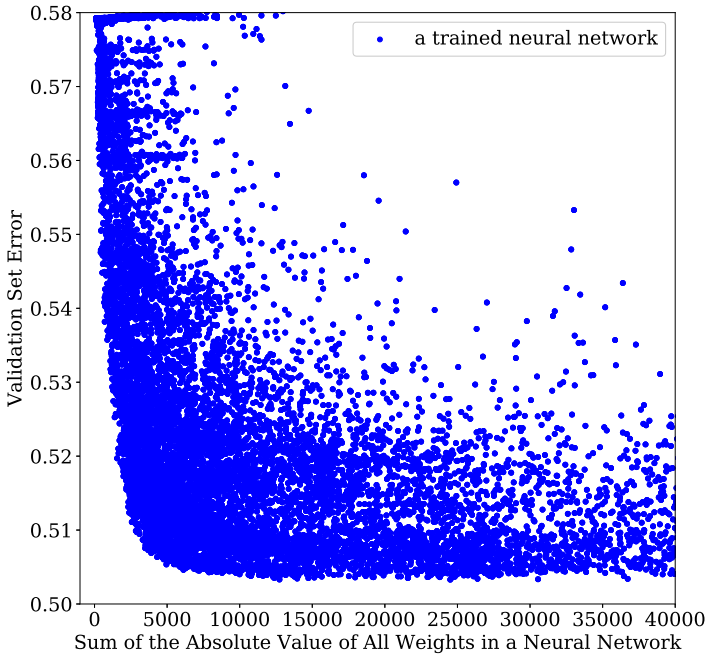
6.4.2 Predicting DEG Status Given SGAs

We examined how well deep learning models can predict DEGs given SGAs as inputs, using different metrics, including cross-entropy loss, AUROC, and AUPR. Cross-validation model selection metrics for RINN and DNN are compared in Table 6. Table 6 shows the mean and standard deviation of the metrics on all 5,259 DEGs. In general, RINN and DNN performed identically across all metrics. RINN and DNN performed significantly better than k-nearest neighbors or random controls. The best RINN and DNN models according to shortest d_x (meaning more regularized) performed similarly, but slightly worse, than the much less regularized RINNs and DNNs. $\sum_1^8 |\boldsymbol{W_i}|$ is a surrogate measure of the density of a neural network, with higher values indicating higher density of edges.

To get a better idea of how well the models performed at predicting individual DEGs from SGAs, we plotted DEG AUROC histograms for individual models and relevant control models (Figure 13). Figure 13 shows AUROCs for all 5,259 DEGs for the best RINN and DNN models. The RINN and DNN models had the highest AUROCs, greatly outperforming the control models. Interestingly, the RINN and DNN models achieved AUROC > 0.8 for a large number of DEGs. Again, DNNs and RINNs performed almost identically when compared to each other. Importantly, there was not a large difference in the AUROC curves between RINN (and DNN) models selected according to lowest CEL (cross-entropy loss) and those selected using shortest ED (euclidean distance), despite the models selected according to shortest ED being much more regularized.



(a) RINN



(b) DNN

Figure 12: Model Selection Results

Table 5: Best RINN Hyperparameters

Hyperparameters for the ten RINN models with lowest Euclidean distance to origin. Each row in the table represents a set of hyperparameters that was used to fully train a RINN model. Also included are the cross-entropy errors and sum of the absolute values of the weights. ($\boldsymbol{W_i}$ is a weight matrix between hidden layers. LR: learning rate, RR: regularization rate, BS: batch size, CEL: cross-entropy loss)

Rank	Train Epochs	Nodes in Hidden	LR	RR	BS	Activation	CEL	$\sum_1^8 \lvert\boldsymbol{W_i}\rvert$
1	303	50	1E-03	5E-06	135	softplus	0.5073	4248
2	144	100	1E-03	6E-06	45	softplus	0.5081	3874
3	738	319	1.64E-04	6.37E-06	30	softplus	0.5089	3373
4	428	326	2.37E-04	5.59E-06	30	softplus	0.5079	4042
5	136	50	1E-03	6E-06	30	softplus	0.5082	4021
6	311	50	1E-03	6E-06	135	softplus	0.5090	3596
7	713	148	1.1E-04	5.17E-06	30	softplus	0.5083	4038
8	326	100	1E-03	6E-06	170	softplus	0.5088	3726
9	149	101	5.23E-04	4.49E-06	30	softplus	0.5057	5030
10	345	180	3.02E-04	5.66E-06	30	softplus	0.5085	3925

Table 6: Cross-Validation Model Selection Results

Comparing the two RINN and DNN models with the lowest mean cross-entropy validation set loss and shortest euclidean distance to the origin after evaluating ∼23,000 different sets of hyperparameters. Values represent mean across three validation sets. The four best models for each metric in bold. ($\boldsymbol{W_i}$ is a weight matrix between hidden layers. CEL: cross-entropy loss, ED: Euclidean distance).

	Description	CEL	AUROC	AUPR	$\sum_1^8 \lvert\boldsymbol{W_i}\rvert$
RINN 1	lowest CEL	**0.5032** $\pm$ 0.0048	**0.7316** $\pm$ 0.0051	**0.5678** $\pm$ 0.0039	18899 $\pm$ 250
RINN 2	2nd lowest CEL	**0.5032** $\pm$ 0.0047	**0.7317** $\pm$ 0.0048	**0.5677** $\pm$ 0.0036	16971 $\pm$ 53
DNN 1	lowest CEL	**0.5033** $\pm$ 0.0052	**0.7318** $\pm$ 0.0058	**0.5676** $\pm$ 0.0041	24503 $\pm$ 124
DNN 2	2nd lowest CEL	**0.5033** $\pm$ 0.0049	**0.7307** $\pm$ 0.0056	0.5663 $\pm$ 0.0032	12739 $\pm$ 75
RINN ED 1	shortest ED	0.5073 $\pm$ 0.0054	0.7258 $\pm$ 0.0054	0.5607 $\pm$ 0.0040	**4248** $\pm$ 18
RINN ED 2	2nd shortest ED	0.5081 $\pm$ 0.0058	0.7231 $\pm$ 0.0057	0.5569 $\pm$ 0.0038	**3874** $\pm$ 48
DNN ED 1	shortest ED	0.5075 $\pm$ 0.0055	0.7238 $\pm$ 0.0057	0.5570 $\pm$ 0.0029	**3491** $\pm$ 18
DNN ED 2	2nd shortest ED	0.5073 $\pm$ 0.0054	0.7240 $\pm$ 0.0054	0.5576 $\pm$ 0.0031	**3704** $\pm$ 45
k-NN	$k = 21$, euclidean	9.6289 $\pm$ 0.1889	0.5599 $\pm$ 0.0034	0.5396 $\pm$ 0.0022	NA
k-NN	$k = 45$, jaccard	9.0740 $\pm$ 0.1579	0.5799 $\pm$ 0.0028	**0.5719** $\pm$ 0.0027	NA
Random	preds $\sim U(0, 1)$	0.9994 $\pm$ 0.0003	0.5003 $\pm$ 0.0003	0.3257 $\pm$ 0.0052	NA

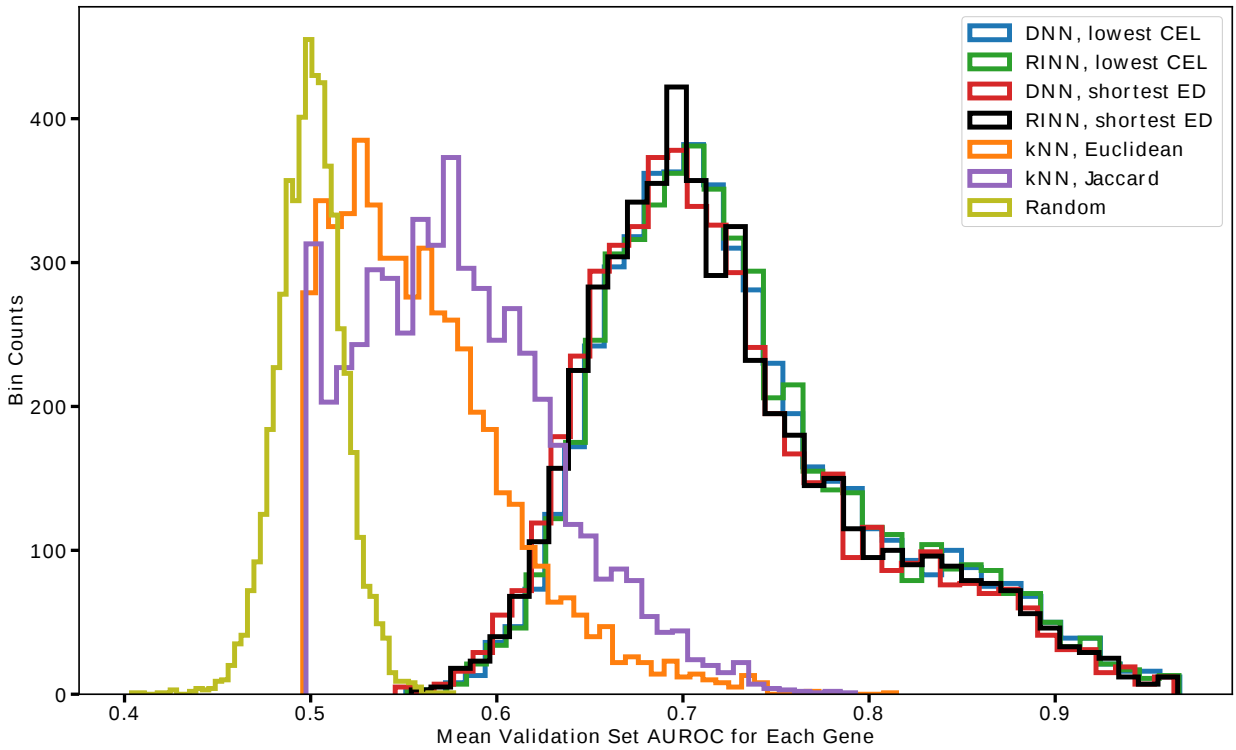


Figure 13: AUROC for RINN and DNN

6.4.3 Learning Shared Functional Impacts of SGAs Using the RINN

RINN allows an SGA to interact with all hidden nodes in the latent hierarchy. After training with regularization, the majority of the weights from SGAs to hidden nodes were set close to 0.0, thus the remaining weights (not close to 0) reflect how the impact of the SGAs is propagated through the latent variables and eventually influences gene expression. In other words, these remaining weights (edges) between the SGAs and hidden nodes serve as a signature (or vector embedding) representing an SGA's functional impact. One can compare the functional impacts of two SGAs by comparing their weight signatures with cosine angle similarity. Here, we assess whether the weight signatures learned by RINN truly reflect the functional impact of SGAs. We hypothesized that if RINN correctly captures the functional impact of SGAs, then the weight signature of a gene (SGA) should be more similar to genes belonging to a common pathway than to genes from other pathways. To this end, we used the well documented cancer pathways recently reported by the TCGA pan-cancer analysis project [Sanchez-Vega et al., 2018] (Figure 11) to determine which genes shared a common pathway.

For each of the 35 driver SGAs that are in the known pathways from the TCGA pan-cancer study [Sanchez-Vega et al., 2018] (and also are in our SGA dataset), we calculated the cosine similarity of its weight signature with respect to all other SGA weight signatures, and listed the three most similar SGAs in Table 7. Clearly, many of the SGAs that had high cosine similarities relative to the query SGA were in the same pathway as the query SGA according to [Sanchez-Vega et al., 2018]. Most of the SGAs in the PI3K pathway had high cosine similarities with the other SGAs in the PI3K pathway. KEAP1 and NFE2L2 (the only members of the Nrf2 pathway in SGA dataset) were both found to be each other's SGA with highest cosine similarity. Also, SMAD4 and ACVR2A (only members of TGFβ pathway in SGA dataset) were also found to be each other's SGA with highest cosine similarity. Additional cosine similarity relationships were found between members of the same pathway for the remaining pathways, but these relationships were less frequent than the results for PI3K, Nrf2, and TGFβ mentioned above. These cosine similarity results suggest that the hidden nodes in our trained RINNs may represent biological entities or components of a

cellular signaling system, meaning that the connectivity found in our trained RINNs are related to cellular signaling pathways.

To compare the RINN with a DNN, we performed the same cosine similarity experiment from Table 7, but with weights from the best three DNN models (Table 8). For this experiment, we only used weights from the input SGAs to the first hidden layer, as these are the only weights in a DNN that are specific to individual SGAs. In Table 8, many of the SGAs that had high cosine similarities relative to the query SGA were in the same pathway as the query SGA according to [Sanchez-Vega et al., 2018]. Overall, there were 28 genes in bold in Table 7 (RINN) and 27 genes in bold in Table 8 (DNN). These results indicate that the DNN weight signature cosine similarities were very similar to the RINN weight signature cosine similarities. The DNNs performed worse at capturing the relationships in the PI3K pathway, but better at capturing pathway relationships in the RTK/RAS pathway (especially with cosine similarities relative to FGFR3). Just as with the RINN, KEAP1 and NFE2L2 were both found to be each other's SGA with highest cosine similarity with the DNN. SMAD4 and ACVR2A were also found to be similar to one another, but not as highly similar as with the RINN. It is also likely that some or many of the hidden nodes in trained DNNs represent biological entities. Although, in a DNN the information is likely to be more convoluted, and the RINN is an attempt to deconvolute the signal into relationships that are more easily interpreted by a human. The similarities between Tables 7 and 8 are quite remarkable and show that the RINN and DNN are capturing similar information overall, which is somewhat expected as they both performed similarly on the same prediction task.

An alternative approach to illustrating the information provided by SGA weight signatures is to visualize the highest cosine similarities between weight signatures as edges in a graph and then apply a community discovery algorithm to identify closely related SGAs (Figure 14). In this figure, the discovered communities are shown as nodes and edges with the same color. In Figure 14, a directed edge connects an SGA (at the tail) to another SGA (at the head) that has high cosine similarity relative to the SGA at the tail of the edge; a bi-directed edge indicates that two SGAs are mutually highly similar; the thickness of an edge represents the amount of cosine similarity; the size of the SGA node represents the degree of that node. Figure 14a is a visualization of the highest cosine similarities for each SGA and Figure 14b is

Table 7: RINN Top Three SGA Weight Signature Cosine Similarity (CS)

SGAs from Figure 11 with highest cosine similarity relative to SGA x (mean CS across three best models ± st. dev.). Bolded SGAs are in the same pathway (according to Sanchez-Vega et al., 2018) as SGA x.

x	Highest CS (Mean ± SD)	2nd Highest	3rd Highest
PTEN	**PIK3R1** (0.8 ± 0.02)	PDGFRA (0.4 ± 0.01)	**PIK3CA** (0.4 ± 0.05)
PIK3CA	**PIK3R1** (0.6 ± 0.04)	**AKT1** (0.4 ± 0.06)	**PTEN** (0.5 ± 0.05)
PIK3R1	**PTEN** (0.8 ± 0.02)	**PIK3CA** (0.6 ± 0.04)	CTNNB1 (0.4 ± 0.05)
AKT1	**PIK3CA** (0.4 ± 0.06)	FGFR2 (0.4 ± 0.03)	DCHS2 (0.3 ± 0.09)
STK11	KEAP1 (0.5 ± 0.05)	FGFR1 (0.5 ± 0.05)	NFE2L2 (0.4 ± 0.10)
KEAP1	**NFE2L2** (0.6 ± 0.09)	STK11 (0.5 ± 0.05)	KRAS (0.4 ± 0.08)
NFE2L2	**KEAP1** (0.6 ± 0.09)	STK11 (0.4 ± 0.10)	FGFR1 (0.3 ± 0.11)
ACVR2A	**SMAD4** (0.5 ± 0.04)	RNF43 (0.4 ± 0.20)	APC (0.2 ± 0.10)
SMAD4	**ACVR2A** (0.5 ± 0.04)	APC (0.4 ± 0.06)	DCHS2 (0.4 ± 0.02)
CDKN2A	NOTCH1 (0.7 ± 0.09)	HRAS (0.5 ± 0.07)	FAT1 (0.4 ± 0.10)
CDKN2B	**RB1** (0.4 ± 0.14)	EGFR (0.4 ± 0.18)	**CDK4** (0.4 ± 0.18)
CCNE1	**E2F3** (0.5 ± 0.23)	**RB1** (0.3 ± 0.20)	MDM4 (0.3 ± 0.20)
CCND1	FGFR2 (0.7 ± 0.05)	DCHS2 (0.3 ± 0.11)	FBXW7 (0.3 ± 0.14)
CDK4	PDGFRA (0.5 ± 0.10)	EGFR (0.4 ± 0.11)	**CDKN2B** (0.4 ± 0.18)
RB1	FGFR3 (0.5 ± 0.07)	**E2F3** (0.4 ± 0.04)	**CDKN2B** (0.4 ± 0.14)
E2F3	**CCNE1** (0.5 ± 0.23)	FGFR3 (0.4 ± 0.10)	**RB1** (0.4 ± 0.04)
EGFR	**PDGFRA** (0.5 ± 0.10)	CDK4 (0.4 ± 0.11)	CDKN2B (0.4 ± 0.18)
FGFR1	DCHS2 (0.6 ± 0.11)	STK11 (0.5 ± 0.05)	FBXW7 (0.5 ± 0.07)
ERBB2	EP300 (0.6 ± 0.08)	**FGFR3** (0.5 ± 0.21)	RB1 (0.4 ± 0.08)
FGFR2	CCND1 (0.7 ± 0.05)	FBXW7 (0.4 ± 0.09)	DCHS2 (0.4 ± 0.11)
FGFR3	RB1 (0.5 ± 0.07)	EP300 (0.5 ± 0.02)	**ERBB2** (0.5 ± 0.21)
PDGFRA	CDK4 (0.5 ± 0.10)	**EGFR** (0.5 ± 0.10)	PTEN (0.4 ± 0.10)
KRAS	CTNNB1 (0.5 ± 0.06)	KEAP1 (0.4 ± 0.08)	RNF43 (0.3 ± 0.15)
HRAS	NOTCH1 (0.7 ± 0.05)	FAT1 (0.6 ± 0.05)	CDKN2A (0.5 ± 0.07)
MDM4	PDGFRA (0.3 ± 0.14)	EGFR (0.3 ± 0.16)	CCNE1 (0.3 ± 0.10)
TP53	E2F3 (0.4 ± 0.03)	RB1 (0.3 ± 0.08)	CCNE1 (0.3 ± 0.09)
RNF43	**APC** (0.4 ± 0.03)	ACVR2A (0.4 ± 0.20)	KRAS (0.3 ± 0.15)
APC	FBXW7 (0.5 ± 0.06)	DCHS2 (0.5 ± 0.04)	SMAD4 (0.4 ± 0.06)
CTNNB1	KRAS (0.5 ± 0.06)	PIK3R1 (0.4 ± 0.08)	FBXW7 (0.4 ± 0.07)
DCHS2	FBXW7 (0.8 ± 0.04)	EP300 (0.6 ± 0.04)	FGFR1 (0.6 ± 0.11)
FAT1	NOTCH1 (0.8 ± 0.07)	HRAS (0.6 ± 0.05)	CDKN2A (0.4 ± 0.10)
NF2	EP300 (0.3 ± 0.16)	FBXW7 (0.3 ± 0.10)	NOTCH1 (0.3 ± 0.03)
FBXW7	DCHS2 (0.8 ± 0.04)	**EP300** (0.6 ± 0.08)	APC (0.5 ± 0.06)
NOTCH1	FAT1 (0.8 ± 0.07)	HRAS (0.7 ± 0.05)	CDKN2A (0.7 ± 0.09)
EP300	**FBXW7** (0.6 ± 0.08)	DCHS2 (0.6 ± 0.04)	ERBB2 (0.6 ± 0.08)

Table 8: DNN Top Three SGA Weight Signature Cosine Similarity (CS)

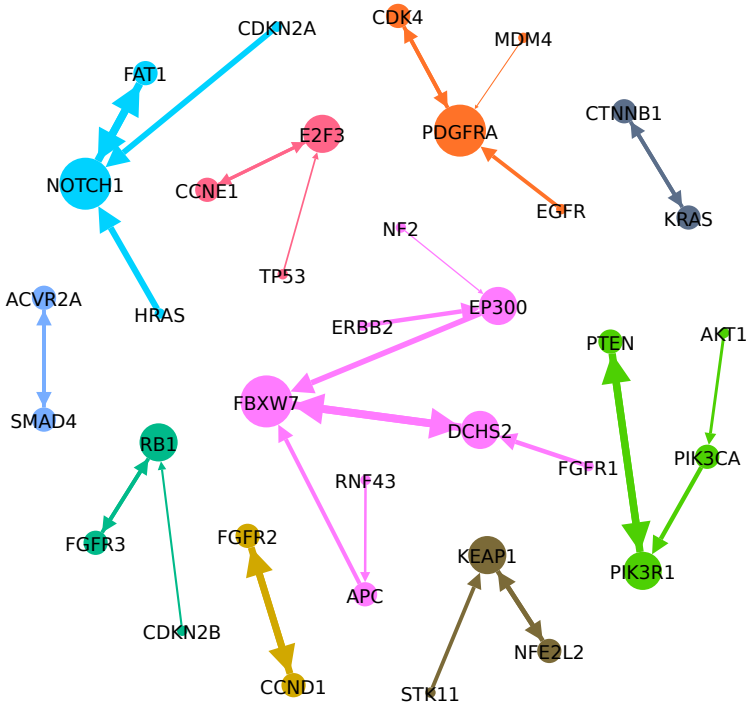
SGAs from Figure 11 with highest cosine similarity relative to SGA x (mean CS across three best models $\pm$ st. dev.). Bolded SGAs are in the same pathway (according to Sanchez-Vega et al., 2018) as SGA x.

x	Highest CS (Mean $\pm$ SD)	2nd Highest	3rd Highest
PTEN	**PIK3R1** (0.7 ± 0.06)	FGFR2 (0.5 ± 0.05)	PDGFRA (0.4 ± 0.03)
PIK3CA	**AKT1** (0.7 ± 0.06)	FGFR2 (0.5 ± 0.02)	CCND1 (0.5 ± 0.10)
PIK3R1	FGFR2 (0.7 ± 0.07)	**PTEN** (0.7 ± 0.06)	CTNNB1 (0.6 ± 0.02)
AKT1	**PIK3CA** (0.7 ± 0.06)	FGFR2 (0.5 ± 0.02)	FGFR1 (0.4 ± 0.14)
STK11	KEAP1 (0.6 ± 0.02)	FGFR1 (0.4 ± 0.12)	NFE2L2 (0.4 ± 0.01)
KEAP1	**NFE2L2** (0.7 ± 0.03)	STK11 (0.6 ± 0.02)	KRAS (0.4 ± 0.03)
NFE2L2	**KEAP1** (0.7 ± 0.03)	STK11 (0.4 ± 0.01)	NOTCH1 (0.4 ± 0.04)
ACVR2A	RNF43 (0.6 ± 0.03)	**SMAD4** (0.5 ± 0.03)	CCNE1 (0.4 ± 0.15)
SMAD4	APC (0.5 ± 0.04)	FBXW7 (0.5 ± 0.02)	**ACVR2A** (0.5 ± 0.03)
CDKN2A	NOTCH1 (0.8 ± 0.06)	**CDK4** (0.7 ± 0.02)	HRAS (0.6 ± 0.02)
CDKN2B	**CDK4** (0.6 ± 0.04)	**RB1** (0.5 ± 0.07)	EGFR (0.4 ± 0.03)
CCNE1	**E2F3** (0.6 ± 0.08)	TP53 (0.4 ± 0.07)	ACVR2A (0.4 ± 0.15)
CCND1	FGFR2 (0.6 ± 0.06)	PIK3CA (0.5 ± 0.10)	CTNNB1 (0.5 ± 0.12)
CDK4	**CDKN2A** (0.7 ± 0.02)	**CDKN2B** (0.6 ± 0.04)	EGFR (0.5 ± 0.03)
RB1	FGFR3 (0.5 ± 0.05)	TP53 (0.5 ± 0.02)	**CDKN2B** (0.5 ± 0.07)
E2F3	**CCNE1** (0.6 ± 0.08)	FGFR3 (0.5 ± 0.06)	**RB1** (0.4 ± 0.06)
EGFR	**PDGFRA** (0.6 ± 0.05)	CDK4 (0.5 ± 0.03)	MDM4 (0.5 ± 0.06)
FGFR1	FBXW7 (0.5 ± 0.10)	DCHS2 (0.5 ± 0.18)	STK11 (0.4 ± 0.12)
ERBB2	EP300 (0.6 ± 0.03)	**FGFR3** (0.5 ± 0.03)	DCHS2 (0.5 ± 0.10)
FGFR2	PIK3R1 (0.7 ± 0.07)	CCND1 (0.6 ± 0.06)	CTNNB1 (0.6 ± 0.04)
FGFR3	**ERBB2** (0.5 ± 0.03)	**HRAS** (0.5 ± 0.07)	EP300 (0.5 ± 0.08)
PDGFRA	**EGFR** (0.6 ± 0.05)	MDM4 (0.5 ± 0.14)	PTEN (0.4 ± 0.03)
KRAS	CTNNB1 (0.6 ± 0.07)	APC (0.5 ± 0.04)	SMAD4 (0.5 ± 0.02)
HRAS	NOTCH1 (0.7 ± 0.03)	CDKN2A (0.6 ± 0.02)	EP300 (0.6 ± 0.10)
MDM4	PDGFRA (0.5 ± 0.14)	EGFR (0.5 ± 0.06)	**CDKN2B** (0.3 ± 0.17)
TP53	RB1 (0.5 ± 0.02)	CCNE1 (0.4 ± 0.07)	KRAS (0.4 ± 0.06)
RNF43	ACVR2A (0.6 ± 0.03)	**APC** (0.4 ± 0.02)	KRAS (0.3 ± 0.03)
APC	FBXW7 (0.6 ± 0.02)	SMAD4 (0.5 ± 0.04)	KRAS (0.5 ± 0.04)
CTNNB1	PIK3R1 (0.6 ± 0.02)	FGFR2 (0.6 ± 0.04)	KRAS (0.6 ± 0.07)
DCHS2	FBXW7 (0.7 ± 0.08)	EP300 (0.6 ± 0.06)	FGFR1 (0.5 ± 0.18)
FAT1	NOTCH1 (0.8 ± 0.04)	CDKN2A (0.6 ± 0.09)	HRAS (0.5 ± 0.05)
NF2	EP300 (0.4 ± 0.05)	**FAT1** (0.4 ± 0.05)	NOTCH1 (0.3 ± 0.04)
FBXW7	DCHS2 (0.7 ± 0.08)	**EP300** (0.7 ± 0.09)	APC (0.6 ± 0.02)
NOTCH1	CDKN2A (0.8 ± 0.06)	FAT1 (0.8 ± 0.04)	HRAS (0.7 ± 0.03)
EP300	**FBXW7** (0.7 ± 0.09)	ERBB2 (0.6 ± 0.03)	DCHS2 (0.6 ± 0.05)

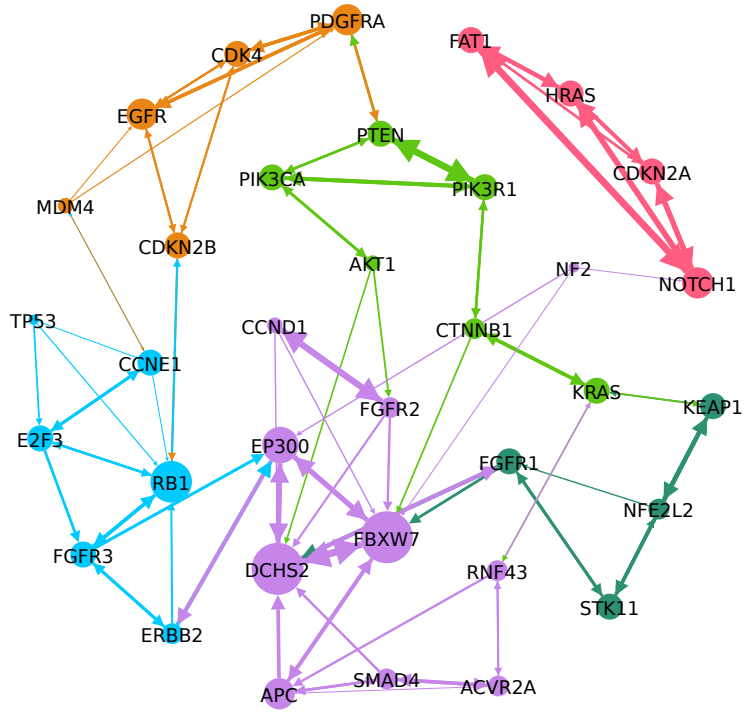
a visualization of the three highest cosine similarities for each SGA. The nodes and edges are colored according to community after running a community detection algorithm. Many of the communities that were discovered with this procedure correspond to the pathways in Figure 11. For example, the green community in Figure 14a captures the majority of the PI3K pathway, the darker blue community and brown community captures the TGFβ and Nrf2 pathways respectively, and the purple community captures many of the genes (SGAs) in the Wnt, Hippo, and Notch pathways. In Figure 14b the majority of the PI3K pathway is part of the green community and a blue community emerges with many members of the cell cycle pathway. Many other additional pathway relationships within the ground truth pathways are seen in Figure 14, reinforcing the hypothesis that the latent structure of trained RINNs contains cellular signaling pathway relationships.

We also visualized the weights from SGAs to hidden nodes (i.e., weight signatures) as a heatmap, where each column represents an SGA and each row represents a hidden node, and the weight from an SGA to a hidden node is represented as an element in the color-coded heatmap. We performed hierarchical clustering on the weight signatures derived from a single RINN model (the RINN with the shortest d_x) as shown in Figure 15. Within this clustering and dendrogram, we observe many relationships that are present in the ground truth pathways. For example, the green cluster contains much of the PI3K pathway, the orange cluster captures the TGFβ pathway, the purple cluster contains the two members of the Nrf2 pathway, and the gray cluster contains the members of the Notch pathway and two of the three members of the Hippo pathway.

In addition to the above clustering relationships, the heatmap also illustrates the capability of RINN to automatically determine the "optimal" number of hidden layers that are needed to encode information from SGAs to DEGs. Here, the hidden layers are numbered starting from input (SGAs) to output (DEGs). This heatmap shows that all the weights before hidden layer 5 have a value of 0.0, and only one hidden node in layer 5 has incoming weights with nonzero values. All of the top ten models (including this one) utilized only three or four hidden layers similar to what is seen in Figure 15. Hidden layer 7 seems to have the most nonzero incoming weights. Interestingly, many of the most important cancer related SGAs (e.g., EGFR, TP53, CDKN2A, APC, PIK3CA) have a larger number of weights and larger



(a) Edges Represent Highest Cosine Similarity for Each SGA



(b) Edges Represent 3 Highest Cosine Similarities for Each SGA

Figure 14: Cosine Similarity of SGA Weight Signatures and Community Detection

These edges do not represent causal relationships, rather cosine similarity (CS) relationships between SGAs, where the head of an edge indicates an SGA with high CS relative to the SGA at the tail.

valued weights than other SGAs in Figure 15, suggesting that this model is capturing relevant biological information.

6.4.4 Inferring Causal Relationships Using RINN

One of the goals of this work was to discover latent causal structure relevant to cancer cellular signaling pathways. We hypothesized that the RINN can reveal the causal relationships among the proteins perturbed by SGAs. For example, if an SGA (intervention) i_1 is connected to a latent variable h_1 and another SGA i_2 is connected to another latent variable h_2; if h_1 is connected directly upstream of h_2 with a nonzero weighted edge in the latent hierarchy, it would suggest that the signal of i_1 is upstream of that of i_2 and that the protein affected by i_1 causally regulates the protein affected by i_2 in the cellular system. To examine this type of relationship, we visualized the relationships among the SGAs from four TCGA pathways via the latent nodes they are connected to, as a means to search for causal structure revealed by RINN models (Figure 16). Overall, the learned causal graphs were much more complex and dense than the graphs from the pan-cancer TCGA analysis (Figure 11) in that an SGA in the RINN was often connected to multiple latent variables (depending on the cutoff threshold). This suggests that there is still difficulty in directly interpreting the weights of a RINN as causal relationships with the current form of the algorithm. It may also suggest that the ground truth we used here is not detailed enough for our purposes, or may have some inaccuracies, and thus is more of a "silver standard". In these graphs there are a large number of false positive edges and redundant causal edges—where a single SGA will act multiple times on the same path.

However, some true causal relationships are seen within these graphs. For example, *both* models 0 and 1 have a directed path as follows: $i_{KEAP1} \rightarrow h_1 \rightarrow h_2 \leftarrow i_{NFE2L2}$, where i represents an intervention (i.e., SGA). If we assume that h_1 is a representation of the KEAP1 protein and h_2 is a representation of NFE2L2 protein, then we can interpret the above path as, KEAP1 $\rightarrow$ NFE2L2, i.e., KEAP1 causing NFE2L2. This is consistent with the NRF2 pathway shown in Figure 11. Model 2 has the following directed path: KEAP1 $\rightarrow h_1 \leftarrow$ NFE2L2, which has a more ambiguous interpretation but does emphasize the correlation

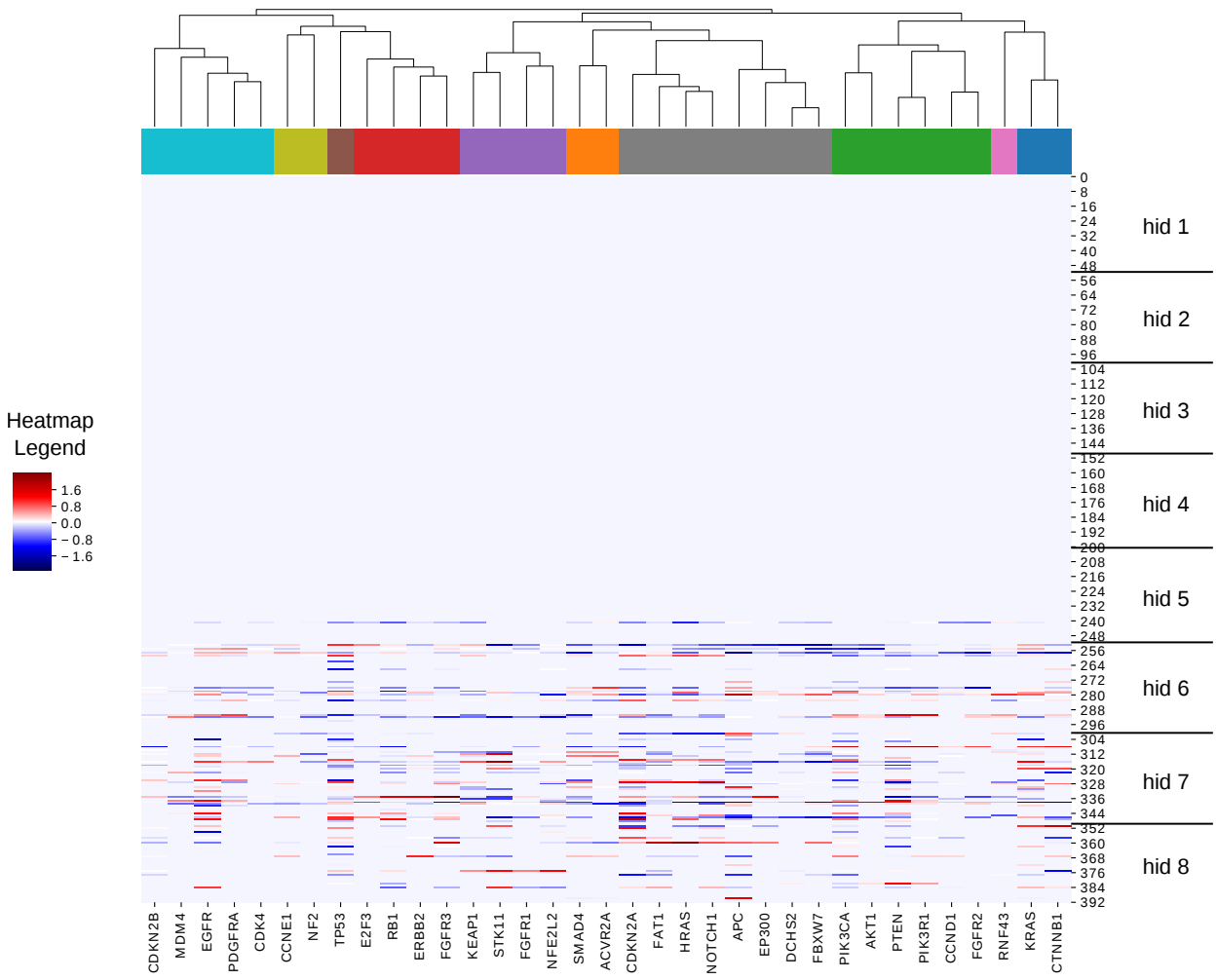


Figure 15: Hierarchical Clustering of RINN with Shortest Euclidean Distance

Heatmap shows all SGA to hidden node weight values for our best trained RINN model. The vertical axis represents hidden node number (numbered in ascending order starting from hidden layer 1—closest to SGAs) and has horizontal lines representing different hidden layer boundaries. The horizontal axis represents the 35 SGAs from Figure 11. The dendrogram and coloring on top show the clustering of the SGAs for one particular clustering cutoff.

between these two SGAs. Also, it is likely that if the weight threshold was decreased for Model 2, there would be an edge between the green hidden node and the blue hidden node, meaning that all three models would then capture the KEAP1 $\rightarrow$ NFE2L2 causal relationship. This emphasizes the importance of finding a more robust means of thresholding for future work.

All the RINNs represented in Figure 16 (and Figure 15) only utilized three or four hidden layers to learn the function mapping SGAs to DEGs. Because we used L_1 regularization, RINNs learned to only use the number of hidden layers necessary to perform well on the prediction task. For example, if only three hidden layers were needed to perform well on prediction, the regularized objective function would set all the weights before the last three hidden layers to 0.0. We set the number of hidden layers to eight (Section 6.3.2) as we hypothesized that eight hidden layers was enough hierarchical complexity to capture cancer cellular signalling pathways in a meaningful way. Figures 15 and 16 suggest that this was a correct assumption in this setting for the data we used. As with other model selection techniques, if our trained RINNs had utilized all eight layers during model selection, we could easily increase the number of hidden layers to more then eight and perform another round of model selection.

6.4.5 Shared Hidden Nodes Across Top Ten Models

As shown in Figure 16, SGAs perturbing members of a common pathway are closely connected to a similar set of hidden variables, and often many SGAs are directly connected to a common hidden node. This suggests that such hidden nodes encode a common signal that is shared by the SGAs. We hypothesize that if SGAs from a pathway are connected to a similar set of shared hidden nodes in multiple different RINN models, this would indicate that the RINN can repeatedly detect the common impact of the SGAs, and therefore use a common set of hidden nodes to encode their impact with respect to DEGs. In other words, the RINN consistently encodes the shared functional impact of SGAs perturbing a common pathway and the function of these hidden nodes in a specific RINN model become partially transparent. We labeled hidden nodes based on their SGA ancestors (Section 6.3.10) and

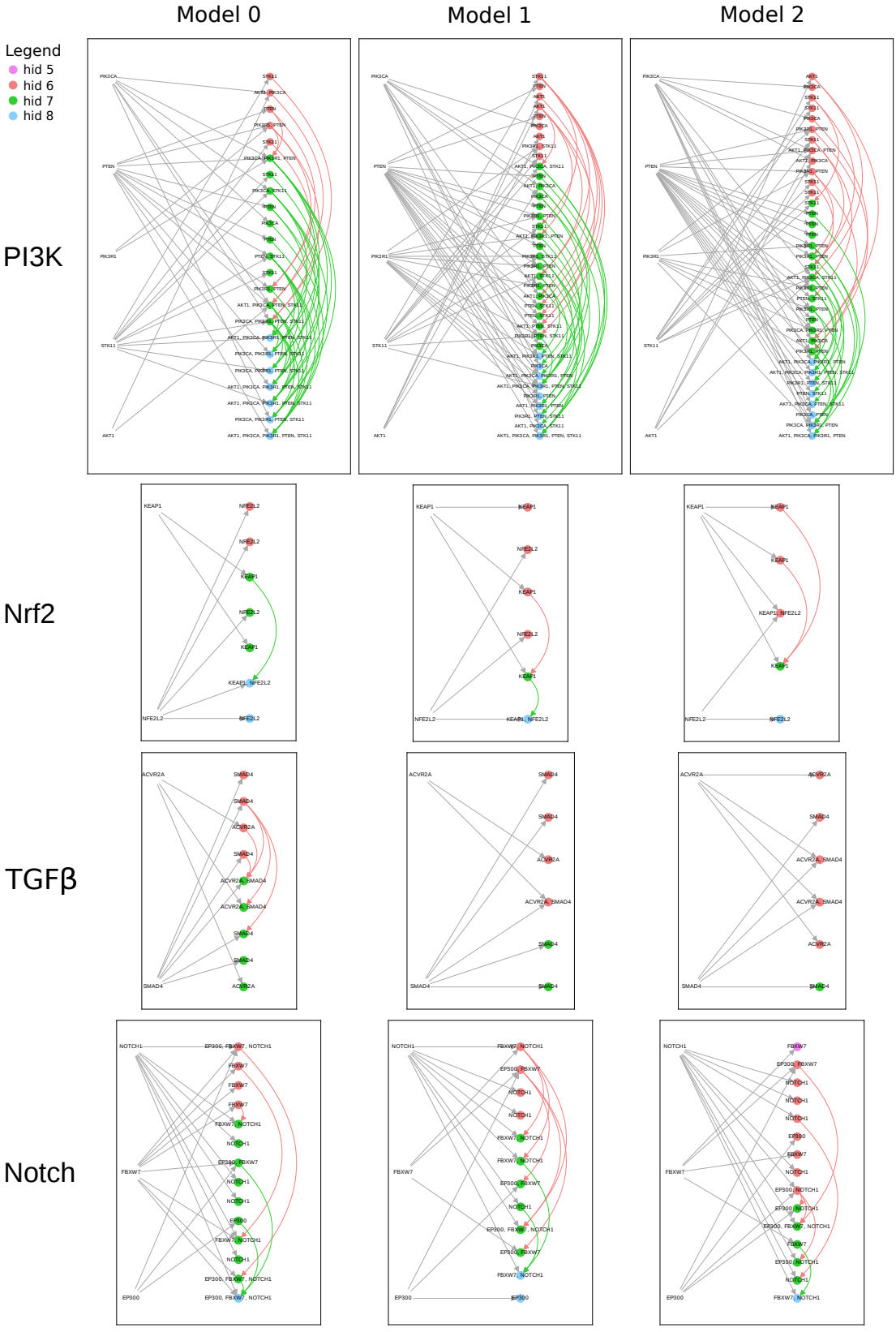


Figure 16: Visualizing the Weights of a RINN as Causal Graphs

Each graph represents the causal relationships for only the SGAs on the left of the graph.

then examined whether such hidden nodes were conserved across models (Table 9). Indeed, many hidden nodes were conserved across the top ten models. More specifically, Table 9 shows the number of models that shared a specific hidden node labeling and whether or not this would be expected by chance. For example, a hidden node mapping to all five members of the PI3K pathway, {AKT1, PIK3CA, PIK3R1, PTEN, STK11}, was found in all of our top ten models. Given five random SGAs, the expected number of the top ten models to share a hidden node mapped to all five random SGAs was 0.0 models (see Section 6.3.10 for our random control methodology). This means that in our 30 replicates of the random control with five random SGAs, none of the top ten models ever shared a hidden node mapped to all five random SGAs. This also means that the result above for all five members of the PI3K pathway is a very strong result and a definite pathway relationship discovered with the RINN setup in this work. Many of the 4-SGA and 3-SGA labeled hidden nodes were also found in many more models than the corresponding random control, such as the {AKT1, PIK3CA, PTEN, STK11}, {PIK3CA, PIK3R1, PTEN, STK11}, {AKT1, PIK3CA, STK11}, {PIK3R1, PTEN, STK11}, and {PIK3CA, PIK3R1, PTEN} labeled hidden nodes that were found.

A hidden node mapped to both members of the Nrf2 and TGFβ pathways was also found in all ten models, which was much higher than the number of models expected given two random SGAs. Given random SGAs, only 0.7 ± 1.2 models of the top ten would be expected to share the same 2-SGA labeled hidden node. Also, a hidden node mapped to all three members of the Notch pathway was shared by nine of the top ten models, which was also well beyond the number of models expected given random SGAs.

6.5 Discussion

In this chapter, we show that deep learning models, RINN and DNN, can capture the statistical relationships between genomic alterations and transcriptomic events in tumor cells with reasonably high accuracy, despite our relatively small number of training cases and the high dimensionality of the data. Our findings further indicate that a regularized deep neural network with redundant inputs (i.e., the RINN model) can capture cancer signaling

Table 9: Cancer Pathway Hidden Nodes Shared Across Top Ten Models

Hidden Node x Mapping represents SGAs that directly act on hidden node x or are ancestors of x. Models are numbered 0 to 9. (n: number of SGAs mapping to a hidden node, m: number of SGAs in a pathway).

Pathway	Hidden Node x Mapping	Models (out of 10; numbered 0 to 9) that Share Hidden Node x	Expected Number of Models that Share an n-SGA Hidden Node Given m Random SGAs
PI3K	AKT1, PIK3CA, PIK3R1, PTEN, STK11	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	0.0 ± 0.0 ($n = 5$, $m = 5$)
	AKT1, PIK3CA, PTEN, STK11	0, 2, 5, 6, 7, 9	0.4 ± 1.4 ($n = 4$, $m = 5$)
	PIK3CA, PIK3R1, PTEN, STK11	0, 4, 5, 7, 8	0.4 ± 1.4 ($n = 4$, $m = 5$)
	AKT1, PIK3CA, PIK3R1, PTEN	1, 3, 6	0.4 ± 1.4 ($n = 4$, $m = 5$)
	AKT1, PIK3R1, PTEN, STK11	2, 3	0.4 ± 1.4 ($n = 4$, $m = 5$)
	AKT1, PIK3CA, STK11	1, 2, 3, 5, 6, 8	1.0 ± 1.4 ($n = 3$, $m = 5$)
	PIK3R1, PTEN, STK11	2, 3, 6, 7, 8, 9	1.0 ± 1.4 ($n = 3$, $m = 5$)
	PIK3CA, PIK3R1, PTEN	0, 2, 6, 7, 9	1.0 ± 1.4 ($n = 3$, $m = 5$)
	AKT1, PIK3R1, PTEN	3, 9	1.0 ± 1.4 ($n = 3$, $m = 5$)
	AKT1, PTEN, STK11	2, 3	1.0 ± 1.4 ($n = 3$, $m = 5$)
	PTEN, STK11	0, 2, 3, 5, 6, 7, 8	3.4 ± 2.0 ($n = 2$, $m = 5$)
	PIK3R1, PTEN	0, 2, 3, 6, 8, 9	3.4 ± 2.0 ($n = 2$, $m = 5$)
	AKT1, PIK3CA	0, 3, 4, 6, 8	3.4 ± 2.0 ($n = 2$, $m = 5$)
	PIK3CA, STK11	0, 1, 2, 4, 8	3.4 ± 2.0 ($n = 2$, $m = 5$)
	AKT1, STK11	1, 2, 3, 4, 5	3.4 ± 2.0 ($n = 2$, $m = 5$)
	PIK3CA, PTEN	5, 6, 8, 9	3.4 ± 2.0 ($n = 2$, $m = 5$)
	AKT1, PIK3R1	2, 5	3.4 ± 2.0 ($n = 2$, $m = 5$)
	PIK3R1, STK11	3, 9	3.4 ± 2.0 ($n = 2$, $m = 5$)
	AKT1, PTEN	5, 9	3.4 ± 2.0 ($n = 2$, $m = 5$)
Nrf2	KEAP1, NFE2L2	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	0.7 ± 1.2 ($n = 2$, $m = 2$)
TGFβ	ACVR2A, SMAD4	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	0.7 ± 1.2 ($n = 2$, $m = 2$)
Notch	EP300, FBXW7, NOTCH1	0, 1, 2, 3, 4, 5, 6, 7, 9	0.2 ± 0.6 ($n = 3$, $m = 3$)
	FBXW7, NOTCH1	0, 2, 3, 6, 7, 8, 9	2.1 ± 2.2 ($n = 2$, $m = 3$)
	EP300, FBXW7	0, 2, 3, 4, 6	2.1 ± 2.2 ($n = 2$, $m = 3$)

pathway relationships within its hidden variables and weights. The RINN model correctly captured much of the functional similarity between SGAs that perturb a common signaling pathway as reflected by their similar interactions with the hidden nodes of the RINN model (i.e., cosine similarity of SGA weight signatures). This shows that SGAs on the same pathway have interactions with a similar set of latent variables. These are very encouraging results for eventually using a future version of the RINN to find robust signaling pathways. Many of the most well known cancer driver genes (EGFR, TP53, CDKN2A, APC, and PIK3CA) were found to have dense SGA weight signatures and weights with larger values relative to the other genes we analyzed, reinforcing the validity of our models and the importance of these genes in driving cancer gene expression. Our results indicate that RINN consistently employs certain hidden nodes to represent the shared functional impact of SGAs perturbing a common pathway, although different instantiations of RINN could use totally different hidden nodes. The capability of RINN to explicitly connect SGAs and hidden nodes throughout the latent hierarchy essentially makes it a partially transparent deep learning model, so that one can interpret which hidden nodes encode and transmit signals (i.e., functional impact) of SGAs in cancer cells. Finally, we show that RINN is capable of capturing some causal relationships (given our interpretation of the hidden nodes) between the signaling proteins perturbed by SGAs. All these results indicate that by allowing SGAs to directly interact with hidden nodes in a deep learning model, the RINN model provides constraint and information to enable certain hidden nodes to encode the impact of specific SGAs.

The similarities between the cosine similarity tables for RINN and DNN are quite remarkable and overall show that both RINN and DNN are capturing similar information in their weights. This is a somewhat expected result, as both RINN and DNN were trained on the same data to perform the same task and had similar AUROCs, meaning they both needed to capture relevant cancer pathway information in their weights in order to perform well on the prediction task. Although, in a DNN the information is likely to be more convoluted. The redundant inputs of the RINN are an attempt to deconvolute the signal into relationships that are more easily interpreted by a human, by giving the network more choices for when to learn certain information (i.e., early in the network at the first hidden layer or sometime later in the network). The hidden layers of a DNN are alternate representations of all the

information in the input necessary to calculate the output, whereas each hidden layer of a RINN does not need to capture all the information in the input necessary to predict the output because there are multiple chances to learn what is needed from the input (i.e., redundant inputs). This difference gives the RINN more freedom in how it chooses to use the information in the input. The last hidden layer of the RINN is the only hidden layer that must contain all relevant information from the input (the last hidden layer also has information from the previous hidden layers). The redundant inputs of the RINN attempt to order the information necessary for the prediction task in a particular way so that more human-recognizable pathway relationships can be generated. In contrast, the DNN must order or combine the information in any convoluted way that best predicts the output, while still constraining *each* hidden layer to represent and propagate all the necessary information in the input required for the prediction task, as a DNN never has access to the input again beyond the first hidden layer.

A DNN cannot capture the same causal relationships that a RINN can. By nature of its architecture and design, a DNN can capture *direct* causal relationships (i.e., edges starting from an observed variable) between the input and the first hidden layer only—DNNs cannot capture direct causal relationships between the input and any other hidden layers. *This means that a DNN cannot be used to generate causal graphs like the ones shown in Figure 16*. If a DNN were used to generate causal graphs in the same format as Figure 16, the graphs would have left-sided inputs connected to a single hidden layer and then cascading hidden nodes and edges (between hidden nodes in different hidden layers only) for multiple hidden layers on the right side of the figure. There would not be any edges from the input on the left side to any hidden layers other then the first hidden layer. *In addition, a DNN cannot capture causal relationships among input SGAs as was described in Section 6.4.4, meaning that one cannot infer the causal relationships among SGAs with a DNN*. This is because DNNs do not have edges between hidden nodes in the same layer. Consider $path_{KEAP1}$ and $path_{NFE2L2}$ as the paths with KEAP1 or NFE2L2 as the source node, respectively, in a DNN. In a DNN, to determine that there is a dependency between KEAP1 and NFE2L2, eventually these two paths would need to collide on a hidden node. When these two paths collide on a hidden node, the number of edges in each path will be the same, meaning that the direction

of the causal relationship is ambiguous. This limitation of a DNN can be remedied by adding redundant inputs (i.e., RINN) or by adding edges between nodes in the same hidden layer in a DNN. Using the RINN architecture allows us to infer order to the causal relationships among SGAs and allows SGAs to directly act on the latent hierarchy, similar to how SGAs in vivo act on signaling pathways. The RINN can capture direct relationships between the input and any hidden layer in the network, allowing one to also infer causal relationships among the input variables—this design difference and extension of causal interpretability are what set it apart from a DNN.

It is intriguing to examine further whether the hierarchy of hidden nodes can capture causal relationships among the signals encoded by SGA-affected proteins. We have shown that many of the pathway relationships and some known causal relationships were present in the hierarchy reflected by the weight matrices of our trained models. However, we also noticed that in our RINN models an SGA is often connected to a large number of hidden nodes, which are in turn connected to a large number of other hidden nodes—meaning that the current RINN model learns relatively dense causal graphs. While one can infer the relationships between the signal perturbed by distinct SGAs of a pathway, our current model cannot directly output a causal network that looks like ones commonly shown in the literature. We plan to develop RINN into an algorithm that is able to find more easily interpretable cellular signaling pathways when trained on SGA and DEG data. The following algorithm modifications potentially will lead to better results in the future: 1. Incorporating differential regularization of the weights, 2. Using constrained and parallelized versions of evolutionary algorithms to optimize the weights and avoid the need to threshold weights, and 3. Training an autoencoder with a bottleneck layer to encourage hidden nodes to more easily represent biological entities and then using these weights (and architecture) to initialize a RINN.

An assumption implicit in this work in order to interpret the weights of a neural network as causal relationships between biological entities is that we are assuming that the causal relationships between biological entities can be approximated by a linear function combined with a simple nonlinear function (e.g., $activation(wx + b)$, where all variables have scalar values and *activation* represents a simple nonlinear function such as ReLU or softplus). This is a necessary assumption in order to interpret all nonzero hidden nodes as biological entities;

however, it could also be the case that some hidden nodes are not biological entities, but rather some intermediate calculation required to compute a relationship between biological entities that cannot be modeled with $activation(wx + b)$. Given the high density of the models learned with TCGA data, it is possible that the relationships between some biological entities cannot be modeled with $activation(wx + b)$, suggesting that more complex activation functions are needed or that biological entities may be present at every other hidden layer. It would be interesting to explore using more complex activation functions, and specifically using an unregularized one hidden layer neural network as an activation function for each hidden node in a RINN. This setup would account for even quite complex relationships between biological entities captured as latent variables. See Section 5.5 for additional discussion of this topic.

A cellular signaling system is a complex information encoding/processing machine, which processes signals arising from extrinsic environment changes or perturbations (genetic or pharmacological) affecting the intrinsic state of a cell. The relationships of cellular signals are hierarchical and nonlinear by nature, and deep learning models are particularly suitable to modeling such a system [Chen et al., 2015, Chen et al., 2016, Young et al., 2017, Lu et al., 2018, Tao et al., 2020]. However, conventional deep learning models behave like "black boxes", such that it is almost impossible to determine what signal a hidden node encodes, with few exceptions in image analysis where human interpretable image patterns can be represented by hidden nodes [Lee et al., 2008, Lee et al., 2011, Le et al., 2011]. Here, we took advantage of our knowledge of cancer biology that SGAs causally influence the transcriptomic programs of cells and we adopted a new approach that allows SGAs to directly interact with hidden nodes in RINN. We conjecture that this approach forces hidden nodes to explicitly and more effectively encode the impact of SGAs on transcriptomic systems. This hypothesis is supported by the discoveries of this chapter that SGAs in a common pathway share similar connection patterns to hidden nodes, and that there are hidden nodes that are connected with multiple members of a pathway in different instances of the model. Essentially, our approach allows certain hidden nodes to be "labeled" and "partially interpretable". An interpretable deep learning model provides a unique opportunity to study how cellular signals are encoded and perturbed under pathological conditions. Understanding and representing

the state of a cellular system further opens directions for translational applications of such information, such as predicting the drug sensitivity of cancer cells based on the states of their signaling systems. To our knowledge, this is the first time that a partially interpretable deep learning model has been developed and applied to study cancer signaling, and we anticipate this approach laying a foundation for developing future explainable deep learning models in this domain.

7.0 Conclusions and Future Directions

In conclusion, a modified deep learning model (RINN with L_1 regularization) can be used to find causal relationships within the learned weights of the network when trained on simulated data. When trained on cancer genomic data, genomic alterations impacting a common cancer signaling pathway (PI3K, NRF2, and TGFβ) had similar interactions (i.e., connections) with the RINN hidden hierarchical structure, meaning they interacted with a similar set of hidden nodes. This similarity of interactions with the latent hierarchy suggests that similar genomic alterations found in this dissertation have related biological functions. In this setting, having genomic alteration input variables directly connected to all latent variables allowed us to label the latent variables with a set of genomic alterations, making the latent variables partially interpretable. With this labeling, we were able to visualize the weights of RINNs as causal graphs and capture at least some of the causal relationships in known cancer signaling pathways. In general, however, the causal graphs learned by RINN were somewhat too dense (despite large amounts of regularization) to compare directly to the known cancer signaling pathways in Figure 11. A prerequisite to interpreting hidden nodes or finding causal graphs with a RINN is robust training of an accurate classifier. Importantly, we showed that differentially expressed genes can be predicted from somatic genome alterations by a RINN (and DNN) with reasonably high AUROCs, especially after considering the very high dimensionality of the prediction task relative to the number of input variables and instances in the dataset. This is a very encouraging result for the future of deep learning models trained on cancer genomic data. We were able to achieve notable results with a relatively small dataset and very high dimensionality—imagine what might be possible with two times to ten times more cancer data.

In this dissertation, we also developed a causal framework for the RINN model describing the causal assumptions that are necessary to use the RINN for causal discovery. We created a robust pipeline for testing an algorithm's ability to capture latent causal structure and developed a variety of simulated data inspired by cellular signaling pathways that might be useful to others interested in discovering latent causal relationships. In addition, we created

a causal ground truth DAG with simple patterns in the hidden nodes to enable quick visual verification of these patterns. Related to this, we developed a simple algorithm to visualize the weights of a small neural network, or part of a larger neural network, using the patterns that are known to be present in simulated data. This visualization algorithm allowed us to quickly verify whether or not a neural network-based algorithm was capturing any of the ground truth causal structure in its weights. This visualization technique inspired a method for calculating precision and recall of the causal structure learned by any algorithm, especially neural network-based algorithms.

7.1 Future Directions

The weights of a RINN, after training on TCGA data, are not quite ready to be directly interpreted as causal relationships, but with some modifications to our algorithm, future versions could yield promising causal graphical models. Listed below are future research directions for the RINN algorithm that address many of the limitations discussed in Sections 5.5 and 6.5.

1. Incorporating differential regularization of the weights.
2. Using constrained and parallelized versions of evolutionary algorithms to optimize the weights and avoid the need to threshold weights.
3. Training an autoencoder with a bottleneck layer to encourage hidden nodes to more easily represent biological entities and then using these weights to initialize a RINN.
4. Directly build a neural network causal graph using neuroevolution.

We describe each of these research directions in more detail here. Despite large amounts of regularization and at times over-regularizing the RINN, the causal graphs learned by the RINN were still somewhat too dense. By using regularization with the RINN, our goal was to find an ideal balance where we weren't overfitting or underfitting the data. Over-regularizing leads to underfitting the data and under-regularizing leads to overfitting the data—both of which lead to poor quality causal graphs with the RINN. The current version of the RINN

only has a single regularization rate in the objective function applied to the L_1 regularization term. In this work, we think we found an ideal balance between sparsity and prediction error with our Euclidean distance measure. However, the models returned were still too dense, suggesting that the data is inadequate for the task, the true model is actually quite dense, or that a more flexible regularization regime is needed. To explore this, we plan to incorporate differential regularization into the RINN model. One way to accomplish this is to have different regularization rates for the weights connecting input variables to hidden variables, and the weights connecting hidden variables to hidden variables or hidden variables to output variables. This would increase the amount of model selection needed, but may generate a better causal graph by allowing for a more biologically correct model. SGAs act on single proteins or at one location in a cellular signalling pathway. Differential regularization as described above may allow the RINN to learn a higher regularization rate on the input variable to hidden variable weights, to enforce this known biological constraint. It would also be interesting to explore regularization techniques that place a hard constraint on each SGA, by only allowing each SGA to have a single nonzero weight.

In Chapter 5, we showed that a RINN optimized with a constrained evolutionary strategy (ES-C) performed moderately well at recovering latent causal structure when compared to a conventional RINN and DNN. ES-C performed well on simulated data despite being trained on significantly less data and only performing somewhat limited model selection as compared to the other strategies (ES-C has more hyperparameters to tune than RINN due to the evolutionary strategies optimization). ES-C has a great advantage relative to the other strategies when it comes to interpretation of the learned model. Because ES-C limits the possible values of the weights, it is much easier to determine a threshold for weight cutoffs that are meaningful. Also, the causal graphs that are generated by a RINN optimized with ES-C would be less dense than a RINN trained with gradient descent, which simplifies the interpretability of the causal model.

The benefits of using ES-C are quite significant for the tasks discussed in this dissertation. The long running times of evolutionary algorithms (and specifically ES-C) can be greatly reduced through parallelization, as the population of individuals can be evolved mostly independently of other individuals. We suspect that a parallelized version of a RINN

optimized with ES-C and trained on the full dataset with as much model selection (or more) than the other strategies would lead to results that are superior to what has been shown here. Also, we have only experimented with a couple different sets of legal weight values for ES-C. A more comprehensive search through this space is warranted. We hope to find a python evolutionary algorithm library that allows for simple parallelization for ES-C or develop the parallelization ourselves in-house. We think that using parallelized constrained evolutionary strategies to optimize RINN is a very promising direction for further research.

As previously mentioned in this dissertation (Section 2.2) and shown in Chapter 3, a DBN (or autoencoder) can learn hidden layer representations that are biologically meaningful, and in some instances can be trained so that some of the hidden nodes represent transcription factors [Chen et al., 2016]. Taking inspiration from this and from the hypothesis described in Figure 1, we plan to combine a trained autoencoder with the RINN. Training a DBN or other type of autoencoder (e.g., variational autoencoder) on TCGA gene expression data in a similar manner as described in Chapter 3 would encourage the DBN to learn hidden layer representations that we hypothesize would represent components of signaling pathways. Once this DBN was completely trained after extensive model selection, we would then use the "decoder" part of the DBN (see Section 4.5.2) to initialize a RINN of the same architecture, but would then switch to a supervised task using genomic alterations to predict differentially expressed genes.

We would need to learn a new matrix of weights from the genomic alterations to the first hidden layer (i.e., the smallest hidden layer), as well as the weights between redundant inputs and hidden layers. But we could use different learning rates for the different subsets of weights in the DBN plus RINN model, possibly using a lower learning rate for the weights learned by the DBN and using a higher learning rate for the new sets of weights added by the RINN. We hypothesize that this setup would encourage the hidden nodes to represent parts of a cellular signaling system [Chen et al., 2015, Chen et al., 2016, Young et al., 2017, Lu et al., 2018, Tao et al., 2020], while the RINN components of the model would encourage the learning of causal relationships between biological entities in the hidden layers. Overall, this idea can be described as using an autoencoder with a bottleneck layer to initialize a supervised RINN, and we think this approach may yield more accurate causal graphs than

shown here with a simpler version of the RINN.

One of the most interesting directions for further development of the RINN is to combine the RINN with neuroevolution, especially the Neuroevolution of Augmenting Topologies (NEAT) algorithm [Stanley and Miikkulainen, 2002]. Neuroevolution encompasses the many ways to evolve a neural network—including using evolutionary algorithms to optimize the weights of the network (evolutionary strategies) or using evolutionary algorithms to learn the topology/architecture of the network (e.g., NEAT algorithm). Considering their massive computation requirements, evolutionary algorithms have been historically limited to low dimensional spaces. However, with the cost of CPUs decreasing and the availability of CPUs increasing, evolutionary algorithms are starting to become competitive, and sometimes even outperform, deep learning algorithms in higher dimensional spaces [Wilson et al., 2018, Such et al., 2017, Salimans et al., 2017, Conti et al., 2018, Miikkulainen et al., 2019]. This is leading to an increase in interest in evolutionary algorithms and neuroevolution from the deep learning community.

NEAT is an algorithm the evolves the topology of a neural network using evolutionary computation [Stanley and Miikkulainen, 2002]. It starts with a population of very simple topologies and, through mutation operators, adds complexity to the topologies (edges, nodes, and eventually hidden layers) as the algorithm progresses. The topologies with the highest fitness (based on some measure of fitness, often the objective function in deep learning applications) reproduce (mating and recombination of topologies) to create new progeny. This is one of the simplest versions of the NEAT algorithm. There are many other neuroevolution techniques to add to the algorithm described above, such as speciation or novelty search [Conti et al., 2018].

With NEAT, one starts with input and output variables, then using evolutionary algorithms, starts to fill in hidden nodes, bias nodes, and edges between all nodes. Also, activation functions for each hidden node can be evolved. This is a flexible method to generate the latent structure between inputs and outputs using neural network-based calculations. NEAT may even be able to add cycles into the topology. To optimize the parameters of the fitness function (i.e., objective function) in neuroevolution applications, backpropagation and stochastic gradient descent can be used as in deep learning, or evolutionary strategies can

be used to search for the best set of weights for the network. Evolutionary strategies allow extreme flexibility when choosing a fitness function, as the function need not be differentiable. Therefore, one can choose a fitness function that is completely procedural as defined by some algorithm.

We suggest that using NEAT to gradually build up a RINN-like architecture using our simulated data first would be a very interesting future direction for the RINN model, and possibly a better method for learning a causal graph than the RINN trained in this dissertation. The RINN could then be optimized using stochastic gradient descent or constrained evolutionary strategies. After training the NEAT algorithm on the simulated data, we would then interpret the architecture and weights learned by NEAT as representing causal relationships and calculate precision and recall as before. Using NEAT allows us to directly build and evolve the causal structure within a neural network rather than using fully-connected hidden layers and waiting for L_1 regularization to prune unnecessary weights.

We are hopeful that modifying the RINN algorithm as described above will allow the RINN to eventually be used to model cancer cellular signaling pathways at the level of the individual patient, and look forward to the great impact that this could have on novel cancer therapies.

Appendix

Unsupervised Deep Learning Applied to Cancer Data

The Author(s) *BMC Bioinformatics* 2017, **18**(Suppl 11):381
DOI 10.1186/s12859-017-1798-2

BMC Bioinformatics

RESEARCH Open Access

Unsupervised deep learning reveals prognostically relevant subtypes of glioblastoma

Jonathan D. Young[1,2*], Chunhui Cai[1,3] and Xinghua Lu[1,3]

From The International Conference on Intelligent Biology and Medicine (ICIBM) 2016
Houston, TX, USA. 08-10 December 2016

Abstract

Background: One approach to improving the personalized treatment of cancer is to understand the cellular signaling transduction pathways that cause cancer at the level of the individual patient. In this study, we used unsupervised deep learning to learn the hierarchical structure within cancer gene expression data. Deep learning is a group of machine learning algorithms that use multiple layers of hidden units to capture hierarchically related, alternative representations of the input data. We hypothesize that this hierarchical structure learned by deep learning will be related to the cellular signaling system.

Results: Robust deep learning model selection identified a network architecture that is biologically plausible. Our model selection results indicated that the 1st hidden layer of our deep learning model should contain about 1300 hidden units to most effectively capture the covariance structure of the input data. This agrees with the estimated number of human transcription factors, which is approximately 1400. This result lends support to our hypothesis that the 1st hidden layer of a deep learning model trained on gene expression data may represent signals related to transcription factor activation. Using the 3rd hidden layer representation of each tumor as learned by our unsupervised deep learning model, we performed consensus clustering on all tumor samples—leading to the discovery of clusters of glioblastoma multiforme with differential survival. One of these clusters contained all of the glioblastoma samples with G-CIMP, a known methylation phenotype driven by the *IDH1* mutation and associated with favorable prognosis, suggesting that the hidden units in the 3rd hidden layer representations captured a methylation signal without explicitly using methylation data as input. We also found differentially expressed genes and well-known mutations (*NF1*, *IDH1*, *EGFR*) that were uniquely correlated with each of these clusters. Exploring these unique genes and mutations will allow us to further investigate the disease mechanisms underlying each of these clusters.

Conclusions: In summary, we show that a deep learning model can be trained to represent biologically and clinically meaningful abstractions of cancer gene expression data. Understanding what additional relationships these hidden layer abstractions have with the cancer cellular signaling system could have a significant impact on the understanding and treatment of cancer.

Keywords: Deep learning, Unsupervised learning, Cancer, Glioblastoma multiforme, Deep belief network, Gene expression, Model selection

* Correspondence: jdy10@pitt.edu
[1]Department of Biomedical Informatics, University of Pittsburgh, 5607 Baum Blvd, Pittsburgh, PA 15206, USA
[2]Intelligent Systems Program, University of Pittsburgh, 5607 Baum Blvd, Pittsburgh, PA 15206, USA
Full list of author information is available at the end of the article

Background

Understanding the cellular signal transduction pathways that drive cells to become cancerous is fundamental to developing personalized cancer therapies that decrease the morbidity and mortality of cancer. Most research studying cellular signaling pathways has concentrated on a handful of signaling proteins in a hypothesis-driven framework. This study uses a deep learning approach to study cancer signaling systems in a large-scale data-driven fashion, with an overall goal of understanding the cellular signaling pathways that drive or cause cancer. Towards this goal, we used unsupervised deep learning to find meaningful structure and relationships in cancer gene expression data.

Deep learning models (DLMs) originated from artificial neural networks (ANN) and learn alternate representations of the original input data. A DLM is composed of multiple layers of latent variables (hidden nodes or units, in the ANN jargon) [1–5], which learn to represent the complex statistical structures embedded in the data, such that different hidden layers capture statistical structure of different degrees of complexity. In other words, DLMs learn novel representations of the statistical structure of the input data through hierarchical decomposition. For example, if one trains a convolutional neural network (a type of DLM) with three hidden layers on a dataset of face images (in order to learn how to recognize specific people in images), the units in these three layers capture abstract representations at different levels. The model may use the 1st hidden layer units (the layer that is closest to input data) of the trained network to capture edges of different orientations present in the original image [6, 7]. The 2nd hidden layer units may learn representations of different parts of a face [7] (e.g., nose, mouth, eye, ear), by combining edges of different orientations (represented by the units in the 1st hidden layer). Finally, units in the 3rd hidden layer may represent generic faces [4, 7], which can be thought of, or represented as, combinations of parts of a face. In this way, deep learning finds hierarchical structure in data by finding alternate representations (often of lower dimension) that best encode the information in the original image. Once a DLM is trained, it can be used to detect a specific person in an image or, depending on the type of model, it can be used to generate new face images that mimic the distribution of the input images. In this study, we aim to use DLMs to find the hidden layer representations of cancer gene expression data that likely represent the state of the signaling systems in cancer cells.

More specifically, we hypothesize that the activation states of signaling pathways regulating transcriptomic activities in cancer cells can be learned using DLMs (Fig. 1). Cancer is a multi-process disease, in that a cancer cell usually has multiple aberrant pathways—each pathway consists of a set of hierarchically organized signaling molecules (Fig. 1a)—driving differentially expressed genes (DEGs) that are involved in multiple oncogenic processes. A transcriptomic profile of a tumor is a convoluted mixture of expressed genes regulated by active pathways in tumor cells, but the

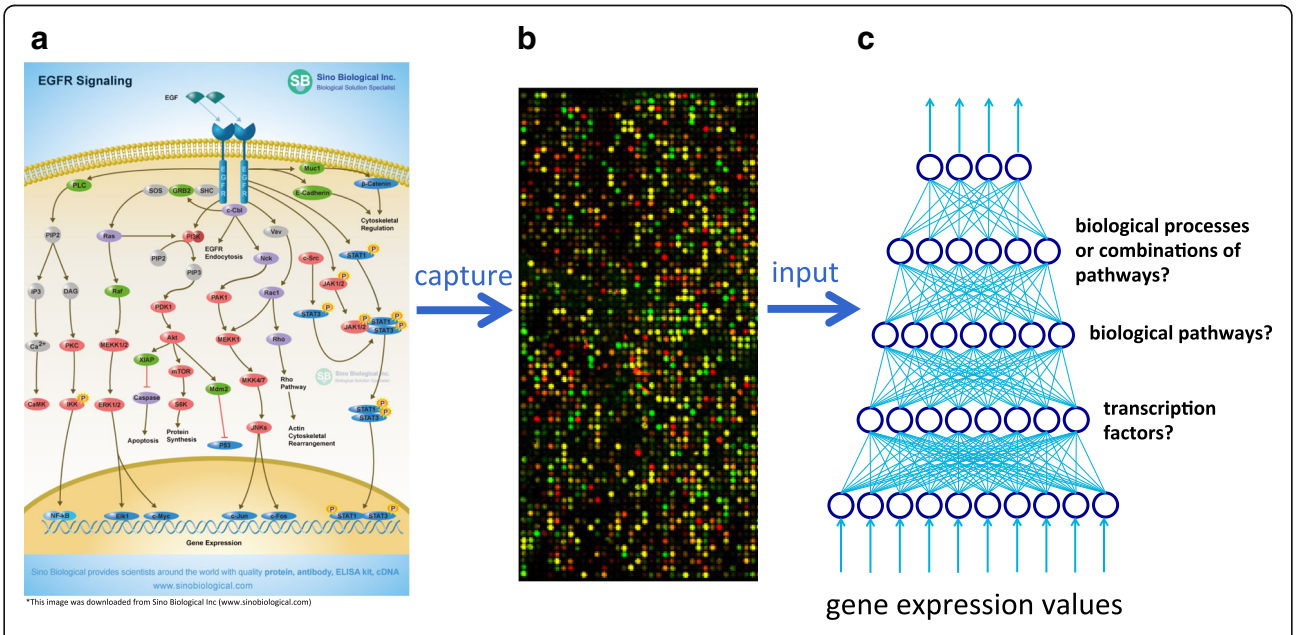


Fig. 1 Inferring the activation states of signaling pathways with deep learning. (**a**) Example EGFR signaling cascade. (**b**) A microarray representing gene expression values. (**c**) Representation of a DBN (fine-tuning encoder network only) trained on gene expression values. Hypothesized biological entities whose activation states may be represented by units in the 1st, 2nd, and 3rd hidden layers are displayed on the *right*

information related to the hierarchical organization of pathways is "collapsed" and distinct signals from different pathways become inseparable (Fig. 1b). Discovering aberrant signals in cancer cells requires de-convolution (decomposition) of such signals, and DLMs are particularly well suited for such a task due to their ability to perform hierarchical decomposition. For example, the hierarchical structure of a signaling pathway (Fig. 1a) could be simulated by the hierarchical structure in a DLM trained on gene expression data (Fig. 1c). Since transcription factor (TF) activation dictates the finest covariance structure of gene expression, which is at the bottom of the signaling pathway in Fig. 1a, the 1st-hidden layer in our DLM (Fig. 1c) may capture the signals encoded by TFs. And just as there are different pathways being activated that regulate transcription factor activation in the middle of Fig. 1a, the 2nd hidden layer of our DLM may represent the activation states of different biological pathways. Continuing with this analogy, units in the 3rd hidden layer could represent biological processes (e.g., inflammation or metastasis) or combinations of pathways. In this way, we aim to learn the hierarchical representation of the signals underlying cancer gene expression data with deep learning.

Recently, we demonstrated that DLMs could be used to learn and represent the cellular signaling system [8, 9]. In one study, we showed that a DLM, more specifically a multi-modal deep belief network (DBN) and a semi-restricted multi-modal DBN can learn representations of the cellular signaling system shared by rat and human cells [9]. In the same study, we also demonstrated that a trans-species DLM could accurately predict human cell responses to a set of unknown stimuli based on data from rat cells treated with the same stimuli. In a more recent study, we showed that DLMs can learn representations of the yeast transcriptomic system, and that the hidden units of these DLMs can be mapped to real biological entities, such as transcription factors and well-known yeast signaling pathways [8]. Another group, Liang et al., integrated multiple types of genomic cancer data with a multi-modal DLM trained to regenerate the observed data (the entire omics profile), but their goal was not to infer aberrant signals, i.e., the differences in signaling between normal and cancer cells [10], as we did in this study.

In this study, we investigated the optimal architectures of DBN-based models to learn accurate representations of the signals underlying cancer transcriptomic changes across 17 cancer types. We show that a DLM can provide novel abstract representations, enabling us to reveal molecular subtypes of cancers, e.g., subtypes of glioblastoma multiforme (GBM), that exhibit significant differences in outcomes. Our analysis revealed different potential disease mechanisms (major driver genes) underlying these molecular subtypes.

Methods

Data

The data used in this study were obtained from The Cancer Genome Atlas (TCGA) Data Portal [11], and included transcriptomic data for 17 different cancer types and non-cancer organ-specific tissue controls, all downloaded from the TCGA Data Portal (Table 1). The total size of the dataset was 7528 samples by 15,404 genes.

We discretized the expression value of a gene in a tumor to 1 or 0 based on whether or not the expression value significantly deviated from the expression observed in normal tissue. To achieve this, we fit the expression values of each gene in each cancer type to a Gaussian distribution based on the non-cancer control samples only from the same tissue of origin. We then set the expression status of a gene in a tumor to 1 (aberrant) if it was outside the 0.001 percentile of distribution of control samples (on either side), otherwise we set it to 0. For genes with low expression variance in normal cells, i.e., standard deviation of expression smaller than 0.2, we used 3-fold change to determine whether the genes were differentially expressed in tumor cells. Through this discretization process, we identified genes that were potentially relevant to the cancer process (aberrantly expressed in cancer only) rather than just using the whole gene expression profile of a cell, which includes both physiological and pathological signals. The gene

Table 1 Number of samples for each cancer type in our dataset

Tissue Type	Number of Samples
Bladder urothelial carcinoma (BLCA)	403
Breast invasive carcinoma (BRCA)	1073
Esophageal carcinoma (ESCA)	183
Colon adenocarcinoma (COAD)	283
Glioblastoma multiforme (GBM)	481
Head and neck squamous cell carcinoma (HNSC)	508
Kidney renal clear cell carcinoma (KIRC)	525
Kidney renal papillary cell carcinoma (KIRP)	288
Liver hepatocellular carcinoma (LIHC)	364
Lung adenocarcinoma (LUAD)	509
Lung squamous cell carcinoma (LUSC)	498
Ovarian serous cystadenocarcinoma (OV)	559
Prostate adenocarcinoma (PRAD)	491
Rectum adenocarcinoma (READ)	93
Stomach adenocarcinoma (STAD)	236
Thyroid carcinoma (THCA)	499
Uterine corpus endometrial carcinoma (UCEC)	535
Total	7528

expression changes due to copy number alteration were also masked; as such changes are not regulated by the cellular signaling system, but are due to genomic alterations. In each tumor, we identified the genes that had expression changes and the genes that had copy number alterations, i.e., GISTIC2.0 [12] score equal to +1 (amplification) and GISTIC2.0 score equal to −1 (deletion). When we discovered gene expression up-regulation co-occurring with a corresponding copy number amplification, or a gene expression down-regulation co-occurring with a corresponding copy number deletion, we masked the gene expression change—as this co-occurrence suggested that the expression changes were caused by the DNA copy number alteration.

Preprocessing

Feature selection was performed to remove genes with low Bernoulli variance because of their general lack of information. We created datasets with different numbers of features by using different variance thresholds for feature selection. We identified genes that had an expression status of 1 (or 0) in 90% (Bernoulli success probability) of tumors and removed them from further analysis due to their low variance. This resulted in a dataset with 7160 features. We repeated this process using a Bernoulli success probability of 0.95 to create a dataset with 9476 features. We also removed any genes that were highly correlated with a specific cancer type or tissue type, by removing all genes with a Pearson correlation coefficient, with respect to cancer or tissue type labels, greater than 0.85.

Model specification

The specific deep learning strategy used for this study is called a DBN (layer-by-layer pre-training followed by "up-down" fine-tuning) [1, 13–15]. Although it may be clearer and more explicit to refer to the strategy used in this study as a stacked restricted Boltzmann machines–deep autoencoder (SRBM–DA), we will use the more traditional DBN terminology [1, 13–15] for the sake of being consistent with the literature. Learning of a DBN consists of two major phases: a pre-training phase and a fine-tuning phase (Fig. 2).

In the pre-training phase, the hierarchical DBN model is treated as multiple restricted Boltzmann machines (RBMs) stacked on top of each other, such that the top layer of an RBM is used as the bottom layer of another RBM above it. Learning of the parameters (often referred to as weights W) of the pre-training phase, starts with the learning of the weights of each of the stacked RBMs in a layer-by-layer fashion.

In more detail, the pre-training phase is a deep generative model consisting of a stack of RBMs. An RBM is a 2-layered undirected probabilistic network that seeks to

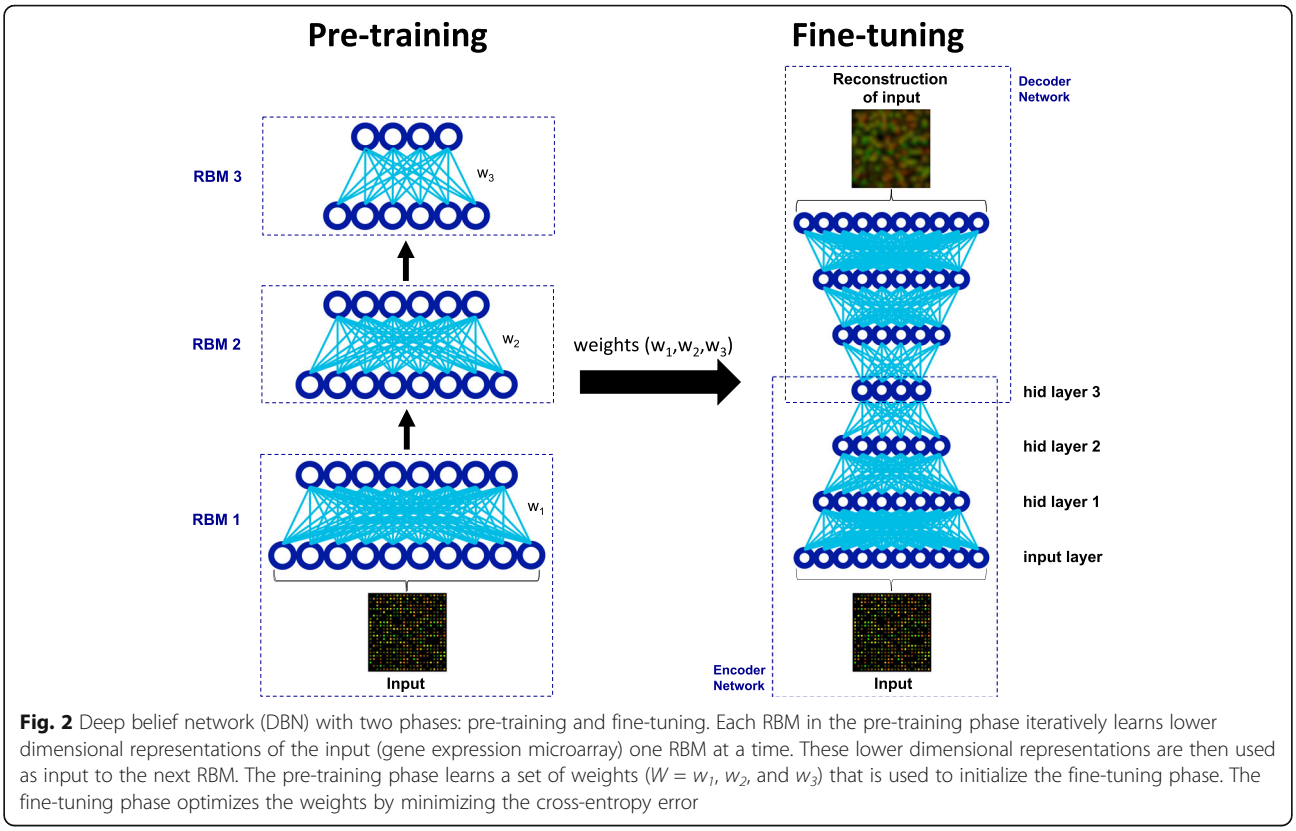


Fig. 2 Deep belief network (DBN) with two phases: pre-training and fine-tuning. Each RBM in the pre-training phase iteratively learns lower dimensional representations of the input (gene expression microarray) one RBM at a time. These lower dimensional representations are then used as input to the next RBM. The pre-training phase learns a set of weights ($W = w_1$, w_2, and w_3) that is used to initialize the fine-tuning phase. The fine-tuning phase optimizes the weights by minimizing the cross-entropy error

learn a latent representation (often of lower dimension) of the input data by optimizing the instantiation of the latent variables (latent representation) and the weights of the network in order to best allow the latent representation to regenerate the input data [14]. The objective of an RBM is to find the parameters θ (including a weight matrix W and offset vectors (biases) b, c) that maximize the probability of the input data (visible layer, v) [1].

$$\underset{\theta}{\operatorname{argmax}} P(v) = \sum_{h} P(v,h) \tag{1}$$

$$P(v,h) = \frac{1}{Z} e^{-E(v,h)} \tag{2}$$

$$E(v,h) = -h^T W v - c^T v - b^T h \tag{3}$$

The joint probability $P(v, h)$ of the hidden h and visible v layers is a function of the energy function $E(.)$ [1]. Z is a normalization factor or partition function.

In the fine-tuning phase, the DBN model is "unfolded" (leading to a network often referred to as a deep autoencoder, DA) in order to learn to reconstruct the input data using the weights learned during the pre-training phase. The fine-tuning phase performs a global optimization of all weights using stochastic gradient descent and the backpropagation algorithm, with the goal of minimizing the difference between the distribution of the data and the distribution formed by the model's reconstructions of the data (cross-entropy error) [13, 14]. Deep networks can be somewhat difficult to train [16, 17]. Using the weights learned during pre-training to initialize a DA, as opposed to random initialization, seems to improve the generalization of the completely trained DBN by minimizing the variance of all possible solutions to the DBN [13, 18].

In more detail, a DA is a multi-layered network composed of an encoder and decoder network [14]. The encoder network learns multiple encodings (hidden layer representations) of the input by propagating the input forward through the network (as one would in a neural network using a linear transformation and a nonlinearity/squashing function), learning alternate representations of the input at each hidden layer [1, 14]. Once the final hidden layer is computed, the decoder network propagates in reverse [1, 14]. When propagating in reverse, the DA uses the final hidden layer of the network to attempt to regenerate the original input data. The output of fine-tuning (DA) is a reconstruction of the input data based on decoder propagation through the network. Cross-entropy error can be used to determine how close these reconstructions are to the original input, and the weights can be updated in the appropriate direction (trained using backpropagation of error derivatives and stochastic gradient descent) [1, 14] in order to minimize the cross-entropy error. Reconstruction error (mean squared error between the data and the reconstructions of the data) is often used to monitor learning progress. More detailed descriptions of training DBNs can be found in [8, 14, 15].

We implemented a DBN [14] using the Python programming language and the Theano library (a symbolic numerical computation python library) [19]. This implementation is compatible with Mac OS or Linux computing environments and is capable of utilizing GPUs if available.

Model selection

In order to investigate the impact of the hyperparameters (network architecture, learning rate, training duration) on modeling the cancer transcriptomic data, we performed a series of model selection experiments. Model selection was performed using a modified 8-fold cross-validation. In order to speed up our model selection (allowing us to explore more sets of hyperparameters), while still training on a large percentage of our dataset (considering our dataset had a somewhat small number of samples relative to the number of features), we only performed four folds of an 8-fold split of the data. Our strategies for deep learning model selection were guided by articles from Bengio [20] and Hinton [21]. A combined random and grid search approach [20] were used with a goal of finding the set of hyperparameters that minimized the average test set reconstruction error and prevented overfitting, while also significantly reducing the dimensionality of the data (i.e., final (top) hidden layer with around 100 units). Please see the Results and discussion section for more information on model selection.

Consensus clustering

After model selection, we trained the deep learning model and then computed and collected the top hidden layer (the most abstract) representations for each sample. We performed consensus clustering on the top hidden layer representations (i.e., 3rd hidden layer representations, meaning the final 100–200 dimensional projections of the input data) of each tumor as calculated by our trained DBN models; as well as, on the high-dimensional input data alone. Consensus clustering was performed using the 'ConsensusClusterPlus' [22] package from the R statistical programming language [23], using agglomerative hierarchical clustering with Euclidean distance and average linkage. Consensus clustering performs multiple trials (in this case 100) of clustering based on randomly sampling a subset of the data (in this case 80%). Each sample is given a final cluster assignment at the end of each trial. A consensus matrix is created after all trials have completed. A

consensus value is a number between 1 and 0 that represents how often two samples did or did not cluster together, respectively.

The output of the consensus clustering was a dendrogram and an associated consensus matrix heatmap (dimensions = number of samples by number of samples), representing how often samples clustered together in repeated clustering trials. The DBN models with the lowest percentage of ambiguous clustering (PAC) [24] values and most visually informative heatmaps were selected for further analysis. The PAC represents the proportion of data points in the consensus matrix with intermediate (between 0.8 and 0.2) consensus values, meaning that the two samples clustered together in some runs of clustering, but not in others.

Kaplan-Meier survival analysis

A Kaplan-Meier plot was created using the clustering assignments from the consensus of consensus clustering for GBM samples. Kaplan-Meier plots were created using the 'survival' package in the R statistical computing language [23]. *P*-values were calculated using the log-rank test.

Correlation between genes and clusters

Correlation studies were performed to find the differentially expressed genes or mutations that correlated with each GBM cluster. The Pearson correlation between each differentially expressed gene (input features) and GBM cluster was measured using the 'cor' function in R. The Pearson correlation between each mutation (TCGA GBM somatic mutation data version "2015-01-27") and GBM cluster was also measured using the 'cor' function in R. Functions of example genes were obtained from www.genecards.org and the Gene Ontology Consortium (http://geneontology.org/).

Results and discussion

Model selection

This study concentrated on finding the network architecture of a DBN model that was capable of learning "optimal" representations of cancer expression data, which is a model selection task. We performed a series of model selection experiments to find the best set of hyperparameters (e.g., number of hidden layers, number of units in each hidden layer, learning rates, training epochs, batch size, etc.). Approximately 1500 different sets of hyperparameters, including models with up to five hidden layers, were evaluated by cross-validation, representing approximately 1.5 months of computation time on a single Tesla k40 GPU.

We started model selection by performing a random search over all hyperparameters, in which hyperparameters were randomly sampled from a range of values. We performed this search subject to some constraints, such as decreasing hidden layer size (e.g., hidden layer 1 always larger than hidden layer 2, etc.) and pre-defined maximum unit thresholds for hidden layer sizes. Based on these results, we selected a partial set of hyperparameters (including pre-training and fine-tuning learning rates, batch size, and pre-training and fine-tuning epochs of training duration) that appeared to perform well over a broad range of hidden layer architectures. Using this partial set of hyperparameters, we performed an extensive grid search over hidden layer architectures (from 1 to 5 hidden layers with varying number of hidden units in each layer) and evaluated the resulting reconstruction errors (Fig. 3). For all experiments in Fig. 3, only the number of hidden layers and number of units in each hidden layer varied. The other hyperparameters (i.e., learning rates, number of pre-training and training epochs, and batch size) were fixed.

Figure 3a shows how reconstruction error changes with respect to the size of the 1st hidden layer and training epochs for networks with only a *single hidden layer*, i.e., a single RBM that is then fine-tuned. As expected, the reconstruction error for a single hidden layer network decreases as the size of the hidden layer increases, and does not provide much insight into choosing the size of the 1st hidden layer. A single hidden layer DBN cannot learn the hierarchical relationships that we are interested in discovering. Therefore, we explored DBNs with multiple hidden layers.

Figure 3b shows the model selection results for different two hidden layer architectures, where the number of hidden units in the 1st hidden layer range from 100 to 3500 (across x-axis), and the number of units in the 2nd hidden layer range from 50 to 3300 (indicated by color code). Each circle in this graph (and Fig. 3c, d) represents the reconstruction error for a network architecture. Figure 3b shows that, with a small number of hidden units (50 – 100 units) in the top (2nd) hidden layer, increasing the size of the 1st hidden layer beyond 1100 does not lead to a significant reduction in reconstruction error. Indeed, as the size of the 1st hidden layer increases beyond 1100 units, the reconstruction errors remain flat and show a tendency to increase slightly. In contrast, when the size of 2nd (top) hidden layer is relatively large (> 500), the reconstruction errors continue to decrease as the size of the 1st hidden layer increases.

We hypothesized that, since a DBN is an encoding machine, an optimal model should be able to encode the input data and pass it through an information bottleneck—a model with a very small number of hidden units in the top layer of the network. Such a model would require each of the layers below this bottleneck to capture as much information as possible in order to efficiently

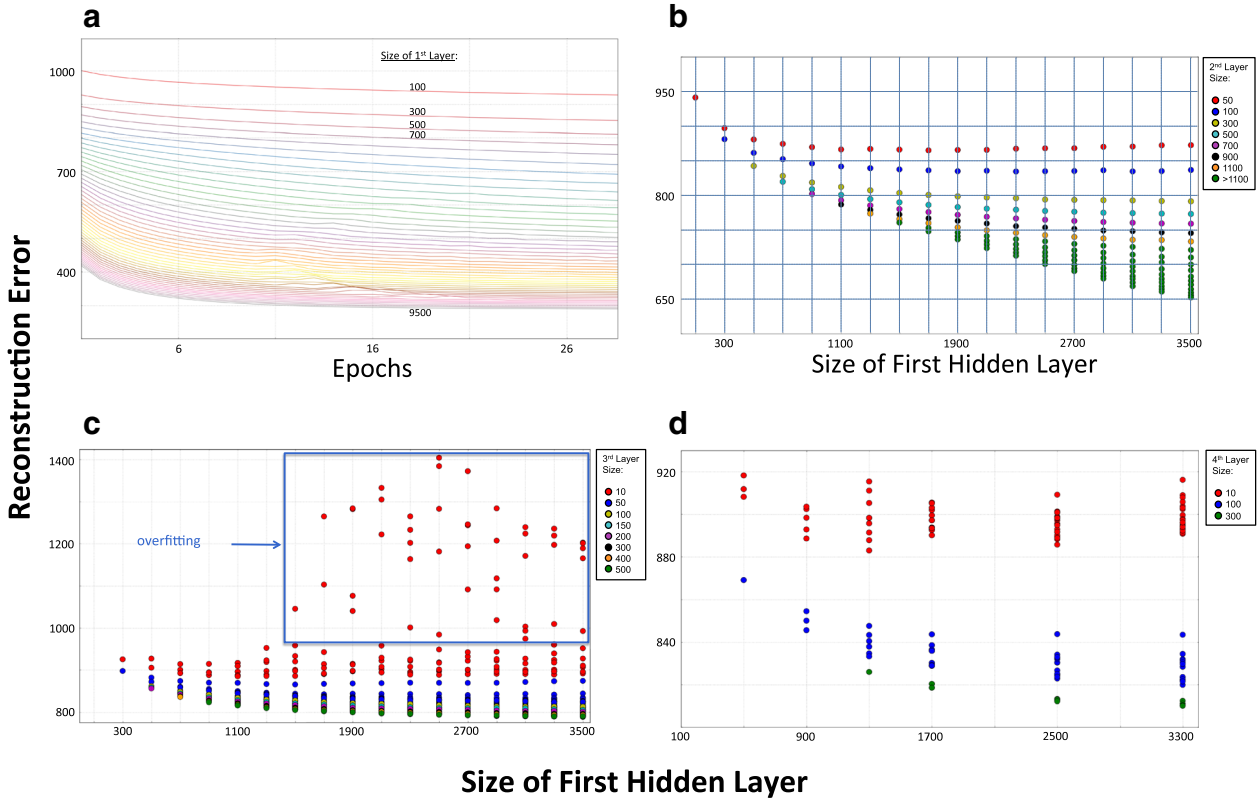


Fig. 3 Model selection cross-validation reconstruction error. For all plots in this figure, the y-axes represent reconstruction error. (**a**) Reconstruction error for multiple sizes of a one hidden layer network. The size of the 1st hidden layer increases by 200 units for each consecutive curve. (**b**) Reconstruction error for two hidden layer networks. (**c**) Reconstruction error for three hidden layer networks. The *blue rectangle* represents network architectures that overfit the data. (**d**) Reconstruction error for four hidden layer networks

pass the bottleneck (i.e., maintaining a low reconstruction error). As such, one can search for an optimal architecture by starting with a very small number of hidden units in the top layer and selecting the optimal number of hidden units layer-by-layer starting from the 1st hidden layer (closest to input data).

Figure 3c shows how reconstruction error changes as the sizes of the 1st, 2nd, and 3rd hidden layers change (units in 2nd hidden layer ranged from 100 to 2300). What really stands out in this graph is that overfitting is observed when the 1st hidden layer is greater than 1300 and the number of hidden units in the top hidden layer is set to 10 (overfitting indicated by a blue rectangle in the figure). Figure 3d shows how reconstruction error for four-hidden-layer DBNs changes as the sizes of the 1st, 2nd, 3rd, and 4th hidden layers change (2nd hidden layer ranged from 100 to 2100 and 3rd hidden layer from 50 to 500). Similar to Fig. 3c, d also shows overfitting when the top hidden layer size is set to 10 and the 1st hidden layer is large. We also examined five-hidden-layer networks, which showed results similar to Fig. 3d (results not shown).

In total, these results suggest that the DBN begins to capture noise in the data when the 1st hidden layer size is greater than 1300 units. Accordingly, a 1st hidden layer size around 1300 units should provide the optimal encoding of the data when the number of hidden units in the top hidden layer (the information bottleneck) is small. We hypothesized that the 1st hidden layer captures the signals encoded by TFs in human cells, and our results suggest that 1300 hidden units most effectively captures the covariance structure (hypothesized to be signals of TFs) within the data at the level of abstraction of the 1st hidden layer. Interestingly, our hypothesis agrees surprisingly well with the current consensus on the number of human TFs (~1400) estimated through analyzing the human genome [25]. These results also correlate with Chen et al. [8], who found nearly a one-to-one mapping between hidden units in the first hidden layer and yeast transcription factors.

As previously mentioned, we searched for optimal hidden layer sizes by finding 'elbows' in the plot of reconstruction error vs. hidden layer size, where the reconstruction error decreases less rapidly (as can be

seen in Fig. 3b, c, and d). We then set the top hidden layer size to be 100 – 200 units to provide a relatively rich representation, while avoiding unnecessary complexity. We found that DBNs with four hidden layers (Fig. 3d) or 5-hidden layer networks (not shown) didn't offer much, if any, improvement over a 3-layer network. We selected four 3-hidden-layer network architectures ([1100-500-100], [1300-700-100], [1300-700-150], [1400-1000-200]) with different combinations of hidden units in the "optimal" range. Next, we performed a random search over the learning rates for the four network architectures selected above and evaluated the reconstruction errors. Finally, we decided to use six different sets of hyperparameters (including network architecture) to test their ability to capture statistical structures in cancer gene expression data (Table 2).

Clustering tumors based on DBN-derived representations

The purpose of training a DBN model with a large number of tumors of multiple cancer types was to use a large amount of data to enhance our learning of statistical structures—to potentially reveal different disease mechanisms. Based on the assumption that the learned representations reflect the state of cellular signaling systems, it would be interesting to learn if these representations can be used to reveal cancer subtypes that share a common pattern of pathway perturbation. To this end, we represented each tumor using the states of the hidden units in the top (3rd) layer and performed consensus clustering. Figure 4 shows that the 3rd hidden layer representations from a trained DBN (Fig. 4a) clustered drastically better than the high-dimension raw gene expression profiles (Fig. 4c, d). When tumors were represented by the 9476 input gene features, consensus clustering failed to find any meaningful clusters (i.e., multiple clusters consisting of a large number of samples).

While it is tempting to use a clustering approach to find common cancer subtypes across multiple cancer types, this approach is complicated by the fact that certain pathways exhibit tissue-specific expression, and clustering will be dominated by these tissue-specific features. This will eventually lead to the clustering of all tumor samples according to tissue type, as demonstrated in the study by Hoadley et al. [26]. Indeed, we also found that virtually all of our tumor samples clustered according to tissue type (Fig. 4a). For example, the top right cluster in the heatmap in Fig. 4a (cluster 4, colored light blue) consisted of all lung tissue samples (lung adenocarcinoma and lung squamous cell carcinoma) except for two outliers.

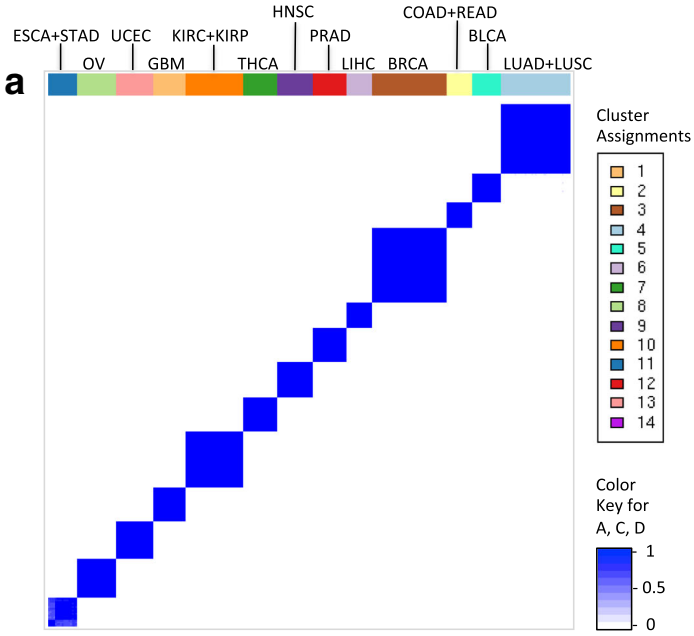
This tissue-specific clustering occurred despite our best attempts to remove all tissue-specific genes by representing a tumor using only genes with extremely high variance (Bernoulli success probability, 0.70) to train DBNs, and then perform consensus clustering (results not shown). Using these high variance genes reduced our number of features (genes) to 2674, with each of these genes being differentially expressed in more than 2250 tumor samples. The genes in this 2674 set cannot be tissue-specific because most cancer/tissue types in our data set have only ~500 tumor samples or less (except breast cancer and lung tissue, which have ~1000 tumors). These results indicate that there are tissue-specific pathway perturbation patterns that lead to a tissue-specific covariance structure of DEGs, which were captured by the DBN, and in turn recognized by consensus clustering. These results illustrate the limitations of using gene-expression-related features (e.g., gene expression, copy number alteration, and DNA methylation) to study disease mechanisms shared by tumors across different tissue types [26]. Therefore, different approaches to studying disease mechanisms should be explored.

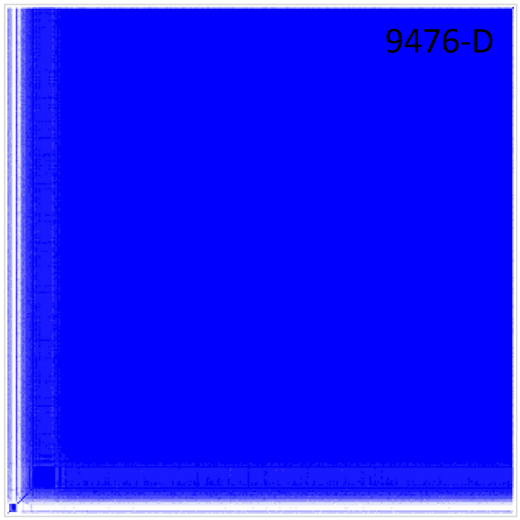
Table 2 Model selection results. Six different hyperparameter sets for final training of DBN

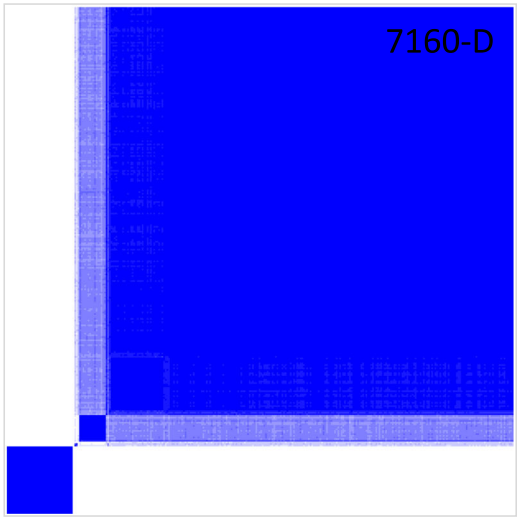
Set ID	Hidden Layer Sizes			Learning Rates		Epochs		Input Size
	1st	2nd	3rd	pretrain	train	pretrain	train	
1	1100	500	100	7.75E-05	2.41E-03	14	101	9476
2	1300	700	100	7.75E-05	2.41E-03	14	91	9476
3	1300	700	150	7.75E-05	2.41E-03	14	88	9476
4	1300	700	150	7.75E-05	2.41E-03	14	62	9476
5	1400	1000	200	3.03E-03	3.26E-03	14	40	7160
6	1300	700	150	7.75E-05	2.41E-03	14	97	7160

Within tissue type clustering revealed clinically relevant subtypes

Although it is difficult to search for common disease mechanisms across multiple cancer types due to the aforementioned limitations, we hypothesized that, within a given tissue type, clustering using DBN-learned representations may reveal distinct disease mechanisms. Since the survival data for glioblastoma multiforme (GBM) patients from TCGA was relatively more complete than other cancer types in our data set (allowing us to perform more robust survival analysis), we studied GBM in more detail. Previously, Verhaak et al. [27] selected a set of genes as features and performed clustering of GBM tumors from TCGA based on their expression values. However, manually selecting features may introduce bias, and therefore we set out to investigate if an unbiased representation of the data would lead to different results. We first used the raw input gene expression data as features and performed consensus clustering, but failed to find any clusters (data not shown). These







b

Cluster	Contents	n	Tissue
1	GBM	480	Brain
2	COAD, READ	283, 93	Lower GI
3	BRCA	1073	Breast
4	LUAD, LUSC	509, 498	Lung
5	BLCA	403	Bladder
6	LIHC	364	Liver
7	THCA	496	Thyroid
8	OV	559	Ovary
9	HNSC	507	Head, Neck
10	KIRC, KIRP	524, 288	Kidney
11	ESCA, STAD	183, 236	Upper GI
12	PRAD, HNSC	491, 1	Prostate
13	UCEC	535	Uterus
14	THCA, LUAD, LUSC	3, 1, 1	Outliers

Fig. 4 Consensus clustering of all samples. (**a**) Consensus clustering of the 3rd hidden layer representations from our DBN model (model 5 in Table 2) captured tissue-specific clustering. The cancer types within each cluster are shown on *top*. (**b**) Composition of each of the clusters learned in part a, including the number of samples of each cancer type in each cluster. (**c**) Consensus clustering results when simply using the high-dimensional (9476 features/genes) raw data without any dimensionality reduction. (**d**) Consensus clustering results for high-dimensional (7160 features/genes) raw data without any dimensionality reduction. All heatmaps evaluated at k = 14 (number of clusters). *Dark blue* corresponds to samples that always cluster together (consensus value = 1). *White* corresponds to samples that never cluster together (consensus value = 0). *Lighter shades of blue* are intermediate between 1 and 0

results underscore the motivation for the feature-selection approach adopted by Verhaak et al. [27].

We then set out to investigate whether an unbiased representation learned by a DBN would reveal subtypes (clusters) of GBM. We trained six DBN models with different architectures and hyperparameters (see Table 2), performed consensus clustering using the results from each model (top layer representations), and we pooled the results to build a 'consensus of consensus clustering'. The heatmap in Fig. 5a shows the general agreement across GBM cluster assignments as derived from the six DBN models. This is a type of ensemble learning where each of the six models gets to 'vote' for which samples should be clustered together. Using PAC scores (see Methods) as a selection criterion, we identified six major clusters (Fig. 5a). We explored this six major cluster separation further to see if the learned clusters were clinically relevant.

Figure 5b shows the Kaplan-Meier plot for the samples in the six major GBM clusters (Fig. 5a). There is a difference in survival between the patient clusters and in particular the red curve/cluster seems to have better

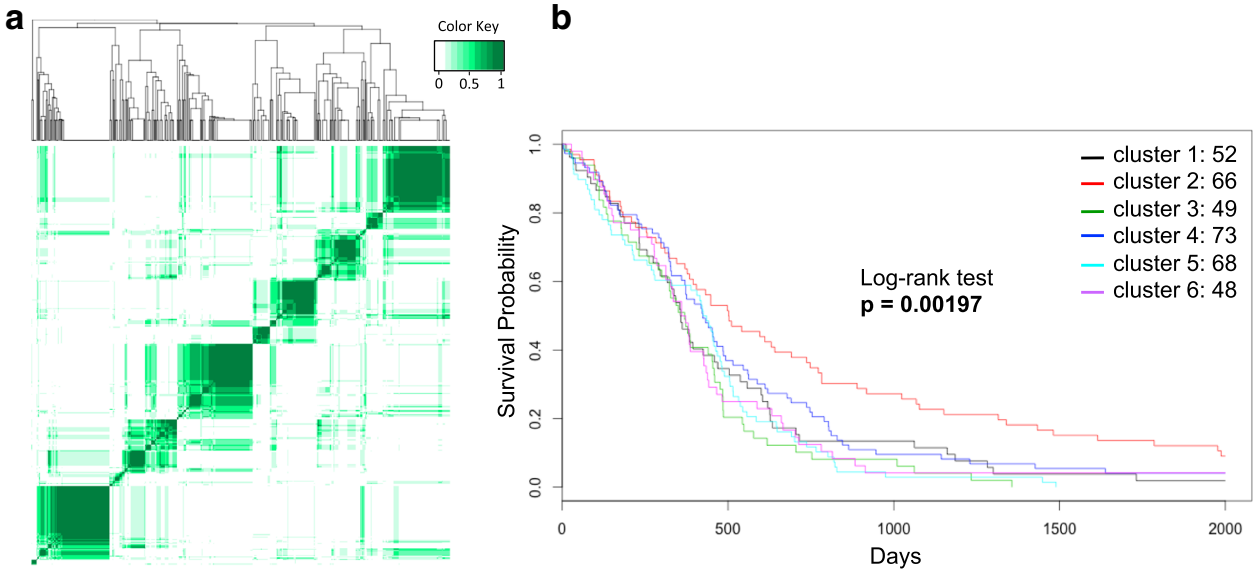


Fig. 5 GBM consensus clustering ensemble results and corresponding Kaplan-Meier plot. (**a**) Heatmap and dendrogram for GBM consensus of consensus clustering results for six major clusters (k = 11). Dark green corresponds to samples that always clustered together (consensus value = 1). *White* corresponds to samples that never clustered together (consensus value = 0). *Lighter shades of green* are intermediate between 1 and 0. (**b**) Kaplan-Meier plot for six largest clusters in part a. Each GBM cluster is represented by a *colored curve* in the plot. *Top right* of figure shows the number of samples in each cluster

survival or prognosis relative to the other clusters. The *p*-value for Kaplan-Meier plot using the log-rank test was $p = 0.00197$.

GBM is a highly aggressive malignant brain tumor. Previous molecular analysis of GBM tissue samples by Verhaak et al. identified four molecular subtypes: mesenchymal, proneural, neural, and classical [27]. The analysis of the four subtypes identified by Verhaak et al. did not reveal significant differences in survival between the four clusters, but the tumors did exhibit different responses to treatments. More recently, Brennan et al. further divided tumors within the proneural subtype into G-C island methylation phenotype (G-CIMP) and non G-CIMP subtypes [28] based on DNA methylation data. Here, our DBN-derived representations separated GBM tumors into six clusters (Fig. 5a), and our subtyping revealed significant differences in patient survival (Fig. 5b), indicating that our novel representations provide more information than using individual gene expression values as features. We compared our subtyping (learned using deep learning and consensus clustering) with the known subtyping of our TCGA GBM samples (Fig. 6) as published by Brennan et al. [28].

		Expression Subtype*				
			Proneural			
		Mesenchymal	Non G-CIMP	G-CIMP	**Neural**	**Classical**
Final Clusters	1 (black)	**44**	1	0	2	4
from this Study	2 (red)	1	**48**	**33**	1	1
(using deep	3 (green)	23	7	0	18	8
learning and	4 (blue)	11	1	0	13	**52**
consensus	5 (lt. blue)	12	2	0	**20**	**51**
clustering)	6 (purple)	28	18	0	6	8

*GBM sample molecular expression subtyping from Brennan et al. *Cell*. 2013

Fig. 6 GBM subtypes in each cluster from Fig. 5b based on molecular subtyping from Brennan et al. [28]

Figure 6 shows the contents of our six GBM clusters based on the subtype published by Brennan et al. [28]. Most of our GBM clusters included tumor samples belonging to multiple different known subtypes. Exceptions to this were our black and red clusters (see Fig. 5b for cluster colors). The black cluster consisted of mostly mesenchymal subtype. The red cluster (cluster with best prognosis) in the Kaplan-Meier plot consisted of almost entirely proneural subtype samples. Interestingly, this red cluster captured all of the samples with the G-CIMP phenotype and the majority of the non G-CIMP proneural tumors, but assigned the rest of non G-CIMP tumors to the purple cluster. The G-CIMP phenotype (samples with hypermethylation at CpG islands) subgroup of GBM has been shown in previous studies to have better survival [28–30]. These results indicate that without utilizing DNA methylation data, DBN learned representations accurately captured the impact of DNA methylation on expression—an indication that our novel representations may reflect disease mechanisms at the pathway level.

Novel clusters provide information regarding disease mechanisms

We investigated the six GBM clusters further using correlation analysis to find DEGs and mutations that were associated with each cluster. Figure 7 (left panel) shows word clouds for the top 10 DEGs with the largest positive correlations with each GBM cluster. Each word cloud of genes is colored according to their corresponding cluster color in the Kaplan-Meier plot. For example, the red colored genes represent the DEGs or mutations with the largest correlations with the red cluster (cluster with the best prognosis). We found genes in each of these groups with functions relevant to cancer. For example, *CHD7* is highly correlated with the red cluster

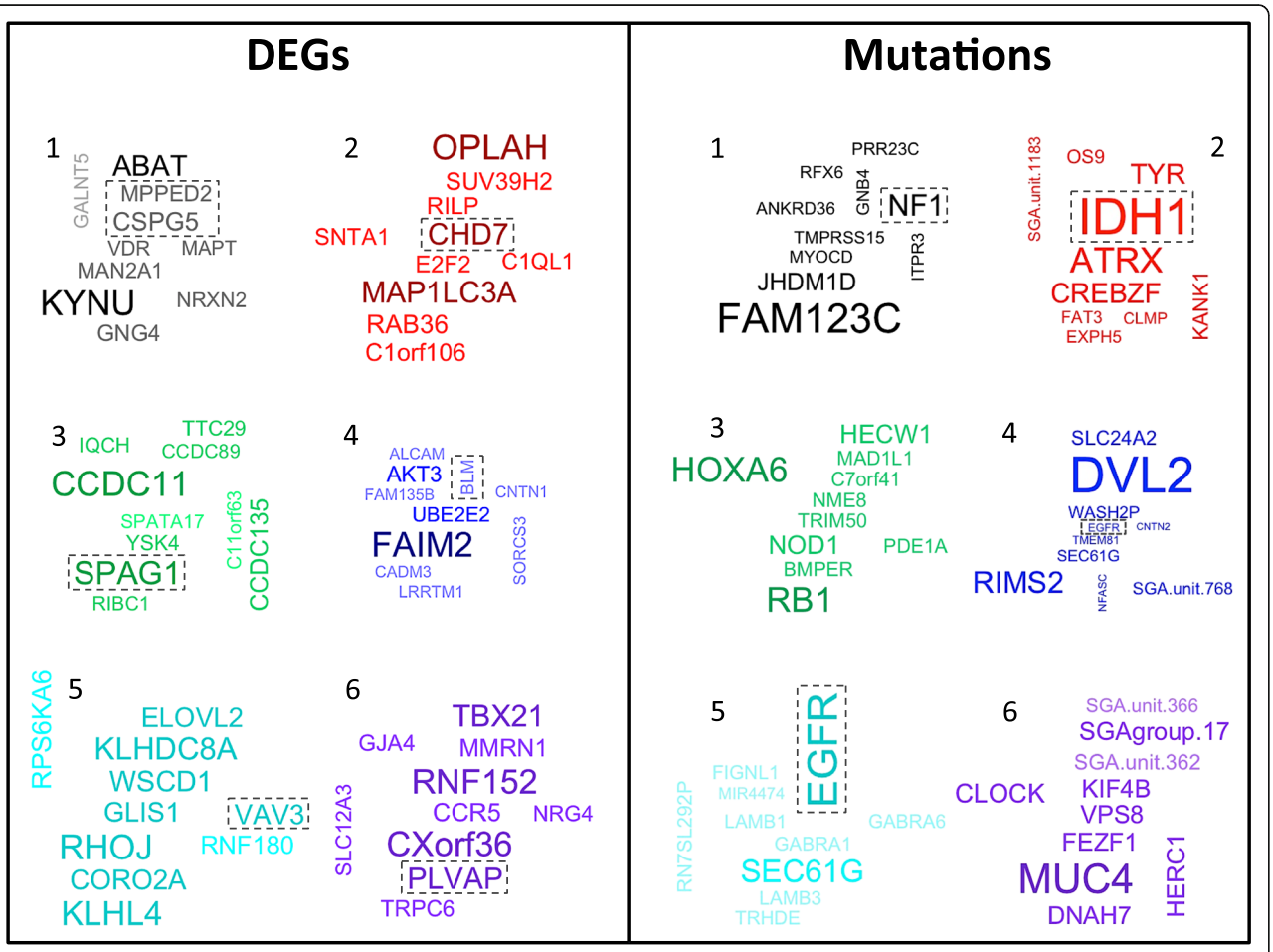


Fig. 7 Word clouds for the top 10 DEGs or mutations with the largest correlations with each GBM cluster. Each GBM cluster is represented by a word cloud of genes, colored according to the corresponding curve in the Kaplan-Meier plot (Fig. 5b). The size and color of each gene in each word cloud correspond to the relative strength of the correlation for that gene. The largest and darkest words in the word clouds correspond to the strongest correlations. Each cluster's word cloud was created independently. Therefore, the differential sizes of each gene (representing correlations) is only relevant when compared to other genes in that cluster. Gene sizes should not be compared across different clusters

and is involved in DNA packaging and chromatin binding. *CSPG5* and *MPPED2* (highly correlated with the black cluster) are involved in nervous system development. *BLM* (blue cluster) is involved in DNA binding and regulation of cell proliferation. *VAV3* (light blue cluster) is involved in GTPase activator activity. *SPAG1* (green cluster) is known to be involved in epilepsy and pancreatic cancer. *PLVAP* (purple cluster) may function in microvascular permeability.

Figure 7 (right panel) shows word clouds for the top 10 mutations with the largest positive correlations with each GBM cluster. This correlation analysis yielded many well-known mutations involved in cancer and GBM. *IDH1* is the mutation with the strongest correlation with the red cluster, which includes all tumors belonging to G-CIMP subtype of GBM [28]. This finding is biologically sensible in that it is known that mutations in *IDH1* lead to significant changes in DNA methylation, which underlie the G-CIMP. Similarly, *NF1* mutations are strongly associated with the black cluster (corresponding to the mesenchymal subtype) and are known to be frequently mutated in the mesenchymal subtype [27].

The above results reveal connections between genomic alterations and DEGs specifically associated with each subtype, which provide information about the disease mechanisms for each subtype. It is reasonable to assume that the genomic alterations associated with a cluster likely perturb pathways underlying the subtype, and hidden units in our DBN models could potentially represent the states of these pathways. Any aberration in these pathways causes a change in the expression of the DEGs associate with that cluster. Studying the potential causal relationships between mutation events and the changing states of hidden units may provide information about how mutations affect signaling pathways in cancer cells.

Conclusions

In this study, we showed that an unsupervised DBN can be used to find meaningful low-dimensional representations of cancer gene expression data. More specifically, first, we designed a rigorous model selection scheme, which enabled us to determine the optimal number of hidden units in the 1st and 3rd hidden layers of our model. We hypothesized that the 1st hidden layer likely represented the TFs utilized by cancer cells and our results correlate with current knowledge of the number of TFs. Second, we showed that consensus hierarchical clustering of GBM tumors using the unbiased representations (the top (final) hidden layer units) revealed more robust clustering results than clustering based on the raw gene expression data. Third, we showed that clinically relevant information was encoded in the representations learned by our DBN. This was demonstrated through the discovery of a subtyping of GBM with differential prognosis, which previously was not discovered by TCGA. Our methods identified a subtype of GBM enriched with the G-CIMP phenotype without using DNA methylation data, and our analysis can partially attribute this subtype to the mutation of *IDH1*. This also agrees with current knowledge. Further investigation may reveal disease mechanisms underlying the different GBM clusters. What role do these genes/mutations have in GBM? What role do they play in survival?

This study represents a novel application of the deep learning algorithm developed by Hinton and Salakhutdinov [14] in the cancer bioinformatics domain. To our knowledge, unsupervised deep learning has not been used to find hidden structure within cancer gene expression data for the purposes of providing insight into disease mechanisms of tumors and patient survival. As for the possible future enhancement of the model, we conjecture that a sparse version of our DBN may more readily encode cellular pathways. A trained model needs to be able to represent all cancer pathways in order to fit the data from the thousands of tumors studied here, however a given tumor likely only hosts a small number of aberrant pathways. A sparse DLM can limit the number of active hidden units in a given layer representing a tumor, thus it theoretically could perform better. This will be investigated in future studies.

Abbreviations

ANN: Artificial neural network; DA: Deep autoencoder; DBN: Deep belief network; DEG: Differentially expressed gene; DLM: Deep learning model; GBM: Glioblastoma multiforme; G-CIMP: G-C island methylation phenotype; PAC: Percentage of ambiguous clustering; RBM: Restricted Boltzmann machine; SRBM–DA: Stacked restricted Boltzmann machines–deep autoencoder; TCGA: The cancer genome atlas; TF: Transcription factor

Acknowledgments

We would like to thank the Pittsburgh Supercomputing Center for providing computing facilities for this study.

Funding

Research reported in this publication was supported by grant R01LM012011 and U54HG008540 awarded by the National Library of Medicine and the National Human Genome Research Institute respectively. This article's publication costs were supported by grant R01LM012011 from the National Library of Medicine. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

Availability of data and materials

The dataset used for the current study is available at https://github.com/young-jon/bmc_bioinformatics_2017.

Authors' contributions

JY and XL designed the study, interpreted the results, and wrote the manuscript. JY performed the computational experiments. CC collected and preprocessed the experimental data, and participated in the manuscript writing. All authors read and approved the final manuscript.

Ethics approval and consent to participate

Not applicable.

About this supplement
This article has been published as part of BMC Bioinformatics Volume 18 Supplement 11, 2017: Selected articles from the International Conference on Intelligent Biology and Medicine (ICIBM) 2016: bioinformatics. The full contents of the supplement are available online at https://bmcbioinformatics.biomedcentral.com/articles/supplements/volume-18-supplement-11.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Publisher's Note

Author details
[1]Department of Biomedical Informatics, University of Pittsburgh, 5607 Baum Blvd, Pittsburgh, PA 15206, USA. [2]Intelligent Systems Program, University of Pittsburgh, 5607 Baum Blvd, Pittsburgh, PA 15206, USA. [3]Center for Causal Discovery, University of Pittsburgh, 5607 Baum Blvd, Pittsburgh, PA 15206, USA.

Published: 3 October 2017

References

1. Deng L, Yu D. Deep Learning: Methods and Applications. Found Trends Signal Process. 2014;7:197–387.
2. Fakoor R, Ladhak F, Nazi A, Huber M. Using Deep Learning to Enhance Cancer Diagnosis and Classification. In: Proceedings of the 30th International Conference on Machine Learning Workshop on the Role of Machine Learning in Transforming Healthcare; 2013.
3. Krizhevsky A, Sutskever I, Hinton G. ImageNet Classification with Deep Convolutional Neural Networks. Adv Neural Inf Proces Syst. 2012:25:1097–105.
4. Le Q, Ranzato M, Monga R, Devin M, Chen K, Corrado G, Dean J, Ng A. Building High-Level Features using Large Scale Unsupervised Learning. In: Proceedings of the 29th International Conference on Machine Learning; 2012.
5. LeCun Y, Bengio Y, Hinton G. Deep learning. Nature. 2015;521(7553):436–44.
6. Lee H, Ekanadham C, Ng A. Sparse deep belief net model for visual area V2. Adv Neural Inf Proces Syst. 2008;20:873–80.
7. Lee H, Grosse R, Ranganath R, Ng A. Unsupervised learning of hierarchical representations with convolutional deep belief networks. Commun ACM. 2011;54(10):95–103.
8. Chen L, Cai C, Chen V, Lu X. Learning a hierarchical representation of the yeast transcriptomic machinery using an autoencoder model. BMC Bioinformatics. 2016;17(Suppl 1):9.
9. Chen L, Cai C, Chen V, Lu X. Trans-species learning of cellular signaling systems with bimodal deep belief networks. Bioinformatics. 2015;31(18):3008–15.
10. Liang MX, Li ZZ, Chen T, Zeng JY. Integrative Data Analysis of Multi-Platform Cancer Data with a Multimodal Deep Learning Approach. IEEE/ACM Trans Comput Biol Bioinform. 2015;12(4):928–37.
11. The Cancer Genome Atlas. http://cancergenome.nih.gov/. Accessed 1 Apr 2015.
12. Mermel CH, Schumacher SE, Hill B, Meyerson ML, Beroukhim R, Getz G. GISTIC2.0 facilitates sensitive and confident localization of the targets of focal somatic copy-number alteration in human cancers. Genome Biol. 2011;12(4):R41.
13. Goodfellow IJ, Bengio Y, Courville A. Deep learning. Book in preparation for MIT Press. 2016. http://www.deeplearningbook.org/. Accessed 10 Jul 2016.
14. Hinton GE, Salakhutdinov RR. Reducing the dimensionality of data with neural networks. Science. 2006;313(5786):504–7.
15. Hinton GE, Osindero S, Teh YW. A fast learning algorithm for deep belief nets. Neural Comput. 2006;18(7):1527–54.
16. Bengio Y. Learning Deep Architectures for AI. Found Trends Mach Learn. 2009;2(1):1–127.
17. Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. Int Conf Artif Intell Stat. 2010;9:249–56.
18. Erhan D, Bengio Y, Courville A, Manzagol PA, Vincent P, Bengio S. Why does unsupervised pre-training help deep learning? J Mach Learn Res. 2010;11:625–60.
19. The Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. 2016. *arXiv preprint arXiv: 160502688*.
20. Bengio Y. Practical Recommendations for Gradient-Based Training of Deep Architectures. In: Montavon G, Orr GB, Müller KR, editors. Neural Networks: Tricks of the Trade. 2nd ed. Berlin, Heidelberg: Springer; 2012. p. 437–78.
21. Hinton GE. A Practical Guide to Training Restricted Boltzmann Machines. In: Montavon G, Orr GB, Müller KR, editors. Neural Networks: Tricks of the Trade. 2nd ed. Berlin, Heidelberg: Springer; 2012. p. 599–619.
22. Wilkerson M, Hayes DN. ConsensusClusterPlus: A class discovery tool with confidence assessments and item tracking. Bioinformatics. 2010;26(12):1572–3.
23. R Core Team. R: A language and environment for statistical computing. R Foundation for Statistical Computing. Vienna: R Foundation for Statistical Computing; 2013. http://www.R-project.org/
24. Senbabaoglu Y, Michailidis G, Li JZ. Critical limitations of consensus clustering in class discovery. Sci Rep. 2014;4:6207.
25. Vaquerizas JM, Kummerfeld SK, Teichmann SA, Luscombe NM. A census of human transcription factors: function, expression and evolution. Nat Rev Genet. 2009;10(4):252–63.
26. Hoadley KA, Yau C, Wolf DM, Cherniack AD, Tamborero D, Ng S, Leiserson MD, Niu B, McLellan MD, Uzunangelov V, et al. Multiplatform analysis of 12 cancer types reveals molecular classification within and across tissues of origin. Cell. 2014;158(4):929–44.
27. Verhaak RG, Hoadley KA, Purdom E, Wang V, Qi Y, Wilkerson MD, Miller CR, Ding L, Golub T, Mesirov JP, et al. Integrated genomic analysis identifies clinically relevant subtypes of glioblastoma characterized by abnormalities in PDGFRA, IDH1, EGFR, and NF1. Cancer Cell. 2010;17(1):98–110.
28. Brennan CW, Verhaak RG, McKenna A, Campos B, Noushmehr H, Salama SR, Zheng S, Chakravarty D, Sanborn JZ, Berman SH, et al. The somatic genomic landscape of glioblastoma. Cell. 2013;155(2):462–77.
29. Baysan M, Bozdag S, Cam MC, Kotliarova S, Ahn S, Walling J, Killian JK, Stevenson H, Meltzer P, Fine HA. G-CIMP status prediction of glioblastoma samples using mRNA expression data. PLoS One. 2012;7(11):e47839.
30. Noushmehr H, Weisenberger DJ, Diefes K, Phillips HS, Pujara K, Berman BP, Pan F, Pelloski CE, Sulman EP, Bhat KP, et al. Identification of a CpG island methylator phenotype that defines a distinct subgroup of glioma. Cancer Cell. 2010;17(5):510–22.

Bibliography

[Ando and Iba, 2001] Ando, S. and Iba, H. (2001). Inference of gene regulatory model by genetic algorithms. In *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546)*, volume 1, pages 712–719. IEEE.

[Bastian et al., 2009] Bastian, M., Heymann, S., and Jacomy, M. (2009). Gephi: an open source software for exploring and manipulating networks. In *Third International AAAI Conference on Weblogs and Social Media*.

[Bengio, 2012] Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. In *Neural Networks: Tricks of the Trade*, pages 437–478. Springer.

[Bengio et al., 1994] Bengio, Y., Simard, P., Frasconi, P., et al. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

[Blondel et al., 2008] Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008.

[Brennan et al., 2013] Brennan, C. W., Verhaak, R. G., McKenna, A., Campos, B., Noushmehr, H., Salama, S. R., Zheng, S., Chakravarty, D., Sanborn, J. Z., Berman, S. H., et al. (2013). The somatic genomic landscape of glioblastoma. *Cell*, 155(2):462–477.

[Cai et al., 2019] Cai, C., Cooper, G., Lu, K., Ma, X., Xu, S., Zhao, Z., Chen, X., Xue, Y., Lee, A., Clark, N., Chen, V., Lu, S., Chen, L., Yu, L., Hochheiser, H., Jiang, X., Wang, Q., and Lu, X. (2019). Systematic discovery of the functional impact of somatic genome alterations in individual tumors through tumor-specific causal inference. *PLoS Computational Biology*, 15(7):e1007088.

[Chen et al., 2015] Chen, L., Cai, C., Chen, V., and Lu, X. (2015). Trans-species learning of cellular signaling systems with bimodal deep belief networks. *Bioinformatics*, 31(18):3008–3015.

[Chen et al., 2016] Chen, L., Cai, C., Chen, V., and Lu, X. (2016). Learning a hierarchical representation of the yeast transcriptomic machinery using an autoencoder model. *BMC Bioinformatics*, 17(1):S9.

[Colombo et al., 2012] Colombo, D., Maathuis, M. H., Kalisch, M., and Richardson, T. S. (2012). Learning high-dimensional directed acyclic graphs with latent and selection variables. *The Annals of Statistics*, pages 294–321.

[Conti et al., 2018] Conti, E., Madhavan, V., Such, F. P., Lehman, J., Stanley, K., and Clune, J. (2018). Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. In *Advances in Neural Information Processing Systems*, pages 5027–5038.

[Cooper, 1999] Cooper, G. (1999). An overview of the representation and discovery of causal relationships using bayesian networks in causation, prediction, and search. In *Computation, Causation, and Discovery*, pages 3–62. Menlo Park, Calif. : AAAI Press ; Cambridge, Mass. : MIT Press.

[Cooper et al., 2018] Cooper, G., Cai, C., and Lu, X. (2018). Tumor-specific causal inference (tci): A bayesian method for identifying causative genome alterations within individual tumors. *bioRxiv*, page 225631.

[De Jong, 2006] De Jong, K. A. (2006). *Evolutionary computation: a unified approach*. MIT press.

[Deng et al., 2014] Deng, L., Yu, D., et al. (2014). Deep learning: methods and applications. *Foundations and Trends in Signal Processing*, 7(3–4):197–387.

[Depeweg et al., 2016] Depeweg, S., Hernández-Lobato, J. M., Doshi-Velez, F., and Udluft, S. (2016). Learning and policy search in stochastic dynamical systems with bayesian neural networks. *arXiv preprint arXiv:1605.07127*.

[Eiben and Smith, 2015] Eiben, A. E. and Smith, J. (2015). From evolutionary computation to the evolution of things. *Nature*, 521(7553):476.

[Elidan and Friedman, 2005] Elidan, G. and Friedman, N. (2005). Learning hidden variable networks: The information bottleneck approach. *Journal of Machine Learning Research*, 6(Jan):81–127.

[Fahlman and Lebiere, 1990] Fahlman, S. E. and Lebiere, C. (1990). The cascade-correlation learning architecture. In *Advances in Neural Information Processing Systems*, pages 524–532.

[Friedman et al., 1997] Friedman, N. et al. (1997). Learning belief networks in the presence of missing values and hidden variables. In *International Conference on Machine Learning*, volume 97, pages 125–133.

[Frot et al., 2019] Frot, B., Nandy, P., and Maathuis, M. H. (2019). Robust causal structure learning with some hidden variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*.

[Glorot and Bengio, 2010] Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*. MIT press Cambridge.

[Goodfellow et al., 2014] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680.

[Harradon et al., 2018] Harradon, M., Druce, J., and Ruttenberg, B. (2018). Causal learning and explanation of deep neural networks via autoencoded activations. *arXiv preprint arXiv:1802.00541*.

[Hartford et al., 2017] Hartford, J., Lewis, G., Leyton-Brown, K., and Taddy, M. (2017). Deep iv: A flexible approach for counterfactual prediction. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 1414–1423.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

[Heinze-Deml et al., 2018] Heinze-Deml, C., Maathuis, M. H., and Meinshausen, N. (2018). Causal structure learning. *Annual Review of Statistics and Its Application*, 5:371–391.

[Hinton, 2012] Hinton, G. E. (2012). A practical guide to training restricted boltzmann machines. In *Neural Networks: Tricks of the Trade*, pages 599–619. Springer.

[Hinton and Salakhutdinov, 2006] Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

[Huang et al., 2017] Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4700–4708.

[Johansson et al., 2016] Johansson, F., Shalit, U., and Sontag, D. (2016). Learning representations for counterfactual inference. In *International Conference on Machine Learning*, pages 3020–3029.

[Kalainathan et al., 2018] Kalainathan, D., Goudet, O., Guyon, I., Lopez-Paz, D., and Sebag, M. (2018). Sam: Structural agnostic model, causal discovery and penalized adversarial learning. *arXiv preprint arXiv:1803.04929*.

[Ke et al., 2019] Ke, N. R., Bilaniuk, O., Goyal, A., Bauer, S., Larochelle, H., Pal, C., and Bengio, Y. (2019). Learning neural causal models from unknown interventions. *arXiv preprint arXiv:1910.01075*.

[Keedwell and Narayanan, 2005] Keedwell, E. and Narayanan, A. (2005). Discovering gene networks with a neural-genetic hybrid. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 2(3):231–242.

[Keedwell et al., 2002] Keedwell, E., Narayanan, A., and Savic, D. (2002). Modelling gene regulatory data using artificial neural networks. In *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290)*, volume 1, pages 183–188. IEEE.

[Kocaoglu et al., 2018] Kocaoglu, M., Snyder, C., Dimakis, A. G., and Vishwanath, S. (2018). Causalgan: Learning causal implicit generative models with adversarial training. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

[Kummerfeld and Ramsey, 2016] Kummerfeld, E. and Ramsey, J. (2016). Causal clustering for 1-factor measurement models. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1655–1664. ACM.

[Lagani et al., 2016] Lagani, V., Triantafillou, S., Ball, G., Tegner, J., and Tsamardinos, I. (2016). Probabilistic computational causal discovery for systems biology. In *Uncertainty in Biology*, pages 33–73. Springer.

[Lambiotte et al., 2008] Lambiotte, R., Delvenne, J.-C., and Barahona, M. (2008). Laplacian dynamics and multiscale modular structure in networks. *arXiv preprint arXiv:0812.1770*.

[LarrañAga et al., 2013] LarrañAga, P., Karshenas, H., Bielza, C., and Santana, R. (2013). A review on evolutionary algorithms in bayesian network learning and inference tasks. *Information Sciences*, 233:109–125.

[Le et al., 2011] Le, Q. V., Ranzato, M., Monga, R., Devin, M., Chen, K., Corrado, G. S., Dean, J., and Ng, A. Y. (2011). Building high-level features using large scale unsupervised learning. *arXiv preprint arXiv:1112.6209*.

[LeCun et al., 2015] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436.

[Lee et al., 2008] Lee, H., Ekanadham, C., and Ng, A. Y. (2008). Sparse deep belief net model for visual area v2. In *Advances in Neural Information Processing Systems*, pages 873–880.

[Lee et al., 2011] Lee, H., Grosse, R., Ranganath, R., and Ng, A. Y. (2011). Unsupervised learning of hierarchical representations with convolutional deep belief networks. *Communications of the ACM*, 54(10):95–103.

[Lehman et al., 2018] Lehman, J., Chen, J., Clune, J., and Stanley, K. O. (2018). Safe mutations for deep and recurrent neural networks through output gradients. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 117–124. ACM.

[Liang et al., 2019] Liang, J., Meyerson, E., Hodjat, B., Fink, D., Mutch, K., and Miikkulainen, R. (2019). Evolutionary neural automl for deep learning. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 401–409.

[Liang et al., 2015] Liang, M., Li, Z., Chen, T., and Zeng, J. (2015). Integrative data analysis of multi-platform cancer data with a multimodal deep learning approach. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 12(4):928–937.

[Liao and Poggio, 2016] Liao, Q. and Poggio, T. (2016). Bridging the gaps between residual learning, recurrent neural networks and visual cortex. *arXiv preprint arXiv:1604.03640*.

[Lopez-Paz et al., 2017] Lopez-Paz, D., Nishihara, R., Chintala, S., Schölkopf, B., and Bottou, L. (2017). Discovering causal signals in images. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017)*.

[Louizos et al., 2017] Louizos, C., Shalit, U., Mooij, J. M., Sontag, D., Zemel, R., and Welling, M. (2017). Causal effect inference with deep latent-variable models. In *Advances in Neural Information Processing Systems*, pages 6449–6459.

[Lu et al., 2018] Lu, S., Fan, X., Chen, L., and Lu, X. (2018). A novel method of using deep belief networks and genetic perturbation data to search for yeast signaling pathways. *PloS One*, 13(9):e0203871.

[Maathuis and Nandy, 2016] Maathuis, M. H. and Nandy, P. (2016). A review of some recent advances in causal inference. In Bühlmann, P., Drineas, P., Kane, M., and van der Laan, M., editors, *Handbook of Big Data*, pages 387–407. Chapman and Hall/CRC, Boca Raton, FL.

[Mandal et al., 2015] Mandal, S., Saha, G., and Pal, R. K. (2015). Neural network based gene regulatory network reconstruction. In *Proceedings of the 2015 Third International Conference on Computer, Communication, Control and Information Technology (C3IT)*, pages 1–5. IEEE.

[Miikkulainen et al., 2019] Miikkulainen, R., Liang, J., Meyerson, E., Rawal, A., Fink, D., Francon, O., Raju, B., Shahrzad, H., Navruzyan, A., Duffy, N., et al. (2019). Evolving deep neural networks. In *Artificial Intelligence in the Age of Neural Networks and Brain Computing*, pages 293–312. Elsevier.

[Murray-Watters, 2014] Murray-Watters, A. (2014). *The DM Algorithm: A Causal Search Algorithm for the Discovery of MIMIC Models, with an Attempt to Recover a Protein Signalling Network from a High-Dimensional Ovarian Cancer Dataset*. PhD thesis, MS thesis, Carnegie Mellon University.

[Murray-Watters and Glymour, 2015] Murray-Watters, A. and Glymour, C. (2015). What is going on inside the arrows? discovering the hidden springs in causal models. *Philosophy of Science*, 82(4):556–586.

[Narayanan et al., 2004] Narayanan, A., Keedwell, E. C., Gamalielsson, J., and Tatineni, S. (2004). Single-layer artificial neural networks for gene expression analysis. *Neurocomputing*, 61:217–240.

[Nguyen and Braun, 2017] Nguyen, P. and Braun, R. (2017). Semi-supervised network inference using simulated gene expression dynamics. *Bioinformatics*, 34(7):1148–1156.

[Pearl, 2018] Pearl, J. (2018). Theoretical impediments to machine learning with seven sparks from the causal revolution. *arXiv preprint arXiv:1801.04016*.

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

[Peters et al., 2017] Peters, J., Janzing, D., and Schölkopf, B. (2017). *Elements of causal inference: foundations and learning algorithms*. MIT Press.

[Raina et al., 2009] Raina, R., Madhavan, A., and Ng, A. Y. (2009). Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the 26th International Conference on Machine Learning*, pages 873–880. ACM.

[Ramsey and Andrews, 2017] Ramsey, J. D. and Andrews, B. (2017). A comparison of public causal search packages on linear, gaussian data with no latent variables. *arXiv preprint arXiv:1709.04240*.

[Rawal and Miikkulainen, 2018] Rawal, A. and Miikkulainen, R. (2018). From nodes to networks: Evolving recurrent neural networks. *arXiv preprint arXiv:1803.04439*.

[Real et al., 2018] Real, E., Aggarwal, A., Huang, Y., and Le, Q. V. (2018). Regularized evolution for image classifier architecture search. *arXiv preprint arXiv:1802.01548*.

[Salimans et al., 2017] Salimans, T., Ho, J., Chen, X., Sidor, S., and Sutskever, I. (2017). Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*.

[Sanchez-Vega et al., 2018] Sanchez-Vega, F., Mina, M., Armenia, J., Chatila, W. K., Luna, A., La, K. C., Dimitriadoy, S., Liu, D. L., Kantheti, H. S., Saghafinia, S., et al. (2018). Oncogenic signaling pathways in the cancer genome atlas. *Cell*, 173(2):321–337.

[Schmidhuber, 2015] Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117.

[Shalit et al., 2017] Shalit, U., Johansson, F. D., and Sontag, D. (2017). Estimating individual treatment effect: generalization bounds and algorithms. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3076–3085.

[Sober, 1998] Sober, E. (1998). Black box inference: When should intervening variables be postulated? *The British Journal for the Philosophy of Science*, 49(3):469–498.

[Soltoggio et al., 2017] Soltoggio, A., Stanley, K. O., and Risi, S. (2017). Born to learn: the inspiration, progress, and future of evolved plastic artificial neural networks. *arXiv preprint arXiv:1703.10371*.

[Spirtes et al., 2000] Spirtes, P., Glymour, C. N., and Scheines, R. (2000). *Causation, prediction, and search*. MIT press Cambridge.

[Spirtes et al., 1995] Spirtes, P., Meek, C., and Richardson, T. (1995). Causal inference in the presence of latent variables and selection bias. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 499–506. Morgan Kaufmann Publishers Inc.

[Srivastava et al., 2015] Srivastava, R. K., Greff, K., and Schmidhuber, J. (2015). Training very deep networks. In *Advances in Neural Information Processing Systems*, pages 2377–2385.

[Stanley et al., 2019] Stanley, K. O., Clune, J., Lehman, J., and Miikkulainen, R. (2019). Designing neural networks through neuroevolution. *Nature Machine Intelligence*, 1(1):24–35.

[Stanley and Miikkulainen, 2002] Stanley, K. O. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127.

[Such et al., 2017] Such, F. P., Madhavan, V., Conti, E., Lehman, J., Stanley, K. O., and Clune, J. (2017). Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *arXiv preprint arXiv:1712.06567*.

[Szerlip et al., 2015] Szerlip, P. A., Morse, G., Pugh, J. K., and Stanley, K. O. (2015). Unsupervised feature learning through divergent discriminative feature accumulation. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

[Tao et al., 2020] Tao, Y., Cai, C., Cohen, W. W., and Lu, X. (2020). From genome to phenome: Predicting multiple cancer phenotypes based on somatic genomic alterations via the genomic impact transformer. In *Pacific Symposium on Biocomputing*, volume 25, pages 79–90.

[van Dijk and Thierens, 2004] van Dijk, S. and Thierens, D. (2004). On the use of a non-redundant encoding for learning bayesian networks from data with a ga. In *International Conference on Parallel Problem Solving from Nature*, pages 141–150.

[Vohradsky, 2001] Vohradsky, J. (2001). Neural network model of gene expression. *The FASEB Journal*, 15(3):846–854.

[Weaver et al., 1999] Weaver, D. C., Workman, C. T., and Stormo, G. D. (1999). Modeling regulatory networks with weight matrices. In *Pacific Symposium on Biocomputing*, pages 112–123.

[Weinstein et al., 2013] Weinstein, J. N., Collisson, E. A., Mills, G. B., Shaw, K. R. M., Ozenberger, B. A., Ellrott, K., Shmulevich, I., Sander, C., Stuart, J. M., Network, C. G. A. R., et al. (2013). The cancer genome atlas pan-cancer analysis project. *Nature Genetics*, 45(10):1113.

[Wilamowski and Yu, 2010] Wilamowski, B. M. and Yu, H. (2010). Neural network learning without backpropagation. *IEEE Transactions on Neural Networks*, 21(11):1793–1803.

[Wilson et al., 2018] Wilson, D. G., Cussat-Blanc, S., Luga, H., and Miller, J. F. (2018). Evolving simple programs for playing atari games. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 229–236.

[Wong et al., 1999] Wong, M. L., Lam, W., and Leung, K. S. (1999). Using evolutionary programming and minimum description length principle for data mining of bayesian networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(2):174–178.

[Xin, 1993] Xin, Y. (1993). A review of evolutionary artificial neural networks. *International Journal of Intelligent Systems*, 8(3):539–565.

[Young et al., 2017] Young, J. D., Cai, C., and Lu, X. (2017). Unsupervised deep learning reveals prognostically relevant subtypes of glioblastoma. *BMC Bioinformatics*, 18(11):381.

[Zheng and Huang, 2018] Zheng, G. and Huang, T. (2018). The reconstruction and analysis of gene regulatory networks. In *Computational Systems Biology*, pages 137–154. Humana Press, New York, NY.