Deep Mining from Omics Data

Abeer Alzubaidi[1*] and Jonathan Tepper[2]

[1]School of Science and Technology, Department of Computer Science, Nottingham Trent University, Clifton Lane, Nottingham, NG11 8NS, United Kingdom, e-mail: abeer.alzubaidi@ntu.ac.uk (ORCID: 0000-0002-5977-564X), [2]Perceptronix Ltd, Hilton, Derbyshire, DE65 5AE, United Kingdom. (e-mail: jtepper@perceptronix.net), ORCID: 0000-0001-7339-0132. *[Abeer Alzubaidi] serves as corresponding author, Emails: abeer.alzubaidi@ntu.ac.uk (A. Alzubaidi)

Abstract

Since the advent of high throughput omics technologies, various molecular data such as genes, transcripts, proteins, and metabolites have been made widely available to researchers. This has afforded clinicians, bioinformaticians, statisticians and data scientists the opportunity to apply their innovations in feature mining and predictive modelling to a rich data resource to develop a wide range of generalisable prediction models. What has become apparent over the last 10 years, is that researchers have adopted deep neural networks (or 'deep nets') as their preferred paradigm of choice for complex data modelling due to the superiority of performance over more traditional statistical machine learning approaches, such as support vector machines. A key stumbling block, however, is that deep nets inherently lack transparency and are considered to be a 'black box' approach. This naturally makes it very difficult for clinicians and other stakeholders to trust their deep learning models even though the model predictions appear to be highly accurate. In this Chapter, we therefore provide a detailed summary of the deep net architectures typically used in omics research, together with a comprehensive summary of the notable 'deep feature mining' techniques researchers have applied to open up this black box and provide some insights into the salient input features and why these models behave as they do. We group these techniques into the following three categories: a) hidden layer visualisation and interpretation; b) input feature importance and impact evaluation; and c) output layer gradient analysis. Whilst we find that omics researchers have made some considerable gains in opening up the black box through interpretation of the hidden layer weights and node activations to identify salient input features, we highlight other approaches for omics researchers, such as employing deconvolutional network-based approaches and development of bespoke attribute impact measures to enable researchers to better understand the relationships between the input data and hidden layer representations formed and thus the output behaviour of their deep nets.

Keywords: Deep mining, Deep Learning, Omics Data, Interpretation, Explainable AI, Knowledge Discovery.

1. Introduction

The completion of the human genome project and the development of microarray technologies have contributed to heralding a new era of genome-based medicine that provided new insight into the complex genetic networks involved in the development of human diseases. Precision genome-based medicine aims to investigate and leverage various kinds of omics data for the development of a new generation of diagnostics and prognostics models and treatment personalisation that can maximise therapeutic benefits and minimise the risk of adverse drug reactions. In genotype-phenotype research studies, facilitating the subtle discrimination between different groups of biological samples, such as healthy/diseased, survival/death or poor/good prognosis is reliant on the recognition of key molecules that significantly exhibit the susceptibility to a disease or clinical outcome, and therefore, they can be employed in the construction of computational models for detecting diseases, monitoring its progression, or estimating susceptibility to treatment therapy. Knowledge discovery based on data mining methods has been incorporated in computational biology and bioinformatics for the extraction and interpretation of novel molecular markers. Although diverse methodologies and techniques from different fields have been merged for mining subtle patterns from omics data, the curse of dimensionality and the relative lack of data samples have significantly challenged those mining models to exploit the data and uncover significant knowledge. Therefore, retaining potential interesting complexity underlying complex data like omics requires sophisticated data mining techniques that focus on reducing the amount of spurious associations and false-positive predictions, whilst capturing what is general for detecting something meaningful about new unseen biological samples. The selection of methods for developing such intelligent systems demands raising the awareness of the key challenges, along with the deep understanding of the required computational and statistical resources to allow for more efficacious solutions to be developed.

This has motivated a significant amount of investigations to be conducted, which have demonstrated the superiority of artificial neural network methods

that form and utilise deep hierarchies of features automatically from the data (these methods are colloquially known as 'deep nets' or 'deep learning'). As will be made clear in this section of the book, such deep nets provide significant advantages over their classical 'shallow' machine learning counterparts (such as Support Vector Machines and Logistic Regression) across a wide range of problem domains [4,11,38,60]. However, the fundamental issue with the deep learning paradigm is that they lack inherent transparency due to the original input being subject to deeply nested nonlinear transformations and are thus considered to be a 'black box' approach. Subsequently, the strength of the deep learning paradigm also appears to be its weakness. There appears to be little consensus in the literature as to how to interpret these complex internal representations underlying the mapping between the input and output layers and therefore explain why these models predict/behave as they do. This makes it very difficult for clinicians and other stakeholders to trust their deep learning models even though the model predictions appear to be highly accurate. Since 2017, a number of notable innovations within the medical and bioinformatics fields, including cognate disciplines such as omics, have sought to decode the 'black box' of these deep learning models to understand what their hidden layers are encoding. This process of decoding the block box in order to make interpretations and inferences regarding the salient input features and reasons for the model's responses can be referred to as *deep mining*. Deep mining essentially refers to the process of extracting meaningful knowledge embedded within the hidden representations automatically formed within a deep network, in order to understand the decisions or responses produced by the model.

The chapter is structured as follows: we first begin with an overview of the fundamental concepts of omics data, focusing on omics data types and providing a summary of omics data sources. Deep Mining is then presented in detail, beginning with a comprehensive overview of the deep learning models currently used in omics research today. We then concentrate of the key aspect of this chapter, which is consideration of the notable deep mining techniques researchers have developed in a bid to open up the so-called 'black box' of deep nets. These techniques principally attempt to discover various ways in which to

sensibly interpret the hidden representations formed within deep nets, thus helping to provide insight into the salient input features identified and decisions made by these models. Where possible, we contextualise the discussion of these techniques to current bioinformatics research, including omics. Finally, we conclude with a discussion to provide readers with some insight into how these deep feature mining or 'knowledge extraction' techniques have been embraced by the omics community and areas for the further exploration.

2. Omics Data

With the advent of high throughput omics technologies, various molecular data have been provided, such as genes, transcripts, proteins, and metabolites. Recently, microarray technologies simultaneously measure genetic variants between individuals and the sequencing of thousands of genes to reveal if there is any abnormality that is associated with a trait [47]. Single Nucleotide Polymorphisms (SNPs) can be considered the most essential source for the variance in the genetic information (DNA) between individuals, and a SNP is a variation at a single DNA site [31]. The microarray can be considered the most commonly used technology for transcriptome analysis to measure mRNA and summarises the actively expressed genes. Microarray-based gene expression profiling technologies have been widely exploited for mining and classifying cancer data to support making accurate and reliable clinical decisions. For instance, van't Veer et al. estimated the clinical outcome of breast cancer using gene expression profiling [96].

Moreover, proteomics and metabolomics areas have also the potential to extract novel molecular signatures from cancer data (e.g., [2]). Proteomics is another interesting area of research that measures all proteins expressed in a cell at a given time or under specific biological conditions and can provide more comprehension to the complex biological procedures due to its direct role in cell physiology [10]. The proteome can be considered a reflection of genomic and environmental factors, therefore, it may hold a promising piece of knowledge, which can address different biological questions of interest [87]. On the other

hand, the metabolome is the outcome of integrating the transcriptome and the proteome, thus, changes in the metabolome are related to changes in this product [94]. The metabolomics area investigates the products of metabolism, such as sugars, lipids and amino acids.

Omics fields of proteomics and metabolomics have similar statistical and computational challenges with the main omics fields of genomics and transcriptomics. The development of the breakthroughs for addressing omics data challenges and the discovery of a good representation of the input data is at the core of personalised and precision medicine. Omics technologies not only have helped to improve our understanding of the physiological system by detecting genomics, transcriptomics, proteomics and metabolomics in a specific biological sample [53], but also offer more comprehension to processes that underlie a disease, and thus expanding our knowledge about the aetiology of complicated diseases, such as cancers and diabetes. Therefore, such data could play a vital role in developing precision diagnostic and prognostic models, and the potential of constructing such risk prediction systems that outperform conventional models increases by boosting the capacity of extracting novel omics variants.

2.1. Omics Data Resources

The recent availability of a wide range of molecular data as well as individual patient characteristics brings tremendous opportunities for clinicians, bioinformaticians, statisticians and data scientists to benefit from this abundance of biomedical data to advance health care research by developing multi-modal high-throughput biomedical systems [6]. For instance, the availability of data repositories such as The Cancer Genome Atlas (TCGA) [100], which is a collaboration between the National Cancer Institute (NCI) [1] and the National Human Genome Research Institute (NHGRI) [2], that fundamentally aims to understand the molecular basis of cancers, through the exploitation of omics

[1] https://www.cancer.gov/
[2] https://www.genome.gov/

analysis technologies. The TCGA is considered one of the largest biomedical data repositories, containing diverse types of data like somatic mutation, copy number, gene expression, miRNA expression, DNA methylation, reverse protein phase array and clinical information for more than 10,000 samples derived from 33 types of cancer with known response groups. The availability of these vast amounts of omics data not only demands sophisticated data mining methods but also flexible infrastructures and efficient tools to use and link these data resources. Therefore, several web portals and communication platforms were originated to support researchers and graduate students to access and employ these biomedical datasets and minimise the complexity of accessing these information resources and allow for diverse analysis and visualisation services to be utilised. For instance, cBioPortal [23,33] is an open-access repository developed by investigators at Memorial Sloan Kettering Cancer Center (MSKCC)[3] for a wide range of cancer omics datasets. cBioPortal provides a flexible platform to many data sets and easy-to-use visualization analysis services.

Besides the need for transparent and flexible bioinformatics software and platforms to share, use and benefit from the wealth of such data, contributions from the research community of computational biology and bioinformatics are also required to advance omics data mining. Mining novel information from any of these omics data types necessitates disentangling high-level abstracted genomic, transcriptomic, proteomic or metabolomic patterns that play an important role in the evolution of disease-based phenotypes. Those generic determinants can be considered key factors in the development of generalisable risk assessment systems for complex diseases, such as cancers or in pharmacogenomics for assessing the efficacy of drug therapies. Furthermore, proposing advanced data mining techniques for disentangling meaningful information from omics data has the potential not only to support the decision-making process in clinical practice but also to enable innovative opportunities for conducting clinical and translational research, along with the potential benefit of

[3] https://www.mskcc.org/

reducing healthcare costs. Therefore, in the next section, we introduce deep mining for data-driven molecular biology.

3. Deep Mining

Deep mining refers to the integration of deep learning technologies into an organisation's workflow processes, tools and techniques for extracting information or knowledge latent across disparate data sources to inform future decision making. In the context of mining omics data, one may want to determine the function of different protein folds/structures (e.g., [5]) or identify robust biomarkers associated with particular diseases, such as breast cancer (e.g., [7]); or classify genomic sequences according to the transcription factor binding site (e.g., [14]). Deep learning is a general term that refers to the development, optimisation and application of artificial neural networks [38, 75] that possess many hidden layers of nonlinear processing elements that transform the original omics input data into hierarchical abstract representations well-suited to the omics task at hand. These type of networks are referred to as 'deep neural networks' (or more simply, 'deep nets') and with their hierarchical abstract representations performing a type of automatic data pre-processing, have provided cutting-edge performance and insights into a variety of omics-related problem domains (e.g., [9,55,69,111]). Subsequently, researchers have rapidly embraced the 'deep net' paradigm as a mainstream approach to modelling and analysing omics data, with the number of omics-focused deep learning publications increasing exponentially since 2010 [110][4].

Whilst deep learning approaches to omics problems has produced superior performance over popular statistical machine learning methods, such as support vector machines [4], they lack inherent transparency and are considered to be a 'black box' approach. Subsequently, the strength of the deep learning paradigm

[4] We direct the reader to the recent survey and guidelines produced by Zhang et al [110] for a more detailed systematic review of deep learning technologies (including available software tools), their application to omics data and good practice guidelines.

also appears to be its weakness. There appears to be little consensus in the literature as to how to interpret these complex internal representations underlying the mapping between the input and output layers and therefore explain why these models predict/behave as they do. This makes it very difficult for clinicians and other stakeholders to trust their deep learning models even though the model predictions appear to be highly accurate. Since 2017, a number of notable innovations within the medical and bioinformatics fields, including cognate disciplines such as omics, have sought to decode the 'black box' of these deep nets to understand what their hidden layers are encoding. These methods range from a) hidden layer visualisation and interpretation (e.g., [7,29,37,56,91,95]); b) feature importance and impact evaluation (e.g., [89,90]); and c) output layer gradient analysis (e.g., [85,86]).

We, therefore, begin in the next section with a comprehensive overview of the deep learning models currently used in omics research today, and then we describe the most prominent techniques used by researchers for extracting useful knowledge from the 'black box' of these deep learning technologies.

3.1. Deep Learning

The most common type of artificial neural network used in deep mining, is the multi-layered perceptron (MLP) network which consists of relatively simple, neuron-like processing elements that are arranged in layers and connected to other neuron-like elements within subsequent, previous and/or the same layer(s). These processing elements are referred to as artificial neurons and are at the core of all deep neural network models. Figure 1 shows a schematic of a typical artificial neuron. It can be seen that the neuron accepts input stimuli, such as pixels from an x-ray image or a numerical vector representing a set of genes, on its input layer (x_1 to x_n) and a separate weight value (w_1 to w_n) is assigned to each input unit to indicate the relative importance of the input during the calculation of the neuron's output. A linear combiner function, such as the dot product, is applied to the subsequent input and weight vectors to produce an overall net input. To determine the actual output (or activation) of the neuron,

this net input is passed through an activation function[5]. Whilst the activation function can be a simple

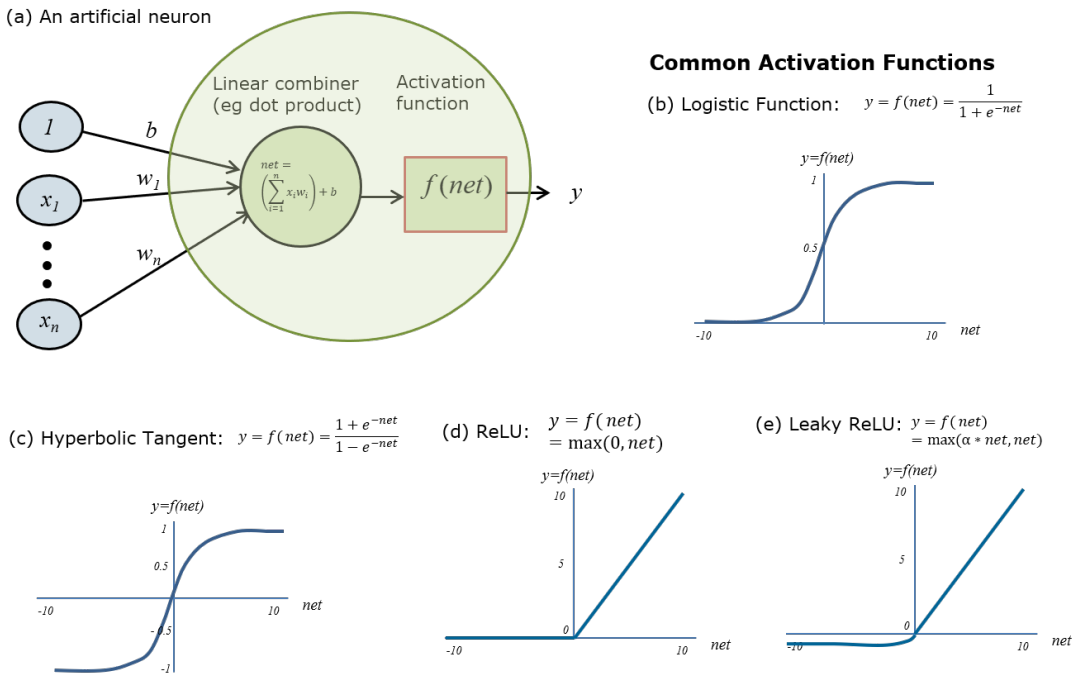


Figure 1: An artificial neuron or 'perceptron'.

identity function (i.e., returning the net input value) or a hard limiting function such as a step function (i.e., returning −1 if 0 or below, otherwise (1), it is most commonly a nonlinear function with the following properties: *output varies nonlinearly with the weighted input, it is continuous, monotonic and everywhere differentiable*. A range of common activation functions are shown in Figures 1 (b) to (e), with the *binary sigmoid* function squashing the net input to a continuous value between 0 and 1; *hyperbolic tangent (tanh)* function squashing to a

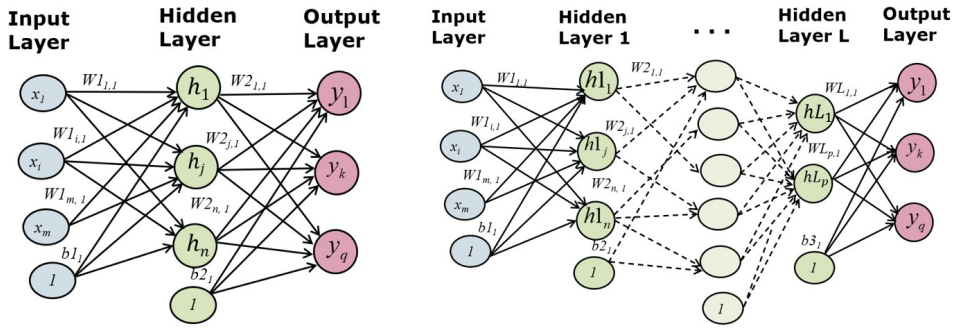
[5] As shown in figure 2.1(a), an additional 'bias' weight, b, associated with a single unit set to a constant value of 1, is assigned to each processing element within the hidden and output layers of an MLP. The bias weight allows the activation function to 'move' horizontally across the net input space (x−axis); whereas the standard weights w1...wn transforms the shape and direction of the activation function to enable it to pivot around the horizontal point, therefore allowing for more adaptive and effective learning [78].

continuous value between −1 and 1 and finally, the *rectified linear unit (ReLU)* and *leaky ReLU* functions providing no upper bounds or 'saturation' on the positive scale [66].

As shown in Figure 1 (a), a simple MLP network typically has at least three layers, an input layer, to receive external input signals from the environment; a hidden layer to represent a learned set of transformations (or features) of the input layer, and finally, an output layer to represent the MLP's response to the environment. A critical requirement for all MLP networks is that the neurons within the hidden layer use nonlinear activation functions to allow for useful 'problem solving' representations to be learned. The weights to a hidden unit represent a transformation of the input pattern whereas the activation of the hidden unit represents its response to that transformation. Unlike the hidden layer, the output layer is permitted to use either linear or nonlinear activation functions. An MLP is said to be a 'deep net' when there is more than one hidden layer (of nonlinear processing elements) separating the input and output layers, where each hidden layer transforms activations from the previous hidden layer to create a more abstract representation of the original input signal (see Figure 2 (b)). When many hidden layers are stacked together in this way, a hierarchical feature representation of the original input vector is formed. It is important to note that the weighted values (or connection strengths [6]) joining neurons together represents the MLP's long-term knowledge (affecting all input pattern responses) whereas a hidden or output neuron's activation represents the MLP's short-term knowledge (referring to a single input pattern response). Also, activation information within an MLP typically flows from the input layer, transformed through the hidden layer(s) and then finally integrated and transformed through the output layer (forming a feed-forward MLP or FF-MLP architecture). However, in the recurrent neural network paradigm (discussed later), activation flow is permitted backwards to the same or preceding layers to create complex internal memory-based systems [67,91].

[6] The terms weights, weighted values, connections and connection strengths will be used synonymously throughout this section.

Deep learning occurs when general recursive rules are applied to adapt the connection strengths between these units when provided with examples of input features and desired output response patterns. The most notable learning algorithm used in deep learning models to-date is variants of the back-propagation learning rule, popularised by [76] in 1986. Back-propagation is a form of stochastic gradient descent that uses the chain rule to efficiently compute the gradient of the loss function, E, with respect to the unit weights, W, and bias weights, b, of the network, enabling the weights to be adapted in a way that minimises the overall error (difference between the actual and desired response) of the system,



(a) Shallow MLPs have a single hidden layer

(b) Deep MLPs have many hidden layers (all connections not shown)

Figure 2: Shallow and deep feedforward multi-layered perceptrons (FF-MLPs).

$E(W,b)$ (see [17,70]). FF-MLPs trained with back-propagation have been theoretically proven to be able to learn any complex learnable function [49], although the practical reality is much different.

At this juncture, it is important to clarify that the back-propagation learning rule can be used for both supervised learning and a form of unsupervised learning. In supervised learning, we provide the MLP with many 'input feature vector' and 'desired output response vector' pairs (e.g., a gray-scale image of a chest x-ray given as an input vector and a disease classification, such as cancer or pneumonia, as the desired output vector) and require back-propagation to learn this mapping with a view to correctly classifying the future unseen input feature

vectors. In this type of learning, we might want to perform classification, where the output vector represents a categorical variable (e.g., disease-type with mutually exclusive values of tumour, edema, pneumonia) or prediction, where the desired output vector represents a continuous variable (e.g., triglyceride concentrations or IgG serum immunoglobulin levels in response to COVID-19). With respect to unsupervised learning, we provide the MLP with just the input feature vectors and require it to reproduce them on its output layer (a task referred to as auto-association), again using back-propagation. When an MLP is used in this manner it is referred to as an Auto-Encoder [44].

3.1.1. Auto-Encoder

The purpose of Auto-Encoder is to perform dimensionality reduction and/or feature learning by typically restricting the size of the hidden layer to have fewer units than the input layer, forming an undercomplete representation that captures the salient features of the input data[7] [12,42]. Figure 3 illustrates a deep Auto-encoder network showing the two unit code or 'bottle-neck' layer, representing a highly reduced description of the original five unit input layer. Hidden layer units and weights to the left of the code layer are referred to as the 'encoder' layers of the network, responsible for transforming the input signal in a useful way; whilst hidden layer units and weights to the right of the code layer, refers to the 'decoder' layers of the network as they're responsible for recovering the original input vector from the code layer, by 'decoding' the hidden unit transformations made in the encoder layer(s). A shallow Auto-encoder network is simply an Auto-encoder with a single hidden layer that forms an undercomplete representation of the input vector.

Before the landmark papers by Hinton et al [42] and Bengio et al [12], learning useful representations within deep nets consisting of two or more hidden layers

[7] The purpose of the hidden layers within an Auto-encoder is not to reproduce a precise copy of the higher dimensional input vector on the output layer. Rather it is meant to be a rough approximation (within an allowable error tolerance) that is less sensitive to variations within the training data, thereby filtering out noise and forcing the network to concentrate on covariances within the input data by nonlinear projections onto a lower dimensional space – akin to a form of nonlinear principal component analysis.

proved notoriously difficult with back-propagation. The three main reasons for this were: a) random weight initialisations typically resulted in poor solutions [42]; b) the back-propagated error gradients vanished exponentially as they propagated from the output layer back to the input layer, such that only the weights of the hidden layers close to the output layer experienced meaningful adaptation, and little adaptation or learning took place at the weights nearest the input layer [13,45]; c) training times were intractable for networks with large numbers of connections (e.g., 100,000+) due to both the need of larger data sets to constrain the weights and the limitations of mid-range to high-end desktop computers and workstations at that time, such as those with Intel Core2 Duo and Xeon processors, where training times could exceed weeks or months for

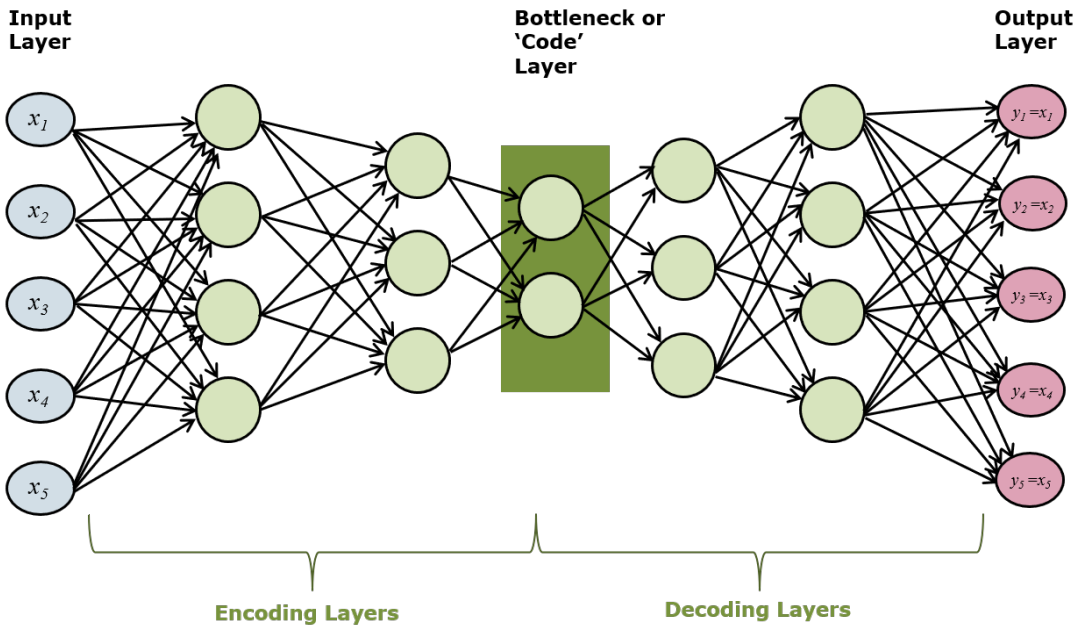


Figure 3: Deep auto-encoder (DAE) network (bias weights not shown)

realistic data sets[8].

[8] For example, training an MLP with back-propagation for 100,000 epochs to learn the MNIST hand-written digits benchmark data sets [61], consisting of 60,000 images for training and 10,000 for testing, would not be feasible on such multi-core serial processors) due to insufficient memory and/or lengthy training times (months) caused by intractably slow processing [28].

The first two issues were primarily solved by decomposing the task of training a deep net into two basic stages: i) unsupervised pre-training of each hidden layer, one at a time, using Auto-Encoders in a recursive manner to ensure input information is maintained throughout the evolving network; and ii) supervised fine-tuning where the final 'stacked hidden layers of the individual Auto-encoders' is augmented with an output layer, typically representing a classification or prediction task, using back-propagation or some other method of gradient descent to minimise the desired cost function. The unsupervised pre-training procedure is referred to as a greedy layer-wise approach as every additional hidden layer is realised by training an Auto-Encoder with a single hidden layer that tries to reconstruct its input. Whilst the stacked Auto-Encoder shown in Figure 4 is based on feed-forward MLP models trained with back-propagation, the same type of stacked model can be successfully realised with Restricted Boltzmann Machines

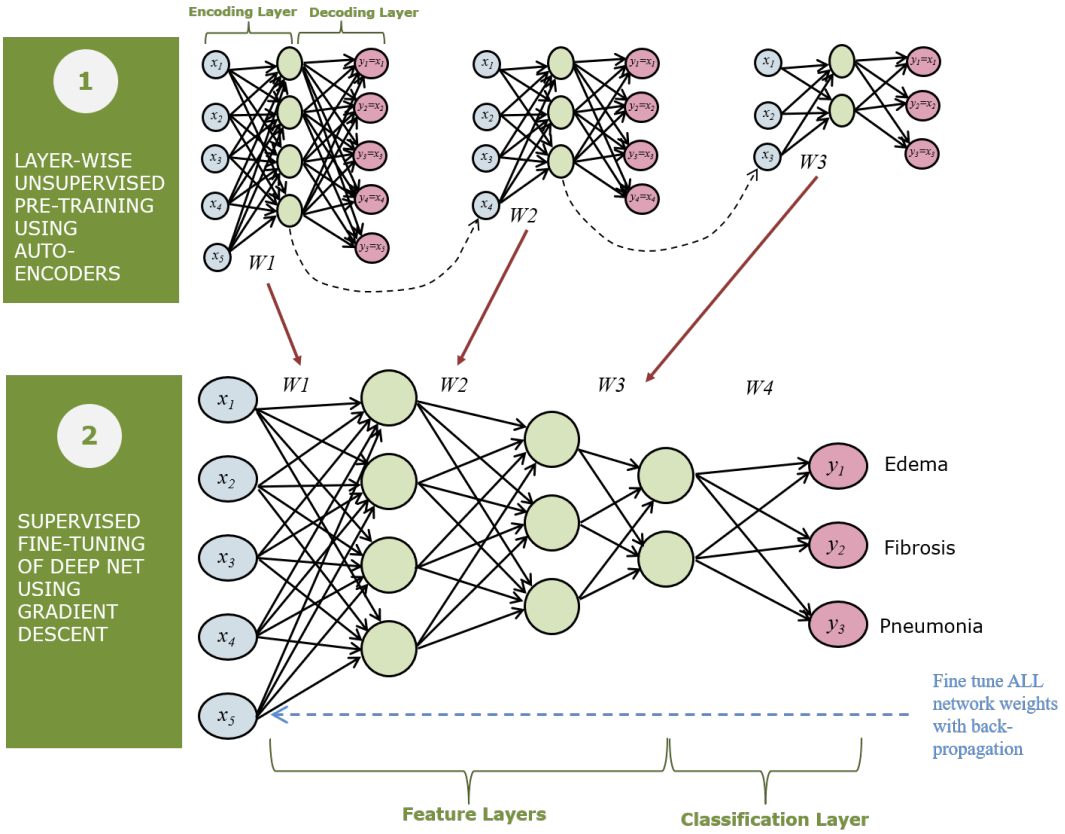


Figure 4: Layer-wise training of deep nets using recursive Auto-encoding (bias weights not shown) (RBMs) [42,43].

3.1.2. Restricted Boltzmann Machines

The Restricted Boltzmann Machines (RBMs) architecture consists of an input or 'visible' layer and a hidden feature layer connected together with bi-directional symmetric links and no links amongst nodes within the same layer. Rather than using the back-propagation rule, the symmetrical weights are adjusted using a probabilistic energy-based learning rule such as 'contrastive divergence' which utilises Gibbs Sampling within a gradient descent procedure [21]. Hinton et al [42] and Bengio et al [12] comprehensively demonstrated that RBMs can be recursively stacked in a similar way to that shown in Figure 4 to form a visible layer connected to a deep network of many hidden feature layers augmented with an output layer to form so-called the Deep Belief Network whose weights are finetuned with the back-propagation learning rule. Whilst there has been a number of successful applications of Deep Belief Networks to omics data [109] this class of model is much more computationally intensive than their MLP counter-parts and less suited to large data sets [64]. More generally, the computational efficacy of forming a deep net in this recursive layerwise-manner (using either MLP- or RBM-based Auto-Encoders) was cogently argued in [12,32,62,98] with empirical evidence demonstrating that these models avoid getting stuck in the typical poor random starting conditions associated with random initialisations.

The third issue preventing the wide-spread use of deep nets pre 2010, long intractable training times using mainstream desktop computers, was overcome by the increasing availability of Graphical Processing Units (GPUs). For example, Ciresan et al [28] demonstrated that a very large MLP network (with up to 9 hidden layers with 10 to 2500 neurons per layer; and between 1.34 and 12.11 million adjustable parameters; scaled tanh activation functions) could be trained with simple back-propagation without the greedy layer-wise approach to successfully classify handwritten digits from the famous MNIST benchmark

dataset [61]. The model was successfully trained with 60,000 images using an nVidia GTX280 over a period of 114.5 hours. The GTX280 augments a PC with an additional 240 cores and 622 GFLOPS for floating point computations (as opposed to 6 to 8 cores and up to 72.51 GFLOPS for typical Xeon processors of the time). Ciresan et al [28] reported a 0.35% error rate across 10,000*test* images surpassing the greedy-layer wise approaches by Simard et al [84] (who reported an initial state-of-the-art error rate of 0.4%) and Ranzato et al [62] (who reported a 0.39% error rate). Arguably the most significant step change in terms of the size of deep network that can be successfully trained to produce state-of-the-art performance was that demonstrated by Krizhevsky et al [58] with their AlexNet program for classifying 1.2 million high resolution images from the ImageNet ILSVRC benchmark dataset (with a 1000 different output classes). The network architecture was a deep convolutional neural network (initially designed for image classification and discussed further below) consisting of 60 million adjustable parameters, 650,000 neurons derived from: an input layer of 150,528 dimensions; 7 hidden 'convolutional' or 'pooling' layers of between 4,096 and 253,440 neurons each; and a 'softmax' output layer of 1,000 neurons representing the possible different pattern classes. As with Ciresan et al [28], Krizhevsky et al [58] did not use greedy layer-wise training; however, they abandoned the tanh activation functions used by Ciresan et al [28], in favour of the non-saturating ReLU units as defined by [66] (see Figures 1 (d) and (e)); reporting accelarated learning up to seven times faster than with tanh. Also, to avoid over-fitting, the 'drop out' approach to regularisation was adopted (where random subsets of hidden nodes are switched off across the network during learning. In order to train their network, Krizhevsky et al [58] spread their deep net across two nVidia GTX580 GPUs[9] in parallelised scheme, ensuring each GPU processes roughly half of the neurons each. Subsequently, the network took 5 to 6 days to learn 1.2 million training images and was tested on 150,000 test images yielding a state-of-the-art error rate of 37.5% (17% better than the nearest rival

[9] The nVidia GTX580 GPU provides up to 512 cores and between 197.6 GLOPS and 1.581 TFLOPS floating point compute power.

in the ImageNet LSVRC-2010 contest); a revised model submitted to the ImageNet LSVRC-2012 contest reported a winning test error rate of 15.3% (compared to 26.2% achieved by their nearest rival) highlighting the superiority in performance of this model class.

It is clear from the above, that for omics researchers to leverage the full benefit of the representational power of deep neural networks trained with backpropagation, they must at least make the following design considerations:

(a) Incremental or holistic network construction. As indicated above, there is evidence to support both the incremental greedy layer-wise approach to constructing a deep neural network and the simple approach of training a very deep network using back-propagation with substantial data and computing resources. However, if a greedy layer-wise approach is not applied, there is an important need to consider whether non-saturating activation functions are essential in this context given the research by Krizhevsky et al [58]), He et al 2015[10] [40] with their 'resnet' model of residual functions, and more recently, Ramachandran et al [74] from Google, who introduced so-called swish activation function [11] and demonstrated that it worked better than ReLU on deeper models across a number of challenging datasets.

(b) Suitable architecture and learning algorithm. Whilst MLP-based deep neural networks trained with back-propagation appear to show much success, state-of-the-art performance with deep nets has mostly been reported with convolutional neural networks for image recognition (as discussed above) and more recently, tasks relating to omics data [3,59,65,72,72,93]. The 'best' choice as to which type of network therefore

[10] Residual functions make stronger reference to layer inputs by allowing the architecture to include 'jump connections' where the input from one layer can be provided as input to a hidden feature layer n layers ahead, where $n > 1$. This is represented in an equation as $F(x) = H(x) - x$ where $H(x)$ represents the nonlinear hidden mapping we want to learn with input x through an array of stacked nonlinear hidden layers. $H(x)$ is then reformulated as $F(x) + x$, where $F(x)$ and x represent the stacked nonlinear layers and identity function respectively.

[11] Which is simply $f(x) = x \cdot \text{sigmoid}(x)$.

remains open and subject to empirical exploration and determination; as are the choices for critical hyper-parameters such as the number of hidden layers, types of hidden layer (e.g., normal flat 1D array or 2D feature maps used in convolutional neural networks), the number of hidden units per layer and the connection patterns between layers.

(c) Software architecture and hardware platform. In order to realise their research goals with deep nets, omics researchers must consider the most appropriate software framework and tools for designing and implementing their experiments on systems endowed with GPUs. Popular open source software tools such as Caffe [52], TensorFlow [1] and PyTorch [68] have been notably recommended for Omics researchers wanting to apply large deep neural networks across GPUs [64,110].

3.1.3. Convolutional Neural Networks

Given the rising importance of Convolutional Neural Networks (CNNs) across a range of problem domains, including omics, this section concludes with a brief overview of this particular paradigm together with that of Long short-term memory (LSTM) models, a popular type of recurrent neural network, which is also seeing increased popularity within the omics field and there is a high likelihood of the need to also mine feature information from these complex models.

CNNs are a class of artificial neural networks designed by LeCunn et al [61] to tackle the problem of recognising images of two dimensions (2D) in a more robust and reliable way (e.g., recognising a particular object appearing within a series of images at varying orientations and resolutions). As shown in Figure 5, the CNN architecture consists of two interconnected parts: i) a series of feature extraction layers and ii) a series of classification layers. The feature extraction layers consist of: the input layer, containing a matrix of pixels from a 2D image of interest; one or more convolutional layers, where each convolutional layer represents a feature map formed by passing a small receptive field over the original input and

applying a convolutional operator; and a pooling layer consisting of one 'pooled feature map' per feature map in the convolutional layer where each pooled feature map is derived from a smaller receptive field (e.g., 25% or 50% the size of the convolutional feature map) passed over the convolutional feature map where either an averaging or maximum operation is performed. Figure 5 provides an illustration of the both the convolutional operator and pooling operator that are used in the feature extraction layers. The first classification layer is connected to the last pooling layer where these reduced feature maps are transformed into a simple 1D vector of activations, which are then fed forward through a series of standard hidden layers to either a softmax output layer for classification or a set of linear activation functions for prediction. The back-propagation learning rule and its variants can then be applied to adjust the weights according to the derivative of the cost function.

In essence, the feature extraction layers achieve effective detection and representation of invariant features through the use of local receptive fields, shared weights and spatial sub-sampling. The use of local receptive fields was inspired by the research of Hubel and Wiesel (1962) [50] relating the cat's visual system[12]. Each single feature map has its own weight matrix (called a filter) and those weights are shared as its local receptive field (i.e., a square subset of input features from the layer before) shifts around the input image one positional change

[12] Hubel and Wiesel's discovered that the cat's visual system had locally-sensitive, orientation-selective neurons.

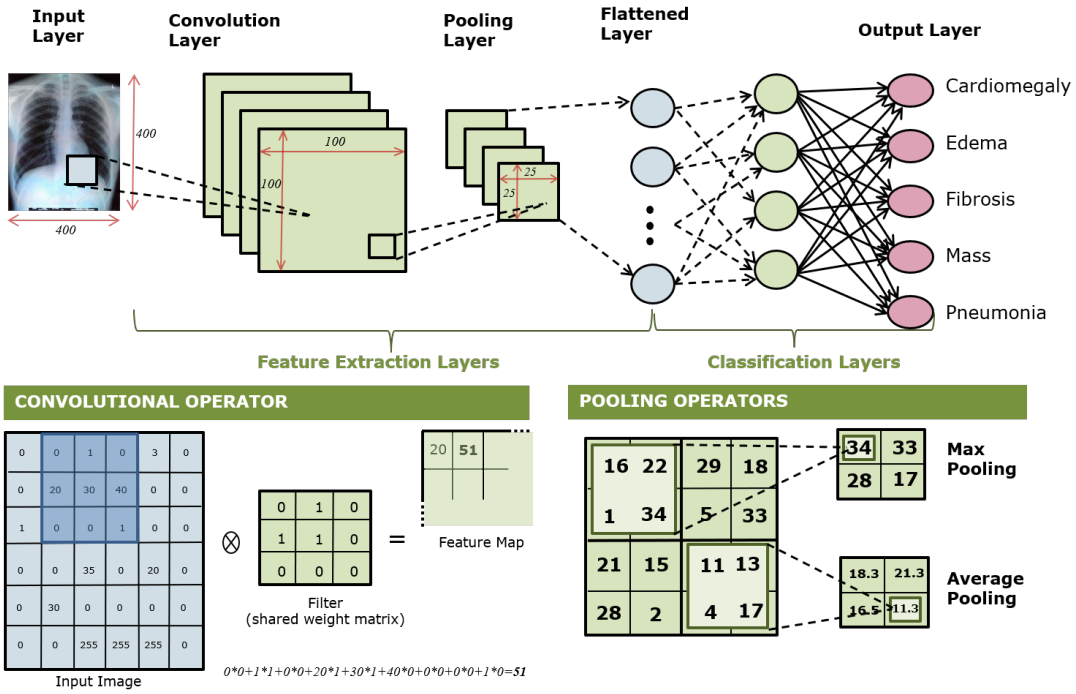


Figure 5: Convolutional neural network.

at a time, performing a convolutional operation at each step. The feature map responses to an input is passed through the ReLU (or similar) nonlinear activation function. A feature map will eventually learn to recognise a specific feature at varying locations within an image, e.g., line end-points or object corners. These low-level features are then combined by the subsequent layers to detect higher order features. As a complete convolutional layer is composed of many feature maps (each with their own different weight vector), multiple features can be extracted at each location. To maximise the probability that an extracted feature is position invariant, the pooling layer is utilised, which is a sub-sampling layer that reduces the spatial resolution of the feature map its associated with and performs a local averaging or maximisation of the feature map (as seen in Figure 5). This creates a very sophisticated feature detection mechanism that appears to be far superior to the hidden layer representations form within standard Deep MLP networks, and likely responsible for its state-of-the-art performance in a variety of omics tasks (as indicated above). Omics data

is typically non-image data, therefore for CNNs to be used, the omics data of interest must be transformed into a useful 2D form as demonstrated by Min et al [65] with their DeepEnhancer model for classifying enhancers from background genomic sequences; and more recently, Sharma et al [82] with their DeepInsight model which utilises CNNs to study 6216 samples of 60,483 genes from RNA-seq or gene expression TCGA dataset (https://cancergenome.nih.gov) to predict ten types of cancer.

The deep nets discussed above would have significant difficulty in processing omics data that is temporal in nature i.e., a sequence of chronologically ordered observations, such as genes, where each observation is typically dependent indirectly or directly on the preceding observation. This is due to the lack of memory faculty within its architecture to relate one observation to another. The final type of deep neural network discussed in this section is therefore the Long Short-term Memory (LSTM) model [46], Gers et al [35] from the recurrent neural network paradigm.

3.1.4. Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are typically adaptations of the traditional feed-forward MLP trained with back-propagation or its variants. RNNs extend the feed-forward MLP architecture to include recurrent connections that allow network activations to feedback as inputs to units within the same or preceding layer(s). Units that receive feedback values are often referred to as context, memory or state units. Such internal memories enable the RNN to then construct dynamic internal representations of temporal order and dependencies that exist within the data (e.g., the formation of syntactic structure or semantic meaning of a sentence as it is presented one word at a time). An RNN processes a sequence of observations (e.g., genes), one at a time, calculates unit responses to each observation and the subsequent error of its response (with respect to some desired target value and cost function) and then calculates the resulting weight changes for each observation. A variant of back-propagation called backpropagation through time (BPTT) [101,102] is commonly applied to ensure

the resulting weight changes are with respect to a sequence of observations rather than a single observation within it (achieved by sharing the same weight values across an input sequence which is akin to unrolling the network over-time to per-

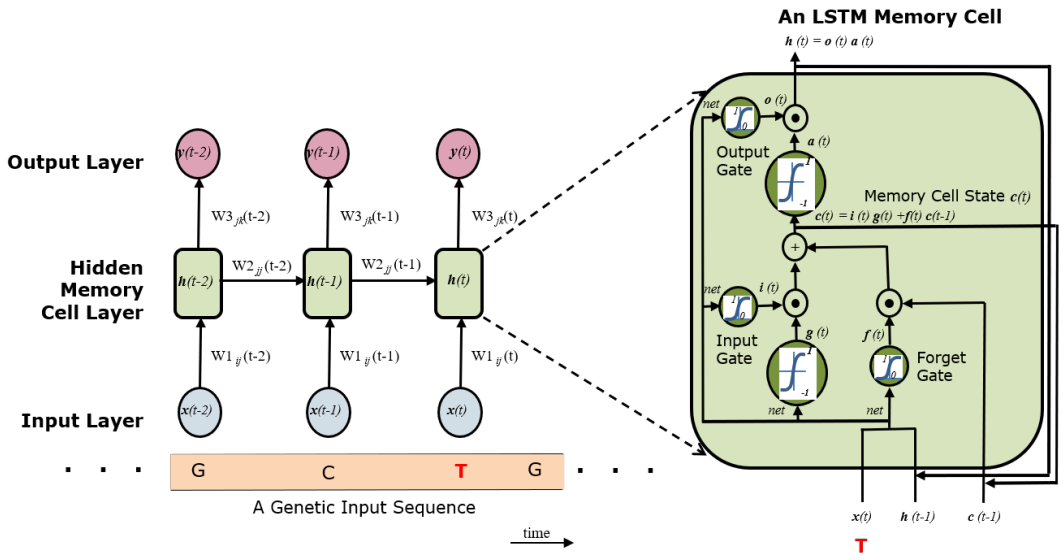


Figure 6: A Long Short-term Memory (LSTM) recurrent neural network.

form the above calculations and then re-rolling the network back up to perform the weight changes). These properties have led to wide appeal of RNNs for modelling time-series data [15,20,30,34,67,91,92]. The most common form of RNN used for omics data is the LSTM model and Figure 6 depicts a schematic of the model's architecture. As can be seen, the LSTM architecture has been unrolled over three time steps, processing three genes within a four gene sequence (*G*, *C*, *T*, and *G*, with *T* being the observation being processed). It can be seen that the hidden layer response to *G* at time-step $t-2$, is fed as input to the hidden layer at time-step $t-1$, with the corresponding gene input *C*; likewise the hidden layer response for gene input *C* is also passed as input to the hidden layer at time *t*, together with gene *T* and so on. What is notably different about the LSTM model is the use of memory cells or blocks rather than units for its hidden layer. Each memory block utilises multiple activation functions whose input is controlled by

three gating mechanisms (input, output and forget gates) to enable the LSTM to retain only important input information over time, whilst forgetting less important inputs.

A detailed treatment of the LSTM memory mechanism is given by Hochreiter and Schmidhuber [46] and Gers et al [35]. In terms of recent application to omics data, Sekhon et al [79] used a hierarchy of LSTMs in their DeepDiff model to model how various histone modifications cooperate automatically in their endeavour to predict differential gene expression from histone modification signals to better understand the functional heterogeneity of cells; Karimi et al [54] presented their DeepAffinity model which integrated CNNs with an RNN containing gated recurrent units [26], an LSTM of reduced parameter and gating complexity, to successfully predict compound–protein affinity from sequences alone with high applicability, accuracy and interpretability to aid drug discovery; more recently, Chung et al [27] used an LSTM to predict measurements of proteins and metabolites as part of their deep hybrid system to perform unsupervised classification of proteins and metabolites in mice during cardiac remodeling;

4. Deep Feature Mining Techniques

As mentioned in the introduction to deep mining section, whilst deep learning approaches to a variety of bioinformatics problem domains, including omics, have consistently produced superior performance over popular statistical machine learning methods (e.g., logistic regression [105], support vector machines [41] and decision tree methods such as random forests [19] and XG Boost [25]), they lack inherent transparency and are considered to be a 'black box' approach by most clinicians and bioinformatics researchers. This is due to the number of nonlinear transformations the original input signal is exposed to as it is propagated through the hidden layers of a deep net [7]; and if deep nets are to gain broad clinical acceptance then some method to explain or interpret these latent representations is essential [81].

This section therefore provides a brief comprehensive overview of the notable techniques that researchers have applied in order to gain useful insight into what

the hierarchical abstract representations embedded within these deep nets are representing and the salient input features identified. we broadly categorise these techniques as follows: a) hidden layer visualisation and interpretation; b) Feature importance and impact evaluation; and finally, c) Output layer gradient analysis. It is left until the discussion in the next section to highlight the current and future significance of these approaches to omics research.

4.1. Hidden layer visualisation and interpretation

In the context of deep feature mining, we refer to hidden layer visualisation, as the process of evaluating the hidden layer weights and node activations of a deep net that has typically been trained to perform an image classification task (e.g., such as the CNN used by Krizhevsky et al [58] to classify 1.2 million high resolution images from the ImageNet ILSVRC benchmark dataset). At the simplest level of visualisation, the weights feeding in to each hidden node within a hidden layer[13] can be viewed as a specific transformation of the input eminating from the previous layer. As the raw input to the deep net is image data then these weights (and feature map activations in CNNs) can be rendered as an image themselves and therefore be visualised. If the original input image was that of a face, for example, then these particular weight transformations might represent important low- or high-level features of faces, where the hidden layers closer to the input layer are representing the lower level features (e.g., face contours, eye, lip, nostril etc) and those closer to the output layer representing more abstract features (e.g., outline of head-shape of specific spatial orientations, grainy integration of lower level features such as nose and mouth, opaque grey-scale clusters with highly activated regions representing position of notable features such as eye, ears, jaw and so forth). This type of weight-based visualisation has a long history and has been notably been demonstrated for face recognition (see Taigman et al [88] and the survey paper by Wang et al [99]) and autonomous vehicle control for both shallow (Pomerleau, D.A. [71]) and deep nets (Bojarski et al [16]). Zeiler et al [107] proposed a more sophisticated technique for visualising

[13] Which could be a standard MLP hidden layer or a convolutional or pooling layer in a CNN.

the representations formed in CNNs. Zeiler et al proposed a visualization technique that directly identifies aspects of the original input image that meaningfully activate individual CNN feature maps at any layer in the model. The authors build on their previous work on 'deconvolutional networks[14]' [108] to project the hidden layer responses back to the input pixel space. The authors train a CNN in the same way as proposed by Krizhevsky et al [58] and then attach a deconvnet to each of its layers in order to provide a continuous path back to the original image pixels. To start, an input image is presented to the CNN and features computed throughout the layers. The authors then examine specific feature map activations within the CNN by setting other activations in the feature map to zero and passing these as input to their attached deconvnet layer. Zeiler et al then reverse the CNN's convolutional and pooling operations by successively applying (i) unpooling, (ii) rectification and (iii) filtering to reconstruct the activity in the layer beneath that gave rise to the chosen activation. This process is repeated, traversing back through the CNN's hidden layers, until the original input pixel space is reached. Unlike the previous visualisation technique mentioned, this also enables Zeiler et al to better understand which aspects of the input image are most important to the classification task by ablating particular regions of the input image and evaluating the subsequent impact. The authors reported that the projections from each layer show the hierarchical nature of the latent features formed by the CNN (e.g., layers close to the input layer, such as Layer 2, represents lower level features such as corners and other edge/color conjunctions; whereas the higher layers closer to the output layer, such as layers 4 and 5, show class-specific variation (e.g., dog faces, birds legs) and entire objects with significant pose variation (e.g., dogs and keyboards). In a bid to support researchers with their efforts to visualise the latent representations of such CNNs used by Zeiler et al, Yoskini et al 2015 [106] developed two important open source image visualisation software tools: i) a tool to visualise the hidden layer activation profiles produced on CNNs as it processes imaging or live stream video

[14] A deconvolutional net or 'deconvnet' uses the same filtering and pooling layers of a standard CNN but in reverse, so instead of mapping pixels to feature maps, it maps features to input pixels.

data in real-time; and ii) a new regularisation method for improving the quality and interpretability of the features on the hidden layers. These tools were developed with accessibility in mind and thus have gained wide acceptance due to their broad applicability to many problem domains. More recently, Simonyan et al [85] notably presented two visualisation techniques for deep classification using CNNs based on computing the gradient of the class score with respect to the input image. The first generates an artificial image, which is representative of a class of interest and maximises the class score and representation of that class; whereas the second technique computes a class saliency map, specific to a given image and class. The benefit of the class saliency map is that it can be utilised for weakly supervised object segmentation for image classification using proposed Simonyan et al also demonstrated consistencies between their proposed gradient visualisation methods and the visualisation techniques for CNNs proposed by Zeiler et al [107].

We now consider hidden layer interpretation, which for our purposes we take to mean either i) the standardised distribution of the weights and the relationship of highly positive or negative weight values to the connected input and hidden units; or ii) the relationship of hidden unit responses to either the input tokens in a feed-forward MLP, where these responses might represent a higher order concept spanning multiple input tokens; or to an underlying sequence of inputs processed by an RNN where the pattern of hidden unit responses represent some traversal across basins of attractors within state space representing the underlying temporal structure of the data (e.g., a grammar describing the linguistic input). Tan et al [90] were arguably the first to successfully demonstrate the value of analysing the magnitude and direction of hidden weight and node activation values within an Auto-Encoder to extract useful information from omics data. More specifically, they trained a Denoising Auto-Encoder (DA) [97] with a single hidden layer on the METABRIC breast cancer dataset to construct and summarise useful features concerning breast cancer. A DA is simply a standard Auto-Encoder trained with inputs that have been corrupted with random noise and whose subsequent learned representations are far more resilient to input variations than standard Auto-Encoders [97]. To demonstrate

the robustness of their DA representations, Tan et al evaluated their trained model on an independent breast cancer data set, taken from the TCGA archive, and demonstrated that their DA constructed latent features that: i) discriminate between subjects with tumours and those without; ii) classify patient's estrogen receptor (ER) status; iii) summarise intrinsic subtypes; and iv) identifies activity of key transcription factors. Tan et al also observed that the extracted features were more predictive of patient survival than the commonly used markers (tumour grade and ER status). The approach adopted by Tan et al to reveal these insights was based on interpreting the weights to each hidden unit and also the node activation distribution in response to a series of input patterns. As shown in figure 7, the weights to each hidden unit followed a normal distribution and the hidden unit activations followed a bimodal distribution.

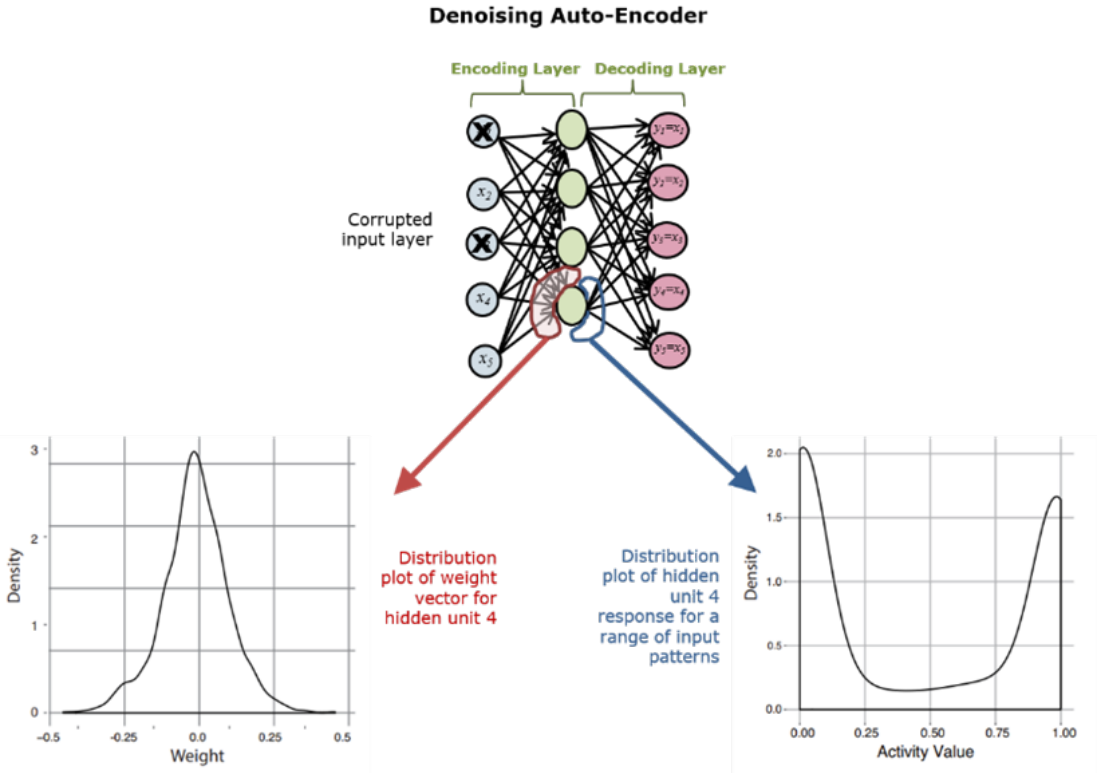


Figure 7: Hidden unit interpretation (adapted from Tan et al [90]).

The authors analysed the hidden unit activations to understand how they related to different sample characteristics (e.g., features that distinguish between those who have a tumour and those that do not). To achieve this, Tan et al divided their METABRIC sample into two parts: two thirds for training or 'discovery' and the remaining third for testing. Each hidden node's activation was binarised based on the minimum and maximum activation values for the discovery set (which was facilitated by the natural bimodal distribution of responses) and 10 activation intervals were defined within this range. Each of these node intervals were evaluated against the discovery set for predicting the desired sample characteristic. The nodes with the highest balanced classification accuracies were selected and their corresponding thresholds recorded. These thresholds were then used when the model was applied to the test set. To avoid sampling bias, Tan et al repeated this process ten times with random selections of the patterns for the training and test sets. To evaluate the crossover of this approach to their independent TCGA test set, Tan et al performed an independent evaluation by calculating the average of the thresholds determined from the ten METABRIC training and test experiments and applying these to the TCGA test set. Specific nodes were identified as being useful for predicting survival, that is separating good and poor prognosis. With respect to hidden weight analysis, Tan et al developed a weight interpretation approach to link transcription factors to the hidden units (constructed features). The approach is based on the inherent assumption that weight values connecting the input and hidden layers determine how each gene in the input layer influences the hidden unit activity values. The authors found that the distribution of weights for all genes to a single node approximately resembled a normal distribution centered at zero (as shown in figure 7 which was adapted from Tan et al). The key insight reported, which was further supported by subsequent findings by Danaee et al [29] and Alzubaidi et al [7], is that most input genes were assigned weight values of zero or values around zero for each hidden node; a small number of genes, however, gave highly positive or highly negative weight values (at least +/- 2 standard deviations from the mean) and it is therefore these genes that warrant further consideration. Tan et al found that highly weighted genes were over-represented by genes bound to

one transcription factor and this allowed them to perform further useful analysis proving the significance of these transcription factor genes. A notable limitation of the work presented by Tan et al is that their analysis of the hidden layer was restricted to a shallow model, i.e., a single layer of hidden units; and thus providing little insight into how one might perform such weight interpretation for deep nets, which have many hidden layers.

Interpreting the weights from stacked Auto-Encoders for omics research was notably demonstrated by Danaee et al [29], who presented a Stacked Denoising Auto-Encoder (SDAE) to derive hierarchical abstract features from a training set of RNA-seq expression data from TCGA database for both tumor and healthy breast samples. The data was heavily imbalanced (1097 breast cancer samples, and only 113 healthy samples) so synthetic minority over-sampling technique (SMOTE) [24] was used to create an evenly balanced dataset. Danaee et al proposed a SDAE with an input layer of 15,000 gene expressions and three hidden layers of dimension 10,000, 2,000, and 500 nodes respectively. To analyse the importance of the genes by interpreting the hidden layer weights, the authors computed the product of the weight matrices for each layer of the SDAE, resulting in a 500 × G dimensional matrix W, where G is the number of genes in the expression data, computed for an n-layer SDAE by

$$\mathbf{W} = \prod_{i=1}^{n} \mathbf{W}_i \,. \tag{1}$$

As with Tan et al, Danaee et al found that the terms of matrix W were strongly normally distributed and were able to identify a subset of 500 genes (same dimension as SDAE last feature layer), called DIFFEXP500, by calculating p-values from a two-tailed statistical test to establish which genes associated with the connections through the network that were either highly negative or highly positive. The matrix W is considered to be a linearisation of the compounded effect of each gene on the SDAE hidden units and therefore genes with the largest weights in W are considered to be the most strongly connected to the extracted and highly predictive features. Danaee et al referred to these genes as Deeply Connected Genes (DCGs). The authors identified 244 up-regulated and 256 down-

regulated genes using this process. It is important to state that Danae et al evaluated a number of classifiers to determine whether or not a breast tissue sample has cancer, with some using just the most useful abstract feature layer from SDAE as input and others using the 500 extracted genes just mentioned. The SVM-based classifier trained with SDAE's actual abstract representations achieved an F-measure score of 98.3% for the test set compared to a score of 75.5% obtained by the best SVM-based classifier trained on the extracted gene expression data; demonstrating the quality of the hidden unit layer representations formed by the deep net (which are possibly utilising more encoded gene expression information than those extracted through the weight interpretation process). An issue with Danaee et al's approach is that it is not clear how they defined the DCGs especially when each gene has 500 values and there is no evidence whether they have considered the the negative direction. Alzubaidi et al [7] subsequently introduced a weight interpretation method better suited for extracting features from deep Auto-Encoders. Alzubaidi et al [7] proposed a revised Auto-Encoder deep net framework, referred to as the Sparse Compressed Auto-Encoder (SCAE), to learn omics problems that are characterised by high dimensionality and small sample size (HDSSS). SCAE is similar to a standard Auto-Encoder, however, it is deliberately constructed in an under-complete manner, where a sparsity penalty is added to the cost function based on the values of the hidden units, in a similar way L1 and L2 regularisation adds a penalty term to the cost function based on either the scaled total absolute sum of the weights or the scaled squared sum of the weights respectively [38]. The intention is to avoid overfitting by restricting the magnitude of the hidden unit activations to form sparse representations where activations are close to zero and thus unable to be highly responsive to a specific inputs.

To evaluate SCAE and the weight interpretation method, Alzubaidi et al used ovarian cancer data from the FDA-NCI Clinical Proteomics Program Databank to identify serum (blood-derived) proteomic patterns that differentiate the serum of patients with ovarian cancer from that of women without ovarian cancer (i.e., 216 samples and 15000 features). The METABRIC Breast Cancer dataset was also used for classification relating to the status of Estrogen Receptor (ER) and

Progesterone Receptor (PR); that is if breast cancer cells have high ER, the cancer is described as ER-positive (ER+), and if breast cancer cells have high PR, the disease is specified as PR-positive (PR+) cancer. ER and PR expressions are accepted as robust indicators for the evaluation of breast cancer. The mRNA expression dataset used contained 24368 genes and 1904 samples. The proposed deep mining model was used to open the black box of such deep learning models to find which genes were dominant within its internal representations. The deep mining model relies on leveraging the Input Weight matrix (IW) of the Stacked SCAE (SSCAE) with its Layers Weight (LWs) matrices, which results in a d × 1 weight vector called DM, where d corresponds to the number of features in the original datasets so that each feature has a weight score that reflects its contribution, as follows:

$$\mathbf{DM} = \mathbf{IW}^{\top} \prod_{i=1}^{L} \mathbf{LW}_i^{\top}. \tag{2}$$

The weight vector DM resembles a normal distribution as shown in Figure 8 of ovarian cancer dataset at fold 1 of the cross validation procedure. A small per-

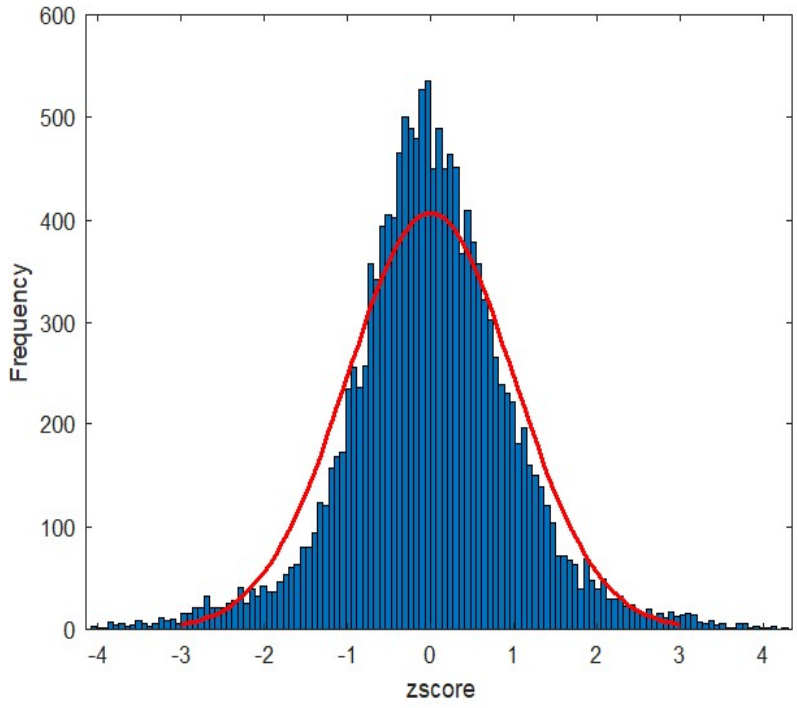


Figure 8: Histogram of z-scores of the weight vector.

centage of features in the DM exhibit High Positive (HP) or High Negative (HN) weight as shown in Figure 8. The findings of this research have demonstrated the capability of the proposed deep mining model to recognise the biomarkers that exhibit High Positive (HP) and High Negative (HN) weight scores over the depth of the network. HP weighted biomarkers are the molecules that have a strong positive correlation with the positive group, where HN weighted biomarkers are the molecules that have an inverse association with the positive group. This explains the internal mechanism of the deep mining model in assigning HP weight to the features that are highly expressed for the positives in comparison to most of the negatives. In contrast, HN weights were allocated by the model to the features that are lowly expressed for most of the positives in comparison to the negatives. The validation process reveal that the discovered biomarkers demonstrate computational and biological relevance as well as the capability to construct highly accurate and reliable prediction models. This provides

significant evidence that the deep mining model was very effective in offering explainability to the deep learning model and detecting key determinants underlying its latent representations.

To conclude this section on hidden layer visualisation and interpretation, we now turn our attention briefly to recurrent neural networks. RNNs have the added complication of an internal memory mechanism, typically situated on the input layer, although interim hidden layers in deep RNNs might also contain internal memory layers accepting past activations from the layer in front. As discussed earlier, the effect is that each hidden layer response to the current external input is also dependent on the previous activation history of the network in order to generate a new set of state values and an output response back to the environment. When discussing interpretation we therefore need to consider sequences of related input stimuli (e.g., representing words in a sentence or nucleotides in a DNA sequence) and the corresponding hidden unit activations generated in response to these input sequences. This sequence of hidden unit activations could be considered a trace across state space where positions within the state space represent some underlying concept. In essence, by understanding how these hidden units respond to each input symbol over time, it is possible to determine whether or not these networks have formed an adequate representation of the underlying data generation process (so for a linguistic problem this might typically refer to a grammar; and for a DNA sequencing problem this might refer to the underlying information a cell needs to assemble protein and RNA molecules). For a comprehensive overview of knowledge extraction methods for RNNs and associated challenges, we refer the reader to the following notable approaches and discussion articles [22,36,48,51,57,104]. To give the reader an appreciation of how knowledge can be extracted from an RNN, we briefly discuss the approach adopted by [91]. Assuming the input stimuli is a sequence of related symbolic tokens representing say words or some other unit of meaning, Principal Component Analysis (PCA)[15] [103] can be applied to

[15] PCA is commonly used for dimensionality reduction by establishing a new set of variables, from the original set, that better captures the variation in the sample and can be viewed as rotations of the original variable set. There are fewer new variables than the original and these

the hidden state activations of an RNN trained on some symbolic induction or transducer task. The internal representation formed by the weights of an RNN after training can be revealed by analysing the resulting hidden unit activations and how they vary with respect to each new input within a sequence. This can be determined by calculating the principal components of the covariance matrix of hidden unit activations and their respective mean values. The average activation for each hidden unit is based on the unit's response to all symbols in the training set. The resulting principal components are ordered according to the magnitude of the corresponding eigenvalues, where the highest eigenvalue represents the most feature information captured within the internal (hidden) state and the lowest represents the least. Two Principal Components (PCs) can then be used to generate a two-dimensional view of the state-space of the networks. During the training process, it is expected that within its internal state, a trained RNN will form clusters of activations in state-space that represent some concept relevant to the mapping task (e.g., states of an underlying grammar for linguistic input). It is further expected that where training has been successful, these state-based activations will themselves be arranged in patterns that distinguish between regions of the solution space (e.g., different sections of the underlying grammar generating the input sequence). The key aim of using PCA in this way is to allow two-dimensional views of the RNN's state-space to be generated, thereby facilitating visualisation of the above clusters and patterns and providing opportunity for further insight into the networks' inner workings and testing of our understanding of their operation. Tepper et al cogently demonstrated the usefulness of this technique for extracting and evaluating approximations of finite state machines formed by a range of different RNNs for a complex grammar induction task. Whilst further research is required to understand how to extract useful knowledge from deep RNNs, these rule extraction methods do show

new variables are referred to as principal components and are uncorrelated and ordered by the fraction of the total information retained. A limitation is that there is an underlying linearity assumption of the relationship between the data and set of variables.

promise and indeed the potential for examining the internal dynamics of the RNNs used by omics researchers such as [26,27,54].

4.2. Feature importance and impact evaluation

Due to the original inputs to a deep net being subject to successive layers of nonlinear transformations, it is extremely difficult to ascertain precisely which input features have the most impact on the model predictions. The process of trying to establish the subset of features of the highest importance to the classification or prediction task is referred to as 'feature selection'. Simple linear models, such as linear regression or logistic regression, could be considered a simple one-layer perceptron with input features connected to a linear unit (or sigmoid function for logistic regression) via coefficients indicating the importance of the feature in the final calculation for the prediction. For example, with logistic regression[16] the coefficients associated with an input feature are said to represent a measure of importance of that feature in the form of a log odds ratio, i.e., $\log(p/q) = \beta_0 + \beta 1 x_1$ and where p represents the probability of success, and q of failure, i.e., $q = 1 - p$)[17]. So, if x_1 represented the feature of gefitinib dosage for a logistic regression model to classify patients as responders or non-responders to various lung cancer therapies; and the associated coefficient β_1 is assigned a value of 1.3392; then this indicates a unit change in x_1 results in a subsequent 1.3392 unit change in the log of the odds associated with the probability. However, as with linear regression, logistic regression holds an inherent assumption of linearity amongst its predictor and response variables, therefore affecting the reliability of the coefficients and limiting it to relatively simple problem domains. Decision trees abandon any such linearity assumption, and methods such as C4.5 [73] and Random Forests [18], use the inverse binary log rule, first introduced by Claude Shannon in his seminal work on Information

[16] A logistic regression model establishes a relationship between a binary outcome or 'response' variable and a group of feature or 'predictor' variables. It models the logit-transformed probability as a linear relationship with the predictor variables – for more information please see [63,77].

[17] Actual probabilities can be recovered by simply exponentiating the log odds ratios.

Theory to estimate the amount of disorder or entropy within a communication channel [80] to determine the 'information gain' or impurity of a feature variable based on whether its values are able separate the data into the different pattern classes. Random forests typically apply the Gini importance measure as follows (for a simple binary classification task):

$$G = p_1(1 - p_1) + p_2(1 - p_2). \quad (3)$$

where $p1$ and $p2$ refer to the probabilities of class 1 and 2 respectively being the dominant class for that feature. As the probabilities approach zero the Gini index, representing node impurity (degree to which feature is unable to reliably split the data into the two categories), is said to be minimized. The total decrease in Gini index is calculated after each node split and subsequently averaged over all trees where the lower the impurity, the higher the importance of the feature [103], Saarela et al [77]]. The importance scores can provide the researcher with a stronger indicator as to the important feature subset which could be used as input to other prediction models such as a deep net.

Two common approaches to force the coefficients to be less sensitive to the individual input features by restricting their values are the $L1$ and $L2$ regularisation methods. $L1$ regularisation, also referred to as Least Absolute Shrinkage and Selection Operator (LASSO) regularisation [38], adds a scaled penalty term to the cost function based on the absolute total sum of the coefficients, thus as the magnitude of coefficients rise, so does the error and likely correction to reduce the value of the coefficients. This has the distinct advantage of driving the weights or coefficients of less important features closer to 0 and thus performing a type of feature selection. L2 regularisation, also referred to as ridge or Tikhonov regularisation [38], on the other hand, adds a scaled penalty term to the cost function based on the total sum of the squared coefficients. The effect of this penalty term is commonly referred to as 'weight decay' as it drives the coefficients closer to the origin by multiplicatively shrinking the weight vector by a constant factor on each gradient calculation step [38]. Whilst $L2$ regularisation restricts the growth of the coefficients and helps to minimise over-

fitting, it does not perform the desired feature selection as $L1$ does. $L1$ and $L2$ regularisation methods can be used for key machine learning methods such as linear regression, logistic regression and deep nets and are considered a global approach to determining feature importance due to the impact on all input patterns caused by the coefficients being directly affected. $L1$ and $L2$ penalties of the lasso and ridge methods can be integrated to further restrict the weights and coefficients (known as the 'elastic net' [112]. In terms of computational complexity, these methods are between so-called wrapper and filter [18] approaches to feature selection.

Rather than directly restricting the weights or coefficients of a deep net using the above regularisation techniques, Shao et al (2021) [81] developed a novel impact scoring technique that evaluates the impact of input features representing clinical observations on the outcome (probability of death within one year of a major cardiovascular procedure (MCVP)). More specifically, they examined various features of a cohort of 21,355 veterans, namely demographic data (e.g., age, gender, race) and medical history (diagnosis and procedure codes for cardiovascular disease; medication; all-cause hospitalisations and clinical notes over the last two years). The authors split their feature variables into two categories: temporal features (discretised patient history data spanning the last two years) and static variables, which included the demographic information. In order for convolutional neural networks to be used, the input data was converted into a two-dimensional plane (akin to image data format). The structure of the

[18] Wrapper-based approaches train a machine learning model (e.g., Support Vector Machine, Logistic Regression or Decision Tree) on a range of independent subsets of features and compare their predictive accuracy on a hold-out test set. The model's error rate associated with the test set is the score given for the subset of features used. Given the iterative training procedure, the approach is computationally very intensive. Filter methods on the other hand, use common statistical algorithms such as Pearson product-moment correlation coefficient and mutual information, to score each feature/class combination. Filter methods are much faster than wrapper-based approaches and can be used as a pre-cursor to the wrapper method to filter out irrelevant features before the iterative training process. A common such approach is Recursive Feature Elimination [39] where feature subsets are trained with Support Vector Machines and features with low weights are removed and the process repeated. See [103] for more information.

CNN was such that the temporal data was first passed through a number of convolutional and pooling layers; and the demographic data was then provided as input to the fully connected hidden layer (located deeper into the architecture) which served to integrate the demographic data with the flattened pooling layer emerging from the earlier convolutional process with the temporal data. After successfully training the CNN based on a standard early stopping using the validation set and selecting the optimum model using the area under the receiver operating characteristic curve (AUC) , Shao et al evaluated the value of their input features within their deep net by setting non-zero 'pixel' values to zero (essentially ablating the input) and evaluated the impact on the prediction accuracy using the logit function. Furthermore, they evaluated the impact of these input features at the population level by simply averaging the logit-based impact scores for each individual ablated temporal and non-temporal feature across all patients in the training set who had an observation or impact score available. The authors compared their approach with a standard logistic regression model and associated log odds ratio for determining feature importance. The signs of the resulting impact scores (for deep net) and log odds ratio (for logistic regression) proved pivotal for the comparative analysis of feature importance given by the different approaches. The AUC reported for the deep net was 0.787 and 0.746 for the logistic regression model. The resulting top ten features for both models were examined and compared using Pearson's correlation, Spearman's rank correlation and sign agreement between the deep net impact scores and logistic regression log odds ratio. Shao et al reported significant correlations between the variable selections for the respective top 18 features suggesting that the nonlinear deep net and linear logistic regression model identified similar features. Unfortunately, there are a number of notable limitations of Shao et al's study (some of which they recognised), namely, the small sample size; the lack of insight with respect to the integration of features relevant to the outcome and finally, the role of nonlinearity in the deep net selections, i.e., how important were those features the CNN selected as significant where the logistic regression model did not?

4.3. Output layer gradient analysis

As mentioned earlier when discussing visualisation methods to identify important input features, techniques were notably developed where either the network operation was reversed to project back onto important aspects of the original input image (which may have ablated inputs) [107] or gradients of the output layer were computed for the class score with respect to the input image, i.e., we have a product of the input layer and output gradient [85]. Sundararajan et al [86] extend the gradient based approach, where the machine learning process itself is adapted to better associate input features with output responses to enable some rationale to be identified for a deep net's predictions, e.g., which part of an input image should a doctor focus on when understanding a deep nets recommendation? The general principle underpinning the gradient approach is that the higher the gradient for a feature variable, the greater the importance on the output response for that specific feature value of interest. The gradient for a specific feature varies according to the feature's value and interacts with other features to effect an output response, thus generating a local gradient for the specific feature of interest. For example, to understand the impact of a feature, its value is typically set to zero (the reference value) and the impact on the output and associated gradient evaluated. Sundararajan et al specifically devise an approach to ensure that the sensitivity and implementation invariant requirements are not broken, i.e., in cases where a network response to the change of a specific feature value is different, a non-zero attribution value (rather than the 0 reference value) should be applied to the input feature when the gradient for the specific data point is zero. Sundararajan et a refer to this new attribution approach as 'Integrated Gradients' as the output gradients are integrated as one traverses the path from the reference input value to a particular output response. The authors cogently argue their position by exemplifying the implementation invariance issue with the DeepLift model [83], a gradient method which links the output prediction of a deep net to a specific input feature by backpropagating the gradients (contributions of all neurons in the network to every feature of the input) and then comparing the activation of each neuron to

its ‘reference activation’ and assigns contribution scores according to the difference. As DeepLift replaces continuous gradients with a discretised version and applies a modified form of back-propagation this causes incompatibilities with the chain-rule used by back-propagation resulting in implementation invariance, i.e., the network becomes sensitive to poor predictors. As with Zeiler et al and Simonyan et al, the focus of this research is centred on associating local regions of an input image with particular responses and its usefulness for omics-related problems has yet to be determined; however, it is a proven approach to associating output responses with input features and worthy of exploration by omics researchers who seek to open the block box of their deep nets.

5. Discussion

Since the advent of high throughput omics technologies, various molecular data such as genes, transcripts, proteins, and metabolites have been made widely available to researchers. In particular, the availability of omics data repositories such as The Cancer Genome Atlas (TCGA) [100], containing diverse types of data like somatic mutation, copy number, gene expression, mRNA expression, and DNA methylation has revolutionaised omics research by providing high quality multimodal data. This has afforded clinicians, bioinformaticians, statisticians and data scientists the opportunity to apply their innovations in feature mining and predictive modelling to a rich data resource to develop a wide range of transparent and generalisable prediction models.

What has become apparent from research over the last 10 years, is that bioinformatics researchers, including those in cognate disciplines such as omics, have adopted deep neural networks as their preferred paradigm of choice for complex data modelling. This is not surprising, as since the inherent limitations of training artificial neural networks with many hidden layers using back-propagation was largely overcome in 2006, performance of deep learning models has outstripped that of their ’shallow feature learning’ counter parts such as support vector machines, logistic regression and decision trees, providing cutting-edge performance and insights into a variety of omics-related problem

domains (e.g. [7,53,66,107]. When utilising deep nets, we highlighted the need for omics researchers to make key design and implementation decisions such as the type of architecture (e.g., feed-forward, recurrent, convolutional or pooling hidden layers); training approach (e.g., incremental layer-wise training of Auto-Encoders or use one large network with non-saturating activation functions, such as ReLU or similar); software framework (e.g., TensorFlow or Pytorch); and implementation platform (e.g., GPUs for fast parallel computations). The reasons for their success are generally attributed to their ability to automatically transform the original omics input data into hierarchical abstract representations formed through multiple layers of nonlinear transformations, much better-suited to the omics task at hand. A key stumbling block, however, is that deep nets lack inherent transparency and are considered to be a 'black box' approach. In addition, there appears to be little consensus in the literature as to how to interpret these complex internal representations underlying the mapping between the input and output layers and therefore explain why these models predict/behave as they do. This naturally makes it very difficult for clinicians and other stakeholders to trust their deep learning models even though the model predictions appear to be highly accurate.

We subsequently presented a number of notable deep feature mining techniques researchers from various disciplines have developed to open up the 'black box' of deep nets in a bid to provide readers with some insight into the current direction of travel. We grouped these techniques into the following three categories: a) hidden layer visualisation and interpretation; b) input feature importance and impact evaluation; and c) output layer gradient analysis.

In terms of hidden layer visualisation, where hidden weights or node activations can be rendered as images providing some visual insight into what higher level features have been formed from the original input image, this approach has largely been restricted to image recognition problems (such as face recognition or autonomous vehicle control) using either MLP networks or convolutional networks. Likewise, approaches such as that by [107,108] that trace back from features across multiple hidden layers to the original input image have also shown promise in revealing what these hidden layers are encoding.

However, the direct relevance to omics problems remain unclear at this time. As discussed previously, CNNs have been successfully applied to omics problems as cogently demonstrated by [62,78,81], where an initial pre-processing stage is required to map the $1d$ omics input data to a $2d$ input plane so that it can be presented as input to a CNN. It might be that such visualisation could be applied to the CNNs to reveal useful bespoke visual patterns relating to how the input has been hierarchically organised for the tasks at hand? At this point, it is worth briefly mentioning that it is our view that approaches for visualising or interpreting sequences of hidden states within recurrent neural networks in a stable and reliable manner remain firmly within its infancy. Whilst Tepper et al [91] and others [22,36,48,51,57,104] have demonstrated how various statistical techniques, such as PCA, can be used to reduce the dimensionality of state space so that it can be meaningfully visualised, further research is certainly required to understand how we can automatically identify meaningful collections of stable states from deep recurrent networks, such as those used by omics researchers [52,75].

Omics researchers have reported notable success with opening the black box of the hidden layer(s) using the weight interpretation approach, where the weights between layers are typically standardised and those weights below a particular p-value are subject to further investigation. Tan et al [90], found that the distribution of weights for all of their input genes to a single hidden node within a shallow network approximately resembled a normal distribution centered at zero and this was supported by findings by Danaee et al [29]. Alzubaidi et al [7] found that two subsets of genes are assigned highly positive and highly negative weight values (at least +/- 2 standard deviations from the mean), and these generic genes exhibit a positive and negative association with cancers of interest, in which highly positive (HP) weights were assigned by the deep mining model to the genes that are highly expressed for the positives in comparison to most of the negatives. In contrast, highly negative (HN) weights were allocated by the model to the genes that are lowly expressed for most of the positives in comparison to the negatives. These generic subsets of molecular indicators were utilised for developing generalisable risk prediction models

across multiple independent breast invasive carcinoma datasets. The work so far has demonstrated that the genes selected in this manner offer superior performance in data-driven molecular biology than other feature mining methods (such as genetic algorithm-based approaches) [8].

When determining attribute importance within deep neural networks, it is likely that $L1$ and $L2$ regularisation techniques with its associated hyperparameters will remain a ubiquitous restriction applied to the learning process to minimise the possibility of over-fitting and maximise the likelihood of only the most salient features having the higher weightings [38]. It is anticipated that in addition to continuing widespread use of the standard wrapper and filter methods alluded to earlier, further bespoke attribute impact measures, such as the promising approach proposed by Shao et al [81], will emerge to enable researchers to better understand the global impact of individual input attributes on the response of a deep neural network.

We then briefly presented the output gradient analysis approach for interpreting a deep net's black box. As discussed, the basic premise of the gradient approach is that the higher the computed gradient for a feature variable (by backpropagation), the greater the importance on the output response for that specific feature value of interest. The array of different attribute values are evaluated by simply comparing the model's performance against the situation where that feature variable is fixed to a known reference value, such as 0. Whilst these approaches have shown some traction in image recognition tasks, the applicability to omics research remains unclear. That said, it does appear that the general premise and technique could easily be transferred to non-image problems and thereby help omics researchers to associate output responses with input features and so in our view, worthy of further consideration by omics researchers.

In this current era of high quality omics data and the success of the deep neural network paradigm in this domain, we are clearly in an exciting period of innovation in the field of deep mining from omics data. It is also very clear that there is a concomitant need to not only embrace existing approaches but to further tailor and exploit the deep feature mining methods discussed in this

Chapter to imbue bioinformaticians, clinicians and researchers, from cognate disciplines, with the confidence needed to trust their accurate deep learning models.

References

[1] Mart´ın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016. 16

[2] Cory Abate-Shen and Michael M Shen. The prostate-cancer metabolome. *Nature*, **457**[7231]:799–800, 2009. 3

[3] Babak Alipanahi, Andrew Delong, Matthew T Weirauch, and Brendan J Frey. Predicting the sequence specificities of dna-and rnabinding proteins by deep learning. *Nature biotechnology*, **33**[8]:831, 2015. 16

[4] Alexander Aliper, Sergey Plis, Artem Artemov, Alvaro Ulloa, Polina Mamoshina, and Alex Zhavoronkov. Deep learning applications for predicting pharmacological properties of drugs and drug repurposing using transcriptomic data. *Molecular pharmaceutics*, **13**[7]:2524– 2530, 2016. 2, 6

[5] Mohammed AlQuraishi. Alphafold at casp13. *Bioinformatics*, **35**[22]:4862–4865, 2019. 6

[6] Abeer Alzubaidi. Challenges in developing prediction models for multimodal high-throughput biomedical data. In *Proceedings of SAI Intelligent*
Systems Conference, pages 1056–1069. Springer, 2018. 4

[7] Abeer Alzubaidi, Jonathan Tepper, et al. A novel deep mining model for effective knowledge discovery from omics data. *Artificial Intelligence in Medicine*, page 101821, 2020. 6, 7, 21, 26, 27, 37, 39

[8] Abeer HA Alzubaidi. *Evolutionary and deep mining models for effective biomarker discovery*. PhD thesis, Nottingham Trent University, 2019. 39

[9] Christof Angermueller, Heather Lee, Wolf Reik, and Oliver Stegle. Accurate prediction of single-cell dna methylation states using deep learning. *BioRxiv*, page 055715, 2017. 6

[10] Francisco Azuaje. *Bioinformatics and biomarker discovery*. Wiley Online Library, 2010. 4

[11] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, **35**[8]:1798–1828, 2013. 2

[12] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. pages 153– 160, 2007. 11, 13, 14

[13] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, **5**[2]:157–166, 1994. 11

[14] Ivan Berest, Christian Arnold, Armando Reyes-Palomares, Giovanni Palla, Kasper Dindler Rasmussen, Holly Giles, Peter-Martin Bruch, Wolfgang Huber, Sascha Dietrich, Kristian Helin, et al. Quantification of differential transcription factor activity and multiomics-based classification into activators and repressors: difftf. *Cell reports*, **29**[10]:3147–3159, 2019. 6

[15] Jane M Binner, Peter Tino, Jonathan Tepper, Richard Anderson, Barry Jones, and Graham Kendall. Does money matter in inflation forecasting? *Physica A: Statistical Mechanics and its Applications*, **389**[21]:4793–4808, 2010. 20

[16] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016. 22

[17] Leon Bottou and Patrick Gallinari´ . A framework for the cooperation of learning algorithms. In *Advances in neural information processing systems*, pages 781–788, 1991. 10

[18] Leo Breiman. Bagging predictors. *Machine learning*, **24**[2]:123–140, 1996. 32

[19] Leo Breiman. Random forests. *Machine learning*, **45**[1]:5–32, 2001. 21

[20] Qing Cao, Bradley T Ewing, and Mark A Thompson. Forecasting wind speed with recurrent neural networks. *European Journal of Operational Research*, **221**[1]:148–154, 2012. 20

[21] Miguel A Carreira-Perpinan and Geoffrey E Hinton. On contrastive divergence learning. In *Aistats*, **10**, pages 33–40. Citeseer, 2005. 13

[22] Bo Cartling. On the implicit acquisition of a context-free grammar by a simple recurrent neural network. *Neurocomputing*, **71**[7-9]:1527–1537, 2008. 30, 38

[23] Ethan Cerami, Jianjiong Gao, Ugur Dogrusoz, Benjamin E Gross, Selcuk Onur Sumer, Bulent Arman Aksoy, Anders Ja-¨ cobsen, Caitlin J Byrne, Michael L Heuer, Erik Larsson, et al.
The cbio cancer genomics portal: an open platform for exploring multidimensional cancer genomics data, 2012. 5

[24] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and
W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, **16**:321–357, 2002. 26

[25] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016. 21

[26] Kyunghyun Cho, Bart Van Merrienboer, Dzmitry Bahdanau,¨ and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014. 21, 31

[27] Neo Christopher Chung, Bilal Mirza, Howard Choi, Jie Wang, Ding Wang, Peipei Ping, and Wei Wang. Unsupervised classification of multi-omics data during cardiac remodeling using deep learning. *Methods*, **166**:66–73, 2019. 21, 31

[28] Dan Claudiu Cire¸san, Ueli Meier, Luca Maria Gambardella, and Jurgen Schmidhuber¨ . Deep, big, simple neural nets for handwritten digit recognition. *Neural computation*, **22**[12]:3207–3220, 2010. 12, 14, 15

[29] Padideh Danaee, Reza Ghaeini, and David A Hendrix. A deep learning approach for cancer detection and relevant gene identification. In *PACIFIC SYMPOSIUM ON BIOCOMPUTING 2017*, pages 219–229. World Scientific, 2017. 7, 26, 39

[30] Georg Dorffner. Neural networks for time series processing. In *Neural network world*. Citeseer, 1996. 20

[31] Darius M Dziuda. *Data mining for genomics and proteomics: analysis of gene and protein expression data*, **1**. John Wiley & Sons, 2010. 3

[32] Dumitru Erhan, Yoshua Bengio, Aaron Courville, PierreAntoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, **11**[Feb]:625–660, 2010. 14

[33] Jianjiong Gao, Bulent Arman Aksoy, Ugur Dogrusoz, Gideon¨ Dresdner, Benjamin Gross, S Onur Sumer, Yichao Sun, Anders Jacobsen, Rileen Sinha, Erik Larsson, et al. Integrative analysis of complex cancer genomics and clinical profiles using the cbioportal. *Sci. Signal.*, **6**[269]:pl1–pl1, 2013. 5

[34] Felix A Gers and E Schmidhuber. Lstm recurrent networks learn simple context-free and context-sensitive languages. *IEEE Transactions on Neural Networks*, **12**[6]:1333–1340, 2001. 20

[35] Felix A Gers, Jurgen Schmidhuber, and Fred Cummins¨ . Learning to forget: Continual prediction with lstm. *Neural computation*, **12**[10]:2451–2471, 2000. 19, 20

[36] C Lee Giles, Steve Lawrence, and Ah Chung Tsoi. Noisy time series prediction using recurrent neural networks and grammatical inference. *Machine learning*, **44**[1]:161–183, 2001. 30, 38

[37] Vanessa Gomez-Verdejo, Emilio Parrado-Hern´ andez, and Jussi´ Tohka. Sign-consistency based variable importance for machine learning in brain imaging. *Neuroinformatics*, **17**[4]:593–609, 2019. 7

[38] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016. 2, 6, 28, 33, 39

[39] Isabelle Guyon and Andre Elisseeff´ . An introduction to variable and feature selection. *Journal of machine learning research*, **3**[Mar]:1157– 1182, 2003. 34

[40] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015. 15

[41] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, **13**[4]:18–28, 1998. 21

[42] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, **313**[5786]:504–507, 2006. 11, 13

[43] Geoffrey E Hinton, Terrence J Sejnowski, et al. Learning and relearning in boltzmann machines. *Parallel distributed processing: Explorations in the microstructure of cognition*, **1**[282-317]:2, 1986. 13

[44] Geoffrey E Hinton and Richard S Zemel. Autoencoders, minimum description length, and helmholtz free energy. *Advances in neural information processing systems*, **6**:3–10, 1994. 10

[45] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jurgen¨ Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001. 11

[46] Sepp Hochreiter and Jurgen Schmidhuber¨ . Long short-term memory. *Neural computation*, **9**[8]:1735–1780, 1997. 19, 20

[47] Richard P Horgan and Louise C Kenny. 'omic'technologies: genomics, transcriptomics, proteomics and metabolomics. *The Obstetrician & Gynaecologist*, **13**[3]:189–195, 2011. 3

[48] Bill G Horne and Don R Hush. Bounds on the complexity of recurrent neural network implementations of finite state machines. *Neural networks*, **9**[2]:243–252, 1996. 30, 38

[49] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, **2**[5]:359–366, 1989. 10

[50] David H Hubel and Torsten N Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, **160**[1]:106–154, 1962. 17

[51] Henrik Jacobsson and Tom Ziemke. Cryssmex, a novel rule extractor for recurrent neural networks: Overview and case study. In *International Conference on Artificial Neural Networks*, pages 503–508. Springer, 2005. 30, 38

[52] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on*

Multimedia, pages 675–678, 2014. 16, 38

[53] Andrew R Joyce and Bernhard Ø Palsson. The model organism as a system: integrating 'omics' data sets. *Nat Rev Mol Cell Biol*, **7**:198–210, 2006 Mar 2006. 4, 37

[54] Mostafa Karimi, Di Wu, Zhangyang Wang, and Yang Shen.

Deepaffinity: interpretable deep learning of compound–protein affinity through unified recurrent and convolutional neural networks. *Bioinformatics*, **35**[18]:3329–3338, 2019. 21, 31

[55] David R Kelley, Jasper Snoek, and John L Rinn. Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks. *Genome research*, 2016. 6

[56] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*, pages 2668–2677. PMLR, 2018. 7

[57] John F Kolen. Fool's gold: Extracting finite state machines from recurrent network dynamics. *Advances in neural information processing systems*, pages 501–501, 1994. 30, 38

[58] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 14, 15, 22

[59] Jack Lanchantin, Ritambhara Singh, Zeming Lin, and Yanjun

Qi. Deep motif: Visualizing genomic sequence classifications. *arXiv preprint arXiv:1605.01133*, 2016. 16

[60] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, **521**[7553]:436, 2015. 2

[61] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick´ Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, **86**[11]:2278–2324, 1998. 12, 14, 17

[62] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. *Predicting structured data*, **1**[0], 2006. 14, 38

[63] Stanley Lemeshow and Melvin L Moeschberger. Review of regression methods in biostatistics: linear, logistic, survival, and repeated measures models by vittinghoff, glidden, shiboski, and mcculloch. *The Stata Journal*, **5**[2]:274–278, 2005. 32

[64] Mufti Mahmud, M Shamim Kaiser, T Martin McGinnity, and Amir Hussain. Deep learning in mining biological data. *Cognitive Computation*, **13**[1]:1–33, 2021. 14, 16

[65] Xu Min, Ning Chen, Ting Chen, and Rui Jiang. Deepenhancer: Predicting enhancers by convolutional neural networks. In *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 637–644. IEEE, 2016. 16, 19

[66] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010. 8, 15, 37

[67] Oluwatamilore Orojo, Jonathan Tepper, TM McGinnity, and Mufti Mahmud. Time sensitivity and self-organisation in multi-recurrent neural networks. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2020. 9, 20

[68] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019. 16

[69] Natasha L Patel-Murray, Miriam Adam, Nhan Huynh, Brook T Wassie, Pamela Milani, and Ernest Fraenkel. A multi-omics interpretable machine learning model reveals modes of action of small molecules. *Scientific reports*, **10**[1]:1–14, 2020. 6

[70] Barak A Pearlmutter. Gradient calculations for dynamic recurrent neural networks: A survey. *IEEE Transactions on Neural networks*, **6**[5]:1212–1228, 1995. 10

[71] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA ARTIFICIAL INTELLIGENCE AND PSYCHOLOGY ..., 1989. 22

[72] Ryan Poplin, Pi-Chuan Chang, David Alexander, Scott Schwartz, Thomas Colthurst, Alexander Ku, Dan Newburger, Jojo Dijamco, Nam Nguyen, Pegah T Afshar, et al. A universal snp and small-indel variant caller using deep neural networks. *Nature biotechnology*, **36**[10]:983–987, 2018. 16

[73] J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014. 32

[74] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017. 15

[75] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985. 6, 38

[76] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, **323**[6088]:533–536, 1986. 9

[77] Mirka Saarela and Susanne Jauhiainen. Comparison of feature importance measures as explanations for classification models. *SN Applied Sciences*, **3**[2]:1–12, 2021. 32, 33

[78] Sandhya Samarasinghe. *Neural Networks for Applied Sciences and Engineering: From Fundamentals to Complex Pattern Recognition*. Auerbach
Publications, 2006. 8, 38

[79] Arshdeep Sekhon, Ritambhara Singh, and Yanjun Qi. Deepdiff:
Deep-learning for predicting differential gene expression from histone modifications. *Bioinformatics*, **34**[17]:i891–i900, 2018. 21

[80] Claude E Shannon. A mathematical theory of communication. *The Bell system technical journal*, **27**[3]:379–423, 1948. 32

[81] Yijun Shao, Yan Cheng, Rashmee U Shah, Charlene R Weir, Bruce E Bray, and Qing Zeng-Treitler. Shedding light on the black box: Explaining deep neural network prediction of clinical outcomes. *Journal of Medical Systems*, **45**[1]:1–9, 2021. 21, 34, 38, 39

[82] Alok Sharma, Edwin Vans, Daichi Shigemizu, Keith A Boroevich, and Tatsuhiko Tsunoda. Deepinsight: A methodology to transform a non-image data to an image for convolution neural network architecture. *Scientific reports*, **9**[1]:1–7, 2019. 19

[83] Avanti Shrikumar, Peyton Greenside, Anna Shcherbina, and Anshul Kundaje. Not just a black box: Learning important features through propagating activation differences. *arXiv preprint arXiv:1605.01713*, 2016. 36

[84] Patrice Y Simard, David Steinkraus, John C Platt, et al. Best practices for convolutional neural networks applied to visual document analysis. In *Icdar*, **3**. Citeseer, 2003. 14

[85] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013. 7, 23, 35

[86] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pages 3319–3328. PMLR, 2017. 7, 35

[87] Anna Louise Swan, Ali Mobasheri, David Allaway, Susan Liddell, and Jaume Bacardit. Application of machine learning to proteomics data: classification and biomarker identification in postgenomics biology. *Omics: a journal of integrative biology*, **17**[12]:595–610, 2013. 4

[88] Yaniv Taigman and Marc Aurelio Ranzato. Tel aviv, and menlo park. deepface: Closing the gap to human-level performance in face verification. CVPR, 2014. 22

[89] Jie Tan, John H Hammond, Deborah A Hogan, and Casey S Greene. Adage-based integration of publicly available pseudomonas aeruginosa gene expression data with denoising autoencoders illuminates microbe-host interactions. *MSystems*, **1**[1]:e00025–15, 2016. 7

[90] Jie Tan, Matthew Ung, Chao Cheng, and Casey S Greene. Unsupervised feature construction and knowledge extraction from genome-wide assays of breast cancer with denoising autoencoders. In *Pacific Symposium on Biocomputing Co-Chairs*, pages 132–143. World Scientific, 2014. 7, 24, 25, 38

[91] Jonathan A Tepper, Mahmud S Shertil, and Heather M Powell. On the importance of sluggish state memory for learning long term dependency. *Knowledge-Based Systems*, **96**:104–114, 2016. 7, 9, 20, 30, 38

[92] Claudia Ulbricht. Multi-recurrent networks for traffic forecasting. In *AAAI*, pages 883–888, 1994. 20

[93] Ramzan Kh Umarov and Victor V Solovyev. Recognition of prokaryotic and eukaryotic promoters using convolutional deep learning neural networks. *PloS one*, **12**[2]:e0171410, 2017. 16

[94] Ewa Urbanczyk-Wochniak, Alexander Luedemann, Joachim Kopka, Joachim Selbig, Ute Roessner-Tunali, Lothar
Willmitzer, and Alisdair R Fernie. Parallel analysis of transcript and metabolic profiles: a new approach in systems biology. *EMBO reports*, **4**[10]:989–993, 2003. 4

[95] Betty van Aken, Benjamin Winter, Alexander Loser, and Fe-¨ lix A Gers. How does bert answer questions? a layer-wise analysis of transformer representations. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1823–1832,
2019. 7

[96] Laura J Van't Veer, Hongyue Dai, Marc J Van De Vijver, Yudong D He, Augustinus AM Hart, Mao Mao, Hans L Peterse, Karin Van Der Kooy, Matthew J Marton, Anke T Witteveen, et al. Gene expression profiling predicts clinical outcome of breast cancer. *nature*, **415**[6871]:530, 2002. 3

[97] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and PierreAntoine Manzagol. Extracting and composing robust features with denoising autoencoders machine learning. In *Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June*, 2008. 24

[98] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, **11**[Dec]:3371–3408, 2010. 14

[99] Mei Wang and Weihong Deng. Deep face recognition: A survey. *CoRR*, **abs/1804.06655**, 2018. 22

[100] John N Weinstein, Eric A Collisson, Gordon B Mills, Kenna R Mills Shaw, Brad A Ozenberger, Kyle Ellrott, Ilya Shmulevich, Chris Sander, Joshua M Stuart, Cancer Genome Atlas Research Network, et al. The cancer genome atlas pan-cancer
analysis project. *Nature genetics*, **45**[10]:1113, 2013. 4, 37

[101] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, **78**[10]:1550–1560, 1990. 19

[102] Ronald J Williams and Jing Peng. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural computation*, **2**[4]:490–501, 1990. 19

[103] Ian H Witten, Eibe Frank, Mark A Hall, and Data Mining.
Practical machine learning tools and techniques. *Morgan Kaufmann*, 2011. 30, 33, 34

[104] Sung Hwan Won, Iickho Song, Sun Young Lee, and Cheol Hoon
Park. Identification of finite state automata with a class of recurrent neural networks. *IEEE transactions on neural networks*, **21**[9]:1408–1421, 2010. 30, 38

[105] Raymond E Wright. Logistic regression. 1995. 21

[106] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and
Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015. 23

[107] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014. 22, 24, 35, 37, 38

[108] Matthew D Zeiler, Graham W Taylor, and Rob Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *2011 International Conference on Computer Vision*, pages 2018–2025. IEEE, 2011. 22, 38

[109] Sai Zhang, Jingtian Zhou, Hailin Hu, Haipeng Gong, Ligong Chen, Chao Cheng, and Jianyang Zeng. A deep learning framework for modeling structural features of rna-binding protein targets. *Nucleic acids research*, **44**[4]:e32–e32, 2016. 14

[110] Zhiqiang Zhang, Yi Zhao, Xiangke Liao, Wenqiang Shi, Kenli Li, Quan Zou, and Shaoliang Peng. Deep learning in omics: a survey and guideline. *Briefings in functional genomics*, **18**[1]:41–57, 2019. 6, 16

[111] Jian Zhou and Olga G Troyanskaya. Predicting effects of noncoding variants with deep learning–based sequence model. *Nature methods*, **12**[10]:931, 2015. 6

[112] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*, **67**[2]:301–320, 2005. 33