bioRxiv preprint doi: https://doi.org/10.1101/563601; this version posted February 28, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

Deep learning in bioinformatics: introduction, application, and perspective in big data era

Yu Li
KAUST
CBRC
CEMSE

Chao Huang
NICT
CAS

Lizhong Ding
IIAI

Zhongxiao Li
KAUST
CBRC
CEMSE

Yijie Pan
NICT
CAS

Xin Gao *
KAUST
CBRC
CEMSE

Abstract

Deep learning, which is especially formidable in handling big data, has achieved great success in various fields, including bioinformatics. With the advances of the big data era in biology, it is foreseeable that deep learning will become increasingly important in the field and will be incorporated in vast majorities of analysis pipelines. In this review, we provide both the exoteric introduction of deep learning, and concrete examples and implementations of its representative applications in bioinformatics. We start from the recent achievements of deep learning in the bioinformatics field, pointing out the problems which are suitable to use deep learning. After that, we introduce deep learning in an easy-to-understand fashion, from shallow neural networks to legendary convolutional neural networks, legendary recurrent neural networks, graph neural networks, generative adversarial networks, variational autoencoder, and the most recent state-of-the-art architectures. After that, we provide eight examples, covering five bioinformatics research directions and all the four kinds of data type, with the implementation written in Tensorflow and Keras. Finally, we discuss the common issues, such as overfitting and interpretability, that users will encounter when adopting deep learning methods and provide corresponding suggestions. The implementations are freely available at `https://github.com/lykaust15/Deep_learning_examples`.

1 Introduction

With the significant improvement of computational power and the advancement of big data, deep learning has become one of the most successful machine learning algorithms in recent years [77]. It has been continuously refreshing the state-of-the-art performance of many machine learning tasks [72, 152, 60, 151] and facilitating the development of numerous disciplines [10, 149, 42, 91]. For example, in the computer vision field, methods based on convolutional neural networks have already dominated its three major directions, including image recognition [72, 52, 60], object detection [126, 125], and image inpainting [178, 186] and super-resolution [188, 78]. In the natural language processing field, methods based on recurrent neural networks usually represent the state-of-the-art performance in a broad range of tasks, from text classification [197], to speech recognition [55] and machine translation [151]. Researchers in the high energy physics field have also been using deep

*All correspondence should be addressed to Xin Gao (xin.gao@kaust.edu.sa).

Preprint. Work in progress.

bioRxiv preprint doi: https://doi.org/10.1101/563601; this version posted February 28, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

Table 1: Abbreviations.

Abbreviations	Full words	Reference
AUC	Area under curve	[171]
CNN	Convolutional neural networks	[72]
Cryo-EM	Cryogenic electron microscopy	[102]
Cryo-ET	Cryogenic electron tomography	[50]
CT	Computed tomography	[76]
DenseNet	Densely connected convolutional networks	[60]
DNN	Deep fully connected neural networks	[77]
DPN	Dual path networks	[16]
DR	Dimensionality reduction	[129]
EEG	Electroencephalogram	[149]
GAN	Generative adversarial networks	[43]
GCN	Graph convolutional neural networks	[69]
RNN	Recurrent neural networks	[103]
ResNet	Residual networks	[52]
LSTM	Long short-term memory	[44]
MRI	Magnetic resonance imaging	[133]
PET	Positron emission tomography	[84]
PseAAC	Pseudo amino acid composition	[20]
PSSM	Position specific scoring matrix	[85]
ReLU	Rectified linear unit	[108]
SENet	Squeeze-and-excitation networks	[58]
SGD	Stochastic gradient descent	[196]
VAE	Variational auto-encoder	[30]

learning to promote the search of exotic particles [10]. In psychology, deep learning has been used to facilitate Electroencephalogram (EEG) (the abbreviations used in this paper are summarized in Table 1) data processing [149]. Considering its application in material design [97] and quantum chemistry [135], people also believe deep learning will become a valuable tool in computational chemistry [42]. Within the computational physics field, deep learning has been shown to be able to accelerate flash calculation [91].

Meanwhile, deep learning has clearly demonstrated its power in promoting the bioinformatics field [104, 5, 18], including sequence analysis [198, 3, 25, 156, 87, 6, 80, 169, 157, 158, 175], structure prediction and reconstruction [167, 90, 38, 168, 180, 196, 170], biomolecular property and function prediction [85, 204, 75, 4], biomedical image processing and diagnosis [35, 66, 41, 22, 160], and biomolecule interaction prediction and systems biology [95, 201, 203, 144, 145, 67, 165, 191]. Specifically, regarding sequence analysis, people have used deep learning to predict the effect of noncoding sequence variants [198, 166], model the transcription factor binding affinity landscape [25, 3, 166], improve DNA sequencing [87, 154] and peptide sequencing [156], analyze DNA sequence modification [143], and model various post-transcription regulation events, such as alternative polyadenylation [81], alternative splicing [80], transcription starting site [159, 157], noncoding RNA [181, 7] and transcript boundaries [139]. In terms of structure prediction, [167, 54] use deep learning to predict the protein secondary structure; [38, 196] adopt deep learning to model the protein structure when it interacts with other molecules; [170, 180, 168] utilize deep neural networks to predict protein contact maps and the structure of membrane proteins; [90] accelerates the fluorescence microscopy super-resolution by combining deep learning with Bayesian inference. Regarding the biomolecular property and function prediction, [85] predicts enzyme detailed function by predicting the Enzyme Commission number (EC numbers) using deep learning; [75] deploys deep learning to predict the protein Gene Ontology (GO); [4] predicts the protein subcellular location with deep learning. There are also a number of breakthroughs in using deep learning to perform biomedical image processing and biomedical diagnosis. For example, [35] proposes a method based on deep neural networks, which can reach dermatologist-level performance in classifying skin cancer; [66] uses transfer learning to solve the data-hungry problem to promote the automatic medical diagnosis; [22] proposes a deep learning method to automatically predict fluorescent labels from transmitted-light images of unlabeled biological samples; [41, 160] also propose deep learning methods to analyze

bioRxiv preprint doi: https://doi.org/10.1101/563601; this version posted February 28, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

Table 2: The applications of deep learning in bioinformatics.

Research direction	Data	Data types	Candidate models	References
Sequence analysis	Sequence data (DNA sequence, RNA sequence, *et al.*)	1D data	CNN, RNN	[198, 3, 25, 156, 87, 6, 80, 157, 158, 175, 169]
Structure prediction and reconstruction	MRI images, Cryo-EM images, fluorescence microscopy images, protein contact map	2D data	CNN, GAN, VAE	[167, 90, 38, 168, 180, 196, 170]
Biomolecular property and function prediction	Sequencing data, PSSM, structure properties, microarray gene expression	1D data, 2D data, structured data	DNN, CNN, RNN	[85, 204, 75, 4]
Biomedical image processing and diagnosis	CT images, PET images, MRI images	2D data	CNN, GAN	[35, 66, 41, 22, 160]
Biomolecule interaction prediction and systems biology	Microarray gene expression, PPI, gene-disease interaction, disease-disease similarity network, disease-variant network	1D data, 2D data, structured data, graph data	CNN, GCN	[95, 201, 203, 165, 71, 191, 67, 88]

the cell imagining data. As for the final main direction, i.e., biomolecule interaction prediction and systems biology, [95] uses deep learning to model the hierarchical structure and the function of the whole cell; [203, 165] adopt deep learning to predict novel drug-target interaction; [201] takes advantage of multi-modal graph convolutional networks to model polypharmacy sides effects.

In addition to the increasing computational capacity and the improved algorithms [61, 148, 52, 60, 86, 146], the core reason for deep learning's success in bioinformatics is the data. The enormous amount of data being generated in the biological field, which was once thought to be a big challenge [99], actually makes deep learning very suitable for biological analysis. In particular, deep learning has shown its superiority in dealing with the following biological data types. Firstly, deep learning has been successful in handling sequence data, such as DNA sequences [198, 166, 143], RNA sequences [110, 181, 7], protein sequences [85, 75, 4, 156], and Nanopore signal [87]. Trained with backpropagation and stochastic gradient descent, deep learning is expert in detecting and identifying the known and previously unknown motifs, patterns and domains hidden in the sequence data [140, 25, 3, 85]. Recurrent neural networks and convolutional neural networks with 1D filters are suitable for dealing with this kind of data. However, since it is not easy to explain and visualize the pattern discovered by recurrent neural networks, convolutional neural networks are usually the best choice for biological sequence data if one wants to figure out the hidden patterns discovered by the neural network [140, 3]. Secondly, deep learning is especially powerful in processing 2D and tensor-like data, such as biomedical images [90, 35, 22, 41, 160] and gene expression profiles [172, 14]. The standard convolutional neural networks [72] and their variants, such as residual networks [53], densely connected networks [60], and dual path networks [16], have shown impressive performance in dealing with biomedical data [22, 41]. With the help of convolutional layers and pooling layers, these networks can systematically examine the patterns hidden in the original map in different scales and map the original input to an automatically determined hidden space, where the high level representation is very informative and suitable for supervised learning. Thirdly, deep learning can also be used to deal with graph data [69, 202, 201, 88], such as symptom-disease networks [199], gene co-expression networks [184], protein-protein interaction networks [130] and cell system hierarchy [95], and promote the state-of-the-art performance [202, 201]. The core task of handling networks is to perform node embedding [48], which can be used to perform downstream analysis, such as node classification, interaction prediction and community detection. Compared to shallow embedding, deep learning based embedding, which aggregates the information for the node neighbors in a tree manner, has less parameters and is able to incorporate domain knowledge [48]. It can be seen that all the aforementioned three kinds of data are raw data without much feature extraction process when we input the data to the model. Deep learning is very good at handling the

bioRxiv preprint doi: https://doi.org/10.1101/563601; this version posted February 28, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

raw data since it can perform feature extraction and classification in an end-to-end manner, which determines the important high level features automatically. As for the structured data which have already gone through the feature extraction process, deep learning may not improve the performance significantly. However, it will not be worse than the conventional methods, such as support vector machine (SVM), as long as the hyper-parameters are carefully tunned. The applications of deep learning in those research directions and datasets are summarized in Table 2.

Considering the great potential of deep learning in promoting the bioinformatic research, to facilitate the the development and application, in this review, we will first give a detailed and thorough introduction of deep learning (Section 2), from shallow neural networks to deep neural networks and their variants mentioned above, which are suitable for biological data analysis. After that, we provide a number of concrete examples (Section 3) with implementation available on Github, covering five bioinformatic research directions (sequence analysis, structure prediction and reconstruction, biomolecule property and function prediction, biomedical image processing and diagnosis, and biomolecular interaction prediction and systems biology) and all the four types of data (1D sequences, 2D images and profiles, graphs, and preprocessed data). In terms of network types, those examples will cover fully connected neural networks, standard convolutional neural networks (CNN) [72], recurrent neural networks (RNN) [103], residual networks (ResNet) [53], generative adversarial networks (GAN) [43], variational autoencoder (VAE) [30], and graph convolutional neural networks (GCN) [69]. After those concrete examples, we will discuss the potential issues which researchers may encounter when using deep learning and the corresponding possible solutions (Section 4), including overfitting (Setion 4.2), data issues (Section 4.1 and 4.3), interpretability (Section 4.4), uncertainty scaling (Section 4.5), catastrophic forgetting (Section 4.6) and model compression (Section 4.7).

Notice that there are a number of other review papers available, introducing the applications of deep learning in bioinformatics, biomedicine and healthcare [82, 5, 104, 98, 65, 18, 164, 36, 155], which are summarized in Table 3. Despite their comprehensive survey of recent applications of deep learning methods to the bioinformatics field and their insightful points about the future research direction combining deep learning and biology, seldom have those reviews introduced how those algorithms work step-by-step or provided tutorial type of reviews to bridge the gap between the method developers and the end users in biology. Biologists, who do not have strong machine learning background and want incorporate the deep learning method into their data analysis pipelines, may want to know exactly how those algorithms work, avoiding the pitfalls that people usually encounter when adopting deep learning in data analytics. Thus, complementary to the previous reviews, in this paper, we provide both the exoteric introduction of deep learning, and concrete examples and implementations of its representative applications in bioinformatics, to facilitate the adaptation of deep learning into biological data analysis pipeline. This review is in tutorial-style. For the completeness of this review, we also surveyed the related works in recent years, as shown in Table 2, as well as the obstacles and corresponding solutions when using deep learning (Section 4).

2 From shallow neural networks to deep learning

In this section, we will first introduce the form of a shallow neural network and its core components (Section 2.1). After that, we introduce the key components of the standard CNN and RNN (Section 2.2). Since the standard CNN and RNN have been improved greatly over the past few years, we will also introduce several state-of-the-art architectures (Section 2.3), including ResNet, DenseNet and SENet [58]. After introducing the architecture for regular 1D and 2D data, we introduce graph neural networks for handling network data (Section 2.4). Then, we introduce two important generative models (Section 2.5), GAN and VAE, which can be useful for biomedical image processing and drug design. Finally, we give an overview of the currently available frameworks, which make the application of deep learning quite handy, for building deep learning models (Section 2.6). Notice that all the architectures and models are further illustrated and supported by the examples in Section 3. One can find the link between Section 2 and Section 3 with Table 4.

2.1 Shallow neural networks and their components

Fig. 1 shows the major components of a shallow neural network. In general, the whole network is a mapping function. Each subnetwork is also a mapping function. Taking the first hidden node as an example, which is shown in Fig. 1 (A), this building block function mapping contains two

bioRxiv preprint doi: https://doi.org/10.1101/563601; this version posted February 28, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

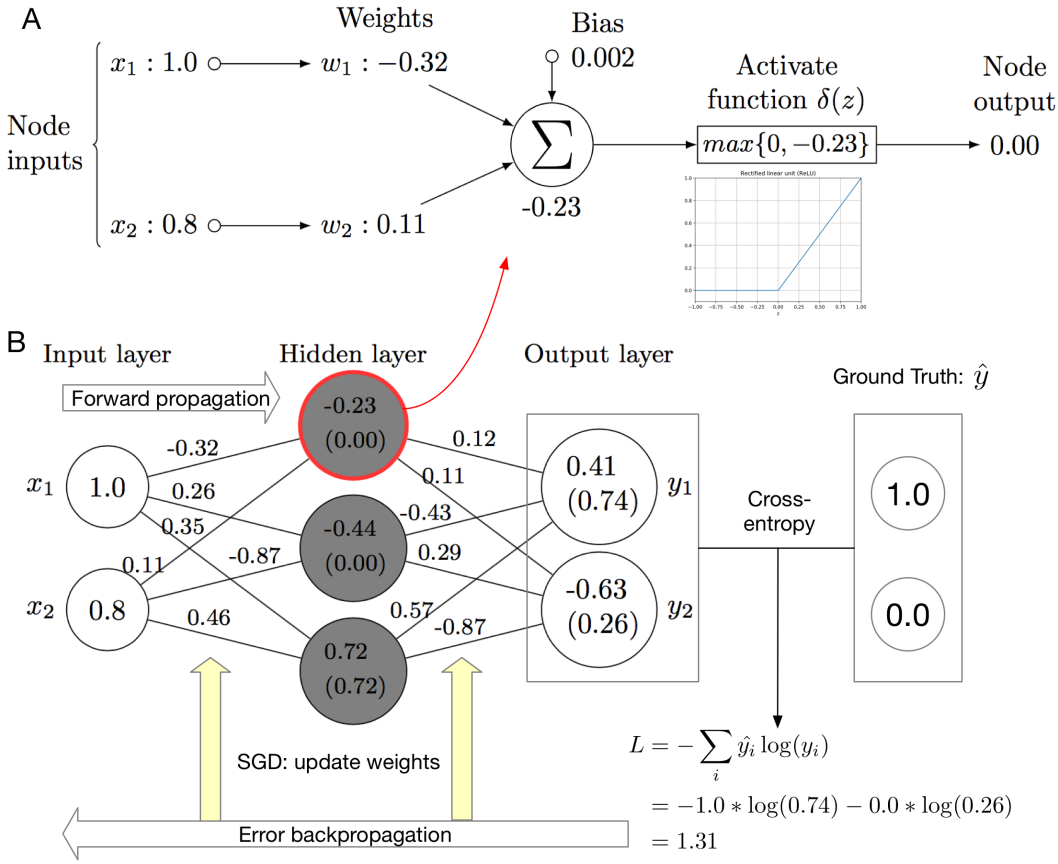


Figure 1: Illustration of a shallow neural network. (A) The operation within a single node. The input of the node will go through a linear transformation (dot product with the weight parameters and added with a bias) and a non-linear activation function to get the node output. (B) Illustration of the key components of a shallow neural network with one hidden layer. When training the model, we will run both the forward propagation and error back-propagation. Feeding the input value to the neural network and running forward propagation, we can obtain the output of the model, with which we can compare the target value and get the difference (loss or error). Then, we run error back-propagation and optimization to adjust the parameters of the model, making the output as close to the target value as possible. When the model is well-trained and we want to use the model, we will feed the input to the model and run forward propagation to obtain the output. Notice that in the hidden layer, the activation function is ReLU and the last output layer's activation function is Softmax. The numbers in the parenthesis are the results of applying activation functions to the linear transformation outputs.

bioRxiv preprint doi: https://doi.org/10.1101/563601; this version posted February 28, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

Table 3: Summaries of related review papers.

Reference	Key words	Summary
[82]	Machine learning, Genomic medicine	A review of machine learning tasks and related datasets in genomic medicine, not specific to deep learning
[5]	Regulatory genomics, Cellular imaging	A review of deep learning's applications in computational biology, with an emphasis on regulatory genomics and cellular imaging
[104]	Omics, Biomedical imaging, Biomedical signal processing	An general review of the CNN and RNN architectures, with the applications in omics, image processing and signal processing
[98]	Biomarker development, Transcriptomics	It discussed the key features of deep learning and a number of applications of it in biomedical studies
[93]	Medical imaging, CNN	It focuses on the applications of CNN in biomedical image processing, as well as the challenges in that field.
[65]	Protein sequences	A tutorial-style review of using deep learning to handle sequence data, with three application examples
[18]	Biomedicine, Human disease	It provides an review of the deep learning applications in biomedicine, with comprehensive discussion of the obstacles when using deep learning
[164]	Biomedicine, Model flexibility	It discusses the history and advantage of deep learning, arguing the potential of deep learning in biomedicine
[36]	Healthcare, End-to-end system	It discusses how techniques in computer vision, natural language processing, reinforcement learning can be used to build health care system.
[155]	High-performance system, Artificial intelligence, Medicine	It discusses the applications of artificial intelligence into medicine, as well as the potential challenges.
[17]	Model compression, Acceleration	A review of methods in compressing deep learning models and reducing the computational requirements of deep learning methods.
[111]	Catastrophic forgetting, Incremental learning	A survey of incremental learning in neural network, discussing how to incorporate new information into the model
[12]	Class imbalance	It investigates the impact of data imbalance to deep learning model's performance, with a comparison of frequently used techniques to alleviate the problem
[74]	Overfitting, Regularization	A systematic review of regularization methods used to combat overfitting issue in deep learning
[195]	Interpretability, Visual interpretation	It revisits the visualization of CNN representations and methods to interpret the learned representations, with an perspective on explainable artificial intelligence
[153]	Transfer learning, Data shortage	It reviews the transfer learning methods in deep learning field, dealing with the data shortage problem
[92]	Model interpretability	A comprehensive discussion of the interpretability of machine learning models, from the definition of interpretability to model properties

components, a linear transformation $\mathbf{w}*\mathbf{x}$ and a non-linear one $\delta(z)$. By aggregating multiple building block functions into one layer and stacking two layers, we obtain the network shown in Fig. 1 (B), which is capable of expressing a nonlinear mapping function with a relatively high complexity.

When we train a neural network to perform classification, we are training the neural network into a certain non-linear function to express (at least approximately) the hidden relationship between the features $\mathbf{x}$ and the label $\hat{\mathbf{y}}$. In order to do so, we need to train the parameters $\mathbf{w}$, making the model fit the data. The standard algorithm to do so is forward-backward propagation. After initializing the network parameters randomly, we run the network to obtain the network output (forward propagation). By comparing the output of the model and the ground truth label, we can compute the difference

bioRxiv preprint doi: https://doi.org/10.1101/563601; this version posted February 28, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

('loss' or 'error') between the two with a certain loss function. Using the gradient chain rule, we can back-propagate the loss to each parameter and update the parameter with certain update rule (optimizer). We will run the above steps iteratively (except for the random initialization) until the model converges or reaches a pre-defined number of iterations. After obtaining a well-trained neural network model, when we perform testing or use it, we will only run the forward propagation to obtain the model output given the input.

From the above description, we can conclude that multiple factors in different scales can influence the model's performance. Starting from the building block shown in Fig. 1 (A), the choice of the activation function can influence the model's capacity and the optimization steps greatly. Currently, the most commonly used activation functions are ReLU, Leaky ReLU, SELU, Softmax, TanH. Among them, ReLU is the most popular one for the hidden layers and the Softmax is the most popular one for the output layer. When combining those building blocks into a neural network, we need to determine how many blocks we want to aggregate in one layer. The more nodes we put into the neural network, the higher complexity the model will have. If we do not put enough nodes, the model will be too simple to express the complex relationship between the input and the output, which results in underfitting. If we put too many nodes, the model will be so complex that it will even express the noise in the data, which causes overfitting. We need to find the balance when choosing the number of nodes. After building the model, when we start training the model, we need to determine which loss function we want to use and which optimizer we prefer. The commonly used loss functions are cross-entropy for classification and mean squared error for regression. The optimizers include stochastic gradient descent (SGD), Momentum, Adam [68] and RMSprop. Typically, if users are not very familiar with the problem, Adam is recommended. If users have clear understanding of the problem, Momentum with learning rate decay is a better option.

2.2 Legendary deep learning architectures: CNN and RNN

2.2.1 Legendary convolutional neural networks

Although the logic behind the shallow neural network is very clear, there are two drawbacks of shallow neural networks. Firstly, the number of parameters is enormous: consider one hidden layer having N_1 nodes and the subsequent output layer containing N_2 nodes, then, we will have $N_1 * N_2$ parameters between those two layers. Such a large number of parameters can cause serious overfitting issue and slow down the training and testing process. Secondly, the shallow neural network considers each input feature independently, ignoring the correlation between input features, which is actually common in biological data. For example, in a DNA sequence, a certain motif may be very important for the function of that sequence [3]. Using the shallow network, however, we will consider each base within the motif independently, instead of the motif as a whole. Although the large amount of parameters may have the capability of capturing that information implicitly, a better option is to incorporate that explicitly. To achieve that, the convolution neural network has been proposed, which has two characteristics: local connectivity and weight sharing. We show the structure of a legendary convolutional neural network in Fig. 2. Fig. 2 (B) shows the data flow logic of a typical convolutional neural network, taking a chunk of Nanopore raw signals as input and predicting whether the corresponding DNA sequence is methylated or not. The Nanopore raw signals are already 1D numerical vectors, which are suitable for being fed into a neural network (in terms of string inputs, such as DNA sequences and protein sequences in the fasta format, which are common in bioinformatics, the encoding will be discussed in Section 3). After necessary pre-processing, such as denoising and normalization, the data vector will go through several (N) convolutional layers and pooling layers, after which the length of the vector becomes shorter but the number of channels increases (different channels can be considered to represent the input sequence from different aspects, like the RGB channels for an image). After the last pooling layer, the multi-channel vector will be flatten into a single-channel long vector. The long vector fully connects with the output layer with a similar architecture shown in Fig. 1.

Fig. 2 (A) shows the convolutional layer within CNN in more details. As shown in the figure, for each channel of the convolutional layer output, we have one weight vector. The length of this vector is shorter than the input vector. The weight vector slides across the input vector, performing inner product at each position and obtaining the convolution result. Then, an activation function is applied to the convolution result elementwisely, resulting in the convolutional layer output. Notice that each element of the output is only related to a certain part of the input vector, which is referred as local

bioRxiv preprint doi: https://doi.org/10.1101/563601; this version posted February 28, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

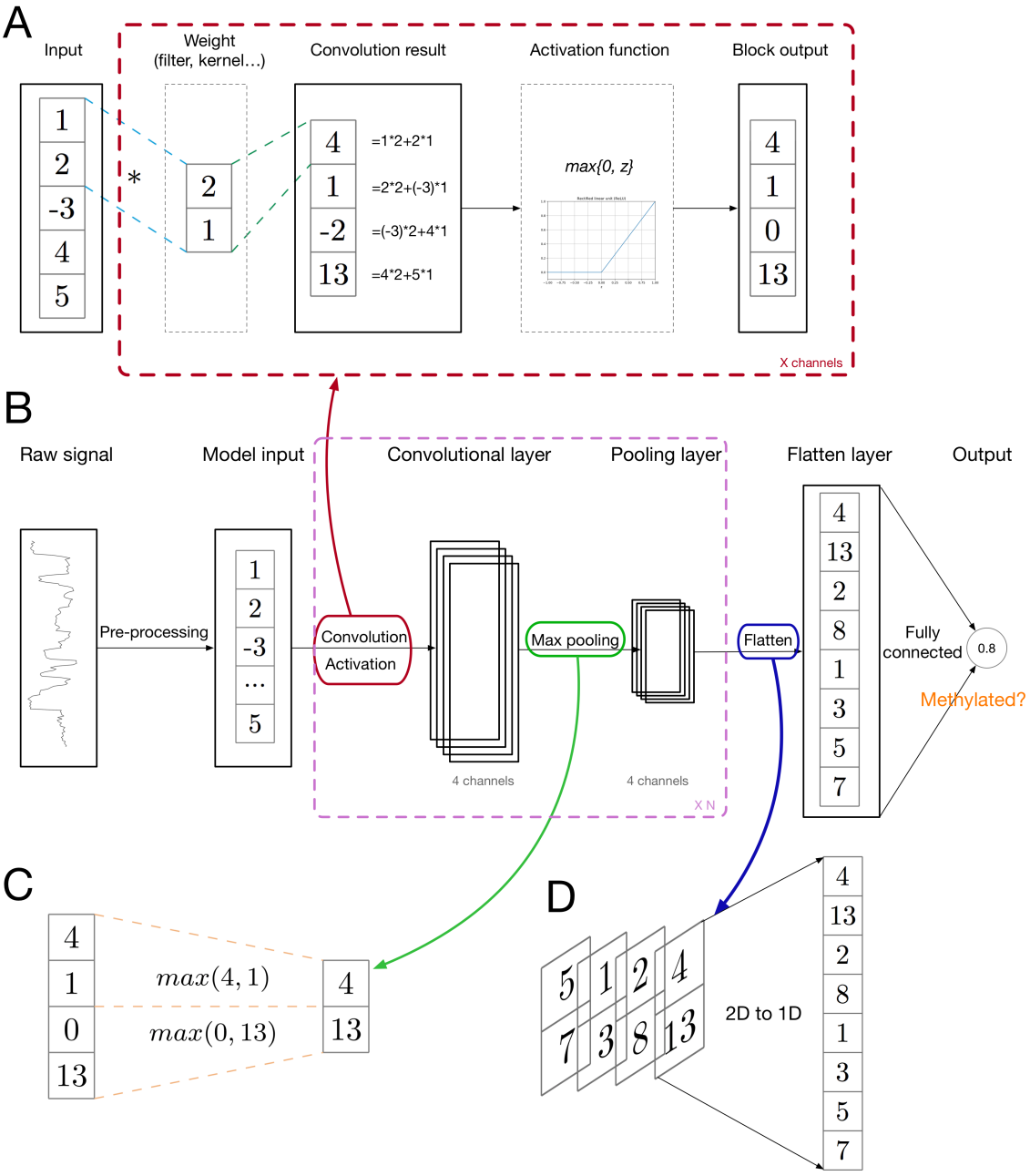


Figure 2: Illustration of a convolution neural network. (A) The convolutional layer operation. The input goes through a convolution operation and a non-linear activation operation to obtain the output. Those two operations can be repeated X times, so the output can have X channels. (B) The general data flow of a convolutional neural network. After pre-processing, the data usually go through convolutional layers, pooling layers, flatten layers, and fully connected layers to produce the final output. As for the other components, such as back-propagation and optimization, it is the same as the shallow neural network. (C) The max pooling operation. (D) The flatten operation, which converts a vector with multiple channels into a long vector with only one channel.

bioRxiv preprint doi: https://doi.org/10.1101/563601; this version posted February 28, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

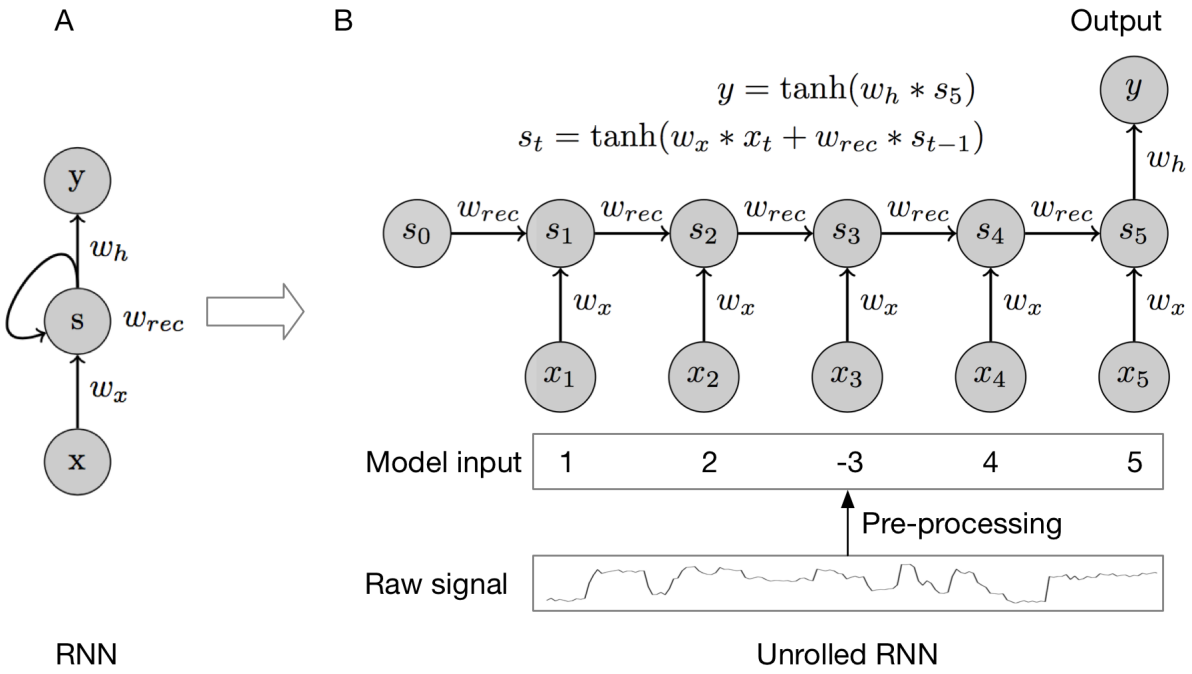


Figure 3: Illustration of a recurrent neural network. (A) A typical rolled RNN representation. (B) An easy-to-understand unrolled RNN representation. The model processes the input element by element which enables the model to capture the temporal information within the input.

connectivity. Besides, unlike the structure in Fig. 1, in convolutional layers, different parts of the input vector share the same weight vector when performing the sliding inner product (convolution), which is the weight-sharing property. Under this weight-sharing and local connectivity setting, we can find that the weight vector serves as a pattern or motif finder, which satisfies our original motivation of introducing this architecture. Notice that the original input is a vector with one channel. After the first convolutional layer, the input of the next convolutional layer usually has more than one channel. Correspondingly, the weight vector should have the same number of channels to make the convolutional operation feasible. For example, in Fig. 2 (B), the output of the first convolutional layer has 4 channels. Accordingly, the filters of the next convolutional layer should have 4 channels. If the filter length is the same as what is shown in Fig. 2 (A), which is 2, the filters should have the dimensionality as 2 by 4. If the output of the second convolutional layer has 16 channels, we should have 16 filters, one for each output channel. Considering all the above factors, the filter tensor of the second convolutional layer should have a dimensionality of 2*4*16, which is a 3D tensor. For image inputs, we usually have a 4D tensor as the filter of each convolutional layer, whose four dimensions correspond to filter length, filter width, input channels and output channels, respectively.

Fig. 2 (C) shows the max pooling operation. In the max pooling layer, each element of the output is the maximum of the corresponding region in the layer input. Depending on the applications, people can choose the pooling region size and steps (filter size and stride), or a different pooling method, such as average pooling. This pooling layer enables the network to capture higher level and long range property of the input vector, such as the long range interaction between the bases within the corresponding DNA sequences of the input signals. Fig. 2 (D) shows the flatten layer. This operation is straightforward, just concatenating the input vectors with multiple channels into a long vector with only one channel to enable the downstream fully connected layer operation.

With the convolutional layers and the pooling layers, this neural network architecture is able to capture the inputs' property in different levels and at different scales, including both the local motif and long range interaction.

bioRxiv preprint doi: https://doi.org/10.1101/563601; this version posted February 28, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

2.2.2 Legendary recurrent neural networks

In addition to the spatial dependency within the input, we also need to consider the temporal or order dependency in the input. For example, in DNA sequences, the order of motifs can influence the function of the sequence chunk [120]. In a document, we need to consider the order of words and sentences to categorize the document [94]. Similar to the case of spatial information, since the shallow neural networks shown in Fig. 1 consider each element of the input independently, the temporal information may also be lost. Recurrent neural networks are specifically designed to exploit the temporal relationship within a sequential input. The basic structure of a recurrent neural network is shown in Fig. 3. Let us consider an RNN with only one hidden recurrent cell, as shown in Fig. 3 (A). To explain how RNN works, we unroll the network into Fig. 3 (B), using the simplified Nanopore raw signals as an example. As in CNN, we first perform necessary pre-processing, such as normalization and denoising. Then the vector will be fed to the model element by element. The hidden recurrent cell has an initial state, which is denoted as s_0 and can be initialized randomly. After taking the first value, the recurrent node is updated, considering the previous state s_0 and x_1. Under our setting, the update rule is $s_1 = \tanh(w_x * x_1 + w_{rec} * s_0)$. When the second value comes in, the node state is updated to s_2 with the same rule. We repeat the above process until we consider all the elements. Then the node information will be fed forward to make predictions, e.g., whether the corresponding DNA sequence of the input Nanopore raw signals is methylated or not, with the similar downstream structure in Fig. 1. Notice that w_x and w_{rec} are shared among all the time steps or positions for that specific hidden recurrent node. Usually, one single recurrent node is not enough to capture all the temporal information. Under that circumstance, we can increase the number of recurrent nodes, each with its own pair of w_x and w_{rec}. For the last hidden layer, we can use a fully connected layer to connect all the hidden recurrent nodes to the output node, like we is done in the CNN scenario. In terms of training and optimization, it is the same as the case in shallow neural networks. We will run error back-propagation to make the weight parameters fit the training data and perform prediction afterwards.

2.3 State-of-the-art deep architectures

In the past several years, the above two legendary deep learning architectures have been improved greatly. For convolutional neural networks, AlexNet [72], VGG [142], GoogleNet [21, 152], ResNet [52, 53], ResNext [179], SENet [58], DenseNet [60] and DPN [16] have been proposed. For recurrent neural networks, there are LSTM [39], Bi-RNN [44], GRU [23], Memory network [173] and Attention network [162]. In Fig. 4, we exhibit some typical CNN architectures to deal with 2D image data, showing the evolving of convolutional neural networks. In general, the more advanced CNN architectures allow people to stack more layers, with the hope to extract more useful hidden informations from the input data at the expense of more computational resources. For recurrent neural networks, the more advanced architectures help people deal with the gradient vanishing or explosion issue and accelerate the execution of RNN. However, how to parallelize RNN is still a big problem under active investigation [187].

2.4 Graph neural networks

In this section, we briefly introduce graph neural networks [69, 48] to deal with network data, which is a common data type in bioinformatics. Unlike the sequence and image data, network data are irregular: a node can have arbitrary connection with other nodes. Although the data are complex, the network information is very important for bioinformatics analysis, because the topological information and the interaction information often have a clear biological meaning, which is a helpful feature for perform classification or prediction. When we deal with the network data, the primary task is to extract and encode the topological and connectivity information from the network, combining that information with the internal property of the node [48]. For example, Fig. 5 (A) shows a protein-protein interaction network and each node in the network represents a protein. In addition to the interaction information, each protein has some internal properties, such as the sequence information and the structure information. If we want to predict whether a protein in the network is an enzyme, both the interaction information and the internal properties can be helpful. So, what we need to do is to encode the protein in the network in a way that we consider both the network information and the properties, which is known as an embedding problem. In other words, we embed the network data into a regular space, as shown in Fig. 5 (B), where the original topological information is

bioRxiv preprint doi: https://doi.org/10.1101/563601; this version posted February 28, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

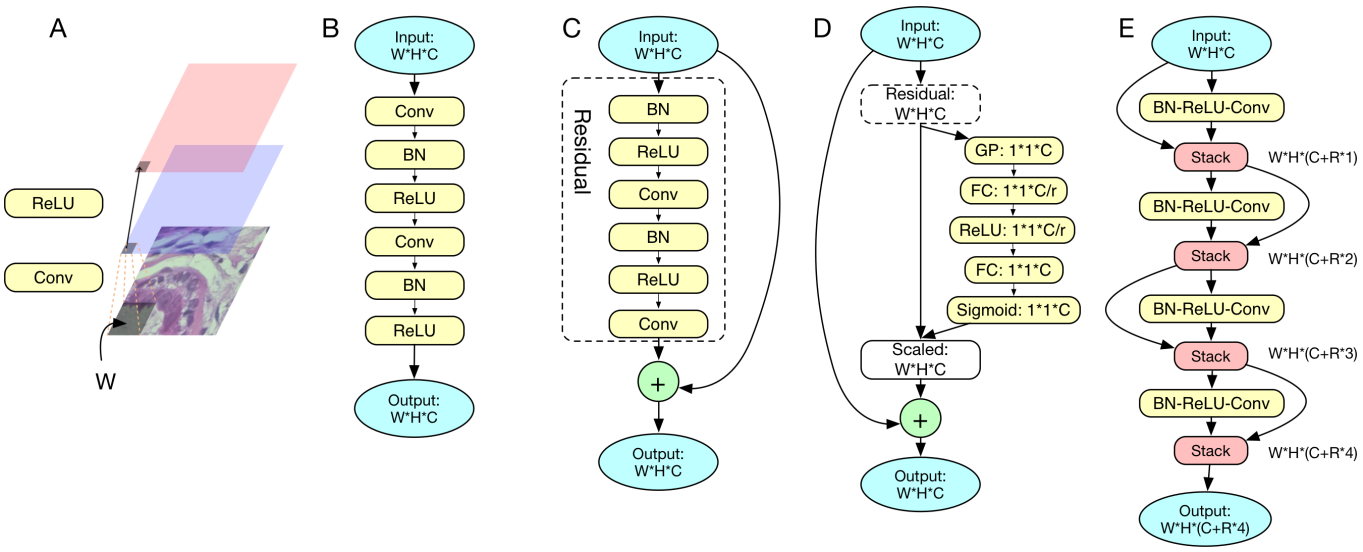


Figure 4: Illustration of different convolutional neural network architectures. (A) The convolutional neural network building block. 'Conv' represents the convolutional operation. 'ReLU' represents the activation function. (B) The legendary convolutional neural network with two convolutional layers. Each convolutional layer contains 'Conv', 'BN', and 'ReLU', where 'BN' represents batch normalization, which accelerates the training process [61]. (C) The residual block of a residual network [53]. In addition to the legendary convolutional layers along the data flow process, there is a shortcut between the input and the output, which adds the input elementwisely to the convolutional layer output to obtain the residual block output. In fact, unlike the architecture in (B), where the convolutional layers are used to model the mapping between the input and the output, here in (C), those convolutional layers are used to model the residual between the input and the residual block output. (D) The building block of SENet [58]. This architecture further improves the residual block, which introduces the idea of Attention [162] into the residual network. In (C), different channels of the output are considered of equal importance. However, in reality, some channels can contain more information and thus more important. The SENet block considers that, learning a scaling function and applying the scaling function to the output of the residual block to obtain the SENet block output. In the figure, 'GP' represents global pooling; 'FC' represents a fully connected layer. (E) The DenseNet block [60]. Unlike (C,D), which have the elementwise addition shortcut, DenseNet block's shortcut is stacking operation ('stack' in the figure), which concatenates the original input to the output of a certain convolutional layer along the channel dimension. This block can result in outputs with a large number of channels, which is proved to be a better architecture.

preserved. For an embedding problem, the most important thing is to aggregate information from a node's neighbor nodes [48]. Graph convolutional neural networks (GCN), shown in Fig. 5 (C) are designed for such a purpose. Suppose we consider a graph neural network with two layers. For each node, we construct a neighbor tree based on the network (Fig. 5 (C) shows the tree constructed for node *a* and node *b*). Then we can consider layer 0 as the neural network inputs, which can be the proteins' internal property encoding in our setting. Then, node *a*'s neighbors aggregate information from their neighbors, followed by averaging and activating, to obtain their level 1 representation. After that, node *a* collects information from its neighbors' level 1 representation to obtain its level 2 representation, which is the neural networks' output embedding result in this example.

Take the last step as an example, the information collection (average) rule is: $\mathbf{h}_{a,2} = \mathbf{w}_2 * \mathbf{x}_{average(b,c,d),1} + \mathbf{s}_2 * \mathbf{x}_{a,1}$, where $\mathbf{x}_{average(b,c,d),1}$ is the average of node *b,c,d*'s level 1 embedding, $\mathbf{x}_{a,1}$ is node *a*'s level 1 embedding, and $\mathbf{w}_2$ and $\mathbf{s}_2$ are the trainable parameters. To obtain the level 2 embedding, we apply an activation function to the average result: $\mathbf{x}_{a,2} = \sigma(\mathbf{h}_{a,2})$. Notice that between different nodes, the weights within the same neural layer are shared, which means the graph neural network can be generalized to previously unseen network of the same type. To train the graph neural network, we need to define a loss function, with which we can use the back-propagation to perform optimization. The loss function can be based on the similarity, that is, similar nodes should

bioRxiv preprint doi: https://doi.org/10.1101/563601; this version posted February 28, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

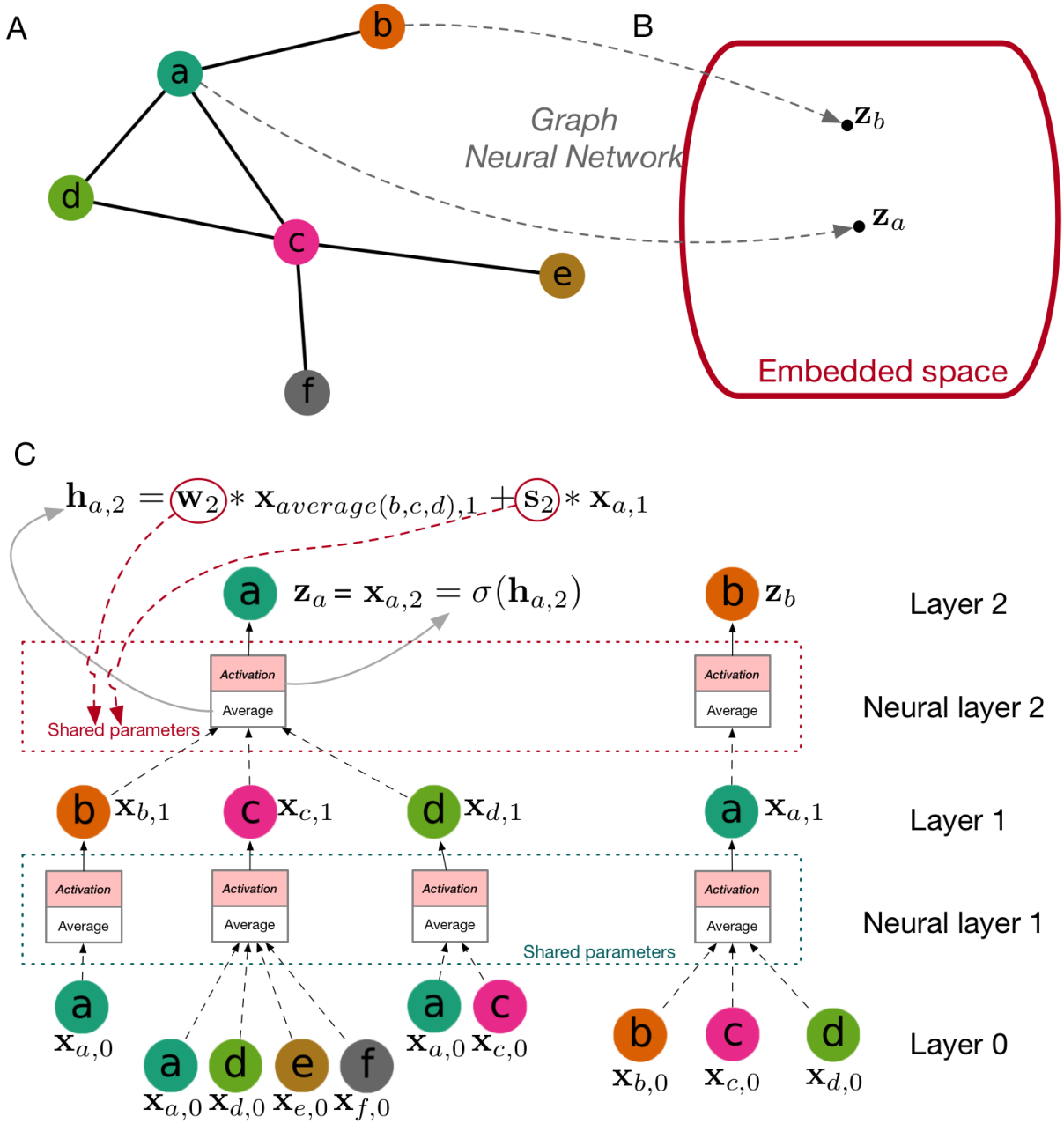


Figure 5: Illustration of a graph neural network. (A) A typical example of graph data. (B) The embedding space. In this embedding space, each data point is represented by a vector while the original topological information in (A) is preserved in that vector. (C) The graph neural network for embedding the network in (A). We use node *a* and *b* as examples. The internal properties of each node are considered as the original representations. In each layer, the nodes aggregate information from their neighbors and update the representations with averaging and activation function. The output of layer 2 are considered as the embedding result in this example. Notice that the parameters within the same layer between different trees are shared so this method can be generalized to previously unseen graph of the same type.

bioRxiv preprint doi: https://doi.org/10.1101/563601; this version posted February 28, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

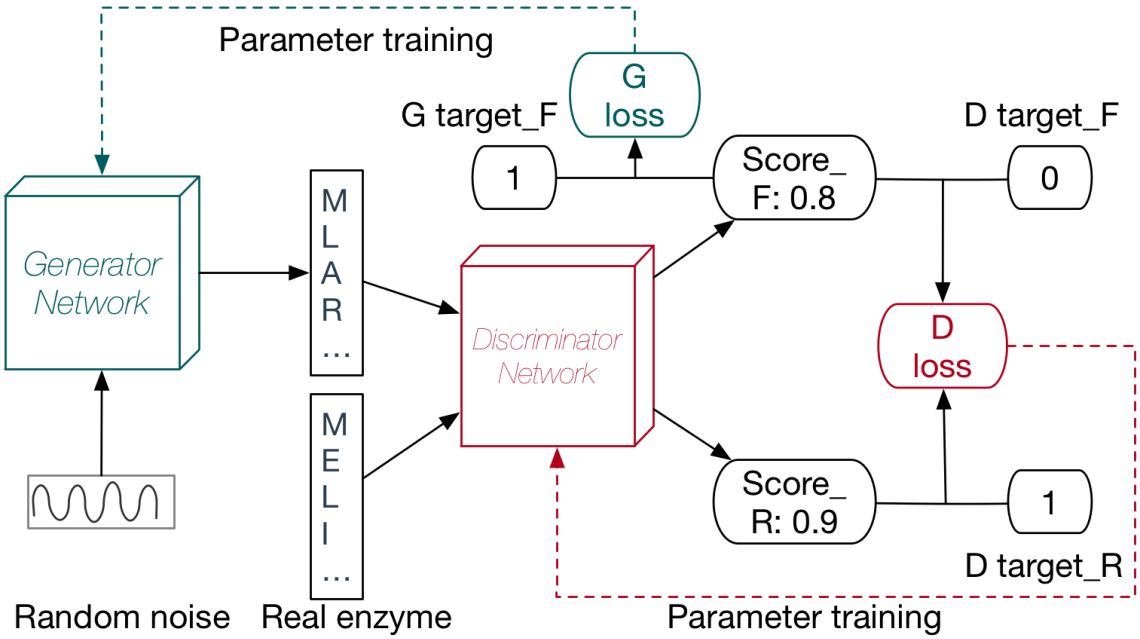


Figure 6: Illustration of GAN. In GAN, we have a pair of networks competing with each other at the same time. The generator network is responsible for generating new data points (enzyme sequences in this example). The discriminator network tries to distinguish the generated data points from the real data points. As we train the networks, both models' abilities are improved. The ultimate goal is to make the generator network able to generate enzyme sequences that are very likely to be real ones that have not been discovered yet. The discriminator network can use the architectures in Fig. 4. As for the generator network, the last layer should have the same dimensionality as the enzyme encoding.

have similar embeddings [45, 116, 127]. Or we can use a classification task directly to train the GCN in a discriminative way [33, 69, 202]: we can stack a shallow neural network, CNN or RNN, on the top of GNN, taking the embedding output of the GCN as input and training the two networks at the same time.

2.5 Generative models: GAN and VAE

In this section, we introduce two generative networks, GAN [43] and VAE [30], which can be useful for biological and biomedical image processing [183, 90, 137] and protein or drug design [132, 119, 122]. Unlike supervised learning, using which we perform classification or regression, such as the task of predicting whether a protein is an enzyme, the generative models belong to unsupervised learning, which cares more about the intrinsic properties of the data. With generative models, we want to learn the data distribution and generate new data points with some variations. For example, given a set of protein sequences which are enzymes as the training dataset, we want to train a generative model which can generate new protein sequences that are also enzymes.

Generative adversarial networks (GAN) have achieved great success in the computer vision field [43], such as generating new semantic images [43], image style transfer [62, 200], image inpainting [186] and image super-resolution [78, 90]. As shown in Fig. 6, instead of training only one neural network, GAN trains a pair of networks which compete with each other. The generator network is the final productive neural network which can produce new data samples (novel enzyme sequences in this example) while the discriminator network distinguishes the designed enzyme sequences from the real ones to push the generator network to produce protein sequences that are more likely to be enzyme sequences instead of some random sequences. Both of the generator network and the discriminator network can be the networks mentioned in Section 2.2.1. For the generator network, the last layer needs to be redesigned to match the dimensionality of an enzyme sequence encoding.

bioRxiv preprint doi: https://doi.org/10.1101/563601; this version posted February 28, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

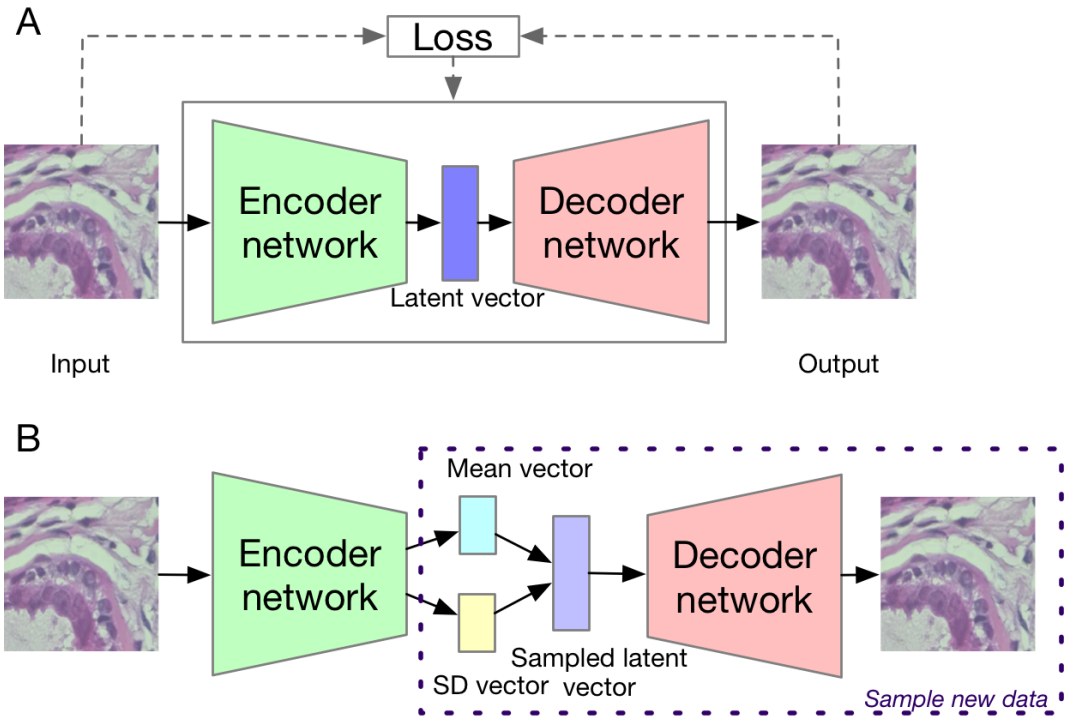


Figure 7: (A) Illustration of autoencoder, which is composed of encoder and decoder. The encoder network compresses the input into the latent vector and the decoder network reconstructs data from the latent vector. The loss is usually defined as the difference between the input and the decoder output. Notice that the latent vector's dimensionality is much smaller than the original input. (B) Illustration of VAE. To enable the network to sample new data points, instead of mapping the inputs into fixed latent vectors, VAE maps the inputs into a distribution in the latent space with the encoder network. When we need to sample new data, we can first sample the latent vector from the distribution and then generate the data using the decoder network with the sampled latent vector as input.

To introduce variational autoencoder, let us first go through autoencoder, which is shown in Fig. 7 (A). The autoencoder is usually used to encode high dimensional input data, such as an image, into a much lower dimensional representation, which can store the latent information of the data distribution. It contains two parts, the encoder network and the decoder network. The encoder network can transform the input data into a latent vector and the decoder network can reconstruct the data from the latent vector with the hope that the reconstructed image is as close to the original input as possible. Autoencoder is very convenient for dimensionality reduction. However, we cannot use it to generate new data which are not in the original input. Variational antoencoder overcomes the bottleneck by making a slight change to the latent space. Instead of mapping the input data to one exact latent vector, we map the data into a low dimensional data distribution, as shown in Fig. 7 (B). With the latent distribution, when we need to sample new data points, we can first sample the latent vector from the latent vector distribution and then construct the new data using the decoder network.

2.6 Frameworks

Building the whole network and implementing the optimizers completely from scratch will be very tedious and time-consuming. Fortunately, there are a number of handy frameworks available, which can accelerate the process of building networks and training the model. After evolving for several years, the following frameworks are commonly used, and actively developed and maintained: Tensorflow [1], Pytorch [112], Caffe2, MXNet [13], CNTK [138] and PaddlePaddle. Another famous warpper of Tensorflow is Keras, with which one can build a model in several lines of code. It is very convenient and easy-to-use, although it lacks the flexibility provided by Tensorflow, using which one can control almost every single detail.

bioRxiv preprint doi: https://doi.org/10.1101/563601; this version posted February 28, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

Table 4: Summary of the examples.

Example	Model	Data type	Research direction	Task
Enzyme function prediction	DNN	Structured	Biomolecular function prediction	Classification
Gene expression regression	DNN	Structured	Biomolecular property prediction	Regression
RNA-protein binding sites prediction	CNN	1D data	Sequence analysis	Classification
DNA sequence function prediction	CNN, RNN	1D data	Sequence analysis	Classification
Biomedical image classification	ResNet	2D data	Biomedical image processing	Classification
Protein interaction prediction	GCN	Graph	Biomolecule interaction prediction	Embedding, Classification
Biology image super-resolution	GAN	2D image	Structure reconstruction	Data generation
Gene expression data embedding	VAE	2D data	Systems biology	DR, Data generation

3 Applications of deep learning in bioinformatics

This section provides eight examples. Those examples are carefully selected, typical examples of applying deep learning methods into important bioinformatic problems, which can reflect all of the above discussed research directions, models, data types, and tasks, as summarized in Table 4. In terms of research directions, Sections 3.3 and 3.4 are related to sequence analysis; Section 3.7 is relevant to structure prediction and reconstruction; Section 3.1 is about biomolecular property and function prediction; Sections 3.7 and 3.5 are related to biomedical image processing and diagnosis; Sections 3.2, 3.6, and 3.8 are relevant to biomolecule interaction prediction and systems biology. Regarding the data type, Sections 3.1 and 3.2 use structured data; Sections 3.3 and 3.4 use 1D sequence data; Sections 3.5, 3.7, and 3.8 use 2D image or profiling data; Section 3.6 uses graph data.

From the deep learning point of view, those examples also cover a wide range of deep learning models: Sections 3.1 and 3.2 use deep fully connected neural networks; Section 3.3 uses CNN; Section 3.4 combines CNN with RNN; Section 3.6 uses GCN; Section 3.5 uses ResNet; Section 3.7 uses GAN; Section 3.8 uses VAE. We cover both supervised learning, including classification (Sections 3.1, 3.3, 3.4, and 3.5) and regression (Section 3.2), and unsupervised learning (Sections 3.6, 3.7, and 3.8). We also introduce transfer learning using deep learning briefly (Section 3.5).

3.1 Identifying enzymes using multi-layer neural networks

Enzymes are one of the most important types of molecules in human body, catalyzing biochemical reactions *in vivo*. Accurately identifying enzymes and predicting their function can benefit various fields, such as biomedical diagnosis and industrial bio-production [85]. In this example, we show how to identify enzyme sequences based on sequence information using deep learning based methods.

Usually, a protein sequence is represented by a string (such as, 'MLAC...'), but deep learning models, as mathematical models, take numerical values as inputs. So before building the deep learning model, we need to first encode the protein sequences into numbers. The common ways of encoding a protein sequences are discussed in [85]. In this example, we use a sparse way to encode the protein sequences, the functional domain encoding. For each protein sequence, we use HMMER [34] to search it against the protein functional domain database, Pfam [37]. If a certain functional domain is hit, we encode that domain as 1, otherwise 0. Since Pfam has 16306 functional domains, we have a 16306D vector, composed of 0s and 1s, to encode each protein sequence. Because the dimensionality of the feature is very high, the traditional machine learning method may encounter the curse of dimensionality. However, the deep learning method can handle the problem quite well. As for the dataset, we use the dataset from [85], which contains 22168 enzyme sequences and 22168 non-enzyme protein sequences, whose sequence similarity is under 40% within each class.

bioRxiv preprint doi: https://doi.org/10.1101/563601; this version posted February 28, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

In our implementation, we adopt a similar architecture as Fig. 1, but with much more nodes and layers. We use ReLU as the activation function, cross-entropy loss as the loss function, and Adam as the optimizer. We utilize dropout, batch normalization and weight decay to prevent overfitting. With the help of Keras, we build and train the model in 10 lines. Training the model on a Titan X for 2 minutes, we can reach around 94.5% accuracy, which is very close to the state-of-the-art performance [85]. Since bimolecular function prediction and annotation is one of the main research directions of bioinformatics, researchers can easily adopt this example and develop the applications for their own problems.

3.2 Gene expression regression

Gene expression data are one of the most common and useful data types in bioinformatics, which can be used to reflect the cellular changes corresponding to different physical and chemical conditions and genetic perturbations [161, 14]. However, the whole genome expression profiling can be expensive. To reduce the cost of gene profiling, realizing that different genes' expression can be highly correlated, researchers have developed an affordable method of only profiling around 1000 carefully selected landmark genes and predicting the expression of the other target genes based on computational methods and landmark gene expression [32, 14]. Previously, the most commonly used method is linear regression. Recently, [14] showed that deep learning based regression can outperform the linear regression method significantly since it considered the non-linear relationship between genes' expression.

In this example, we use deep learning method to perform gene expression prediction as in [14], showing how to perform regression using deep learning. We use the Gene Expression Omnibus (GEO) dataset from [14], which has already gone through the standard normalization procedure. For the deep learning architecture, we use a similar structure as in Section 3.1. However, the loss function for this regression problem is very different from the classification problem in Section 3.1. As we discussed in Section 2.1, for classification problems, we usually use cross-entropy loss, while for the regression problem, we use the mean squared error as the loss function. Besides, we also change the activation function of the last layer from Softmax to TanH for this application. Using Keras, the network can be built and trained in 10 lines. Trained on a Titan X card for 2 mins, it can outperform the linear regression method by 4.5% on a randomly selected target gene.

Our example code can be easily used and adopted for other bioinformatics regression problems. On the other hand, if one is sure about the regression target value range, then the target range can be divided into several bins and the regression problem is converted into a classification problem. With the classification problem in hand, people can use the classification examples introduced in the other sections.

3.3 RNA-protein binding sites prediction with CNN

RNA-binding proteins (RBP) play an important role in regulating biological processes, such as gene regulation [40, 110]. Understanding their behaviors, for example, their binding site, can be helpful for cuing RBP related diseases. With the advancement of high-throughput technologies, such as CLIP-seq, we can verify the RBP binding sites in batches [83]. However, despite its efficiency, those high-throughput technologies can be expensive and time-consuming. Under that circumstance, machine learning based computational methods, which are fast and affordable, can be helpful to predict the RBP binding site [110].

In fact, the deep learning methods are especially suitable for this kind of problems. As we know, RBP can have sequence preference, recognizing specific motifs and local structures and binding to those points specifically [123]. On the other hand, as we discussed in Section 2.2.1, CNN are especially good at detecting special patterns in different scales. Previous studies have shown the power of CNN in finding motifs [110, 25].

In this example, we show how to predict the RBP binding site using CNN, with the data from [110]. Specifically, the task is to predict whether a certain RBP, which is fixed for one model, can bind to a certain given RNA sequence, that is, a binary classification problem. We use one-hot encoding to convert the RNA sequence strings of 'AUCG' into 2D tensors. For example, for 'A', we use a vector $(1, 0, 0, 0)$ to represent it; for 'U', we use a vector $(0, 1, 0, 0)$ to represent it. Concatenating those 1D vectors into a 2D tensor in the same order as the original sequence, we obtain the one-hot encoding

bioRxiv preprint doi: https://doi.org/10.1101/563601; this version posted February 28, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

for a certain RNA sequence. We use a similar architecture as the model shown in Fig. 2. We use similar settings as the previous examples in terms of the activation function, the optimizer, the loss function, *etc.*. Notice that for the one-hot encoding, we can consider it either as a 2D map with 1 channel or a 1D vector with 4 channels. Correspondingly, for the convolutional layers, we can choose either 2D convolutions or 1D convolutions. In this example, we follow the previous research setting, using 2D convolutions. The original implementation [110] is in Pytorch, which is very lengthy, we reimplemented the idea using Keras, which builds the model in 15 lines.

Furthermore, this example can be easily adopted for other applications with similar background. That is, there are motif or pattern preferences within the input, such as transcription starting site identification [159, 157] and Poly(A) site prediction [177].

3.4 DNA sequence function prediction with CNN and RNN

Understanding the properties and functions of DNA sequences, which is the most important genetic material, is a profound and challenging task for bioinformatics and biology [163]. Among those sequences, the non-coding DNA functionality determination is especially challenging, since over 98% of human genome are non-coding DNA and it is very difficult to perform biological experiment to investigate functionality of every non-coding DNA sequence chunk [113]. Computational methods, which are cheap and highly parallelable, can help people address the problem greatly. Previous studies have shown the success of using deep learning to predict the functionality of non-coding DNA sequences [198, 120].

In this example, we show how to use CNN and RNN to predict the functionality of non-coding DNA sequences. We use the data from [198]. As described in [198, 120], the human GRCh37 reference genome was segmented into non-overlapping 200-bp bins. The inputs of the deep learning model are the 1000-bp DNA sequences which are centered on the 200-bp bins. In terms of the labels of those sequences, they were generated by collecting profiles from ENCODE and Roadmap Epigenomics data releases, which resulted in a 919 binary vector for each sequence (690 transcription factor binding profiles, 125 DNase I–hypersensitive profiles and 104 histone-mark profiles). To encode the DNA sequence string into a mathematical form which can be fed to the model, we use the one-hot encoding as discussed in Section 3.3. In terms of the model, because for DNA sequences, not only do the specific motifs matter, but also the interaction between the upstream and downstream motifs also plays important roles in determining the sequence functionality, we combine CNN, as shown in Fig. 2, and RNN, as shown in Fig. 3, stacking a bi-directional LSTM layer on top of 1D convolutional layers. The original implementation of [120] requires Theano, which has been discontinued. We reimplemented the idea solely using Keras.

This example can be adopted to perform other important predictions, which are similar to this problem, for DNA and RNA sequences, such as DNA methylation state prediction [6] and long non-coding RNA function prediction [101].

3.5 Biomedical image classification using transfer learning and ResNet

The classification of biomedical data has a broad range of applications in biomedical diagnosis. For example, the classification of biomedical images has been used to assist the diagnosis of skin cancer [35] and retinal diseases [66], which even reaches the expert-level performance. Researchers have also used deep learning to classify the electronic health record (EHR), predicting several medical events, such as, in-hospital mortality, which set the new state-of-the-art performance [121]. However, on the other hand, although deep learning has shown great capability of dealing with the image data, it is highly data-hungry. For example, the famous ImageNet dataset contains over 10 million images [27]. In terms of biomedical data, because of technology limitation and privacy issues, that amount of data is usually not available. In order to maximize the power of deep learning in dealing with biomedical data, transfer learning [185] is usually used. Transfer learning is an important machine learning direction, which has its own field [109]. But for deep learning models, people usually perform transfer learning in the following way. They first train a standard deep learning model for image classification on the ImageNet dataset. After that, people use the real dataset to fine-tune the model. When fine-tunning the model, they freeze the weights of convolutional layers and use a new fully-connected layer, which has the same number of nodes as the real application's classes, to substitute the last fully-connected layer in the standard model. Then, they train the weights of that

bioRxiv preprint doi: https://doi.org/10.1101/563601; this version posted February 28, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

new layer. Those frozen convolutional layers are considered as the feature extractor and the newly added fully-connected layer can be considered as the classifier. Although the biomedical images can have some differences from the ImageNet dataset, they usually share similar features. Using transfer learning, the model's performance can be improved significantly [66].

In this example, we use the chest X-ray dataset from [66]. We use Keras to implement this example. We first load the ResNet, whose basic idea is shown in Fig. 4 (C), trained on ImageNet, and then freeze all the layer's weights, except for the last 4 layers'. We substitute the last layer with a new layer containing 2 nodes, since that dataset has two classes. Besides, we resize the chest X-ray images to make them the same dimensionality as the original ResNet input image using bilinear interpolation. Finally, we run the standard optimization procedure with cross-entropy loss and Adam optimizer. In order to prevent overfitting, like what we have done in the previous examples, we use dropout [148] combined with batch normalization [61] and weight decay.

This example can be easily adopted for other biomedical image processing and classification applications [93], such as breast cancer classification [147], CT image classification [76] and fMRI imaging classification [133].

3.6 Graph embedding for novel protein interaction prediction using GCN

In this example, we use graph embedding to handle protein-protein interaction (PPI) networks [49]. Specifically, we want to predict novel protein-protein interaction in the yeast PPI network. Understanding protein-protein interaction can be helpful for predicting the functionality of uncharacterized proteins and designing drugs [136, 130]. Recently, because the development of large-scale PPI screening techniques, such as the yeast two-hybrid assay [63], the amount of PPI data has increased greatly. Despite the development of those techniques, the current PPI network is still incomplete and noisy, since there are so many proteins existing and so do the complicated interactions. Using computational methods to predict the interaction can be a convenient way to discover novel and important protein-protein interactions [45].

We use the yeast protein-protein interaction network as the dataset. In terms of the model, we utilize the graph neural network [69] described in Fig. 5, which can be used to learn the node (protein) embedding based on the network topology. To define the interaction from node embedding, we use the inner product of the embeddings of two nodes as the interaction operation. The higher the product is, the more confidently we believe they have an interaction. Since we have already known some interactions within the network, that is, the existing edges, we can train the graph neural network in a supervised way, using the existing edges (and of course, some protein pairs which do not have interactions as the negative training data) as targets and the cross-entropy as the loss function. After running the optimization until the model converges, we can obtain stable node embeddings for each node based on the network topology. Then we apply the interaction operation (inner product) to each pair of nodes. If the result is higher than a certain threshold, we will predict that there is an interaction between the two nodes. We implemented this graph example using Tensorflow.

Notice that the graph data are very common in bioinformatics, such as symptoms-disease networks [199], gene co-expression networks [184], and cell system hierarchy [95]. This example can be easily adopted for the other network problems.

3.7 Biology image super-resolution using GAN

Although, currently, the usage of GAN is still limited to image processing and structure reconstruction [90, 15], as a generative model, it has the potential for designing new molecules and drugs [132, 122]. To achieve this goal, researchers need to make much more efforts. In this example, we will introduce how to use GAN to perform a legendary but useful task: image super-resolution. In biology, due to the limitation of the imaging acquiring technology, the obtained images are often noisy and of low resolution, such as Cryo-EM images [102] and fluorescence images [59], which requires post-processing to obtain high-resolution, noise-free images. The target of image super-resolution is to output high resolution images given low resolution ones. For example, the input image may only have the resolution as 60 by 60 but the output image can have a resolution as 240 by 240. We also require the super-resolution image to have more true details but less fake ones. This image super-resolution technique has achieved great success in fMRI image fast acquisition [15] and fluorescence microscopy super-resolution [90].

bioRxiv preprint doi: https://doi.org/10.1101/563601; this version posted February 28, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

As discussed in Section 2.5 and shown in Fig. 6, we train a pair of deep learning models: a generator network and a discriminator network. Given the low-resolution images, the generator network outputs the super-resolution images. The discriminator tries to distinguish the model-generated super-resolution images and the actual high-resolution ones. The discriminator network competes with the generator network so that it can push the generator network to produce the super-resolution images as real as possible. During training, we train both the two networks at the same time to make both of them better. Our ultimate goal is to make the generator network output super-resolution images very close to the real high-resolution ones even if the input low-resolution images are previously unseen in the training data. In terms of the dataset, we use the DIV2K dataset [2]. We use ResNet [53] as the generator and VGGNet [142] as the discriminator. We also add the perceptual loss to the loss function, which stabilizes the high level representation of the super resolution images [64]. We use Tensorflow combined with a higher-level package, Tensorlayer [31], to implement the GAN idea.

The example can be adopted for other biological or biomedical image processing applications, such as Cryo-EM images [102] and Cryo-ET images [46]. If data are available, this example probably can also be used for designing new proteins and new drugs [132, 122].

3.8 High dimensional biological data embedding and generation with VAE

As GAN, VAE is also a generative model. Thus, theoretically, VAE can also do what GAN can do. As discussed in Section 2.5, after we approximate the data distribution in the bottleneck layers, we can sample new data from the approximate distribution. Combining the idea of VAE and optimization, we are able to design more efficient drugs [132, 122]. At the same time, since the bottleneck layer's dimension is usually much smaller than the original input, this model can also be used to perform dimensionality reduction and mine the important features within the original input [172].

In this example, we briefly show how to use VAE to perform dimensionality reduction for gene expression data [172]. We use the dataset from [172], preprocessed from the TCGA database, which collects the gene expression data for over 10,000 different tumors. Within the database, the RNA-seq data describe the high-dimensional state of each tumor. In the dataset we use, the dimensionality is 5,000 for each tumor. Using VAE, we are able to reduce the dimensionality to 100. In that space, it is easier for us to identify the common patterns and signatures between different tumors. We used Keras to implement VAE. For the loss, we use mean-squared error.

In addition to gene expression data [117], this example can be adopted to other applications, which encounter the high dimensionality issue, such as protein fingerprints [26]. In terms of drug design, VAE will have a bright future in this direction, although more efforts need to be made [132].

4 Perspectives: limitations and suggestions

Currently, there are some difficulties that are frequently encountered when using deep learning. Here we list some of them and give the corresponding suggestions. We also list some related review papers in Table 3.

4.1 Lack of data

As we know, deep learning is very data-hungry, since it also contains the representation learning [86]. To obtain a deep learning model with good performance, we usually need much more data than the shallow algorithms. Besides, the more data we have, the better performance the deep learning model may achieve. Although under most circumstances, the biological data are enough to obtain a good deep learning model [99], sometimes the data may not be enough to directly use deep learning [66]. There are three suggested ways to deal with this difficulty. Firstly, we can collect the data from similar tasks and use the idea of transfer learning. Although the related data will not increase the amount of real data directly, those data can help learn a better mapping function and a better representation of the original input [185], and thus boost the performance of the model. On the other hand, we can also use a well trained model from another similar task and fine tune the last one or two layers using the limited real data (one example is shown in Section 3.5). A review of the transfer learning methods in the deep learning field can also be referred to [153]. Secondly, we can perform data augmentation [115]. This task is very useful for image data, because rotation, mirroring, and translation of an image usually do not change the label of the image. However, we should be careful

bioRxiv preprint doi: https://doi.org/10.1101/563601; this version posted February 28, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

when using this technique for bioinformatic data. For example, if we mirror an enzyme sequence, the resulted sequence may no longer be an enzyme sequence. Thirdly, we can consider using simulated data to increase the amount of training data. Sometimes, if the physical process behind the problem is well-known, we can build simulators based on the physical process, which can result in as much simulated data as we want. [90] gives us an example of handling the data requirement for deep learning using simulation.

4.2 Overfitting

Since deep learning models have very high model complexity, with huge amount of parameters which are related in a complex way, the models have high risks of getting overfitted to the training data and being unable to generalize well on the testing data [148]. Although this problem is not specific to the application of deep learning into bioinformatics, it comes along with almost every deep learning model, which should be fully considered and properly handled when adopting deep learning methods. Though recent studies [194, 146, 86] suggest that the implicit bias of the deep learning training process helps model get rid of serious overfitting issues, we still need some techniques to deal with the overfitting issue. Over the years, people have proposed a number of algorithms to alleviate the overfitting issue in deep learning, which can be classified into three types. The first type of techniques, which contains the most famous ones, acts on the model parameters and the model architecture, including dropout [148], batch normalization [61] and weight decay [73]. Dropout is the most well-known technique in the deep learning field. It regularizes the network by randomly discarding nodes and connections during training, which prevents the parameters from co-adapting to the training data too much [148]. Although batch normalization [61] was first proposed to deal with the internal covariate shift and accelerate the training process, it is also proved to be able to ease the overfitting issue [61]. Weight decay [73], as a universal regularizer, which is widely used in almost all the machine learning algorithms, is also one of the default techniques in the deep learning field. The second category acts on the inputs of the model, such as data augmentation and data corruption [96]. One of the reasons that deep learning models are prone to overfitting is that we do not have enough training data so that the learned distribution may not reflect the actual distribution. Data augmentation increases the amount of training data explicitly and marginalized data corruption [96] can help solve the problem without augmenting the data explicitly. The last type acts on the output of the model. Recently, a method [114] was proposed to regulate the model by penalizing the over-confident outputs, which was shown to be able to regulate both CNN and RNN. A systematic review of regularization methods used to combat the overfitting issue in deep learning can be referred to [74].

4.3 Imbalanced data

The biological data are usually imbalanced, with the positive samples being largely outnumbered by the negative ones [182]. For example, the number of non-enzyme proteins is much larger than that of a certain type of enzyme proteins [85]. The data imbalancing issue also appears in transcription starting site prediction [159], Poly(A) site prediction [176], *etc.*. Using the imbalanced data to train a deep learning model may result in undesirable results. For example, the model might predict all the test data with the label from the largest class: all the data, which is composed of 99 negative and 1 positive samples, are predicted to be negative. Although the model's performance is excellent if evaluated using accuracy (as high as 99%), the result is in fact terrible as we consider the performance on the small class. To solve the issue, we can use the following techniques. Firstly, we need to use the right criteria to evaluate the prediction result and the loss. For the imbalanced data, we want the model to perform well not only on the large classes but also on the smaller ones. As a result, we should use AUC as the criteria and the corresponding loss [171]. Secondly, if we still want to use the cross entropy loss, we can use the weighted cross entropy loss which penalizes the model if it performs terribly on the smaller classes. At the same time, we can upsample smaller classes or downsample larger ones when training the model. Finally, since biological systems often have hierarchical label space, we can build models for each hierarchical level to make the data balanced, as shown in [85]. For the readers' reference, [12] investigates the impact of data imbalance to deep learning model's performance comprehensively, with a comparison of frequently used techniques to alleviate the problem, although those techniques are not specific to biological problems.

bioRxiv preprint doi: https://doi.org/10.1101/563601; this version posted February 28, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

4.4 Interpretability

Although deep learning methods are sometimes criticized for acting like a black-box, it is in fact interpretable [92]. In the bioinformatics field, we usually want to interpret deep learning in a way that we can know the useful patterns and motifs detected by the model. For example, after building a model for predicting the DNA-protein binding affinity, we may be curious about which motifs on DNA contribute more to the binding affinity landscape [25]. Training a deep learning model to perform disease diagnosis, we not only require the prediction and diagnosis results but also want to know how the model makes the decisions and based on which evidences, which can increase our confidence on the model's predictions [19]. To achieve that, we can assign example-specific importance score for each part of a specific example. In that direction, people can use perturbation-based approaches or backpropagation-based methods [18]. Perturbation-based approaches [3, 198, 159, 128] change a part of the input and observe its impact on the model's output. Although this idea is easy to understand, it has high computational complexity. Backpropagation-based methods [131, 140] propagate backward the signal from the output layer to the input layer for checking the importance score of different parts of the input. This kind of methods is proved to be useful and under active development [150]. Model interpretability can have various meanings in different scenarios. For more comprehensive discussion of deep learning interpretability and all the methods that have been used in bioinformatics, one can refer to [92] and Section 5.3 in [18].

4.5 Uncertainty scaling

When using machine learning methods to perform prediction, we usually do not just want a final predicted label but also want a confidence score for each query from the model, showing how confident the model is sure about the prediction [118]. That is an important property, no matter in which application scenarios, because the confidence score can prevent us from believing misleading and unreliable predictions. In biology, it can save us time and resources wasted on validating the misleading prediction results. The uncertainty scaling is even more important in healthcare, which can help us evaluate the reliability of machine learning-based disease diagnosis and an automatic clinical decision [79]. The probability score from the direct deep learning Softmax output is usually not in the right scale, since deep learning models can output overconfident prediction [114]. To obtain reliable probability scores, we need to perform post-scaling to the Softmax output. Several methods have been proposed to output the probability score in the right scale, such as the legendary Platt scaling [118], histogram binning [189], isotonic regression [190], Bayssian Binning into Quantiles (BBQ) [107]. Recently, temperature scaling was proposed for the deep learning methods specifically, which was shown to be much better than the other methods [47].

4.6 Catastrophic forgetting

A plain deep learning model is usually unable to incorporate new knowledge without interfering the learned knowledge, which is known as catastrophic forgetting [70]. For example, after we train a model which can classify 1,000 types of flowers, the 1,001st class of flowers come in. If we only fine-tune the model with the new data, the fine-tuned model's performance on the older classes will be unacceptable [89]. This scenario is actually very common in biology since the biological data are alway accumulating and updating. For example, the number of entries in PDB [11] has increased from 13,590 in 2000 to 147,595 in 2018. The size of Swiss-Prot [9] has also increased from around 100,000 in 2000 to 559,077 in 2018. With new data being generated, it is very likely that in the future, we will have new classes as shown in the Enzyme Commission (EC) number system [8]. Although training a completely new model from scratch using both new data and old data can be a straightforward solution, it is computationally intensive and time-consuming to do so, which can also make the learned representation of the original data unstable. Currently, there are three kinds of machine learning methods to deal with the problem based on the neurophysiological theories of human brain [192, 100], which is free of catastrophic forgetting. The first kind of methods is based on regularizations, such as EWC [70]. The second kind of methods uses the dynamic neural network architecture and rehearsal training methods, such as iCaRL [89, 124]. And the last kind of models is based on dual-memory learning systems [57]. More details can be referred to [111, 89]

bioRxiv preprint doi: https://doi.org/10.1101/563601; this version posted February 28, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

4.7 Reducing computational requirement and model compression

Because deep learning models are usually very complex and have lots of parameters to be trained, it is often computationally demanding and memory intensive to obtain well-trained models and even for the productive usage of the models [17]. Those requirements seriously limit the deployment of deep learning in machines with limited computational power, especially in the field of bioinformatics and healthcare, which is also data intensive. For example, as illustrated in [134], the bursting of DNA sequence data even wins over "Moore's Law", which requires distributed system or cloud computing to bridge the gap between the computational capability and the tremendous data size. As for the healthcare data, the multiple ways of evaluating people's health and the heterogeneous property of the data have made them more complex, with much larger size [29], which makes the problem even more computationally demanding [36]. To adopt deep learning methods into those bioinformatics problems which are computational and data intensive, in addition to the development of new hardware devoted to deep learning computing, such as GPUs and FPGAs [193], several methods have been proposed to compress the deep learning model, which can reduce the computational requirement of those models from the beginning. Those methods can be divided into four categories. The first type is parameter pruning, which reduces the redundant parameters that do not contribute to the model's performance significantly, including the famous DeepCompresion [51]. The second category is knowledge distillation [56], which trains a more compact model with distilled knowledge from the larger model. The third category is to use compact convolutional filters to save parameters [24]. And the last category uses low rank factorization [28] to estimate the informative parameters to preserve. Here we only show the most representative methods for model compression. More comprehensive discussions can be referred to [17].

5 Conclusion

Deep learning is a highly powerful and useful technique which has facilitated the development of various fields, including bioinformatics. With the advancement of big data era in biology, to further promote the usage of deep learning in bioinformatics, in this review, we first reviewed the achievements of deep learning. After that, we gave a brief and easy-to-understand introduction from shallow neural networks, to legendary convolutional neural networks, legendary recurrent neural networks, graph neural networks, generative adversarial neural networks and variational autoencoder. We also provided detailed examples with implementations to facilitate researchers in adopting and developing their own methods which are based on deep learning. Finally, we pointed out the common difficulties of using deep learning and provided corresponding suggestions. Although this review does not cover all the aspects of deep learning, such as deep reinforcement learning [106, 141, 105] and the theoretical aspect of deep learning [146, 194, 86, 174], it covers most aspects of the deep learning applications in bioinformatics. We believe this review will shed light on the future development and application of deep learning in bioinformatics, as well as biomedicine [164] and healthcare [36].

Acknowledgements

The research reported in this publication was supported by funding from King Abdullah University of Science and Technology (KAUST), under award number FCC/1/1976-18-01, FCC/1/1976-23-01, FCC/1/1976-25-01, FCC/1/1976-26-01, URF/1/3007-01-01, and URF/1/3450-01-01.

References

[1] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

[2] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[3] B. Alipanahi, A. Delong, M. T. Weirauch, and B. J. Frey. Predicting the sequence specificities of dna- and rna-binding proteins by deep learning. *Nat Biotechnol*, 33(8):831–8, 2015.

bioRxiv preprint doi: https://doi.org/10.1101/563601; this version posted February 28, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

[4] Jose Juan Almagro Armenteros, Casper Kaae Sønderby, Søren Kaae Sønderby, Henrik Nielsen, and Ole Winther. Deeploc: prediction of protein subcellular localization using deep learning. *Bioinformatics*, 33(21):3387–3395, 2017.

[5] C. Angermueller, T. Parnamaa, L. Parts, and O. Stegle. Deep learning for computational biology. *Mol Syst Biol*, 12(7):878, 2016.

[6] Christof Angermueller, Heather J Lee, Wolf Reik, and Oliver Stegle. Deepcpg: accurate prediction of single-cell dna methylation states using deep learning. *Genome biology*, 18(1):67, 2017.

[7] Junghwan Baek, Byunghan Lee, Sunyoung Kwon, and Sungroh Yoon. lncrnanet: Long non-coding rna identification using deep learning. *Bioinformatics*, 1:9, 2018.

[8] Amos Bairoch. The enzyme database in 2000. *Nucleic acids research*, 28(1):304–305, 2000.

[9] Amos Bairoch and Rolf Apweiler. The swiss-prot protein sequence database and its supplement trembl in 2000. *Nucleic acids research*, 28(1):45–48, 2000.

[10] P. Baldi, P. Sadowski, and D. Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nat Commun*, 5(1):4308, 2014.

[11] Helen M Berman, John Westbrook, Zukang Feng, Gary Gilliland, Talapady N Bhat, Helge Weissig, Ilya N Shindyalov, and Philip E Bourne. The protein data bank, 1999–. In *International Tables for Crystallography Volume F: Crystallography of biological macromolecules*, pages 675–684. Springer, 2006.

[12] Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106:249–259, 2018.

[13] Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*, 2015.

[14] Yifei Chen, Yi Li, Rajiv Narayan, Aravind Subramanian, and Xiaohui Xie. Gene expression inference with deep learning. *Bioinformatics*, 32(12):1832–1839, 2016.

[15] Yuhua Chen, Yibin Xie, Zhengwei Zhou, Feng Shi, Anthony G Christodoulou, and Debiao Li. Brain mri super resolution using 3d deep densely connected neural networks. In *Biomedical Imaging (ISBI 2018), 2018 IEEE 15th International Symposium on*, pages 739–742. IEEE, 2018.

[16] Yunpeng Chen, Jianan Li, Huaxin Xiao, Xiaojie Jin, Shuicheng Yan, and Jiashi Feng. Dual path networks. *arXiv*, 2017.

[17] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*, 2017.

[18] Travers Ching, Daniel S Himmelstein, Brett K Beaulieu-Jones, Alexandr A Kalinin, Brian T Do, Gregory P Way, Enrico Ferrero, Paul-Michael Agapow, Michael Zietz, Michael M Hoffman, et al. Opportunities and obstacles for deep learning in biology and medicine. *Journal of The Royal Society Interface*, 15(141):20170387, 2018.

[19] Edward Choi, Mohammad Taha Bahadori, Jimeng Sun, Joshua Kulas, Andy Schuetz, and Walter Stewart. Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. In *Advances in Neural Information Processing Systems*, pages 3504–3512, 2016.

[20] K. C. Chou. Pseudo amino acid composition and its applications in bioinformatics, proteomics and system biology. *Current Proteomics*, 6(4):262–274, 2009.

[21] Szcgcdy Christian, Liu Wei, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Erhan Dumitru, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[22] Eric M Christiansen, Samuel J Yang, D Michael Ando, Ashkan Javaherian, Gaia Skibinski, Scott Lipnick, Elliot Mount, Alison O'Neil, Kevan Shah, Alicia K Lee, et al. In silico labeling: Predicting fluorescent labels in unlabeled images. *Cell*, 173(3):792–803, 2018.

[23] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[24] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999, 2016.

[25] H. Dai, R. Umarov, H. Kuwahara, Y. Li, L. Song, and X. Gao. Sequence2vec: a novel embedding approach for modeling transcription factor binding affinity landscape. *Bioinformatics*, 33(22):3575–3583, 2017.

[26] Payel Das, Mark Moll, Hernan Stamati, Lydia E Kavraki, and Cecilia Clementi. Low-dimensional, free-energy landscapes of protein-folding reactions by nonlinear dimensionality reduction. *Proceedings of the National Academy of Sciences*, 103(26):9885–9890, 2006.

bioRxiv preprint doi: https://doi.org/10.1101/563601; this version posted February 28, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

[27] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[28] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in neural information processing systems*, pages 1269–1277, 2014.

[29] Ivo D Dinov. Volume and value of big healthcare data. *Journal of medical statistics and informatics*, 4, 2016.

[30] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[31] Hao Dong, Akara Supratak, Luo Mai, Fangde Liu, Axel Oehmichen, Simiao Yu, and Yike Guo. Tensorlayer: a versatile library for efficient deep learning development. In *Proceedings of the 2017 ACM on Multimedia Conference*, pages 1201–1204. ACM, 2017.

[32] Qiaonan Duan, St Patrick Reid, Neil R Clark, Zichen Wang, Nicolas F Fernandez, Andrew D Rouillard, Ben Readhead, Sarah R Tritsch, Rachel Hodos, Marc Hafner, et al. L1000cds 2: Lincs l1000 characteristic direction signatures search engine. *NPJ systems biology and applications*, 2:16015, 2016.

[33] Francis Dutil, Joseph Paul Cohen, Martin Weiss, Georgy Derevyanko, and Yoshua Bengio. Towards gene expression convolutions using gene interaction graphs. *arXiv preprint arXiv:1806.06975*, 2018.

[34] S. R. Eddy. Accelerated profile hmm searches. *PLoS Comput Biol*, 7(10):e1002195, 2011.

[35] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115–118, 2017.

[36] Andre Esteva, Alexandre Robicquet, Bharath Ramsundar, Volodymyr Kuleshov, Mark DePristo, Katherine Chou, Claire Cui, Greg Corrado, Sebastian Thrun, and Jeff Dean. A guide to deep learning in healthcare. *Nature medicine*, 25(1):24, 2019.

[37] R. D. Finn, P. Coggill, R. Y. Eberhardt, S. R. Eddy, J. Mistry, A. L. Mitchell, S. C. Potter, M. Punta, M. Qureshi, A. Sangrador-Vegas, G. A. Salazar, J. Tate, and A. Bateman. The pfam protein families database: towards a more sustainable future. *Nucleic Acids Res*, 44(D1):D279–85, 2016.

[38] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. Protein interface prediction using graph convolutional networks. In *Advances in Neural Information Processing Systems*, pages 6530–6539, 2017.

[39] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.

[40] Tina Glisovic, Jennifer L Bachorik, Jeongsik Yong, and Gideon Dreyfuss. Rna-binding proteins and post-transcriptional gene regulation. *FEBS letters*, 582(14):1977–1986, 2008.

[41] William J Godinez, Imtiaz Hossain, Stanley E Lazic, John W Davies, and Xian Zhang. A multi-scale convolutional neural network for phenotyping high-content cellular images. *Bioinformatics*, 33(13):2010–2019, 2017.

[42] Garrett B Goh, Nathan O Hodas, and Abhinav Vishnu. Deep learning for computational chemistry. *Journal of computational chemistry*, 38(16):1291–1307, 2017.

[43] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[44] Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. Hybrid speech recognition with deep bidirectional lstm. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 273–278. IEEE, 2013.

[45] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM, 2016.

[46] Kay Grünewald, Prashant Desai, Dennis C Winkler, J Bernard Heymann, David M Belnap, Wolfgang Baumeister, and Alasdair C Steven. Three-dimensional structure of herpes simplex virus from cryo-electron tomography. *Science*, 302(5649):1396–1398, 2003.

[47] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. *arXiv preprint arXiv:1706.04599*, 2017.

[48] William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.

[49] Jing-Dong J Han, Nicolas Bertin, Tong Hao, Debra S Goldberg, Gabriel F Berriz, Lan V Zhang, Denis Dupuy, Albertha JM Walhout, Michael E Cusick, Frederick P Roth, et al. Evidence for dynamically organized modularity in the yeast protein–protein interaction network. *Nature*, 430(6995):88, 2004.

[50] Renmin Han, Xiaohua Wan, Lun Li, Albert Lawrence, Peng Yang, Yu Li, Sheng Wang, Fei Sun, Zhiyong Liu, Xin Gao, et al. Autom-dualx: a toolkit for fully automatic fiducial marker-based alignment of dual-axis tilt series with simultaneous reconstruction. *Bioinformatics*, 35(2):319–328, 2018.

[51] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

bioRxiv preprint doi: https://doi.org/10.1101/563601; this version posted February 28, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

[52] K. M. He, X. Y. Zhang, S. Q. Ren, and J. Sun. Deep residual learning for image recognition. *2016 Ieee Conference on Computer Vision and Pattern Recognition (Cvpr)*, pages 770–778, 2016.

[53] K. M. He, X. Y. Zhang, S. Q. Ren, and J. Sun. Identity mappings in deep residual networks. *Computer Vision - Eccv 2016, Pt Iv*, 9908:630–645, 2016.

[54] R. Heffernan, K. Paliwal, J. Lyons, A. Dehzangi, A. Sharma, J. Wang, A. Sattar, Y. Yang, and Y. Zhou. Improving prediction of secondary structure, local backbone angles, and solvent accessible surface area of proteins by iterative deep learning. *Sci Rep*, 5:11476, 2015.

[55] G. Hinton, L. Deng, D. Yu, and G. E. Dahl. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal . . .* , 2012.

[56] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv*, 2015.

[57] Geoffrey E. Hinton and David C. Plaut. Using fast weights to deblur old memories. *Proceedings of the 9th Annual Conference of the Cognitive Science Society*, pages 177–186, 1987.

[58] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. *arXiv*, 2017.

[59] B. Huang, M. Bates, and X. Zhuang. Super-resolution fluorescence microscopy. *Annu Rev Biochem*, 78(1):993–1016, 2009.

[60] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. *arXiv*, 2016.

[61] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv*, 2015.

[62] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017.

[63] Takashi Ito, Tomoko Chiba, Ritsuko Ozawa, Mikio Yoshida, Masahira Hattori, and Yoshiyuki Sakaki. A comprehensive two-hybrid analysis to explore the yeast protein interactome. *Proceedings of the National Academy of Sciences*, 98(8):4569–4574, 2001.

[64] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pages 694–711. Springer, 2016.

[65] Vanessa Isabell Jurtz, Alexander Rosenberg Johansen, Morten Nielsen, Jose Juan Almagro Armenteros, Henrik Nielsen, Casper Kaae Sønderby, Ole Winther, and Søren Kaae Sønderby. An introduction to deep learning on biological sequence data: examples and solutions. *Bioinformatics*, 33(22):3685–3690, 2017.

[66] Daniel S Kermany, Michael Goldbaum, Wenjia Cai, Carolina CS Valentim, Huiying Liang, Sally L Baxter, Alex McKeown, Ge Yang, Xiaokang Wu, Fangbing Yan, et al. Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell*, 172(5):1122–1131, 2018.

[67] Ji-Sung Kim, Xin Gao, and Andrey Rzhetsky. Riddle: Race and ethnicity imputation from disease history with deep learning. *PLoS computational biology*, 14(4):e1006106, 2018.

[68] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv*, 2014.

[69] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[70] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell. Overcoming catastrophic forgetting in neural networks. *Proc Natl Acad Sci U S A*, 114(13):3521–3526, 2017.

[71] Vasiliki Kordopati, Adil Salhi, Rozaimi Razali, Aleksandar Radovanovic, Faroug Tifratene, Mahmut Uludag, Yu Li, Ameerah Bokhari, Ahdab AlSaieedi, Arwa Bin Raies, et al. Des-mutation: System for exploring links of mutations and diseases. *Scientific reports*, 8(1):13359, 2018.

[72] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the Acm*, 60(6):84–90, 2017.

[73] Anders Krogh and John A Hertz. A simple weight decay can improve generalization. In *Advances in neural information processing systems*, pages 950–957, 1992.

[74] Jan Kukačka, Vladimir Golkov, and Daniel Cremers. Regularization for deep learning: A taxonomy. *arXiv preprint arXiv:1710.10686*, 2017.

[75] Maxat Kulmanov, Mohammed Asif Khan, and Robert Hoehndorf. Deepgo: predicting protein functions from sequence and interactions using a deep ontology-aware classifier. *Bioinformatics*, 34(4):660–668, 2017.

[76] Devinder Kumar, Alexander Wong, and David A Clausi. Lung nodule classification using deep features in ct images. In *Computer and Robot Vision (CRV), 2015 12th Conference on*, pages 133–138. IEEE, 2015.

bioRxiv preprint doi: https://doi.org/10.1101/563601; this version posted February 28, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

[77] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–44, 2015.

[78] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. *arXiv*, 2016.

[79] Christian Leibig, Vaneeda Allken, Murat Seçkin Ayhan, Philipp Berens, and Siegfried Wahl. Leveraging uncertainty information from deep neural networks for disease detection. *Scientific reports*, 7(1):17816, 2017.

[80] M. K. Leung, H. Y. Xiong, L. J. Lee, and B. J. Frey. Deep learning of the tissue-regulated splicing code. *Bioinformatics*, 30(12):i121–9, 2014.

[81] Michael Ka Kit Leung, Andrew Delong, and Brendan J Frey. Inference of the human polyadenylation code. *bioRxiv*, page 130591, 2017.

[82] Michael KK Leung, Andrew Delong, Babak Alipanahi, and Brendan J Frey. Machine learning in genomic medicine: a review of computational problems and data sets. *Proceedings of the IEEE*, 104(1):176–197, 2016.

[83] Jun-Hao Li, Shun Liu, Hui Zhou, Liang-Hu Qu, and Jian-Hua Yang. starbase v2. 0: decoding mirna-cerna, mirna-ncrna and protein–rna interaction networks from large-scale clip-seq data. *Nucleic acids research*, 42(D1):D92–D97, 2013.

[84] Rongjian Li, Wenlu Zhang, Heung-Il Suk, Li Wang, Jiang Li, Dinggang Shen, and Shuiwang Ji. Deep learning based imaging data completion for improved brain disease diagnosis. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 305–312. Springer, 2014.

[85] Y. Li, S. Wang, R. Umarov, B. Xie, M. Fan, L. Li, and X. Gao. Deepre: sequence-based enzyme ec number prediction by deep learning. *Bioinformatics*, 34(5):760–769, 2018.

[86] Yu Li, Lizhong Ding, and Xin Gao. On the decision boundary of deep neural networks. *arXiv preprint arXiv:1808.05385*, 2018.

[87] Yu Li, Renmin Han, Chongwei Bi, Mo Li, Sheng Wang, and Xin Gao. Deepsimulator: a deep simulator for nanopore sequencing. *Bioinformatics*, 34(17):2899–2908, 2018.

[88] Yu Li, Hiroyuki Kuwahara, Peng Yang, Le Song, and Xin Gao. Pgcn: Disease gene prioritization by disease and gene embedding through graph convolutional neural networks. *bioRxiv*, page 532226, 2019.

[89] Yu Li, Zhongxiao Li, Lizhong Ding, Peng Yang, Yuhui Hu, Wei Chen, and Xin Gao. Supportnet: solving catastrophic forgetting in class incremental learning with support data. *arXiv preprint arXiv:1806.02942*, 2018.

[90] Yu Li, Fan Xu, Fa Zhang, Pingyong Xu, Mingshu Zhang, Ming Fan, Lihua Li, Xin Gao, and Renmin Han. Dlbi: deep learning guided bayesian inference for structure reconstruction of super-resolution fluorescence microscopy. *Bioinformatics*, 34(13):i284–i294, 2018.

[91] Yu Li, Tao Zhang, Shuyu Sun, and Xin Gao. Accelerating flash calculation through deep learning methods. *arXiv preprint arXiv:1809.07311*, 2018.

[92] Zachary C. Lipton. The mythos of model interpretability. *arXiv*, 2016.

[93] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen AWM van der Laak, Bram Van Ginneken, and Clara I Sánchez. A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88, 2017.

[94] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*, 2016.

[95] Jianzhu Ma, Michael Ku Yu, Samson Fong, Keiichiro Ono, Eric Sage, Barry Demchak, Roded Sharan, and Trey Ideker. Using deep learning to model the hierarchical structure and function of a cell. *Nature methods*, 15(4):290, 2018.

[96] Laurens Maaten, Minmin Chen, Stephen Tyree, and Kilian Weinberger. Learning with marginalized corrupted features. In *International Conference on Machine Learning*, pages 410–418, 2013.

[97] Itzik Malkiel, Achiya Nagler, Michael Mrejen, Uri Arieli, Lior Wolf, and Haim Suchowski. Deep learning for design and retrieval of nano-photonic structures. *arXiv preprint arXiv:1702.07949*, 2017.

[98] Polina Mamoshina, Armando Vieira, Evgeny Putin, and Alex Zhavoronkov. Applications of deep learning in biomedicine. *Molecular pharmaceutics*, 13(5):1445–1454, 2016.

[99] V. Marx. Biology: The big challenges of big data. *Nature*, 498(7453):255–60, 2013.

[100] J. L. Mcclelland, B. L. Mcnaughton, and R. C. Oreilly. Why there are complementary learning-systems in the hippocampus and neocortex - insights from the successes and failures of connectionist models of learning and memory. *Psychological Review*, 102(3):419–457, 1995.

[101] Tim R Mercer, Marcel E Dinger, and John S Mattick. Long non-coding rnas: insights into functions. *Nature reviews genetics*, 10(3):155, 2009.

bioRxiv preprint doi: https://doi.org/10.1101/563601; this version posted February 28, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

[102] Alan Merk, Alberto Bartesaghi, Soojay Banerjee, Veronica Falconieri, Prashant Rao, Mindy I Davis, Rajan Pragani, Matthew B Boxer, Lesley A Earl, Jacqueline LS Milne, et al. Breaking cryo-em resolution barriers to facilitate drug discovery. *Cell*, 165(7):1698–1707, 2016.

[103] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[104] S. Min, B. Lee, and S. Yoon. Deep learning in bioinformatics. *Brief Bioinform*, 18(5):851–869, 2017.

[105] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[106] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[107] Mahdi Pakdaman Naeini, Gregory F Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *AAAI*, pages 2901–2907, 2015.

[108] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. pages 807–814, 2010.

[109] Sinno Jialin Pan, Qiang Yang, et al. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[110] Xiaoyong Pan and Hong-Bin Shen. Predicting rna-protein binding sites and motifs through combining local and global deep convolutional neural networks. *Bioinformatics*, 2018.

[111] German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *arXiv*, 2018.

[112] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[113] Len A Pennacchio, Nadav Ahituv, Alan M Moses, Shyam Prabhakar, Marcelo A Nobrega, Malak Shoukry, Simon Minovitsky, Inna Dubchak, Amy Holt, Keith D Lewis, et al. In vivo enhancer analysis of human conserved non-coding sequences. *Nature*, 444(7118):499, 2006.

[114] Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*, 2017.

[115] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017.

[116] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

[117] Emma Pierson and Christopher Yau. Zifa: Dimensionality reduction for zero-inflated single-cell gene expression analysis. *Genome biology*, 16(1):241, 2015.

[118] John Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10, 1999.

[119] Mariya Popova, Olexandr Isayev, and Alexander Tropsha. Deep reinforcement learning for de novo drug design. *Science advances*, 4(7):eaap7885, 2018.

[120] Daniel Quang and Xiaohui Xie. Danq: a hybrid convolutional and recurrent deep neural network for quantifying the function of dna sequences. *Nucleic acids research*, 44(11):e107–e107, 2016.

[121] Alvin Rajkomar, Eyal Oren, Kai Chen, Andrew M Dai, Nissan Hajaj, Michaela Hardt, Peter J Liu, Xiaobing Liu, Jake Marcus, Mimi Sun, et al. Scalable and accurate deep learning with electronic health records. *npj Digital Medicine*, 1(1):18, 2018.

[122] Ladislav Rampasek, Daniel Hidru, Petr Smirnov, Benjamin Haibe-Kains, and Anna Goldenberg. Dr. vae: Drug response variational autoencoder. *arXiv preprint arXiv:1706.08203*, 2017.

[123] Debashish Ray, Hilal Kazan, Esther T Chan, Lourdes Pena Castillo, Sidharth Chaudhry, Shaheynoor Talukder, Benjamin J Blencowe, Quaid Morris, and Timothy R Hughes. Rapid and systematic analysis of the rna recognition specificities of rna-binding proteins. *Nature biotechnology*, 27(7):667, 2009.

[124] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. *arXiv*, 2016.

[125] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. *arXiv preprint*, 2017.

bioRxiv preprint doi: https://doi.org/10.1101/563601; this version posted February 28, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

[126] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (6):1137–1149, 2017.

[127] Sungmin Rhee, Seokjun Seo, and Sun Kim. Hybrid approach of relation network and localized graph convolutional filtering for breast cancer subtype classification. *arXiv preprint arXiv:1711.05859*, 2017.

[128] Marco Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. *arXiv*, pages 1135–1144, 2016.

[129] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

[130] Jean-François Rual, Kavitha Venkatesan, Tong Hao, Tomoko Hirozane-Kishikawa, Amélie Dricot, Ning Li, Gabriel F Berriz, Francis D Gibbons, Matija Dreze, Nono Ayivi-Guedehoussou, et al. Towards a proteome-scale map of the human protein–protein interaction network. *Nature*, 437(7062):1173, 2005.

[131] W. Samek, A. Binder, G. Montavon, S. Lapuschkin, and K. R. Muller. Evaluating the visualization of what a deep neural network has learned. *IEEE Trans Neural Netw Learn Syst*, 28(11):2660–2673, 2017.

[132] Benjamin Sanchez-Lengeling and Alán Aspuru-Guzik. Inverse molecular design using machine learning: Generative models for matter engineering. *Science*, 361(6400):360–365, 2018.

[133] Saman Sarraf and Ghassem Tofighi. Classification of alzheimer's disease using fmri data and deep learning convolutional neural networks. *arXiv preprint arXiv:1603.08631*, 2016.

[134] Michael C Schatz, Ben Langmead, and Steven L Salzberg. Cloud computing and the dna data race. *Nature biotechnology*, 28(7):691, 2010.

[135] Kristof T Schütt, Farhad Arbabzadah, Stefan Chmiela, Klaus R Müller, and Alexandre Tkatchenko. Quantum-chemical insights from deep tensor neural networks. *Nature communications*, 8:13890, 2017.

[136] Duncan E Scott, Andrew R Bayly, Chris Abell, and John Skidmore. Small molecules, big targets: drug discovery faces the protein–protein interaction challenge. *Nature Reviews Drug Discovery*, 15(8):533, 2016.

[137] K Seeliger, U Güçlü, L Ambrogioni, Y Güçlütürk, and MAJ van Gerven. Generative adversarial networks for reconstructing natural images from brain activity. *NeuroImage*, 181:775–785, 2018.

[138] Frank Seide and Amit Agarwal. Cntk: Microsoft's open-source deep-learning toolkit. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2135–2135. ACM, 2016.

[139] Mingfu Shao, Jianzhu Ma, and Sheng Wang. Deepbound: accurate identification of transcript boundaries via deep convolutional neural fields. *Bioinformatics*, 33(14):i267–i273, 2017.

[140] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. *arXiv preprint arXiv:1704.02685*, 2017.

[141] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[142] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv*, 2014.

[143] J. T. Simpson, R. E. Workman, P. C. Zuzarte, M. David, L. J. Dursi, and W. Timp. Detecting dna cytosine methylation using nanopore sequencing. *Nature Methods*, 14(4):407–+, 2017.

[144] Fatima Zohra Smaili, Robert Hoehndorf, and Xin Gao. Onto2Vec: joint vector-based representation of biological entities and their ontology-based annotations. *Bioinformatics*, 34(13):i52–i60, 06 2018.

[145] Fatima Zohra Smaili, Robert Hoehndorf, and Xin Gao. OPA2Vec: combining formal and informal content of biomedical ontologies to improve similarity-based prediction. 11 2018.

[146] Daniel Soudry, Elad Hoffer, Mor Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *arXiv*, 2017.

[147] Fabio A Spanhol, Luiz S Oliveira, Caroline Petitjean, and Laurent Heutte. A dataset for breast cancer histopathological image classification. *IEEE Transactions on Biomedical Engineering*, 63(7):1455–1462, 2016.

[148] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[149] I. Sturm, S. Lapuschkin, W. Samek, and K. R. Muller. Interpretable deep neural networks for single-trial eeg classification. *J Neurosci Methods*, 274:141–145, 2016.

[150] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. *arXiv preprint arXiv:1703.01365*, 2017.

bioRxiv preprint doi: https://doi.org/10.1101/563601; this version posted February 28, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

[151] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *arXiv*, 2014.

[152] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv*, 2016.

[153] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. In *International Conference on Artificial Neural Networks*, pages 270–279. Springer, 2018.

[154] Haotian Teng, Minh Duc Cao, Michael B Hall, Tania Duarte, Sheng Wang, and Lachlan JM Coin. Chiron: Translating nanopore raw signal directly into nucleotide sequence using deep learning. *GigaScience*, 7(5):giy037, 2018.

[155] Eric J Topol. High-performance medicine: the convergence of human and artificial intelligence. *Nature medicine*, 25(1):44, 2019.

[156] Ngoc Hieu Tran, Xianglilan Zhang, Lei Xin, Baozhen Shan, and Ming Li. De novo peptide sequencing by deep learning. *Proceedings of the National Academy of Sciences*, 114(31):8247–8252, 2017.

[157] Ramzan Umarov, Hiroyuki Kuwahara, Yu Li, Xin Gao, and Victor Solovyev. Promid: human promoter prediction by deep learning. *arXiv preprint arXiv:1810.01414*, 2018.

[158] Ramzan Umarov, Hiroyuki Kuwahara, Yu Li, Xin Gao, and Victor Solovyev. Promoter analysis and prediction in the human genome using sequence-based deep learning models. 01 2019.

[159] 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.

[160] David A Van Valen, Takamasa Kudo, Keara M Lane, Derek N Macklin, Nicolas T Quach, Mialy M DeFelice, Inbal Maayan, Yu Tanouchi, Euan A Ashley, and Markus W Covert. Deep learning automates the quantitative analysis of individual cells in live-cell imaging experiments. *PLoS computational biology*, 12(11):e1005177, 2016.

[161] 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.

[162] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv*, 2017.

[163] J Craig Venter, Mark D Adams, Eugene W Myers, Peter W Li, Richard J Mural, Granger G Sutton, Hamilton O Smith, Mark Yandell, Cheryl A Evans, Robert A Holt, et al. The sequence of the human genome. *science*, 291(5507):1304–1351, 2001.

[164] Michael Wainberg, Daniele Merico, Andrew Delong, and Brendan J Frey. Deep learning in biomedicine. *Nature Biotechnology*, 36(9):829, 2018.

[165] Fangping Wan, Lixiang Hong, An Xiao, Tao Jiang, and Jianyang Zeng. Neodti: neural integration of neighbor information from a heterogeneous network for discovering new drug–target interactions. *Bioinformatics*, page bty543, 2018.

[166] Meng Wang, Cheng Tai, Weinan E, and Liping Wei. Define: deep convolutional neural networks accurately quantify intensities of transcription factor-dna binding and facilitate evaluation of functional non-coding variants. *Nucleic acids research*, 46(11):e69–e69, 2018.

[167] S. Wang, J. Peng, J. Z. Ma, and J. B. Xu. Protein secondary structure prediction using deep convolutional neural fields. *Scientific Reports*, 6:18962, 2016.

[168] Sheng Wang, Shiyang Fei, Zongan Wang, Yu Li, Jinbo Xu, Feng Zhao, and Xin Gao. Predmp: a web server for de novo prediction and visualization of membrane proteins. *Bioinformatics*, 1:3, 2018.

[169] Sheng Wang, Zhen Li, Yizhou Yu, and Xin Gao. Wavenano: a signal-level nanopore base-caller via simultaneous prediction of nucleotide labels and move labels through bi-directional wavenets. *Quantitative Biology*, 6(4):359–368, 2018.

[170] Sheng Wang, Siqi Sun, Zhen Li, Renyu Zhang, and Jinbo Xu. Accurate de novo prediction of protein contact map by ultra-deep learning model. *PLoS computational biology*, 13(1):e1005324, 2017.

[171] Sheng Wang, Siqi Sun, and Jinbo Xu. Auc-maximized deep convolutional neural fields for sequence labeling. *arXiv preprint arXiv:1511.05265*, 2015.

[172] Gregory P Way and Casey S Greene. Extracting a biologically relevant latent space from cancer transcriptomes with variational autoencoders. *bioRxiv*, page 174474, 2017.

[173] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. corr abs/1410.3916, 2014.

[174] Lei Wu, Zhanxing Zhu, and E. Weinan. Towards understanding generalization of deep learning: Perspective of loss landscapes. *arXiv*, 2017.

[175] Zhihao Xia, Yu Li, Bin Zhang, Zhongxiao Li, Yuhui Hu, Wei Chen, and Xin Gao. Deerect-polya: a robust and generic deep learning method for pas identification. *Bioinformatics*, page bty991, 2018.

bioRxiv preprint doi: https://doi.org/10.1101/563601; this version posted February 28, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

[176] M. S. Xiao, B. Zhang, Y. S. Li, Q. Gao, W. Sun, and W. Chen. Global analysis of regulatory divergence in the evolution of mouse alternative polyadenylation. *Mol Syst Biol*, 12(12):890, 2016.

[177] B. Xie, B. R. Jankovic, V. B. Bajic, L. Song, and X. Gao. Poly(a) motif prediction using spectral latent features from human dna sequences. *Bioinformatics*, 29(13):i316–25, 2013.

[178] Junyuan Xie, Linli Xu, and Enhong Chen. Image denoising and inpainting with deep neural networks. pages 341–349, 2012.

[179] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *arXiv*, 2016.

[180] Dapeng Xiong, Jianyang Zeng, and Haipeng Gong. A deep learning framework for improving long-range residue–residue contact prediction using a hierarchical strategy. *Bioinformatics*, 33(17):2675–2683, 2017.

[181] Cheng Yang, Longshu Yang, Man Zhou, Haoling Xie, Chengjiu Zhang, May D Wang, and Huaiqiu Zhu. Lncadeep: An ab initio lncrna identification and functional annotation tool based on deep learning. *Bioinformatics*, 2018.

[182] Pengyi Yang, Zili Zhang, Bing B Zhou, and Albert Y Zomaya. Sample subset optimization for classifying imbalanced biological data. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 333–344. Springer, 2011.

[183] Qingsong Yang, Pingkun Yan, Yanbo Zhang, Hengyong Yu, Yongyi Shi, Xuanqin Mou, Mannudeep K Kalra, Yi Zhang, Ling Sun, and Ge Wang. Low dose ct image denoising using a generative adversarial network with wasserstein distance and perceptual loss. *IEEE transactions on medical imaging*, 2018.

[184] Yang Yang, Leng Han, Yuan Yuan, Jun Li, Nainan Hei, and Han Liang. Gene co-expression network analysis reveals common system-level properties of prognostic genes across cancer types. *Nature communications*, 5:3231, 2014.

[185] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[186] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Generative image inpainting with contextual attention. *arXiv preprint*, 2018.

[187] Zeping Yu and Gongshen Liu. Sliced recurrent neural networks. *arXiv preprint arXiv:1807.02291*, 2018.

[188] L. W. Yue, H. F. Shen, J. Li, Q. Q. Yuan, H. Y. Zhang, and L. P. Zhang. Image super-resolution: The techniques, applications, and future. *Signal Processing*, 128:389–408, 2016.

[189] Bianca Zadrozny and Charles Elkan. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *Icml*, volume 1, pages 609–616. Citeseer, 2001.

[190] Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 694–699, 2002.

[191] Haoyang Zeng, Matthew D Edwards, Ge Liu, and David K Gifford. Convolutional neural network architectures for predicting dna–protein binding. *Bioinformatics*, 32(12):i121–i127, 2016.

[192] F. Zenke, W. Gerstner, and S. Ganguli. The temporal paradox of hebbian learning and homeostatic plasticity. *Curr Opin Neurobiol*, 43:166–176, 2017.

[193] Chen Zhang, Peng Li, Guangyu Sun, Yijin Guan, Bingjun Xiao, and Jason Cong. Optimizing fpga-based accelerator design for deep convolutional neural networks. In *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 161–170. ACM, 2015.

[194] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

[195] Quan-shi Zhang and Song-Chun Zhu. Visual interpretability for deep learning: a survey. *Frontiers of Information Technology & Electronic Engineering*, 19(1):27–39, 2018.

[196] Sixin Zhang, Anna E Choromanska, and Yann LeCun. Deep learning with elastic averaging sgd. In *Advances in Neural Information Processing Systems*, pages 685–693, 2015.

[197] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.

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

[199] XueZhong Zhou, Jörg Menche, Albert-László Barabási, and Amitabh Sharma. Human symptoms–disease network. *Nature communications*, 5:4212, 2014.

[200] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networkss. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

bioRxiv preprint doi: https://doi.org/10.1101/563601; this version posted February 28, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

[201] Marinka Zitnik, Monica Agrawal, and Jure Leskovec. Modeling polypharmacy side effects with graph convolutional networks. *arXiv preprint arXiv:1802.00543*, 2018.

[202] Marinka Zitnik and Jure Leskovec. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*, 33(14):i190–i198, 2017.

[203] Nansu Zong, Hyeoneui Kim, Victoria Ngo, and Olivier Harismendy. Deep mining heterogeneous networks of biomedical linked data to predict novel drug–target associations. *Bioinformatics*, 33(15):2337–2344, 2017.

[204] Zhenzhen Zou, Shuye Tian, Xin Gao, and Yu Li. mldeepre: Multi-functional enzyme function prediction with hierarchical multi-label deep learning. *Frontiers in Genetics*, 9:714, 2018.