Deep learning techniques for biomedical data processing

Deep Learning Techniques for Biomedical Data Processing

Monica Bianchini and Giovanna Maria Dimitri

Dipartimento di Ingegneria dell'Informazione e Scienze Matematiche
Università degli Studi di Siena
Via Roma 56, I–53100, Siena, ITALY

Abstract

The interest in Deep Learning (DL) has seen an exponential growth in the last ten years, producing a significant increase in both theoretical and applicative studies. On the one hand, the versatility and the ability to tackle complex tasks have led to the rapid and widespread diffusion of DL technologies. On the other hand, the dizzying increase in the availability of biomedical data has made classical analyses, carried out by human experts, progressively more unlikely. Contextually, the need for efficient and reliable automatic tools to support clinicians, at least in the most demanding tasks, has become increasingly pressing. In this survey, we will introduce a broad overview of DL models and their applications to biomedical data processing, specifically to medical image analysis, sequence processing (RNA and proteins) and graph modeling of molecular data interactions. First, the fundamental key concepts of DL architectures will be introduced, with particular reference to neural networks for structured data, convolutional neural networks, generative adversarial models, and siamese architectures. Subsequently, their applicability for the analysis of different types of biomedical data will be shown, in areas ranging from diagnostics to the understanding of the characteristics underlying the process of transcription and translation of our genetic code, up to the discovery of new drugs. Finally, the prospects and future expectations of DL applications to biomedical data will be discussed.

1 Introduction

In recent years, deep learning (DL) techniques have achieved state–of–the–art performance in several different tasks, from image semantic segmentation [1, 2] to object detection [3, 4], from modelling traffic flows [5] to bioinformatics applications [6, 7, 8, 9]. Even though we understand the world through the interaction with the environment we observe, this empirical realism called *experience* is somewhat limiting. Despite being subject to the limitations of our senses, such a flat view of the world has been the driving force in many fields and was able

to lay the foundations of the former artificial intelligence systems. However, recent real–world challenges are changing this perspective, showing that the world around us, and the answers we are looking for, admit a non–Euclidean structure. More specifically, we could say that a critical research question is how the data are described. We could therefore define two main ways of representing data: through a symbolic or through a structural approach [10]. In the first case, the data are expressed through feature vectors, while, in the second, we use structured data, like sequences, trees or graphs, taking into account complex relationships existing between the basic information entities in a real scenario. Nevertheless, moving to structured representations means increasing the complexity of processing data and sometimes even not having a direct way to extend operations commonly performed in vector spaces. To give an example, while the concept of similarity between two vectors is well defined, and corresponds to calculate a distance within a metric space, this does not hold for graphs [10].

Indeed, with the accumulation of the so called non–Euclidean data [11], graphs have become extremely common and widely used. Graph structures can in fact represent biological networks at the molecular, protein, or species level; they can describe drug molecules, so as 3D protein structures or metabolic networks. On the other hand, digital images can be represented as pixel lattices, i.e. in the form of regular graphs. Even simpler data, though structured, are also very common in biological applications, such as sequences — for DNA and RNA data — and trees, for reconstructing molecular phylogenies. For what concerns DL techniques, these are becoming ubiquitous in chemistry, biology and medicine [12], including not only genome annotation and transcriptome analysis, but also predictions of protein binding sites, identification of major cancer transcription factors, predictions of metabolic functions in complex microbial communities, drug discovery and re–purposing, and precision medicine applications.

Among deep networks commonly used to process biomedical data, Recurrent Neural Networks (RNNs) were specifically tailored to process sequential data. In addition to feedforward connections, they are equipped with delayed connections, which make them able to process a sequence one element at a time — in the context of a protein sequence, for instance, one residue after another —, considering therefore their natural *flow*. In this way, memory arises and the neural network acquires the ability to store and integrate information from past inputs. Long–Short Term Memory (LSTM) networks are a special type of RNNs composed by memory cells, where context–dependent input, output, and forget gates are able to control what is the information processed and passed through at each stage [13]. Thus, LSTMs are capable of learning long–term dependencies, easily storing and exclusively transmitting selected inputs.

Graph Neural Networks (GNNs) [14], instead, are able to process input data encoded as general labeled graphs, and they can directly be applied, for instance, for molecule processing in the context of biomedical data processing. The state at each node is iteratively evaluated based on local information solely, realizing a sort of data diffusion across the whole graph. Moreover, GNNs are provided with

a supervised learning algorithm that, besides the standard input–output data fitting cost, incorporates a criterion aimed at enforcing a contractive dynamics, to ensure the convergence of the state computation. Both node–focused and graph–focused problems can be addressed by GNNs, meaning that an output is produced for each node or for a unique node of the graph. GNNs can also be applied to network–medicine, a brand new research field, which have brought to the definition of a highly interconnected and tight network of diseases that are interdependent and needs to be studied in a network perspective [15]. Finally, they can be used as an engine for graph generation, allowing to design new drug molecules, starting from a dataset of known compounds [16].

Besides, segmented images can be represented through region adjacency graphs, with nodes — labeled with vectors which describe visual features, such as texture, perimeter and area in pixels, etc. — describing homogeneous regions and arcs representing the adjacency relationship among regions. Therefore, they can be processed by GNNs, for example to perform object localization or detection (node–focused and graph–focused tasks, respectively). However, the most commonly used DL architectures for image processing are Convolutional Neural Networks (CNNs) [17, 18], due to their ability to integrate the feature selection process within the network training. Moreover, since hierarchical patch–based convolution operations are employed in CNNs, computational costs are reduced and images are abstracted on different feature levels. CNNs are also particularly effective in processing medical images, which however are often not enough to train a deep network. In such a case, data augmentation, namely synthetic image generation, is the only viable solution, which can be implemented using Generative Adversarial Networks (GANs) [19]. GANs use two neural networks competing one against the other, a generative model G and a discriminative model D, where G generates synthetic realistic data while D evaluates the authenticity of the data (if they belong to the training dataset or not). However, the use of GANs is not limited to synthetic image generation, and actually these models have been used, for instance, for creating human genomes [20].

Finally, all the above–described architectures can constitute the basis for the construction of a Siamese network. Siamese networks are composed by two or more *identical* sub–networks, where identical entails the fact that they have the same configuration with the same parameters and weights. Parameter updating is mirrored across sub–networks. They have been employed to implement a sort of similarity learning, since they are able to compare feature vectors describing two or more patterns, obtained with *ad hoc* networks (i.e., RNNs for sequences, GNNs for graphs, or CNNs for images). Siamese networks are particularly useful, for example, to search in medical image databases [21].

The rest of this survey is organized as follows. In Section 2, we present the commonly used types of structured data. In Section 3, we introduce the DL architectures used to address the various biomedical problems described in the following Section 4. Finally, Section 5 draws some conclusions and introduces open challenges for future work.

With the awareness that an exhaustive survey on the proposed topic would require the writing of numerous books, the arguments addressed in this paper are

particularly focused on the research activity of the Bio–SAILab of the University of Siena, of which both authors are members.

2 Structured Data

Structured data have a hybrid nature, both symbolic and sub–symbolic, and cannot be represented independently from the links between some basic constitutive entities. Symbolic information (collected in feature vectors) is used to label each base entity (a node). Instead, the sub–symbolic information is carried by edges between nodes, which represent relationships, symmetrical or not, such as inclusion, adjacency, presence of a chemical bond, etc. Edges can also be labeled, in order to characterize the relationship they describe. Both entities and their relationships can be homogeneous throughout the structure or not.

In a way that may seem counter–intuitive, we begin describing the most complex structured data, that is by giving the definition of a graph. In fact, the other types of structured data, which we will examine, all represent particular cases of graphs, just as the networks that are normally used to process them (recurrent and recursive networks) can be considered special cases of the GNN model. Images, as a type of complex aggregated data, will be described separately at the end of this section.

2.1 Graphs, trees, and sequences

A **graph** is defined as a pair $G = (V, E)$, being V the set of nodes and $E \subset V \times V$ the set of edges. Given two nodes $u, v \in V$, the edge connecting them is identified by (u, v). Graphs can be of two types: directed or undirected. In the first case, the edges are oriented — and they are commonly called arcs. An arc normally define a *causal* non–symmetric relationship between two nodes, such as inclusion. Instead, in the case of undirected graphs, the presence of an edge between two nodes stands for a *contextual* flow of information between them, typical in symmetric relationships. Graphically speaking, directed edges are usually depicted as arrows, while undirected ones are represented by segments. Moreover, attributes, in the form of feature vectors, can be added to both nodes and edges. For instance, in a protein–protein interaction network, nodes represent proteins while edges describe the presence of an interaction between two specific proteins. Protein features can be added to each node (type, location, etc.) as well as weights defining the strength of a connection can be attached to the edges. Attributes which are attached to each node or edge/arc are called labels. In general, we assume that a consistent labeling space is used for all of the nodes and edges constituting the graph. Therefore, a labeled graph can be defined as a quadruple $G_L = (V, E, \mathcal{L}_v, \mathcal{L}_e)$, with $\mathcal{L}_v : V \to L_v \in \mathbb{R}^q$ and $\mathcal{L}_e : E \to L_e \in \mathbb{R}^p$, being $\mathcal{L}_v$ and $\mathcal{L}_e$ the node and edge labeling functions, respectively. Moreover we will define the neighborhood of a node v as the set of the nodes directly connected to it or, more formally, $\text{ne}[v] = \{u \in V | (u, v) \in E \vee (v, u) \in E\}$. Graphs may also be heterogeneous, meaning

that different entities can be collected within a single graph, where relationships of different nature coexist [22].

Trees can be considered as a special type of directed graph, where each node has a unique parent, while there is no limit to the number of its descendants — we therefore call an m–ary tree the one that has m children at most at each node. Furthermore, trees are (commonly) rooted, i.e. there is a node from which all the other nodes can be reached, giving rise to a structure that easily represents a hierarchical arrangement of the basic entities that it collects. Finally, **sequences** are 1–ary trees, as the (labeled) entities follow one after the other.

2.2 Digital images

A digital image is a matrix of pixel values or, in other words, a regular grid of pixels. The size of the pixel is equal to the spatial resolution of the image and depends upon the instrument providing the data. Similarly, the number of elements in the vector describing each pixel is determined by the ability of the equipment to distinguish variations (in colors, texture, etc.). An image can therefore be considered as a regular graph (a lattice), where each pixel represents a node and each edge stands for a vicinity relation. DL techniques for image processing, however, do not take into account this structured interpretation of images, but instead work on pixel matrices, implementing specific layer to realize classical algebraic operations like upsampling or downsampling.

3 Deep Learning Architectures

In this section, we briefly introduce the neural network models used for the biological and biomedical applications described in Section 4. A particular attention will be devoted to Graph Neural Networks (GNNs), for which also the layered [23] and composite [24] versions will be sketched.

3.1 Recurrent neural networks

Network architectures able to process *plain* data, collected within arrays, are said to be static; they just define a *mapping* between the sets of input and output values. In fact, once the network has been trained, it computes a function between inputs and outputs, calculated according to the learning set, where the output at time t depends only on the input at the same time. This can be summarized just asserting that the network does not have *short–term* memory.

The simplest dynamic data type is a sequence, which represents one of the most natural ways to model temporal/sequential domains. In speech recognition, for instance, the words naturally flow to constitute a temporal sequence of acoustic features while, in molecular biology, proteins are organized in amino acid strings. The simplest dynamic architectures are recurrent networks, able to model temporal/sequential phenomena. Indeed, recurrent networks are able

to capture the temporal structure of the input and to produce a timeline output, thanks to neuron activations that can change even in presence of the same input pattern. Architectures composed by units having feedback connections, both between neurons belonging to the same layer or to different layers, show such a dynamic behaviour. More formally, a network is said to be recurrent if it contains some neurons whose activations depend directly or indirectly from their outputs. In other words, following the signal transmission through the network, cyclic paths exist that connect one or more neurons with itself/themselves.

A **Recurrent Neural Network** (RNN) processes a temporal sequence by using an internal state representation

$$\mathbf{x}(t) = f(l_t, \mathbf{x}(t-1)|\theta_f) \tag{1}$$

(a particular instance of the state update equation for GNNs, see Eq. (2)), that appropriately encodes all the past information injected into its inputs together with the input at time t, l_t. Using a multilayer perceptron (MLP), with weights collected into θ_f, as the basic block for the state updating, multiple types of recurrent networks may be defined, depending on which neurons are involved in the feedback. Recurrent networks can be trained to solve *node–focused* or *sequence–focused* problems, based on the presence of target values on all the nodes (at each time step), on some nodes (at some selected instants) or only at the end of the sequence. Anyway, the behaviour of a recurrent network (during a time sequence) can be reproduced by unfolding it in time, and obtaining the corresponding feedforward network, that contains as many copies of the original RNN as the length of the sequence, and can be trained based on a BackPropagation–like algorithm, called BackPropagation Through Time (BPTT). Indeed, BPTT forces the different copies of the weights in the unfolding network to remain the same over time (for congruence with the original RNN). Therefore, we can think the RNN as a feedforward architecture with shared weights and then train the unfolded network with weight constraints.

When training a deep neural network — as the unfolded network — with a gradient–based learning method like BackPropagation (BP), the partial derivatives are calculated by traversing the network from the final layer to the initial layer; using the chain rule, the deeper layers in the network go through continuous matrix multiplications to calculate their derivatives. If the derivatives are large, then the gradient will increase exponentially during BP, eventually exploding. Conversely, if the derivatives are small, then the gradient will decrease exponentially, possibly vanishing. In the case of exploding gradients, the accumulation of large derivatives results in the model being very unstable and incapable of effective learning, while the accumulation of small gradients results in a model that is incapable of extracting meaningful information from data, since the weights and biases of the initial layers, which tends to learn the core features from the inputs, will not be updated effectively. Anyway, long–term dependencies are difficult to be learned due to the very deep architecture they correspond to.

3.1.1 Long–short term memories

Long–Short Term Memories, LSTMs for brevity [13], are a variant of RNNs that introduce a number of special, internal gates, which help with the problem of learning relationships within both long and short sequences. Instead of the classical state transition function, calculated by an MLP, LSTMs consist of cells — where each cell is responsible for keeping track of the dependencies between the elements in the input sequence — that collect different gates, each dedicated to a specific function. In particular:

- the *input gate* controls the extent to which a new value flows into the cell;
- the *forget gate* controls the extent to which a value remains in the cell;
- the *output gate* controls the extent to which the value in the cell is used to compute the output of the specific LSTM unit.

In this way, LSTMs learn when it is necessary to retain a state or to forget it. They have many more internal parameters, which must be learned and constantly updated as new data arrives, which is their strength and weakness, as they are much more flexible than ordinary RNNs, but also much more expensive to be trained.

3.2 Graph neural networks

In this section, we will briefly describe the original GNN model, presented in [14]. **Graph Neural Networks** perform a local computation, which produces a state vector $\mathbf{x}_v \in \mathbb{R}^s,\ \forall\, v \in V$, that stores a hidden representation depending on v and its context in the graph. In fact, the state is computed for each node through a diffusion process based on the graph topology, which guarantees to encode the information related both to the graph structure and to the node labels. The encoding is performed by a learnable function that can be tuned to extract the relevant features suitable for the desired processing. The state variables $\mathbf{x}_v$ become additional labels attached to the graph nodes, and represent the current state of the computation. Thus, the GNN model can be thought of as a set of identical computational units that calculate a local state for each node, depending on the states of its neighbours and on the node label and connectivity. Formally, the state update equations can be expressed as

$$\mathbf{x}_v = f(\mathbf{l}_v, \mathbf{x}_{ne[v]}, \mathbf{l}_{ne[v]}|\theta_f) \tag{2}$$

being $\mathbf{l}_v$ the label of node v, and $\mathbf{x}_{ne[v]}$ and $\mathbf{l}_{ne[v]}$ the states and the labels of the node neighbours, respectively. The state transition function f can be implemented using a Multi–Layer Perceptron (MLP). The network will have an appropriate architecture to deal with a variable number of arguments (i.e. inputs), since the degree of each node v could ideally be different. Apart from this requirement, the number of outputs will be equal to the state space dimension s and the network architecture will be defined by the choice of the type and

number of neurons, the number of layers, etc., as for the classical feedforward networks. The parameter vector θ_f will collect all the neural network connection weights. It should be noted that the same function (i.e. the same weights) is exploited for each node in the input graph.

Given the state transition function and an input graph G, the result of the state computation is a vector assigned to each node v in G, so that the global computation performed on the input graph can be described by the following vectorial equation

$$\mathbf{x} = \mathbf{F}(\mathbf{x}, \mathbf{l}|\theta_f) \tag{3}$$

where $\mathbf{F}$ is the global transition function, whose entries are obtained by stacking the transition functions f for each node v, with an appropriate projection of the two vectors $\mathbf{x}$ and $\mathbf{l}$ to yield the variables related to the neighbourhood of v. The vector of the learnable parameters is still θ_f, since all the local instances of the transition function share the same weights.

Eq. (3) is a system of non–linear equations in the variables $\mathbf{x}$. In general, this equation can have multiple solutions, but we are interested only in those cases when the solution is unique. This requirement can be satisfied if the function $\mathbf{F}$ is a contraction map — i.e. it is Lipshitzian with Lipshitz constant $\mathcal{L} < 1$ — with respect to the state variables $\mathbf{x}$. In this case, by the Banach–Cacioppoli Fixed Point Theorem, Eq. (3) has a unique solution. The contractivity condition must be enforced during the learning process. The same theorem also provides the procedure for calculating the fixed point of Eq. (3). In fact, the solution can be achieved through an iterative state update process over time.

For each node v, an output vector $\mathbf{o}_v \in \mathbb{R}^o$ can also be computed given the state at convergence and the label of the node. The local output function g can be realized by a different MLP, parameterized by a set of learnable weights θ_g. Generally, this function yields an output vector for each node. Therefore, to implement a graph–focused function, the output can be considered only for a predefined node in the graph with specific properties, or alternatively can be averaged over all of the nodes, depending on the problem at end.

Since the state calculation proceeds for T steps, until convergence, we can proceed to unfold the GNN — that resembles the topology of the graph to be learnt — in time, obtaining an unfolding network, each layer of which corresponds to a time instant and contains a copy of all the units of the GNN; connections between layers depends on the GNN connectivity. With the unfolded network, the error backprogation follows a scheme that is obtained by combining Backpropagation Through Structure and Backpropagation Through Time [25, 26]. As well as recursive models (in terms of their particular scope), GNNs are (almost) universal approximators, i.e., they can approximate in probability, and up to any degree of precision, all the "practically useful" functions on the graph space. GNNs can be applied on different problems involving graphs, some of which are shown in Figure 1.

Composite Graph Neural Networks (CGNNs) are a particular type of GNNs that can process heterogeneous graphs [24]. Heterogeneous graphs are often used to represent information about different types of entities interacting

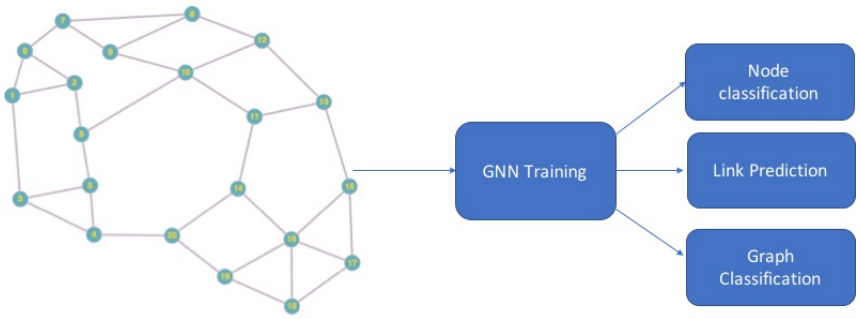

Figure 1: Different problems on graphs solvable by GNNs.

in multiple modes. Typical examples of this case are knowledge graphs, in which entities of different types, and often containing different features, need to be encoded into a single relational graph. Molecular graphs can also be seen as heterogeneous graphs, considering atom species as different node types. CGNNs label edges representing different types of relationships with the one–hot encoding of the relationship type. If edge features are present, these are concatenated to the edge labels. Node types instead are treated as subsets of the node set V, and each type has a dedicated state network. As a consequence, given the number of node types nv in the dataset, the model employs nv different state networks to build its encoding network. Each state network learns its own version f_i of the state updating function in Eq. (2). The output dimension of each state network is the same and corresponds to the state dimension s. To allow nodes to communicate through messages, which are coherent between different types, the node label is not part of the message, as it can assume different dimensions and meaning for different node types. Moreover, the output network g is unique and depends only on the state of the neuron at convergence.

The learning process described before still holds. The only difference, in the heterogeneous setting, consists in the fact that the GNN network, and consequently the unfolding network, are composed of nv state networks and one output network as building blocks. As a consequence, the learnable parameters are distributed in $nv+1$ MLPs, in contrast to the two MLPs of the homogeneous setting.

Computational issues have been reported in training dynamical networks to perform tasks in which spatio–temporal contingencies present in the input structures span long intervals. Indeed, as we have observed in the case of RNNs, gradient based learning algorithms face an increasingly difficult problem as the duration of the dependencies to be captured increases. In other words, there is a trade–off between efficient learning by gradient descent and latching of information for long "periods". In GNNs, the long–term dependency problem is observed when the output on a node depends on far nodes (i.e. neurons connected by long paths).

To solve this computational issue, both standard and composite GNNs can

be *cascaded*, obtaining a **Layered GNN** (LGNN) architecture [23, 24] such that, in each layer, the $(i-1)$–th GNN takes a graph in input with the same connectivity of the original input graph with node labels "enriched" by the information produced at the previous layer, for instance with the output of the i–th GNN, or the state of the i–th GNN, or both. Intuitively, each GNN solves the original problem, but it can make use of the expertise acquired in the previous layers. Therefore, LGNNs can incrementally incorporate the dependencies into the labels, since the output of a given node can collect information extracted from its neighborhood while, at each layer, the label contains information about a larger neighborhood. In order to not reintroduce long–term dependencies in the LGNN training, the training phase is carried out layer by layer, using always the same (original) target.

Finally, independently of the type of GNN — standard, composite or layered — both inductive and *transductive* learning can naturally be exploited [27]. In the inductive learning framework, a parametric model I_w is learnt by adjusting its weights, $w = (\theta_f, \theta_g)$, based on a training set. Then, the model can be applied to novel test patterns without further accessing the training set. With transductive learning instead, the training set patterns and their targets are used in conjunction with the test patterns. The decision on the test set is taken using a diffusion mechanism, e.g., exploiting the intuition that patterns with similar features are expected to be similar and belong to the same class.

3.3 Convolutional neural networks

Convolutional Neural Networks (CNNs) are basically feedforward neural networks in which neurons are locally connected to the neurons of the previous layer (in contrast with classical MLPs that are, instead, fully–connected). Furthermore, the weights of the connections between neurons are shared in groups, with different neurons of the same layer performing the same type of processing in different portions of the input, implying a strong reduction for what concerns the number of weights within the network.

CNNs are mainly used in computer vision for various types of image analysis tasks, like image segmentation, object detection and recognition, etc. They are made up of a hierarchy of levels, as depicted in Figure 2:

- an input layer, which acts as a buffer for the image pixels (or for any input information);
- some intermediate layers, which have local connections and shared weights, and are mainly a combination of convolutional and pooling layers;
- one or two terminal, fully–connected layers.

More in details: the input layer consists of a set of neurons responsible for passing the data representing the pixels of the input image. In the case of a colored image of $n \times n$ pixels, for instance, the input vector has a length of $n \times n \times 3$. Indeed, in this case, three values will be associated to the three colors

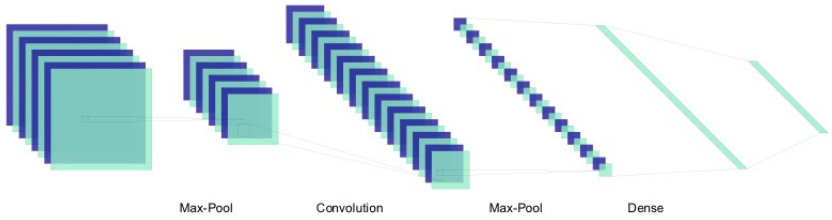

Figure 2: Example of a simple CNN architecture made of a series of convolutional and max–pooling layers, leading to the last dense layer.

in the RGB space. Obviously, the matrix depth will change based on the chosen color space. All neurons with the same depth in the 3D matrix form a *feature map*. As for the convolutional layers, they have the goal of identifying patterns within the image, such as edges, curves, circles and particular shapes. In general, several convolutional layers compose a CNN or, in other words, many filters are applied to images since the greater their number, the greater the complexity of the features that can be identified. In fact, in a convolutional layer, a digital filter is scrolled along the image and, for each position, an output value is computed evaluating the scalar product between the filter and the portion of the input image. Elementary filters are employed in the first layers, capable of identifying low–level features like borders, while gradually more sophisticated filters are used in the top layers, able to extract and classify very abstract characteristics within the image (for instance, able to recognize entire objects). Pooling layers are used to reduce the size of the feature maps. Therefore, the pooling operation also reduces the number of parameters to be learned and the amount of computation performed by the network. More specifically, we could say that they summarize the features present in a region of the feature map previously generated by a convolutional layer. Subsequently, further operations are performed on summarized features instead of precisely placed features. This makes the model more and more robust to variations in the position of objects in the input image. In the end, fully–connected layers are in fact those that eventually lead to the resolution of the image processing problem (segmentation, object/image classification, etc.) using data that comes as the result of the processing of all previous layers.

Local and shared connections imply that neurons process in the same way different portions of the image, producing a biological–like behavior, since different regions of the field of view contain the same type of information (edges, borders, portions of objects, etc.).

3.3.1 Generative adversarial networks

Generative Adversarial Networks (GANs) are a DL–based generative model [19, 28]. GAN architectures consists of two submodels:

- the *generator*, which is devoted to the generation of new plausible synthetic examples, based on the input data;
- the *discriminator*, which aims at detecting whether the examples received actually belong to the input domain (real data) or come from the output of the generator (fake data).

In other words, GANs are based on a game theory scenario in which the generator must compete against an opponent. The generator directly produces samples. Its opponent, the discriminator, tries to distinguish between samples taken from the training data and synthetically generated. In practice, both the generator and the discriminator are convolutional neural networks where the output of the former is directly connected to the input of the second. More specifically, the generator generates synthetic samples given a random noise (sampled from a latent space) while the discriminator is simply a binary classifier that discriminates between whether the input sample is real or fake. Through error backpropagation, the classification carried out by the discriminator provides the generator with the information necessary to update its weights. Once the generator training process is completed, the discriminator is discarded, since the GAN has learnt its task, i.e. has acquired the capability to generate realistic, synthetic data.

As a consequence of the fact that the GAN training actually corresponds to the training of two distinct models, it proceeds alternately: the generator is kept frozen during the training of the discriminator, so that the latter can understand what the defects of the generator are; conversely, during the generator training, it is the discriminator that is kept constant, otherwise it would practically try to hit a moving target and therefore would never converge. It is this *back and forth* procedure that, allowing the GAN to separate the training of its two components, guarantees to get a model capable of effectively addressing generative problems, otherwise intractable. However, as the generator improves performance, the performance of the discriminator deteriorates, since it is no longer able to correctly distinguish fake data from real ones and, in the case of a perfect generator, the discriminator would have a 50% accuracy (assuming that half of the examples come from the generator). This progression poses a problem for the convergence of the GAN training: the discriminator feedback becomes less significant over time, given that beyond a certain threshold it will decide intrinsically more and more random. In this case, if the generator training is not interrupted, it will receive *junk* feedback and the quality of the samples generated will get worse.

3.4 Siamese networks

Siamese neural networks allow to compare two (or more) input patterns, eventually assessing if they belong to the same category. Therefore, a Siamese network contains two (or more) identical sub–networks, where identical means that they have the exact same configuration with the same parameters and weights.

Even if Siamese networks can be used for any type of inputs [29], they are most commonly applied to image processing tasks, and they were shown to be particularly useful for image retrieval, verification, and few–shot learning [30, 31, 32, 33]. More precisely, let us consider a Siamese network able to compare pairs of input images. In this framework, the architecture of a Siamese network is constituted by a two–branch convolutional neural network, which is used on both the input images in order to extract their features, and by a distance function, which measures their similarity. In order to evaluate the similarity between two images I_1, I_2, a metric has to be defined in the embedding feature space, namely a parametric function S_w, realized by the CNNs contained in the Siamese. A common choice is the Euclidean distance, denoted as E_W in the following (to evidence its dependence from the network weights). However, the adoption of a contrastive loss function [34] was shown to be more effective to learn similarities.

The contrastive loss presumes the availability of a supervised similarity label Y for each pair of images I_1, I_2, whose value is 0, for similar images, and 1 otherwise. Formally, by re–writing $E_W(I_1, I_2)$ as E_W for brevity, the contrastive loss function L_C is:

$$L_C(W, Y, I_1, I_2) = (1 - Y)E_W^2 + Y \cdot max(0, m - E_W)^2$$

where $m > 0$ is a margin defined so that a pair contributes to the loss only if its distance E_W belongs to $(0, m)$. Intuitively, this loss makes the embeddings $S_W(I_1)$, $S_W(I_2)$ closer for similar inputs, and distant if they are different. Pairs of patterns that are close in terms of the embedding distance E_W produce a very low loss if the patterns are similar, whereas the loss is large if they are not similar. A Siamese network can be trained end–to–end using a common optimization method. Indeed, the learning procedure is similar to that of a standard CNN, with few peculiarities. The training set consists of pairs of images (query–reference), which have to be constructed according to some predefined criterion. The contrastive loss allows to compute, for each pair of the dataset, the gradient with respect to the Siamese network parameters, namely the parameters of the embedded CNNs. Finally, any common gradient–based optimization method can be applied.

4 DL Techniques in Biomedical Applications

In this section, we will present an overview of DL techniques used in various contexts of biomedical data analysis. In detail, in Section 4.1, we will show some

applications of recurrent neural networks and, particularly, LSTMs to sequence analysis. In Section 4.2, we will introduce GNN applications to biological data, with special reference to molecule analysis and drug discovery. Finally, in Section 4.3, we will draw some examples and briefly sketch applications of CNNs to image processing and biomedical data analysis for diagnostic purposes. A graphical summary of this section is proposed in Figure 3.

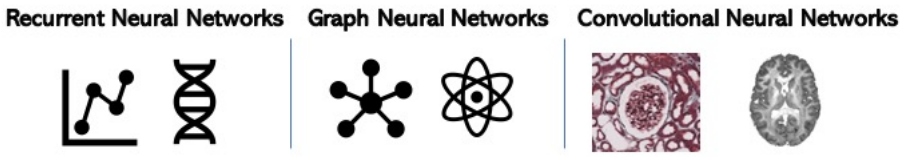

Figure 3: Structured data processing with deep neural networks.

4.1 Sequence processing via recurrent neural networks

The application of recurrent architectures has been proved to be particularly successful in many different biomedical fields, from heart–related ECG analysis [35] to brain signals, as for instance fMRI signals [36, 37].

In this section, we will provide several examples concerning the application of DL techniques for the analysis of sequences of various types: from ECG to protein and genomic data. Indeed, RNNs have been profitably employed for the analysis of cardiac pathologies, for instance, for the detection of cardiac arrhythmia [35], as well as for heart failure prediction in primary care patients [38]. Another example of the application of RNNs to the ECG time series analysis is reported in [39], where the Echo State Network (ESN) is employed for the automatic identification of patterns related to the Brugada Syndrome, which is a rare cardiac disease, whose diagnosis — through the analysis of the ECG — is particularly difficult. The ESN recurrent architecture can actually offer an efficient clinical tool for the early detection of such pathology and represents a valid support for cardiologists.

If, in the context of heart–related diseases, applications of RNNs have proved to be successful, the same can be said in a different biomedical context, i.e. for protein sequence analysis. In fact, several works are present in the literature on this topic [40, 41]. For example, in [42], a new deep learning framework is provided, denominated DeepPPISP, in which contextual and local features are combined for protein–protein interaction site predictions. Moreover, in [43], a comprehensive comparison of different machine learning models, including LSTMs, is realized for the prediction of biological signals characterizing the formation of α–helices. The analysis was carried out on a large experimentally collected dataset, and the results obtained showed the usefulness of the application of deep attention based mechanisms for the analysis of sequence data [44]. In fact, based on attention, it was proved that all different models focus on the same subsequences, which can be seen as codes driving the secondary

structure formation. Moreover, the analysis showed which amino acids are most important in determining the network output, demonstrating that the network can learn biological properties of the subsequences in order to calculate its predictions. Also, the heatmap of attention over the different sequence positions relative to the α–helix and the amino acid type is consistent with the expectations of the biomedical literature on the formation of such secondary structures [43].

Other interesting applications of RNNs to biological sequences come from their use for the analysis of protein sequences. For instance, in [45], a new computational environment is provided for modelling proteins and functions of non–coding DNA sequences. Similarly in [46], a siamese network, composed of a pair of identical (weight–sharing) LSTMs, was proposed to realize a new similarity score between protein sequences, able to resemble the BLAST score. In particular, this work focuses on the comparison between circulating common cold coronaviruses and SARS–CoV–2, and was able to prove how a preexisting immune memory due to exposure to common cold HCoVs has a significant impact on the COVID–19 disease severity, thus suggesting the fundamental role of the protein sequence similarities with different circulating coronaviruses to understand SARS–CoV–2 cross–reactivity [46]. In particular, it was found that the spike protein bring the largest cross–reactivity potential, as well as other proteins which figure on the surface proteins, proportionally to their importance in the immune response and memory. Also structural proteins bear a potential of cross–reactivity, yet this is limited and has therefore limited predictive power, as highlighted using an attention mechanism inside the LSTM. Moreover, the SARS–CoV–2 proteome had been also investigated from the point of view of protein–protein interactions in [47]. The focus of this latter work was finding a mechanism for disrupting the spike trimerization, therefore hampering the formation of the virus' most powerful weapon for penetrating human cells.

4.2 Graph processing via graph neural networks

Graphs emerge naturally in several biological contexts, from protein interaction networks to biomedical images, from metabolic networks to disease interactions. In other words, different important applications can be performed based on a graph–representation of data, where the topological information can be exploited under different analysis perspectives, namely for node, edge, or graph–focused classification or regression problems, for link prediction, in a generative framework, etc. (see Fig. 1). Therefore, the capabilities of GNNs in the biomedical field are huge and lead to a vast amount of applications in very different domains [48].

As a generative framework, GNNs can be used for drug discovery and repurposing. The design of new drugs is, in fact, a time consuming process, made up of several steps, from drug target determination to lead compound discovery and optimization and to pre–clinical assessment [11]. The process turns out to be extremely expensive and therefore the use of automatic methodologies for performing drug development becomes a crucial step. In particular, in recent

years, the use of machine learning models has become fundamental to speed up the process and help both in the design and the analysis of new proposed drugs. While several machine learning models have been devised for drug development and repurposing [49, 50, 51, 52], many research directions remain unexplored from the point of view of applying DL architectures to solve the relevant problems, which is particularly true for graph neural networks. This phenomenon is mainly due to the lack of sufficiently large datasets, as well as to the complexity of drug interaction data, which must be considered in this context.

Apart from the limitations due to the scarcity of appropriately labeled data, GNNs can provide a valid tool for the generation of molecules as well as to predict drug side effects. In this context, several studies can be found in the literature, such as [53], for *de novo* drug design, or [54], for molecular property explanation. In [16], a sequential graph generator for molecular structures is proposed, named MG^2N^2 (Molecular Generative Graph Neural Networks). Each node in the molecular graph corresponds to an atom, and each edge to a chemical bond. The MG^2N^2 algorithm is based on an iterative process in which, at each iteration, a node is added to the molecular structure. In order to do so, the model is composed of three GNN modules: the first generates new atom nodes, the second connects each new node to the atom it was generated as a nieghbour of, and the third module generates the (optional) edges connecting the new node to the rest of the graph. The training procedure of each module is independent from the others, a characteristic which guarantees a faster training, and an easier retraining in case one of the modules should be upgraded, also enhancing the model's interpretability, as suggested for this kind of model [55]. As hinted in the description of the second module, the molecular graphs are generated through an expansion process, in which the GNNs focus on one atom after the other, generating all the neighbors of that atom before moving to the following one. Atoms are expanded following a first–in–first–out queue, in which the first generated atoms are expanded first. This guarantees that the generated graph will not have disconnected components. Since the three modules are basically thought as classifiers, to avoid mode collapse and give a stochastic behaviour suitable for the generative purpose to the GNNs, each module is equipped with a Gumbel softmax output layer [56], instead than a regular softmax layer. The main advantage of MG^2N^2 with respect to other sequential generators based on RNNs, reinforcement learning, or GAN–like mechanisms, is represented by the smaller information loss thanks to the capability of GNNs of natively processing graphs, while the other methods need to simplify the data representation using SMILES and other types of vectorial or sequential representations of the graph. Moreover, generating the graph step by step allows to retain a more interpretable, error–aware, and easier to train mechanism with respect to SMILES–based VAE generators and even graph–based VAE generators, that generate all the molecules in one shot by sampling from their latent space [16].

Even drug side effects have a high impact on health system costs and drug discovery processes. Predicting their probability before their occurrence is fundamental to reduce this impact. Indeed, candidate molecules could be screened

before undergoing clinical trials, reducing the costs in terms of time, money, and health of the participants. Drug side effects are triggered by complex biological processes involving many different entities, from drug structures to protein–protein interactions. In [22], GNNs are applied for this task. Specifically, heterogeneous data sources were used and integrated in a unique graph, conveying information on drug structures and chemical features, drug–gene and gene–gene interactions, and drug–drug similarities. In this way, the relational context established among drugs and genes, on which they produce effects, together with the interactions existing among genes, are taken into account and exploited to predict drug side effects. The network makes use of this sort of knowledge graph to mine the information relevant for each prediction, creating a model of great usability that can predict drug side effects of newly submitted drugs without retraining and with a small amount of new information. The two models proposed in [16] and [22] could even be combined, with MG^2N^2 generating molecular graphs of possible drug candidates in large quantities, which can be subsequently screened to filter out all the compounds with high probabilities of occurrence of dangerous side effects or that simply produce too many side effects.

Similarly, further works in this scope were developed with the aim of studying drug–target interactions [57], as well as performing drug–drug interaction prediction [58].

Several other applications of GNNs in the context of biomedical data concern the analysis of metabolic and genomic networks [59, 60, 61] and the prediction of disease–disease and protein–protein interactions. For example, in [62], GNNs are employed for the prediction of Protein–Protein Interfaces (PPI). In particular, the study was focused on the binding site identification, allowing to determine the functionality and the quaternary structure of protein–protein complexes. Interacting peptides were represented as graphs, in which each secondary structure corresponds to a node and edges model the physico–chemical bonds between secondary structures. A *correspondence graph* can be built, describing their interaction, in which secondary structures that show correspondence are linked together. As it was proved in [63], finding the maximum clique in the correspondence graph allows to identify the secondary structure elements belonging to the interface site. Although the maximum clique problem is NP–complete, GNNs represent a soft–computing tool able to solve the problem in an affordable time. The GNN can be trained on a relatively small number of examples labeled with the Bron–Kerbosch algorithm [64], learning to replicate the algorithm solution in a fraction of the time employed by the traditional implementation. It can then be exploited to predict new interfaces in a short time and with high accuracy.

Also, the power of GNNs in community analysis was exploited to build the proof of concept of a mechanism to create a community of caregivers of rare disease patients. The implementation of a smartphone app to connect caregivers with each other would be beneficial to them as being a caregiver is often a challenge from many point of views, and sharing experiences and sensations could improve their capability of facing such challenges [65].

4.3 Medical image processing via convolutional networks

Recently, DL models have had a huge impact in computer vision applications: from image semantic segmentation to object detection, most of the computer vision tasks reached state–of–the–art performance with the use of Deep Learning. In this section, we will describe some applications of the models presented in Section 3, showing examples of standard CNNs, GANs and Siamese architectures applied to medical image processing.

A way to support clinicians in the decision–making process is to provide a system for retrieving cases that are similar to the examined one. In this way, doctors can compare cases and directly assess the similarity with past exams. The exploitation of this comparison is particularly useful to calibrate diagnoses and treatments, moving toward a precision medicine approach. Such a framework can be implemented by a Content–Based Image Retrieval system (CBIR), capable of retrieving the most similar images to a query one. In [66], a novel supervised Siamese–based architecture was proposed, which is able to treat multi–modal and multi–view MR images, and retrieve similar lesions in the case of prostate images. Similarly, in [67], a Siamese network was devised for prostate cancer classification.

CNNs were also exploited for eye–tracking during trail making tests aimed at diagnosing particular pathological conditions, like extrapyramidal syndromes and chronic pain [68].

A different task in which DL models have proved to reach state–of–the–art performance is image generation and, in particular, medical image processing has highly benefited from the use of DL architectures, as for tomographic image analysis [69] or PET image reconstruction [70]. The need for medical image generation directly comes from the commonly small dimension of image datasets, which contain not enough data to be used for training a deep network. Among the most notable CNN–based architectures for image generation we can find autoencoders [1] and GANs.

In [71], an autoencoder network was proposed for 3D brain MRI reconstruction. Here, the autoencoder is able to reproduce high–quality 3D images, as well as to retain meaningful information in the most hidden latent embedding dimension. In the context of brain MRI analysis, in fact, one of the main challenges is represented by the need of treating high–dimensional images. A way to address this significant computational burden is to consider slices of the 3D images to treat brain MRI scans. An example of this approach can be found in [72], where 2D slices of brain NMR scans were used to predict the patients' biological age.

GANs also proved to be suitable for medical image generation. Urinary tract infections (UTIs) are considered to be the most common bacterial infection and, actually, it is estimated that about 150 million UTIs occur yearly on a

[1] A variational autoencoder (VAE) is a type of deep network that learns to reproduce its input — actually, it represents a self–supervised model —, and also map data to a latent (hidden) space, in which the information is compressed based on a maximum–conservation principle.

world wide basis. To automatically analyze Petri plates coming from urine culture, in [73], a two stage computational workflow for image segmentation was presented. Indeed, the original dataset was augmented using a GAN, to be later segmented with the use of a CNN–based architecture. Finally, a standard MLP can be used to classify the type of infection and its severity [74]. Similarly, in [75], GANs were used to produce high–quality retinal images, together with the corresponding semantic label maps, to estimate the retinal vessel tortuosity. The two–step approach is based on a first phase in which a progressive growing GAN (PGGAN [76]) is trained to produce the semantic label maps. These maps, in fact, describe the blood vessel structures, and are needed to detect possible retinal or circulatory diseases. Subsequently, an image–to–image translation approach was used to obtain retinal images generated by the means of first sketching the vessel network. In this way, the generation process is simplified and requires less computational effort. Finally, in [77], GANs are employed in a multi–stage fashion, requiring a smaller amount of data, for multi–organ chest X–ray image generation. Analogously to the approach in [73], the generation process is followed by the organ segmentation step and then by the chest X–ray image reconstruction.

Even in the case of image segmentation, CNN–based architectures have proved to be particularly performing for medical applications such as, for example, in histopathologic image analysis (of kidneys, liver, lung, breast and other biological tissues [78, 79, 80]). In particular, in [81, 82], a DeepLab based architecture [83] for the construction of an automatic tool for kidney glomeruli segmentation was proposed. Such tool can provide an important support for clinicians to count the number of glomeruli present in a renal section and understand how many are sclerotised or not.

Different types of images — natural and coming from various instruments — can be subject to segmentation in medical image processing. For example, in [84], a CNN–based approach is used to segment aorta CT scans, in order to early discover alterations in its morphology which may portend an aortic dissection, i.e. the rupture of the innermost layer of the aorta which allows blood to flow between the layers of the aortic wall, forcing the layers to separate. On the other hand, in [85] a weakly supervised approach was implemented to realize skin lesion segmentation and identify possible harmful melanomas from benign nevi. This research topic was also further exploited, investigating a multi–modal approach, i.e. fusing anamnestic patient information with skin lesion images. Indeed, in [86], a DL tool for the early diagnosis of skin lesions was developed. The input data consisted in images and in demographic features of the patients (including age and gender), together with the position of the lesion. The DL classifier can efficiently discern between benign and malignant lesions, allowing clinicians to be supported in their diagnosis, possibly avoinding surgery. Finally, in [87], CNNs were used for segmenting oocyte images in order to support medical specialists in improving medical assisted procreation.

However, CNNs are also used for applications not strictly related to images, possibly adjusting the architecture to suit the particular type of data. As an example of a convolutional network application to RNA sequences, in [88], a 1–D

CNN was used for the analysis of ribosome profiling data (Ribo–seq). In particular high–reproducible Ribo–seq profiles were analysed, to produce a prediction on the translation speed associated with the sequence. This is possible because the Ribo–seq profile is a measurement of the quantity of fragments found inside ribosomes for each part of a sequence. Fragments which are frequently found inside ribosomes have been demonstrated to have slower translation (the ribosome spends more time on it) while fragments with low frequency have high translation speed. E.Coli Ribo–seq profiles were collected from various sources, though the consensus among these sources is high only on 40 of the various thousand genes of E.Coli. These 40 sequences were used as reference to train and test a neural network predictor of the translation speed of subsequences. Another example of cross–field application of CNNs is using them for analysing molecular graphs from the QM9 dataset: as it was recently proved in [89], in some particular cases, graphs of very small size can be translated to images and successfully processed with image–oriented CNNs. Indeed, GNNs are still recommended for the vast majority of graph–based tasks, as suggested by the fact that they are universal approximators on graphs [90].

5 Conclusions

Deep learning is one of the fastest growing fields of research and has had a significant impact on different types of bioinformatics applications. The analysis of biomedical data in fact poses a wide variety of problems that can be effectively addressed by building decision support systems, capable of providing substantial help to human experts, especially in the most routine tasks. We are conscious that, in this paper, we have offered only a partial view of the research field, mainly proposing a survey of the works carried out within the Bio–SAILab of the University of Siena, of which the two authors are members, but trying to show how our research is well supported by the general and widespread interest that both DL techniques and biomedical applications receive in the research community. Challenges and future perspective for this field are several. The availability of new data will support the development of new deep learning techniques and the improvement of existing ones. Furthermore, future methods cannot be separated from being trustworthy and explainable [91], a need that is very much felt in all DL applications, but which is particularly sensitive in the case of biomedical applications where a wrong choice, or simply not very reliable or understandable, can have a significant impact on the health of individuals.

Authors Biographies

Monica Bianchini

Prof. Monica Bianchini received her master's degree cum laude in Applied Mathematics in 1989 and her PhD in Computer Science and Control Systems in 1995 from the University of Florence, Italy.
She is currently Associate Professor at the Department of Information Engineering and Mathematics of the University of Siena. Her main research interests are in the field of machine learning, with emphasis on neural networks for structured data and deep learning, approximation theory, bioinformatics, and image processing.
Monica Bianchini has authored more than one hundred and thirty papers `https://scholar.google.it/citations?hl=it&user=cKBOwzwAAAAJ` and has been the editor of books and special issues on international journals in her research field. She has served/serves as Associate Editor for IEEE Transactions on Neural Networks, Neurocomputing, Int. J. of Knowledge-Based and Intelligent Engineering Systems, Int. J. of Computers in Healthcare, Frontiers in Genetics, and was editor of numerous books and special issues in international journals on neural networks, structural pattern recognition and bioinformatics.

Giovanna Maria Dimitri

Dr Giovanna Maria Dimitri holds a PhD in Computer Science, obtained at the University of Cambridge (UK), supervised by Prof. Pietro Liò, with

the dissertation: "Multilayer network methodologies for brain data analysis and modelling". She graduated in July 2015 in the MPhil in Advanced Computer Science at the University of Cambridge, with distinction. She is a life member of Clare Hall college, University of Cambridge. Previously she received her master and bachelor thesis (both 110/110 cum laude) in Computer and Automation Engineering at the University of Siena (Italy), supervised by Prof. Michelangelo Diligenti. She is teaching Business Intelligence for the master in Engineering Management (DIISM, University of Siena) since A.Y. 2019/2020. She is the associate editor for Neurocomputing. Her research interests mainly concerns developing deep learning and machine learning models for computer vision and bioinformatics. She is currently a post-doc researcher at the DIISM, University of Siena, Italy. Google Scholar Profile: `https://scholar.google.com/citations?user=2Ohb_AUAAAAJ&hl=it&oi=ao`.

References

[1] Bonechi S, Bianchini M, Scarselli F, Andreini P. Weak supervision for generating pixel—level annotations in scene text segmentation. Pattern Recognition Letters. 2020;138:1-7.

[2] Andreini P, Dimitri GM. Deep Semantic Segmentation Models in Computer Vision. In: ESANN 2022 proceedings European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, https://doi.org/10.14428/esann/2022.ES2022-5; 2022. p. 305-14.

[3] O'Mahony N, Campbell S, Carvalho A, Harapanahalli S, Hernandez GV, Krpalkova L, et al. Deep learning vs. traditional computer vision. In: Science and information conference. Springer; 2019. p. 128-44.

[4] Zhao ZQ, Zheng P, Xu St, Wu X. Object detection with deep learning: A review. IEEE Transactions on Neural Networks and Learning Systems. 2019;30(11):3212-32.

[5] Rossi A, Barlacchi G, Bianchini M, Lepri B. Modelling Taxi Drivers' Behaviour for the Next Destination Prediction. IEEE Transactions on Intelligent Transportation Systems. 2020;21(7):2980-9.

[6] Tsuruoka Y. Deep learning and natural language processing. Brain Nerve. 2019;71(1):45-55.

[7] Min S, Lee B, Yoon S. Deep learning in bioinformatics. Briefings in Bioinformatics. 2017;18(5):851-69.

[8] Dimitri GM, Spasov S, Duggento A, Passamonti L, Toschi N, et al. Unsupervised stratification in neuroimaging through deep latent embeddings. In: 2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC). IEEE; 2020. p. 1568-71.

[9] Maj C, Azevedo T, Giansanti V, Borisov O, Dimitri GM, Spasov S, et al. Integration of machine learning methods to dissect genetically imputed transcriptomic profiles in Alzheimer's disease. Frontiers in Genetics. 2019;10:726.

[10] Bianchini M, Dimitri GM, Maggini M, Scarselli F. Deep neural networks for structured data. In: Computational Intelligence for Pattern Recognition. Springer; 2018. p. 29-51.

[11] Zhang XM, Liang L, Liu L, Tang MJ. Graph neural networks and their current applications in bioinformatics. Frontiers in Genetics. 2021;12.

[12] Zitnik M, Nguyen F, Wang B, Leskovec J, Goldenberg A, Hoffman MM. Machine learning for integrating data in biology and medicine: principles, practice, and opportunities. Information Fusion. 2019;50:71-91.

[13] Hochreiter S, Schmidhuber J. Long–short term memory. Neural Computation. 1997;9(8):1735-80.

[14] Scarselli F, Gori M, Tsoi AC, Hagenbuchner M, Monfardini G. The graph neural network model. IEEE transactions on neural networks. 2008;20(1):61-80.

[15] Barabási AL, Gulbahce N, Loscalzo J. Network medicine: a network–based approach to human disease. Nature Reviews Genetics. 2011;12(1):56-68.

[16] Bongini P, Bianchini M, Scarselli F. Molecular generative graph neural networks for drug discovery. Neurocomputing. 2021;450:242-52.

[17] Goodfellow I, Bengio Y, Courville A. Deep Learning. MIT Press; 2016.

[18] Ciresan DC, Meier U, Masci J, Gambardella LM, Schmidhuber J. Flexible, High Performance Convolutional Neural Networks for Image Classification. In: Proceedings of the Twenty–Second International Joint Conference on Artificial Intelligence; 2013. p. 1237-42.

[19] Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. Generative Adversarial Nets. In: Ghahramani Z, Welling M, Cortes C, Lawrence N, Weinberger KQ, editors. Advances in Neural Information Processing Systems. vol. 27. Curran Associates, Inc.; 2014. p. 1-9.

[20] Yelmen B, Decelle A, Ongaro L, Marnetto D, Tallec C, Montinaro F, et al. Creating artificial human genomes using generative neural networks. PLOS Genetics. 2021:1009303.

[21] Rossi A, Hosseinzadeh M, Bianchini M, Scarselli F, Huisman H. Multi–Modal Siamese Network for Diagnostically Similar Lesion Retrieval in Prostate MRI. IEEE Transactions on Medical Imaging. 2021;40(3):986-95.

[22] Bongini P, Pancino N, Dimitri GM, Bianchini M, Scarselli F, Lio P. Modular multi–source prediction of drug side–effects with DruGNN. IEEE/ACM Transactions on Computational Biology and Bioinformatics. 2022.

[23] Bandinelli N, Bianchini M, Scarselli F. Learning long–term dependencies using layered graph neural networks. In: The 2010 International Joint Conference on Neural Networks (IJCNN); 2010. p. 1-8.

[24] Pancino N, Bongini P, Scarselli F, Bianchini M. GNNkeras: A Keras–based library for Graph Neural Networks and homogeneous and heterogeneous graph processing. SoftwareX. 2022;18:101061.

[25] Almeida LB. A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. In: IEEE International Conference on Neural Networks. vol. 2; 1987. p. 609-18.

[26] Pineda FJ. Recurrent Back–Propagation and the dynamical approach to adaptive neural computation. Neural Computation. 1989;1:161-72.

[27] Ciano G, Rossi A, Bianchini M, Scarselli F. On Inductive–Transductive Learning With Graph Neural Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2022;44(2):758-69.

[28] Radford A, Metz L, Chintala S. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In: Bengio Y, LeCun Y, editors. 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2–4, 2016, Conference Track Proceedings; 2016. p. 1-16.

[29] Bromley J, Guyon I, LeCun Y, Säckinger E, Shah R. Signature Verification using a "Siamese" Time Delay Neural Network. In: Cowan J, Tesauro G, Alspector J, editors. Advances in Neural Information Processing Systems. vol. 6. Morgan–Kaufmann; 1993. p. 669-88.

[30] Zagoruyko S, Komodakis N. Learning to Compare Image Patches via Convolutional Neural Networks. In: IEEE Conference on Computer Vision and Pattern Recognition 2015. Boston, United States; 2015. p. 4353-61.

[31] Chopra S, Hadsell R, LeCun Y. Learning a similarity metric discriminatively, with application to face verification. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). vol. 1; 2005. p. 539-46.

[32] Yi D, Lei Z, Liao S, Li SZ. Deep Metric Learning for Person Re–identification. In: 2014 22nd International Conference on Pattern Recognition; 2014. p. 34-9.

[33] Hadsell R, Chopra S, LeCun Y. Dimensionality Reduction by Learning an Invariant Mapping. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06). vol. 2; 2006. p. 1735-42.

[34] Xu J, Xiang L, Liu Q, Gilmore H, Wu J, Tang J, et al. Stacked Sparse Autoencoder (SSAE) for Nuclei Detection on Breast Cancer Histopathology Images. IEEE Transactions on Medical Imaging. 2016;35(1):119-30.

[35] Singh S, Pandey SK, Pawar U, Janghel RR. Classification of ECG arrhythmia using recurrent neural networks. Procedia Computer Science. 2018;132:1290-7.

[36] Zhang X, Yao L, Kanhere SS, Liu Y, Gu T, Chen K. Mindid: Person identification from brain waves through attention–based recurrent neural network. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies. 2018;2(3):1-23.

[37] Kusztos R, Dimitri GM, Liò P. Neural Models for Brain Networks Connectivity Analysis. In: International Meeting on Computational Intelligence Methods for Bioinformatics and Biostatistics. Springer; 2018. p. 212-26.

[38] Chen R, Stewart WF, Sun J, Ng K, Yan X. Recurrent neural networks for early detection of heart failure from longitudinal electronic health record data: implications for temporal modeling with respect to time before diagnosis, data density, data quantity, and data type. Circulation: Cardiovascular Quality and Outcomes. 2019;12(10):e005114.

[39] Dimitri GM, Gallicchio C, Micheli A, Morales MA, Ungaro E, Vozzi F. A preliminary evaluation of Echo State Networks for Brugada syndrome classification. In: 2021 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE; 2021. p. 1-8.

[40] Hawkins J, Bodén M. The applicability of recurrent neural networks for biological sequence analysis. IEEE/ACM Transactions on Computational Biology and Bioinformatics. 2005;2(3):243-53.

[41] Pollastri G, Baldi P. Prediction of contact maps by GIOHMMs and recurrent neural networks using lateral propagation from all four cardinal corners. Bioinformatics. 2002;18(suppl_1):S62-70.

[42] Zeng M, Zhang F, Wu FX, Li Y, Wang J, Li M. Protein–protein interaction site prediction through combining local and global features with deep neural networks. Bioinformatics. 2020;36(4):1114-20.

[43] Visibelli A, Bongini P, Rossi A, Niccolai N, Bianchini M. A deep attention network for predicting amino acid signals in the formation of α–helices. Journal of Bioinformatics and Computational Biology. 2020;18(05):2050028.

[44] Villegas-Morcillo A, Gomez AM, Cordovilla JAM, Calle VES. Protein fold recognition from sequences using convolutional and recurrent neural networks. IEEE/ACM Transactions on Computational Biology and Bioinformatics. 2020.

[45] Quang D, Xie X. DanQ: a hybrid convolutional and recurrent deep neural network for quantifying the function of DNA sequences. Nucleic Acids Research. 2016;44(11):e107-7.

[46] Cicaloni V, Costanti F, Pasqui A, Bianchini M, Niccolai N, Bongini P. A Bioinformatics approach to investigate structural and non–structural proteins in human coronaviruses. Frontiers in Genetics. 2022:1303.

[47] Bongini P, Trezza A, Bianchini M, Spiga O, Niccolai N. A possible strategy to fight COVID-19: interfering with spike glycoprotein trimerization. Biochemical and biophysical research communications. 2020;528(1):35-8.

[48] Bongini P, Pancino N, Scarselli F, Bianchini M. BioGNN: How Graph Neural Networks Can Solve Biological Problems. In: Artificial Intelligence and Machine Learning for Healthcare. Springer; 2023. p. 211-31.

[49] Dimitri GM, Liò P. DrugClust: a machine learning approach for drugs side effects prediction. Computational Biology and Chemistry. 2017;68:204-10.

[50] Zhang W, Liu X, Chen Y, Wu W, Wang W, Li X. Feature–derived graph regularized matrix factorization for predicting drug side effects. Neurocomputing. 2018;287:154-62.

[51] Menden MP, Iorio F, Garnett M, McDermott U, Benes CH, Ballester PJ, et al. Machine learning prediction of cancer cell sensitivity to drugs based on genomic and chemical properties. PLoS one. 2013;8(4):e61318.

[52] Periwal V, Bassler S, Andrejev S, Gabrielli N, Patil KR, Typas A, et al. Bioactivity assessment of natural compounds using machine learning models trained on target similarity between drugs. PLoS Computational Biology. 2022;18(4):e1010029.

[53] Xiong J, Xiong Z, Chen K, Jiang H, Zheng M. Graph neural networks for automated de novo drug design. Drug Discovery Today. 2021;26(6):1382-93.

[54] Hao Z, Lu C, Huang Z, Wang H, Hu Z, Liu Q, et al. ASGN: An active semi–supervised graph neural network for molecular property prediction. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining; 2020. p. 731-52.

[55] Dai E, Wang S. Towards self-explainable graph neural network. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management; 2021. p. 302-11.

[56] Jang E, Gu S, Poole B. Categorical reparameterization with gumbel-softmax. arXiv preprint arXiv:161101144. 2016.

[57] Zhang Z, Chen L, Zhong F, Wang D, Jiang J, Zhang S, et al. Graph neural network approaches for drug–target interactions. Current Opinion in Structural Biology. 2022;73:102327.

[58] Lin X, Quan Z, Wang ZJ, Ma T, Zeng X. KGNN: Knowledge Graph Neural Network for Drug–Drug Interaction Prediction. In: IJCAI. vol. 380; 2020. p. 2739-45.

[59] Sun F, Sun J, Zhao Q. A deep learning method for predicting metabolite–disease associations via graph neural network. Briefings in Bioinformatics. 2022;23(4):bbac266.

[60] Alghamdi N, Chang W, Dang P, Lu X, Wan C, Gampala S, et al. A graph neural network model to estimate cell–wise metabolic flux using single–cell RNA–seq data. Genome Research. 2021;31(10):1867-84.

[61] Sun Z, Yin H, Chen H, Chen T, Cui L, Yang F. Disease prediction via graph neural networks. IEEE Journal of Biomedical and Health Informatics. 2020;25(3):818-26.

[62] Pancino N, Rossi A, Ciano G, Giacomini G, Bonechi S, Andreini P, et al. Graph Neural Networks for the Prediction of Protein–Protein Interfaces. In: ESANN; 2020. p. 127-32.

[63] Grindley HM, Artymiuk PJ, Rice DW, Willett P. Identification of tertiary structure resemblance in proteins using a maximal common subgraph isomorphism algorithm. Journal of molecular biology. 1993;229(3):707-21.

[64] Bron C, Kerbosch J. Algorithm 457: finding all cliques of an undirected graph. Communications of the ACM. 1973;16(9):575-7.

[65] Guerranti F, Mannino M, Baccini F, Bongini P, Pancino N, Visibelli A, et al. CaregiverMatcher: graph neural networks for connecting caregivers of rare disease patients. Procedia Computer Science. 2021;192:1696-704.

[66] Rossi A, Hosseinzadeh M, Bianchini M, Scarselli F, Huisman H. Multi–modal siamese network for diagnostically similar lesion retrieval in prostate MRI. IEEE Transactions on Medical Imaging. 2020;40(3):986-95.

[67] Rossi A, Bianchini M, Scarselli F. Robust prostate cancer classification with siamese neural networks. In: International Symposium on Visual Computing. Springer; 2020. p. 180-9.

[68] Pancino N, Graziani C, Lachi V, Sampoli ML, Ștefănescu E, Bianchini M, et al. A mixed statistical and machine learning approach for the analysis of multimodal trail making test data. Mathematics. 2021;9(24):3159.

[69] Wang G, Ye JC, De Man B. Deep learning for tomographic image reconstruction. Nature Machine Intelligence. 2020;2(12):737-48.

[70] Reader AJ, Corda G, Mehranian A, da Costa-Luis C, Ellis S, Schnabel JA. Deep learning for PET image reconstruction. IEEE Transactions on Radiation and Plasma Medical Sciences. 2020;5(1):1-25.

[71] Dimitri GM, Spasov S, Duggento A, Passamonti L, Liò P, Toschi N. Multimodal and multicontrast image fusion via deep generative models. Information Fusion. 2022;88:146-60.

[72] Rossi A, Vannuccini G, Andreini P, Bonechi S, Giacomini G, Scarselli F, et al. Analysis of brain NMR images for age estimation with deep learning. Procedia Computer Science. 2019;159:981-9.

[73] Andreini P, Bonechi S, Bianchini M, Mecocci A, Scarselli F. Image generation by GAN and style transfer for agar plate image segmentation. Computer Methods and Programs in Biomedicine. 2020;184:105268.

[74] Andreini P, Bonechi S, Bianchini M, Garzelli A, Mecocci A. Automatic image classification for the urinoculture screening. Computers in Biology and Medicine. 2016;70:12-22.

[75] Andreini P, Ciano G, Bonechi S, Graziani C, Lachi V, Mecocci A, et al. A Two–Stage GAN for High–Resolution Retinal Image Generation and Segmentation. Electronics. 2021;11(1):60.

[76] Karras T, Aila T, Laine S, Lehtinen J. Progressive Growing of GANs for Improved Quality, Stability, and Variation. ArXiv. 2018;abs/1710.10196.

[77] Ciano G, Andreini P, Mazzierli T, Bianchini M, Scarselli F. A multi–stage GAN for multi–organ chest X–ray image generation and segmentation. Mathematics. 2021;9(22):2896.

[78] Van der Laak J, Litjens G, Ciompi F. Deep learning in histopathology: the path to the clinic. Nature Medicine. 2021;27(5):775-84.

[79] Sultan AS, Elgharib MA, Tavares T, Jessri M, Basile JR. The use of artificial intelligence, machine learning and deep learning in oncologic histopathology. Journal of Oral Pathology & Medicine. 2020;49(9):849-56.

[80] Bayramoglu N, Kannala J, Heikkilä J. Deep learning for magnification independent breast cancer histopathology image classification. In: 2016 23rd International conference on pattern recognition (ICPR). IEEE; 2016. p. 2440-5.

[81] Dimitri GM, Andreini P, Bonechi S, Bianchini M, Mecocci A, Scarselli F, et al. Deep Learning Approaches for the Segmentation of Glomeruli in Kidney Histopathological Images. Mathematics. 2022;10(11):1934.

[82] Meconcelli D, Bonechi S, Dimitri GM. Deep learning approaches for mice glomeruli segmentation. In: ESANN 2022 proceedings European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning; 2022. p. 1.

[83] Chen LC, Papandreou G, Kokkinos I, Murphy K, Yuille AL. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2018;40(4):834-48.

[84] Bonechi S, Andreini P, Mecocci A, Giannelli N, Scarselli F, Neri E, et al. Segmentation of Aorta 3D CT Images Based on 2D Convolutional Neural Networks. Electronics. 2021;10(20):2559.

[85] Bonechi S. A weakly supervised approach to skin lesion segmentation. In: ESANN 2022 proceedings European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning; 2022. p. 1.

[86] Bonechi S, Bianchini M, Bongini P, Ciano G, Giacomini G, Rosai R, et al. Fusion of visual and anamnestic data for the classification of skin lesions with deep learning. In: International Conference on Image Analysis and Processing. Springer; 2019. p. 211-9.

[87] Andreini P, Pancino N, Costanti F, Eusepi G, Corradini BT. A Deep Learning approach for oocytes segmentation and analysis. In: ESANN 2022 proceedings European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning; 2022. p. 1.

[88] Giacomini G, Graziani C, Lachi V, Bongini P, Pancino N, Bianchini M, et al. A Neural Network Approach for the Analysis of Reproducible Ribo–Seq Profiles. Algorithms. 2022;15(8):274.

[89] Benini M, Bongini P, Trentin E. A Novel Representation of Graphical Patterns for Graph Convolution Networks. In: IAPR Workshop on Artificial Neural Networks in Pattern Recognition. Springer; 2023. p. 16-27.

[90] Scarselli F, Gori M, Tsoi AC, Hagenbuchner M, Monfardini G. Computational capabilities of graph neural networks. IEEE Transactions on Neural Networks. 2008;20(1):81-102.

[91] Oneto L, Navarin N, Biggio B, Errica F, Micheli A, Scarselli F, et al. Towards learning trustworthily, automatically, and with guarantees on graphs: An overview. Neurocomputing. 2022;493:217-43.