CHAPTER

Medical image diagnosis for disease detection: A deep learning approach

3

Mrudang D. Pandya*, Parth D. Shah*, Sunil Jardosh†

Information Technology Department, CHARUSAT, Anand, India Progress Software, Hyderabad, India†*

CHAPTER OUTLINE

1 INTRODUCTION

Before recent decade healthcare data availability was very hard, if it was available at that time the size of the data was very small. But in the current world, those issues are gone. Because of the incredible development in image procurement devices, the data is quite large, which makes it more applicable and effective for image analysis. This express progress in biomedical digital images and modalities involves widespread and monotonous efforts by the experts of medical domains such as radiologists, general physicians, etc. Especially for digital image analysis that includes human error,

U-Healthcare Monitoring Systems. https://doi.org/10.1016/B978-0-12-815370-3.00003-7

required huge analysis and diagnosis variations depending on subject experts. To solve these kinds of issues, machine learning (ML) techniques provide automated diagnosis solutions; however, traditional ML methods are not appropriate to handle complex problems. Collaboration between high-performance computing (HPC) and ML gives us promising results in terms of complex problems. HPC and ML collaborations ultimately lead to Deep Learning (DL), and this will give ultimate results on large medical image datasets. This will give accurate analysis and diagnosis results. DL provides us automated feature selection furthermore it does not only identify the disease but also quantify prognostic goal and delivers tortuous estimate models to increase the diagnosis ability of medical experts.

Machine Learning is a most important field that is mainly helpful to make decision-making systems, recommendation systems, image analyzing systems, web searching, and so on. Several ML algorithms such as Artificial Neural Network (ANN), Random Forest (RF), the Hidden Markov Model (HMM), SVM (Support Vector Machine), Gaussian Networks (GN), etc., are applied in the fields of biomedical image processing.

Medical image processing comes under bioinformatics research. Medical digital image processing is one of the data modalities of bioinformatics. Apart from medical images, there are two more data modalities: omics (mostly sequential data) and biomedical signaling. Over the last few years, the use of such advanced DL techniques applied in the fields of medical image processing has grown rapidly. The current evidence of such systems is IBM's WATSONS [1] and GOOGLE DeepMind [2]. These projects have used DL algorithms to solve several bioinformatics problems. IBM's Watson use the ontology of patient health specification records collected by doctors for finding optimal solutions [1]. Google's DeepMind created a DeepMind health system for solving heath-related problems and terminology [2]. Medical imaging domains contain raw data that may not be applied to the traditional algorithms of ML. The reason behind that is structure of medical images same times very complex (number of features are very high). So, handcrafted feature selection (ML) approaches not good for Medical Image analysis. These traditional algorithms that are mainly responsible to extract the feature Vector among such input sequences, for analysis the imaging.

Neural network architecture mainly needs feature vectors for input data. Medical image domain data needs to process the raw data images for feature vectors to give this input to such neural networks. Preprocessing of such raw data is much more expansive and quite more time consuming to do. A Deep Neural Network (DNN) helps to solve these problems through DL. Bioinformatics is broad domain to explore the DL phenomena. A beautiful definition by Merriam-Webster calls "bioinformatics as a collection of data, to do classification make prediction using feature extraction by analysis of biochemical properties and biological information using the computers." Research domains in bioinformatics are protein structure prediction, Gene expression regulation, protein classification, and anomaly classification [3–6]. Data availability in the fields of bioinformatics fall into these

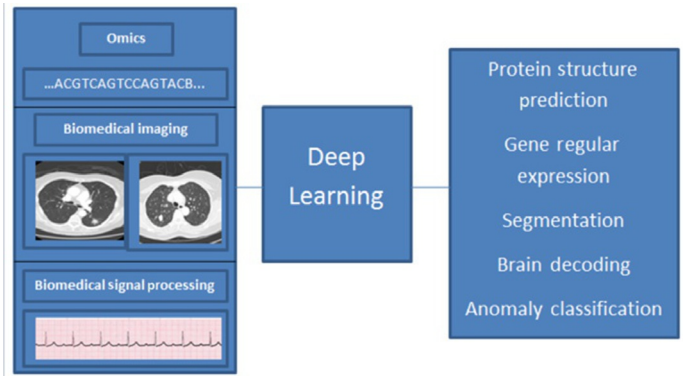

FIG. 1

Bioinformatic modalities through Deep Learning research areas [7].

categories: omics (RNA, DNA, RNase, Protein Sequences, etc.), biomedical imaging (CTscan, MRI, PET, etc.), and biomedical signal processing (ECG, EEG, EMG, etc.). In this chapter, our main focus is on biomedical images. In Fig. 1, we show how DL can be useful in bioinformatic research areas using bioinformatic data modalities.

This chapter provides fundamental knowledge and state-of-the-art approaches about DL in the domain of medical image processing and analysis also gives the broad review of DL algorithms in biomedical image investigation problems in terms of present works and upcoming direction for future scope.

1.1 RELATED WORK

Within the scope of the u-healthcare system, Kiran Khatter and Sapana Malik [8] have contributed in malicious application detection and classification using Android mobile applications. With the use of machine learning algorithms, they have achieved malicious application detection accuracy up to 98.2% and malicious application classification accuracy up to 87.3%.

Furthermore, excellent research done by saba Lon in the field of feature classification on liver diseases using ultrasound data images. This was the automated approach using random partitioning using a back propagation algorithm [9]. They have represented their result in terms of four performance metrics: (1) sensitivity (98.08%), (2) specificity (97.22%), (3) positive predictive values (96.23%), and (4) negative predictive values (98.59%).

Komalsharma and Jitendra Virmani [10] have made a decision support system for classification of normal and medical renal disease with the use of ultrasound images. They have achieved 93.3% accuracy.

2 REQUIREMENT OF DEEP LEARNING OVER MACHINE LEARNING

Machine Learning becomes vitally important in this era that makes an individual's life easier. ML algorithms (MLA) are used for decision-making systems, recommendation systems, identifying objects from images, searching the web, etc. MLAs use training examples to uncover underlying patterns, construct models, and then make predictions on the new database that is testing an example on the model. There are several MLAs used for classification, including an Artificial Neural Network (ANN). The simple architecture of ANN consists of an input layer, an output layer, and hidden layers with processing units. These processing units take input as training examples and learn features that are used to predict class with the help of activation functions and parameters called weight. The network provides a reasonable response to noisy or incomplete input because of its property of distributed associative memory. Experience shows that these networks are very good pattern recognizers that also have the ability to learn and construct unique structures for different problems. Conventional machine-learning techniques were limited in their ability to process natural data in their raw form.

If we construct machine learning systems, then we require domain expertise and careful engineering to develop feature extractors, which convert raw data in to feature vectors that we can use as input to the neural network. Perceptron and shallow Neural Network requires handmade features as input whereas DNNs extract features by themselves with multiple hidden layers. DL is a subfield of ML. In DL feature selection is done by multiple hidden layers [11]. The invention of DL came from the study of ANNs and multilayer perceptrons because they have multiple hidden layers. DL is also called representation learning. For example, in image recognition, it can be interpreted that feature learning is done in the order of pixel, edge, texton, motif, part, and object. Similarly, in text recognition, features are learned in the order of character, word, word group, clause, sentence, and story [7]. There are three categories of DL architecture, namely DNN, Convolution Neural Network (CNN), and Recurrent Neural Network (RNN).

DNN has different architecture depending on which learning algorithm is used (supervised or unsupervised). Based on this, there are mainly three DNN architectures: Multilayer Perceptron (MLP), Deep Belief Network (DBN), and Stacked Auto-Encoder (SAE) [11].

Here is the comparison table of ML and DL; this will give excellent insight into both.

As we are concentrating on medical image analysis, there are some benefits to doing the same thing with DL instead of ML. DL has automated feature selection capability and good performance while we are dealing with large data; these features are really useful while we are dealing with medical image data (Table 1).

Table 1 Machine Learning Versus Deep Learning

Sr No.	Comparison Area	Description
1	Data dependencies	The most important difference between Deep Learning and traditional Machine Learning is its performance as the scale of data increases
2	Hardware dependencies	Deep Learning algorithms heavily depend on high-end machines, contrary to traditional Machine Learning algorithms, which can work on low-end machines
3	Feature engineering	In Machine Learning, most of the applied features need to be identified by an expert and then hand coded as per the domain and data type Deep Learning algorithms try to learn high-level features from data. This is a very distinctive part of Deep Learning and a major step ahead of traditional Machine Learning
4	Problem-solving approach	When solving a problem using a traditional Machine Learning algorithm, it is generally recommended to break the problem down into different parts, solve them individually, and combine them to get the result. Deep Learning, in contrast, advocates solving the problem end-to-end
5	Execution time	A Deep Learning algorithm takes a long time to train. This is because there are so many parameters in such an algorithm that training them takes longer than usual. The state-of-the-art Deep Learning algorithm ResNet takes about 2 weeks to train completely from scratch whereas Machine Learning comparatively takes much less time to train, ranging from a few seconds to a few hours
6	Interpretability	It is easy to interpret the reasoning behind the solution in Machine Learning but it is hard to defend the solution in terms of Deep Learning

2.1 FUNDAMENTAL DEEP LEARNING ARCHITECTURES

Before going into DL, we must know *the requirements of deep architecture over neural networks*. When we want to deal with large complex input and output processing at that time, DLA gives excellent performance instead of using neural networks. While we are dealing with neural networks, feature selection must be done externally. However, in DLA, feature selection will be done automatically. For these reasons, we have to select DLA over neural network architecture.

The fundamental construction of a DNN contains an input layer, many more hidden layers, and output layers. One of the important methodology for representation of learning is Deep Learning through which we can learn and discover several patterns available in data by increasing the level of abstraction [2]. The next question arises, Why do we need DNNs? One of the most important reasons is because it contains several layers of perceptrons (mainly containing one input, several hidden layers,

and one output layer). Other reason is single layer neural networks cannot deal with exclusive-OR like nonlinearly separable functions. For that we required multilayer networks or DNNs. DNN layers mainly compute output when the input sequence of data is given. Each layer of the input vector consists of output values of each unit of the layers that is multiplied with its weight vector of each unit that is present in each layer producing the weighted sum. Nonlinear functions such as sigmoid, hyperbolic tangent, and retified linear unit (ReLU) are applied to the weighted sum that will compute the vision for output layers. The training of DNN architecture mainly aims to optimize the weight so that the appropriate result could be able to learn. Based on different types of layer contraction used in DNNs we have classified that networks as Multilayer Perceptron (MLP), Stacked Auto-Encoder (SAE) and Deep Belief Networks (DBN) etc.

Before we start to introduce all architecture, let us discus the criteria used to determine the network structure.

There are basically three common criteria for all different kinds of techniques. Some techniques have different structures (CNN, RNN) but mainly all have common structures. Here, I have listed three criteria of the structures:

(1) Input layer
(2) Output layer
(3) Hidden layers

If we talk about input layer, there is only one input layer in every architecture. This layer consists of nodes or neurons. The number of nodes is dependent on the features of your data. Sometimes architecture includes one more additional layer called bias.

The output layer count in architecture is the same as the input layer. The output of this layer is in terms of class labels or values. Values you can compare with regression mode and class labels you can compare with machine mode.

Hidden layers are totally dependent on the size of the data. The bigger the data, the more hidden layers you have. When you put more hidden layers, sometimes it leads to a vanishing gradient problem. DLAs can phase this type of problem but RBM and auto encoders have solved this kind of problem. More hidden layers require more computing power. There is no such technique available that gives you the exact number of hidden layers. It always leads to trial and error.

2.1.1 Multilayer Perceptron

MLPs are an extension of ANNs, but these are more toward nonlinearity and layers are stacked together. These networks are trained mainly with supervised learning that requires labeled data. For training, they mainly use a back-propagation algorithm and gradient descent (by minimizing the cost of function by finding the values of parameters). Through these kinds of training techniques, we can archive optimized high-dimensional parameter space. Fig. 2 indicates the basic structure of MLP with one Hidden layer.

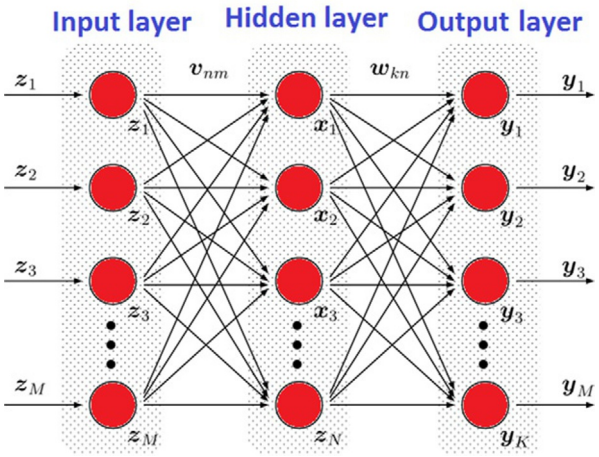

FIG. 2

Basic structure of Multilayer Perceptron (MLP).

2.1.2 Deep Belief Networks

It is one kind of the generative model. It mainly joins the distribution of probability among observation and label P (label | observation). DBNs have "learning features" that means network can learn layer-by-layer with higher level features that are discovered and passes through past layer, thus more complex features that are acknowledges to get better reflected knowledge inside the input structures. The learning phase in DBN mainly contains two phases: a pretraining phase and fine-tuning. The pretraining phase is nothing but unsupervised leaning along with Stacked Restricted Boltzmann's Machine. RBM is also used to extract the features, and after extracting the features, it mainly reconstructs the input. Several studies have been done with DBN/RBM for solving protein residue-residue contact prediction. For understanding the basic structure and flow, Fig. 3 defines the basic flow of a DBN; Fig. 4 defines how we can use RBM over DBN.

Invented by Geoff Hinton, a Restricted Boltzmann machine is an algorithm useful for dimensionality reduction, classification, regression, collaborative filtering, feature learning and topic modeling. It is using Contrastive divergences method to provide approximation for learning the weights. For training the single RBM, weights are updated with gradient descent with the following equation:

$$\Delta w_{ij}(t+1) = w_{ij}(t) + \eta \frac{\partial \log(p(v))}{\partial w_{ij}} \tag{1}$$

Where $p(v)$ is the probability of visible vector, Z is the partition function (used for normalizing the value), and $E(v, u)$ is the energy function assigned to the network.

These are the following steps to perform:

Visible units initialization to a training vector.

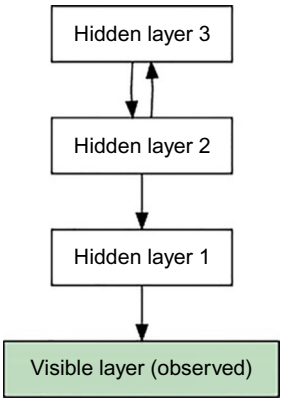

FIG. 3

Basic flow of a Deep Belief Network (DBN).

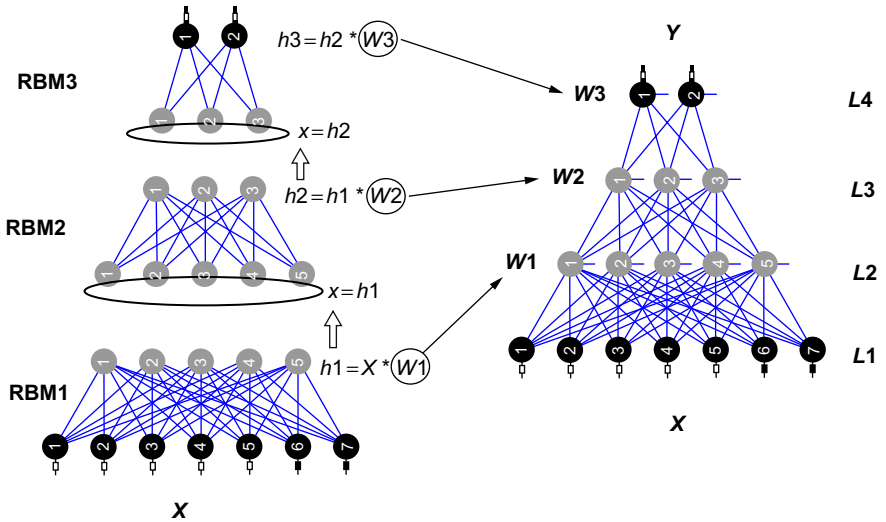

FIG. 4

DBN using RBM.

(1) With given visible units update the hidden units:

$$p(h_j = 1 \mid \mathbf{V}) = \sigma\left(b_j + \sum_i v_i w_{ij}\right). \tag{2}$$

(2) With given hidden units update the visible units:

$$p(v_i = 1 \mid \mathbf{H}) = \sigma\left(a_i + \sum_j h_i w_{ij}\right). \tag{3}$$

(3) Reupdate the hidden units in parallel given the reconstructed visible units using the same equation as in step 2.
(4) Do a weight update:

$$\Delta w_{ij} \propto \langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{reconstruction}} \tag{4}$$

2.1.3 Stacked Auto-Encoder

Auto-Encoder is a part of ANN, which is unsupervised pretraining procedure to train the networks. It mainly contains one input layer, one hidden layer,output layer is also similar, among these hidden layers the neurons that are used is less in number as compare to input and output layers. The working process of the network mainly compresses the original input and at output it will decompress it again to get the data back. By definition of Auto-Encoder, it mainly says that this network takes the original input, compresses it, and reconstructs the inputs that have specific meaning to extract the features for some specific application. The main goal of the network is to regenerate input by getting the target value among the input. These are again divided into three types:

(a) *Denoising Auto-Encoder*: It is used for reconstruction of corrupted data.
(b) *Sparse Auto-Encoder*: This type of network mainly has more numbers of several hidden units between input and output sequences that can learn from the low dimensional feature unit. It can also add some extra features to enhance the function to improve the loss for "sparsity" constraint form these several hidden units.
(c) *Stacked Auto-Encoder*: Stacked Auto-Encoders are mainly used for pretraining the network that stacked in with a greedy layer-wise structure. It will make network to learn more deeper. These type of network mainly contains several layers. Each layer have specific function to extract the feature. As an example some layers are responsible for extracting edge form given image and other layers can find several hidden patterns. These layers are stacked together to learn the high an accurate order of features.

Auto-Encoder in biomedical images: Li et al. [12] developed the method with the use of a deeply stacked auto-encoder for denoising for solving problems of protein structure reconstruction. By using these techniques, it mainly gets the matching score to approximately 70% of the accurate result is obtained (Fig. 5).

2.1.4 Convolution Neural Networks

CNN is a specialized kind of network for processing data into grids such as topology. A CNN indicates to the network employee the mathematical operation called "convolution." Convolution is a special kind of linear operation. The CNN approach is one kind of simple matrix multiplication (Fig. 6).

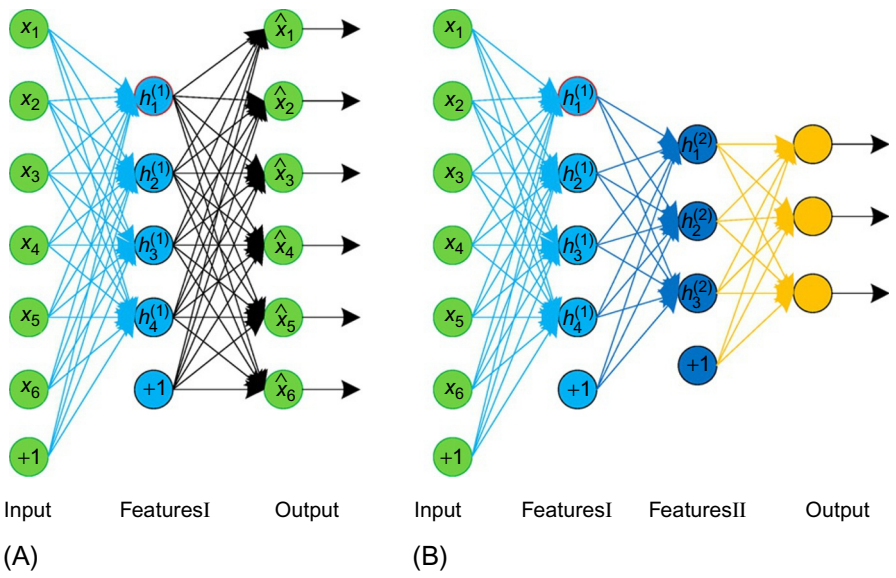

FIG. 5

Model of Auto-Encoder.

Convolution architecture

The basic architecture contains three layers:

(1) Conv layers (convolution).
(2) Dimensionality reduction layers (Pooling).
(3) Fully connected layers.

Convolution layers. The first layer in the convolutional layer is to make the input convolve. Suppose input size of an image is $32 \times 32 \times 3$. The best way to express these conv layers is to imagine a flashlight that shines over the top left of the image. The flashlight covers a 5×5 area; now these will slide across the input images. In machine learning, these flashlights are called filters (also known as kernels) and the region that it is shining over is called the receptive fields. The filter has to be the same as the depth of the input so these dimensions are $5 \times 5 \times 3$. The first position of the filter could be at the top left corner. As the filter slides over the convolving around the input images, it multiplies the values in the filter with the original input of the image (element-wise multiplications). These multiplications are all summed so now we have a single number. Repeat the process. The next step is to move these filters to 1 unit and then right again 1 unit. These processes continue. Each unique location on the input volume produces a number after sliding the filter over all the locations. Now these feature map or activation function is mapped with array size of $28 \times 28 \times 1$ (because input size is $32 \times 32 \times 3$) array of number. The reason we will

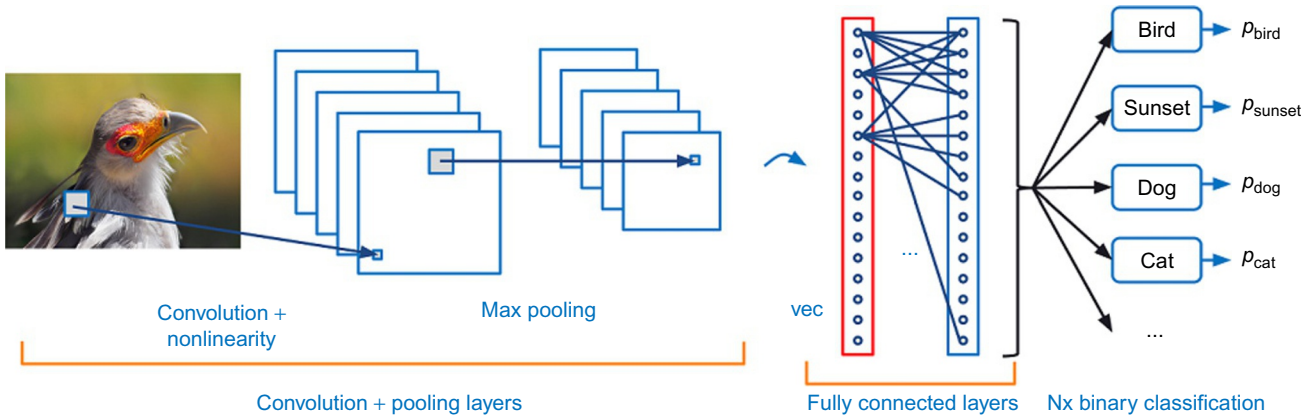

FIG. 6

CNN architecture.

get 28 × 28 accuracy is that there is a total of 784 different locations, that is, 5 × 5 filters on 32 × 32 input images. These 784 are mapped with 28 × 28 arrays.

Stride and pooling layers. These are the two main parameters to modify the behavior of each layer. After choosing the filter size, we also choose the stride and padding. Stride controls the filters convolving around the input volume. These filters convolve around the input volume by shifting one unit at a time. By default, these stride sizes are kept at one. Stride is normally set away to get the output in the form of an integer value (Fig. 7).

After applying a 5 × 5 × 3 filter to a 32 × 32 × 3 input volume, thus output volume would be 28 × 28 × 3. The spatial dimensions will decrease as, applying these convolution layers, the size of the volume will decrease faster. In each layer of the network if we want to preserve the information for the original input volume. For that we want our output volume same as input volume (32 × 32 × 3). For such kind of things we required zero padding of size 2. Zero padding pads to the input volume worth zeros around the border (Fig. 8).

If a stride of and the size of zero padding is calculated by the formula given below:

$$\text{Zero padding} = \frac{(k-1)}{2} \tag{5}$$

where $k =$ is the filter size of the input and output volume for spatial dimensions. The formula for calculating the output size given by convolution layers is:

$$O = \frac{(W-K+2P)}{S} + 1 \tag{6}$$

where $O =$ is the output height/length, W is the input height/length, K is the filter size, P is the padding, and S is the stride. Apply these nonlinear layers immediately after the convolution layer. The purpose of this nonlinearity to a system is to compute the linear operation during the convolution layers. Nonlinear functions such as tanh

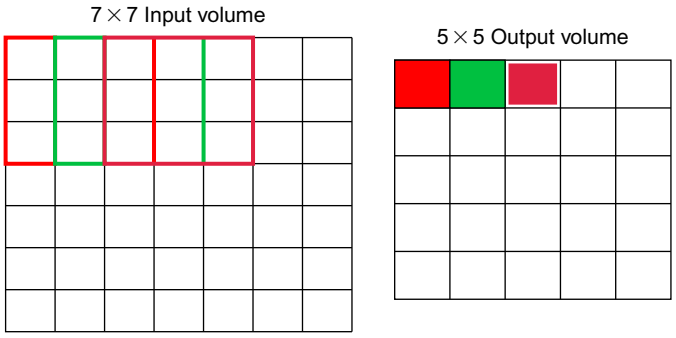

FIG. 7

Filter processing.

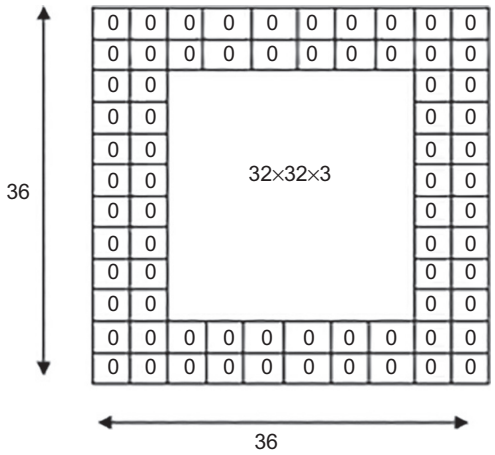

FIG. 8

The input volume is 32 × 32 × 3. If we imagine two borders of zeroes around the volume, this gives us a 36 × 36 × 3 volume. Then, when we apply our conv layer with our three 5*5*3 filters and a stride of 1, then we will also get a 32 × 32 × 3 output volume.

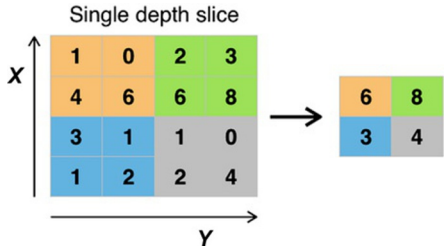

FIG. 9

Example of Maxpool with a 2 × 2 filter and stride of 2.

and sigmoid are used. After applying these nonlinear programs, we choose this pooling layer to downsample the layer. There are several pooling options to input vthe olume and output the maximum number in every subregion that filters convolve around (Fig. 9).

Fully connected. Layer is for the detection of higher level of features which is situated at the end of the network. This layer basically takes input volumes and outputs an *N* dimensional vector where *N* is the number of classes of the program. Each number of classes defines the probability of a certain class. For example, if the resulting vector for a digit classification program is [0.0.1.75.0.0.0.0.0.5], this represents that a 100% probability that the image is 2 digit. 75% probability that the image is 2 digit and so on. In this way fully connected layer extract the features to correlate the classes.

2.1.5 Recurrent Neural Network

This type of network mainly deals with sequential data. Like all other Feed Forward Networks, when all the input as well as output sequences are independent of each other (for example like predicting the next word of a sentence based on the previous knowledge of the sentence during training). In RNN architecture mainly has direct cycles for feeding back the data again to the network that mainly takes the previous information through the network and makes prediction of the next sequences. The main idea behind a RNN is to mainly have memory that mainly captures the data by performing several calculations $S_t = f(Uxt + Ws(t-1)$. S_t is having a hidden state over t time, U is the sum of the weight between the current and hidden, W is the weight among the previous and current hidden units of the layer in the network, V is the weight among the input and output layer, and f is the hidden layer of function (activation functions are used such as ReLu, hyperbolic tangent, etc.) RNNs mainly share these above-shown parameters (U,W,V) at every layer of the network. These mainly use back propagation while training the networks. Fig. 10 shows the RNN and bidirectional RNN networks.

T when $t = 1$ to $t = \tau$, we update the equation:

$$
\begin{aligned}
a^{(t)} &= b + Wh^{(t-1)} + Ux^{(t)} \\
h^{(t)} &= \tanh\left(a^{(t)}\right) \\
o^{(t)} &= c + Vh^{(t)} \\
\hat{y}^{(t)} &= \text{softmax}\left(\boldsymbol{o}^{(t)}\right)
\end{aligned}
$$

In above questions b and c are bias vectors along with the weight matrix U, V and W form input to hidden unit, hidden to output unit, hidden to hidden connections. These above figure maps RNN to the input sequence and output sequence of the same

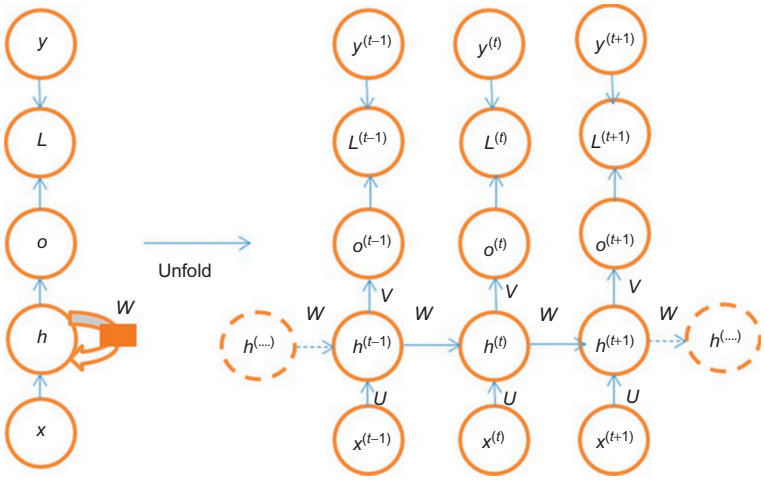

FIG. 10

RNN Deep Learning structure.

length. The total loss for a given sequence of value x is paired with sequence y value then take sum of the loss over all the time stamp.

An RNN is a network of neurons with each one directly connected to every other node. Each node is time varying by calculating the real-valued activation function. Each connection has a modifiable real value of weight node. Output nodes calculate the result and several hidden nodes mainly modify the data as per the input and output data feed to the network.

The basic difference between a feed-forward neural network and a RNN is shown in the figure. These feed-forward neurons have one connection form input to output. RNN has an extra connection from output again to input that is mainly feedback to network and along with these it also contain activation function that can flow to the loop. When many of these feed-forward and RNN are connected together, they mainly form a RNN.

RNNs mainly have two advantages:

- *Store information*: These recurrent networks are feedback to store the information over time in the form of an activation function so it mainly has to store information in a form of memory.
- *Learn from sequential data*: These may have sequential data of any length. As shown in the above figure, it is stated that using one fixed input and one fixed output may have one to many, many to one, many to many input to output sequences. So using these approaches, one can solve the problem for sequential data input to predict the next upcoming sequences.

Researchers are using the RNN along with LSTM and GRU to get more powerful results. Several researchers are using RNN along with GRU, which is the popular method for solving the problem for extracting the features for protein structure prediction. By using LSTM, it mainly has three gates: an input gate, a forget gate, and an output gate.

Why use an RNN? This type of network has advantage to make unit context to extract the information over a large set of neighborhoods specially for recognizing the images. With the help of these information these RNN can have watch over a large input space, CNN have this limitation over higher layer of abstraction. So, a recurrent network connection increases the depth when they keep the number parameter low by sharing the weight vector. Additionally, these RNNs also have the ability to handle the sequential data. These types of network also handle biologically inspired tasks, as brains can do.

How does LSTM improve the RNN?

One of the drawbacks of an RNN is that the contextual information is limited and, by using back propagation, this is quite time consuming and will not properly work. So, it is notable for vanishing or exploding the output result, mainly called the Vanishing Gradient or Exploding Gradient. These take huge time to solve the problem of vanishing gradients.These exploding gradients lead to weight oscillation that reduces the quality of the network. For storing the information over some time interval exploding

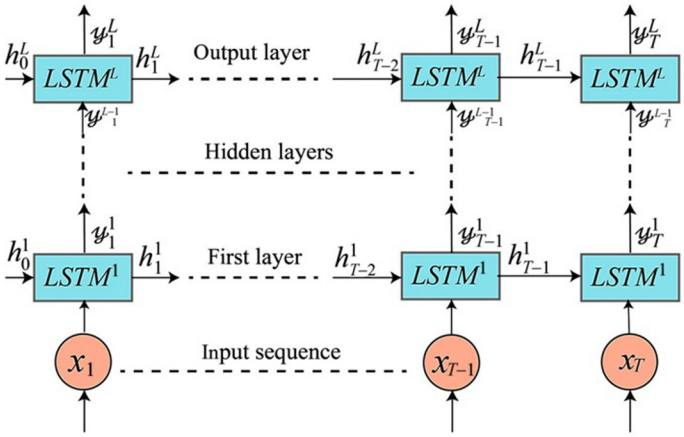

FIG. 11

LSTM basic architecture.

gradients took longer time and created the error flow. To address this problem Hocheriter uses LSTM to solve the problem of error backflow. In LSTM they are using special units that improve the RNN and create a bridge.

LSTM mainly consists of a forget gate, input gates, and an output gate. The LSTM cell is a step-wise gate, and each cell is connected in a network. The first step is the forget gate at h_{t-1} and x_t that mainly computes the output f_t over a number between 0 and 1. These cells are multiplied with the cell c_{t-1} that gets the cell to either forget everything or keep the result of the output. For example, a value of 0.5 means the cell loses/forgets 50% of the information. The next step of the input gate is updating the cell so it mainly multiplies the output of I t and ct and then adding these values to input $c_{t-1} \times f_t$. The final output is computed and multiplied o_t with tanh to get the previous result to $ht = o_t \times \tanh(C_t)$ and $o_t = \sigma \times W_o[ht-1,xt] + bo$ (Fig. 11).

3 IMPLEMENTATION ENVIRONMENT

DL is a vital new domain of ML that incorporates a varied range of NN architecture intended to finish various tasks. CNN has proven to be an excellent approach for healthcare and medical image analysis. Because conventional ML needs determination and calculation of features from which the algorithm learns, DL approaches (e.g., CNN) learn the important features as well as the optimized weighting of those features to make predictions for new data. Here we will describe some of the libraries and tools that exist to aid in the proficient execution of DL as applied to healthcare and medical images.

3.1 TOOLKIT SELECTION/EVALUATION CRITERIA [13]

In this section, we have presented all criteria that are required for selection or evaluation of toolkits so you can have a clear choice of toolkit on which to work. Table 2 describes all information regarding the selection or evaluation of toolkits.

3.2 TOOLS AND TECHNOLOGY AVAILABLE FOR DEEP LEARNING [13]

Table 3 describes all the available tools and technology, with a short description and some sample code with the application, "How to create a deep learning model?"

3.3 DEEP LEARNING FRAMEWORK POPULARITY LEVELS [14]

Here we have shown the rankings of 23 open-source DL libraries that are valuable for data science, based on Github and Stack Overflow activity as well as Google search results. Table 4 shows standardized scores where a value of 1 means one standard deviation above average (average = score of 0). See below for methods.

Table 2 Toolkit Selection Criteria [13]

Evaluation Criteria	Details
Languages	Languages are one of the important parameters for toolkit selection. Computer languages show how efficient a particular toolkit is. If we know that this particular toolkit builds using some language and we are also familiar with that language, we can work properly and if we want to modify the toolkit, then we can also give our input
Documentation	The quality of documentation and the given and explained examples in the documentation will lead you to create good solutions for your problem. Well-maintained documentation is also an indication of good stability of a toolkit
Development environment	To reduce the programing complexity for network creation, there is a clear need for a development environment. Through this we can have a connection with graphical tools and some visualization tools. So, after implementing a program, we want some graphical representation or visualization, that problem can be solved by the development environment
Execution speed	This is related to the speed of essentially organizing or segmenting the image with the trained network. This thing involved lots of calculations per pixel. For medical applications, training is less important than training speed. So execution speed is one of the important parameters for toolkit selection
Training speed	Training Speed is mainly dependent on the math libraries used in a toolkit. It is also dependent on the type of tasks and images
GPU support	In today's era of big data, medical image data size becomes very large. For this purpose, we must speed up things. GPU can give us excellent speed in comparison to CPU processing. So those

Continued

Table 2 Toolkit Selection Criteria [13]—Cont'd

Evaluation Criteria	Details
	toolkits that have GPU integration support have more benefits than other toolkits. So GPU support we can consider as one of the selection parameters of toolkits
Maturity level	Maturity level is totally our subjective assessment about how mature a toolkit is? That measurement you can get by grouping a big user base, some error fixing in previous months or decent backing community
Model library	Toolkits have a library of code that generates networks, and the nodes have associated weights. This is like a platform where one may download various popular orientations of networks and weights
GitHub Commits	The older the toolkit, the more GitHub Commits are available. It is the indicator of how many times toolkits have changed after putting on the GitHub. Sometimes it is misleading because newly released toolkits have fewer commits than older ones. So at that time we cannot any decision about maturity and stability of that toolkit
GitHub contributors	The more contributors, the more user support you have. You can explore more applications when user contributors are greater

Table 3 Deep Learning Toolkits

Toolkits	Description
Caffe	• One of the highest rated and most mature tools created by Berkeley Vision and Learning Center • Architecture is based on pure C++ or CUDA to support DL • It has support of command line, Python, and MATLAB interfaces • Fast • Easy switching between CPU and GPU–Caffe::set mode(Caffe::GPU)
Deeplearning4j	• It is written in Java with a Scala API • It has GPU support • Not very popular in medical imaging • Good performance and supports multiple GPUs
Tensor flow	• Created by Google and gathered lots of popularity • It provides excellent performance and multiple CPUs and GPUs support • Educational tool available as a web app (http://playground.tensorflow.org/)
Theano	• Uses symbolic logic and written in Python • Easy to build a network but challenging to create a full solution • Documentation quality is fair

Table 3 Deep Learning Toolkits—Cont'd

Toolkits	Description
Keras	• Created in Python and we can use with Theano or Tensorflow backend • Easy to read and build • Version 2.1.1is the latest for keras
MXNet	• Written in C++ with collaboration with many languages • Multi-GPU support • For learning purposes, training examples are listed on GitHub
Lasagne	• Created in Python and we can say that it extension of Theano • Easier to build than Theano
Cognitive Network Toolkit (CNTK)	• Developed by Microsoft • Performance is generally good • Usage is currently less than many others
DIGITS	• Developed by NVIDIA • Web-based tool for developing deep networks • For identification of errors in the text file, it has a network visualization tool • Multiple GPU support (https://www.youtube.com/watch?v=dgxe15vCR7s)
Torch	• Created in C • Performance is very good • You can get reference for Torch from the below link (http://torch.ch/docs/tutorials-demos.html)
PyTorch	• Newcomer • It is a Python front end to the Torch computational engine. It is an integration of Python with the Torch engine • Performance is higher than Torch with GPU integration facility • Flexible • You can get reference for PyTorch from the below link (http://pytorch.org/tutorials/beginner/pytorch_with_examples.html#pytorch-nn)
Chainer	• Difference between other toolkits and chainer is that it is building the network as a section of its computation • Stores its computations rather than the programming logic

Table 4 Deep Learning Library Ranking [14]

Library	Rank	Overall	Github	Stack Overflow	Google Results
tensorflow	1	10.87	4.25	4.37	2.24
keras	2	1.93	0.61	0.83	0.48
caffe	3	1.86	1.00	0.30	0.55
theano	4	0.76	−0.16	0.36	0.55
pytorch	5	0.48	−0.20	−0.30	0.98

Continued

Table 4 Deep Learning Library Ranking [14]—Cont'd

Library	Rank	Overall	Github	Stack Overflow	Google Results
sonnet	6	0.43	−0.33	−0.36	1.12
mxnet	7	0.10	0.12	−0.31	0.28
torch	8	0.01	−0.15	−0.01	0.17
cntk	9	−0.02	0.10	−0.28	0.17
dlib	10	−0.60	−0.40	−0.22	0.02
caffe2	11	−0.67	−0.27	−0.36	−0.04
chainer	12	−0.70	−0.40	−0.23	−0.07
paddlepaddle	13	−0.83	−0.27	−0.37	−0.20
deeplearning4j	14	−0.89	−0.06	−0.32	−0.51
lasagne	15	−1.11	−0.38	−0.29	−0.44
bigdl	16	−1.13	−0.46	−0.37	−0.30
dynet	17	−1.25	−0.47	−0.37	−0.42
apache singa	18	−1.34	−0.50	−0.37	−0.47
nvidia digits	19	−1.39	−0.41	−0.35	−0.64
matconvnet	20	−1.41	−0.49	−0.35	−0.58
tflearn	21	−1.45	−0.23	−0.28	−0.94
nervana neon	22	−1.65	−0.39	−0.37	−0.89
opennn	23	−1.97	−0.53	−0.37	−1.07

4 APPLICABILITY OF DEEP LEARNING IN FIELD OF MEDICAL IMAGE PROCESSING [15]

Recognizing anomalies and quantifying dimensions and modifications over time are the initial tasks for image analysis. Machine learning based automated tools are really key features to enhance the quality of image analysis. The DL technique is a comprehensively useful technique that provides state-of-the-art accuracy. DL has opened new doors in medical image scrutiny. DL applications in medical domains cover a large range of complications, going from cancer detection and disease monitoring to custom-made behavior suggestions. Nowadays, numerous sources of data from medical digital imaging (X-ray, CT, and MRI scans) and genomic sequences have brought a huge quantity of information to the physician.

4.1 CURRENT RESEARCH APPLICATIONS IN THE FIELD OF MEDICAL IMAGE PROCESSING

Here is the list of current research applications of DL in the field of medical image processing:

- Tumor detection (skin cancer, lung cancer, breast cancer, brain cancer etc.)
- Tracking tumor development

- Blood flow quantification and visualization
- Medical interpretation
- Diabetic retinopathy [16]
- Alzheimer's and Parkinson's diseases detection
- Liver disease detection [9]
- Renal disease detection [10]

5 HYBRID ARCHITECTURES OF DEEP LEARNING IN THE FIELD OF MEDICAL IMAGE PROCESSING [17]

One of the state-of-the-art approaches is the DL hybrid architecture approach. This approach generally combines two or more individual DLAs and makes a hybrid architecture. Here is the list of some hybrid architectures, although they are not at the maturity level.

- Convolutional Auto-Encoder (CAE).
- Deep Spatiotemporal Neural Networks (DST-NNs).
- Multidimensional Recurrent Neural Network (MD-RNNs).

Below is a list of different papers that include work on hybrid approaches with their application domains (Table 5).

Table 5 Hybrid Approaches in Medical Domains [18–20]

Author	Database	Application
Tom Brosch [18]	They have used 5–250 samples for segmentation from the MS clinical dataset	Multiple sclerosis lesion segmentation *Model used*: CAE
Mohammed Shameer Iqbal [19]	Here they have taken 20 students dataset of their 0–9 handwritten digits images and audio data. In totality they have created 2000 handwritten images and 2000 audio recording data of 0–9 digits	Character classification from two different modalities: image and audio *Model*: Deep belief network with contractive stacked restricted Boltzmann machine
Sayan Ghosh [20]	1. Flickr8k and Flickr30k (Image) 2. DISFA and BP4D datasets (Psychological Distress)	1. Prediction of psychological distress 2. Detection of facial action units using multilabel CNN *Model*: Multilabel CNN

6 CHALLENGES OF DEEP LEARNING IN THE FIELDS OF MEDICAL IMAGINING [17]

Although DL has given excellent performance in almost all bioinformatic applications, it still faces some challenges, as described below:

Unavailability of large data & imbalance data: Especially in cancer domain because of some security reasons, we cannot get large amounts of data. Biomedical imagining data modalities have lots of imbalanced data for classification into subcategories. Until now, several DLAs have achieved outstanding results over large data. To do this for biomedical imaging is quite a challenging task. To overcome these problems for such Medical Imagining through use of good preprocessing algorithms, cost-responsive learning and algorithmic change.

Changing from a higher level to a very specialized level: There is one unfair criticism against DL, and that is it's higher level representation, which we can call black box. DL gives exceptional outcomes but because of higher level representation or black box kind of approach we know extremely few things regarding how such outcomes are resulted inside. In biomedical areas, it is quite difficult to produce accurate solutions because of several studies that are mainly related to human health. So, DL methodologies have to adopt specialized level (white box) learning instead of higher level learning.

Thus selection of an appropriate architecture and hyper parameters of Deep Learning: One should know the proper function of each architecture in DL for obtaining vigorous and consistent outputs. As per the input data types and research objectives of such DLAs, performance may vary. Yet several functionalities are not understood. When DLA is selected at that time, there are lots of hyper parameters that we have to take under consideration, for example, hyper parameters such as learning rates, the number of hidden layers, initial weight values, bias values, learning iterations, etc. Proper value selection of the hyper parameter makes a valuable change in the performance of the DLA.

Hybrid and multimodality DL: Hybrid architecture we can generally know as a combination of more than one DLA. The biomedical fields offer different types of data modalities, as described in the above sections. But if we consider biomedical image modality, they offer lots of formats such as X-ray, MRI, PET, CT etc. So with the use of such modalities, we get better solutions to the problems. Some of the researchers, such as Suk et al. and Soleymani et al. [11], have proposed different multimodalities for Alzheimer's disease classification and emotion detection, respectively.

Speeding up these DLAs: In this era of big data, any application-related database becomes larger and larger. To analyze these kind of data using DL architectures require large amount of time. To get faster result and speed up the performance of the learning we can use GPUs.

7 CONCLUSION

Over the last few years, these DLAs have found a vital spot in the direction of the automation or computerization of our everyday life, delivering tremendous improvements by comparing these conventional machine learning algorithms (MLAs). By looking toward the contribution of DL in today's era, researchers are thinking that by the next 10–15 years, DL will give full automation to humans for their daily activities. Utilization of DL benefits in healthcare, especially in the medical image domain, is growing faster and faster. First, we have stated the basic requirements of the DL algorithm over machine learning architectures. After that we have detailed understanding of all architectures along with limitations and advantages. We have also stated the implementation environments for evaluating the performance. In the final section, we highlighted state-of-the-art applications of biomedical imaging analysis using DL.

Therefore, several research organizations are mainly working on DL, based on the efficient solution that mainly encourages using these techniques on biomedical imagining. By analyzing machine learning or deep learning effects on the real world, human will soon be replaced in almost all medical-related applications. However, we should not only consider the solution but also several challenges that reduce the growth of medical data. One of the greatest challenges is the unavailability of such an annotated dataset. Generally Deep Learning methods provides positive feedback but if we are are considering healthcare data for analysis it is also a challenging task for DL methods because healthcare data required higher sensitive values.

REFERENCES

[1] J.D. Miller, Learning IBM Watson Analytics, Packt Publication Ltd, Livery Place, 2016, pp. 1–25. ISBN 978-1-78588-077-3.
[2] D.M. Health, Google DeepMind, https://www.deepmind.com/health, 2016.
[3] P. Baldi, G. Pollastri, The principled design of large-scale recursive neural network architectures-dag-RNNs and the protein structure prediction problem. J. Mach. Learn. Res. 4 (2003) 575–602.
[4] O. Denas, J. Taylor, Deep modeling of gene expression regulation in an Erythropoiesis model, International Conference on Machine Learning workshop on Representation Learning, Atlanta, Georgia, USA, 2013.
[5] S. Lee, M. Choi, H-S. Choi, et al. FingerNet: deep learning-based robust finger joint detection from radiographs. In: Biomedical Circuits and Systems Conference (BioCAS), 2015 IEEE. 2015. p. 1–4. IEEE.
[6] P. Mirowski, D. Madhavan, Y. LeCun, et al., Classification of patterns of EEG synchronization for seizure. Int. Fed. Clin. Neurophysiol. 120 (2009) 1927–1940, Published by Elsevier Ireland Ltd. https://doi.org/10.1016/j.clinph.2009.09.002.
[7] S. Min, B. Lee, S. Yoon. "Deep learning in bioinformatics." arxiv preprint arxiv:1603.06430 (2016).

[8] S. Malik, K. Khatter, Malicious application detection and classification system for android mobiles, Int. J. Ambient Comput. Intell. 9 (1) (2018) 95–114.
[9] L. Saba, N. Dey, A.S. Ashour, S. Samanta, S.S. Nath, S. Chakraborty, et al., Automated stratification of liver disease in ultrasound: an online accurate feature classification paradigm, Comput. methods Prog. Biomed. 130 (2016) 118–134.
[10] K. Sharma, J. Virmani, A decision support system for classification of normal and medical renal disease using ultrasound images: a decision support system for medical renal diseases, Int. J. Ambient Comput. Intell. 8 (2) (2017) 52–69.
[11] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.
[12] H. Li, Q. Lyu, J. Cheng, A template-based protein structure reconstruction method using deep autoencoder learning. J. Proteomics Bioinform. 9 (2016) 306–313, https://doi.org/10.4172/jpb.1000419.
[13] B.J. Erickson, P. Korfiatis, Z. Akkus, T. Kline, K. Philbrick. "Toolkits and libraries for deep learning", Published online: 17 March 2017
[14] https://github.com/thedataincubator/datascience/blogs/blob/master/output/DL_libraries_final_Rankings.csv.
[15] A.-R. Ali, "Deep learning applications in medical imaging", September 14, 2017.
[16] Z. Li, N. Dey, A.S. Ashour, L. Cao, Y. Wang, D. Wang, et al., Convolutional neural network based clustering and manifold learning method for diabetic plantar pressure imaging dataset, J. Med. Imaging Health Inform. 7 (3) (2017) 639–652.
[17] M. Pandya, S. Jardosh, Applications and challenges of deep learning in the field of bioinformatics, Int. J. Comput. Sci. Inform. Secur. 15 (7) (2017).
[18] T. Brosch, Y. Yoo, L.Y.W. Tang, D.K.B. Li, A. Traboulsee, R. Tam, Deep convolutional encoder networks for multiple sclerosis lesion segmentation, in: N. Navab et al., (Ed.), MICCAI 2015, Part III, 2015, pp. 3–11. LNCS 9351.
[19] M.S. Iqbal, D.L. Silver, A scalable unsupervised deep multimodal learning system, Proceedings of the Twenty-Ninth International Florida Artificial Intelligence Research Society Conference, 2016.
[20] S. Ghosh "Challenges in deep learning for multimodal applications", Proceedings of the 2015 ACM on International Conference on Multimodal Interaction, Pages 611–615, University of Southern California, Los Angeles, CA.

FURTHER READING

[21] S.S. Ahmed, N. Dey, A.S. Ashour, D. Sifaki-Pistolla, D. Bălas-Timar, V.E. Balas, J.M.R. Tavares, Effect of fuzzy partitioning in Crohn's disease classification: a neuro-fuzzy-based approach, Med. Biol. Eng. Comput. 55 (1) (2017) 101–115.
[22] N. Dey, A.S. Ashour, A.S. Althoupety, Thermal imaging in medical science, in: Recent Advances in Applied Thermal Imaging for Industrial Applications, IGI Global-United States of America, 2017, pp. 87–117.
[23] N. Dey, A.S. Ashour (Eds.), Classification and Clustering in Biomedical Signal Processing, first ed., IGI Global-United States of America, 2016 (April 7, 2016).
[24] O. Azzabi, C.B. Njima, H. Messaoud, New approach of diagnosis by timed automata, Int. J. Ambient Comput. Intell. 8 (3) (2017) 76–93.
[25] M.Y. Khachane, Organ-based medical image classification using support vector machine, Int. J. Synth. Emot. 8 (1) (2017) 18–30.