Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 5548884, 15 pages
https://doi.org/10.1155/2021/5548884

Review Article

The Understanding of Deep Learning: A Comprehensive Review

Ranjan Kumar Mishra,[1] **G. Y. Sandesh Reddy**,[2] **and Himanshu Pathak**[3]

[1]*Department of Mechanical Engineering, Shiv Nadar University, Gautam Buddha Nagar, Greater Noida, India*
[2]*Department of Mechanical Engineering, NIT Trichy, Trichy, India*
[3]*School of Engineering, IIT Mandi, North Campus, VPO Kamand, Mandi, Himachal Pradesh, India*

Correspondence should be addressed to Ranjan Kumar Mishra; rm920@snu.edu.in

Received 8 February 2021; Revised 19 March 2021; Accepted 24 March 2021; Published 5 April 2021

Academic Editor: Ali Ahmadian

Deep learning is a computer-based modeling approach, which is made up of many processing layers that are used to understand the representation of data with several levels of abstraction. This review paper presents the state of the art in deep learning to highlight the major challenges and contributions in computer vision. This work mainly gives an overview of the current understanding of deep learning and their approaches in solving traditional artificial intelligence problems. These computational models enhanced its application in object detection, visual object recognition, speech recognition, face recognition, vision for driverless cars, virtual assistants, and many other fields such as genomics and drug discovery. Finally, this paper also showcases the current developments and challenges in training deep neural network.

1. Introduction

There are various tasks like multiplying big numbers or search operations, which are difficult for human beings to perform but are easy to perform on computers, and there are certain tasks like driving or language conversations that are tough to be performed on the machines. Machine learning is used to make the computers perform the tasks which can be done better by the human beings. Machine learning is the use of computer algorithms, which enables the machine to learn to access the data automatically with an improved experience. It has made life easy and has become an essential tool in many sectors like agriculture, banking, optimization, robotics, structural health monitoring, etc. It can be used in devices like cameras for object recognition; image, color; and pattern recognition; data collection; data sorting; and translation of speech to text. One of the machine learning approaches that is dominant in these fields of applications is deep learning.

Machine learning works similar to that of a new born baby. There are billions of neurons in the brain, which are interconnected and are activated when a message is passed to the brain. For example, when a baby is shown a car, a certain set of neurons are activated, when the baby is shown another car of a different model, the same set of neurons with some additional neurons might be activated, and when the baby is shown a lion, a different set of neurons will be activated. So, human beings are trained in their childhood and they learn things, and during this process the neurons and the paths connecting them are adjusted. Machine learning also works like this, where the machine is trained by many examples that are present in the training data sets and the neural networks are trained and adjust the paths followed. Then, the machine is given a new set of inputs and gets outputs. Some of the real-life examples where this technique is used are spam filters in Gmail, yahoo, and true caller app that filter the spam mails, and amazon ALEXA and the recommended videos that come in our YouTube homepage according to the type of videos that we watched earlier. Some companies like TESLA, Apple, and Nissan are working on driverless technology based on deep learning.

Deep learning is one of the techniques used in machine learning. Deep learning works on the principle of extracting features from the raw data by using multiple layers for identifying different aspects relevant to input data. Deep learning techniques include convolutional network,

recurrent neural network, and deep neural network. Deep learning uses artificial neural network, especially the convolutional network. In the past, machine learning use was limited due to its inability to process the raw input data. Deep learning has helped in overcoming this limitation, as it has the ability to operate on large volumes of data and thus has been an effective and useful technique of machine learning. Deep learning has also picked up the pace due to the hardware advancements of computers. A deep experience in feature extraction was necessary to convert the raw data into a suitable form so that the subsystem of the machine can recognize and classify the raw data.

Machine learning performance greatly depends on data representation. Due to this, much of the time is consumed in preprocessing design, which makes the algorithms labor intensive. Representation learning is used to extract only useful information during data classification by learning data representation. Feature learning has replaced manual involvement for data representation and allows the machine to discover the representations automatically to make it useful for classification. Representation learning has found its application in both academic as well as industrial fields. Representation learning has been used for speech recognition, language processing [1], and object recognition. Microsoft created MAVIS in 2012, which is speech-based feature learning. Deep learning is also used in Deep architectures that give the advantage of reusing features and can also capture higher number of input configurations. Deep learning uses representation learning by creating multiple layers of representations. This creates a path connecting the input and output node, and the depth of the path influences the representation learning. It indicates the number of ways in which the different paths can be used. For example, in image recognition, different layers represent different aspects like edges, surface, letters, or faces. Here, each neuron or node in the input layer works on a subtask and passes the result of the subtask to the hidden layers and then the hidden layers pass the results of the subsequent subtasks to the output layer to detect the face, which represents the final output. Thus, deep learning methods have reduced human dependence by generating different methods for extracting features.

Deep learning has provided a breakthrough in the field of Artificial intelligence, which had limitations in the past decades. Deep learning has provided methods to classify millions of images into lesser number of classes, thereby reducing the error percentage [1]. Scene labeling is used to label every pixel of the image to categorize the body into the class it belongs to [2]. Body poses of human beings can be estimated in the images by using different network combinations for modeling joint relationship [3]. Different architectures can be used to optimize the computational costs for deep learning methods for image recognition [4]. Deep neural networks are trained to effectively reduce the error in speech recognition tasks [5]. Neural networks have been implemented successfully in acoustic modeling and for reducing variations in speech signals [6, 7]. Deep neural networks evolved where effective algorithms and advancement in the computer technology have made them a dominant approach in machine learning and have proved to be effective for drug discoveries [8]. It has also been useful in identifying electron or particles in a filtering system and to solve physics problems [9, 10], neuron reconstruction in retina [11] for classifying the DNA variants, and improved prediction of the splicing patterns in tissues [12, 13]. Further advances have created a unified neural network that can be used for solving various tasks like answering questions from a vast range of topics, speech recognition, etc., rather than using separate networks for specific tasks, thus reducing the computational costs [14, 15]. They have also been used for improving the performance of language translation having very large vocabulary with optimized complexity [16, 17]. Mammone et al. [18] used electroencephalographic [EFG] to investigate the motor planning activity for decoding motor preparation phases. They successfully obtained results in the time-frequency domain by implementing deep learning phenomena. Yang et al. [19] briefed about the recent developments on signal processing and machine learning (SPML) research. Morabito et al. [20] implemented the deep learning concept based on convolutional neural network (CNNs) approach and also on autoencoder multilayered feedforward perceptron for classifying Alzheimer's disease patients from mild cognitive impairment and healthy controls subjects. Li et al. [21] recently used a novel neural network for wideband spectrograms to detect signals and classify signal types. Figure 1 illustrates the block diagram for the basic structure of the convolutional neural network (CNN) model where convolutional layer and max pooling layers are the initial phases that are chased by a fully connected layer as the linear discriminator. Deep learning methods use multiple layers for extracting features from raw data and do not require human participation. The paper is organized as follows: an overview of supervised learning and its developments are described in Section 2. In Section 3, the technique of backpropagation is shown for training multilayer networks. Convolutional networks are explained in Section 4, and image understanding with deep convolutional networks are presented in this section. In Section 6, an overview of deep representations and language processing is shown. Section 7 describes the recurrent neural networks. Finally, Section 8 concludes the paper, which summarizes the current understanding of deep learning and their challenges in computer vision.

2. Supervised Learning

Supervised learning is a method of learning a function to map input data which is in the form of a vector to the output data which is a supervisory signal. It produces a function that can be used for mapping other examples. Supervised learning first decides the type of data that should be used for different training examples. After gathering the data, the input and outputs are gathered from human experts and the input data are to be represented in the form of a function, and its accuracy depends on the way the input feature is represented so as to signify the object of interest. Then, the structure of the function and algorithm are determined and the algorithm is made to run on the gathered data, and the

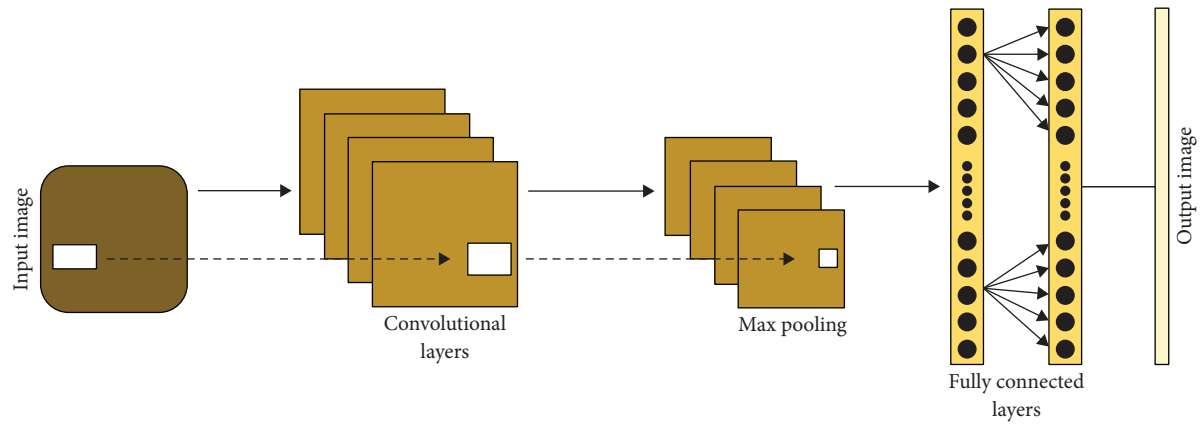

Figure 1: Block diagram of the basic structure of CNNs model (https://www.mdpi.com/2072-4292/12/9/1444/htm).

parameters are adjusted and the accuracy of the function is evaluated where it is made to run for different data or examples. Suppose we want to create a system that classifies the images of different objects with different categories like fan, tube light, human being, etc. A large number of images of the corresponding categories are collected and these images are shown to the machine, and it produces an output in the form of vector scores for each image of the particular category during the training. Here, a function is obtained that gives the error between the output vector scores obtained after training and the desired vector scores decided upon before training. Then, the machine optimizes these parameters and reduces its error. These optimized parameters are called weights and are used to define the learned function of the machine.

The algorithm computes a gradient vector to adjust the weights used. These gradients are useful for reducing the error between the vectors by changing the weight vector according to the gradient vector produced. Here, the functions are taken as a landscape and the negative gradient indicates the descent of the landscape and thus reduces the overall average.

Most of the users use Stochastic Gradient Descent (SGD) due to its better performance. Stochastic means randomness. Gradient indicates the slope and gradient descent means the decrease in the slope. The main objective of a gradient function is to find the value of the input variable or independent variable, which gives minimum value of the objective function. It is an iterative approach in which a point is selected first and then this point is updated for every step as the slope of the function decreases. Gradient descent algorithm is very slow when the number of iterations and data points are increased, thus making them limited for larger data sets. Therefore, stochastic gradient descent is used, which introduces randomness in the algorithm during selecting the data points. SGD reduces the number of computations by selecting any one data point at random during each iteration. The algorithm produces the output, errors, and gradients for the given input set and then adjusts the weights until the slope or the gradient descent stops decreasing. One can get the required accuracy of the training set without going through all the examples. So, a stochastic algorithm can find the averaged weights quickly as compared to the optimization techniques [22]. After achieving the accuracy, the performance of the algorithm is tested on the test set that has different examples. The performance on the test set indicates the machine's ability to give the answers for a new set of inputs that were not used during the training of the algorithm.

Linear classifiers are being used in machine learning applications because they are the fastest when it comes to the classification speed. Linear classifiers are used to classify the data into classes, which depend on the linear combination of the input feature values. These feature values are given as input to the machine in the form of vectors. The output obtained is in the form of a function of weight vector and the input features. The weights are adjusted during the training. Here, the function value becomes 1 when the product of the weight vector and the input feature crosses the threshold value, and the function value becomes zero for the other values of the dot product of the weight and input feature vectors. These classifiers separate the data by using hyperplane or a plane. For a two-class classification problem, a linear classifier splits the input space using a hyperplane where the points of one type are classified on one side of the hyperplane and the others on the other side of the hyperplane [23]. But, linear classifiers are sensitive to problems where the inputs vary like the variation in the position, poses, and accent of the speaker. For example, consider a Siberian tiger and a Bengal tiger, both of whom have very little variation in their features. If we consider any one breed at a minute level with a different background, then the images will be different but when the two different breeds are placed in the same pose and same background, the linear classifier cannot differentiate the images and may show that both of these animals or images belong to the same category or the same breed. Here, feature extractor becomes important to remove the confusion while judging the images that do not vary with respect to the poses or the background. Kernel methods can make the classifiers more powerful and can be used for solving nonlinear problems. Kernel methods are used for pattern analysis and classification [24]. They can transform raw data into implicit feature vector so that the feature space of the input data is operated without

computing the data coordinates in that space. Actually, it computes the product of the images of the pairs in the data set and reduces the computational time unlike the higher computation time obtained in explicit computation. Here, kernel method converts a linear model into a nonlinear model. One of the kernel methods is support vector machine, which uses the algorithm to classify the input; for example, they classify the tiger into the animal group and Maruti 800 into the car group. It converts a lower dimension problem into a higher dimension problem, and then solves it and gives the output. For example, it converts a 2D problem into a 3D problem, and the hyperplane used in a 2D problem to classify the data is converted into a plane, which classifies the data in the 3D problem. Another kernel method is the Gaussian process, which uses normal distribution for data classification. It follows the stochastic process, where the linear combinations of random variables are normally distributed. But, this method is sensitive to variations of the target. They require a large number of examples in the data set, and this number increases with the increase in the number of variations [25]. Here, feature extractor becomes important, as it can be designed conventionally or learned automatically by using deep learning.

Deep learning architecture is based on the working of the neural networks present in the human brain. Just like a cell acts as the basic building block of the human body, a neuron acts as the heart of the neural network. A neuron acts as a node which does mathematical operations on the input to give an output. Neurons are placed in multiple layers and are stacked together either in parallel or series combination that give the output after processing the input from each layer, and when a lot of these layers are stacked, it forms a deep neural network. The weights act as channels to transfer the data from one layer to another, then the subsequent neurons in the hidden layer will be associated with a number called bias and the bias is added with the sum of the weighted inputs and this sum is applied to the activation function, which tells whether the neuron gets activated or not. Then, the activated neurons pass the information to the next layer and this follows up to the final hidden layer and then the neuron activated in the output layer will represent the corresponding output. Training of neural network requires Graphical Processing Units (GPU) that are costly compared to the CPU due to the thousands of cores present in it. Deep learning architecture depends on the amount of data and the number of layers required to produce the output. Therefore, it can take months or hours depending on the data set.

3. Backpropagation Technique to Train Multilayer Networks

"Deep learning models are used to reduce the human intervention for extracting data because features designed by hand requires a lot of hands-on experience and skill in the domain." Therefore, a good feature extractor is required, which can be trained by the multiple layers used in deep neural networks. The Pandemonium model is used, which helps in recognizing a pattern that has not been used in the data set during the training. Perceptron is a system that works in a manner that is close to processing in a brain and hence depends on probabilistic results for their operations [26, 27]. Backpropagation procedures are very useful to increase the accuracy of prediction in the neural networks by finding the gradient descents. The idea of back-propagating errors was proposed by many groups way in back the 1980s [28–31]. Backpropagation is done to reduce the error by sending the error back to the input layer to update the weights, and an activation function like the sigmoidal function is used to get the results at the nodes of the hidden and the output layers. Actually, the inputs at the nodes of the hidden layers are written as the sum of the weighted inputs and bias. This sum obtained at the node in the hidden layer is applied to the activation function to get the output at that particular node in the hidden layer. Then, the same procedure is applied for the successive hidden layers and at last, the sum of the weighted inputs and the bias are fed to the output node and the output at that node is obtained after applying the activation function. In forward propagation, the output values obtained are compared with the target values and when they do not match, the error is calculated between the target values and the output values at each node in the output layer. In order to get an optimal value of the weight for which the error value is the least, the gradient descent is calculated by starting from the weights that have maximum error to the weights where the error is minimized, i.e., starting from the output layer to the input layer. The summation of the errors found at each node is calculated and these errors are backpropagated to update the new weights. The weights are differentiated partially with respect to errors to give the change in the weights, and this value is multiplied by the learning rate whose value lies between 0 and 1, and this product is subtracted from the previous weights to give the updated weights. The same procedure is repeated for the precedent hidden layers and weights and the precedent weights are updated. Again, the updated weights are used in forward propagation and the output is compared with the target values, and the errors are calculated. Again, backpropagation is done to update the weights, and this process goes on until the total error obtained at the output is minimized or equal to zero.

Let $\delta^{(l+1)}$ be the error term for the $(l+1)$-st layer in the network with a cost function $J(W, b; x, y)$ where (W, b) are the parameters and (x, y) are the training data and label pairs. If the l-th layer is densely connected to the $(l+1)$-st layer, then the error for the l-th layer is computed as

$$\delta^{(l)} = \left(\left(W^{(l)}\right)^T \delta^{(l+1)}\right).f'\left(z^{(l)}\right), \tag{1}$$

and the gradients are

$$\nabla_{W^{(l)}} J(W, b; x, y) = \left(\delta^{(l+1)}\left(a^{(l)}\right)^T\right), \tag{2}$$

$$\nabla_{b^{(l)}} J(W, b; x, y) = \delta^{(l+1)}. \tag{3}$$

If the l-th layer is a convolutional and subsampling layer, then the error is propagated through as

$$\delta_k^{(l)} = \text{upsample}\left(\left(W_k^{(l)}\right)^T \delta_k^{(l+1)}\right). f' z_k^{(l)}, \tag{4}$$

where k indexes the filter number and $f'(z_k^{(l)})$ is the derivative of the activation function. The upsample operation has to propagate the error through the pooling layer by calculating the error w.r.t to each unit incoming to the pooling layer. For example, if we have mean pooling, then upsample simply uniformly distributes the error for a single pooling unit among the units which feed into it in the previous layer. In max pooling, the unit which was chosen as the max receives all the error since very small changes in the input would perturb the result only through that unit. Finally, to calculate the gradient w.r.t to the filter maps, we rely on the border handling convolution operation again and flip the error matrix $\delta_k^{(l)}$ the same way we flip the filters in the convolutional layer.

$$\nabla_{W_k^{(l)}} J(W, b; x, y) = \sum_{i=1}^{m} \left(a_i^{(l)}\right) * \text{rot90}\left(\delta_k^{(l+1)}, 2\right), \tag{5}$$

$$\nabla_{b_k^{(l)}} J(W, b; x, y) = \sum_{a,b} \left(\delta_k^{(l+1)}\right)_{a,b}, \tag{6}$$

where $a^{(l)}$ is the input to the l-th layer, and $a^{(1)}$ is the input image. The operation $(a_i^{(l)}) * \delta_k^{(l+1)}$ is the "valid" convolution between i-th input in the l-th layer and the error w.r.t. the k-th filter.

In deep learning, depending on the input, signals are sent to the neuron and these signals are combined and sent to the activation functions, and that generates the output. Activation function adds nonlinear properties to the network. There are three types of activation functions such as linear, Heaviside, and sigmoidal function. Out of these, linear function is known for its simplicity where the bias is linearly combined with the weighted sum. Heaviside step functions are conditional functions that are dependent on the conditions and give an output either as 0 or 1 on comparing the value of the weighted sum with that of a threshold value. Heaviside functions have real-life applications. For example, when a hot object is touched by the hand, the skin receptors in the fingers sense if the temperature is safe for the skin or not, but as the temperature of that object increases or crosses a threshold value, the receptors will send a signal to the neurons and the neurons will produce an action to remove the hand from the object. So, when the temperature crosses the threshold value, the function generates output as 1 and as long as the temperature is below the threshold value, the output generated will be 0 to indicate safe temperature. Sigmoid function represents an S-shaped curve and has a nature similar to the $\tanh(x)$ function and is just rescaled and shifted up. Sigmoid function is solely dependent on the weighted function and the function is given by $1/(1 + e^{-v})$ where "v" is the weighted sum. The value of the sigmoid function lies between 0 and 1, and the active range of the weighted sum of values is between −5 and 5, and when any value of the weighted function beyond this range is given as an input, the sigmoid function will generate the value as either 0 or 1 only. As the value of the function lies between 0 and 1, these functions are used for probability prediction. Sigmoidal functions are used instead of linear functions because if the value of the weighted function goes beyond the active range, then the linear model will give the function value greater than 1 or less than 1 and can no longer be used to predict probability. The rate of change of the linear function cannot capture the exact results. For example, if the weight sum value changes from 0 to 1, there will be a spike in the value of probability, and this mapping can be captured by using a sigmoidal curve instead of a linear function, which gives poor approximation of the results. Nonlinear activation functions are required to calculate the nonlinear error gradient to perform the backpropagation strategy to study the nonlinear complex behavior. So, when data get more complicated, a nonlinear function becomes appropriate for classifying the data. Another such nonlinear function that has become popular is rectified linear unit (ReLU). During backpropagation, the gradients go on decreasing and vanish and the information is squeezed. Secondly, sigmoidal function has very slow convergence and is not zero-centered and all the gradients become positive, thus making the optimization harder. Tanh has a value between −1 and 1 and is thus zero-centric and are preferred over the sigmoids, but this function also suffers the vanishing gradient problem. ReLU removes these limitations. The ReLU function is given as $R(z) = \max(0, z)$. Its value becomes 0 when $z = 0$ and becomes linear with slope of 1 when z is greater than 0. When compared with tanh, ReLU showed a 6-fold improvement in the convergence for image net classification [1]. They do not require any unsupervised pretraining on supervised tasks for a large data set [32]. ReLU does not use expensive operations like sigmoid or tanh and hence learns faster and avoids the vanishing gradient problem. All deep learning networks use ReLU for the hidden layers and the output layers use softmax for classification [1–12].

Back in the 1990s, the researchers were working on small neural networks that produced least convergence; it was thought that backpropagation and neural networks were not useful for speech recognition and computer vision. They feared that the gradient descent would produce many poor local minima and that the error minimization would not reduce. Local minimum is harder to recover for larger neural networks and also leads to overfitting. But, recent results show that the local minima is not a problem anymore and the main problem arises due to the building up of the saddle points in high dimensions that generates errors around them and creates a delusion of existence of the local minimum. As the variables go on increasing, the saddle points also increase exponentially. For this second order, saddle-free Newton method is applied for better optimized performance that skips these saddle points [33, 34].

Deep learning has evolved since 2000. Earlier, the algorithms were limited to only few thousand layers and neurons, and it was difficult to work on the deep networks that would have billions of parameters and neurons just like the human brain has. But, with the advancement in the learning algorithms, this number increased to million parameters for recognizing handwritten digits [35]. The researcher used a unsupervised learning algorithm. Unlike supervised learning,

the data here are unlabeled in the training sets and are passed as an input. Unsupervised learning tries to understand the structure from the unlabeled data and extracts useful features from them. Clustering algorithm is one of the unsupervised learning techniques that analyze the unlabeled data, learn their structure, and cluster the data into groups that form distinct clusters, and the data can be labeled using this result [36]. Learning becomes very difficult when there are a large number of hidden layers and the approximations produced using variational methods are poor. A fine-tuned generative algorithm produced better results for handwritten numbers [36]. Unsupervised learning is also used in the autoencoder, which is a neural network. It encodes the input and produces a reconstructed image after decoding the compressed representation into a decompressed representation. It is useful for learning complex concepts. The objective of an autoencoder is to reduce the reconstruction loss. Autoencoders are data-specific and can be used only for the data for which they were trained and cannot be used for other images or data in the real world. Each layer of the autoencoder is pretrained [37]. The researcher created a pseudo code for each trained layer. Sparsity turns most neurons randomly and this forces the neural network to learn representations more accurately than the generalized training data. Learning of sparse features uses a model that has linear encoder and decoder and an energy code to minimize the energy and to adjust the parameters of encoder and decoder and makes sampling inexpensive and learning fast [38]. Principal component analysis (PCA) is used for reducing high-dimensional data into lower dimensional data, necessary for classification. An effective way for nonlinear dimensionality reduction is to initialize the weights that help in transforming high-dimensional data into low-dimensional code [39]. Unsupervised auto encoders have also been useful for pedestrian detection, which can avoid accidents and can revolutionize the future driverless cars [40]. The researcher used sparse coding algorithm and a predictor function, and each layer is trained in an unsupervised manner by using the input representations from the previous layers and the architecture is fine-tuned after training of all the layers is completed.

The first breakthrough for pretraining came after the advent of Graphics processing units (GPUs), which are about 10 to 15 times faster than the Central processing units (CPUs). GPUs are faster in processing compared to CPUs and are task-specific. Nowadays, GPUs play an important role in the gaming laptops; for example, a person having a higher-level processor with no graphic card and a person having a low-level processor with high-end graphic card. The latter will be having better gaming performance compared to the former. GPUs are specific purpose processors that handle only graphics by means of parallel computing. For example, a game like GTA SANADREAS has a number of different things like clouds, people, cars, buildings, etc., that can be seen. When the game is played, the view of the player changes as the player moves forward or backward based on the control keys, and the player is able to see different things with time. So, for such tasks, the processing needs to be very fast. The different things visible in the game do not depend on each other, so the GPU can do parallel processing in which each section can control different things at the same time and thus generate a complete view altogether. Due to the availability of dedicated GPUs, development in video rendering and image-processing activities has gained pace [41]. The rendering process can be done in blocks in which the image or video can be divided into small blocks and can be computed in parallel. With the use of GPU, one can reduce the time of processing in unsupervised learning from several weeks to few days and hence have created a revolution in neural network. The speech recognition system generally uses hidden Markov models (HMM) to build a sequence for the signals, but the feed forward networks are being used as an alternative to Gaussian mixture models by taking the window of feature vectors as an input [6, 42]. The processing capabilities of GPUs and effective algorithms have led to the development of powerful architecture that have produced promising results for Large Vocabulary Speech Recognition (LVSR) for phone recognition by effectively speeding up the learning process by 30-fold compared to a normal CPU [43]. By 2012, GPU allows a large number of examples during pretraining and updating the parameters effectively for building powerful algorithms in order to learn good representations of data in quest [44]. Unsupervised learning worked better for the smaller set of data and prevents overfitting. Pretraining became important only for training the machine for small data sets during unsupervised learning.

As the evolution in deep learning started, another network called convolutional neural network (CNN) started to gain dominance for their ability to train the data easily compared to other deep networks. These networks are used predominantly in image recognition by using the stacked convolutional layers to solve the translational invariance in the images. These networks have found their application in recognizing handwritten digits or characters in the documents like bank checks without involving much of preprocessing and the input in the form of images, thus improving the scaling properties and reducing the learning time [45, 46]. These networks came to the aid of deep learning when neural networks were not preferred much.

4. Convolutional Networks

Conv net is an artificial neural network (ANN), which is predominantly used for image analysis due to their ability to detect patterns and can be used for other applications like data analysis and classification. Conv net is based on the biological processing of the brain. The hidden layers in CNN are also called convolutional networks and hence the network gets the name. They treat data as special, unlike the other neural networks where each neuron in the conv net is connected to the adjacent or closer neurons and all the weights have the same value. The word convolutional means the filtering of the data in the network. Each convolutional layer has a number of filters that are used to detect the pattern. Here, the pattern can be the edges, corners, objects, and shapes like circles, squares. These filters are applied to a certain patch in the image. The data processed by the filter goes to a set of pooling layers where the patch recognized by the convolutional layer is

shortened to half and again this goes to the next set of convolutional and pooling layers where the patches in the preceding layers are acted upon by the filters and this process goes on. At the end of the network, the data go to the fully connected layer and from the multiple sets of output from the pooling layer the fully connected layer will produce the images with top probabilities and the image with the top probability is recognized as the patch or pattern. Due to the pattern recognition ability, the convolution network is used in almost all the applications like mobile phone, surveillance cameras, driverless cars, etc. CNN has two phases, a training phase and an inference phase. Training phase is one in which the network is trained by taking a lot of data as input and the filtered data sets are created by means of back-propagation and are used for the inference phase. On dwelling deep into the architecture of convolutional network, the convolutional layer acts as heart of the network. It has a set of filters which are applied to the input image and produce different activation features in the input image, and the inputs are in the form width x height x depth. The filters are the matrices of any specific order with the same depth like input image. Here, the activation filters are generated for the input image, and these sets of filters are then applied to the inputs for which an output is generated that has special dimensions as the same as that of the image and also the depth of it is equal to the number of activation filters. A stride is used to process the pixels in the image and is used for reducing the special dimensions of the output generated from each convolutional layer by a factor equal to the assigned stride value. A nonlinear activation function like ReLU is used in to add nonlinearity in the linear convolutional layer and reduces the problem of overfitting. Pooling layer is used to down sample the output sample obtained from the preceding convolutional network. Maxpooling is done to reduce the dimensionality of the image by reducing the number of pixels in the output from the convolutional layer. The maximum value of a certain region in the input convolutional network is stored in the output and then the region is shifted by a factor equal to the filter size and the maximum value in that region is stored in the output discarding the lower value pixels and this goes on and thus reduces the dimension of the input by a factor equal to stride value. By pooling the computational cost, the memory allocation is reduced. Finally, the data pass through the network and reach the final layer called the fully connected layer. The fully connected layer identifies the final output category and creates a set of output data and of these output sets, the top three output data are selected by using probability distribution algorithm and the output with the largest probability value is selected. Fully connected layers increase the load and computational cost and hence the recent networks are being made more convolution-driven networks to balance the architecture. In general, we have that

$$x_j^l = f\left(\sum_{i \in M_j} x_j^{l-1} * k_{ij}^l + b_j^l\right), \tag{7}$$

where x_j^l is the output of the current layer, x_i^{l-1} is the previous layer output, k_{ij}^l is the kernel for the present layer, and b_j^l are biases for the current layer. M_j represents a selection of input maps. For each output map, an additive bias b is given. However, the input maps will be convolved with distinct kernels to generate the corresponding output maps.

Conv networks are similar to the cerebral cells of an animal that respond to the light rays falling on the eyes. This study indicates different relationships between different areas like the visual system with the cortical cells in animals like monkeys that have advanced visual system compared to human beings [47, 48]. With the evolution of deep convolutional neural networks, the performance of neural networks produces results comparable to the IT cortex in visual processing [49]. Conv net has also been applied to neocognitron network used for pattern recognition, which uses unsupervised learning and recognizes patterns based on different shapes and are not sensitive to the position shifts or shape distortion [50]. Time delay neural networks have been used for recognition of phonemes and words and achieved higher recognition rate when compared to recognition techniques like HMM model [51, 52].

Convolutional networks have evolved a lot starting from the 1990s using networks for speech recognition, image recognition to document recognition of bank checks. Convolutional networks have been preferred over the fully connected networks for document verification tasks like handwriting recognition by expanding the training set used [53]. Conv nets have also been used for face recognition and detecting multiple faces in a single image and captures the faces in different poses and variance in scale or translation [54–56]. Figure 2 describes the working model of basic structure for convolutional neural network (CNN).

5. Image Understanding with Deep Convolutional Networks

Conv nets have been dominant in the applications of image analysis, and document and digits recognition. Recently, they are also being used in traffic signal recognition, which is even better than the natural human recognition system. Furthermore, with the implementation of GPUs, the preprocessing of data and recognition became faster [57]. They have been used in biological research to study the cell structure and embryo development in the body and image segmentation of brain images [58, 59]. They have been predominant and reliable in detecting different faces in the image under the influence of variation in poses, lightning, facial expressions, and rotations, as shown in Figure 3. With the improvements in conv nets architecture, the networks can also be trained for performing multiple tasks like detecting faces and estimating the poses and the joint location of a human body simultaneously for various ranges and produced more promising results compared to the ones which are trained to perform one task at a time [60–62]. Further advancements in the field of face recognition have opened ways for 3D modeling of face after training them on data sets having images of different identities [63].

Long-range, vision-based systems in the robots have helped in navigating them through the paths and detecting the obstacles with ease and are being used in the

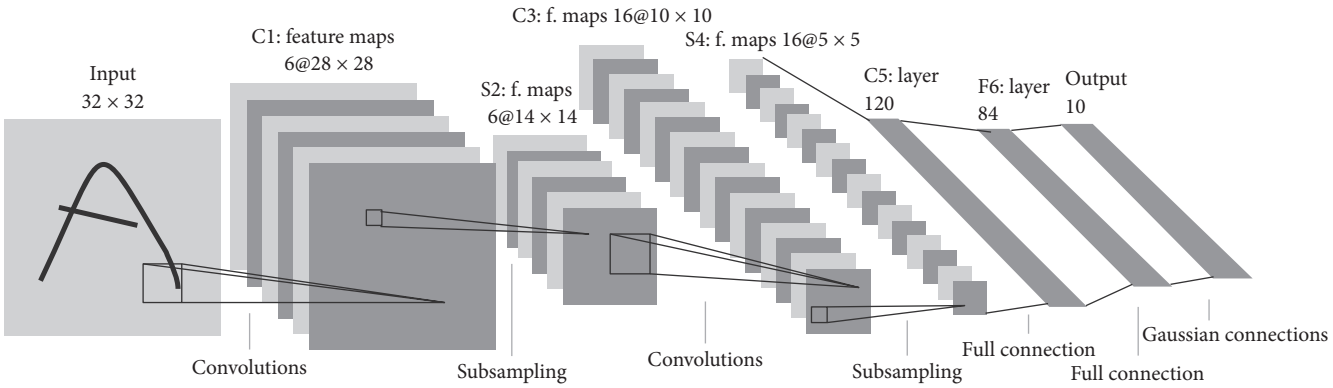

Figure 2: CNN architecture (https://towardsdatascience.com/understanding-cnn-convolutional-neural-network-69fd626ee7d4).

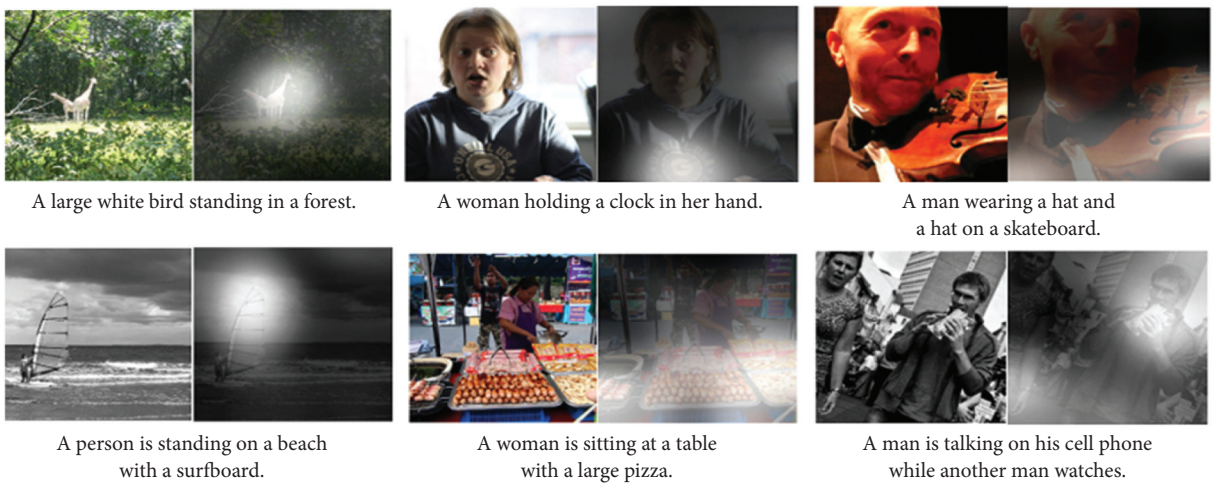

Figure 3: From image to text. Captions generated by a recurrent neural network (RNN) taking, as extra input, the representation extracted by a deep convolution neural network (CNN) from a test image, with the RNN trained to "translate" high-level representations of images into captions. When the RNN is given the ability to focus its attention on a different location in the input image (middle and bottom; the lighter patches were given more attention) as it generates each word (underlined), we found that it exploits this to achieve better 'translation' of images into captions (https://casmls.github.io/general/2016/10/16/attention_model.html).

development of autonomous cars for the future with minimal involvement of human beings [64, 65]. Companies like TESLA and NISSAN have started implementing the technology to produce first self-driving cars. These networks have also achieved breakthrough in applications of speech recognition and various language understanding using unsupervised training of the input data set [7, 14].

Until the invention of ImageNet classification, the machines were able to train quite a few image data sets with a rough estimation of around thousand sets, but with imageNet one can train some millions of images at higher resolution belonging to thousands of different categories [1]. Earlier the use of conv net was limited due to the heavy computational cost for recognizing higher resolution images and problems like overfitting. But, with advancement in the hardware technology, use of GPUs and advanced nonlinear activation functions like ReLU has made the training of the data set quite easier due to the use of lesser parameters and connections involved in the network. Techniques like dropout has been effective in reducing the overfitting problem in neural networks by using smaller weights and improves the performance of the network for various tasks like image, speech recognition, etc. [66] They have also been effective in detection and localization tasks by combining the input low-level features of images with the high-level features from object detectors by training the network for a large data set and modifying the previous networks by pretuning them for scarce data sets, which proved to be effective for classification purposes [67, 68]. Researchers have also played with the depth of the networks by varying the number of weight layers to study their effect on the accuracy in image recognition field [69].

Back in the 1990s, computers were not able to process data within a reasonable time period due to unavailability of large storage devices. Until 2010, neural network was not considered worthy; it gained recognition from 2010 onward when an annual ImageNet competition was held to decide on the performances of the algorithms presented by the competitors and their contributions toward the AI field. The use of activation function like ReLU in convolutional networks enabled the large networks to train the data several times faster compared to the activation functions like tanh and sigmoid. The GPUs

enabled the networks to train the data in parallel and reduced the computational time.

Technical giants like yelp, Pinterest, Facebook, Baidu Hubspot, IBM, and Salesforce have been using conv net for image classification, filtering out the spam content, generating human voices, natural language processing and to analyze customer relationship during ales process.

Rapid progress in neural networks demands powerful processors that can be used for a variety of applications. The networks used to solve the pattern classification problems accurately have thousands of complex parameters and connections, and this network needs to be embedded in a single chip. The network processors like VLSI can perform thousands of operations like multiplication and addition in parallel and has allowed to explore new algorithms [70].

6. Deep Representations and Language Processing

Sparse representations have only one bit on and are not used for many applications. Classic algorithms are sensitive to variations in target and proportionally a large number of training examples are used for different variations thus making them computationally costly. These representations are constructed specifically for words. The information in the word is distributed out in many directions. The in hot encoding involves the construction of the representation by fixing the vocabulary length consisting of common words, and this document is represented by a vector of dimensions equal to the number of common words that give position for every word and the value at that position gives the number of times the word appears in the document. Here, the word is represented by 1, which gives its coordinate in the document. But, this encoding is sensitive to the words that are antonyms and is insensitive to the meanings of the words and the way they are related to. Distributed representation assigns a vector for each word such that words with similar meanings will have a vector closer and the vector for the antonym will be far away from these vectors. Distributed representation is a representation for language that allows representing words based on their meanings; one can relate two words that belong to a similar category and can use for applications like question answering. Here, the meaning of the word is recognized based on the words that are in neighborhood with it in any sentence [25]. By means of probability distribution, the word's meaning is captured by seeing the number of occurrences in the vocabulary and the nearest words used in the context. The deep architecture has multiple layers consisting of neurons similar to the architecture of human brain, and parameters required for probability distributions can be obtained from the power of the combination of different products [71–73]. The depth of the architecture can be varied depending on the complexity of the problem and each layer here gets the number of inputs equal to the exponential of the input layers [74].

The main aim of natural language processing is to predict the next word in the sequence, and training with distributed representations helps the words to sense exponential number of semantic words in the vicinity [75]. The basic working behind this is that the words with some kind of similarly are clubbed together and the word which is semantic is given the value of 1 and the other words are given the value of 0, and the semantic words are mapped closer to each other and the rest are placed away from them. In language modeling, the neural network converts an input given in the form of a word to an image and can do the vice versa operation also, and depending upon the context and the number of times the word is used; the machine can predict the next word to be used in the sentence. The representations can be learned on a large data set necessary for deep learning. It takes a bunch of text and computes the statistics and generates the representations for the words found in the text and the output encodes the meaning of the word and looks at the nearest neighbors in the space. For example, the neighboring word of dog will give cats and the breeds of the dog. The matrix product of all words in vocabulary with weight of word of our interest will give the output that signifies higher probability of the word similar to the word of our interest. The sentimental analysis used in twitter and Facebook used for giving reactions and likes is one of the applications of the language process. It is also used for extracting information from any website. The words are predicted by the neural network without any human intervention and can translate a sentence in one language to a sentence of other language by maintaining the sequence [76–80]. There have been debates for logic-inspired paradigms and neural-network-inspired paradigms. The logic-inspired paradigms do not have an internal structure and uses symbols to tell whether they are identical or not and are bound by some rules of inference, but in neural networks large matrices and weights are used to perform calculations and get the output.

Natural language processing is not only used to make the sense of words but is also used to understand the words that denote sarcasm and also that denote a different meaning in the context. For example, a word can be used as a noun in a sentence and the same word can also be used as a verb in some other sentence by taking the context of the sentence. N grams serve the purpose by looking at the word that come before the target word and the word that comes after it and determines if the target word used is a noun or a verb in the context. N gram models predict nth word from $n-1$ words. The N here represents the number of words we are looking at. Representations like unigram, bigram, and trigram and up to n gram can be used for understanding the word form. Bi-grams represent two pairs of words that occur in a sentence and are used to train the machine to learn about the specific form of word and also to predict the next word from the previous words just like how the fill in the blank questions are solved. N- grams are used to capture the sarcasm of the text sentence by training the machine to look at the before and after words used adjacent to the target word and gives an opposite meaning of the sentence by seeing the presence of the negative word. These language models are used for spell correction and speech recognition by computing the probability of a sentence. These models are used in WhatsApp or messengers to suggest the next words that can be used to make sense or to replace an incorrect word by

a correct and meaningful word. In preprocessing of the neural networks for statistical processing of natural language, the *n* grams and stop words are eliminated. When any sentence is entered in Google search, the machine eliminates the articles, prepositions, and the words that share relations between the keywords and takes only keywords into account to produce the data.

7. Recurrent Neural Networks

Human beings can understand the words based on the previous words they read in any paragraph and are able to understand that paragraph based on the sequencing operations that run in their brain. Recurrent neural networks (RNNs) also work in the same manner. RNNs are the neural networks that are used for processing sequence data and are most commonly used for pattern recognition tasks. Here, the sequence data in the form of input is chopped into chunks and are fed to the RNNs. They use sequential memory which makes the networks easier to recognize the patterns that are used in speech translation, stock predictions, language translation, and image recognition. The architecture of RNN consists of input in the form of text which can have a number of features and hidden layers and an output with respect to time. For forward propagation, the RNNs will preprocess one word at a time and try to maintain the sequence of the words by adding the previous words in loop with the new words fed as input to the hidden layers, as shown in Figure 4. For example, an input sentence of four words is given to the network, then for the time 1 it will take the first word in the form of a vector to a particular hidden layer, which consists of some neurons, and these vectors are multiplied by the weights assigned to them. These sums are applied to the activation function like ReLU, and it gives the output for that hidden layer. The output obtained at that particular hidden layer is again given to the same hidden layer in the form of a loop and at time 2 the second word is fed to the same hidden layer as input and the same weights assigned for the first word will be assigned to the second word also. But, the output obtained for the first word in time 1 will be assigned different weights before being fed to the same hidden layer neurons in the form of loop for time 2. After this, the sum of the weighted inputs of the second word and the weighted output of the first word is given to the activation function to produce the output at time 2. Thus, the output of the second word will depend on the input of the second word as well as the output of the first word, and this allows the network to maintain the sequence up to the last word of the sentence. Similarly, again the sum of the weighted input of the third word and the output of the second word is fed to the activation function to produce the output for the third word for time 3 and this processing goes on until the final word is reached. The output obtained after the processing of the fourth word will be assigned a new weight and will be fed to an activation function like softmax that will give the predicted value as the output. After the processing, the loss function is calculated between the obtained output value and the model predicted value, and is minimized. This type of architecture of RNNs is used in outputs of Google and chat box output based on the input given by the user in the form of the statement.

After the calculation of the loss function, the weights are updated during the backpropagation. Here, the derivative of loss with respect to the output is calculated and is used for updating the weights. The derivative of loss with respect to the weight is calculated by using the chain rule and the updated value of the weight at the output of the last word is obtained by subtracting this derivative with the weight and the preceding weights are updated in a similar manner. Again, through forward propagation, the loss function is calculated and the weights are updated with respect to time through the backpropagation procedure and these iterations repeat until the loss function becomes minimum and get the global minima, and at this point the training of the network stops.

But, RNNs also have some problems, such as vanishing and exploding gradient. During the backpropagation, the derivative of the activation functions like sigmoidal functions lie between 0 and 1, so when the chain rule is applied to this derivative while moving toward the weights assigned to the first function, the derivative value becomes a negligible value due to the continuous derivation. This in turn would update the weights negligibly and will not allow the solution to converge to the global minima and creates a vanishing gradient problem. When the sigmoidal activation function is replaced by ReLU, the derivative of the activation function will give a value greater than 1 and this will change the value of the weights by a large amount and will not allow the solution to reach global minima. Such a problem is referred to as the exploding gradient problem.

RNNs in spite of being powerful are tough to train due to the exploding and vanishing gradient problem that does not allow the solution to converge [81, 82] and were limited to machine learning applications. Keeping this in mind, new advancements were done in the architecture of simple RNNs and an efficient method called long short-term memory (LSTM) RNN [83, 84] was introduced to solve the vanishing gradient problem. Also, new methods of training these networks [85, 86] have made the training of RNNs easier and improved the performance of the architecture. Such type of architecture has become dominant for applications like noisy pattern recognition and natural language modeling tasks to predict the next word or a character in the text document [79, 87]. RNNs are also used for complex applications like translating the languages that include long sentences. They have also been used for representations that are not varied from active or passive voices [17]. For translating from English to French, modern RNNs use an encoder decoder pair, which represents two RNNs. Here, the encoder converts the words into a length vector and the decoder again converts the vector to a variable length target to get the target sequence, and the simultaneous operation of the two RNNs maximize the probability of the target sequence [76, 80]. These RNNs generate French sentences in terms of conditional probability based on the given input English statements. RNNs are used for semantic cognition tasks, which are similar to the ones implemented in the brain [88, 89].

Apart from these applications, they are used for spam classifier to predict if the statement is true or not. Google

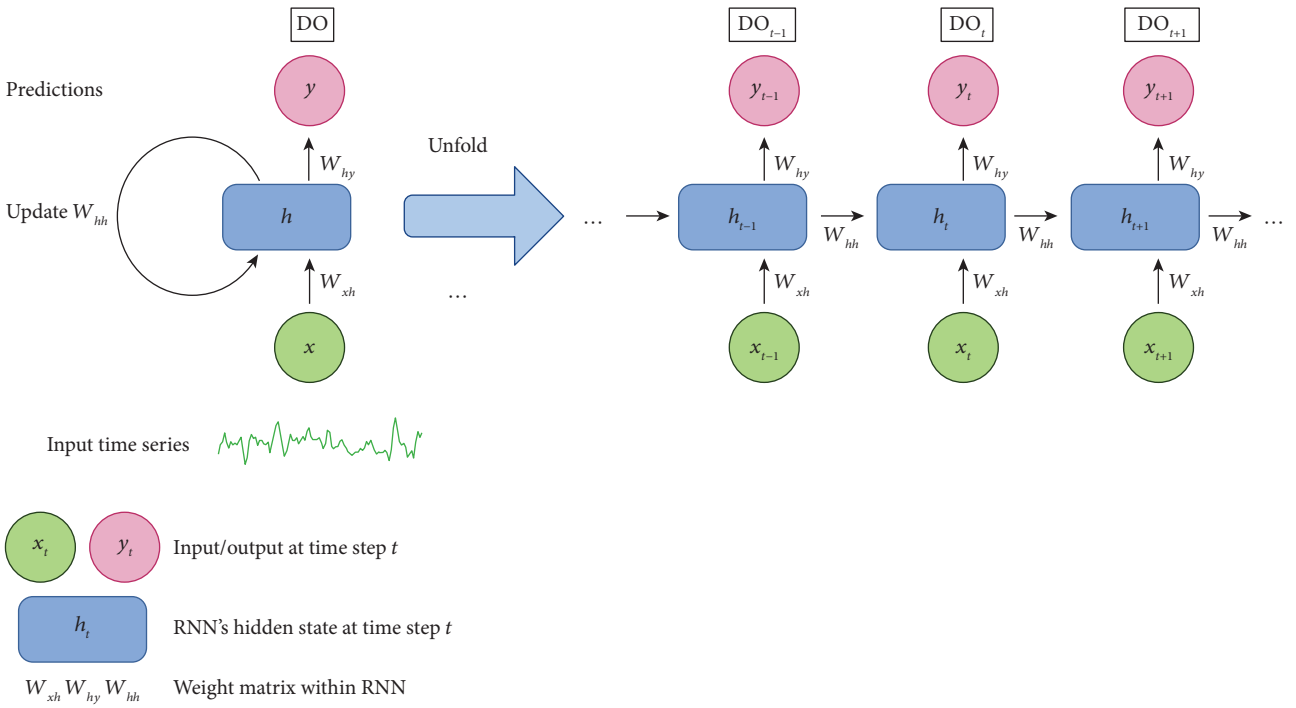

Figure 4: A recurrent neural network and the unfolding in time of the computation involved in its forward computation. The artificial neurons (for example, hidden units grouped under node h with values h_t at time t) get inputs from other neurons at previous time steps (this is represented with the black square, representing a delay of one-time step, on the left). In this way, a recurrent neural network can map an input sequence with elements x_t into an output sequence with elements o_t, with each o_t depending on all the previous x'_t (for $t' \leq t$). The same parameters (matrices W_{xh}, W_{hy}, W_{hh}) are used at each time step. Many other architectures are possible, including a variant in which the network can generate a sequence of outputs (for example, words), each of which is used as inputs for the next time step. The backpropagation algorithm can be directly applied to the computational graph of the unfolded network on the right, to compute the derivative of a total error (for example, the log-probability of generating the right sequence of outputs) with respect to all the states h_t and all the parameters (https://www.mdpi.com/2073-4441/12/2/585/htm).

image search also works on RNN, where the word we search for gets converted into an image and produces the required image as the output. Image captioning is another application of RNN. An image is given to the system as an input and the RNN converts it into a sentence. It can determine the objects in the image and express their relations [90]. Recent architectures with combination of CNNs and RNNs have been used to convert the sentences into image representation and again decode back to the language sentences. RNNs use same weights for all hidden layers. Although the RNNs are used widely for recognition or prediction problems, they face difficulties while training due to the use of gradient descent algorithm due to long-term dependencies [82]. In such cases, short-term dependencies are used to get better converged solutions and LSTM comes into play.

The LSTM architecture consists of memory cell, forget gate, input gate, and output gate. Memory cell is used to remember and forget the input depending on the context of the input. When the context of the statement changes, the memory cell remembers some of the previous information and should also be able to add some new information to it [83]. The first operation done in memory cell is point-wise operation where the output of the first word in the hidden layer after time 1 in the form of a vector of some dimension is multiplied with the input of the second word at time 2, which is also in the form of a vector with different entries but of the same dimension and gives an output for the second word of the same vector dimension. If the output vector has any zero entry, then that particular information is forgotten due to the change in context and some information will be added. In the forget gate, the weights of the first output are concatenated with the weight of the second input and this generalized weighted input is added to the bias and this sum is passed through the sigmoid activation function and gives an output vector. The point-wise operation adds information to the memory cell. Tanh activation function is also used to add any new information when the context changes and produces an output vector from the sum of generalized weighted input and bias and gives output in the form of vector having entries between −1 and +1. The point-wise operation is performed to get new information and adds this to the dissimilar vector obtained from the sigmoid function and this output is added to the memory cell. So, here the entries in the output nearer to 1 will be only passes on to the memory cell and the rest is skipped. These operations are done in the input layer. In the output layer, again the concatenation of the information is passed to a sigmoid function and it is combined with the information from the memory cell after it passes through the tanh activation function with the point-wise operation and will retrieve the

information having meaningful context and pass it to the next cell, and this process goes on. LSTM removes the problem of long-term dependency on the derivatives taken in simple RNNs and introduces temporal dependencies and can remember some of the information from the past information and the last layer of the output can have all information from the first layer input itself. The combination of LSTM RNN architecture and the effective training methods has proved to be very effective for handwriting recognition and speech recognition [91].

Recent advancements in RNNs have produced systems similar to a Turing machine that uses a combination of biological memory and computer design and have been used in applications like copying, writing, or reading [92], and the powerful memory networks for question answers tasks and their application can be extended to other domains like vision sensing [93].

Turing machines consist of a tape structure, which has a number of cells, and in each cell an alphabet also called a tape is placed. It is defined by seven tuples. A pointer called read-write head points at a particular cell at a time and is flexible to move in right or left directions by giving the notations, and this movement is controlled by a controller. The machine looks at the tape one cell at a time, and it gives the code in the form of a question or a problem to be solved. Attentive neural Turing machines are used for cold start knowledge-tracing problems where little data are available to train the network and predict the output accurately by using an external memory bank to store the useful information learned through optimized iterations. ANTM are outperforming LSTM networks in tasks involving predictions by producing relatively higher accuracy [94, 95].

8. Future and Drawbacks of Deep Learning

Purely supervised learning had successively come about through a system in spite of the catalytic effects of unsupervised learning. But, there has been a massive increase in the importance of unsupervised learning for a long period of time. Animals and human beings are mostly subjected to unsupervised learning because the observation is done for discovering the structure of the world rather than telling the names of objects. For the active processes of human vision, the optical arrays are sampled in tasks specific and intelligent ways using a smaller, high resolutions forvea with a big, lower resolutions surroundings. By combining with recurrent neural networks, the reinforcement learning is used for making decision where to see and is trained end to end. In this paper, a comprehensive state-of-the-art review is accomplished in the current scenario. Although deep learning and simple reasoning have been used for speech and handwriting recognition for a long time, new paradigms are needed to replace rule-based manipulation of symbolic expressions by operations on large vectors. The following are the drawbacks of deep learning:

(a) Large amount of data are required for performing better than any other technique

(b) Due to complex data modules, it is very expensive to train

(c) Deep learning needs hundreds of machines and expensive GPUs for processing data and thus it is costly

(d) It requires classifier for comprehending the output based on mere learning

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proceedings of the Advances in Neural Information Processing Systems*, vol. 25, pp. 1090–1098, Lake Tahoe, NV, USA, December 2012.

[2] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1915–1929, 2013.

[3] J. Tompson, A. Jain, Y. LeCun, and C. Bregler, "Joint training of a convolutional network and a graphical model for human pose estimation," in *Proceedings of the Advances in Neural Information Processing Systems*, vol. 27, pp. 1799–1807, Montreal, Canada, December 2014.

[4] C. Szegedy, W. Liu, Y. Jia et al., "Going deeper with convolutions," 2014, http://arxiv.org/abs/1409.4842.

[5] T. Mikolov, A. Deoras, D. Povey, L. Burget, and J. Cernocky, "Strategies for training large scale neural network language models," in *Proceedings of the Automatic Speech Recognition and Understanding*, pp. 196–201, Waikoloa, HI, USA, December 2011.

[6] G. Hinton, L. Deng, D. Yu et al., "Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[7] T. Sainath, A.-R. Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep convolutional neural networks for LVCSR," in *Proceedings of the Acoustics, Speech and Signal Processing*, pp. 8614–8618, Vancouver, Canada, May 2013.

[8] J. Ma, R. P. Sheridan, A. Liaw, G. E. Dahl, and V. Svetnik, "Deep neural nets as a method for quantitative structure-activity relationships," *Journal of Chemical Information and Modeling*, vol. 55, no. 2, pp. 263–274, 2015.

[9] T. Ciodaro, D. Deva, J. de Seixas, and D. Damazio, "Online particle detection with neural networks based on topological calorimetry information," *Journal of Physics: Conference Series*, vol. 368, Article ID 012030, 2012.

[10] K. Higgs, *Boson Machine Learning Challenge*, Kaggle, San Francisco, CA, USA, 2014, https://www.kaggle.com/c/higgs-boson.

[11] M. Helmstaedter, K. L. Briggman, S. C. Turaga, V. Jain, H. S. Seung, and W. Denk, "Connectomic reconstruction of the inner plexiform layer in the mouse retina," *Nature*, vol. 500, no. 7461, pp. 168–174, 2013.

[12] M. K. K. Leung, H. Y. Xiong, L. J. Lee, and B. J. Frey, "Deep learning of the tissue-regulated splicing code," *Bioinformatics*, vol. 30, no. 12, pp. i121–i129, 2014.

[13] H. Y. Xiong, "The human splicing code reveals new insights into the genetic determinants of disease," *Science*, vol. 347, p. 6218, 2015.

[14] R. Collobert, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, pp. 2493–2537, 2011.

[15] A. Bordes, S. Chopra, and J. Weston, "Question answering with subgraph embeddings," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, October 2014.

[16] S. Jean, K. Cho, R. Memisevic, and Y. Bengio, "On using very large target vocabulary for neural machine translation," in *Proceedings of the ACL-IJCNLP*, Beijing, China, July 2015.

[17] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proceedings of the Advances in Neural Information Processing Systems*, vol. 27, pp. 3104–3112, Montreal, Canada, December 2014.

[18] N. Mammone, C. Ieracitano, and F. C. Morabito, "A deep CNN approach to decode motor preparation of upper limbs from time-frequency maps of EEG signals at source level," *Neural Networks*, vol. 124, pp. 357–372, 2020.

[19] G. Yang, S. Q. Cao, and Y. Wu, "Recent advancements in signal processing and machine learning," *Mathematical Problems in Engineering*, vol. 2014, Article ID 549024, 4 pages, 2014.

[20] F. C. Morabito, M. Campolo, C. Ieracitano et al., "Deep convolutional neural networks for classification of mild cognitive impaired and Alzheimer's disease patients from scalp EEG recordings," in *Proceedings of the 2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a Better Tomorrow (RTSI)*, pp. 1–6, IEEE, Piscataway, NJ, USA, September 2016.

[21] W. Li, K. Wang, and L. You, "A deep convolutional network for multitype signal detection and classification in spectrogram," *Mathematical Problems in Engineering*, vol. 2020, Article ID 9797302, 6 pages, 2020.

[22] L. Bottou and O. Bousquet, "The tradeoffs of large-scale learning," in *Proceedings of the Advances in Neural Information Processing Systems*, vol. 20, pp. 161–168, Vancouver, Canada, December 2007.

[23] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, Wiley, Hoboken, NJ, USA, 1973.

[24] B. Schölkopf and A. Smola, *Learning with Kernels*, MIT Press, Cambridge, MA, USA, 2002.

[25] Y. Bengio, O. Delalleau, and N. Le Roux, "The curse of highly variable functions for local kernel machines," in *Proceedings of the Advances in Neural Information Processing Systems*, vol. 18, pp. 107–114, Vancouver, Canada, December 2005.

[26] O. G. Selfridge, "Pandemonium: a paradigm for learning in mechanisation of thought processes," in *Proceedings of the Symposium on Mechanisation of Thought Processes*, pp. 513–526, November 1958.

[27] F. Rosenblatt, "The perceptron—a perceiving and recognizing automaton," Technical Report, Cornell Aeronautical Laboratory, Buffalo, NY, USA, 1957.

[28] P. Werbos, *Beyond regression: new tools for prediction and analysis in the behavioral sciences*, PhD thesis, Harvard University, Cambridge, MA, USA, 1974.

[29] D. B. Parker, *Learning Logic Report TR–47*, MIT Press, Cambridge, MA, USA, 1985.

[30] Y. LeCun, "Une procédure d'apprentissage pour Réseau à seuil assymétrique in Cognitiva 85: a la Frontière de l'Intelligence Artificielle, des Sciences de la Connaissance et des Neurosciences," *Support-Vector Networks*, pp. 599–604, 1985, in French.

[31] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[32] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, pp. 315–323, Ft. Lauderdale, FL, USA, April 2011.

[33] Y. Dauphin, "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization," in *Proceedings of the Advances in Neural Information Processing Systems*, vol. 27, pp. 2933–2941, Montreal, Canada, December 2014.

[34] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun, "The loss surface of multilayer networks," in *Proceedings of the Conference on AI and Statistics*, Reykjavic, Iceland, April 2014.

[35] G. E. Hinton, "What kind of graphical model is the brain?" in *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pp. 1765–1775, San Francisco; CA, USA, July 2005.

[36] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[37] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Proceedings of the Advances in Neural Information Processing Systems*, vol. 19, pp. 153–160, Vancouver, Canada, December 2006.

[38] M. Ranzato, C. Poultney, S. Chopra, and Y. LeCun, "Efficient learning of sparse representations with an energy-based model," in *Proceedings of the Advances in Neural Information Processing Systems*, vol. 19, pp. 1137–1144, Vancouver, Canada, December 2006.

[39] G. E. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[40] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. LeCun, "Pedestrian detection with unsupervised multi-stage feature learning," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, Portland, OR, USA, June 2013.

[41] R. Raina, A. Madhavan, and A. Y. Ng, "Large-scale deep unsupervised learning using graphics processors," in *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 873–880, Montreal, Canada, June 2009.

[42] A.-R. Mohamed, G. E. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 14–22, 2012.

[43] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large vocabulary speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, pp. 33–42, 2012.

[44] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: a review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[45] Y. LeCun, "Handwritten digit recognition with a back-propagation network," in *Proceedings of the Advances in Neural Information Processing Systems*, pp. 396–404, Denver, CO, USA, November 1990.

[46] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[47] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of Physiology*, vol. 160, no. 1, pp. 106–154, 1962.

[48] D. J. Felleman and D. C. Van Essen, "Distributed hierarchical processing in the primate cerebral cortex," *Cerebral Cortex*, vol. 1, no. 1, pp. 1–47, 1991.

[49] C. F. Cadieu, "Deep neural networks rival the representation of primate it cortex for core visual object recognition," *PLoS Computational Biology*, vol. 10, Article ID e1003963, 2014.

[50] K. Fukushima and S. Miyake, "Neocognitron: a new algorithm for pattern recognition tolerant of deformations and shifts in position," *Pattern Recognition*, vol. 15, no. 6, pp. 455–469, 1982.

[51] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 3, pp. 328–339, 1989.

[52] L. Bottou, F. Fogelman-Soulié, P. Blanchet, and J. Lienard, "Experiments with time delay networks and dynamic time warping for speaker independent isolated digit recognition," in *Proceedings of the EuroSpeech*, vol. 89, pp. 537–540, Paris, France, September 1989.

[53] D. Simard, P. Y. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks," in *Proceedings of the Document Analysis and Recognition*, pp. 958–963, Edinburgh, Scotland, August 2003.

[54] R. Vaillant, C. Monrocq, and Y. LeCun, "Original approach for the localisation of objects in images," *IEE Proceedings—Vision, Image, and Signal Processing*, vol. 141, no. 4, pp. 245–250, 1994.

[55] S. Nowlan and J. Platt, "A convolutional neural network hand tracker," in *Proceedings of the Advances in Neural Information Processing Systems*, pp. 901–908, San Mateo, CA, USA, 1995.

[56] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: a convolutional neural-network approach," *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 98–113, 1997.

[57] D. Ciresan, U. Meier, J. Masci, and J. Schmidhuber, "Multi-column deep neural network for traffic sign classification," *Neural Networks*, vol. 32, pp. 333–338, 2012.

[58] F. Ning, D Delhomme, Y LeCun, F Piano, L Bottou, and P. E Barbano, "Toward automatic phenotyping of developing embryos from videos," *IEEE Transactions on Image Processing: A Publication of the IEEE Signal Processing Society*, vol. 14, no. 9, pp. 1360–1371, 2005.

[59] S. C. Turaga, J. F. Murray, V. Jain et al., "Convolutional networks can learn to generate affinity graphs for image segmentation," *Neural Computation*, vol. 22, no. 2, pp. 511–538, 2010.

[60] C. Garcia and M. Delakis, "Convolutional face finder: a neural architecture for fast and robust face detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 11, pp. 1408–1423, 2004.

[61] M. Osadchy, Y. LeCun, and M. Miller, "Synergistic face detection and pose estimation with energy-based models," *Journal of Machine Learning Research*, vol. 8, pp. 1197–1215, 2007.

[62] J. Tompson, R. R. Goroshin, A. Jain, Y. Y. LeCun, and C. C. Bregler, "Efficient object localization using convolutional networks," in *Proceedings of the Conference on Computer Vision and Pattern Recognition*, Columbus, OH, USA, June 2014.

[63] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: closing the gap to human-level performance in face verification," in *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pp. 1701–1708, Columbus, OH, USA, June 2014.

[64] R. Hadsell, P. Sermanet, J. Ben et al., "Learning long-range vision for autonomous off-road driving," *Journal of Field Robotics*, vol. 26, no. 2, pp. 120–144, 2009.

[65] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Scene parsing with multiscale feature learning, purity trees, and optimal covers," in *Proceedings of the International Conference on Machine Learning*, Edinburgh, Scotland, June 2012.

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

[67] P. Sermanet, "Overfeat: integrated recognition, localization and detection using convolutional networks," in *Proceedings of the International Conference on Learning Representations*, Banff, AB, Canada, April 2014.

[68] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pp. 580–587, Columbus, OH, USA, June 2014.

[69] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of the International Conference on Learning Representations*, Banff, AB, Canada, April 2014.

[70] B. E. Boser, E. Sackinger, J. Bromley, Y. Le Cun, and L. D. Jackel, "An analog neural network processor with programmable topology," *IEEE Journal of Solid-State Circuits*, vol. 26, no. 12, pp. 2017–2025, 1991.

[71] C. Farabet, "Large-scale FPGA-based convolutional networks," in *Scaling up Machine Learning: Parallel and Distributed Approaches*, R. Bekkerman, M. Bilenko, and J. Langford, Eds., pp. 399–419, Cambridge University Press, Cambridge, MA, USA, 2011.

[72] Y. Bengio, *Learning Deep Architectures for AI*, Universit´e de Montreal, Montreal, Canada, 2009.

[73] G. Montufar and J. Morton, "When does a mixture of products contain a product of mixtures?" *Journal of Discrete Mathematics*, vol. 29, pp. 321–347, 2014.

[74] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks," in *Proceedings of the Advances in Neural Information Processing Systems*, vol. 27, pp. 2924–2932, Montreal, Canada, December 2014.

[75] Y. Bengio, R. Ducharme, and P. Vincent, "A neural probabilistic language model," in *Proceedings of the Advances in Neural Information Processing Systems*, vol. 13, pp. 932–938, Vancouver, Canada, December 2001.

[76] K. Cho, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 1724–1734, Doha, Qatar, October 2014.

[77] H. Schwenk, "Continuous space language models," *Computer Speech & Language*, vol. 21, no. 3, pp. 492–518, 2007.

[78] R. Socher, C. C.-Y. Lin, C. Manning, and A. Y. Ng, "Parsing natural scenes and natural language with recursive neural networks," in *Proceedings of the International Conference on*

Machine Learning, pp. 129–136, Bellevue, WA, USA, June 2011.

[79] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the Advances in Neural Information Processing Systems*, vol. 26, pp. 3111–3119, Lake Tahoe, NV, USA, December 2013.

[80] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proceedings of the International Conference on Learning Representations*, San Diego, CA, USA, May 2015.

[81] S. Hochreiter, "Untersuchungen zu dynamischen neuronalen Netzen [in German]," Diploma thesis, T.U. Münich, Munich, Germany, 1991.

[82] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[83] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[84] S. ElHihi and Y. Bengio, "Hierarchical recurrent neural networks for long-term dependencies," in *Proceedings of the Advances in Neural Information Processing Systems*, vol. 8, Denver, CO, USA, November 1995.

[85] I. Sutskever, *Training recurrent neural networks*, PhD thesis, University of Toronto, Toronto, Canada, 2012.

[86] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proceedings of the 30th International Conference on Machine Learning*, pp. 1310–1318, Atlanta, GA, USA, June 2013.

[87] I. Sutskever, J. Martens, and G. E. Hinton, "Generating text with recurrent neural networks," in *Proceedings of the 28th International Conference on Machine Learning*, pp. 1017–1024, Bellevue, WA, USA, June 2011.

[88] G. Lakoff and M. Johnson, *Metaphors We Live by*, University of Chicago Press, Chicago, IL, USA, 2008.

[89] T. T. Rogers and J. L. McClelland, *Semantic Cognition: A Parallel Distributed Processing Approach*, MIT Press, Cambridge, MA, USA, 2004.

[90] K. Xu, "Show, attend and tell: neural image caption generation with visual attention," in *Proceedings of the International Conference on Learning Representations*, San Diego, CA, USA, May 2015.

[91] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pp. 6645–6649, Vancouver, Canada, May 2013.

[92] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," 2014, http://arxiv.org/abs/1410.5401.

[93] J. Weston, S. Chopra, and A. Bordes, "Memory networks," 2014, http://arxiv.org/abs/1410.3916.

[94] J. Weston, A. Bordes, S. Chopra, and T. Mikolov, "Towards AI-complete question answering: a set of prerequisite toy tasks," 2015, http://arxiv.org/abs/1502.05698.

[95] J. Zhao, S. Bhatt, and C. Thille, "Cold start knowledge tracing with attentive neural turing machine," in *Proceedings of the Seventh ACM Conference on Learning @ Scale*, New York, NY, USA, August 2020.