Received: 17 July 2018 | Revised: 28 January 2019 | Accepted: 1 March 2019

DOI: 10.1111/exsy.12400

ORIGINAL ARTICLE

WILEY Expert Systems

A systematic review on deep learning architectures and applications

Aditya Khamparia | Karan Mehtab Singh

School of Computer Science and Engineering, Department of Intelligent Systems, Lovely Professional University, Phagwara, Punjab, India

Correspondence
Aditya Khamparia, School of Computer Science and Engineering, Phagwara, Punjab, India.
Email: aditya.khamparia88@gmail.com

Abstract

The amount of digital data in the universe is growing at an exponential rate, doubling every 2 years, and changing how we live in the world. The information storage capacity and data requirement crossed the zettabytes. With this level of bombardment of data on machine learning techniques, it becomes very difficult to carry out parallel computations. Deep learning is broadening its scope and gaining more popularity in natural language processing, feature extraction and visualization, and almost in every machine learning trend. The purpose of this study is to provide a brief review of deep learning architectures and their working. Research papers and proceedings of conferences from various authentic resources (*Institute of Electrical and Electronics Engineers*, *Wiley*, *Nature*, and *Elsevier*) are studied and analyzed. Different architectures and their effectiveness to solve domain specific problems are evaluated. Various limitations and open problems of current architectures are discussed to provide better insights to help researchers and student to resume their research on these issues. One hundred one articles were reviewed for this meta-analysis of deep learning. From this analysis, it is concluded that advanced deep learning architectures are combinations of few conventional architectures. For example, deep belief network and convolutional neural network are used to build convolutional deep belief network, which has higher capabilities than the parent architectures. These combined architectures are more robust to explore the problem space and thus can be the answer to build a general-purpose architecture.

KEYWORDS

autoencoders, convolutional neural networks, deep learning, deep networks, restricted Boltzmann's machine, stacked autoencoders, tensor deep stack networks

1 | INTRODUCTION

Artificial intelligence is gaining more popularity than humans due to its increasing efficiency in various domains like sensory processing and high-level reasoning. Its stoic nature is very effective in each and every critical aspect of decision making. From the start of the fifth generation of computers, the sole motive of researchers is to mimic the effectiveness and robustness of the human brain. A human brain can handle an enormous amount of information and can store it in a synaptic importance hierarchy for future use. Knowledge representation and information handling capability of the brain is still beyond the current capabilities of artificial intelligence. The principal problem is the increase in the dimensionality of data (Bellman & Lee, 1984). The complexity of a learning algorithm increases exponentially with the linear increase in data dimensionality. This

problem can be solved by extracting the necessary features of the raw data by using various preprocessing techniques. But human-engineered preprocessing techniques are generally erroneous; those can yield incorrect results.

The working of deep learning is quite similar to the working of animal neocortex (Yamins & DiCarlo, 2016). In early stage of sensory processing, on and off-centre detectors in the retina and lateral geniculate nucleus perform a sort of image whitening and contrast normalization and allow the incoming information to pass through a number of levels or modules, arranged in a hierarchical manner. In a similar fashion, deep learning works with modules to reduce the complexity of learning algorithms.

Deep learning is a subclass of machine learning algorithms, which tends to learn the multiple representations of data. These representations are learned with different level of abstraction at each level (LeCun, Bengio, & Hinton, 2015). It can be supervised or unsupervised. The working of deep learning is very much similar to human brain. Deep learning uses multiple levels of non-linear processing units to extract the important features of the given data. The output of current layer will be used as the input to the next layer (Bengio, 2012a, 2012b; Deng, 2014). In the training phase, the stochastic gradient descent approach is used via back-propagation algorithm to search for an optimum; that is, algorithm searches around a stationary point.

The enormous amount of data is a problem for traditional learning algorithms. With more and more data coming into the picture, the size of network increases, and the performance of older algorithms either becomes saturated or degrades from the point.

As shown in Figure 1, performance curve took a shape of a plateau with increased data. The first reason for this saturation in performance is that the traditional machine learning algorithms were based on handcrafted rules and creating a large number of rules manually are an erroneous task. Use of sigmoid activation in many algorithms is the second reason, because it fails due to the vanishing gradient problem. When gradient becomes almost 0, the change is negligible in the performance. On the contrary, in deep learning, the performance keeps on increasing with the feeding of more data into the system. The reason for increasing performance lies in the fact that it uses more than one level of non-linear feature transformation of the given data as shown in Figure 2.

Another reason for the high performance of deep learning is its power to represent a function in a very compact form, which, if represented by a shallow architecture, will be very computationally expensive. Deep learning algorithm uses multiple layers of abstraction, means more sequential computations than parallel computations (Bengio & LeCun, 2007).

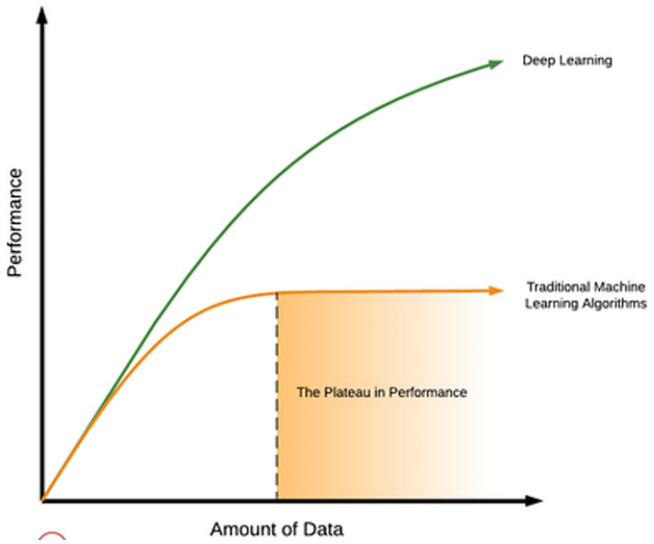

FIGURE 1 Deep learning versus traditional learning (Ng, 2015)

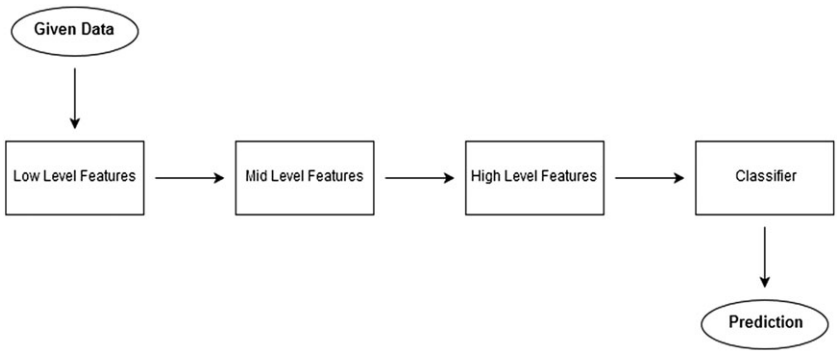

FIGURE 2 Multiple feature transformations (Zeiler & Fergus, 2014)

The rest of the paper is structured as follows: Section 2 lists the most popular techniques and architectures of deep learning. Section 3 provides an overview of the areas where deep learning is or can be applied. Section 4 offers the classification of academic papers used for this paper. Open problems in deep learning are discussed in Section 5. Finally, in Section 6, the review study is concluded.

2 | DEEP ARCHITECTURES

The deep architectures are different from conventional neural network architectures. Traditional computing hardware based on central processing units is not adequate to support multilayer architectures because in deep architectures, there are thousands of interconnections between these layers. Graphics processing unit (GPU) processing techniques empower these machines to execute deep architectures to detect a large number of features and higher-order feature interactions (Dean et al., 2012; Raina, Madhavan, & Ng, 2009).

Various deep architectures are used to implement deep learning algorithms. Some of the most used architectures with related applications are discussed in Table 1.

2.1 | Deep feed-forward neural networks

As shown in Figure 3, a basic neural network is made up of an input layer, a hidden layer, and an output layer. In a simple neural network, no computation is performed in any of the input nodes—they just pass on the information to the hidden nodes. Therefore, only two layers are there in this network. There is no restriction on the number of layers between the input and output layers, but going beyond two layers was impractical for a few years in the past (Glorot & Bengio, 2010). A simple neural network is initialized with random weights, and network is then trained with the help of gradient descent approach. If multiple hidden layers are present in the network, gradient descent approach gives a poor solution due to the vanishing gradient problem (Hochreiter, 1998).

Usage of GPU computing and availability of huge amount of training data made it possible to add more layers between input and output layers.

Adding two or more layers in a simple neural network modifies it to a deep neural network (Bengio, Ducharme, Vincent, & Janvin, 2003). Figure 3 shows a basic deep neural network with two hidden layers. Training of deep neural network is performed in three steps (Larochelle, Bengio, Louradour, & Lamblin, 2009).

1. Pretraining of one layer at a time in a greedy way.
2. Unsupervised learning at every layer to identify the factor of variations like background images and rotated images.
3. Fine tuning of all parameters of network using a global supervised cost function.

In greedy approach, supervised fine-tuning optimization will be initialized in a parameter space from where better local minima can be attained. With this technique of training a network, the performance increases, and complexity decreases, because this training is a guided training. Instead of learning random representations, the intermediate layers have a hint of what should be learned. This helps the optimization process to cover a specific region of parameter space where better local minima can be attained.

2.2 | Deep belief network

Deep belief networks are probabilistic generative graphical models with multiple layers of hidden units. As shown in Figure 4, the fundamental building block of a deep belief network is an undirected bipartite graphical model known as restricted Boltzmann's machine (RBM) (Salakhutdinov & Murray, 2008).

The construction of deep belief network is similar to the sigmoidal belief network (Bengio, 2009). The only difference is instead of having a global parameterization for the model in sigmoidal belief network as described in Equations 2.1 and 2.2:

$$P\left(x, \mathrm{h}^1, \ldots \mathrm{h}^l\right) = P\left(\mathrm{h}^l\right)\left(\prod_{k=1}^{l-1} P\left(\mathrm{h}^k | \mathrm{h}^{k+1}\right)\right) P\left(x | \mathrm{h}^1\right), \tag{2.1}$$

$$P\left(h_i^k = 1 / \mathrm{h}^{k+1}\right) = sigm\left(b_i^k + \sum_j W_{ij}^{k+1}\, h_i^{k+1}\right), \tag{2.2}$$

(where x is input vector, $h^{k\,=\,1\ \text{to}\ l}$ is hidden layers, and h_i^k hidden node i in kth layer),

TABLE 1 Comparison of deep architectures and related applications

Authors	Year	Journal	Deep architecture/methods	Application
Alezandro M.	2018	*Parallel and Distributed Computing*	Convolutional neural network and autoencoders	Bioinformatics
Wang B., Sun Y., Xue B., and Zhang M.	2018	*IEEE*	Convolutional neural network and swarm intelligence	Image classification
Daneshzand M., Ibrahim S. A., Faezipour M., and Barkana B. D.	2017	*IEEE*	Deep brain stimulation	Treatment of Parkinson's disease
Daneshzand, M., Faezipour, and Barkana B. D.	2017	*Frontiers in Computational Neuro Science*	Deep brain simulation	Bioinformatics—motor treatment in neurological disorders
Testolin A., Filippo M. D., Grazia D., and Zorzi M.	2017	*Frontiers in Computational Neuro Science*	Restricted Boltzmann machine	Modelling neural information processing in biological system
Zhang S., Wang J., Tao X., Gong Y., and Zheng N.	2017	*Pattern Recognition*	Deep sparse coding network	Image classification
Karpathy A. and Fei-Fei L.	2017	*IEEE Transactions on Pattern Analysis and Machine Intelligence*	Convolutional and multimodal recurrent neural network	Natural language descriptions of images and their regions
DeVries P., Thompson T., and Meade B.	2017	*Geophysical Research Letters*	Deep convolutional network	Understanding repetition of earthquake cycle activity
Testolin A., Stoianov I., and Zorzi M.	2017	*Nature Human Behaviour*	Deep belief network	Letter recognition
Sabour S., Frosst N., and Hinton G. E	2017	*Neural Information Processing System*	Capsule network	Handwritten digit recognition
Lotter W., Kreiman G., and Cox D.	2017	*International Conference on Machine Learning (ICLR)*	Deep predictive coding network	Prediction of future frames in video sequences
Shwartz S., Shamir O., and Shammah S.	2017	*International Conference on Machine Learning (ICLR)*	Stochastic gradient-based deep learning network	Audio and natural language processing
Chang J., Li C., Poczos B., and Kumar B.	2017	*IEEE*	Deep projection	Image inpainting and super resolution
Gowda S.	2017	*Computer Vision and Pattern Recognition*	Deep belief network	Human activity recognition
Yamins D. L. and DiCarlo J. J.	2016	*Nature Neuroscience*	Hierarchical deep convolutional network	Object categorization and detection
Testolin A. and Zorzi M.	2016	*Frontiers in Computational Neuroscience*	Generative models	Treatment of neurocognitive disorders and impairments
Wilson A. G., Hu Z., Salakhutdinov R., and Xing E. P.	2016	*Artificial Intelligence and Statistics*	Deep belief network with Gaussian kernels	Face identification
Hwang J. and Zhou Y.	2016	*World Conference on Computer Vision*	Convolutional neural network	Image colorization
Varga D. and Sziranyi T.	2016	*International Conference on Pattern Recognition (ICPR)*	Convolutional neural network	Automatic black and white to full RGB colorization
Assael Y. M., Shillingford B., Whiteson S., and Freitas N.	2016	*International Conference on Computer Vision and Pattern Recognition*	Convolutional and recurrent network	Sentence-based lip reading
Wilson A. G., Hu Z., Salakhutdinov R., and Xing E. P.	2016	*Advances in Neural Information Processing Systems*	Deep kernel learning	Classification, multitask learning
Mnih V. et al.	2015	*Nature*	Deep Q network	Development of artificial agents
Owens A. et al.	2015	*International Conference on Computer Vision and Pattern Recognition*	Recurrent neural network	Sound prediction

(Continues)

TABLE 1 (Continued)

Authors	Year	Journal	Deep architecture/methods	Application
Schmidhuber J.	2015	*Neural Networks*	Deep autoencoders	Pattern recognition and machine learning
Radford A., Metz L., and Chintala S.	2015	*Computer Vision and Pattern Recognition*	Deep convolutional GAN networks	Image representations and computer vision
Deng L.	2014	*Foundation and Trends in Signal Processing*	Deep learning models	Speech recognition and computer vision
David E. and Greental I.	2014	*ACM*	Genetic algorithm—deep neural network	Image translation and gene identification
Zeiler M. and Fergus R.	2014	*Computer Vision ECCV*	Convolutional model	Automatic image classification, description, and augmentation
Ren Y. and Wu Y.	2014	*IJCNN*	Deep belief network	Feature extraction of EEG signals
Srivastava N. and Salakhutdinov R.	2014	*Journal of Machine Learning Research*	Deep Boltzmann machine	Multimodal fusion of image–text and audio–video data
Courville A., Desjardins G., Bergstra J., and Bengio Y.	2014	*IEEE Transaction on Pattern Analysis and Machine Intelligence*	Restricted Boltzmann machine	Discrete and sparse data representations
Graves A. and Jaitly N.	2014	*International Conference on Machine Learning (ICML)*	Recurrent networks	Speech recognition and translation
Wiebe N., Kapoor A., and Svore K. M.	2014	*Neural and Evolutionary Computing*	Deep restricted Boltzmann machine	Quantum physics
Zorzi M. et al.	2013	*Frontiers in Psychology*	Stochastic recurrent neural network	Modelling language and cognitive processing
Hutchinson B., Deng L., and Yu D.	2013	*IEEE Transactions on Pattern Analysis and Machine Intelligence*	Tensor deep stack networks	Handwritten digit recognition and phone classification
Deng L., He X., and Gao J.	2013	*IEEE Conference on Acoustic, Speech and Signal Processing*	Tensor deep stack network	Information retrieval
Mnih V., Kavukcuoglu K., Silver D., Graves A., Antonoglou I., Wierstra D., and Riedmiller M.	2013	*NIPS*	Convolutional neural network	Game playing
Deng L., Hinton G., and Kingsbury B.	2013	*IEEE Conference on Acoustic, Speech and Signal Processing*	Recurrent neural network	Speech recognition
Deng L., Yu D., and Platt J.	2012	*IEEE ICASSP*	Deep stack networks	Image classification
Hinton et al.	2012	*IEEE Signal Processing Magazine*	Deep neural network	Acoustic modelling in speech recognition using Markov chain
Bengio Y.	2012	*ICML Unsupervised and Transfer learning*	Unsupervised network	Feature detection and classification
Krizhevsky A., Sutskever I., and Hinton G. E.	2012	*Advances in Neural Information Processing Systems*	Convolutional neural network	Image classification and fusion
Ciresan D., Meier U., and Schmidhuber J.	2012	*IEEE Computer Vision and Pattern Recognition*	Convolutional neural network	Traffic sign detection and classification
Huang G. B., Lee H., and Learned-Miller E.	2012	*IEEE Computer Vision and Pattern Recognition*	Convolutional deep belief network	Face detection and verification
Mohamed A., Dahl G., and Hinton G.	2012	*IEEE Transactions on Audio Signal and Image Processing*	Deep belief network	Acoustic modelling
Lee H., Grosse R., Ranganath R., and Ng A.	2011	*Communication of ACM*	Deep belief network	DNA strands hierarchical classification
Le Q., Ngiam J., Coates A., Lahiri A., Prochnow B., and Ng A.	2011	*International Conference on Machine Learning*	Convolutional neural network	Optimization

(Continues)

TABLE 1 (Continued)

Authors	Year	Journal	Deep architecture/methods	Application
Coates A., Ng A., and Lee H.	2011	*Artificial Intelligence and Statistics*	Autoencoders	Canadian Institute for Advanced Research (CIFAR) and Modified National Institute of Standards and Technology Database (MNIST) handwritten
Courville A., Bergstra J., and Bengio Y.	2011	*Artificial Intelligence and Statistics*	Restricted Boltzmann Machine	Image extension and fusion
Ngiam J., Khosla A., Kim M., Nam J., Lee H., and Ng A. Y.	2011	*International Conference on Machine Learning*	Deep convolutional network	Multimodal image representation and fusion
Mohamed A., Sainath T., Dahl G., Ramabhadran B., Hinton G., and Picheny B.	2011	*Acoustic Speech and Signal Processing*	Deep belief network	Speech identification
Vincent P., Larochelle H., Lajoie I., Bengio Y., and Manzagol P.	2010	*Journal of Machine Learning Research*	Stacked autoencoder	Denoising
Courville A., Bergstra J., and Bengio Y.	2010	*NIPS*	Restricted Boltzmann machine	Image modelling and transformation
Mitrovic J., Sejdinovic D., and The Y. W.	2010	*ICLR*	Deep kernel machine	Game designing
Lin Y., Zhang T., Zhu S., and Yu K.	2010	*Advances in Neural Information Processing Systems*	Deep coding network	Automatic sound recognition
Arel I., Rose D., and Karnowski T.	2010	*IEEE Computational Intelligence Magazine*	Convolutional neural network and deep belief network	Pattern recognition
Raina R., Madhavan A., and Ng A. Y.	2009	*International Conference on Machine Learning*	Deep belief and sparse coding network	Parallel programming and graphic processing hypertuning
Larochelle H., Bengio J., Louradour, and Lamblin P.	2009	*Journal of Machine Learning Research*	Deep belief and restricted Boltzmann machine	Weight optimization
Lee H., Grosse R., Ranganath R., and Ng A.	2009	*ICML*	Convolutional deep belief network	Cluster formation and learning
Salakhutdinov R. and Hinton G.	2009	*Artificial Intelligence and Statistics*	Deep Boltzmann machine	Image labelling and picture formation
Salakhutdinov R. and Larochelle H.	2009	*Artificial Intelligence and Statistics*	Restricted deep Boltzmann machine	Linguistics and computer vision
Kolda T. and Bader B.	2009	*SIAM Review*	Tensor network	Robotics
Cho Y. and Saul L.	2009	*Neural Information Processing System*	Tensor-encoding network	Linguistics
Weinberger K. and Saul L.	2009	*Journal of Machine Learning Research*	K-nearest neighbour	Clustering
Jarrett K., Kavukcuoglu K., Ranzato M., and LeCun Y.	2009	*Computer Vision*	Deep convolutional network	Object recognition
Salakhutdinov R. and Murray I.	2008	*International Conference on Machine learning*	Deep belief network	Image matching
Roux N. L. and Bengio Y.	2008	*Neural Computation*	Restricted Boltzmann machine and deep belief network	Robotics and linguistics
Collobert R. and Weston J.	2008	*ICML*	Recurrent neural network using long short-term memory	Natural language processing
Vincent P., Larochelle H., Bengio Y., and Manzagol P. A.	2008	*ICML*	Autoencoders	Acoustic processing
Bengio Y. and LeCun Y	2007	*Book Series*	Deep kernel machines	Meta-heuristic searching

(Continues)

TABLE 1 (Continued)

Authors	Year	Journal	Deep architecture/methods	Application
Ranzato M., Boureau Y., and LeCun Y.	2007	*Neural Information Processing System*	Deep belief networks	Feature classification and mapping in computer applications
Hinton G. E., Osindero S., and The Y.	2006	*Neural Computation*	Deep belief networks	Image recognition and pattern recognition
Hinton G.	2006	*Science*	Principal component analysis	Pattern classification
Graves A. and Schmidhuber J.	2005	*Joint Conference on Neural Networks*	Long short-term memory-based recurrent neural networks	Phoneme classification
Abbas H. A.	2002	*Artificial Intelligence in Medicine*	Convolutional neural model	Breast cancer diagnosis
Gers F., Schmidhuber J., and Cummins F.	2000	*Neural Computation*	Recurrent network	Continual prediction in voice
Yao X.	1999	*IEEE*	Deep network model	Image recognition
Hochreiter S.	1998	*International Journal of Uncertainty Fuzziness and Knowledge based systems*	Recurrent neural network	To avoid vanishing gradient problem in convergence applications
Lawrence S., Giles C., Tsoi A. C., and Back A.	1997	*IEEE Transactions on Neural Networks*	Convolutional neural network	Face recognition
Hinton G. E. and Zemel R. S.	1994	*Advances in Neural Information Processing Systems*	Autoencoders	Image categorization
LeCun Y. et al.	1989	*Neural Computation*	Backpropagation gradient descent network	Handwritten digit recognition

Note. EEG: electroencephalography; GAN: generative adversarial network; RGB: red–blue–green.

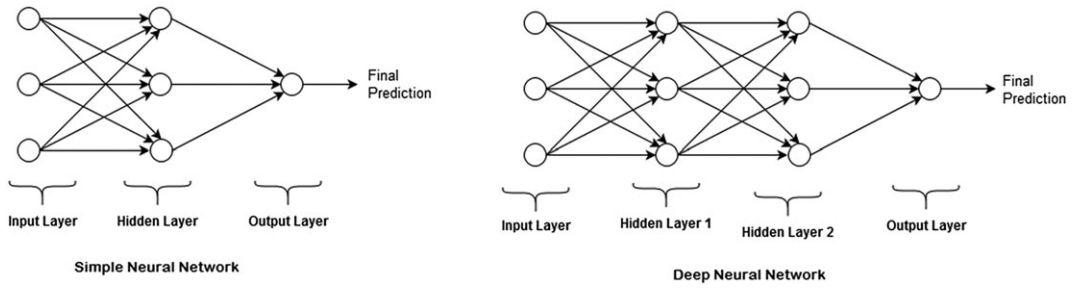

FIGURE 3 Simple neural network and deep neural network

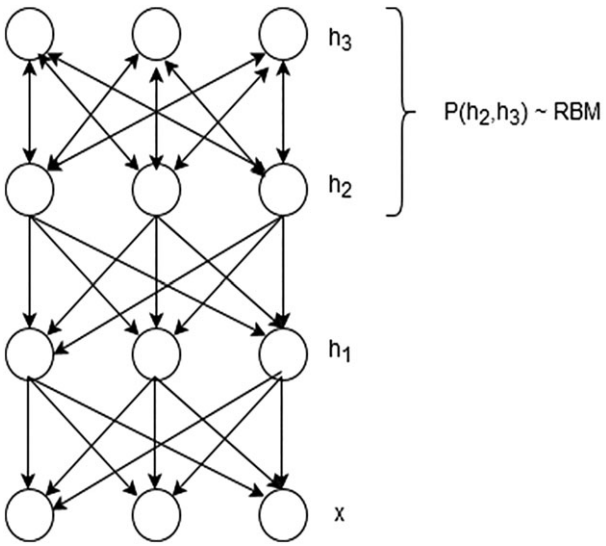

FIGURE 4 Deep belief network

in deep belief network, the parameterization for top two layers, that is, l and $l-1$, is

$$P\left(x, h^1, \ldots h^l\right) = P\left(h^{l-1}, h^l\right)\left(\prod_{k=1}^{l-2} P\left(h^k | h^{k+1}\right)\right) P\left(x | h^1\right). \tag{2.3}$$

Learning for deep belief network is different because it exploits the greedy layer by layer technique of training. In layer-by-layer training, while training the lower layers, it is assumed that higher layers do not exist. A new approach was introduced in 2006, in which these higher layers are not ignored but the tied weights are used (Hinton, Osindero, & The, 2006). The usage of tied weight to construct complementary prior and progressively untying the weights in the current layer from higher-layer weights is much efficient from old techniques (Zorzi, Testolin, & Stoianov, 2013).

2.3 | Convolutional neural network

The current artificial neural network technology that we are using is far behind the complexity and sophistication of biological neural network. The development of convolutional neural network is based on the mammalian visual system (Hubel & Wiesel, 1962; Daneshzand, Faezipour, & Barkana, 2017). The convolutional neural network is very effective in visual recognition systems because of its complexity. A simple convolutional neural network is made up of various layers.

The success of convolutional neural network is because of three important properties (Lawrence, Giles, Tsoi, & Back, 1997). These are as follows:

a. local receptive fields,
b. shared weights, and
c. spatial subsampling.

Local receptive fields mean that the response of a neuron is influenced by a specific 2D portion of an image. Shared weights are the plus points for convolutional neural network as with these weights, the overall number of parameters in the network is reduced. This helps to achieve an effective generalization (LeCun & Bengio, 1995). Spatial subsampling is another layer in the network that performs local averaging and subsampling. Subsampling is used to reduce the resolution of the feature map. This solves the problem of distortion and shifts in the final output.

As shown in Figure 5, convolutional neural network contains a number of convolutional and subsampling layers in one or more planes. Every plane, which is subsampled from centred and normalized previous layer image, is known as feature map, which has fixed number of feature detectors. The reason for using multiple planes is to detect the maximum number of features of the image. A subsampling factor is used for local averaging (LeCun et al., 1989). The network is trained using a gradient descent approach via back-propagation algorithm (Riesenhuber & Poggio, 1999).

2.4 | Convolutional deep belief network

Networks like deep belief networks are very popular due to few factors like unsupervised learning and hierarchical generative models (Testolin & Zorzi, 2016). The downside of these networks is that these cannot be scaled to higher dimensional full-size images (Lee, Grosse, Ranganath, & Ng, 2009). Both RBM and deep belief network ignore the 2D structure of an image. Due to this drawback, all the weights, those detect a particular feature in an image, must be learned separately for each portion. To overcome this problem, convolutional deep belief network was introduced in 2011 (Lee, Grosse, Ranganath, & Ng, 2011). It is a hierarchical generative model, which can be scaled to high dimensional images.

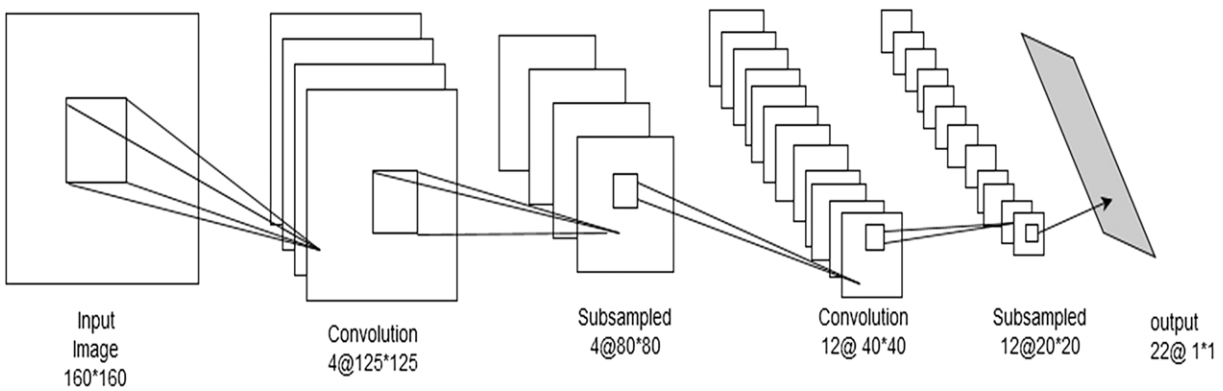

FIGURE 5 Convolutional neural network

In convolutional deep belief network, weights are shared among all positions in a given image. This architecture is an extension of deep belief network with a weight-sharing capability. Weight sharing is a technique in which same filter weights are used in the convolutional layer to generate the feature map. Dot product is calculated of filter weights and the image pixel by sliding the same filter window over the entire image. The fundamental building block of convolutional deep belief network is convolutional RBM.

As shown in Figure 6, a simple convolutional RBM consists of two layers, one visible layer of Nv × Nv pixels and other one, which is a hidden layer of Nh × Nh pixels. *K* is the number of convolution filter weights between hidden and visible node. Each filter covers an area of Nw × Nw pixels (Ren & Wu, 2014).

The architecture of convolutional deep belief network consists of multiple max-pooling convolutional RBM stacked on one another (Lee et al., 2011).

The max-pooling layer shrinks the representation of filtering layer by a constant factor c. This shrinkage in representation removes the excess computational load from higher layers as these layers become invariant to small translations occurring in input vectors. Training is carried out through sparsity regularization (means only a few units are active with respect to the provided stimulus) and contrastive divergence (Hinton, 2002).

2.5 | Deep Boltzmann's machines

The deep Boltzmann's machine is a network of symmetrically coupled stochastic binary units. A simple deep Boltzmann's machine contains a set of visible units and multiple layers of hidden units in a sequence. The connections between visible-to-hidden (bottom layer) and hidden-to-hidden layers are undirected (Salakhutdinov & Hinton, 2009).

Deep Boltzmann's machines are quite popular these days because of its undirected architecture that can be used to build a top-down feedback, which provides a better way to handle the ambiguous inputs (Salakhutdinov & Larochelle, 2009).

A simple deep Boltzmann's machine with three hidden layers is depicted in Figure 7; the energy of the state $\{v, h\}$ is defined by

$$E(v, h; \theta) = -v^T W^1 h^1 - h^{1T} W^2 h^2 - h^{2T} W^3 h^3, \tag{2.4}$$

where $h = \{h^1, h^2, h^3\}$ is set of hidden units and $\theta = \{W_1, W_2, W_3\}$ is set of model parameters representing visible-to-hidden– (bottom layer) and hidden-to-hidden–layer interaction weights.

The learning of deep Boltzmann's machine is performed in two steps (Srivastava & Salakhutdinov, 2014).

1. Greedy layer-by-layer pretraining of units. This helps to initialize the interaction weights (between visible to hidden and hidden to hidden) to a reasonable value.
2. In the second step, separate recognition model is used to initialize the latent variable in the hidden layers followed by a single cycle of mean-field inference.

2.6 | Deep stacking network

Deep stacking network is based on stacked generalization (Wolpert, 1992). Stacked generalization is a method of combining multiple models for classification tasks to lower the generalization error. In stacked generalization, more than one equally or more capable modules of same or

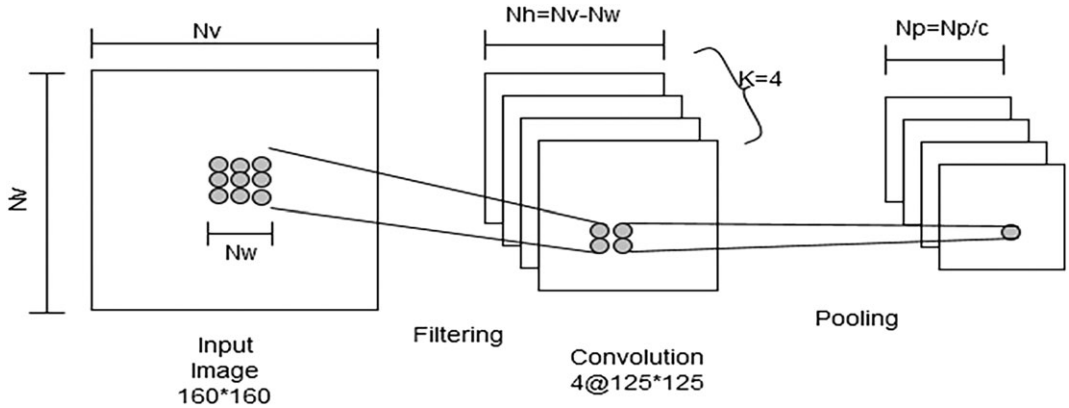

FIGURE 6 Convolutional restricted Boltzmann's machine

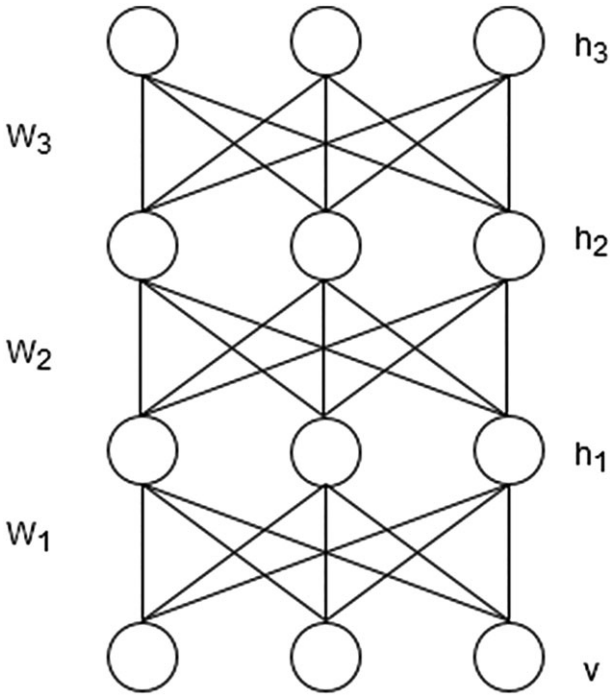

FIGURE 7 Deep Boltzmann's machine

different architecture are stacked over each other (Hutchinson, Deng, & Yu, 2013). This offers a better generalization as compared with single module architecture. Stacked generalization works by deducing the biases of the generalizer(s) with respect to a provided learning set. Stacked generalization when used with multiple generalizers works effectively and better than cross-validation strategy adopted by individual generalizers. When used with a single generalizer, stacked generalization is a scheme for estimating the error of a generalizer, which has been trained on a particular learning set. Deep stacking network is a scalable architecture, which is capable of parallel weight learning without using back propagation. The power of deep stacking network in scalable learning is due to its simple training objective. It uses mean square error between the target value and the value predicted by the current modules as an objective function (Deng, He, & Gao, 2013).

where x is D-dimensional input vector, W_i is weight matrix for input-to-hidden layer, H_i is hidden layer of ith module, U_i is weight matrix of hidden-to-prediction layer, Y^i is prediction of ith block, and * means the output and original input vector concatenation.

As shown in Figure 8, the bottom module consists of three layers:

1. raw input layer X_i,
2. second unit is a non-linear unit, consists of a set of sigmoidal hidden units H_1, and
3. linear output/prediction layer Y_1.

Hidden layer's output is given by $=\sigma(W^T X_i)$, where $\sigma(.)$ is the sigmoidal function, W is trainable weight matrix of dimension D × L, and L is the number of hidden units.

Prediction layer's output is given by $=(U^T H_i)$, where W is trainable weight matrix of dimension L × C and C is the number of output units.

The next modules structure is very much similar to the bottom layer; the only difference is in the input layer. Original input is concatenated with the output of the previous module. This concatenated vector is treated as the input to the second module. The dimensionality of the augmented input vectors is a function of block number m, counted in a bottom-up manner (Deng, Yu, & Platt, 2012).

$$D_m = D + C\,(m - 1), \tag{2.5}$$

where m = 1 for bottom module.

W and U are determined to minimize the loss function

$$E = \|Y - T\|^2 = Tr\left[(Y - T)(Y - T)^T\right], \tag{2.6}$$

where T is the target vector and Y is set of set of $\{y_1, y_2, \ldots, y_N\}$ and N is total number of training examples.

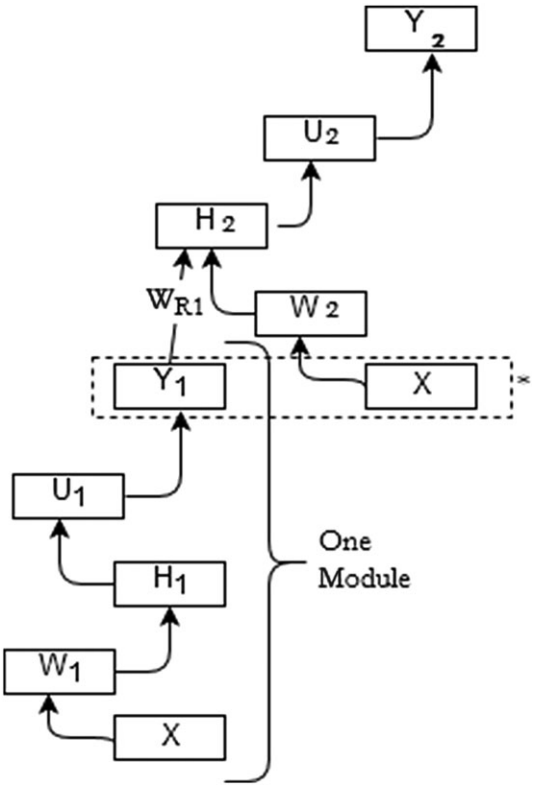

FIGURE 8 Deep stacking network

The upper-layer weight matrix U in each module can be formulated by setting the gradient

$$\frac{\partial E}{\partial U} = 2H\left(U^T H - T\right)^T, \tag{2.7}$$

to 1. This is a convex optimization problem and has a straightforward closed-form solution,

$$U = \left(HH^T\right)^{-1} HT^T. \tag{2.8}$$

Lower-layer weight matrix W can be learned using batch mode gradient descent approach. The output of every module is based upon the concatenated value of original input and predicted value by the previous module. This means the performance of current block will always be better than the previous module.

2.7 | Tensor deep stacking network

The architectural foundation of tensor deep stacking network is based on deep stacking network (Kolda & Bader, 2009). The differences between these two architectures are as follows:

1. Instead of having a single layer of hidden units between input and output layers, in tensor deep stacking network, there are two parallel layers of hidden unit. As shown in Figure 9, $H^1{}_2$ and $H^1{}_2$ are two hidden layers of first block.
2. Between prediction and hidden layer, there is a third-order weight tensor.

Tensor deep stacking network uses two parallel hidden representations and combines these two representations using a bilinear mapping, to give the final prediction (Weisstein, 2012). In tensor deep stacking network, the working starts with the mapping of input data $X \in \mathbf{R}^D$ to a pair of parallel hidden layers h_1 ($\in \mathbf{R}^{L1}$) and h_2 ($\in \mathbf{R}^{L2}$). L1 and L2 represent the number of units in h_1 and h_2. Each layer produces a different non-linear view of the data via function $h_j = \sigma\left(W_j^T X\right)$; j has a value either 1 or 2, and $W_1 \in \mathbf{R}^{D \times L1}$ and $W_2 \in \mathbf{R}^{D \times L2}$ are two weight matrices.(x is D-dimensional input vector, W^i_j is weight matrix for input-to-jth hidden layer of ith module, H^i_j is jth hidden layer of ith module, U_i is weight tensor for hidden-to-prediction layer, and Y_i prediction of ith block)

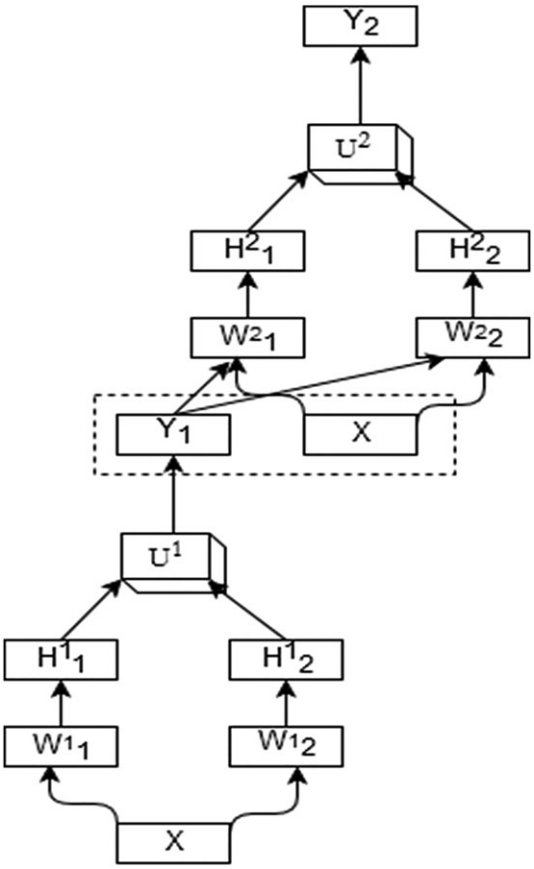

FIGURE 9 Tensor deep stacking network

Between hidden layers and prediction layer, a tensor of third-order $\mu \in \boldsymbol{R}^{L1 \times L2 \times C}$ is used. C is the number of output units in the prediction layer. Output of prediction layer in tensor notation is

$$y_k = \sum_{i=1}^{L1} \sum_{J=1}^{L2} \mu_{ijk} h_{(1)i} h_{(2)j} = h_{(1)}^T \mu_k h_{(2)}, \tag{2.9}$$

where $\mu \in \mathbf{R}^{L1 \times L2}$ denotes the slice of tensor, which can be obtained by using k in the third index, and first and second index can vary.

The weight matrices W_j^i can be initialized to random values. Most of the training of tensor deep stacking network is gradient computation based, and this can be parallelized to speed up the training process.

2.8 | Stacked autoencoders

An autoencoder is used to encode the input $x \in \mathbf{R}^D$ into a representation $c(x)$, in such a way that input x can be reconstructed from the representation $c(x)$ (Hinton & Zemel, 1994; Vincent, Larochelle, Lajoie, Bengio, & Manzagol, 2010). Autoencoder is also known as autoassociator or diabolo network and is used for dimensionality reduction. In encoding phase, it uses a set of recognition weights to convert an input vector into a code vector. In the decoding phase, it uses a generative set of weights for the approximate reconstruction of input vector (Hinton & Zemel, 1994). The complete process is described in Figure 10.

$$y = \sigma(Wx + b), \tag{2.10}$$

$$x' = \sigma'\left(W'y + b'\right), \tag{2.11}$$

where σ is element-wise activation function, W is weight matrix, and b is bias. σ' ,W', b' may be different from the encoding phase parameters.

If the number of hidden units is lower than the number of visible units, the system is forced to learn a compressed representation of the input. Due to this effective results, if a sparsity constraint is introduced on hidden layers, the results will become better (Bengio, Lamblin, Popovici, &

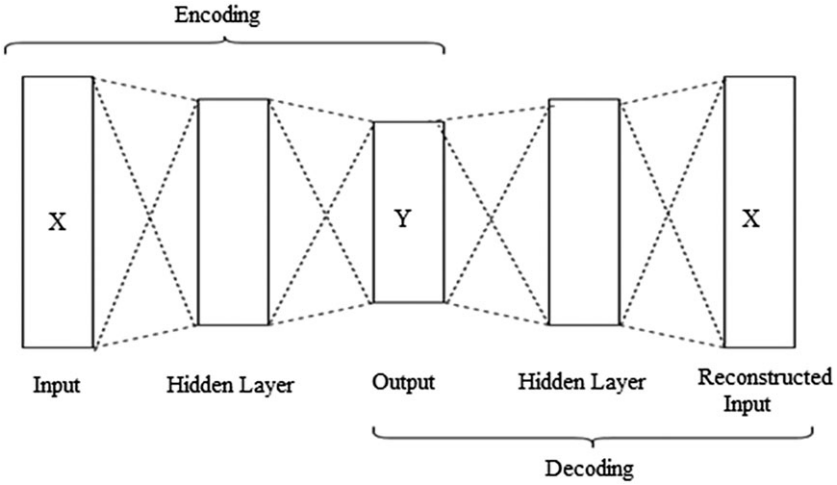

FIGURE 10 Autoencoder

Larochelle, 2007; Hinton, 2006; Le et al., 2011; Ng, 2011). The constraint keeps most of the neuron inactive. An autoencoder with this sparsity penalty is known as sparse autoencoder (Coates, Ng, & Lee, 2011; Testolin, De Filippo De Grazia, & Zorzi, 2017).

A stacked autoencoder is a network made up of multiple layers of sparse autoencoders. Stacked autoencoders have more expressive power than the other autoencoders. The first layer of this will detect the first-order features of the input data, and the second layer will detect the second-order features present in first-order features captured by previous layer and so on.

2.9 | Spike and slab RBMs

The spike and slab RBM is constructed by introducing a real-valued vector (slab) and a binary-valued variable (spike) for each of the hidden units (Courville, Bergstra, & Bengio, 2011a). Like RBM, the spike and slab RBM is also restricted to bipartite graph structure between two types of nodes (Courville, Bergstra, & Bengio, 2010; Courville, Desjardins, Bergstra, & Bengio, 2014).

The visible layer is represented in a similar way as in Gaussian RBM approach. The difference between the spike and slab RBM and other RBM variants lies in the definition of the hidden-layer latent variables. In spike and slab RBM, these latent variables are defined as element-wise multiplication of a real-valued vector with a binary variable. The use of dual latent variable allows the choice of data representation based on the type of tasks, which is to be done.

For N number of hidden units, visible vector $v \in \mathbf{R}^D$, if ith hidden unit is associated with a real-valued vector $s_i \in \mathbf{R}^K$ (pooling over K features) and binary values variable $h_1 \in \{0, 1\}$, the energy function for single example will be

$$E(v, s, h) = \frac{1}{2} v^T \Lambda v - \sum_{i=1}^{N} \left(v^T W_i s_i h_i + \frac{1}{2} s_i^T \alpha_i s_i + b_i h_i \right) , \tag{2.12}$$

where W_i is jth weight matrix, b_i is bias associated with h_i, and α_i and Λ are diagonal matrices for penalize large values of $||s_i||^2$ and $||v||^2$, respectively (Courville, Bergstra, & Bengio, 2011b).

The spike and slab model of RBM focuses on the learning process to stabilize the model, unlike others, models those only guarantee that the model forms a well-defined probability model. This model depends on the property of likelihood gradient to reject samples drawn from region supported by nearby examples. This helps to stabilize a model. Along with this, the use of decreasing learning rate also adds to the stability of model in a stable region of parameter space.

2.10 | Deep kernel machines

Stacking kernels over one another in a manner similar to deep neural network constitute deep kernel machines (Cho and Saul, 2009; Mitrovic, Sejdinovic, & The, 2010; Weinberger & Saul, 2009). The structure of deep kernel machines can be linear or non-linear. The power of deep kernel machines is based on the combination of nonparametric flexibility of kernel techniques with the structural properties of deep architectures (Wilson, Hu, Salakhutdinov, & Xing, 2016).

The learning of deep kernel machines is performed by applying kernel principal component analysis at each layer. The output of current layer will be considered as input to the next layer (Schölkopf, Smola, & Müller, 1998). Not all the components of the current layer output are transmitted to the next layer because some are uninformative and are discarded.

Selection of kernel principal component analysis is purely dependent on the user, but arc cosine kernels are considered better over others to work on large neural networks. Figure 11 shows a deep kernel machine with a Gaussian process layer. *N*th-order arc cosine kernel function is defined in integral representation as follows:

$$k_n(x,y) = 2\int \partial w \frac{e^{\frac{-||w||^2}{2}}}{2\pi^{\frac{d}{2}}} \theta(w.x)\theta(w.y)(w.x)^n w.y^n. \tag{2.13}$$

x, y ∈ $\mathbf{R}^D$ are input vectors, $\theta(z) = \frac{1}{2}(1 + \text{sign}(z))$ denotes the heavy side-step function, and *w* is weight matrix.

These kernel functions are positive semidefinite. For *l*-layer deep kernel machine, the training procedure is as follows:

1. Prune the uniformative features from the input vector.
2. Repeat *l* times
   a. Apply principal component analysis in feature vector induced by non-linear kernel.
   b. Prune uninformative components from the output.
3. Learn Mahalanobis distancemetric using large nearest-neighbour classification.

2.11 | Deep coding networks

The construction of the deep coding network is inspired by the sparse coding network. In a deep coding network, there are multiple layers as compared with a single layer in sparse coding network (Hwang & Zhou, 2016; Lin, Zhang, Zhu, & Yu, 2010). The sparse coding network is very effective in various classification problems. The sparse coding network was further extended by adding a locality constraint called local coordinate coding and that is even further extended by adding local tangent directions (Yu & Zhang, 2010). The idea of multilayer coding network is derived from the same approach, and this has multiple advantages over the single layer. The structure of deep coding network is similar to the structure of deep belief networks.

This hierarchical sparse coding network has two main advantages over the single layer model:

1. Hierarchical sparse coding network has a higher computational efficiency as compared with single layer counterpart.
2. The in-depth local search provided by the extended layer for each function avoids the problem of overfitting.

This approach uses a fitting mechanism using more than one small model for every individual layer, instead of using a single mechanism for whole network (Lin et al., 2010).

2.12 | "Long short-term memory" recurrent neural network

The construction of long short-term memory recurrent neural network (LSTM-RNN) is based on the context awareness problems. Humans link the precious required knowledge with the currently gained knowledge to predict the next scenario. For example, a sentence "I lived in France for 30

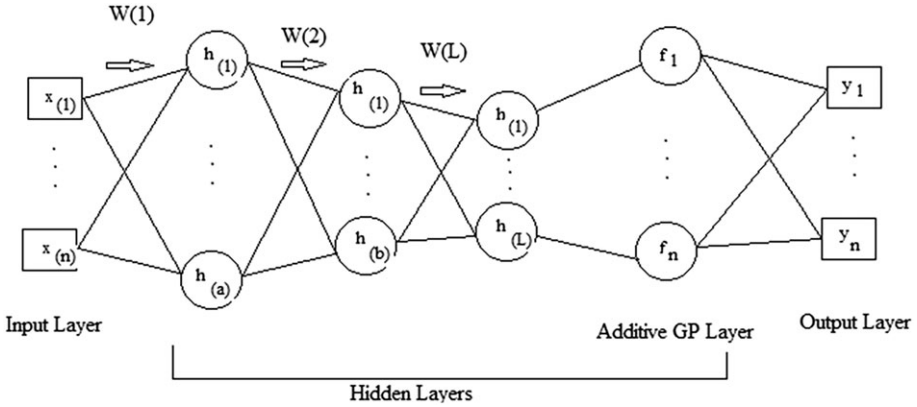

FIGURE 11 Deep kernel machine

years, I speak fluent?" this incomplete sentence can be filled by using and linking the previously acquired knowledge. RNN allows the loops in the network to link the previous knowledge. A problem with RNN arises when the gap between the prediction and the acquired knowledge becomes large; RNN fails to link the previous knowledge. For example, RNN can answer a question like "the colour of sky is? but where the gap is large like in previous sentence, system cannot predict the answer 'french.'"

LSTM network has the capability to learn long-term dependencies (Gers, Schmidhuber, & Cummins, 2000; Hochreiter & Schmidhuber, 1997). LSTM architecture consists of various memory blocks known as cells. These cells are used to store information for future use. Apart from cells, there are three types of gates used in LSTM. These gates are forget gate, input gate, and output gate. When a new information is encountered, input gate is used to control the threshold for incoming information into the cell. Forget gate controls the extent up to which the input values are present in the cell. Output gate is used to use these stored input values for computing output activations. Training of LSTM is carried out through gradient descent approach.

3 | APPLICATIONS OF DEEP LEARNING

Deep learning is not limited to one domain of research; it can be used in any application. Deep learning is already in use in various applications, from simple speech recognition to most scientific studies like earthquake prediction systems. Few of the interesting examples of deep learning are discussed below.

3.1 | Image recognition

This area is the most famous area for deep learning applications. Various successful products are already there in the market for image recognition and classification using deep learning. Few examples are as follows:

Automatic colorization of grey scale images. Deep learning algorithms are trained on millions of images, and these algorithms try to learn basic colour patterns of everyday life like the "sky is blue" and "apple is red." This information is then used to colour the detected objects in an image (Varga & Sziranyi, 2016; Zhang, Wang, Tao, Gong, & Zheng, 2017).

Super-resolution is another problem domain where deep learning has a very good scope. Super-resolution is a way where a high resolution is constructed from a very low-resolution image. Recently, in the Google brain residency programme, an 8 × 8 image is used to construct a 32 × 32-pixel image using pixel recursive super-resolution model (Dahl, Norouzi, & Shlens, 2017).

3.2 | Image description or caption generation

Facebook and Google are already using this feature to classify your images, and "tag a person" suggestion is totally based on deep learning. Google photos are a real-life application of this. The system is trained using deep learning and the ImageNet data set of labelled images.

Deep learning is not limited to just classify your image, but it can describe it as well in simple English. A new approach multimodal RNN, which is a combination of convolutional neural network and bidirectional RNN, achieved a good performance in describing a given image (Karpathy & Fei-Fei, 2017). The model is initially trained using described images that means the image is supplied to the training algorithm with its description. Convolutional neural network has used over an image, and RNN helps to learn the description of an image. Figure 12 shows an example of the output of this model (DeVries, Thompson, & Meade, 2017).

FIGURE 12 Image description (Karpathy & Fei-Fei, 2017)

Apart from these domains, deep learning is also used to perform various scientific calculations to speed up the system. For example, its ability to represent complex functions in a very compact way leads it to the large-scale calculations in the viscoelastic earthquake cycle model. These calculations are carried out for thousands of times and in various locations to generate a final geometry of the region. Previous models took hundreds of hours to perform these calculations. Deep neural network is used in this experiment, and the results are astonishing; DNN speeds up the system by more than 50,000% (DeVries et al., 2017).

3.3 | Speech recognition

Speech signals are sound waves travelling through a medium. These signals are represented by a spectrogram, means the collection of different frequencies and amplitudes of a wave. These spatial–temporal signals cannot be used efficiently in a simple neural network because of complexity. Previously, hidden Markov model (HMM)–Gaussian mixture model were used in speech recognition, and these models performed well, but there were few shortcomings of these models. In cases where data distribution occupies a much lower dimensional manifold as compared with the high dimensional space, these models are inefficient. A deep neural network–HMM model for acoustic speech recognition works much better than the traditional Gaussian mixture model–HMM (Hinton et al., 2012).

Another model for direct audio to text transcription, which used deep bidirectional LSTM RNN, achieved 27.3% word error rate on *The Wall Street Journal* corpus without any prior linguistic rule book (Graves & Jaitly, 2014; Graves & Schmidhuber, 2005).

Google Assistant and Amazon Echo are two real-life product examples where deep learning is used for speech recognition.

3.4 | Sound restoration in silent videos

Deep learning is not limited to speech recognition only; it is already in use to synthesize the artificial voice with zero manual control. Deep learning is used to synthesize audio for silent videos. Recently, an algorithm was introduced that uses RNN network to predict the features of a silent video, and these features are used to generate a wavefront with the help of an example-based generation method (Owens et al., 2015).

Another application of deep learning in this domain is lip reading. LipNet is a new model that uses a spatio-temporal convolutional neural network and RNN, to perform the end-to-end sentence level lip reading from a real-time scenario or video footage (Assael, Shillingford, Whiteson, & Freitas, 2016). This model achieved the accuracy of 95.2% on GRID corpus data set.

3.5 | Automatic handwriting and text generation

More recently, LSTM-based RNN demonstrated great success in establishing the relationship between strings or characters and then generates text or handwriting for a given phrase. Different handwriting learning styles could be mimicked and learned, which helped forensic science experts to solve mystery. Such recurrent network-based deep learning model is capable of learning how to spell, punctate, writing style capture, and the formation of sentences using character-driven model to generate various characters at a time.

3.6 | Prediction of demographics, earthquakes, and election results

Around 50 million street view images are captured and explored by Google to localize and recognize the car position in the city. The deep model is able to predict the demographics of every area where car haults based on car manufacture unit, model, body type, and year. On the basis of car density passed over an area, researcher predicted whether an individual driven a car voted for a demograhic or republican party in elections. The Harvard scientist discovered deep learning model that performed viscoelastic computation to predict the timing of earthquakes. Although this experiment was computationally expensive, it was useful, and improvement could be vital in saving life of people.

3.7 | Computer robotic games and self-driving cars

Google DeepMind utilized deep reinforcement learning techniques to teach a computer to play Atari game. Similarly, in other games such as Doom and Pong, deep learning models already outperformed players having vast experience of gaming. The computer utilizes the concept of reward from reinforcement learning and learns the game rule themselves after a few hours of game playing. Tesla's world-known invention self-driving car drives electric vehicle without human intervention. It is able to classify and distinguish various street signs, objects, and people moving around the street. Deep learning heavily indulged in robots such as Atlas and SpotMini developed by Boston Dynamics. These robots react to the situations performed by people such as pushing, get up when falling, taking care of daily human tasks, and cleaning cushions and sofa.

4 | CLASSIFICATION OF ACADEMIC JOURNALS

In this section, all the articles used in this study have been classified on the basis of different categories including the year of publication, journals, and conferences in which papers have been published and domain of the articles.

4.1 | Classification by year of publication

As shown in Figure 13, before 2006, the deep learning was not much popular as compared with other machine learning techniques used in artificial intelligence. One possible reason could be the lack of sophisticated GPUs for the training of networks. Deep learning techniques got a huge boost after 2,006.67% of all the research articles are from 2006 to 2015. After 2011, more sophisticated and domain-specific research papers were published as compared with earlier years.

4.2 | Classification by journals and conferences

Ninety nine journals and conference proceeding were selected for this study from various resources such as *Institute of Electrical and Electronics Engineers*, *Nature*, *Journal for Machine Learning Research*, *Neural Information Processing Systems* (*NIPS*), and many others. Figure 14 shows the detailed distribution of all journals and conferences.

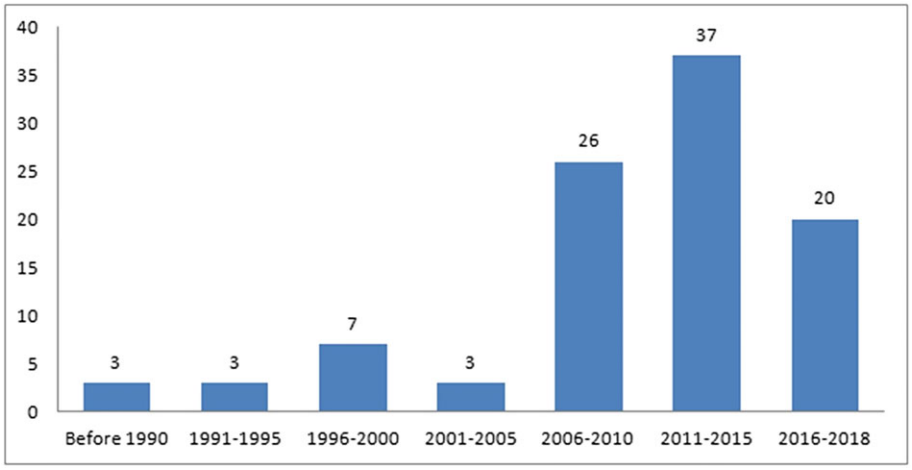

FIGURE 13 Year-wise classification of academic journals

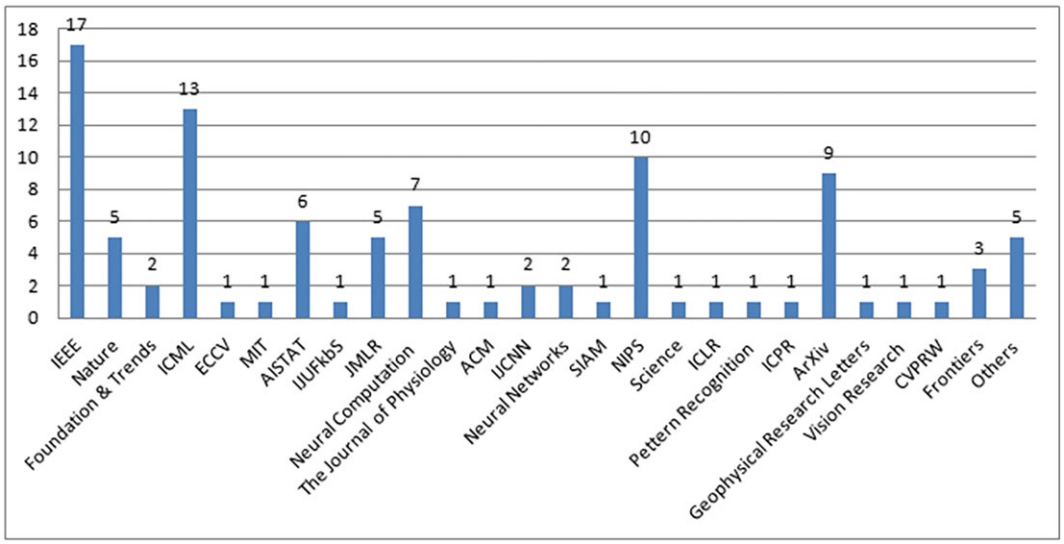

FIGURE 14 Classification by journals and conferences

As shown in Figure 14, Institute of Electrical and Electronics Engineers and proceedings of International Conference on Machine Learning have the major contributions for this meta-analysis on deep learning.

4.3 | Classification by architectures used in articles

Basic deep learning architectures are combined with one another to construct more robust architectures. For example, convolutional neural network and deep belief network are used to build a convolutional deep belief network. This section classifies the papers on the basis of architectures used in the papers. In few papers, more than two architectures are used to present a more robust approach. Figure 15 shows the detailed distribution of articles according to the architecture used in the execution.

5 | LIMITATIONS, OPEN PROBLEMS, AND FUTURE SCOPE

5.1 | Open problems and limitations in deep learning

The processing capabilities of the human brain are still beyond the reach of current deep learning. One possibility that the size of the deep network is still very less as compared with the size of the human brain (10^11 neurons). Few of the problems of deep learning are discussed below.

1. Deep learning models are trained on examples, means for a good result one needs a huge load of correct examples. To achieve the accuracy of >90% in recognition tasks, hundreds of samples are required. On the other hand, a human child after looking at a cat can recognize it with full accuracy. The amount of data remains a problem for good generalization (Bengio, 2012a; Testolin, Stoianov, & Zorzi, 2017).
2. The complexity of learning algorithm is expressed in terms of parameters required in the model; in case of deep learning, these go beyond millions in number. These complex models cannot be trained on simple machines. Many GPUs and machine clusters are required for it. This makes deep learning very expensive for solving the everyday problem.
3. These deep learning models are highly domain specific, means a single model cannot classify audio and images simultaneously without any change. A general purpose system is still not possible with current technology (Ngiam et al., 2011).
4. The two-stage training procedure is used in most architectures where the first step is the layer-by-layer training and the second step is global fine tuning of the model. This procedure is not appreciated in online modes because of the fear of local minima. Is there any better procedure than this?
5. There are two types of parameters in neural networks: simple (weights and bias) and hyperparameters (learning rate, momentum, and the number of hidden layers). Most of the time, there is no optimization for hyperparameters. How to optimize these hyperparameters instead of manual setting?
6. There is no specific procedure for the selection of number of hidden layers. The depth of a network is only changed according to the variation of final output. Therefore, the depth of network is still a question although its construction.

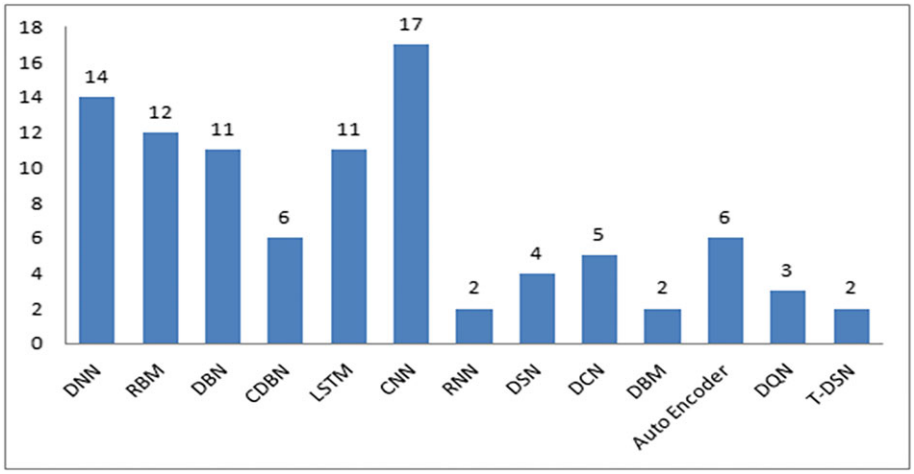

FIGURE 15 Classification by architectures used

7. Deep learning algorithms contradict with the "right to an explanation," means there is no easy or natural language way to explain why a particular prediction was selected by the algorithm (Goodman & Flaxman, 2016).

5.2 | Future of deep learning

1. The current networks produce a single final prediction for the input. For example, an image is classified into either yes or no for a category. But sometimes, the image is actually very confusing or damaged. In these cases, a network should produce an entire set of possible outcomes.
2. Networks are trained on thousands of images to detect objects from images. Current networks work on the structural properties of the objects. For example, classification of numbers 4 and 9 depends on their shape. But if generative models are trained on the physical construction properties on objects, the results will be much better.
3. Capsule architectures are the future of deep learning. The brain works in different regions, not in individual neurons. For example, there is a specific region of the brain, which helps to differentiate a category from others. These regions contain hundreds of neurons. The capsule architecture is a better way for routing information from lower layers to higher layers (Daneshzand, Ibrahim, Faezipour, & Barkana, 2017; Sabour, Frosst, & Hinton, 2017).
4. Generative models such as deep convolutional generative adversarial networks, these models are learning how the world actually looks like instead of memorizing the examples. In these models, there are two systems; one is generating the images, and other is evaluating them. The evaluator tries to distinguish the generated images from the true distribution. The goal is to fool the evaluator system to classify generated images into the true image.

6 | CONCLUSION

This study provides a comprehensive insight into architectural models available for deep learning. As compared with simple neural networks, deep neural networks are more robust in representing complex function and detecting features of the given input data. This study describes various models and the procedure for their training. Few models can be trained using more than one method, so their efficiency depends upon the domain in which they are used and the quality of the training data used. Various open-source data sets are available for different domains. But the quality of the data set is a huge concern. An overview of the few domains where deep learning can be used is given in this review paper. Deep learning can be used in any domain, but the cost of the sophisticated hardware is the concern. It is also observed from this study that deep learning is not limited to neural networks, but it is best explained with their help. This paper would be helpful for the students and researchers to gain insights of deep learning and their models to work in this field.

CONFLICT OF INTEREST

The authors have no conflict of interest to declare.

ORCID

Aditya Khamparia https://orcid.org/0000-0001-9019-8230

REFERENCES

Assael Y. M., Shillingford B., Whiteson S., & Freitas N. D. (2016). LipNet: Sentence-level lipreading. arXiv:1611.01599, 2016.

Bellman, R., & Lee, E. (1984). History and development of dynamic programming. *IEEE Control Systems Magazine*, *4*(4), 24–28. https://doi.org/10.1109/MCS.1984.1104824

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends® in Machine Learning*, *2*(1), 1–127. https://doi.org/10.1561/2200000006

Bengio, Y. (2012a). Deep learning of representations for unsupervised and transfer learning. In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, 4(3):17–36.

Bengio, Y. (2012b). Deep learning of representations for unsupervised and transfer learning. In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, USA, 1(1), 17–36.

Bengio, Y., Ducharme, R., Vincent, P., & Janvin, C. (2003). A neural probabilistic language model. *The Journal of Machine Learning Research*, *3*(2), 1137–1155.

Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. (2007). Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems*, 4-7 December, Canada, 153-160.

Bengio, Y., & LeCun, Y. (2007). Scaling learning algorithms towards AI. In L. Bottou, O. Chapelle, D. DeCoste, & J. Weston (Eds.), *Large scale kernel machines* (pp. 182–192). Cambridge, MA: MIT Press.

Cho, Y., & Saul, L. (2009). Kernel methods for deep learning. *Neural Information Processing Systems*, *8*(2), 342–350.

Coates A., Ng A., & Lee H. (2011). An analysis of single-layer networks in unsupervised feature learning. *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, June 14, USA, 215-223.

Courville, A., Bergstra, J., & Bengio, Y. (2010). Modeling natural image covariance with a spike and slab restricted Boltzmann machine. In *NIPS Workshop on Deep Learning*, 2010.

Courville, A., Bergstra, J., & Bengio, Y. (2011a). A spike and slab restricted Boltzmann machine. *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, *15*(1), 233–241.

Courville A., Bergstra J., & Bengio Y. (2011b). Unsupervised models of images by spike-and-slab RBMs, Proc. Int'l Conf. Machine Learning, USA, 22–29.

Courville, A., Desjardins, G., Bergstra, J., & Bengio, Y. (2014). The spike-and-slab RBM and extensions to discrete and sparse data distributions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *36*(9), 1874–1887. https://doi.org/10.1109/TPAMI.2013.238

Dahl R., Norouzi M., & Shlens J. (2017). Pixel recursive super resolution. *arXiv preprint arXiv:1702.00783*.

Daneshzand, M., Faezipour, M., & Barkana, B. D. (2017). Computational stimulation of the basal ganglia neurons with cost effective delayed Gaussian waveforms. *Frontiers in Computational Neuro Science*, *11*(8), 1013–1029.

Daneshzand, M., Ibrahim, S. A., Faezipour, M., & Barkana, B. D. (2017). Desynchronization and energy efficiency of Gaussian neurostimulation on different sites of the basal ganglia. 2017 IEEE 17th international conference on bioinformatics and bioengineering (BIBE) 2017 Oct 23, USA, 57–62.

Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., & Ng, A. Y. (2012). Large scale distributed deep networks. *Advances in Neural Information Processing Systems*, *34*(3), 1223–1231.

Deng, L. (2014). Deep learning: Methods and applications. *Foundations and Trends® in Signal Processing*, *7*(3), 197–387.

Deng, L., He, X., & Gao, J. (2013). Deep stacking networks for information retrieval, *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 26–31 may, Canada, 3153–3157.

Deng, L., Yu, D., & Platt, J. (2012). Scalable stacking and learning for building deep architectures, *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 25–30 march Kyoto, Japan, 2133–2136.

DeVries, P., Thompson, T., & Meade, B. (2017). Enabling large-scale viscoelastic calculations via neural network acceleration. *Geophysical Research Letters*, *44*(6), 2662–2669. https://doi.org/10.1002/2017GL072716

Gers, F., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. *Neural Computation*, *12*(10), 2451–2471. https://doi.org/10.1162/089976600300015015

Glorot, X., & Bengio Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Thirteenth International Conference on Artificial Intelligence and Statistics*, UK, *4*(1), 249–256.

Goodman, B., & Flaxman, S. (2016). European Union regulations on algorithmic decision-making and a right to explanation. arXiv preprint arXiv:1606.08813, 2016.

Graves A., & Jaitly N. (2014). Towards end-to-end speech recognition with recurrent neural networks, *31st International Conference on Machine Learning*, ICML, 5(3), 1764–1772.

Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM networks, *Proceedings of 2005 IEEE International Joint Conference on Neural Networks*,31-July-4 Aug, Canada, 602–610.

Hinton, G. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation*, *14*(8), 1771–1800. https://doi.org/10.1162/089976602760128018

Hinton, G. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*(5786), 504–507. https://doi.org/10.1126/science.1127647

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A., Jaitly, N., … Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, *29*(6), 82–97. https://doi.org/10.1109/MSP.2012.2205597

Hinton, G. E., Osindero, S., & The, Y. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*(7), 1527–1554. https://doi.org/10.1162/neco.2006.18.7.1527

Hinton, G. E., & Zemel, R. S. (1994). Autoencoders, minimum description length, and Helmholtz free energy. In D. Cowan, G. Tesauro, & J. Alspector (Eds.), *Advances in Neural Information Processing Systems 6 (NIPS'93)* (pp. 3–10). New Jersey, USA: Morgan Kaufmann Publishers, Inc.

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *6*(2), 107–116. https://doi.org/10.1142/S0218488598000094

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hubel, D. H., & Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, *160*(1), 106–154. https://doi.org/10.1113/jphysiol.1962.sp006837

Hutchinson, B., Deng, L., & Yu, D. (2013). Tensor deep stacking networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(8), 1944–1957. https://doi.org/10.1109/TPAMI.2012.268

Hwang, J., & Zhou, Y. (2016). Image colorization with deep convolutional neural networks. Stanford University, Tech. Rep.

Karpathy, A., & Fei-Fei, L. (2017). Deep visual-semantic alignments for generating image descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(4), 664–676. https://doi.org/10.1109/TPAMI.2016.2598339

Kolda, T., & Bader, B. (2009). Tensor decompositions and applications. *SIAM Review*, *51*(3), 455–500. https://doi.org/10.1137/07070111X

Larochelle, H., Bengio, Y., Louradour, J., & Lamblin, P. (2009). Exploring strategies for training deep neural networks. *Journal of Machine Learning Research*, *10*(1), 1–40.

Lawrence, S., Giles, C., Tsoi, A. C., & Back, A. (1997). Face recognition: A convolutional neural-network approach. *IEEE Transactions on Neural Networks*, *8*(1), 98–113. https://doi.org/10.1109/72.554195

Le, Q., Ngiam, J., Coates, A., Lahiri, A., Prochnow, B., & Ng, A. (2011). On optimization methods for deep learning, *Proceedings of the 28th International Conference on Machine Learning*, June 28–July 02, Washington, USA, 265–272.

LeCun, Y., & Bengio, Y. (1995). Convolutional networks for images, speech, and time-series. In *The handbook of brain theory and neural networks*, *3361*(10), 1995.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., & Jackel, L. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, *1*(4), 541–551. https://doi.org/10.1162/neco.1989.1.4.541

Lee, H., Grosse, R., Ranganath, R., & Ng, A., (2009). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations, *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*, Geneva, 265–274.

Lee, H., Grosse, R., Ranganath, R., & Ng, A. (2011). Unsupervised learning of hierarchical representations with convolutional deep belief networks. *Communications of the ACM*, *54*(10), 95–102. https://doi.org/10.1145/2001269.2001295

Lin, Y., Zhang, T., Zhu, S., & Yu, K. (2010). Deep coding network. In *Advances in Neural Information Processing Systems* (pp. 1405–1413). USA: Springer.

Mitrovic, J., Sejdinovic, D., & The, Y. W. (2010). Deep kernel machines via the kernel reparametrization trick. In *ICLR Workshop*.

Ng, A. (2011). Sparse autoencoder. *CS294A Lecture Notes*, *72*(2011), 1–19.

Ng A. (2015). What data scientists should know about deep learning, 44.

Ngiam, J., Khosla, A., Kim, M., Nam, J., Lee, H., & Ng, A. Y. (2011). Multimodal deep learning. *In Proceedings of the 28th international conference on machine learning (ICML-11)*, June-28- July 02, USA, 689-696.

Owens, A., Isola, P., McDermott, J., Torralba, A., Adelson, E., & Freeman, W. (2015). Visually indicated sounds. *arXiv preprint*, *arXiv:* 1512.08512, 2015.

Raina, R., Madhavan, A., & Ng, A. Y. (2009). Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the 26th annual international conference on machine learning*, Geneva, *26*(3), 873–880, ACM.

Ren, Y., & Wu, Y. (2014). Convolutional deep belief networks for feature extraction of EEG signal, *2014 International Joint Conference on Neural Networks (IJCNN)*, 6–11 July, Beijing China, 2850–2853.

Riesenhuber, M., & Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nature Neuroscience*, *2*(11), 1019–1028. https://doi.org/10.1038/14819

Sabour, S., Frosst, N., & Hinton, G. E. (2017). Dynamic routing between capsules. *In Advances in Neural Information Processing Systems*, *92*, 3859–3869.

Salakhutdinov, R., & Hinton, G. (2009). Deep Boltzmann machines, in *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, 16–18 April, Florida, USA, 448–455.

Salakhutdinov, R. & Larochelle, H. (2009). Efficient learning of deep Boltzmann machines, in *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, 16-18 April., Florida, USA, 693-700.

Salakhutdinov, R., & Murray, I. (2008). On the quantitative analysis of deep belief networks, *Proceedings of the 25th international conference on Machine learning - ICML '08*, USA, 10–20.

Schölkopf, B., Smola, A., & Müller, K. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, *10*(5), 1299–1319. https://doi.org/10.1162/089976698300017467

Srivastava, N., & Salakhutdinov, R. R. (2014). Multimodal learning with deep Boltzmann machines. In *Advances in neural information processing systems*, 2222–2230.

Testolin, A., De Filippo De Grazia, M., & Zorzi, M. (2017). The role of architectural and learning constraints in neural network models: A case study on visual space coding. *Frontiers in Computational Neuroscience*, *11*(13). https://doi.org/10.3389/fncom.2017.00013

Testolin, A., Stoianov, I., & Zorzi, M. (2017). Letter perception emerges from unsupervised deep learning and recycling of natural image features. *Nature Human Behaviour*, *1*(9), 657–668. https://doi.org/10.1038/s41562-017-0186-2

Testolin, A., & Zorzi, M. (2016). Probabilistic models and generative neural networks: Towards an unified framework for modeling normal and impaired neurocognitive functions. *Frontiers in Computational Neuroscience*, *10*(73), 101–112.

Varga, D., & Sziranyi, T. (2016). Fully automatic image colorization based on convolutional neural network, *23rd International Conference on Pattern Recognition (ICPR)*, 4–8 December, Mexico, 3691–3696.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., & Manzagol, P. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, *11*(2), 3371–3408.

Weinberger, K., & Saul, L. (2009). Distance metric learning for large margin nearest neighbor classification. *The Journal of Machine Learning Research*, *10*(3), 207–244.

Weisstein, E. (2012). Symmetric bilinear form, MathWorld and Wikipedia.

Wilson, A. G., Hu, Z., Salakhutdinov, R., & Xing, E. P. (2016). Deep kernel learning. *Proceedings of 19th International Conference on Artificial Intelligence and Statistics*, 51(2), 370–378.

Wolpert, D. (1992). Stacked generalization. *Neural Networks*, *5*(2), 241–259. https://doi.org/10.1016/S0893-6080(05)80023-1

Yamins, D. L., & DiCarlo, J. J. (2016). Using goal-driven deep learning models to understand sensory cortex. *Nature Neuroscience*, *1*(3), 356–365.

Yu, K., & Zhang, T. (2010). Improved local coordinate coding using local tangents. In *27th International Conference on International Conference on Machine Learning*, 21–28 June, Israel, 1215–1222.

Zeiler, M., & Fergus, R. (2014). Visualizing and understanding convolutional networks. *Computer Vision – ECCV 2014*, *2*(4), 818–833.

Zhang, S., Wang, J., Tao, X., Gong, Y., & Zheng, N. (2017). Constructing deep sparse coding network for image classification. *Pattern Recognition*, *64*(3), 130–140. https://doi.org/10.1016/j.patcog.2016.10.032

Zorzi, M., Testolin, A., & Stoianov, I. P. (2013). Modeling language and cognition with deep unsupervised learning: A tutorial overview. *Frontiers in Psychology*, *4*(1), 515–527.

AUTHOR BIOGRAPHIES

Dr. Aditya Khamparia is serving as an academician and research person from past 5 years. Currently, he is working as an assistant professor of computer science at Lovely Professional University, Punjab, India. He was awarded with Ph.D. in Computer Science from the Lovely Professional University, India. His research area is machine learning, soft computing, educational technologies, IoT, semantic web, and ontologies. He has published more than 35 scientific research publications in reputed international/national journals and conferences, which are indexed in various international databases. He invited as a faculty resource person/session chair/reviewer/TPC member in different FDP, conferences, and journals. Dr. Aditya received research excellence award in 2016, 2017, and 2018 at Lovely Professional University for his research contribution during the academic year. He is a member of CSI, IET, ISTE, IAENG, ACM, and IACSIT. He is also acting as a reviewer and a member of various renowned national and international conferences/journals.

Karan M. Singh is a research scholar and currently pursuing Master's Degree in Computer Science and Engineering from Lovely Professional University, Punjab, India. His area of research is deep learning architectures and machine learning.

How to cite this article: Khamparia A, Singh KM. A systematic review on deep learning architectures and applications. *Expert Systems*. 2019;e12400. https://doi.org/10.1111/exsy.12400