IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

A SURVEY ON FEED FORWARD NEURAL NETWORKS AND DEEP LEARNING ARCHITECTURES

[1]Prof. Anasuya N Jadagerimath, [2]Prof. Yogesh G. S, [3]Puneeth Kumar P,[4]Mohan Kumar A.V

[1]Professor, [2]Professor, [3]Assistant Professor, [4]Assistant Professor

[1]Computer Science and Engineering (AIML)Department, [2]Electronics and Communication Engineering Department, [3]Computer Science and Engineering(AIML) Department

t[3,4]Don Bosco Institute of Technology Bengaluru-India, [2] East Point College of Engineering and Technology Bengaluru-India,

Abstract— *Feed Forward Neural Network is a part of machine learning based on a set of algorithms that attempt to model high-level abstractions in data by using multiple processing layers with complex structures learning refers to a rather wide class of machine learning techniques and architectures, with the hallmark of using many layers of non-linear information processing that are hierarchical in nature. Here we are discussing about the comparative study of Feed Forward Neural Network*

Index Terms - Artificial Intelligence, Machine learning, Neural network

I. INTRODUCTION

Deep learning is a recently-developed field belonging to Artificial Intelligence. It tries to mimic the human brain, which is capable of processing the complex input data, learning different knowledge's intellectually and fast, and solving different kinds of complicated tasks well. Switching these features of human brain to a learning model, we wish the model can deal with the high-dimensional data, support a fast and intellectual learning algorithm and perform well in the complicated AI tasks like computer vision or speech recognition. Deep architecture is believed to be such kind of model, with good learning algorithms for the deep learning and an excellent performance in solving AI tasks. This paper reviews a history of deep learning, summarizing the components of Convolution Neural Networks (CNNs) and Deep Belief Networks (DBNs) together with their learning algorithms "signal" or "feedback" available to a learning system. These are:[10] and their performances in different applications. Machine learning tasks are typically classified into three broad categories, depending on the nature of the learning and their performances in different applications. Machine learning tasks are typically classified into three broad categories, depending on the nature of the learning

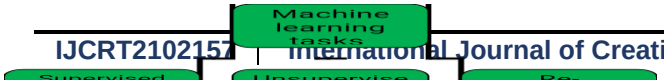

Fig 1 :Types Of Machine Learning Tasks

II . TYPES OF MACHINE LEARNING TASKS

Supervised learning: The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs. Unsupervised learning: No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end.Reinforcement learning: A computer program interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle), without a teacher explicitly telling it whether it has come close to its goal. Another example is learning to play a game by playing against an opponent[3]:Between supervised and unsupervised learning is semi-supervised learning, where the teacher gives an incomplete training signal: a training set with some (often many) of the target outputs missing. Transduction is a special case of this principle where the entire set of problem instances is known at learning time, except that part of the targets are missing.

III. MACHINE LEARNING METHODS

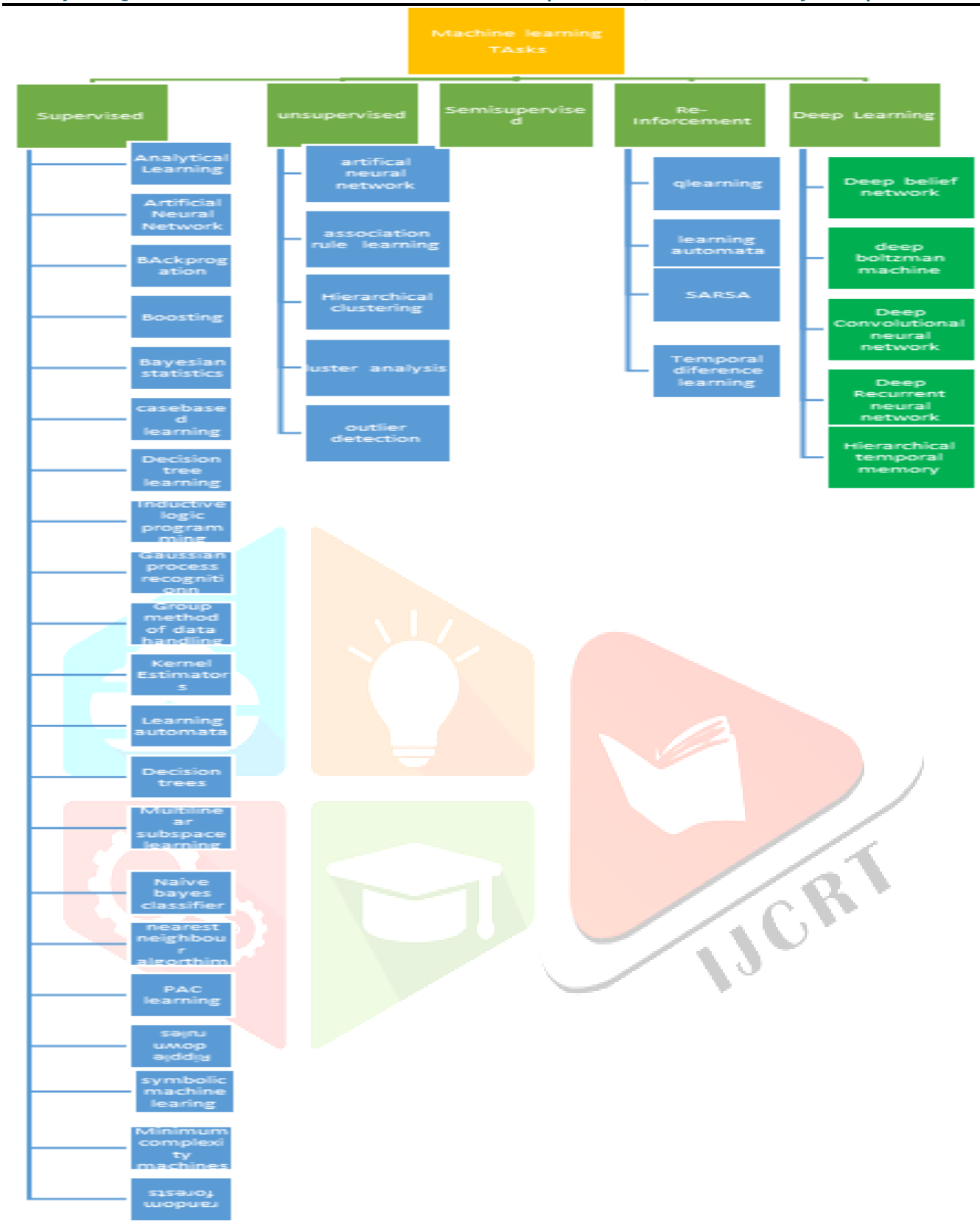

Fig 2:Machine Learning Methods

IV DEEP LEARNING

Deep learning is a branch of machine learning based on a set of algorithms that attempt to model high-level abstractions in data by using multiple processing layers with complex structures, or otherwise composed of multiple non-linear transformations. [1][2][3][4][5][6]Deep learning is part of a broader family of machine learning methods based on learning representations of data. An observation (e.g., an image) can be represented in many ways such as a vector of intensity values per pixel, or in a more abstract way as a set of edges, regions of particular shape, etc. [4]Various deep learning architectures such as deep neural networks, convolutional deep neural networks, deep belief networks and recurrent neural networks have been applied to fields like computer vision, automatic speech recognition, natural language processing, audio recognition and bioinformatics where they have been shown to produce state-of-the-art results on various tasks.[7]Three important reasons for the popularity of deep learning today are drastically increased chip processing abilities (e.g., GPU units), the significantly lowered cost of computing hardware, and recent advances in machine learning and signal/information processing research. Active researchers in this area include those at University of Toronto, New York University, University of Montreal, Microsoft Research, Google, IBM Research, Stanford University, and University of Michigan, Massachusetts Institute of Technology, University of Washington, and numerous other places. These researchers have demonstrated successes of deep learning in diverse applications of computer vision, phonetic recognition, voice search, conversational speech recognition, speech and image feature coding, semantic utterance classification, hand-writing recognition, audio processing, visual object recognition, information retrieval, and even in the analysis of molecules that may lead to discovering new drugs as reported recently in (Mark off, 2012).

FEED FORWARD NEURAL NETWORK

A **feed forward neural network** is an artificial neural network where connections between the units do *not* form a cycle. This is different from recurrent neural networks. The feed forward neural network was the first and simplest type of artificial neural network devised. In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network. In a feed forward network information always moves one direction; it never goes backwards.

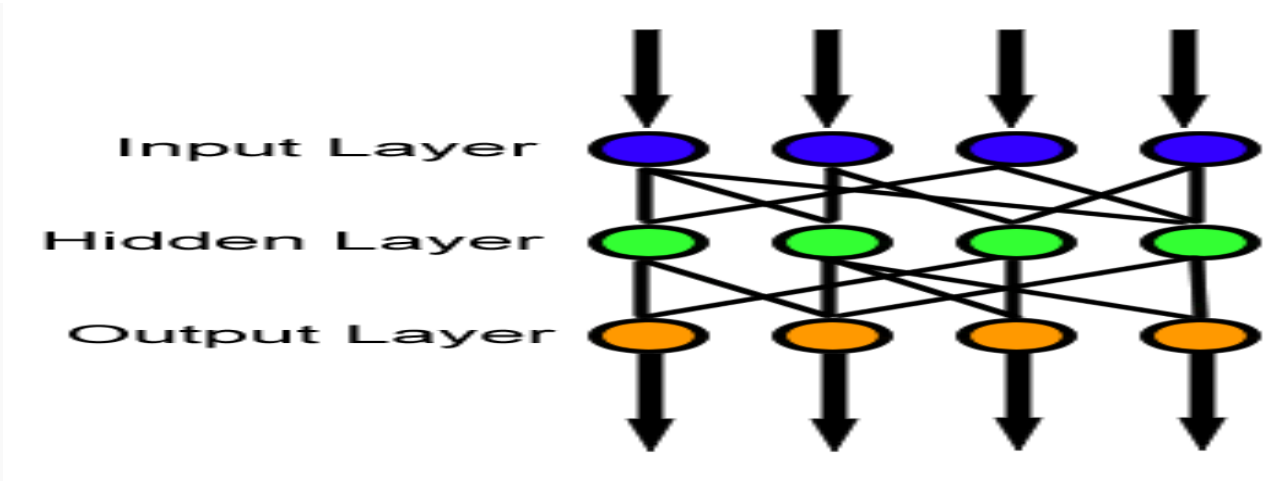

Fig 3: Feed forward neural network

BACK PROPAGATION ALGORITHM

Back propagation is a common method of training artificial neural networks used in conjunction with an optimization method such as gradient descent. The method calculates the gradient of a loss function with respect to all the weights in the network. The gradient is fed to the optimization method which in turn uses it to update the weights, in an attempt to minimize the loss function.Back propagation requires a known, desired output for each input value in order to calculate the loss function gradient. It is therefore usually considered to be a supervised learning method, although it is also used in some unsupervised networks such as auto encoders. It is a generalization of the delta rule to multi-layered feed forward networks, made possible by using the chain rule to iteratively compute gradients for each layer. Back propagation requires that the activation function used by the artificial neurons (or "nodes") be differentiable.The back propagation learning algorithm can be divided into two phases: propagation and weight update.

1) *Phase 1: Propagation*

2) *Each propagation involves the following steps:*

Forward propagation of a training pattern's input through the neural network in order to generate the propagation's output activations.
Backward propagation of the propagation's output activations through the neural network using the training pattern target in order to generate the deltas (the difference between the input and output values) of all output and hidden neurons.

3) *Phase 2: Weight update*

For each weight-synapse follow the following steps:

Multiply its output delta and input activation to get the gradient of the weight. Subtract a ratio (percentage) of the gradient from the weight. This ratio (percentage) influences the speed and quality of learning; it is called the *learning rate*. The greater the ratio, the faster the neuron trains; the lower the ratio, the more accurate the training is. The sign of the gradient of a weight indicates where the error is increasing, this is why the weight must be updated in the opposite direction. Repeat phase 1 and 2 until the performance of the network is satisfactory.

Algorithm

The following is a stochastic gradient descent algorithm for training a three-layer network (only one hidden layer):

initialize network weights (often small random values)
do
  forEach training example ex
    prediction = neural-net-output(network, ex) *// forward pass*
    actual = teacher-output(ex)
    compute error (prediction - actual) at the output units
    compute Δw_h for all weights from hidden layer to output layer *// backward pass*
    compute Δw_i for all weights from input layer to hidden layer *// backward pass continued*
    update network weights *// input layer not modified by error estimate*
until all examples classified correctly or another stopping criterion satisfied
return the network

The lines labeled "backward pass" can be implemented using the back propagation algorithm, which calculates the gradient of the error of the network regarding the network's modifiable weights.[2] Often the term "back propagation" is used in a more general sense, to refer to the entire procedure encompassing both the calculation of the gradient and its use in stochastic gradient descent, but back propagation proper can be used with any gradient-based optimizer, such as L-BFGS or truncated Newton. Back propagation networks are necessarily multilayer perceptron's (usually with one input, multiple hidden, and one output layer). In order for the hidden layer to serve any useful function, multilayer networks must have non-linear activation functions for the multiple layers: a multilayer network using only linear activation functions is equivalent to some single layer, linear network. Non-linear activation functions that are commonly used include the rectifier, logistic function, the softmax function, and the Gaussian function. The back propagation algorithm for calculating a gradient has been rediscovered a number of times, and is a special case of a more general technique called automatic differentiation in the reverse accumulation mode.

V METHODS OF UNSUPERVISED DEEP LEARNING ARCHITECTURE

There are huge number of different variants of deep architectures. Deep learning is a fast-growing field so new architectures, variants, or algorithms may appear every few weeks.

4) Deep Neural Networks

A deep neural network (DNN) is an artificial neural network (ANN) with multiple hidden layers of units between the input and output layers.[11] DNNs can model complex non-linear relationships. DNN architectures, e.g., for object detection and parsing generate compositional models where the object is expressed as layered composition of image primitives. The extra layers enable composition of features from lower layers, giving the potential of modeling complex data with fewer units than a similarly performing shallow network.[12]

DNNs are typically designed as feed forward networks, but recent research has successfully applied the deep learning architecture to recurrent neural networks for applications such as language modeling.

The training DNNs is explained below:

A DNN can be discriminatively trained with the standard back propagation algorithm. The weight updates can be done via stochastic gradient descent using the following equation:

$$w_{ij}(t+1) = w_{ij}(t) + \eta \frac{\partial C}{\partial w_{ij}} \quad (1)$$

Here, η is the learning rate, and C is the cost function. The choice of the cost function depends on factors such as the learning type (supervised, unsupervised, reinforcement, etc.) and the activation function.

For example, when performing supervised learning on a multiclass classification problem, common choices for the activation function and cost function are the softmax function and cross entropy function, respectively. The softmax function is defined as $p_j = \frac{\exp(x_j)}{\sum_k \exp(x_k)}$ where p_j represents the class probability (output of the unit j) and x_j and x_k (2)

represent the total input to units j and k of the same level respectively. Cross entropy is defined as $C = -\sum_j d_j \log(p_j)$ where d_j represents the target probability for output unit j and p_j is the probability output for j after applying the activation function.

These can be used to output object bounding boxes in form of a binary mask. They are also used for multi scale regression to increase localization precision. DNN-based regression can learn features that capture geometric information in addition to being a good classifier. They remove the limitation of designing a model which will capture parts and their relations explicitly. This helps to learn a wide variety of objects. The model consists of multiple layers each of which has a rectified linear unit for non-linear transformation. Some layers are convolution, while others are fully connected. Every convolutional layer has an additional max pooling. The network is trained to minimize L2 error for predicting the mask ranging over the entire training set containing bounding boxes represented as masks.

Two common issues in DNN are over fitting and computation time.

DNNs are prone to over fitting because of the added layers of abstraction, which allow them to model rare dependencies in the training data. Regularization methods such as weight decay (ℓ_2-regularization) or sparsity (ℓ_1-regularization) can be applied during training to help combat over fitting A more recent regularization method applied to DNNs is dropout regularization. In dropout, some number of units are randomly omitted from the hidden layers during training. This helps to break the rare dependencies that can occur in the training data Error-correction training (such as back propagation with gradient descent) have been the dominant method for training these structures due to the ease of implementation and their tendency to converge to better local optima in comparison with other training methods. However, these methods can be computationally expensive, especially when being used to train DNNs. There are many training parameters to be considered with a DNN, such as the size (number of layers and number of units per layer), the learning rate and initial weights. Sweeping through the parameter space for optimal parameters may not be feasible due to the cost in time and computational resources.

5) *Deep Belief Networks:*

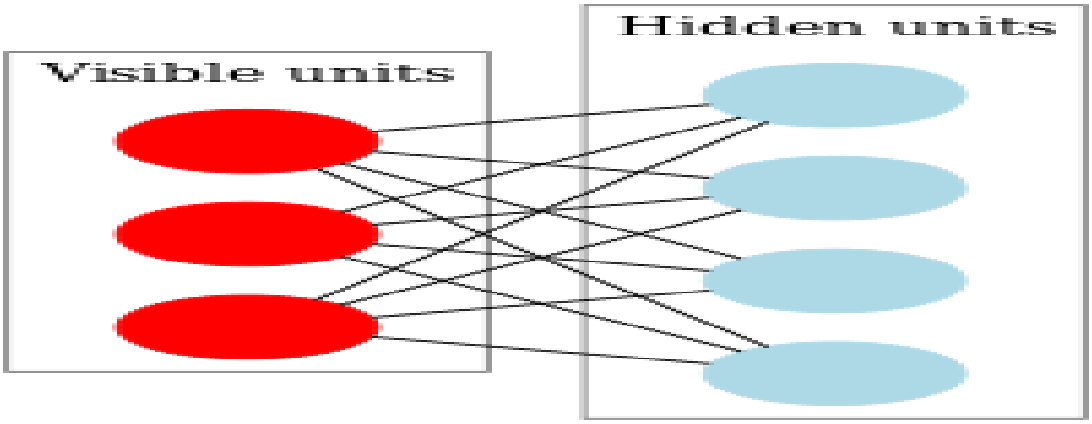

Fig 4: A restricted Boltzmann machine (RBM) with fully connected visible and hidden units.

A deep belief network (DBN) is a probabilistic, generative model made up of multiple layers of hidden units. It can be looked at as a composition of simple learning modules that make up each layerA DBN can be used for generatively pre-training a DNN by using the learned weights as the initial weights. Back propagation or other discriminative algorithms can then be applied for fine-tuning of these weights. This is particularly helpful in situations where limited training data is available, as poorly initialized weights can have significant impact on the performance of the final model. These pre-trained weights are in a region of the weight space that is closer to the optimal weights this allows for both improved modeling capability and faster convergence of the fine-tuning phase. A DBN can be efficiently trained in an unsupervised, layer-by-layer manner where the layers are typically made of restricted Boltzmann machines (RBM). A description of training a DBN via RBMs is provided below. An RBM is an undirected, generative energy-based model with an input layer and single hidden layer. Connections only exist between the visible units of the input layer and the hidden units of the hidden layer; there are no visible-visible or hidden-hidden connections. The training method for RBMs was initially proposed by Geoffrey Hinton for use with training "Product of Expert" models and is known as contrastive divergence (CD).[86] CD provides an approximation to the maximum likelihood method that would ideally be applied for learning the weights of the RBM.[80][87]In training a single RBM, weight updates are performed with gradient ascent via the following equation:

$$\Delta w_{ij}(t+1) = w_{ij}(t) + \eta \frac{\partial \log(p(v))}{\partial w_{ij}} \tag{3}$$

Here, $p(v)$ is the probability of a visible vector, which is given by $p(v) = \frac{1}{Z}\sum_h e^{-E(v,h)}$. Z is the partition function (used for normalizing) and $E(v,h)$ is the energy function assigned to the state of the network. A lower energy indicates the network is in a more "desirable" configuration. The gradient $\frac{\partial \log(p(v))}{\partial w_{ij}}$ has the simple form $\langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{model}}$ where $\langle \cdot \cdot \cdot \rangle_p$ represent averages with respect to distribution p. The issue arises in sampling $\langle v_i h_j \rangle_{\text{model}}$ as this requires running alternating Gibbs sampling for a long time. CD replaces this step by running alternating Gibbs sampling for n steps (values of $n = 1$ have empirically been shown to perform well). After n steps, the data is sampled and that sample is used in place of $\langle v_i h_j \rangle_{\text{model}}$.

The CD procedure works as follows:

Initialize the visible units to a training vector.

Update the hidden units in parallel given the visible units: $p(h_j = 1 \mid \mathbf{V}) = \sigma(b_j + \sum_i v_i w_{ij})$. σ Represents the sigmoid function and b_j is the bias of h_j.

Update the visible units in parallel given the hidden units: $p(v_i = 1 \mid \mathbf{H}) = \sigma(a_i + \sum_j h_j w_{ij})$. a_i is the bias of v_i. This is called the "reconstruction" step.

Reupdate the hidden units in parallel given the reconstructed visible units using the same equation as in step 2.

Perform the weight update: $\Delta w_{ij} \propto \langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{reconstruction}}$.

Once an RBM is trained, another RBM can be "stacked" atop of it to create a multilayer model. Each time another RBM is stacked, the input visible layer is initialized to a training vector and values for the units in the already-trained RBM layers are assigned using the current weights and biases. The final layer of the already-trained layers is used as input to the new RBM. The new RBM is then trained with the procedure above, and then this whole process can be repeated until some desired stopping criterion is met.[2]

Despite the approximation of CD to maximum likelihood being very crude (CD has been shown to not follow the gradient of any function), empirical results have shown it to be an effective method for use with training deep architectures.

6) Convolutional Neural Networks

A CNN is composed of one or more convolutional layers with fully connected layers (matching those in typical artificial neural networks) on top. It also uses tied weights and pooling layers. This architecture allows CNNs to take advantage of the 2D structure of input data. In comparison with other deep architectures, convolutional neural networks are starting to show superior results in both image and speech applications. They can also be trained with standard back propagation. CNNs are easier to train than other regular, deep, feed-forward neural networks and have many fewer parameters to estimate, making them a highly attractive architecture to use. Examples of applications in Computer Vision include Deep Dream.

7) Convolutional Deep Belief Networks

A recent achievement in deep learning is from the use of convolutional deep belief networks (CDBN). A CDBN is very similar to normal Convolutional neural network in terms of its structure. Therefore, like CNNs they are also able to exploit the 2D structure of images combined with the advantage gained by pre-training in Deep belief network. They provide a generic structure which can be used in many image and signal processing tasks and can be trained in a way similar to that for Deep Belief Networks. Recently, many benchmark results on standard image datasets like CIFAR have been obtained using CDBNs.

8) Large Memory Storage And Retrieval (Lamstar) Neural Networks

LAMSTAR (Large Memory Storage And Retrieval) Neural Networks are fast Deep Learning neural networks of many layers and which may employ numerous filters simultaneously. These filters may be nonlinear, stochastic, logic, non-stationary, or even non-analytical. The LAMSTAR is a biologically motivated continuously-learning neural network.

There is no connection between the units of the same layer (like RBM). For the DBM, we can write the probability which is assigned to vector ν as: (4)

$$p(\boldsymbol{\nu}) = \frac{1}{Z} \sum_{h} e^{\sum_{ij} W_{ij}^{(1)} \nu_i h_j^{(1)} + \sum_{jl} W_{jl}^{(2)} h_j^{(1)} h_l^{(2)} + \sum_{lm} W_{lm}^{(3)} h_l^{(2)} h_m^{(3)}},$$

where $\boldsymbol{h} = \{\boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)}, \boldsymbol{h}^{(3)}\}$ are the set of hidden units, and $\theta = \{\boldsymbol{W}^{(1)}, \boldsymbol{W}^{(2)}, \boldsymbol{W}^{(3)}\}$ are the model parameters, representing visible-hidden and hidden-hidden *symmetric interaction*, since they are undirected links. As it is clear by setting $\boldsymbol{W}^{(2)} = 0$ and $\boldsymbol{W}^{(3)} = 0$ the network becomes the well-known Restricted Boltzmann machine.[111]

There are several reasons which motivate us to take advantage of deep Boltzmann machine architectures. Like DBNs, they benefit from the ability of learning complex and abstract internal representations of the input in tasks such as object or speech recognition, with the use of *limited number* of *labeled* data to fine-tune the representations built based on a large *supply* of *unlabeled* sensory input data.

Since the *exact maximum likelihood* learning is intractable for the DBMs, we may perform the *approximate maximum likelihood* learning. There is another possibility, to use *mean-field* inference to estimate data-dependent expectations, incorporation with a *Markov chain Monte Carlo (MCMC)* based stochastic approximation technique to approximate the expected *sufficient statistics* of the model.[12]We can see the difference between DBNs and DBM. In DBNs, the top two layers form a restricted Boltzmann machine which is an undirected graphical model, but the lower

layers form a directed generative model. Apart from all the advantages of DBMs discussed so far, they have a crucial disadvantage which limits the performance and functionality of this kind of architecture. The approximate inference, which is based on mean-field method, is about 25 to 50 times slower than a single bottom-up pass in DBNs. This time consuming task make the joint optimization, quite impractical for large data sets, and seriously restricts the use of DBMs in tasks such as feature representations (the mean-field inference have to be performed for each new test input)

Table 1: Comparison of unsupervised deep learning architecture

Sl .No	Architecture	Type of Machine Learning Task	Designed as	Algorithm used	Formula used for Weight calculation	Applications
1	*Deep Neural Networks*	Unsupervised	Feed forward	Back propagation	Wij (t+1)=wij(t)+n∂C Gradient descendent $w_{ij}(t+1) = w_{ij}(t) + \eta \frac{\partial C}{\partial w_{ij}} \eta$ is the learning rate, and C is the cost function.	Language modeling. Automatic speech recognition
2	*Deep belief networks*	Unsupervised	Restricted Boltzmann Machines (Rbm)	Gradient-Based **Contrastive Divergence**	as contrastive divergence (CD). ere, $p(v)$ is the probability of a visible vector, which is given by $p(v) = \frac{1}{Z} \sum_h e^{-E(v,h)}$. Z is the partition function (used for normalizing) and $E(v,h)$ is the energy function assigned to the state of the network.	Dimensionality reduction classification, collaborative filtering feature learning[5] topic modelling.
3	*Convolutional neural networks*	Unsupervised	feed-forward artificial neural network	Back propagation	*Back propagation* $Q(w) = \sum_{i=1}^{n} Q_i(w),$ where the parameter w^* which minimizes $Q(w)$ is to be estimated. Each summand function Q_i is s typically associated with the i-th observation in the data set (used for training).	*Image recognition* *Video analysis* *Natural language processing* Drug discovery
4	*Large Memory Storage and Retrieval (LAMSTAR) neural networks*	Unsupervised	fast deep learning neural network	Back propagation	$w(n + 1) = w(n) + \alpha$ $*(x - w(n))$ where, α — learning constant = 0.8 w — weight at the input of the neuron x — subword $z = w*X$ **Input Normalization:** Each subwords of every input pattern is normalized as follows: $xi-= xi$ $\Sigma x2j$ where, x — sub word of an input pattern. During the process, those sub words, which are all zeros, are identified and their normalized values are manually set to zero.	Image recognition[Computer-based medical diagnosis system Fault diagnosis

VI DEEP LEARNING SOFTWARE LIBRARIES

TensorFlow — Google's open source machine learning library in C++ and Python with APIs for both. It provides parallelization with CPUs and GPUs.[231]

Torch — An open source software library for machine learning based on the Lua programming language.

Theano — An open source machine learning library for Python.

Deeplearning4j — An open source deep learning library written for Java. It provides parallelization with CPUs and GPUs.

OpenNN — An open source C++ library which implements deep neural networks and provides parallelization with CPUs.

NVIDIA cuDNN — A GPU-accelerated library of primitives for deep neural networks.

DeepLearnToolbox — A Matlab/Octave toolbox for deep learning.

convnetjs — A Javascript library for training deep learning models. It contains online demos.

Gensim — A toolkit for natural language processing implemented in the Python programming language.

Caffe — A deep learning framework.

Apache SINGA — A deep learning platform developed for scalability, usability and extensibility.

RNNLM — RNN language model open source.
RNNLMPara — Parallel RNN language model trainer open source.

VI *DEEP LEARNING RESEARCH GROUPS*

Some labs and research groups that are actively working on deep learning:
University of Toronto - **Machine Learning Group** (Geoffrey Hinton, Rich Zemel, Ruslan Salakhutdinov, Brendan Frey, Radford Neal)
Université de Montréal – **MILA Lab** (Yoshua Bengio, Pascal Vincent, Aaron Courville, Roland Memisevic)
New York University – **Yann Lecun**, **Rob Fergus**, David Sontag and Kyunghyun Cho
Stanford University – **Andrew Ng**, **Christopher Manning**'s, **Fei-fei Li**'s group
University of Oxford – **Deep learning group**, **Nando de Freitas** and **Phil Blunsom**, Andrew Zisserman
Google Research – Jeff Dean, Geoffrey Hinton, Samy Bengio, Ilya Sutskever, Ian Goodfellow, Oriol Vinyals, Dumitru Erhan, Quoc Le et al
Google DeepMind - Alex Graves, Karol Gregor, Koray Kavukcuoglu, Andriy Mnih, Guillaume Desjardins, Xavier Glorot, Razvan Pascanu, Volodymyr Mnih et al
Facebook AI Research(FAIR) - Yann Lecun, Rob Fergus, Jason Weston, Antoine Bordes, Soumit Chintala, Leon Bouttou, Ronan Collobert, Yann Dauphin et al.
Twitter's Deep Learning Group – Hugo Larochelle, Ryan Adams, Clement Farabet et al
Microsoft Research – **Li Deng** et al
SUPSI – **IDSIA** (**Jurgen Schmidhuber**'s group)
UC Berkeley – **Bruno Olshausen**'s group, **Trevor Darrell**'s group, **Pieter Abbeel**
UCLA – **Alan Yuille**
University of Washington – **Pedro Domingos**' group
IDIAP Research Institute - **Ronan Collobert**'s group
University of California Merced – **Miguel A. Carreira-Perpinan**'s group
University of Helsinki - **Aapo Hyvärinen**'s Neuroinformatics group
Université de Sherbrooke – **Hugo Larochelle**'s group
University of Guelph – **Graham Taylor**'s group
University of Michigan – **Honglak Lee**'s group
Technical University of Berlin – **Klaus-Robert Muller**'s group
Baidu – **Kai Yu**'s and Andrew Ng's group
Aalto University - **Juha Karhunen and Tapani Raiko** group
U. Amsterdam – **Max Welling**'s group
CMU – **Chris Dyer**
U. California Irvine – **Pierre Baldi**'s group
Ghent University – **Benjamin Shrauwen**'s group
University of Tennessee – **Itamar Arel**'s group
IBM Research – **Brian Kingsbury** et al
University of Bonn – **Sven Behnke's** group
Gatsby Unit @ University College London – Maneesh Sahani, Peter Dayan**Computational Cognitive Neuroscience Lab** @ University of Colorado Boulder

VII CONCLUSION

Given the far-reaching implications of artificial intelligence coupled with the realization that deep learning is emerging as one of its most powerful techniques, the subject is understandably attracting both criticism and comment, and in some cases from outside the field of computer science itself. Others point out that deep learning should be looked at as a step towards realizing strong AI, not as an all-encompassing solution. Despite the power of deep learning methods, they still lack much of the functionality needed for realizing this goal entirely. Some currently popular and successful deep learning architectures display certain problematical behaviors. The researchers hypothesized that these behaviors are tied with limitations in the internal representations learned by these architectures, and that these same limitations would inhibit integration of these architectures into heterogeneous multi-component AGI architectures. It is suggested that these issues can be worked around by developing deep learning architectures that internally form states homologous to image-grammar decompositions of observed entities and events.

References

[1] Deng, L.; Yu, D. (2014). "Deep Learning: Methods and Applications" (PDF). Foundations and Trends in Signal Processing 7: 3–4.
[2] Bengio, Yoshua (2009). "Learning Deep Architectures for AI" (PDF). Foundations and Trends in Machine Learning 2 (1): 1–127.
[3] Bengio, Y.; Courville, A.; Vincent, P. (2013). "Representation Learning: A Review and New Perspectives".IEEE Transactions on Pattern Analysis and Machine Intelligence 35 (8): 1798–1828. arXiv:1206.5538.
[4] Schmidhuber, J. (2015). "Deep Learning in Neural Networks: An Overview". Neural Networks 61: 85–117. arXiv:1404.7828.doi:10.1016/j.neunet.2014.09.003.
[5] Bengio, Yoshua; LeCun, Yann; Hinton, Geoffrey (2015)."Deep Learning". Nature 521: 436–444.doi:10.1038/nature14539.
[6] Three Classes of Deep Learning Architectures and Their Applications: A Tutorial Survey Li Deng Microsoft Research, Redmond, WA 98052, USA E-mail: deng@microsoft.com, Tel: 425-706-2719
[7] D, Graupe, H. Kordylewski, (1996), "Network based on SOM (self-organizing-map) modules combined with statistical decision tools", Proc. IEEE 39th Midwest Conf. on Circuits and Systems, 1:471-475.
[8] D, Graupe, H. Kordylewski, (1998), "A large memory storage and retrieval neural network for adaptive retrieval and diagnosis", International Journal of Software Engineering and Knowledge Engineering, 1998.
[9]H. Kordylewski, D Graupe, K. Liu, "A novel large-memory neural network as an aid in medical diagnosis applications", IEEE Transactions on Information Technology in Biomedicine, 5(3):202-209.
[10]Bengio, Yoshua (2009). "Learning Deep Architectures for AI" (PDF). Foundations and Trends in Machine Learning 2 (1): 1–127.