A SURVEY ON DEEP LEARNING TECHNIQUES, APPLICATIONS AND CHALLENGES

V. Pream Sudha[1], R. Kowsalya[2]

[1, 2]Department of Computer Science, PSGR Krishnammal College for Women, India

ABSTRACT

Deep learning is an emerging research area in machine learning and pattern recognition field. Deep learning refers to machine learning techniques that use supervised or unsupervised strategies to automatically learn hierarchical representations in deep architectures for classification. The objective is to discover more abstract features in the higher levels of the representation, by using neural networks which easily separates the various explanatory factors in the data. In the recent years it has attracted much attention due to its state-of-the-art performance in diverse areas like object perception, speech recognition, computer vision, collaborative filtering and natural language processing. As the data keeps getting bigger, deep learning is coming to play a key role in providing big data predictive analytics solutions. This paper presents a brief overview of deep learning, techniques, current research efforts and the challenges involved in it.

Keywords : Auto-Encoders, CNN, Deep learning , RBM

I. INTRODUCTION TO DEEP LEARNING

Deep learning comes from the concept of human brain having multiple types of representation with simpler features at the lower levels and high-level abstractions built on top of that. Humans arrange their ideas and concepts hierarchically. Humans first learn simple concepts and then compose them to represent more abstract ones. The human brain is like deep neural network, consisting of many layers of neurons which act as feature detectors, detecting more abstract features as the levels go up. This way of representing information in a more abstract way is easier to generalize for the machines.

The main advantage of deep learning is its compact representation of a larger set of functions than shallow networks used by most conventional learning methods. A deep architecture is more expressive than a shallow one provided the same number of non-linear units. But functions compactly represented in k layers may require exponential size when expressed in 2 layers. Formally, it can be proved that a k-layer network can represent functions compactly but a $(k-1)$-layer network cannot represent them unless it has an exponentially large number of hidden units. A lot of factors like faster CPU's, parallel CPU architectures, GPU computing enabled training of deep networks and made it computationally feasible. Neural networks are often represented as a matrix of weight vectors and GPU's are optimized for very fast matrix multiplication.

Perceptrons was invented in 1960's and when Papert and Minsky [1] proved that perceptrons can only learn to model linearly separable functions, the interest in perceptrons rapidly declined. There was revival of interest in neural networks due to the invention of back propagation for training multiple layers of non-linear features. Backpropagation takes errors from the output layer and propagates them back through the hidden layers. Many researchers gave up on backpropagation as it could not make efficient use of multiple hidden layers. In mid 2000 Geoffrey Hinton [2] trained deep belief networks layer by layer on un-labeled data using back propagation to fine tune weights on labeled data. Bengio [3] in 2006 examined deep auto-encoders as an alternative to Deep Boltzmann Machines.

II. MOTIVATION FOR DEEP LEARNING

Machine learning has successfully grown to be a major computer science discipline with extensive applications in science and engineering for many years. The computer extracts knowledge through supervised experience, where a human operator is involved in helping the machine learn by giving it hundreds or thousands of training examples, and manually correcting its mistakes. While machine learning has become leading within the field of AI, it does have its problems. It is particularly time consuming and is still not a true measure of machine intelligence as it relies on human resourcefulness to come up with the abstraction that allow computer to learn.

A primary challenge to machine learning is the lack of adequate training data to build accurate and reliable models in many realistic situations. When quality data are in short supply, the resulting models can perform very poorly on a new domain, even if the learning algorithms are best chosen. Unlabeled data is cheap and plentiful, unlike labeled data which is expensive to obtain. The promise of self-taught learning is that by exploiting the massive amount of unlabeled data, much better models can be learnt. By using unlabeled data to learn a good initial value for the weights in all the layers, the algorithm is able to learn and discover patterns from massive amounts of data than purely supervised approaches. This frequently results in much better classifiers being learned.

Deep learning is mostly unsupervised contrasting machine learning which is supervised. It involves creating large-scale neural nets that allow the computer to learn and compute by itself without the need for direct human intervention. Learning in machine learning applications depends on hand-engineering features where the researcher manually encodes relevant information about the task at hand and then there is learning on top of that. This contrasts with deep learning which tries and gets the system to engineer its own features as much as is viable.

The recent Google experiments on deep learning have shown that it is possible to train a very large unsupervised neural network to automatically develop features for recognizing cat faces. The data scarcity problem associated with extremely large-scale recommendation systems provides strong motivation for finding new ways to transfer knowledge from auxiliary data sources.

III. LITERATURE STUDY

There were attempts at training deep architectures before 2006 but failed because training a deep supervised feed forward neural network yielded worse results both in training and in test error than shallow ones with 1 or 2 hidden

layers. The scenario was changed by three important papers by Hinton, Bengio and Ranzato[2],[3],[4]. The key principles found in all three papers are on unsupervised learning of representations used to pre-train each layer. The unsupervised training in these works is done one layer at a time, on top of the previously trained ones. The representation learned at each level is the input for the next layer. Then supervised training is used to fine-tune all the layers.

Geoffrey Hinton [2] trained deep belief networks by stacking Restricted Boltzman Machines (RBMs) on top of one another as deep belief network. The Deep Belief Networks use RBMs for unsupervised learning of representation at each layer. The Bengio [3] paper explores and compares RBMs and auto-encoders. The Ranzato [4] et al paper uses sparse auto-encoder in the context of a convolutional architecture.

Recently notable progresses have been made to lessen the challenges related to high data volumes. When there is huge volume of data it is often impossible to train a deep learning algorithm with a central processor and storage. Hence distributed frameworks with parallelized machines are ideal. Deng et al. [5] proposed a modified deep architecture called Deep Stacking Network (DSN), which can be parallelized. A DSN is a combination of several specialized neural networks with a single hidden layer. Stacked modules with inputs composed of raw data vector and the outputs from previous module form a DSN.A new deep architecture called Tensor Deep Stacking Network (T-DSN), which is based on the DSN, is implemented using CPU clusters for scalable parallel computing. Recent models make use of clusters of CPUs or GPUs to increase the training speed.

Deep learning algorithms possess one of the unique characteristics of using unlabeled data during training. Training with vastly more data is preferable to using smaller number of exact, clean, and carefully curated data, though incompleteness and noisy labels are part of data. To address the the effect of noisy labels, a more efficient cost function and novel training strategy may be needed.

IV. DEEP LEARNING TECHNIQUES

Most of the current deep learning architectures consist of learning layers of RBM's or Auto-Encoders both of which are 2 layer neural networks that learn to model their inputs. RBM's model their inputs as a probability distribution whereas Auto-Encoders learn to reproduce inputs as their outputs.

RBM is a two layer undirected neural network consisting of visible layer and hidden layer. There are no connections within each layer, but connections run visible to hidden. It is trained to maximize the expected log probability of the data. The inputs are binary vectors as it learns Bernoulli distributions over each input. The activation function is computed the same way as in a regular neural network and the logistic function usually used is between 0-1. The output is treated as a probability and each neuron is activated if activation is greater than random variable. The hidden layer neurons take visible units as inputs. Visible neurons take binary input vectors as initial input and then hidden layer probabilities.

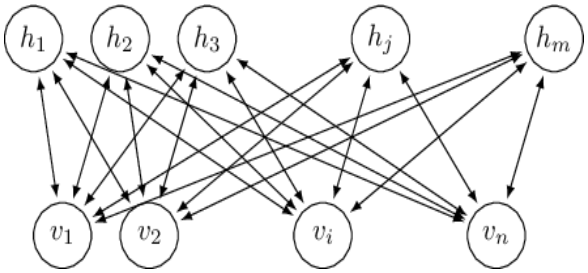

Fig 1. Visible and hidden layers in an RBM

In the training phase Gibbs Sampling (MCMC technique) is performed and is equated to computing a probability distribution using a Markov Chain Monte Carlo approach. In PASS 1 hidden layer probabilities h is computed from inputs v. In PASS 2 those values back down to the visible layer, and back up to the hidden layer to get v' and h'. The weights are updated using the differences in the outer products of the hidden and visible activations between the first and second passes. To approach the optimal model, a vast number of passes are needed, so this approach provides proximate inference, but works well in practice. After training, the hidden layer activations of an RBM can be used as learned features.

An autoencoder is classically a feedforward neural network which aims to learn a compressed, distributed representation of a dataset. An auto-encoder is a 3 layer neural network, which is trained to reconstruct its inputs by using them as the output. It needs to learn features that capture the variance in the data so it can be reproduced. It can be shown to be equivalent to PCA, if linear activation functions are only used and can be used for dimensionality reduction. Once trained, the hidden layer activations are used as the learned features, and the top layer can be discarded.

Auto encoders are trained using the strategies like de-noising, contraction and sparseness. During de-noising in Auto-Encoders some random noise is added to the input. The encoder is required to reproduce the original input. Randomly deactivating inputs during training will improve the generalization performance of regular neural networks. In contractive Auto-Encoders, setting the number of nodes in the hidden layer to be much lower than the number of input nodes forces the network to perform dimensionality reduction. This prevents it from learning the identity function as the hidden layer has insufficient nodes to simply store the input. Sparse Auto-Encoders are trained by applying a sparsity penalty to the weight update function. It penalizes the total size of the connection weights and causes most weights to have small values.

RBM's or Auto-Encoders can be trained layer by layer. The features learned from one layer are fed into the next layer, so that first a network with 1 hidden layer is trained, and only after that is done, a network with 2 hidden layers is trained, and so on. At each step, the old network with k- 1 hidden layers is taken and an additional *k*-th hidden layer is added that takes as input the previous hidden layer $k - 1$ that was trained.

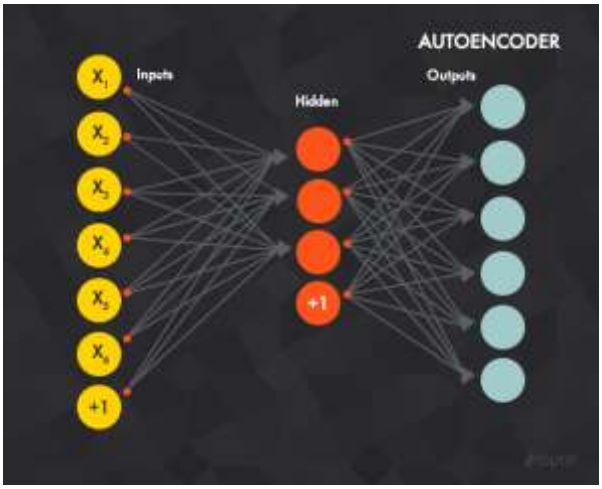

Fig 2. Autoencoders

Training can either be supervised, but more frequently it is unsupervised. The top-layer activations can be treated as features and fed into any suitable classifier like Random Forest, SVM, etc. The weights from training the layers individually are then used to initialize the weights in the final deep network, and then the entire architecture is fine-tuned. On the other hand, an additional output layer can be placed on top, and the network fine-tuned with back propagation. Back propagation works well in deep networks only if the weights are initialized close to a good solution. The layer wise pre-training ensures this. Many other approaches like dropout, maxout exist for fine tuning deep networks.

Convolutional Neural Networks (CNN) are biologically-inspired variants of MLPs. A typical Convolutional Neural Network consists of many layers of hierarchy with some layers for feature representations and others as a type of conventional neural networks for classification. There are two altering types of layers called convolutional and subsampling layers. The convolutional layers perform convolution operations with several filter maps of equal size, while subsampling layers reduce the sizes of proceeding layers by averaging pixels within a small neighborhood.

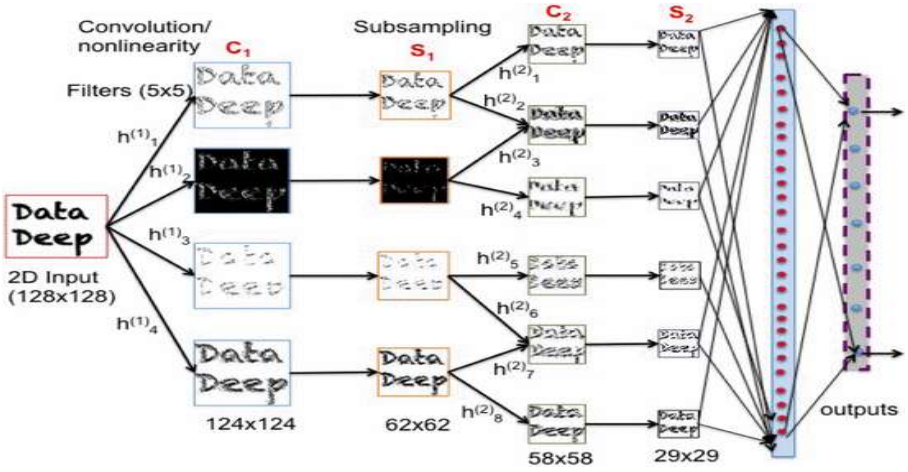

Fig 3. A typical architecture of CNN

The input is first convoluted with a set of filters. These 2D filtered data are called feature maps. After a nonlinear transformation, a subsampling is further performed to reduce the dimensionality .The sequence of convolution or subsampling can be repeated many times. The lowest level of this architecture is the input layer. With local receptive fields, upper layer neurons extract some elementary and complex features. Each convolutional layer is composed of multiple feature maps, which are constructed by convolving inputs with different filters. In other words, the value of each unit in a feature map is the result depending on a local receptive field in the previous layer and the filter.

CNN algorithms learn a hierarchical feature representation by utilizing strategies like local receptive fields, shared weights, and subsampling. Each filter bank can be trained with either supervised or unsupervised methods.

V. APPLICATIONS OF DEEP LEARNING

Deep learning is typically applied to computer vision, speech recognition, and NLP. These are non-linear classification problems where the inputs are highly hierarchal in nature. In 2011, Google Brain project, created a neural network trained with deep learning algorithms, which recognized high level concepts, like cats, after watching just YouTube videos and without being told what a "cat" is. Facebook is creating solutions using deep learning expertise to better identify faces and objects in the photos and videos uploaded to Facebook each day. Another example of deep learning in action is voice recognition like Google Now and Apple's Siri. According to Google, the voice error rate in the new version of Android stands at 25% lower than previous versions of the software after adding insights from deep learning.

Another emerging area of application is natural language processing because the possibility of understanding the meaning of the text that people type or say is very important for providing better user interfaces, advertisements, and posts. Learning from text, audio, and video is evolving into a new frontier of deep learning, beginning to be accepted by research communities including speech processing, natural language processing, computer vision, machine learning, information retrieval, cognitive science, artificial intelligence and knowledge management.

The extraordinary growth of data in recent years, has led to a rush in interest in effective and scalable parallel algorithms for training deep models. The use of great computing power to speed up the training process has shown significant potential in Big Data deep learning. Multiple CPU cores, can be used to scale up DBNs with each core dealing with a subset of training data. These implementations can supplement the performance of modern CPUs more for deep learning.

VI. CHALLENGES

Many problems using deep networks require enough examples to fit the parameters of a complex model, which is difficult. Training on insufficient data would also result in overfitting, due to the high degree of expressive power of deep networks. Training a shallow network with 1 hidden layer using supervised learning usually resulted in the parameters converging to reasonable values, but when training a deep network, this turns out with bad local optima. When using back propagation to compute the derivatives, the gradients that are propagated backwards from the

output layer to the earlier layers of the network rapidly diminish in magnitude as the depth of the network increases. The weights of the earlier layers change slowly when using gradient descent and the earlier layers fail to learn much leading to "diffusion of gradients." Another challenge associated is data incompleteness and noisy labels, as majority of data may not be labeled, or if labeled, there exist noisy labels. Advanced deep learning methods are required to deal with noisy data and to tolerate some messiness.

VII. CONCLUSION

High performance computing infrastructure-based systems together with theoretically sound parallel learning algorithms and novel architectures are needed to build the future deep learning system. As there is continuous growth in computer memory and computational power through parallel or distributed computing environment, further research and effort on addressing issues associated with computation and communication management are needed for scaling-up to very large data sets. There will be challenges involved in reasoning and inference over complex, hierarchical relationships and knowledge sources comprising numerous entities and semantic concepts. In the coming years solutions to address the scalability, reliability and adaptability of the unsupervised learning models will take central stage. These research challenges posed are timely, and will also bring ample opportunities for deep learning, providing major advances in science, medicine, and business.

REFERENCES

[1] Marvin minsky, Seymour Papert, “Perceptrons : An introduction to computational geometry”, 1969

[2] Hinton, G. E., Osindero, S. and Teh, Y., “ A fast learning algorithm for deep belief nets”, Neural Computation 18:1527-1554, 2006

[3] Yoshua Bengio, Pascal Lamblin, Dan Popovici and Hugo Larochelle, “Greedy Layer-Wise Training of Deep Networks”, in J. Platt et al. Advances in Neural Information Processing Systems 19 (NIPS 2006), pp. 153-160, MIT Press, 2007

[4] Marc’Aurelio Ranzato, Christopher Poultney, Sumit Chopra and Yann LeCun “Efficient Learning of Sparse Representations with an Energy-Based Model”, in J. Platt et al. (Eds), Advances in Neural Information Processing Systems (NIPS 2006), MIT Press, 2007

[5] Li Deng, Xiaodong He, Jianfeng Gao, Deep stacking networks for information retrieval , Acoustics, speech and signal processing, 2013 IEEE International conference