# Review on Deep Learning for Big Data: Challenges and Perspectives

Posted by Mohamad Ivan Fanany

This writing summarizes and reviews a paper on deep learning for big data: Big Data Deep Learning: Challenges and Perspectives

**Motivations**:

- Deep learning and Big Data are two hottest trends in the rapidly growing digital world.
- Big Data: exponential growth and wide availability of digital data
- The Internet is processing 1,826 Petabytes of data per day [1]
- Digital information has grown nine times in volume in just five years [2].
- By 2020, digital information in the world will reach 35 trillion gigabytes [3].
- Machine learning techniques together with advances in available computational power, play a vital role in Big Data analytics and knowledge discovery [5, 6, 7, 8]
- Deep learning together with Big Data: "big deals and the bases for an American innovation and economic revolution" [9].
- Deep learning is coming to play a key role in providing big data predictive analytics solutions.
- Big Data deep learning that uses great computing power to speed up the training process has shown significant potential in Big Data.

**Addressed problems**:

- Provide a brief overview of deep learning, and highlight current research efforts and the challenges to big data, as well as the future trends.
- Not a comprehensive survey of all the related work in deep learning,

**Deep learning**:

- Definition: machine learning techniques that use supervised and/or unsupervised strategies to automatically learn multiple level hierarchical representations in deep architectures for classification [10], [11].
- Inspired by biological observations on human brain mechanisms for processing of natural signals.
- State-of-the-arts performance:
  - speech recognition [12], [13],
  - collaborative fultering [14],
  - computer vision [15], [16].
  - Apple's Siri [17]
  - Google's deep learning [18
  - Microsoft Bing's voice search [19]
  - IBM brain-like computer [18, 20
- Two well-established deep architectures:
  - Deep belief networks (DBNs) [21, 22, 23]
  - Convolutional neural networks (CNNs) [24, 25, 26].

**A. Deep Belief Networks**:

- Conventional neural networks:
  - Prone to get trapped in local optima of a non-convex objective function [27].
  - Cannot take advantage of unlabeled data, which are often abundant and cheap to collect in Big Data.
- Deep belief network (DBN) uses a deep architecture that is capable of learning feature representations from both the labeled and unlabeled data presented to it [21].
- DBN incorporates both unsupervised pre-training and supervised fine-tuning strategies to construct the models.
- DBN architecture is a stack of Restricted Boltzmann Machines (RBMs) and one or more additional layers for discrimination tasks.
- RBMs are probabilistic generative models that learn a joint probability distribution of observed (training) data without using data labels [28]

- DBN can effectively utilize large amounts of unlabeled data for exploiting complex data structures.
- Once the structure of a DBN is determined, the goal for training is to learn the weights (and biases) between layers.
- Training is conducted firstly by an unsupervised learning of RBMs.
- RBM consists of two layers: nodes in one layer are fully connected to nodes in the other layer and there is no connection for nodes in the same layer
- Train the generative weights of each RBMs using Gibbs sampling [29, 30].
- Before fine-tuning, a layer-by-layer pre-training of RBMs is performed: the outputs of an RBM are fed as inputs to the next RBM and the process repeats until all the RBMs are pretrained.
- Layer-by-layer unsupervised learning helps avoid local optima and alleviates the over-fitting problem that is observed when millions of parameters are used.
- The algorithm time complexity, is linear to the number and size of RBMs [21].
- Features at different layers contain different information about data structures with higher-level features constructed from lower-level features.
- Number of stacked RBMs is a parameter predetermined by users and pre-training requires only unlabeled data (for good generalization).
- Weights are updated based on an approximate method called contrastive divergence (CD) approximation [31].
- While the expectations may be calculated by running Gibbs sampling infinitely many times, in practice, one-step CD is often used because it performs well [31]. Other model parameters (e.g., the biases) can be updated similarly.
- After pre-training, DBN adds a final layer representing the desired outputs.
- The overall network is fine tuned using labeled data and back propagation strategies for better discrimination.
- The final layer is called associative memory.
- Instead of using RBMs, other variations of pre-training:
  - Stacked denoising auto-encoders [32], [33],
  - Stacked predictive sparse coding [34].

- Recent results show that when a large number of training data is available, a fully supervised training using random initial weights instead of the pre-trained weights (i.e., without using RBMs or auto-encoders) will practically work well [13], [35].

**B. Convolutional neural networks**:

- CNN is composed of many layers of hierarchy with some layers [24] for
  - Feature representations (or feature maps) layers
  - Classification layers

- The feature representations layers starts with two altering types of layers called convolutional and subsampling layers.
- The convolutional layers perform convolution operations with several filter maps of equal size.
- The subsampling layers reduce the sizes of proceeding layers by averaging pixels within a small neighborhood (or by max-pooling [36], [37]).
- 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.
- The value of each unit is followed by a nonlinear activation funtion. Most recent: a scaled hyperbolic tangent function [38].
- The key parameters to be learned: weights between layers.
- Standard training: backpropagation using a gradient descent algorithm
- Loss function:mean squared-error
- Training deep CNN architectures can be unsupervised.
- Unsupervised training of CNNs: Predictive sparse decomposition (PSD) [39] that approximate inputs with a linear combination of some basic and sparse functions.
- Inspired by biological processes [40], CNN learns a hierarchical feature representation by utilizing strategies like:
- Local receptive fields: the size of each filter is normally small,
- Shared weights: using the same weights to construct all the feature maps at the same level to significantly reduces the number of parameters,
- Subsampling to further reduce the dimensionality.
- A CNN is capable of learning good feature hierarchies automatically and providing some degree of translational and distortional invariances.

**Deep learning for massive amount of data**:

- A surge in interest in effective and scalable parallel algorithms for training deep models for big data [ 12, 13, 15, 41, 42, 43, 44].
- Large-scale deep learning: large volumes of data and large models.
- Large-scale learning algorithms:
  - Locally connected networks [24], [39],
  - Improved optimizers [42],

- New structures (Deep Stacking Network or DSN) that can be implemented in parallel [44].

- A DSN consists of several specialized neural networks (called modules) with a single hidden layer.
- A new deep architecture called Tensor Deep Stacking Network (T-DSN) is based on the DSN, is implemented using CPU clusters for scalable parallel computing [45].
- One way to scale up DBNs is to use multiple CPU cores, with each core dealing with a subset of training data (data-parallel schemes).
- Some aspects of technical details of paralellization [46]:
  - Carefully designing data layout,
  - Batching of the computation,
  - Using SSE2 instructions,
  - Leveraging SSE3 and SSE4 instructions for fixed-point implementation. (These implementations can enhance the performance of modern CPUs more for deep learning.)

- Parallelize Gibbs sampling of hidden and visible units by splitting hidden units and visible units into n machines, each responsible for 1/n of the units [47].
- Data transfer between machines is required (i.e., when sampling the hidden units, each machine will have the data for all the visible units and vice verse).
- If both the hidden and visible units are binary and also if the sample size is modest, the commnication cost is small.
- If large-scale data sets are used the communication cost can rise up quickly.
- FPGA-based implementation of large-scale deep learning [48]:
  - A control unit implemented in a CPU,
  - A grid of multiple full-custom processing tiles
  - A fast memory.

- As of August 2013, NVIDIA single precision GPUs exceeded 4.5 TeraFLOP/s with a memory bandwidth of near 300 GB/s [49].
- A typical CUDA-capable GPU consists of several multi-processors.
- Each multi-processor (MP) consists of several streaming multiprocessors (SMs) to form a building block
- Each SM has multiple stream processors (SPs) that share control logic and low-latency memory.
- Each GPU has a global memory with very high bandwidth and high latency when accessed by the CPU (host).
- GPU two levels of parallelism:
  - Instruction (memory) level (i.e., MPs) and
  - Thread level (SPs).

- This SIMT (Single Instruction, Multiple Threads) architecture allows for thousands or tens of thousands of threads to be run concurrently, which is best suited for operations with large number of arithmetic operations and small access times to memory.
- The parallelism can also be effectively utilized with special attention on the data flow when developing GPU parallel computing applications.
- Reduce the data transfer between RAM and the GPU's global memory [50] by transferring data with large chunks.
- Upload as large sets of unlabeled data as possible and by storing free parameters as well as intermediate computations, all in global memory.
- Data parallelism and learning updates can be implemented by leveraging the two levels of parallelism:
  - Input examples can be assigned across MPs,
  - Individual nodes can be treated in each thread (i.e., SPs).

**Large-scale DBN**:

- Raina et al. [41] proposed a GPU-based framework for massively parallelizing unsupervised learning models including DBNs (stacked RBMs) and sparse coding [21].
- Number of free parameters to be trained:
- Hinton & Salakhutdinov [21]: 3.8 million parameters for free images
- Ranzato and Szummer [51]: three million parameters for text processing
- Raina et al. [41]: More than 100 million free parameters with millions of unlabeled training data.
- Transferring data between host and GPU global memory is time consuming.
- Minimize host-device transfers and take advantage of shared memory.
- Store all parameters and a large chunk of training examples in global memory during training parameter to allow updates to be carried out fully inside GPUs [41].
- Utilize MP/SP levels of parallelism.
- Each time, select a few of the unlabeled training data in global memory to compute the updates concurrently across blocks (data parallelism).
- Meanwhile, each component of the input example is handled by SPs.
- When implementing the DBN learning, Gibbs sampling [52], [53] is repeated.
- Gibbs sampling can be implemented in parallel for the GPU, where each block takes an

example and each thread works on an element of the example.
- Weight update operations can be performed in parallel using linear algebra packages for the GPU after new examples are generated.
- 45 million parameters in a RBM and one million examples, the GPU-based implementation increases the speed of DBN learning by a factor of up to 70, compared to a dual-core CPU implementation (around 29 minutes for GPU-based implementation versus more than one day for CPU-based implementation) [41].

**Large-scale CNN**:

- CNN is a type of locally connected deep learning methods.
- Large-scale CNN learning is often implemented on GPUs with several hundred parallel processing cores.
- CNN training involves both forward and backward propagation.
- For parallelizing forward propagation, one or more blocks are assigned for each feature map depending on the size of maps [36].
- Each thread in a block is devoted to a single neuron in a map.
- Computation of each neuron includes:
  - Convolution of shared weights (kernels) with neurons from the previous layers,
  - Activation
  - Summation in an SP.
  - Store the outputs in the global memory.

- Weights are updated by back-propagation of errors.
- Parallelizing backward propagation can be implemented either by pulling or pushing [36].
- Pulling error signals: the process of computing delta signals for each neuron in the previous layer by pulling the error signals from the current layer.
- Beware of border effects problem in pulling caused by subsampling and convolution operations: the neurons in the previous layer may connect to different numbers of neurons in the previous layer [54].
- For implementing data parallelism, one needs to consider the size of global memory and feature map size.
- At any given stage, a limited number of training examples can be processed in parallel.
- Within each block, where comvolution operation is performed, only a portion of a feature map can be maintained at any given time due to the extremely limited amount of shared memory.
- For convolution operations, use limited shared memory as a circular buffer [37], which only holds a small portion of each feature map loaded from global memory each time.
- Convolution will be performed by threads in parallel and results are written back to global memory.
- To further overcome the GPU memory limitation, a modified architecture with both the convolution and subsampling operations being combined into one step [37].
- To further speedup, use two GPUs for training CNNs with five convolutional layers and three fully connected classification layers [55].
- The CNN that uses Rectified Linear Units (ReLUs) as the nonlinear function (f (x) = max(0, x)), has been shown to run several times faster than other commonly used functions [55].
- For some layers, about half of the network is computed in a single GPU and the other portion is calculated in the other GPU; the two GPUs communicated at some other layers without using host memory.

**Combination of Data and Model Paralellism**:

- DistBelief [56]: distributed training and learning in deep networks with very large models (e.g., a few billion parameters) and large-scale data sets.
- DistBelief: large-scale clusters of machines to manage both data and model parallelism via multithreading, message passing, synchronization as well as communication between machines.
- The model is partitioned into 169 machines, each with 16 CPU cores.
- To deal with large-scale data with high dimensionality, deep learning often involves many densely connected layers with a large number of free parameters (i.e., large models).
- **Model parallelism**:
  - Allowing users to partition large network architectures into several smaller structures (called blocks), whose nodes will be assigned to and calculated in several machines.
  - Each block will be assigned to one machine.
  - Boundary nodes (nodes whose edges belong to more than one partitions) require data transfer between machines.
  - Fully-connected networks have more boundary nodes and often demand higher communication costs than locally-connected structures.
  - As many as 144 partitions, which have been reported for large models in DistBelief, leads to significant improvement of training speed.

- **Data parallelism**:
  - Employs two separate distributed optimization procedures:
    - Downpour stochastic gradient descent (SGD)
    - Sandblaster [56],

- In practice, the Adagrad adaptive learning rate procedure [57] is integrated into the Downpour SGD for better performance.
- DistBelief is implemented in two deep learning models:
- Fully connected network with 42 million model parameters and 1.1 billion examples,
- Locally-connected convolutional neural network with 16 million images of 100 by 100 pixels and 21,000 categories (as many as 1.7 billion parameters).

- Experimental results: locally connected learning models will benefit more from DistBelief. With 81 machines and 1.7 billion parameters, the method is 12x faster than using a single machine.
- Scale up from single machine to thousands of machines is the key to Big Data analysis.
- Train a deep architecture with a sparse deep autoencoder, local receptive fields, pooling, and local contrast normalization [50].
- Scale up the dataset, the model, and the resources all together.
- Multiple cores allow for another level of parallelism where each subset of cores can perform different tasks.
- Asynchronous SGD is implemented with several replicas of the core model and mini-batch of training examples.
- The framework was able to train as many as 14 million images with a size of 200 by 200 pixels and more than 20 thousand categories for three days over a cluster of 1,000 machines with 16,000 cores.
- The model is capable of learning high-level features to detect objects without using labeled data.

**The COTS HPC Systems**:

- DistBelief can learn with very large models (more than one billion parameters), its training requires 16,000 CPU cores, which are not commonly available for most researchers.
- Most recently, Coates et al. presented an alternative approach that trains comparable deep network models with more than 11 billion free parameters by using just three machines [58].
- The Commodity Off-The-Shelf High Performance Computing (COTS HPC) system is comprised of a cluster of 16 GPU servers with Infiniband adapter for interconnects and MPI for data exchange in a cluster.
- Each server is equipped with four NVIDIA GTX680 GPUs, each having 4GB of memory. With well-balanced number of GPUs and CPUs, COTS HPC is capable of running very large-scale deep learning.
- Coates et al. [58] fully take advantage of matrix sparseness and local receptive field by extracting nonzero columns for neurons that share identical receptive fields, which are then multiplied by the corresponding rows.
- This strategy successfully avoids the situation where the requested memory is larger than the shared memory of the GPU.
- Matrix operations are performed by using a highly optimized tool called MAGMA BLAS matrix-matrix multiply kernels [59].
- GPUs are further being utilized to implement a model parallel scheme:
- Each GPU is only used for a different part of the model optimization with the same input examples;
- Their communication occurs through the MVAPICH2 MPI. This very large scale deep learning system is capable of training with more than 11 billion parameters, which is the largest model reported by far, with much less machines.
- It has been observed in several groups (see [41]) that single CPU is impractical for deep learning with a large model. With multiple machines, the running time may not be a big concern any more (see [56]).
- Major research efforts are toward experiments with GPUs with their running times:
  - DBN [41]: NVIDIA GTX 280 GPU, 1 GB Mem; 1 million images, 100 million parameters, ~1 day
  - CNN [55]: 2 NVIDIA GTX 580, each 3GB Mem; 1.2 million high resolution (256x256) images, 60 million parameters; ~5-6 days
  - Distbelief [56]: 1000 CPUs, downpour SGD, Adagrad; 1 billion audio samples, 42 mi llion model parameters; ~16 hours
  - Sparse autoencoder [50]: 1000 CPUs, 16,000 cores; 100 million 200x200 images, 1 billion parameters; ~3 days
  - COTS HPC [58]: 64 NVIDIA GTX 680 GPUs, each with 4GB Mem; 100 million 200x200 images, 11 billion parameters, ~3 days

**Remaining Challenges and Perspectives**:

- Traditional machine learning: data completely loaded into memory.
- Many significant challenges posted by Big Data [63]:
  - volume: large scale of data
  - variety: different types of data
  - velocity: speed of streaming data

- Future deep learning system should be scalable to Big Data,
- Develop high performance computing infrastructure-based systems together with theoretically sound parallel learning algorithm or novel architecture.
- Big Data is often incomplete resulting from their disparate origins.

- Majority of data may not be labeled, or if labeled, there exist noisy labels.
- Deep learning is effective in integrating data from different sources. For example, Ngiam et al. [69] developed a novel deep learning algorithms to learn representations by integrating audio and video data.
- Deep learning is generally effective in
  - learning single modality representations through multiple modalities with unlabeled data
  - learning shared representations capable of capturing correlations across multiple modalities.

- Multimodal Deep Boltzmann Machine (DBM) that fuses real-valued dense image data and text data with sparse word frequencies [70]
- Different sources may offer conflicting information.
- Current deep learning: mainly tested upon bi-modalities (i.e., data from two sources).
  - Will the system performance benefits from significantly enlarged modalities?
  - At what levels in deep learning architectures are appropriate for feature fusion with heterogeneous data?

- Data are generating at extremely high speed and need to be processed in a timely manner. One solution: online learning approaches.
- Online learning learns one instance at a time and the true label of each instance will soon be available, which can be used for refining the model [71] [72] [73] [74] [75] [76].
- Conventional neural networks have been explored for online learning but only limited progress.
- Instead of proceeding sequentially one example at a time, the updates can be performed on a minibatch basis [37].
- Data are often non-stationary, i.e., data distribution is changing over time.
- Deep online learning – online learning often scales naturally and is memory bounded, readily parallelizable, and theoretically guaranteed [98].
- Deep learning for high variety and velocity of Big Data: transfer learning or domain adaption, where training and test data may be sampled from different distributions [99] [100] [101] [102] [103] [104] [105] [106] [107].
- Recent domain adoption deep learning:
- Unsupervisedly trains on a large number of unlabeled data from a set of domains, then applied it to train a classifier with few labeled examples from only one domain [100].
- Applied deep learning of multiple level representations for transfer learning where training examples may not well represent test data [99]. more abstract features discovered by deep learning approaches are most likely generic between training and test data.

**My Review**:

- This paper is very interesting state-of-the-art review on Large-scale Deep Learning for Big Data application, the challenges and future trends.
- The state-of-the-art solutions seems to be DistBelief and COTS HPC.
- No mention on Cloud based solutions so that average researchers can harness the high performance computation on the cloud without need to build their own cluster that will soon become out dated.
- Before, reading this review, I hoped the author to give some advices to average researchers that can not afford high end, industrial-level computing power, on what direction is still open to explore without such machine (working on theoretical aspects would off course one of the easy answer (smile)).

*Mar 29, 2015 2:24 am* *0 Notes*