# A Literature Review on Scalable Machine Learning with Focus on Deep Learning

Seminar Paper

UNIVERSITEIT VAN AMSTERDAM

VU VRIJE UNIVERSITEIT AMSTERDAM

**Author**: Saba Amiri (Student ID: 11980079)
**Supervisor**: Prof. Dr. Adam Belloum

Department of Computer Science
University of Amsterdam

February 14, 2019

## Abstract

Deep Learning is becoming an important tool in modern computing applications. Based on the idea of deep neural networks and inspired by the interconnected model of mammal brain, it ~~able at~~ observing and inferring from very large bodies of data. It has been successfully implemented for a wide range of different disciplines, from classification to medical applications to playing games to autonomous cars. Due to the nature of these systems and the fact that a considerable portion of their use-cases deal with large volumes of data, training them is a ~~very~~ time and resource consuming task and requires vast amounts of computing cycles. To overcome this issue, it is only natural to try to scale deep learning applications to be able to run them across a variety of different infrastructure and achieve fast and manageable training speeds while maintaining a high level of accuracy. This paper addresses the issue of scaling deep learning and gives and overview of how to scale deep learning both on a hardware and software level and how to benchmark the performance of these kinds of systems from different aspects.

# Contents

# List of Figures

# 1 Background and Rationale

There is a set of tasks in machine learning, e.g., image classification and speech recognition, which can not be solved using the traditional methods of engineering features and training a model-or a stacked set of models to predict values or classes for new input samples(drawn from an unknown set of samples with the same distribution as the training data). To solve these specific tasks, the need arose for new paradigms in machine learning and researchers began looking at biologically-inspired models. There has been extensive research in the field of neuroscience on how mammals learn. Studies suggest that neocortex might consist of identical cortical circuits with a hierarchical structure(Douglas & Martin, 2004). This model helps solve the above mentioned subset of tasks by learning hierarchical features without the need for hand-crafted features. These structures enable mammals to represent partition high-dimensional information effectively and model them based on emerging patterns which leads to a scalable learning system, capable of handling complicated patterns and a theoretically infinite flow of information. This model was the inspiration behind Hinton's famous paper in 2006 which formally introduced deep neural networks (G. E. Hinton & Salakhutdinov, 2006), and has been the main inspiration behind deep learning models like Convolutional Neural Networks (LeCun, Boser, et al., 1990), Deep Belief Networks (G. E. Hinton, Osindero, & Teh, 2006) and Convolutional Deep Belief Networks (H. Lee, Grosse, Ranganath, & Ng, 2009). With the deep models' increase in popularity and the advent of scaling efforts on deep neural networks and also developments on the hardware front, the size of these networks has grown with a fast pace, with deep neural networks with over a billion parameters being constructed and trained (Dean et al., 2012; Le, 2013; Chilimbi, Suzue, Apacible, & Kalyanaraman, 2014). Deep neural networks have proven to be powerful tools in tasks such as image classification, speech coding and recognition, machine translation, human motion modeling and information retrieval. (G. E. Hinton et al., 2006; Taylor, Hinton, & Roweis, 2007; Salakhutdinov & Hinton, 2009; Seide, Li, & Yu, 2011; Bahdanau, Cho, & Bengio, 2014; G. Hinton et al., 2012; Dahl, Yu, Deng, & Acero, 2012). However, since training these kinds of architectures are computationally intensive, the need for a scalable approach is imminent. Methods like gradient descent which are used to fine-tune deep neural networks hardly lend themselves to the scalable paradigm. With the shift of training from CPUs to GPUs, we've seen a significant increase in performance of single processing-unit training of deep neural networks (Seide et al., 2011; Dahl et al., 2012; Zeiler & Fergus, 2014). But even GPUs have their ~~own~~ limitations and with the emergence of big data-both in volume and the platforms to store, access and process them- the move towards scalable methods is unavoidable. To this end, a substantial amount of research has been done on deep learning scalability and several frameworks have been designed to perform the learning task of the network in a distributed manner. Some of notable examples include DeepImage (Wu, Yan, Shan, Dang, & Sun, 2015), Chainer (Tokui, Oono, Hido, & Clayton, 2015), TensorFlow (Abadi et al., 2016), FireCaffe (Iandola, Moskewicz, Ashraf, & Keutzer, 2016) and S-Caffe (Awan, Hamidouche, Hashmi, & Panda, 2017). But scaling across clusters of computers brings its own set of problems to the table like synchronization, communication overhead, the need for a global clock, storage and data access, redundancy and fault tolerance. The fast speed of development in this area and the complications involved in setting up an accurate, scalable and functional system has lead to a very diverse field of study. This review aims to try to categorize the existing efforts in scalable deep learning in different categories, explore different perspectives and concerns in this research area and introduce the most prominent efforts to solve them.

# 2 Research Basics

In this section, the basic research workflow and plans are defined. Research questions are formulated and search and selection process if formalized.

## 2.1 Research Process and Planning

To conduct this study, we carried out the following steps:

- **Initial search and selection with focus on reviews and surveys**: In this phase, we conducted an initial search of available literature using lax filters and criteria to form the big picture regarding to the initial research subject. 183 papers including reviews and book chapters were selected and reviewed in this stage and a hierarchical structure was produced about different subjects and categories observed in the literature.

- **Defining a focus point**: Based on the results from previous step, different main areas of research were identified. Among them, “deep learning” was chosen as the main focus point due to extensive research currently being done in this area. Using the previously gathered research, special care went into making sure the subject was neither too vague and broad neither too narrow to explore.

- **Second search and selection with focus on reviews and surveys on focus point**: A second phase of research was done on this topic. Different surveys and reviews were studied in the field and an outline of research was created.

- **Forming research questions**: Based on the study of previous step, 3 research questions were formed. The research questions were formed in a way to capture main trends and areas of contribution of each research and enable us to compare them.
- **Defining framework for comparison and evaluation**: To answer research questions, a framework has been designed to enable us to categorize and compare different research areas and to review them from different angles.
- **Final search and selection to cover as many different trends as possible**: Based on the different traits and research interests identified in the previous step, a comprehensive search was made to identify and obtain notable and novel research work in each of the defined criteria. This selection was filtered out based on our inclusion and exclusion criteria to select the final candidates.
- **Data extraction**: The selected works were thoroughly reviewed and relevant data was extracted from selected research work. The results were reported to define the different paradigms and trends in this research area. Also, preliminary bibliographic research has been done on this topic.
- **Analysis and results**: Finally research questions were answered and the report was concluded with an overview over the research done and the future works.

## 2.2 Research Questions

In this section, three research questions resultant from our initial research are defined and the rationale for each is given.

### 2.2.1 RQ1: What are the primary and notable algorithms and learning methods for scalable deep learning?

Since the research trend around deep learning is well established and we have already reasoned about the need for scalable deep learning, it is important to review the latest developments in this area. This includes new methods, frameworks and paradigms(e.g., deep learning in an IoT context).

#### 2.2.2 RQ2: Are existing frameworks for deep learning sufficient for the current demand? If not, what kind of research is being done to extend and enhance them?

There are a handful of scalable deep learning-and more generally, distributed computing- frameworks available, many of which created and maintained by big tech giants like Google(TensorFlow) and Facebook(PyTorch). The relevant question would be: are these frameworks performing adequately for the current needs in this field? If not, what notable research has been done to extend them and develop them furthermore?

#### 2.2.3 RQ3: What are the most notable works in benchmarking and evaluation of available methods and frameworks?

As diverse as the research in the field of deep learning and consequently scalable deep learning is, evaluation of different frameworks and methods' performance-from many different aspects- is a challenging and complicated subject. The scalable deep learning systems are fairly complicated, with different parts like pre-training, partitioning, optimization, etc.. Scaling them on clusters of compute nodes only adds to the complications and makes evaluation of methods and frameworks and their comparison all the more challenging and essential.

## 2.3 Search and Selection Process

### 2.3.1 Data Sources

The data sources used to do our research using VU and UvA library tools which include results from publishers like IEEE, Elseview and ACM among others. Also, Google Scholar was consulted to gather edge cases not included in the library tools.

### 2.3.2 Search Query

The search query used to gather the initial set of papers was:

(Deep Learning **OR** Deep Neural Network **OR** DNN) **AND** (Scalable **OR** Scalability **OR** Distributed **OR** Cluster **OR** Large Scale )

It should be noted that during the course of research, a number of papers were added to the initial collection of papers.

### 2.3.3 Inclusion/Exclusion Criteria

In this section, the main inclusion or exclusion conditions for research results from our search queries are included.

**Inclusion Criteria**

**I1**: The study has proposed a new theory, framework, method, algorithm or toolbox with significant contribution to the paradigm of "scalable deep learning"

**I2**: The study has been done from 2006(the year Hinton's seminal paper on deep neural networks was published) onwards.

**I3**: The research results have been published in English.

**Exclusion Criteria**

**E1**: No empirical results have been offered in for of real world experiments or simulations

**E2**: Study has been focused solely on hardware design and architecture.

**E3**: The study results have been refuted or challenged by notable researchers in the field.

# 3 Scalability in Deep Learning: A Review

## 3.1 Bibliographic Research

In this section we are going to do some basic research into publications on the subject of this report and provide some insight into the available research and literature. This data is gathered from SCOPUS.

Figure 1: Trend of "scalable deep learning" publications, 2000-2018,

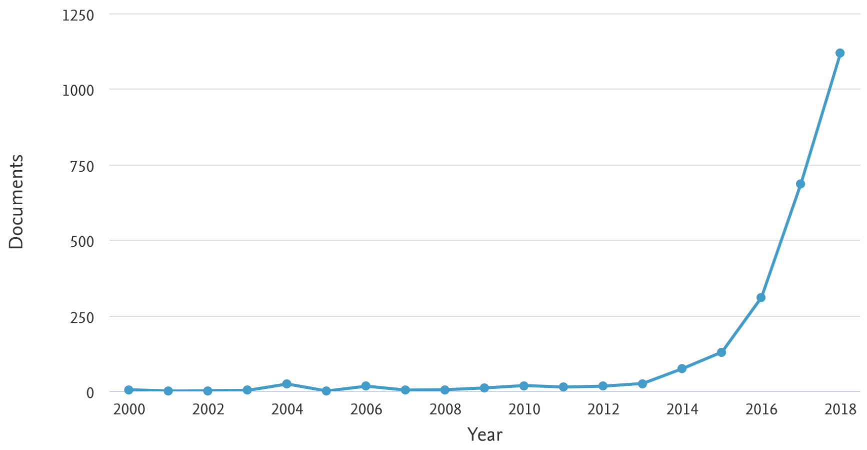


Figure 1 show the trend of publications on the subject "scalable deep learning" from 2000-2018. The search query used is:

Figure 2: Trend of "deep learning" publications, 2000-2018,

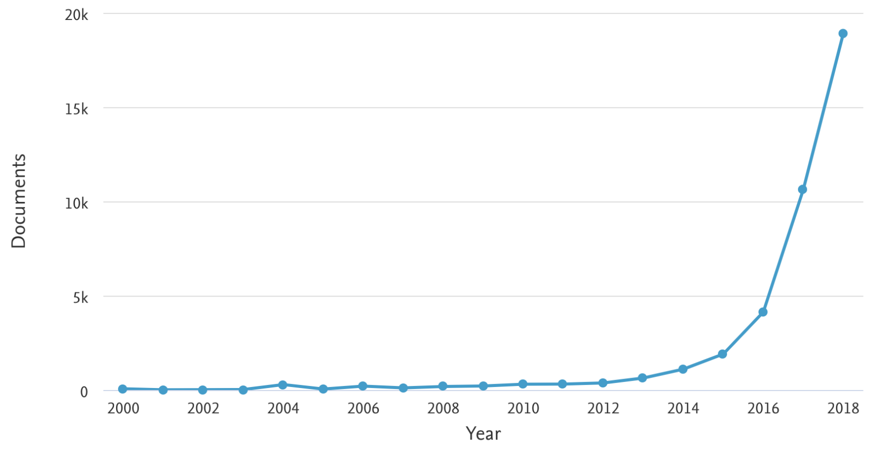
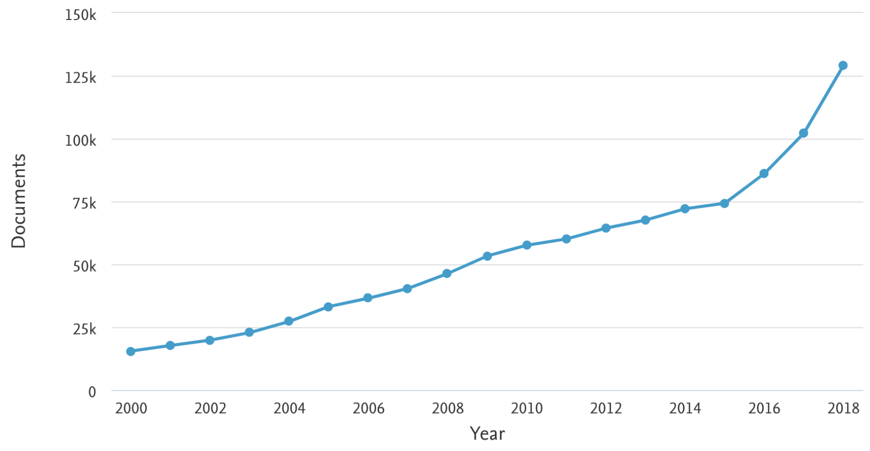


Figure 3: Trend of "neural networks" publications, 2000-2018,



(Deep Learning **OR** Deep Neural Network **OR** DNN) **AND** (Scalable **OR** Scalability **OR** Distributed **OR** Cluster **OR** Large Scale ).

As can be seen, the research in this area began to thrive in 2013 and gained full momentum in 2016. In comparison, Figure 2 shows the publications on the subject of "deep learning". The query in use for this section is

(Deep Learning **OR** Deep Neural Network **OR** DNN)

As can be observed, although the number of papers published is much higher than the "scalable" research area, both figures show almost the same trend, which leads us to conclude that the interest in "scalable deep learning" is rising with the same speed as the interest in "deep learning". In comparison, Figure 3 shows the trend of publications on "neural networks" using the query:

(Neural Networks)

As can be seen, the increase in research shows an almost linear trend up until 2016 and it can be inferred that a part of increase in interest in neural networks from 2018 onwards is in part due to research on "deep learning" gaining momentum.

Figure 4: Contribution of countries to the "deep learning" research area, 2000-2018

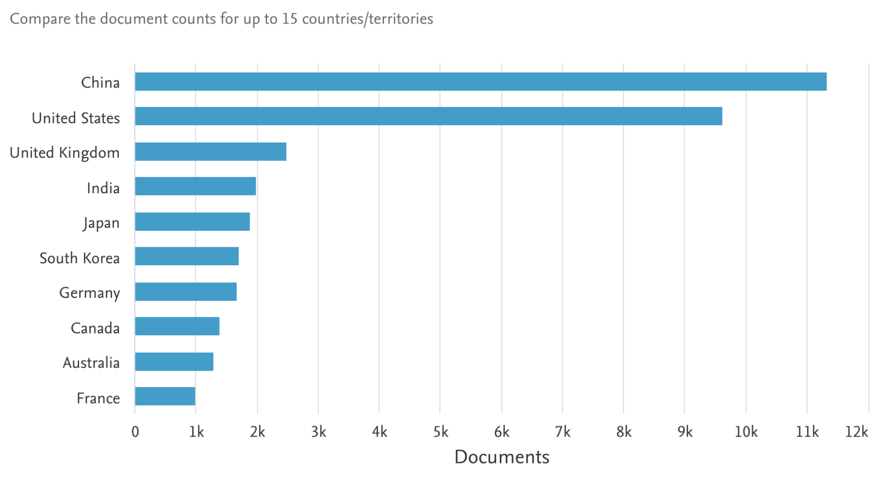


Figure 4 shows the contribution of countries to the "deep learning" research. As can be seen, China and United States are responsible for the biggest share of research in this area. It should also be noted that the most cited paper in the field of "scalable deep learning" is Dean et al.'s 2012 paper on large scale distributed learning (Dean et al., 2012).

## 3.2 Study of Existing Literature

### 3.2.1 RQ1: Methods and Frameworks

DeSTIN(Deep SpatioTemporal Inference Network) was one of the first notable scalable frameworks proposed for scalable deep learning (Arel, Rose, & Coop, 2009). Based on Bayesian inference and unsupervised learning for dynamic pattern representation, the proposed architecture provides a scalable model capable of dealing with high-dimensional input data while modeling the spatiotemporal dependencies in the data via an unsupervised method. Each node in this architecture acts like a cortical building block, modeling input patterns using clustering methods while constructing a belief state over the distribution of sequences using Bayesian inference. Nodes don't need to share information and states in any stage of the learning process, so the clustering and update process of belief rules are performed independently, making the architecture highly scalable and suitable to be run on GPUs. A big difference of this method with deep belief networks is in the fact that this method doesn't require initialization through pre-training while DBNs must be initialized by being trained in a greedy layer-wise manner.

Project Adam is another framework designed as a distributed system to train large deep neural networks on commodity machines (Chilimbi et al., 2014). The aim of this project is to optimize and balance communication and computation across the platform, achieve high performance and scalability by exploiting the underlying algorithms' ability to deal with inconsistencies, and improve the accuracy of the model. The communication and computation is optimized in this framework by partitioning the model across the cluster, hence minimizing the need for communication and synchronization. The high performance and scalability is achieved through multithreaded dirty parameter updates, asynchronous batch updates and by permitting computations over stale parameters. The machines and partitioned into data-serving machines and model training machines, with the global model updated asynchronously across multiple replicas via a global parameter server. The data serving machines are in charge of data storage and initial pre-processing and transformations(to avoid overfitting). The images are pre-cached in system memory and incoming requests are processed asynchronously. Across model training machines, the model is partitioned vertically to minimize the cross-machine communication. On each machine, training is performed in a multi-threaded fashion, with different images assigned to threads, which then perform the feed-forward and back-propagation computations. The shared model weights are updated across threads in a dirty fashion without using locks to increase performance. The race conditions and stale parameter computations are permitted, with the presumption that the model can handle these kinds of discrepancies. The weight updates are stored in a local buffer and sent to the parameter server after the processing of $k$ images. After receiving weight updates from model training machines, the parameter server directly applies them to the stored weights for convolutional layers since the number of weight parameters is low. For fully connected layers instead of weights, activation and gradient error vectors are sent to the parameter server. Parameter server then does the matrix multiplication locally and computes and applies weight updates.

In 2012, Deng et al. proposed Deep Stacking Network, which uses a parallel training algorithm on its deep architecture (Deng, Yu, & Platt, 2012). Inspired by deep belief networks and deep Boltzman machine, DSN stacks the cortical building blocks of the deep learning architecture, with each processing unit solving a convex learning problem. The learning is done in batches by adjusting weight matrices module by module and without the need for any global fitting over the entire architecture, making it overfitting-prone and suitable for being deployed over different machines and thus scalable. The main idea of this method is based on function stacking, with simple functions put in a chain and the output of each function used as the input of the next one (Wolpert, 1992).

In their 2017 paper, Wang et al. argue that popular deep learning frameworks are not suitable for performing distributed deep learning over GPUs on large clusters and propose Nexus as a platform that lets existing deep learning frameworks scale out across large clusters and perform GPU-based deep learning effectively (Y. Wang, Zhang, Ren, & Zhang, 2017). The problem with distributed GPU-based learning is the overhead cost of communication and data movement between host and device memory and also among nodes in the network. Nexus is a high performance parameter orchestration framework which can be used alongside existing platforms and enable them to scale efficiently on GPU clusters. It employs data-parallel execution model which enables the current applications developed on the aforementioned frameworks to be still usable without any modifications, while Nexus handles the parameter communication and updates on GPU clusters. On the framework level itself, Nexus exploits unique characteristics of both the frameworks and the DNNs to reduce the communication overhead. It is done by hierarchical and hybrid parameter caching and update frequency adjustment based on performance of the individual networks. Also, Nexus exploits the sparse nature of parameters and transforms the format of data structures containing parameters to reduce network load. Lastly, Nexus leverages protocols on high performance hardware to speed up the training process, including the use of RDMA protocol for transferring large tensors and performing parameter aggregation on GPUs.

Chen and Huo did propose a scalable training algorithm for deep learning by incremental block training and blockwise model updates (Chen & Huo, 2016). The incremental block training is done by partitioning data into non-overlapping chunks. After blocking the data, each block is selected and $N$ splits of this block is distributed to N workers in a single processing node which will work in parallel to optimize the local model of the processing unit. Intra-block optimization can be performed with different algorithms. A simple broadcast method is utilized here which communicates the global model to all workers to initialize local models and then each of these models are updated locally by the workers independently with 1-sweep mini-batch stochastic gradient descent. In the end, the final model would be obtained by averaging the optimal local models. After intra-block optimization is done in the parallel scheme described, a Blockwise Model-Update Filtering(BMUF) is proposed to update the global model through a block-level stochastic optimization process which will stabilize the learning process.

The Livermore Tournament Fast Batch Learning(LTFB) method proposed by Jacobs et al. is a multi-level tournament voting parallel algorithm that deals with the problem of synchronization between workers in a distributed learning scheme by creating a set of models which will be trained independently and periodically will compete with each other in local tournaments (Jacobs, Dryden, Pearce, & Van Essen, 2017). In this approach, a number of deep learning models are created and are trained independently and in parallel. Training data is partitioned across compute nodes and each node has a copy of the model. At certain intervals, each model selects another candidate model, exchange models with the candidate and then run a tournament with a hold-out dataset. The winning model will continue the training on local dataset. This method is dependant on heterogeneity and large amount of memory available per worker node and is in fact deployed on HPC machines available at Lawrence Livermore National Laboratory. Large memories on compute nodes allows for caching of hold-out dataset for utilization in local tournaments. A communication graph is maintained to manage the frequency of pairwise communication, as well as to control how far each compute node's model is allowed to digress from other nodes' models. As for the model parameters, instead of performing hyperparameter optimization on a single model, diversity is ensured in parameters of different trainers' models and and after a model wins a tournament, its metadata and set of hyperparameters are communicated to the new compute node.

Another multi-model approach is proposed by Ding et al. (Ding, Hu, Karmoshi, & Zhu, 2017). In this paper a two stage learning model is proposed for DNN training. It uses a subset of training data to train several independent local models. These local models are used for fast response in case of real time services. Then, based on these DNNs a global re-optimized global DNN model is constructed with higher accuracy compared to local models. There are two global models: The combination DNN and the optimization DNN; The combination DNN is linearly synthesized using pre-trained local models with parameters unchanged, while the optimization DNN fuses features resulting from combination DNN and learns a new set of features which will lead to a more accurate classification. This global model is then communicated to local nodes.

BAIPAS is a distributed deep learning framework proposed by Lee et al. which tackles the problem of the data transfer from the locality of access angle (M. Lee, Shin, Hong, & Song, 2017). In this framework, data is distributed and stored on compute nodes by a data locality and shuffling manager. It analyzes the training data and the local state of compute nodes. The data is then distributed based on the free space available at each compute node and its performance. To avoid overfitting, the manager also shuffles the training data and moves the data already used for training to other compute nodes. The shuffling is done parallel to training and on each worker, the new dataset-copied from another worker which was already trained on it- is used for the new round of training.

Yet another approach for data distribution in a distributed deep learning system was proposed by Wang et al. based on a distributed variant of Newton method (C.-C. Wang et al., 2018). Using variable and feature-dependant partitioning of data, the Jacobian matrix is used for matrix-vector products in the Newton method. A diagonalization method is employed so that the Newton direction can be calculated without the need of intra-worker communication. Subsampled Gauss-Newton matrices are used to reduce the runtime and communication overhead. To further reduce runtime, a termination criteria is defined which will kill the worker threads during the process of finding the Newton direction whether they have finished their run or not.

DeepSecure is another scalable deep learning framework that tackles the challenges of privacy in a setup where a IaaS company holds a pre-trained model and clients use this service to process their own private data (Rouhani, Riazi, & Koushanfar, 2017). It is based on the assumption that neither the IaaS provider of the cloud servers that host the deep learning models nor the clients who own the data are willing to reveal and share their information. This framework uses Yao's Garbled Circuit (GC) protocol to keep both the data and model parameters secure. The function that needs to be evaluated in the secure environment should be represented as a set of Boolean logic gates, also known as *netlists*. Prior to computation, the *netlist* of the publicly known deep learning architecture is obtained. During the execution the client garbles the Boolean circuit of the deep learning architecture, then sends the garbled tables to the cloud server along with the input wire labels. Next, the cloud server evaluates the garbled circuit and computes the corresponding encrypted data inference and sends the encrypted results back to the client to be decrypted using garbled keys.

SCALEDEEP is a scalable server architecture proposed by Venkataramani et al. with processing, memory and interconnect subsystems specialized for scalable training of DNNs (Venkataramani et al., 2017). SCALEDEEP's main focus is on training stage rather than inference or evaluation stages. It includes heterogeneous processing tiles and chips, hierarchical memory and a 3-tiered interconnect topology specialized for communications and memory access in DNNs, an efficient synchronization mechanism based on hardware data-flow trackers and a scheme to map DNNs to the mentioned architecture. A compiler was developed to allow the translation of any deep topology into SCALEDEEP and a simulator to estimate performance. Lim et al. in their 2017 paper have proposed a shared memory architecture for scalable deep learning across clusters (Lim, Ahn, & Choi, 2017). This shared memory structure is used for parameter storage and maintenance of deep models. The memory server is connected to the network via a high-speed interface and exposes its memory which is accessible via low-latency, high-performance RDMA and is shared among compute nodes. A virtual address space is defined for each process and it is mapped to the shared memory space of the server. A space in worker's physical memory is reserved and mapped to the virtual memory. Whenever the worker wants to access that region of its local memory, shared memory driver handles the resulting page fault and transfers the data from the corresponding page in the shared memory to the local memory of the worker node. Thus, the threads in compute node can access the shared memory and read from and write to it like their own local memory since the memory access is abstracted by the proposed shared memory scheme and is transparent to the worker thread.

### 3.2.2 RQ1: Scalability by Workload Reduction

A different approach towards scalability would be to reduce the computational load instead of trying to distribute it to a large number of machines. In case of deep networks, we will try to reduce the number of calculations by calculating only a subset of network parameters. There are notable methods for reduction of parameters like optimal brain damage (LeCun, Denker, & Solla, 1990), factored restricted Boltzmann machines (Krizhevsky, Hinton, et al., 2010), randomly connected deep convolutional neural network (Cireşan, Meier, Masci, Gambardella, & Schmidhuber, 2011), approximate convolutional filter banks (Rigamonti, Sironi, Lepetit, & Fua, 2013) and locally connected features (Coates, Ng, & Lee, 2011). Here we will mention the method proposed by Denil et al. in 2013 (Denil, Shakibi, Dinh, Ranzato, & de Freitas, 2013). Instead of distributing the computations across different processing units, their proposed method reduces the workload by computing a small subset of weight values and predicting the rest. A usual approach in training a large network is to distribute many copies of the model across a number of machines and update them independently, and use a synchronization method to align the same parameters over different machines. As can be seen in (Dean et al., 2012), this method proves ineffective in utilizing parallel resources, with 81 machines speeding up the mini-batch training time by a factor of 12, and increasing the number of machines by almost 50% slightly increasing the speed up factor to 14. In their work, Denil et al. leverage the fact that weights in learned networks are structured to estimate parameters. Here, the weight matrix is represented as low rank product of two smaller matrices. The size of the parameterization is controlled by rank of the factored weight matrix.

### 3.2.3 RQ1: Scalability in IoT, Wireless Networks and Embedded Systems

Another consideration on scalability of deep learning methods is the infrastructure on which data is gathered and the computation is performed. Most of the methods we discussed previously start from the simple assumption of a large dataset being already available-e.g., in case of ImageNet competition- and since processing of this data in a non-distributed manner is not scalable, they try to achieve scalability by distributing the workload, or in some cases by reduction of it. This assumption would not hold in some the more modern contexts, namely IoT and sensor networks. A sensor network has serious limitations on communication between sensors and the central processing facilities and it might not be feasible to gather all the data from a vast network of sensors on a fusion system and then perform machine learning tasks on them. Li et al. tackle this problem by devising a deep learning framework with different layers places on the sensors themselves (Li, Xie, Huang, Wang, & Niu, 2015). This approach helps avoid huge computational load on fusion servers and saving the wireless sensor network's transmitting power. Two challenges are addressed here: the synchronization between calculating nodes, and the trade-off between calculating power consumption and transmission power consumption. To overcome there problems, the DNN is trained in the fusion center. Data is sampled from all WSN sensors, the DNN is trained and the results of parameter calculation are sent to DNN layers located in different calculating units. Moreover, a formula is provided to calculate the upper limit of calculations where transmission of the data directly to the fusion center would become the more viable option. Another notable method that deals with IoT and embedded devices is the SC-DCNN proposed by Ren et al. (Ren et al., 2017). Stochastic Computing-Deep Convolutional Neural Network(SC-DCNN) is based on the principles of stochastic computing, which represents a number between [-1,1] by counting the number of ones in the bit-stream encoding the number. This enables us to do add and multiply operations using AND gates and Multiplexers, which has a significantly lower power consumption. This reduction in power consumption and hardware utilization leads to enhanced scalability.

### 3.2.4 RQ2: Systems Based on and Extending Existing Frameworks

Since many sophisticated and powerful frameworks for deep learning have been proposed and implemented in recent years, some of the research in this area have leveraged these existing frameworks and have built upon them their system of choice. Zhang and Chen in (Zhang & Chen, 2014) propose a distributed framework for training deep belief networks with restricted Boltzmann machines using MapReduce. A series of distributed RBMs are stacked for pre-training and a distributed back-propagation method is introduced for model fine-tuning. Each level RBM is stored in a key-values structure and pre-training step is done in a layer-wise distributed fashion on MapReduce. The back-propagation stage is also distributed across different machines, also on MapReduce. A main controller is in charge of both processes. The utilization of MapReduce framework contributes to the scalability of the proposed method and the scalability of the proposed method is only limited by the MapReduce framework itself.

Another example of methods based on existing frameworks is the work of Rao et al. (Rao, Kumar, Cadabam, & Praveena, 2017) on a distributed deep reinforcement learning system based on TensorFlow. They use the Q-learning algorithm to train their deep model. In this method, the agent constantly observes the environment-e.g., images, signals- and also gathers feedback from the environment. The deep Q-learning algorithm produces the optimal action based on the input observation and feedback, while the hyperparameters of the deep Q-network will remain unchanged in different environments. The algorithm is implemented on the distributed TensorFlow and is designed to work on GPUs. The weights of deep Q-network are stored on a parameter server and are replicated across training systems. the deep Q-learning algorithm calculates the loss value which is then used to update the weights of the deep Q-network.

ChainerMN by Akiba et al. is an extension to the Chainer framework which enables it to perform the deep network training in a distributed manner (Akiba, Fukuda, & Suzuki, 2017). They have implemented a data-parallel extension to the Chainer framework with synchronous updates. The mentioned data-parallel approach has four steps: feed-forward step, back-propagation step, Allreduce step and optimization step. In the Allreduce phase, compute nodes communicate and calculate the average of gradients calculated by other nodes and communicate this average across the network. Next, all compute nodes will update their local models using the communicated average gradient.

GossipGraD is a gossip communication based stochastic gradient descent algorithm which extends the Caffe framework with the aim of scaling up deep learning on clusters of compute nodes (Daily, Vishnu, Siegel, Warfel, & Amatya, 2018). This algorithm tries to reduce the complexity of communication to a constant $O(1)$ while maintaining asynchronous communications. Each node at each step communicates with only one other node, and after $\log(p)$ steps each node would have communicated with all other nodes either directly or indirectly. Direct diffusion of model updates is done by batch-wise rotation of partners and updates to the network parameters are asynchronously communicated to other nodes in a layer-wise/batch-wise manner. Like BAIPAS, memory shuffling is employed-in an asynchronous fashion- to avoid over-fitting. The shuffle step is done during the feed-forward phase, so there is no extra communication overhead.

### 3.2.5 RQ3: Evaluation and Benchmarking

Although many methods and frameworks have been proposed to scale up deep learning in recent years, since many of them are optimized for specific tasks-e.g., image processing, speech processing-, it is hard to compare them and also to evaluate them individually in a more holistic manner. In this section we try to focus on methods and rationale proposed in literature for evaluation of these types of scalable systems. It should be noted that we are not interested in the actual results of the evaluations, but rather the metrics and the methods employed for the evaluation. One of the recent efforts in evaluation of current scalable deep learning methods have been published by Shi et al. (Shi, Wang, & Chu, 2017). This work evaluates the performance of Caffe-MPI, CNTK, MXNet and TensorFlow frameworks in different settings. The low level parameters recorded for each experiment include:

- *Number of total GPUs*
- *Number of GPUs on each node*
- *Number of training samples per GPU in a mini-batch*
- *Time of an iteration*
- *Time of I/O in each iteration*
- *Data transfer time from CPU to GPU in each iteration*
- *Time of the forward phase in each iteration*
- *Time of the backward phase in each iteration*
- *Time of the backward phase of the layer i in each iteration*
- *Time of the model update in each iteration*
- *Time of the gradients aggregation in each iteration*
- *Gradients aggregation time of layer i in each iteration.*

They have carried out their experiments on a cluster with homogeneous hardware. The experiments were designed in single-node/single-GPU, single-node/multi-GPU and multi-node/multi-GPU settings. Th nodes are connected using a high-bandwidth, low-latency network. Three convolutional neural networks(AlexNet, GoogLeNet, and ResNet-50) have been used as driver models. To remove the overhead of disk I/O from calculations, the results from the first epoch are discarded.

DAWNBench is a benchmark and competition framework focusing on end-to-end training time to achieve high accuracy level and the inference time elapsed to reach the pre-set level of accuracy (Coleman et al., 2017). It used only the metrics mentioned above and is mostly focused on the effect of different design decisions and settings of the models like type of optimizer used, stochastic depth and the infrastructure(CPU, GPU; single, multi compute elements) on the end to end performance of the model.

Pumma et al. in their 2017 paper focus on the I/O subsystem of Caffe (Pumma, Si, Feng, & Balaji, 2017). They perform extensive scalability tests on this platform and conclude that the I/O subsystem of Caffe is inefficient. The reason for this shortcoming is, Caffe's I/O subsystem, namely LMDB, relies on memory-mapped I/O to access its database which is proven to be ineffective on large scale systems because of the high volume of interactions in generates with process scheduling system and the parallel storage system.

Bianco et al. have done an in-depth analysis of the state of the art deep neural network systems in the field of image recognition (Bianco, Cadene, Celona, & Napoletano, 2018). The DNN systems include AlexNet, GoogLeNet, ResNet-18, -34, -50, -101, -152, DenseNet, etc. The evaluation is done on the PyTorch framework, but the focus of this research is on the models themselves, not the framework. The performance metrics evaluated include:

- Accuracy rate
- Model complexity
- Memory usage
- Computational complexity
- Inference time
- Learning power
- Accuracy rate vs Learning power
- Accuracy rate vs Inference time

- Memory usage vs Model complexity

## 4 Conclusion

In this report, we did a review of available literature and research in the area of scalable deep learning. We established the importance of deep learning and the impact of this theory on both academic and industrial efforts. We defined the need for deep learning algorithms and methods to be scalable to be able to keep up with its intense computational complexities and the considerably large size of the training data available. We defined our research strategy and based on this strategy, defined three research questions related to the notable methods and algorithms in the area of deep learning, already existing frameworks and efforts to extend them towards scalability and the methods to evaluate and benchmark scalable deep learning methods. We discussed several notable scalable deep learning methods and frameworks. We've observed that many of these methods have evolved from running on CPUs to running on clusters of CPUs to running on multi-GPU single compute nodes and finally to GPU clusters. We've also observed that the main challenges in scalable deep learning these methods aim to tackle are related to I/O, data partitioning and locality, network throughput and latency and synchronization.

Some of the literature we reviewed were directly aimed at super computers, which are not readily available to masses and are difficult and costly to set up and maintain, but are not affected by some of the problems of more common clusters, e.g., homogeneity of hardware.

We have also looked at the scalability issue from security perspective and introduced a framework which assumes a IaaS scheme in which the model is trained by the provider and the data is the property of the client and neither are willing to share the details of their intellectual property.

We briefly discussed methods which try to achieve scalability by reducing the workload instead of scaling the intensive workload through a series of compute nodes.

We argued that deep learning need not always be in an ideal setting in which data is available in one place and there is a cluster of compute nodes present. We investigated this situation in wireless sensor networks and introduced methods that are designed to perform deep learning in such an environment.

We've observed a considerable amount of research being done on existing frameworks provided by large tech companies and reviewed some notable efforts to extend and scale them.

Lastly, we discussed metrics and methods to evaluate the performance and the "scalability" of the existing methods and frameworks.

For future works, one can also add a hardware perspective to this research which could be argued that is not explicitly related to deep learning, but rather a field of research and development driven directly by the computational needs of deep learning methods.

Also, the evaluation and benchmarking methods have a lot more to cover, which could be a very interesting topic of research.

## References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... Zheng, X. (2016, May). TensorFlow: A system for large-scale machine learning. *arXiv:1605.08695 [cs]*. Retrieved 2018-11-15TZ, from `http://arxiv.org/abs/1605.08695` (arXiv: 1605.08695)

Akiba, T., Fukuda, K., & Suzuki, S. (2017, October). ChainerMN: Scalable Distributed Deep Learning Framework. *arXiv:1710.11351 [cs]*. Retrieved 2019-02-09TZ, from `http://arxiv.org/abs/1710.11351` (arXiv: 1710.11351)

Arel, I., Rose, D. C., & Coop, R. (2009). Destin: A scalable deep learning architecture with application to high-dimensional robust pattern recognition. In *Aaai fall symposium: Biologically inspired cognitive architectures.*

Awan, A. A., Hamidouche, K., Hashmi, J. M., & Panda, D. K. (2017). S-Caffe: Co-designing MPI Runtimes and Caffe for Scalable Deep Learning on Modern GPU Clusters. In *Proceedings of the 22nd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming* (pp. 193–205). New York, NY, USA: ACM. Retrieved 2019-02-08TZ, from `http://doi.acm.org/10.1145/3018743.3018769` doi: 10.1145/3018743.3018769

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bianco, S., Cadene, R., Celona, L., & Napoletano, P. (2018). Benchmark Analysis of Representative Deep Neural Network Architectures. *IEEE Access*, *6*, 64270–64277. doi: 10.1109/ACCESS.2018.2877890

Chen, K., & Huo, Q. (2016, March). Scalable training of deep learning machines by incremental block training with intra-block parallel optimization and blockwise model-update filtering. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 5880–5884). doi: 10.1109/ICASSP.2016.7472805

Chilimbi, T. M., Suzue, Y., Apacible, J., & Kalyanaraman, K. (2014). Project adam: Building an efficient and scalable deep learning training system. In *Osdi* (Vol. 14, pp. 571–582).

Cireşan, D. C., Meier, U., Masci, J., Gambardella, L. M., & Schmidhuber, J. (2011). High-performance neural networks for visual object classification. *arXiv preprint arXiv:1102.0183*.

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

Coleman, C., Narayanan, D., Kang, D., Zhao, T., Zhang, J., Nardi, L., ... Zaharia, M. (2017). Dawnbench: An end-to-end deep learning benchmark and competition. *Training*, *100*(101), 102.

Dahl, G. E., Yu, D., Deng, L., & Acero, A. (2012). Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on audio, speech, and language processing*, *20*(1), 30–42.

Daily, J., Vishnu, A., Siegel, C., Warfel, T., & Amatya, V. (2018, March). GossipGraD: Scalable Deep Learning using Gossip Communication based Asynchronous Gradient Descent. *arXiv:1803.05880 [cs]*. Retrieved 2019-02-09TZ, from `http://arxiv.org/abs/1803.05880` (arXiv: 1803.05880)

Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., ... others (2012). Large scale distributed deep networks. In *Advances in neural information processing systems* (pp. 1223–1231).

Deng, L., Yu, D., & Platt, J. (2012, March). Scalable stacking and learning for building deep architectures. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 2133–2136). doi: 10.1109/ICASSP.2012.6288333

Denil, M., Shakibi, B., Dinh, L., Ranzato, M. A., & de Freitas, N. (2013). Predicting Parameters in Deep Learning. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 26* (pp. 2148–2156). Curran Associates, Inc. Retrieved 2018-12-09TZ, from `http://papers.nips.cc/paper/5025-predicting-parameters-in-deep-learning.pdf`

Ding, C., Hu, Z., Karmoshi, S., & Zhu, M. (2017, August). A Novel Two-stage Learning Pipeline for Deep Neural Networks. *Neural Processing Letters*, *46*(1), 159–169. Retrieved 2019-01-18TZ, from `https://doi.org/10.1007/s11063-017-9578-6` doi: 10.1007/s11063-017-9578-6

Douglas, R. J., & Martin, K. A. (2004). Neuronal circuits of the neocortex. *Annu. Rev. Neurosci.*, *27*, 419–451.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., ... others (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.

Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006, July). A fast learning algorithm for deep belief nets. *Neural Comput.*, *18*(7), 1527–1554. Retrieved from http://dx.doi.org/10.1162/neco.2006.18.7.1527 doi: 10.1162/neco.2006.18.7.1527

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, *313*(5786), 504–507.

Iandola, F. N., Moskewicz, M. W., Ashraf, K., & Keutzer, K. (2016). FireCaffe: Near-Linear Acceleration of Deep Neural Network Training on Compute Clusters. In (pp. 2592–2600). Retrieved 2018-11-07TZ, from https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/Iandola_FireCaffe_Near-Linear_Acceleration_CVPR_2016_paper.html

Jacobs, S. A., Dryden, N., Pearce, R., & Van Essen, B. (2017). Towards Scalable Parallel Training of Deep Neural Networks. In *Proceedings of the Machine Learning on HPC Environments - MLHPC'17* (pp. 1–9). Denver, CO, USA: ACM Press. Retrieved 2019-02-09TZ, from http://dl.acm.org/citation.cfm?doid=3146347.3146353 doi: 10.1145/3146347.3146353

Krizhevsky, A., Hinton, G., et al. (2010). Factored 3-way restricted boltzmann machines for modeling natural images. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (pp. 621–628).

Le, Q. V. (2013). Building high-level features using large scale unsupervised learning. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on* (pp. 8595–8598).

LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E., & Jackel, L. D. (1990). Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems* (pp. 396–404).

LeCun, Y., Denker, J. S., & Solla, S. A. (1990). Optimal brain damage. In *Advances in neural information processing systems* (pp. 598–605).

Lee, H., Grosse, R., Ranganath, R., & Ng, A. Y. (2009). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations.

In *Proceedings of the 26th annual international conference on machine learning* (pp. 609–616).

Lee, M., Shin, S., Hong, S., & Song, S. (2017, November). BAIPAS: Distributed Deep Learning Platform with Data Locality and Shuffling. In *2017 European Conference on Electrical Engineering and Computer Science (EECS)* (pp. 5–8). doi: 10.1109/EECS.2017.10

Li, C., Xie, X., Huang, Y., Wang, H., & Niu, C. (2015, July). Distributed Data Mining Based on Deep Neural Network for Wireless Sensor Network. *International Journal of Distributed Sensor Networks*, *11*(7), 157453. Retrieved 2019-01-18TZ, from `http://journals.sagepub.com/doi/10.1155/2015/157453` doi: 10.1155/2015/157453

Lim, E., Ahn, S., & Choi, W. (2017, October). Accelerating training of DNN in distributed machine learning system with shared memory. In *2017 International Conference on Information and Communication Technology Convergence (ICTC)* (pp. 1209–1212). doi: 10.1109/ICTC.2017.8190900

Pumma, S., Si, M., Feng, W., & Balaji, P. (2017, December). Towards Scalable Deep Learning via I/O Analysis and Optimization. In *2017 IEEE 19th International Conference on High Performance Computing and Communications; IEEE 15th International Conference on Smart City; IEEE 3rd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)* (pp. 223–230). doi: 10.1109/HPCC-SmartCity-DSS.2017.29

Rao, P. A., Kumar, B. N., Cadabam, S., & Praveena, T. (2017, September). Distributed Deep Reinforcement Learning using TensorFlow. In *2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC)* (pp. 171–174). doi: 10.1109/CTCEEC.2017.8455196

Ren, A., Li, Z., Ding, C., Qiu, Q., Wang, Y., Li, J., … Yuan, B. (2017). SC-DCNN: Highly-Scalable Deep Convolutional Neural Network Using Stochastic Computing. In *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems* (pp. 405–418). New York, NY, USA: ACM. Retrieved 2019-01-18TZ, from `http://doi.acm.org/10.1145/3037697.3037746` doi: 10.1145/3037697.3037746

Rigamonti, R., Sironi, A., Lepetit, V., & Fua, P. (2013). Learning separable filters. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 2754–2761).

Rouhani, B. D., Riazi, M. S., & Koushanfar, F. (2017, May). DeepSecure: Scalable Provably-Secure Deep Learning. *arXiv:1705.08963 [cs]*. Retrieved 2019-02-09TZ, from `http://arxiv.org/abs/1705.08963` (arXiv: 1705.08963)

Salakhutdinov, R., & Hinton, G. (2009). Semantic hashing. *International Journal of Approximate Reasoning*, *50*(7), 969–978.

Seide, F., Li, G., & Yu, D. (2011). Conversational speech transcription using context-dependent deep neural networks. In *Twelfth annual conference of the international speech communication association.*

Shi, S., Wang, Q., & Chu, X. (2017, November). Performance Modeling and Evaluation of Distributed Deep Learning Frameworks on GPUs. *arXiv:1711.05979 [cs]*. Retrieved 2019-02-09TZ, from `http://arxiv.org/abs/1711.05979` (arXiv: 1711.05979)

Taylor, G. W., Hinton, G. E., & Roweis, S. T. (2007). Modeling human motion using binary latent variables. In *Advances in neural information processing systems* (pp. 1345–1352).

Tokui, S., Oono, K., Hido, S., & Clayton, J. (2015). Chainer: a next-generation open source framework for deep learning. In *Proceedings of workshop on machine learning systems (learningsys) in the twenty-ninth annual conference on neural information processing systems (nips)* (Vol. 5, pp. 1–6).

Venkataramani, S., Dubey, P., Raghunathan, A., Ranjan, A., Banerjee, S., Das, D., . . . Kaul, B. (2017). ScaleDeep: A Scalable Compute Architecture for Learning and Evaluating Deep Networks. In *Proceedings of the 44th Annual International Symposium on Computer Architecture - ISCA '17* (pp. 13–26). Toronto, ON, Canada: ACM Press. Retrieved 2019-02-09TZ, from `http://dl.acm.org/citation.cfm?doid=3079856.3080244` doi: 10.1145/3079856.3080244

Wang, C.-C., Tan, K. L., Chen, C.-T., Lin, Y.-H., Keerthi, S. S., Mahajan, D., . . . Lin, C.-J. (2018, January). Distributed Newton Methods for Deep Neural Networks. *arXiv:1802.00130 [cs, math, stat]*. Retrieved 2019-01-18TZ, from `http://arxiv.org/abs/1802.00130` (arXiv: 1802.00130)

Wang, Y., Zhang, L., Ren, Y., & Zhang, W. (2017, September). Nexus: Bringing Efficient and Scalable Training to Deep Learning Frameworks. In *2017 IEEE 25th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)* (pp. 12–21). doi: 10.1109/MASCOTS.2017.34

Wolpert, D. H. (1992). Stacked generalization. *Neural networks*, *5*(2), 241–259.

Wu, R., Yan, S., Shan, Y., Dang, Q., & Sun, G. (2015). Deep image: Scaling up image recognition. *arXiv preprint arXiv:1501.02876*.

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision* (pp. 818–833).

Zhang, K., & Chen, X. (2014). Large-Scale Deep Belief Nets With MapReduce. *IEEE Access*, *2*, 395–403. doi: 10.1109/ACCESS.2014.2319813