



# A Snapshot of Parallelism in Distributed Deep Learning Training

Hairol Romero-Sandí<sup>1</sup>, Gabriel Núñez<sup>1</sup>, Elvis Rojas<sup>1,2</sup>

<sup>1</sup> Universidad Nacional, Pérez Zeledón, Costa Rica,

<sup>2</sup> National High Technology Center, San José, Costa Rica

[hromero@una.ac.cr](mailto:hromero@una.ac.cr), [jgabrielnm@una.ac.cr](mailto:jgabrielnm@una.ac.cr), [erojas@una.ac.cr](mailto:erojas@una.ac.cr)

(Received: 6 February 2024; accepted: 18 June 2024, Published online: 30 June 2024)

**Abstract.** The accelerated development of applications related to artificial intelligence has generated the creation of increasingly complex neural network models with enormous amounts of parameters, currently reaching up to trillions of parameters. Therefore, it makes your training almost impossible without the parallelization of training. Parallelism applied with different approaches is the mechanism that has been used to solve the problem of training on a large scale. This paper presents a glimpse of the state of the art related to parallelism in deep learning training from multiple points of view. The topics of pipeline parallelism, hybrid parallelism, mixture-of-experts and auto-parallelism are addressed in this study, which currently play a leading role in scientific research related to this area. Finally, we develop a series of experiments with data parallelism and model parallelism. The objective is that the reader can observe the performance of two types of parallelism and understand more clearly the approach of each one.

**Keywords:** deep learning, parallelism, artificial neural networks.

## 1 Introduction

Deep learning has shown great potential to solve complex problems in several areas, such as computer vision, natural language processing, and speech recognition (Hey, 2020). For this reason, the scientific community has implemented artificial intelligence approaches with great acceptance in many areas of science and engineering (Stevens, et al., 2020). However, as deep learning models have become larger and more complex, the training time required to obtain accurate results has increased significantly. To address this challenge and speed up the training process, parallelism techniques have been implemented and combined in the field of deep learning (DL) (Chen, M, 2023).

The goal of this paper is to provide an overview of the various parallelism techniques used in training DL (Ben-Nun & Hoefler, 2019; Verbraeken, et al., 2020). The current overview of the most important issues related to parallelism in deep learning is presented, including parallelism paradigms, techniques or approaches, optimizations and deep learning frameworks. Also, important concepts are described to support the theoretical basis of artificial neural networks and how they have evolved to reach the most advanced deep learning models used today.

This study focuses on the two main types of parallelism: data parallelism and model parallelism. Data parallelism executes the same task on multiple distributed nodes with a different data set and model parallelism changes the approach by partitioning the neural network model and distributing it across multiple accelerators (Rojas, Quirós-Corella, Jones, & Meneses, 2022). In addition, research related to various parallelism techniques and optimizations is described, such as pipeline parallelism that divides the training of deep learning models into stages and processes them in parallel, hybrid parallelism that takes advantage of the combination of several parallelism approaches (DP and MP) to speed up training, and mixture of experts (MoE) that uses multiple expert models to optimize training performance and learning. As an additional element, a brief description of the state of the research related to fault tolerance in parallel DL training is made.

Finally, through the execution of a series of experiments, a comparison of the training of DL models using two parallelism approaches is presented. The results obtained are analyzed and compared, in order to provide the reader with an experimental vision of the main approaches.

## 2 Important Topics in Deep Learning Parallelism

### 2.1 Artificial Neural Network and Deep Learning

Inspired by biological neural networks. Artificial Neural Network (ANN) are massively parallel computing systems consisting of an extremely large number of simple processors with many interconnections. All of these interconnections have a value, commonly called weight, that is adjusted to allow for learning. Some ANN architectures also have weighted connections not only from one layer to the next, but also to one or more layers below (Hopfield, 1988; Pouyanfar, et al., 2018). An ANN consists of an input layer of neurons (or nodes, units), one or two (or even more) hidden layers of neurons, and a final layer of output neurons and must be configured in such a way that the application of a set of inputs produces the desired set of outputs (Wang, 2003).

Deep learning algorithms are also a subset of ANNs when the use of multilayer structures (hidden layers) is preferred, since they can handle more than one problem at the same time to give a unique answer (Kukačka, Golkov, & Cremers, 2017). Deep learning uses multiple layers to represent data abstractions to build computational models. Deep learning algorithms are mainly based on the well-known Deep Neural Networks (DNN) or also called Convolutional Neural Networks (CNN) (Wu, 2017).

CNN is one of the largest networks in the field of deep learning. They are analogous to traditional ANNs in that they are composed of neurons that selfoptimize through learning. The only notable difference between CNNs and traditional ANNs is that CNNs are mainly used in the field of pattern recognition within images (Albawi, Mohammed, & Al-Zawi, 2017; Li, Liu, Yang, Peng, & Zhou, 2022).

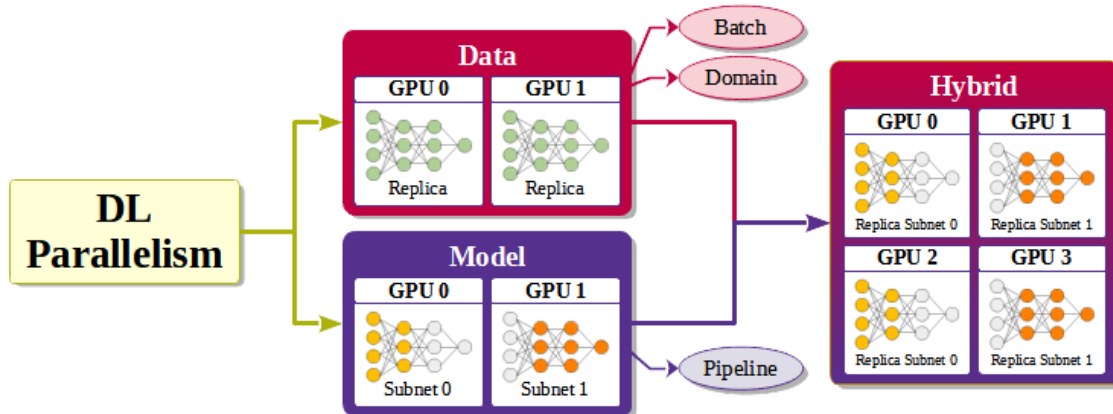


Figure 1. Deep Learning Parallelism

DL Frameworks DL uses multiple layers to represent the abstractions of data to build computational models. Different DL algorithms help to improve the learning performance, broaden the scopes of applications, and simplify the calculation process. DL frameworks (Abadi, et al., 2016; Batur Dinler, Şahin, & Abualigah, 2021; Chen, et al., 2015; Collobert, Bengio, & Mariéthoz, 2002; Jia, et al., 2014; Manaswi, 2018; Ravanelli, Parcollet, & Bengio, 2019) combine the implementation of modularized algorithms, optimization techniques, distribution techniques, and support to infrastructures (Pouyanfar, et al., 2018). In conclusion, the purpose of the DL Frameworks is to help developers and researchers to easily take advantage of the technologies (Mittal & Vaishay, 2019).

### 2.2 Parallelism Paradigms

Currently the components of high performance computing systems are dedicated to supporting parallelism. Neural network algorithms have been adapted for better use in parallelism paradigms. There are two main approaches to train deep neural network models in a distributed manner: data and model parallelism. The figure 1 shows in a general way the operation and relationship that exists between the 3 main DL paradigms that exist.

Data Parallel Data parallelism is usually expressed as a single thread of control operating on data sets distributed over all nodes (Subhlok, Stichnoth, O'Hallaron, & Gross, 1993). Data parallelism divides the data set into partitions whose number is equal to the number of accelerators (GPUs, TPUs). According to the gradient update strategy, data parallelism can be divided into two categories, synchronous parallelism and asynchronous parallelism. In synchronous parallelism, the training rate is limited by the slowest accelerator because the parameter server needs to collect parameters from all accelerators each round. Asynchronous parallelism reduces the timeout of accelerators with an asynchronous gradient update strategy (Zhang, Lee, & Qiao, 2023).

In another study (Gholami, Azad, Jin, Keutzer, & Buluc, 2018), it is described other types of parallelism that are framed within DP and are related to the training input data: batch parallelism y domain parallelism. The first is related to the assignment of groups of input data as a whole to the processes that are executed (this is the option commonly studied in the literature). The second is based on the subdivision of individual input data to processes.

Model Parallel Model parallelism partitions a model among multiple GPUs. Each GPU is responsible for the weight updates of the assigned model layers (Krizhevsky, 2014; Mirhoseini, et al., 2017). This scheme is used when the model is too large to fit in the memory of a single device, and hence data parallelism cannot be used. The model to train is partitioned and every device trains its own portion of the model using the same batch of examples (Harlap, et al., 2018; Moreno-Alvarez, Haut, Paoletti, & Rico-Gallego, 2021).

According to Janbi, Katib and Mehmood (2023), Jiang et al. (2020), Kirby et al. (2020), and Shoeybi et al. (2020) there are several strategies for model partitioning, i.e., how a neural network model is divided into smaller parts for distributed processing across multiple devices. The most common strategy is (1) layer-wise, the model is partitioned by layers, assigning each layer to a different device. For example, in Jia, Lin and Aiken (2018) a layer-wise parallelism is proposed that allows each layer in the network to use an individual parallelization strategy. (2) fine-grained, the model is partitioned into smaller blocks, which allows more granularity in the distribution. This strategy can be subdivided into (2.1) grid-based, the model is divided into a matrix, distributing different sections of the model to separate devices. (2.2) tree-structured, the model is decomposed into a tree structure, where each node in the tree is assigned to a device, allowing a high degree of parallelism and efficient communication between neighboring nodes in the tree, in Wang et al. (2023) propose a system to accelerate communication and computation on multi GPU platforms.

It is necessary to mention the work carried out by Che, Yang and Cheng (2019), where a comparison is made between data parallel and model parallel. This work focuses on three aspects: inter-GPU load balancing, inter-GPU communication, and training efficiency. In the first aspect, with data parallelism, load balancing can be easily maintained, but with model parallelism it is more complex, by dividing the complexity into different layers. In the second aspect, both data parallelism and model parallelism require communications between the GPUs and according to Takisawa, Yazaki, and Ishihata (2020) due to the communication time between the nodes, the learning performance can degrade and overload the training. Finally, in the third aspect, both data parallelism and model parallelism affect the efficiency of DNN training, that is, the rate of convergence and the accuracy of the model.

### 3 Parallelism techniques and optimizations

#### 3.1 Pipeline Parallelism

Pipeline parallelism (PP) improves the efficiency of both memory consumption and computation of deep learning training by partitioning the layers of a model into stages that can be processed in parallel. With this technique it is possible to train large neural network models that do not fit into the memory of an accelerator. In a pipeline parallelism, each piece of data moves from stage to stage, eventually producing a final result (Harlap, et al., 2018; Huang, et al., 2019; Mastoras & Gross, 2018; Narayanan, et al., 2019; Yang P. , Zhang, Zhang, Yang, & Wei, 2022). PP takes advantage of a pipelined execution strategy across different accelerators. With the pipeline execution strategy, PP can get better performance by using accelerators more efficiently compared to the naive parallelism model (Hu, et al., 2021; Zhang, Lee, & Qiao, 2023). Several authors have indicated the challenge that exists between load balancing and the

communication overhead in PP. In Kamruzzaman, Swanson, and Tullsen (2013) is explained a technique used to balance both situations. This technique provides linear speedup for several applications and outperforms prior techniques to exploit pipeline parallelism.

**Table 1.** Pipeline libraries.

Library	Description	References
DeepSpeed	Library for training large models by improving scale, speed, cost and usability, unlocking the ability to train 100 billion parameter models.	(Aminabadi, et al., 2022; Li, et al., 2024; Rajbhandari, et al., 2022; Rasley, Rajbhandari, Ruwase, & He, 2020)
GPipe	Optimization of pipeline parallelism training process. GPipe applies synchronous backward updates, and has been integrated into the PyTorch framework.	(Huang, et al., 2019; Tanaka, Taura, Hanawa, & Torisawa, 2021; Zhang, Lee, & Qiao, 2023)
PipeDream	Supports pipelined training, and automatically determines how to systematically split a given model across the available compute nodes	(Harlap, et al., 2018; Narayanan, et al., 2019; Zhang, Lee, & Qiao, 2023)
Other pipeline libraries: EdgePipe (Hu, et al., 2021; Yoon, Byeon, Kim, & Lee, 2022; Yuan, et al., 2023), BaPipe (Akintoye, Han, Zhang, Chen, & Zhang, 2022; Zhao, et al., BaPipe: Exploration of Balanced Pipeline Parallelism for DNN Training, 2021; Zhao, et al., BaPipe: Balanced Pipeline Parallelism for DNN Training, 2022), XPipe (Guan, Yin, Li, & Lu, 2020), vPipe (Zhao, et al., 2022; Zhao, et al., 2022; Zhu, 2023), , PipeMare (Yang, et al., 2021), Chimera (Li & Hoefler, 2021), TeraPipe (Li, et al., 2021), HetPipe (Park, et al., 2020), PipeTransformer (He, Li, Soltanolkotabi, & Avestimehr, 2021; Miao, et al., 2022), AutoPipe (Liu, et al., 2022; Zhang, et al., 2023), Quantpipe (Wang, et al., 2023), PipePar (Zhang, et al., 2023)		

Shows a summary of some of the main pipeline libraries that exist. The most important libraries are briefly described and other libraries of interest are included at the end of the table.

### 3.2 Hybrid Parallelism

The need to reduce the overhead of training large neural networks opened the possibility to implement approaches that involve more than one type of parallelism. Hybrid parallelism (HP) is one of the most recognized approaches today and that, due to the combination of strategies, is used to solve increasingly complex problems and with enormous volumes of computing. The HP is the combination of data parallelism and model parallelism (Howison, Bethel, & Childs, 2012). Current scientific work based on DL often requires training models with large dimensions which can make training much more expensive due to excessive memory usage (Oyama, et al., 2021).

Research to implement hybrid parallelism schemes to increase performance shows important advances. One of these advances is the combination of intralayer and interlayer parallelism to perform distributed training of DNN (Akintoye, Han, Zhang, Chen, & Zhang, 2022; Camp, Garth, Childs, Pugmire, & Joy, 2011; Oyama, et al., 2021; Song, et al., 2019; Zeng, Liu, Tang, Chang, & Li, 2021). These are investigations that describe optimizations on the parallel training of the models, demonstrating that despite the large volumes of data, memory costs can be improved. One of the reasons why it is necessary to optimize training is to be able to balance the memory capacity of the GPUs. This memory is limited by the number of computational operations required, which can result in excessively long training times (Narayanan, et al., 2021). Other investigations related to HP (Fan, et al., 2021; Li, S., et al., 2023) refer to optimization in processing times and synchronous frameworks that combine DP and Pipeline for large DNN models.

Recent research has focused on Hybrid Synchronous Parallelism (HSP), which alleviates communication contention without excessive speed degradation by removing network congestion and synchronizing all updated parameters after each iteration (Li, Mangoubi, Xu, & Guo, 2021; Li, Y., et al., 2023). In other investigations (Duan, et al., 2022; Liu, Chen, Zhou, & Ling, 2020; Song, et al., 2019; Zhou, et al., 2021) use heterogeneous Clustered HSPs (HPH) with the purpose of improving training by reducing communication time between layers and optimizing memory consumption.

### 3.3 Auto-parallelism

Auto-parallelism (AP) is a parallelization strategy that proposes to train DL models on a large scale in an efficient and practical way in several heterogeneous clusters, thus promoting an improvement in performance and memory consumption (Zheng, et al., 2022). Most efforts to improve model training have been manual. AP contributes to parallelization by developing strategies that provide improvements in the automatic conversion of sequential code into multithreaded or vectorized code to make use of available hardware devices (Liang, et al., 2023). One of these proposed models is Rhino (Zhang, et al., 2023) which is a tensor program acceleration system with AP on an Artificial Intelligence (AI) platform. Rhino efficiently searches for a parallel execution plan to speed up performance and communication within processing clusters. With AP many researchers try to avoid training algorithms that are not so highly personalized, since they contain many parameters and that these can be applied more generally (Liang, et al., 2023). Another proposed algorithm is Frontier Tracking (FT) (Cai, et al., 2022) which minimizes memory consumption when the number of devices is limited and uses the additional resources to reduce execution time. TensorOpt (Cai, et al., 2022) is based on FT and allows users to run distributed DNN training jobs without worrying about the details related to parallelization strategies for searching and encoding. Finally, there are also automation algorithms like Galvatron (Miao, et al., 2022) and Alpa (Zheng, et al., 2022). Galvatron is an algorithm that incorporates multiple dimensions of parallelism and automatically finds the most efficient hybrid parallelism strategy by automatically achieving distributed training with different GPU memory budgets. The other algorithm automates MP training of large DL models by generating execution plans that unify the DP, operators, and PP. Alpa distributes the training of large DL models in two hierarchical levels: inter-operator and intra-operator parallelism.

### 3.4 Mixture-of-Experts Parallelism

Mixture-of-Experts (MoE) models have become one of the most promising model architectures due to their significant reduction in training cost and improved performance compared to equivalent dense models. However, it presents a challenge due to the size of the models and the complex architecture (Rajbhandari, et al., 2022). MoE is an approach that has strong potential for training neural networks with up to trillions of parameters. A MoE layer contains many experts that share the same architecture and are trained by the same algorithm with a routing function that routes inputs to a few experts among all possible candidates (Chen, Deng, Wu, Gu, & Li, 2022). The huge number of parameters of current neural networks means that MoE is closely related to optimization and performance. Studies like Chen et al. (2022), Dai et al. (2022), Li, Jiang, Zhu, Wang, and Xu (2023), and Ma et al. (2018) focus on the optimization of different elements related to MoE. In Chen et al. (2022) a method for the progressive reduction of experts is proposed. The authors propose to progressively remove non-professional experts to reduce a MoE model to a single-expert dense model.

Other authors optimize based on the tasks that run the parallel application. In Ma et al. (2018) they adapt the MoE and propose a multi-task learning approach called MMoE (Multi-gate Mixture-of-Experts) to explicitly learn model relationships from data. So, they seek to build a single model that learns from multiple goals and tasks simultaneously. Other research (Dai, et al., 2022; Li, Jiang, Zhu, Wang, & Xu, 2023; Nie, et al., 2022) focuses on training. In the first work the authors divide the training in two stages to solve routing fluctuation problems with the implementation of StableMoE, in the second work they propose new communication scheduling schemes based on tensor partitioning and finally in Nie et al. (2022) an end-to-end MoE training framework called EvoMoE is proposed. This framework starts from training one single expert and gradually evolves into a large and sparse MoE structure. Furthermore, it is composed of two phases called expert-diversify phase and gate-sparsify phase. Regarding the need to increase performance, there are studies that modify the routing algorithms of MoE (Fedus, Zoph, & Shazeer, 2022; Riquelme, et al., 2024) or use up to a thousand feed-forward subnetworks contained in a layer (Sparsely-Gated Mixture-of-Experts layer) to determine a sparse combination of experts (Hazimeh, et al., 2024; Shazeer, et al., 2017).

Not only have new algorithms or optimization techniques been developed to increase performance. A large number of libraries and frameworks have also been proposed such as FastMoE (He, et al., 2021), DynaMoE (Kossmann, Jia, & Aiken, 2022), DSelect-k (Harlap, et al., 2018), and Tutel (Hwang, et al., 2023), which are designed to solve specific problems that can influence performance and scalability. The FastMoE and Tutel libraries are more focused on the parallelism of the trainings. In the case of FastMoE,

it presents a distributed MoE training system for PyTorch with common accelerators and not relying on TPUs (Tensor Processing Units). On the other hand, Tutel is termed as a scalable stack for MoE. Tutel was designed to implement dynamic adaptive parallelism and pipelining without mathematical inequivalence or tensor migration overhead.

**Table 2.** Hardware configuration

Item	Description
System name	ThetaGPU
Number of nodes	24
Number of CPU per node	2
CPU	ADM Rome
Number of CPU cores	64
Number of GPU per node	8
GPU	NVIDIA DGX A100

**Table 3.** Parallelism types performance: DDP and GPipe

#GPU	Data parallel /DDP			Model parallel / GPipe		
	Accuracy	Time (sec)	STD	Accuracy	Time (sec)	STD
2	69.856	256.83	1.041	67.25	1169.12	0.412
4	69.025	120.24	1.245	66.85	907.39	0.041
6	70.397	83.01	1.134	67.22	1067.43	0.062
8	68.146	58.22	0.092	67.71	1148.52	0.106

## 4 Assessing the parallelism of DL trainings

### 4.1 Experimental methodology

**Hardware configuration:** The system used to carry out the experiments is a supercomputer located at the Argonne Leadership Computing Facility (ALCF) called ThetaGPU. This system is part of a larger system called Theta. We use this system because it has the necessary GPUs to carry out the experiments presented in this study. Table 2 shows the characteristics of the high-performance computing system used in this study.

**Methodology:** In order to test some characteristics of the parallelism types, we decided to implement distributed training with the main types of parallelism in DL. Data parallelism via Horovod and model parallelism with the GPipe library. Both types of parallelism are implemented using the DL Pytorch framework.

It is important to clarify that the two types of parallelism have different characteristics. Model parallelism is used with neural network models that do not fit in the memory of an accelerator (GPU), since it is capable of training a neural network by dividing it into multiple partitions according to the number of GPUs available. In the following experiments, a small neural network (compared to trillion-parameter neural networks) is used with the sole intention of experimentally visualizing the differences between types of parallelism and laying the foundation for future studies.

All the experiments carried out used CIFAR100 as dataset and the ResNet18 neural network. This neural network is a reduced version of ResNet, which allows us to evaluate types of parallelism quickly, due to reasonable training times. In the case of training with model parallelism, the neural network used is transformed to a sequential structure so that it can be correctly computed. Additionally, pipeline parallelism is implemented by dividing batches into micro-batches to make the GPUs work in parallel as much as possible.

The results shown are generated by running 10 epochs. 10 repetitions of each experiment were carried out in order to obtain statistically acceptable results. Tables 3 and 4 show the results of these experiments. Table 3 shows the results of 32-bit training and table 4 shows the results of 16-bit training. To implement 16 bit training we use the NVIDIA APEX library activating the O3 mode (FP16).

**Table 4.** Parallelism with 16-bit floating-point precision

#GPU	Data parallel /DDP			Model parallel / GPipe		
	Accuracy	Time (sec)	STD	Accuracy	Time (sec)	STD
2	69.856	256.83	1.041	67.25	1169.12	0.412
4	69.025	120.24	1.245	66.85	907.39	0.041
6	70.397	83.01	1.134	67.22	1067.43	0.062
8	68.146	58.22	0.092	67.71	1148.52	0.106

## 4.2 Results analysis

The first experiment allows us to compare the performance of the two types of parallelism. Experiments were run on 2, 4, 6, and 8 GPUs. In the case of model parallel training we use an automatic layer distribution by time. This distribution traces the elapsed time of each layer to determine how many layers to allocate per GPU.

Both table 3 and table 4 show execution times of trainings. However, if we look only at the results of table 3 we can make some important observations: 1) The accuracy generated by the two types of parallelism is similar. This shows us that despite the fact that model parallelism is oriented to the training of large neural networks, the results are acceptable and comparable with those of data parallelism. 2) In the previous premise the accuracy is similar. However, the execution times are very different between both types of parallelism. The performance difference when comparing the time of 8 GPUs is noticeable. Data parallel shows a time of 58.22 seconds while model parallel reports a time of 1148.52 seconds. Clearly model parallel performance is degraded by data waiting between GPUs. Even though portions of the neural network run on different GPUs, the neural network is still a sequence of layers that must be respected. 3) Data parallel performance increases when scaling on GPUs. This is very different from model parallel where the time with 2 GPUs is similar to the time with 8 GPUs. It is interesting how with more GPUs a higher overhead is generated because the neural network is even more partitioned. With 2 GPUs the processing is slower, but there is less waiting for the GPUs.

Regarding the results of the table 4 we can highlight two important aspects: First, for both data parallel and model parallel there is a slight degradation of accuracy. This is generated by the reduction in training accuracy. Second, the results are similar to 32-bit training, taking into account the aforementioned aspects regarding the results of the table 3.

## 5 Concluding Remarks

Neural networks are gaining more and more popularity due to their great power to help solve complex problems in many fields of science. In parallel, the complexity of neural network models increases along with the hardware requirements. Due to how difficult and expensive the hardware is, many strategies have been developed to parallelize the processes involved in training neural networks.

In this study we tried to cover the most important concepts together with a large amount of research carried out in the field. The main types of parallelism (data and model parallelism) and the variants that exist, such as pipeline parallelism or hybrid parallelism, were taken into account. In addition, the topic of

Mixture of Experts was addressed, which is relatively new in the field of deep learning. In this work, experimentation with the two main types of parallelism was presented and the result allowed us to contrast in a general and brief way some differences that exist between these types. Finally, it is clear that in this work it is not possible to cover all the research and developments that exist related to parallelism in DL. However, readers will be able to generate a very clear idea of the current panorama and the complexity that surrounds this field.

## ORCID iD

Hairol Romero-Sandí  <https://orcid.org/0000-0002-3199-1244>

Elvis Rojas  <https://orcid.org/0000-0002-4238-0908>

## References

- Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., . . . Zheng, X. (2016, March 14). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *arXiv*(1603.04467 [cs.DC]). doi:10.48550/arXiv.1603.04467
- Agarwal, S., Yan, C., Zhang, Z., & Venkataraman, S. (2023, October). BagPipe: Accelerating Deep Recommendation Model Training. *SOSP '23: Proceedings of the 29th Symposium on Operating Systems Principles (SOSP '23)* (pp. 348-363). Koblenz, Germany: Association for Computing Machinery, New York, NY, USA. doi:10.1145/3600006.3613142
- Akintoye, S. B., Han, L., Zhang, X., Chen, H., & Zhang, D. (2022). A Hybrid Parallelization Approach for Distributed and Scalable Deep Learning. *IEEE Access*, *10*, 77950-77961. doi:10.1109/ACCESS.2022.3193690
- Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). Understanding of a convolutional neural network. *2017 International Conference on Engineering and Technology (ICET)* (pp. 1-6). Antalya, Turkey: IEEE. doi:10.1109/ICEngTechnol.2017.8308186
- Aminabadi, R. Y., Rajbhandari, S., Awan, A. A., Li, C., Li, D., Zheng, E., . . . He, Y. (2022). DeepSpeed-Inference: Enabling Efficient Inference of Transformer Models at Unprecedented Scale. *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis* (pp. 1-15). Dallas, TX, USA: IEEE. doi:10.1109/SC41404.2022.00051
- Batur Dinler, Ö., Şahin, B. C., & Abualigah, L. (2021, November 30). Comparison of Performance of Phishing Web Sites with Different DeepLearning4J Models. *European Journal of Science and Technology*(28), 425-431. doi:10.31590/ejosat.1004778
- Ben-Nun, T., & Hoefler, T. (2019, August 30). Demystifying Parallel and Distributed Deep Learning: An In-depth Concurrency Analysis. *ACM Computing Surveys (CSUR)*, *52*(4), 1-43, Article No. 65. doi:10.1145/3320060
- Cai, Z., Yan, X., Ma, K., Yidi, W., Huang, Y., Cheng, J., . . . Yu, F. (2022, August 1). TensorOpt: Exploring the Tradeoffs in Distributed DNN Training With Auto-Parallelism. *IEEE Transactions on Parallel and Distributed Systems*, *33*(8), 1967-1981. doi:10.1109/TPDS.2021.3132413
- Camp, D., Garth, C., Childs, H., Pugmire, D., & Joy, K. (2011, November). Streamline Integration Using MPI-Hybrid Parallelism on a Large Multicore Architecture. *IEEE Transactions on Visualization and Computer Graphics*, *17*(11), 1702-1713. doi:10.1109/TVCG.2010.259
- Chen, C.-C., Yang, C.-L., & Cheng, H.-Y. (2019, October 28). Efficient and Robust Parallel DNN Training through Model Parallelism on Multi-GPU Platform. *arXiv:1809.02839v4 [cs.DC]*. doi:10.48550/arXiv.1809.02839
- Chen, M. (2023, March 15). Analysis of Data Parallelism Methods with Deep Neural Network. *ITCE '22: Proceedings of the 2022 6th International Conference on Electronic Information Technology and Computer Engineering* (pp. 1857-1861). Xiamen, China: Association for Computing Machinery, New York, NY, USA. doi:10.1145/3573428.3573755
- Chen, T., Huang, S., Xie, Y., Jiao, B., Jiang, D., Zhou, H., . . . Wei, F. (2022, June 2). Task-Specific Expert Pruning for Sparse Mixture-of-Experts. *arXiv:2206.00277v2 [cs.LG]*, 1-13. doi:10.48550/arXiv.2206.00277
- Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., . . . Zhang, Z. (2015, December 3). MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems. *arXiv:1512.01274v1 [cs.DC]*, 1-6. doi:10.48550/arXiv.1512.01274
- Chen, Z., Deng, Y., Wu, Y., Gu, Q., & Li, Y. (2022). Towards Understanding the Mixture-of-Experts Layer in Deep Learning. In A. H. Oh, A. Agarwal, D. Belgrave, & K. Cho (Ed.), *Advances in Neural Information Processing Systems*. New Orleans, Louisiana, USA. Retrieved from <https://openreview.net/forum?id=MaYzugDmQV>
- Collobert, R., Bengio, S., & Mariétoz, J. (2002, October 30). *Torch: a modular machine learning software library*. Research Report, IDIAP, Martigny, Switserland. Retrieved from <https://publications.idiap.ch/downloads/reports/2002/rr02-46.pdf>
- Dai, D., Dong, L., Ma, S., Zheng, B., Sui, Z., Chang, B., & Wei, F. (2022, May). StableMoE: Stable Routing Strategy for Mixture of Experts. In S. Muresan, P. Nakov, & A. Villavicencio (Ed.),



- Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics. 1: Long Papers*, pp. 7085–7095. Dublin, Ireland: Association for Computational Linguistics. doi:10.18653/v1/2022.acl-long.489
- Duan, Y., Lai, Z., Li, S., Liu, W., Ge, K., Liang, P., & Li, D. (2022). HPH: Hybrid Parallelism on Heterogeneous Clusters for Accelerating Large-scale DNNs Training. *2022 IEEE International Conference on Cluster Computing (CLUSTER)* (pp. 313-323). Heidelberg, Germany: IEEE. doi:10.1109/CLUSTER51413.2022.00043
- Fan, S., Rong, Y., Meng, C., Cao, Z., Wang, S., Zheng, Z., . . . Lin, W. (2021, February). DAPPLE: a pipelined data parallel approach for training large models. *PPoPP '21: Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming* (pp. 431-445). Virtual Event, Republic of Korea: Association for Computing Machinery, New York, NY, USA. doi:10.1145/3437801.3441593
- Fedus, W., Zoph, B., & Shazeer, N. (2022, January 1). Switch transformers: scaling to trillion parameter models with simple and efficient sparsity. (A. Clark, Ed.) *The Journal of Machine Learning Research*, 23(1), Article No. 120, 5232-5270. Retrieved from <https://dl.acm.org/doi/abs/10.5555/3586589.3586709>
- Gholami, A., Azad, A., Jin, P., Keutzer, K., & Buluc, A. (2018). Integrated Model, Batch, and Domain Parallelism in Training Neural Networks. *SPAA '18: Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures* (pp. 77-86). Vienna, Austria: Association for Computing Machinery, New York, NY, USA. doi:10.1145/3210377.3210394
- Guan, L., Yin, W., Li, D., & Lu, X. (2020, November 9). XPipe: Efficient Pipeline Model Parallelism for Multi-GPU DNN Training. *arXiv:1911.04610v3 [cs.LG]*. doi:10.48550/arXiv.1911.04610
- Harlap, A., Narayanan, D., Phanishayee, A., Seshadri, V., Devanur, N., Ganger, G., & Gibbons, P. (2018, June 8). PipeDream: Fast and Efficient Pipeline Parallel DNN Training. *arXiv:1806.03377v1 [cs.DC]*, 1-14. doi:10.48550/arXiv.1806.03377
- Hazimeh, H., Zhao, Z., Aakanksha, C., Sathiamoorthy, M., Chen, Y., Mazumder, R., . . . Chi, E. H. (2024). DSelect-k: differentiable selection in the mixture of experts with applications to multi-task learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, & J. W. Vaughan (Eds.), *NIPS'21: Proceedings of the 35th International Conference on Neural Information Processing Systems. Article No. 2246*, pp. 29335-29347. Curran Associates Inc., Red Hook, NY, USA. doi:10.5555/3540261.3542507
- He, C., Li, S., Soltanolkotabi, M., & Avestimehr, S. (2021, July). PipeTransformer: Automated Elastic Pipelining for Distributed Training of Large-scale Models. In M. Meila, & T. Zhang (Eds.), *Proceedings of the 38th International Conference on Machine Learning*. 139, pp. 4150-4159. PMLR. Retrieved from <https://proceedings.mlr.press/v139/he21a.html>
- He, J., Qiu, J., Zeng, A., Yang, Z., Zhai, J., & Tang, J. (2021, March 24). FastMoE: A Fast Mixture-of-Expert Training System. *arXiv:2103.13262v1 [cs.LG]*, 1-11. doi:10.48550/arXiv.2103.13262
- Hey, T. (2020, October 1). *Opportunities and Challenges from Artificial Intelligence and Machine Learning for the Advancement of Science, Technology, and the Office of Science Missions*. Technical Report, USDOE Office of Science (SC), Advanced Scientific Computing Research (ASCR), United States. doi:10.2172/1734848
- Hopfield, J. J. (1988, September). Artificial neural networks. *IEEE Circuits and Devices Magazine*, 4(5), 3-10. doi:10.1109/101.8118
- Howison, M., Bethel, E. W., & Childs, H. (2012, January). Hybrid Parallelism for Volume Rendering on Large-, Multi-, and Many-Core Systems. *IEEE Transactions on Visualization and Computer Graphics*, 18(1), 17-29. doi:10.1109/TVCG.2011.24
- Hu, Y., Imes, C., Zhao, X., Kundu, S., Beerel, P. A., Crago, S. P., & Walters, J. P. (2021, October 28). Pipeline Parallelism for Inference on Heterogeneous Edge Computing. *arXiv:2110.14895v1 [cs.DC]*, 1-12. doi:10.48550/arXiv.2110.14895
- Huang, Y., Cheng, Y., Bapna, A., Firat, O., Chen, M. X., Chen, D., . . . Chen, Z. (2019, December 8). GPipe: efficient training of giant neural networks using pipeline parallelism. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, & E. B. Fox (Eds.), *Proceedings of the 33rd International Conference on Neural Information Processing Systems (NIPS'19). Article No. 10*, pp. 103 - 112. Vancouver, BC, Canada: Curran Associates Inc., Red Hook, NY, USA. doi:10.5555/3454287.3454297
- Hwang, C., Cui, W., Xiong, Y., Yang, Z., Liu, Z., Hu, H., . . . Xiong, Y. (2023, June 5). Tutel: Adaptive Mixture-of-Experts at Scale. *arXiv:2206.03382v2 [cs.DC]*, 1-19. doi:10.48550/arXiv.2206.03382

- Janbi, N., Katib, I., & Mehmood, R. (2023, May). Distributed artificial intelligence: Taxonomy, review, framework, and reference architecture. *Intelligent Systems with Applications*, 18, 200231. doi:10.1016/j.iswa.2023.200231
- Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., . . . Darrell, T. (2014, November 3). Caffe: Convolutional Architecture for Fast Feature Embedding. *MM '14: Proceedings of the 22nd ACM international conference on Multimedia* (pp. 675-678). Orlando, Florida, USA: Association for Computing Machinery, New York, NY, USA. doi:10.1145/2647868.2654889
- Jia, Z., Lin, S., Qi, C. R., & Aiken, A. (2018). Exploring Hidden Dimensions in Accelerating Convolutional Neural Networks. In J. Dy, & A. Krause (Ed.), *Proceedings of the 35th International Conference on Machine Learning*. 80, pp. 2274-2283. PMLR. Retrieved from <https://proceedings.mlr.press/v80/jia18a.html>
- Jiang, W., Zhang, Y., Liu, P., Peng, J., Yang, L. T., Ye, G., & Jin, H. (2020, January). Exploiting potential of deep neural networks by layer-wise fine-grained parallelism. *Future Generation Computer Systems*, 102, 210-221. doi:10.1016/j.future.2019.07.054
- Kamruzzaman, M., Swanson, S., & Tullsen, D. M. (2013, November 17). Load-balanced pipeline parallelism. *SC '13: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis. Article No. 14*, pp. 1-12. Denver, Colorado, USA: Association for Computing Machinery, New York, NY, USA. doi:10.1145/2503210.2503295
- Kirby, A. C., Samsi, S., Jones, M., Reuther, A., Kepner, J., & Gadepally, V. (2020, September). Layer-Parallel Training with GPU Concurrency of Deep Residual Neural Networks via Nonlinear Multigrid. (2007.07336 [cs.LG]), 1-7. doi:10.1109/HPEC43674.2020.9286180
- Kossmann, F., Jia, Z., & Aiken, A. (2022, August 2). Optimizing Mixture of Experts using Dynamic Recompilations. *arXiv:2205.01848v2 [cs.LG]*, 1-13. doi:10.48550/arXiv.2205.01848
- Krizhevsky, A. (2014, April 26). One weird trick for parallelizing convolutional neural networks. *arXiv:1404.5997v2 [cs.NE]*, 1-7. doi:10.48550/arXiv.1404.5997
- Kukačka, J., Golkov, V., & Cremers, D. (2017, October 29). Regularization for Deep Learning: A Taxonomy. *arXiv:1710.10686v1 [cs.LG]*, 1-23. doi:10.48550/arXiv.1710.10686
- Li, C., Yao, Z., Wu, X., Zhang, M., Holmes, C., Li, C., & He, Y. (2024, January 14). DeepSpeed Data Efficiency: Improving Deep Learning Model Quality and Training Efficiency via Efficient Data Sampling and Routing. *arXiv:2212.03597v3 [cs.LG]*, 1-19. doi:10.48550/arXiv.2212.03597
- Li, J., Jiang, Y., Zhu, Y., Wang, C., & Xu, H. (2023, July). Accelerating Distributed MoE Training and Inference with Lina. *2023 USENIX Annual Technical Conference (USENIX ATC 23)* (pp. 945-959). USENIX Association, Boston, MA, USA. Retrieved from <https://www.usenix.org/conference/atc23/presentation/li-jiamin>
- Li, S., & Hoefler, T. (2021, November). Chimera: efficiently training large-scale neural networks with bidirectional pipelines. *SC '21: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. Article No. 27*, pp. 1-14. St. Louis, Missouri, USA: Association for Computing Machinery, New York, NY, USA. doi:10.1145/3458817.3476145
- Li, S., Liu, H., Bian, Z., Fang, J., Huang, H., Liu, Y., . . . You, Y. (2023, August). Colossal-AI: A Unified Deep Learning System For Large-Scale Parallel Training. *ICPP '23: Proceedings of the 52nd International Conference on Parallel Processing* (pp. 766-775). Salt Lake City, UT, USA: Association for Computing Machinery, New York, NY, USA. doi:10.1145/3605573.3605613
- Li, S., Mangoubi, O., Xu, L., & Guo, T. (2021). Sync-Switch: Hybrid Parameter Synchronization for Distributed Deep Learning. *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)* (pp. 528-538). DC, USA: IEEE. doi:10.1109/ICDCS51616.2021.00057
- Li, Y., Huang, J., Li, Z., Zhou, S., Jiang, W., & Wang, J. (2023). HSP: Hybrid Synchronous Parallelism for Fast Distributed Deep Learning. *ICPP '22: Proceedings of the 51st International Conference on Parallel Processing* (pp. 1-11). Bordeaux, France: Association for Computing Machinery, New York, NY, USA. doi:10.1145/3545008.3545024
- Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2022, December). A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. *IEEE Transactions on Neural Networks and Learning Systems*, 33(12), 6999-7019. doi:10.1109/TNNLS.2021.3084827
- Li, Z., Zhuang, S., Guo, S., Zhuo, D., Zhang, H., Song, D., & Stoica, I. (2021). TeraPipe: Token-Level Pipeline Parallelism for Training Large-Scale Language Models. In M. Meila, & T. Zhang (Ed.), *Proceedings of the 38th International Conference on Machine Learning*. 139, pp. 6543-6552. PMLR. Retrieved from <https://proceedings.mlr.press/v139/li21y.html>

- Liang, P., Tang, Y., Zhang, X., Bai, Y., Su, T., Lai, Z., . . . Li, D. (2023, August). A Survey on Auto-Parallelism of Large-Scale Deep Learning Training. *IEEE Transactions on Parallel and Distributed Systems*, 34(8), 2377-2390. doi:10.1109/TPDS.2023.3281931
- Liu, D., Chen, X., Zhou, Z., & Ling, Q. (2020, May 15). HierTrain: Fast Hierarchical Edge AI Learning With Hybrid Parallelism in Mobile-Edge-Cloud Computing. *IEEE Open Journal of the Communications Society*, 1, 634-645. doi:10.1109/OJCOMS.2020.2994737
- Liu, W., Lai, Z., Li, S., Duan, Y., Ge, K., & Li, D. (2022). AutoPipe: A Fast Pipeline Parallelism Approach with Balanced Partitioning and Micro-batch Slicing. *2022 IEEE International Conference on Cluster Computing (CLUSTER)* (pp. 301-312). Heidelberg, Germany: IEEE. doi:10.1109/CLUSTER51413.2022.00042
- Ma, J., Zhao, Z., Yi, X., Chen, J., Hong, L., & Chi, E. H. (2018, July). Modeling Task Relationships in Multi-task Learning with Multi-gate Mixture-of-Experts. *KDD '18: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 1930-1939). London, United Kingdom: Association for Computing Machinery, New York, NY, USA. doi:10.1145/3219819.3220007
- Manaswi, N. K. (2018). Understanding and Working with Keras. In N. K. Manaswi, *Deep Learning with Applications Using Python: Chatbots and Face, Object, and Speech Recognition With TensorFlow and Keras* (pp. 31-43). Berkeley, CA, USA: Apress. doi:10.1007/978-1-4842-3516-4
- Mastoras, A., & Gross, T. R. (2018, February 24). Understanding Parallelization Tradeoffs for Linear Pipelines. In Q. Chen, Z. Huang, & P. Balaji (Ed.), *PMAM'18: Proceedings of the 9th International Workshop on Programming Models and Applications for Multicores and Manycores* (pp. 1-10). Vienna, Austria: Association for Computing Machinery, New York, NY, USA. doi:10.1145/3178442.3178443
- Miao, X., Wang, Y., Jiang, Y., Shi, C., Nie, X., Zhang, H., & Cui, B. (2022, November 1). Galvatron: Efficient Transformer Training over Multiple GPUs Using Automatic Parallelism. *Proceedings of the VLDB Endowment*, 16(3), 470-479. doi:10.14778/3570690.3570697
- Mirhoseini, A., Pham, H., Le, Q. V., Steiner, B., Larsen, R., Zhou, Y., . . . Dean, J. (2017, June 25). Device Placement Optimization with Reinforcement Learning. *arXiv:1706.04972v2 [cs.LG]*, 1-11. doi:10.48550/arXiv.1706.04972
- Mittal, S., & Vaishay, S. (2019, October). A survey of techniques for optimizing deep learning on GPUs. *Journal of Systems Architecture*, 99, 101635. doi:10.1016/j.sysarc.2019.101635
- Moreno-Alvarez, S., Haut, J. M., Paoletti, M. E., & Rico-Gallego, J. A. (2021, June 21). Heterogeneous model parallelism for deep neural networks. *Neurocomputing*, 441, 1-12. doi:10.1016/j.neucom.2021.01.125
- Narayanan, D., Harlap, A., Phanishayee, A., Seshadri, V., Devanur, N. R., Ganger, G. R., . . . Zaharia, M. (2019, October). PipeDream: generalized pipeline parallelism for DNN training. *SOSP '19: Proceedings of the 27th ACM Symposium on Operating Systems Principles* (pp. 1-15). Huntsville, Ontario, Canada: Association for Computing Machinery, New York, NY, USA. doi:10.1145/3341301.3359646
- Narayanan, D., Shoeybi, M., Casper, J., LeGresley, P., Patwary, M., Korthikanti, V., . . . Zaharia, M. (2021). Efficient large-scale language model training on GPU clusters using megatron-LM. *SC '21: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. Article No. 58*, pp. 1-15. St. Louis, Missouri, USA: Association for Computing Machinery, New York, NY, USA. doi:10.1145/3458817.3476209
- Nie, X., Miao, X., Cao, S., Ma, L., Liu, Q., Xue, J., . . . Cui, B. (2022, October 9). EvoMoE: An Evolutional Mixture-of-Experts Training Framework via Dense-To-Sparse Gate. *arXiv:2112.14397v2 [cs.LG]*, 1-14. doi:10.48550/arXiv.2112.14397
- Oyama, Y., Maruyama, N., Dryden, N., McCarthy, E., Harrington, P., Balewski, J., . . . Van Essen, B. (2021, July 1). The Case for Strong Scaling in Deep Learning: Training Large 3D CNNs With Hybrid Parallelism. *IEEE Transactions on Parallel and Distributed Systems*, 32(7), 1641-1652. doi:10.1109/TPDS.2020.3047974
- Park, J. H., Yun, G., Yi, C. M., Nguyen, N. T., Lee, S., Choi, J., . . . Choi, Y.-r. (2020, July). HetPipe: Enabling Large DNN Training on (Whimpy) Heterogeneous GPU Clusters through Integration of Pipelined Model Parallelism and Data Parallelism. *2020 USENIX Annual Technical Conference (USENIX ATC 20)* (pp. 307-321). USENIX Association. Retrieved from <https://www.usenix.org/conference/atc20/presentation/park>

- Pouyanfar, S., Sadiq, S., Yan, Y., Tian, H., Tao, Y., Presa, M. R., . . . Iyengar, S. S. (2018, September 18). A Survey on Deep Learning: Algorithms, Techniques, and Applications. *ACM Computing Surveys (CSUR)*, 51(5), 1-36, Article No. 92. doi:10.1145/3234150
- Rajbhandari, S., Li, C., Yao, Z., Zhang, M., Aminabadi, R. Y., Awan, A. A., . . . He, Y. (2022, July). DeepSpeed-MoE: Advancing Mixture-of-Experts Inference and Training to Power Next-Generation AI Scale. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, & S. Sabato (Ed.), *Proceedings of the 39th International Conference on Machine Learning*. 162, pp. 18332-18346. PMLR. Retrieved from <https://proceedings.mlr.press/v162/rajbhandari22a.html>
- Rasley, J., Rajbhandari, S., Ruwase, O., & He, Y. (2020, August). DeepSpeed: System Optimizations Enable Training Deep Learning Models with Over 100 Billion Parameters. *KDD '20: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 3505 - 3506). Virtual Event, CA, USA: Association for Computing Machinery, New York, NY, USA. doi:10.1145/3394486.3406703
- Ravanelli, M., Parcollet, T., & Bengio, Y. (2019). The Pytorch-kaldi Speech Recognition Toolkit. *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 6465-6469). Brighton, UK: IEEE. doi:10.1109/ICASSP.2019.8683713
- Riquelme, C., Puigcerver, J., Mustafa, B., Neumann, M., Jenatton, R., Pinto, A. S., . . . Hounsby, N. (2024, December). Scaling vision with sparse mixture of experts. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, & J. W. Vaughan (Ed.), *NIPS'21: Proceedings of the 35th International Conference on Neural Information Processing Systems*. Article No. 657, pp. 8583-8595. Curran Associates Inc., Red Hook, NY, USA. doi:10.5555/3540261.3540918
- Rojas, E., Quirós-Corella, F., Jones, T., & Meneses, E. (2022). Large-Scale Distributed Deep Learning: A Study of Mechanisms and Trade-Offs with PyTorch. In I. Gitler, C. J. Barrios Hernández, & E. Meneses (Ed.), *High Performance Computing. CARLA 2021. Communications in Computer and Information Science*. 1540, pp. 177-192. Springer, Cham. doi:10.1007/978-3-031-04209-6\_13
- Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., & Dean, J. (2017). Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. *International Conference on Learning Representations (ICLR 2017)*, (pp. 1-19). Toulon, France. Retrieved from <https://openreview.net/forum?id=B1ckMDqlg>
- Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J., & Catanzaro, B. (2020, March 13). Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism. *arXiv:1909.08053v4 [cs.CL]*, 1-15. doi:10.48550/arXiv.1909.08053
- Song, L., Mao, J., Zhuo, Y., Qian, X., Li, H., & Chen, Y. (2019). HyPar: Towards Hybrid Parallelism for Deep Learning Accelerator Array. *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)* (pp. 56-68). Washington, DC, USA: IEEE. doi:10.1109/HPCA.2019.00027
- Stevens, R., Taylor, V., Nichols, J., Maccabe, A. B., Yelick, K., & Brown, D. (2020, February 1). *AI for Science: Report on the Department of Energy (DOE) Town Halls on Artificial Intelligence (AI) for Science*. Technical Report, USDOE; Lawrence Berkeley National Laboratory (LBNL); Argonne National Laboratory (ANL); Oak Ridge National Laboratory (ORNL), United States. doi:10.2172/1604756
- Subhlok, J., Stichnoth, J. M., O'Hallaron, D. O., & Gross, T. (1993, July 1). Exploiting task and data parallelism on a multicomputer. *ACM SIGPLAN Notices*, 28(7), 13-22. doi:10.1145/173284.155334
- Takisawa, N., Yazaki, S., & Ishihata, H. (2020). Distributed Deep Learning of ResNet50 and VGG16 with Pipeline Parallelism. *2020 Eighth International Symposium on Computing and Networking Workshops (CANDARW)* (pp. 130-136). Naha, Japan: IEEE. doi:10.1109/CANDARW51189.2020.00036
- Tanaka, M., Taura, K., Hanawa, T., & Torisawa, K. (2021). Automatic Graph Partitioning for Very Large-scale Deep Learning. *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)* (pp. 1004-1013). Portland, OR, USA: IEEE. doi:10.1109/IPDPS49936.2021.00109
- Verbraeken, J., Wolting, M., Katzy, J., Kloppenburg, J., Verbelen, T., & Rellermeyer, J. S. (2020). A Survey on Distributed Machine Learning. *ACM Computing Surveys (CSUR)*, 53(2), 1-33, Article No. 30. doi:10.1145/3377454
- Wang, H., Imes, C., Kundu, S., Beerel, P. A., Crago, S. P., & Walters, J. P. (2023). Quantpipe: Applying Adaptive Post-Training Quantization For Distributed Transformer Pipelines In Dynamic Edge

- Environments. *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 1-5). Rhodes Island, Greece: IEEE.  
doi:10.1109/ICASSP49357.2023.10096632
- Wang, S.-C. (2003). Artificial Neural Network. In S.-C. Wang, *Interdisciplinary Computing in Java Programming* (1 ed., Vol. 743, pp. 81-100). Boston, MA, USA: Springer. doi:10.1007/978-1-4615-0377-4\_5
- Wang, Y., Feng, B., Wang, Z., Geng, T., Barker, K., Li, A., & Ding, Y. (2023, July). MGG: Accelerating Graph Neural Networks with Fine-Grained Intra-Kernel Communication-Computation Pipelining on Multi-GPU Platforms. *17th USENIX Symposium on Operating Systems Design and Implementation (OSDI 23)* (pp. 779-795). Boston, MA, USA: USENIX Association. Retrieved from <https://www.usenix.org/conference/osdi23/presentation/wang-yuke>
- Wu, J. (2017, May 1). *Introduction to Convolutional Neural Networks*. Nanjing University, National Key Lab for Novel Software Technology, China. Retrieved from <https://cs.nju.edu.cn/wujx/paper/CNN.pdf>
- Yang, B., Zhang, J., Li, J., Ré, C., Aberger, C. R., & De Sa, C. (2021, March 15). *Proceedings of the 4th Machine Learning and Systems Conference*, 3, pp. 269-296. San Jose, CA, USA. Retrieved from [https://proceedings.mlsys.org/paper\\_files/paper/2021/file/9412531719be7ccf755c4ff98d0969dc-Paper.pdf](https://proceedings.mlsys.org/paper_files/paper/2021/file/9412531719be7ccf755c4ff98d0969dc-Paper.pdf)
- Yang, P., Zhang, X., Zhang, W., Yang, M., & Wei, H. (2022). Group-based Interleaved Pipeline Parallelism for Large-scale DNN Training. *The Tenth International Conference on Learning Representations (ICLR 2022)*, (pp. 1-15). Retrieved from <https://openreview.net/forum?id=cw-EmNq5zfD>
- Yoon, J., Byeon, Y., Kim, J., & Lee, H. (2022, July 15). EdgePipe: Tailoring Pipeline Parallelism With Deep Neural Networks for Volatile Wireless Edge Devices. *IEEE Internet of Things Journal*, 9(14), 11633 - 11647. doi:10.1109/JIOT.2021.3131407
- Yuan, L., He, Q., Chen, F., Dou, R., Jin, H., & Yang, Y. (2023, April 30). PipeEdge: A Trusted Pipelining Collaborative Edge Training based on Blockchain. In Y. Ding, J. Tang, J. Sequeda, L. Aroyo, C. Castillo, & G.-J. Houben (Ed.), *WWW '23: Proceedings of the ACM Web Conference 2023* (pp. 3033-3043). Austin, TX, USA: Association for Computing Machinery, New York, NY, USA. doi:10.1145/3543507.3583413
- Zeng, Z., Liu, C., Tang, Z., Chang, W., & Li, K. (2021). Training Acceleration for Deep Neural Networks: A Hybrid Parallelization Strategy. *2021 58th ACM/IEEE Design Automation Conference (DAC)* (pp. 1165-1170). San Francisco, CA, USA: IEEE.  
doi:10.1109/DAC18074.2021.9586300
- Zhang, J., Niu, G., Dai, Q., Li, H., Wu, Z., Dong, F., & Wu, Z. (2023, October 28). PipePar: Enabling fast DNN pipeline parallel training in heterogeneous GPU clusters. *Neurocomputing*, 555, 126661. doi:10.1016/j.neucom.2023.126661
- Zhang, P., Lee, B., & Qiao, Y. (2023, October). Experimental evaluation of the performance of Gpipe parallelism. *Future Generation Computer Systems*, 147, 107-118.  
doi:10.1016/j.future.2023.04.033
- Zhang, S., Diao, L., Wang, S., Cao, Z., Gu, Y., Si, C., . . . Lin, W. (2023, February 16). Auto-Parallelizing Large Models with Rhino: A Systematic Approach on Production AI Platform. *arXiv:2302.08141v1 [cs.DC]*, 1-16. doi:10.48550/arXiv.2302.08141
- Zhao, L., Xu, R., Wang, T., Tian, T., Wang, X., Wu, W., . . . Jin, X. (2021, January 14). BaPipe: Exploration of Balanced Pipeline Parallelism for DNN Training. *arXiv:2012.12544v2 [cs.DC]*. doi:10.48550/arXiv.2012.12544
- Zhao, L., Xu, R., Wang, T., Tian, T., Wang, X., Wu, W., . . . Jin, X. (2022). BaPipe: Balanced Pipeline Parallelism for DNN Training. *Parallel Processing Letters*, 32(03n04), 2250005, 1-17.  
doi:10.1142/S0129626422500050
- Zhao, S., Li, F., Chen, X., Guan, X., Jiang, J., Huang, D., . . . Cui, H. (2022, March 1). vPipe: A Virtualized Acceleration System for Achieving Efficient and Scalable Pipeline Parallel DNN Training. *IEEE Transactions on Parallel and Distributed Systems*, 33(3), 489-506.  
doi:10.1109/TPDS.2021.3094364
- Zheng, L., Li, Z., Zhang, H., Zhuang, Y., Chen, Z., Huang, Y., . . . Stoica, I. (2022, July). Alpa: Automating Inter- and Intra-Operator Parallelism for Distributed Deep Learning. *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)* (pp. 559-578). Carlsbad,

- CA, USA: USENIX Association. Retrieved from  
<https://www.usenix.org/conference/osdi22/presentation/zheng-lianmin>
- Zhou, Q., Guo, S., Qu, Z., Li, P., Li, L., Guo, M., & Wang, K. (2021, May 1). Petrel: Heterogeneity-Aware Distributed Deep Learning Via Hybrid Synchronization. *IEEE Transactions on Parallel and Distributed Systems*, 32(5), 1030-1043. doi:10.1109/TPDS.2020.3040601
- Zhu, X. (2023, April 28). Implement deep neuron networks on VPipe parallel system: a ResNet variant implementation. In X. Li (Ed.), *Proceedings Third International Conference on Artificial Intelligence and Computer Engineering (ICAICE 2022)*. 12610, p. 126104I. Wuhan, China: International Society for Optics and Photonics, SPIE. doi:10.1117/12.2671359