

CHAPTER 5

SYSTEM EVALUATION

In this chapter, we first explain our experiments setup. We also describe the computers and data sets used to evaluate the performance of our proposed framework. Finally, the results and analysis of our experiments are presented.

5.1 Experiments Setup

The lab of KFUPM Information and Computer Science department is used for running our experiments (see Figure 5.1).



Figure 5.1: KFUPM/ICS Rack used for running our experiments

Three physical machines of Intel Xenon CPU running at 2.00GHz and 64GB RAM each are dedicated for performing our experiments. Each machine is of 2-processors, each processor has 12-cores (see Table 5.1).

Table 5.1: Physical Hardware Specifications

Property	Specification
No. of physical machines	3
CPU Type	Intel Xenon
CPU Speed	2.0GHz
No. of processors/Machine	2
No of Cores/Processor	12
RAM/Machine	64GB
Ethernet	2 of 100.0 MBPS

28-Virtual machines are created on the 3-physical machines. Each virtual machine has 2-cores and 6.0GB RAM (see Table 5.2).

Table 5.2: Virtual Machines Specifications

Property	Specification
No. of virtual machines	28
No. of Cores/Machine	2
RAM/Machine	6GB

Table 5.3 describes in details the software installed for each node in the cluster.

Table 5.3: Software Specifications

Type	version
Oracle Linux	7.6
Hadoop	3.1.0
Yarn	3.1.0
Spark	2.3.0
TensorFlow	1.12
TensorFrames	1.1
Python	3.6.0

VMWare workstation version 14.0 for windows platform software has been installed and configured for the three physical machines. One physical machine has been configured to contain one virtual Hadoop master node and eight virtual Hadoop workers, while the other two physical machines configured to contain ten virtual Hadoop workers for each (i.e., total of one Hadoop master node and 28 Hadoop workers). Table 5.4 shows the configuration details of Hadoop cluster nodes. A snapshot of Hadoop cluster can also be seen in Figure 5.2.

Table 5.4: Cluster Configuration Details

Node Type	Host Name	Host IP	RAM	OS	Pcs	H.Disk
Master	master	10.23.32.234	8GB	Oracle Linux 7.6	2	1 TB
Worker	node101	10.23.32.165	6GB	Oracle Linux 7.6	2	50GB
Worker	node102	10.23.32.166	6GB	Oracle Linux 7.6	2	50GB
Worker	node103	10.23.32.167	6GB	Oracle Linux 7.6	2	50GB
Worker	node104	10.23.32.168	6GB	Oracle Linux 7.6	2	50GB
Worker	node105	10.23.32.169	6GB	Oracle Linux 7.6	2	50GB
Worker	node106	10.23.32.170	6GB	Oracle Linux 7.6	2	50GB
Worker	node106	10.23.32.171	6GB	Oracle Linux 7.6	2	50GB
Worker	node107	10.23.32.172	6GB	Oracle Linux 7.6	2	50GB
Worker	node108	10.23.32.173	6GB	Oracle Linux 7.6	2	50GB
Worker	node201	10.23.32.174	6GB	Oracle Linux 7.6	2	50GB
Worker	node202	10.23.32.175	6GB	Oracle Linux 7.6	2	50GB
Worker	node203	10.23.32.176	6GB	Oracle Linux 7.6	2	50GB
Worker	node204	10.23.32.177	6GB	Oracle Linux 7.6	2	50GB
Worker	node205	10.23.32.178	6GB	Oracle Linux 7.6	2	50GB
Worker	node206	10.23.32.179	6GB	Oracle Linux 7.6	2	50GB
Worker	node207	10.23.32.180	6GB	Oracle Linux 7.6	2	50GB
Worker	node208	10.23.32.181	6GB	Oracle Linux 7.6	2	50GB
Worker	node209	10.23.32.182	6GB	Oracle Linux 7.6	2	50GB
Worker	node210	10.23.32.183	6GB	Oracle Linux 7.6	2	50GB
Worker	node301	10.23.32.184	6GB	Oracle Linux 7.6	2	50GB
Worker	node302	10.23.32.185	6GB	Oracle Linux 7.6	2	50GB
Worker	node303	10.23.32.186	6GB	Oracle Linux 7.6	2	50GB
Worker	node304	10.23.32.187	6GB	Oracle Linux 7.6	2	50GB
Worker	node305	10.23.32.188	6GB	Oracle Linux 7.6	2	50GB
Worker	node306	10.23.32.189	6GB	Oracle Linux 7.6	2	50GB
Worker	node307	10.23.32.190	6GB	Oracle Linux 7.6	2	50GB
Worker	node308	10.23.32.191	6GB	Oracle Linux 7.6	2	50GB
Worker	node309	10.23.32.192	6GB	Oracle Linux 7.6	2	50GB
Worker	node310	10.23.32.193	6GB	Oracle Linux 7.6	2	50GB



Nodes of the cluster

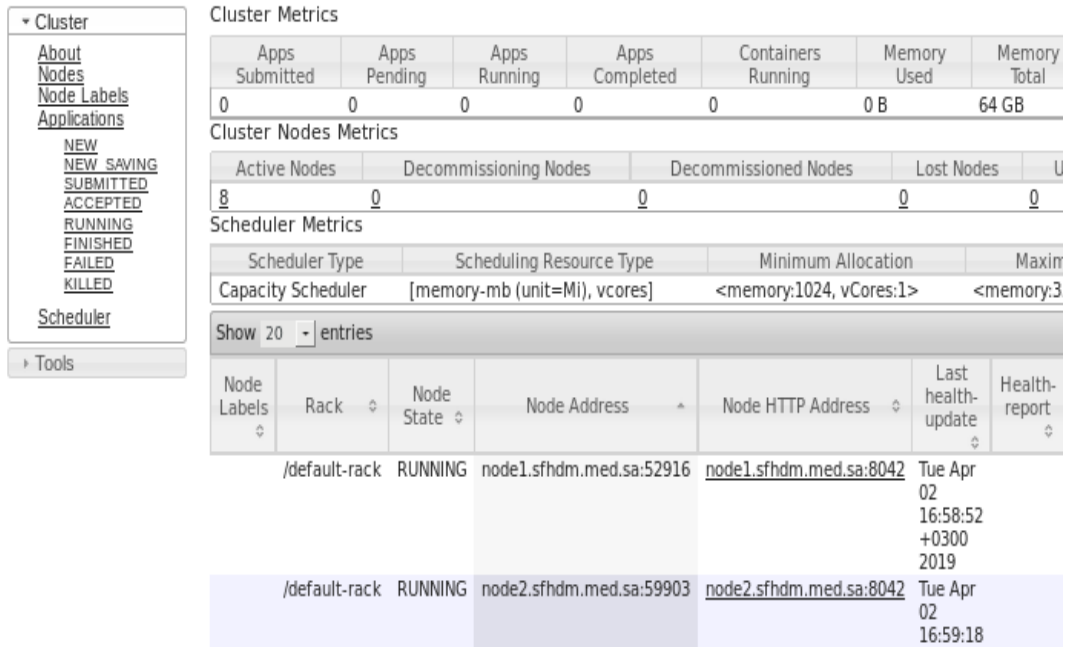


Figure 5.2: Hadoop Cluster Nodes

5.2 Data Sets

Mnist, Higgs, and Molecular are the three datasets used to evaluate our framework. For more details (see Table 5.5 and Figure 5.4).

Table 5.5: Datasets

Data set	Rows	size	Descriptions
Mnist Data	60,000	47.0M	Small number of rows
Higgs Data	10,000,000	280M	Large number of rows
Molecular Data	150,000	430M	Large number of model size

5.2.1 MNIST Dataset

Mixed National Institute of Standards and Technology (MNIST) dataset is a database of handwritten digits used for training image processing systems (see Table 5.3). It has a training set of 60,000 images and a test set of 10,000 images. It is a subset of a larger set available from NIST. The black and white images have been size-normalized to fit into a 28 x 28 box in order to be used for machine learning [104].

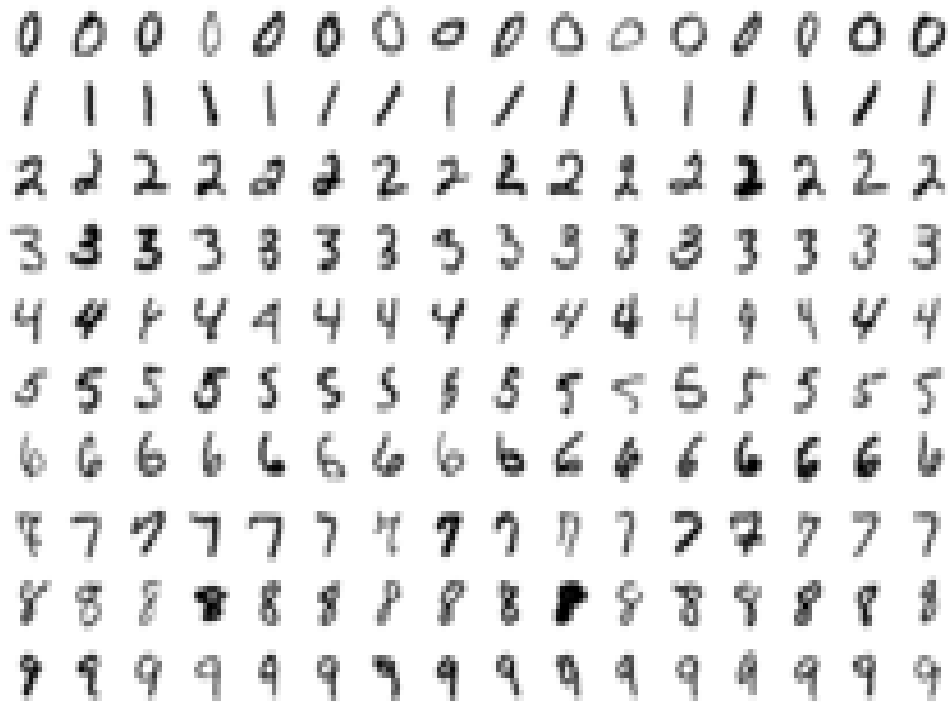


Figure 5.3: Mnist Dataset

5.2.2 Higgs Dataset

Higgs dataset was generated using Monte Carlo simulations. The first 21 features (columns 2-22) are properties measured by the particle detectors in the accelerator. The last seven features are functions of the first 21 features; these are high-level features derived by physicists. [105]. Higgs challenge is a classification problem related to discovering of exotic particle result from collisions at high-energy particle which requires solving complex signal-versus-background classification [106].

5.2.3 Molecular Dataset

Molecular dataset consists of 15 biological activity data sets. Each row of data corresponds to a chemical structure represented by molecular descriptors. The challenge is to predict the activity value for each molecule combination in the test set in order to minimize new medication side effects [107].

Browse Directory

/MyData/MyThesis/Data/bpnn-partitions_results_1_20

Show 25 entries Search:

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size
<input type="checkbox"/>	-rw-r--r--	hd	supergroup	0 B	Mar 12 23:06	28	128 MB
<input type="checkbox"/>	-rw-r--r--	hd	supergroup	2.78 MB	Mar 12 23:06	2	128 MB
<input type="checkbox"/>	-rw-r--r--	hd	supergroup	2.78 MB	Mar 12 23:06	2	128 MB

Figure 5.4: Datasets in HDFS Browser

5.3 Framework Models

To evaluate our framework we build 3 different types of models. These models use different datasets with different sizes. For more details see Table 5.6.

Table 5.6: Models and Datasets

No	Model Name	Data set
1	Mnist Model	Mnist Data
2	Higgs Model	Higgs Data
3	Molecular Model	Molecular Data

Our models are designed using reusable functions which make them more useful

and flexible in such a way that different parameters and arguments values can be used for different testing purposes. For example, the model parameter repartition is used to control the repartitioning of training data after each training epoch. For other model global parameters and their purpose (see Table 5.7).

Table 5.7: Models Global Parameters

Parameter Type	Parameter Value	Purpose
No. of epochs	80	training session epochs
Block size	50	mini-batch size
Learning rate	1e-4	training learning rate
repartition	False/True	repartition of training data after each epoch

5.4 Experiment Results

Our experiments were performed using different cluster and model sizes. In this section, we are going to show the results obtained by our framework.

5.4.1 Mnist Model

The structure of the BPNN Mnist model has an input layer, 2 hidden layers, and an output layer of sizes [784, 1024, 1024, 10]. The Mnist dataset which has 60,000 rows of size 47.0 MB is used to evaluate our framework. Our framework accuracy and execution time is evaluated using different sizes of data parallelism which vary between 4, 8, 20, and 40. Table 5.8 describes the important parameters for this model.

Table 5.8: Mnist Model Parameters

Parameter Type	Parameter Value
Model Size	[784, 1024, 1024, 10]
No of Layers	4
No. of Hidden Layers	2
Input Layer size	784
Output Layer size	10
Total Parameter size	1.8M

(A) Mnist Model: 4-Partitions Results:

The results of evaluating the performance of our framework using Mnist model with data parallelism of size 4 and repartition option set to True is shown in Table 5.9. For models evaluation, data size, BPNN model execution time, MLR model execution time, framework total execution time, and framework accuracy for each partition using Mnist dataset are computed.

Table 5.9: Mnist Model: 4-Partitions Experiment Results

PID	Data size	BPNN E. time	MLR E. Time	Total E. Time	Accuracy
0	14978	10:55.3528	0.027219	10:55.38	96.00
1	14999	8:52.95056	0.059435	8:53.01	96.07
2	15014	9:39.00305	0.037804	9:39.04083	96.314
3	15009	9:25.31036	0.097922	9:25.40826	95.81

Our framework uses Spark hash partition method which attempts to load an equal amount of data in each partition. For example, partition 0 has data of size 14978 rows, partition 1 has data of size 14999 rows, ... etc (see Figure 5.5).

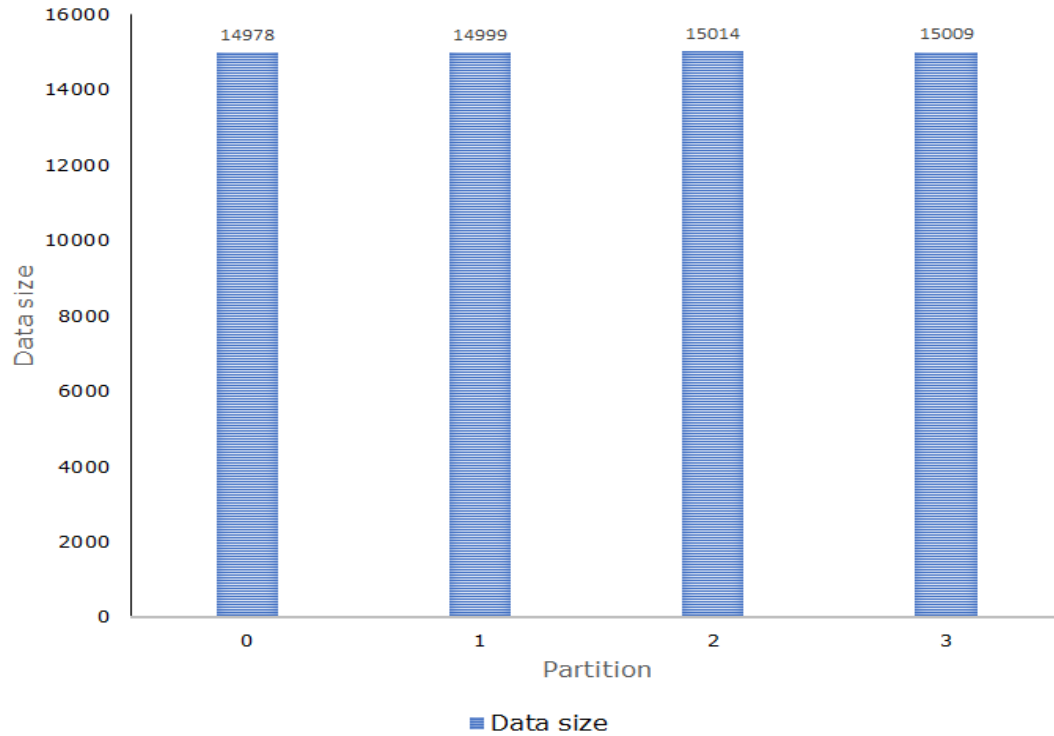


Figure 5.5: Mnist Model: 4-Partitions Data Size (no of rows)

To study the execution time of MLR and BPNN models using Mnist dataset with data parallelism of size 4, the execution time of MLR model is compared to the execution time of BPNN model for each partition using the Mnist dataset. The final result shows that the execution time of MLR model is very low compared to the execution time of BPNN model. For example, For 80 epochs the execution time for MLR model of partition 0 is 0.027219 msec compared to 655353 msec for BPNN model (see Figure 5.6).

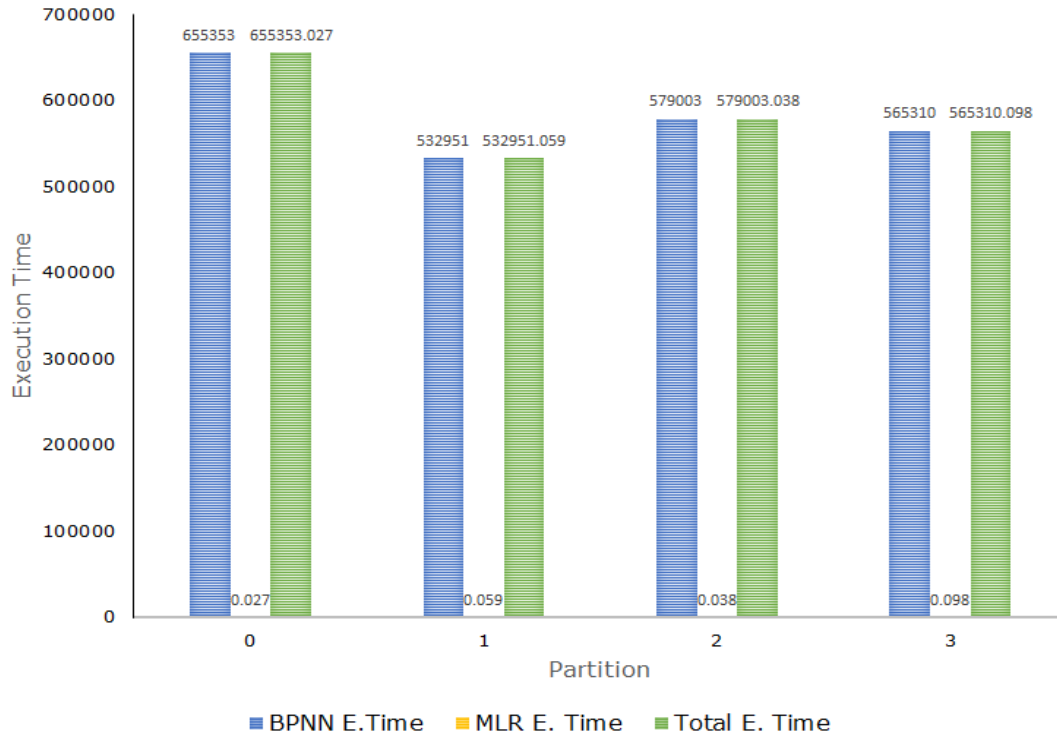


Figure 5.6: Mnist Model: 4-Partitions Elapsed Time (msecs)

To study the accuracy obtained by our framework using Mnist dataset with data parallelism of size 4, the accuracy of BPNN model is compared to the accuracy of the final output of the framework (BPNN + MLR) for each partition using Mnist dataset. The comparison results show that, the employment of the MLR model increase the final accuracy of our framework by 3% - 4% compared to the accuracy obtained by BPNN model only (see Figure 5.7).

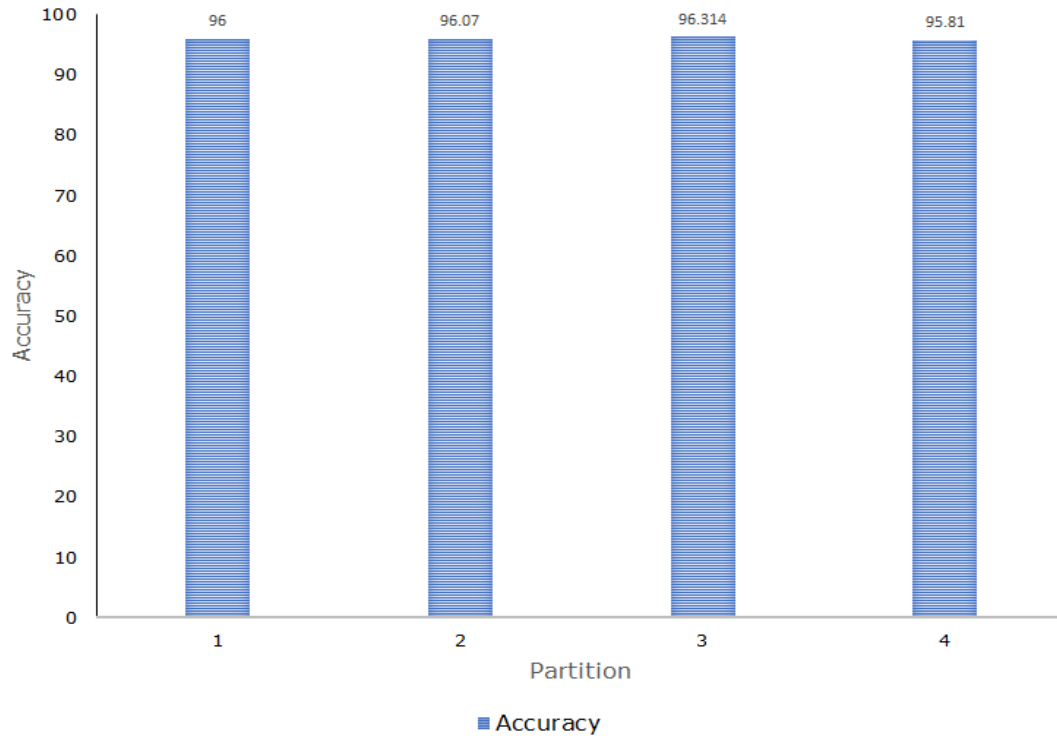



Figure 5.7: Mnist Model: 4-Partitions Accuracy

Yarn resource manager uses 4 different cluster nodes to complete Mnist model training processes. Each node assigns the task of processing BPNN and MLR algorithm on its own data partition, while the optimization process is done using one task running in the master node. Figure 5.8 shows how tasks are assigned to different nodes of the Hadoop cluster.



Application Attempt appattempt_1553906098003_0002

Cluster

- About
- Nodes
- Node Labels
- Applications
- NEW
- NEW SAVING
- SUBMITTED
- ACCEPTED
- RUNNING
- FINISHED
- FAILED
- KILLED

Scheduler

Tools

Application Attempt State: RUNNING

Started: Wed Apr 03 00:01:23 +0300 2019

Elapsed: 1mins, 5sec

AM Container: container_1553906098003_0002_01_000001

Node: 10.23.32.164:0

Tracking URL: ApplicationMaster

Diagnostics Info:

Nodes blacklisted by the application: -

Nodes blacklisted by the system: -

Application Attempt Headroom: <memory:50176, vCores:59>

Total Allocated Containers: 5

Each table cell represents the number of NodeLocal/RackLocal/OffSwitch containers satisfied by NodeLocal/RackLocal/OffSwitch

	Node Local Request	Rack Local Request
Num Node Local Containers (satisfied by)	0	
Num Rack Local Containers (satisfied by)	0	0
Num Off Switch Containers (satisfied by)	0	0

Show 20 entries

Container ID	Node	Container Exit St
container_1553906098003_0002_01_000005	http://node6.sfhdm.med.sa:8042	0
container_1553906098003_0002_01_000004	http://node4.sfhdm.med.sa:8042	0
container_1553906098003_0002_01_000003	http://node3.sfhdm.med.sa:8042	0
container_1553906098003_0002_01_000002	http://node1.sfhdm.med.sa:8042	0
container_1553906098003_0002_01_000001	http://node2.sfhdm.med.sa:8042	0

Showing 1 to 5 of 5 entries

Figure 5.8: Yarn: Scheduling tasks to Hadoop workers

(B) Mnist Model: 8-Partitions Results:

To evaluate our framework models, we compute the data size, BPNN model execution time, MLR model execution time, framework total execution time, and framework accuracy using Mnist dataset. The results of evaluating the performance of our framework using Mnist model with data parallelism of size 8 and repartition option set to True are shown in Table 5.10.

Table 5.10: Mnist Model: 8-Partitions Experiment Results

PID	Data size	BPNN E. time	MLR E. Time	Total E. Time	Accuracy
0	7500	5:44.9766	0.023278	5:44.99994	94.03
1	7478	6:50.383	0.024409	6:50.4074	94.89
2	7506	6:48.79846	0.029138	6:48.82764	94.50
3	7493	6:47.96765	0.024418	6:47.99207	94.55
4	7505	4:25.13953	0.057339	4:25.19687	94.10
5	7509	5:52.2153	0.022014	5:52.23724	94.62
6	7504	4:22.48776	0.070625	4:22.55838	93.98
7	7505	6:32.943	0.063437	6:33.0064	93.71

Our framework partitioning process using Mnist dataset with parallelism of size 8, generates partitions of almost equal sizes. That happens because it uses Spark hash partition method which attempts to load an equal amount of data in each partition. For example, partition 0 has a data size of 7500 rows, partition 1 has a data size of 7478 rows, ... etc (see Figure 5.9).

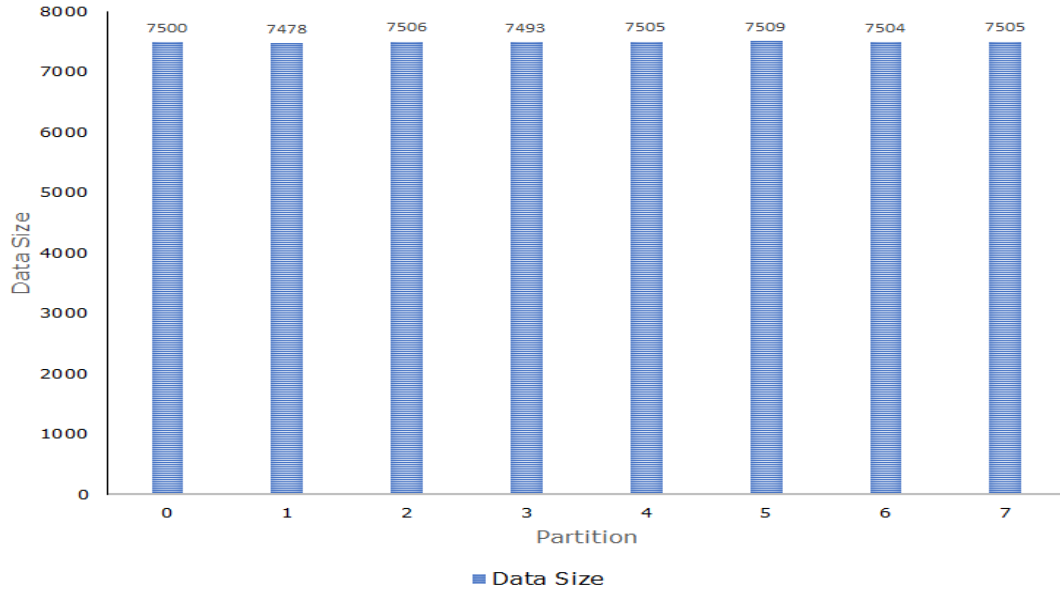


Figure 5.9: Mnist Model: 8-Partitions Data Size (no of rows)

To study the execution time of MLR and BPNN models using Mnist dataset with data parallelism of size 8, the execution time of MLR model is compared to the execution time of BPNN model for each partition using Mnist dataset. The final result shows that the execution time of MLR model is very low compared to the execution time of BPNN model. For example, For 80 epochs the execution time for MLR model of partition 0 is 0.023278 msec compared to 344977 msec for BPNN model (see Figure 5.10).

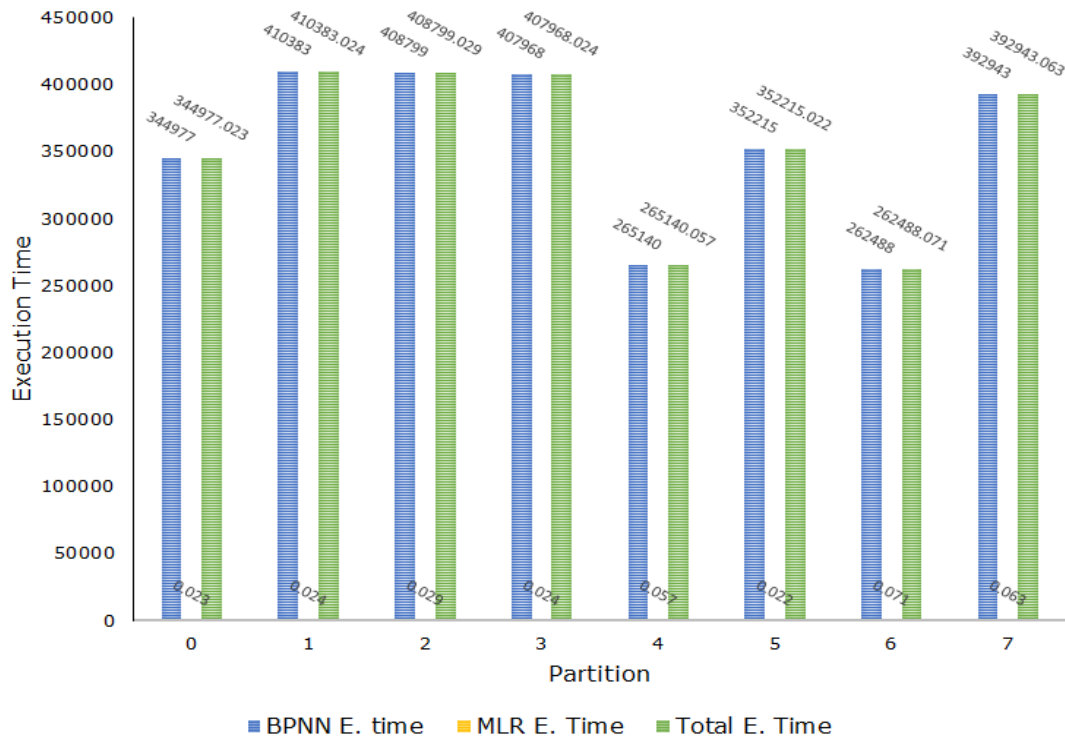


Figure 5.10: Mnist Model: 8-Partitions Elapsed Time (msecs)

To study the accuracy obtained by our framework using Mnist dataset with data parallelism of size 8, the accuracy of BPNN model is compared to the accuracy of the final output of the framework (BPNN + MLR) for each partition using Mnist dataset. The comparison results show that, the employment of the

MLR model increase the final accuracy of our framework by 3% - 4% compared to the accuracy obtained by BPNN model only (see Figure 5.11).

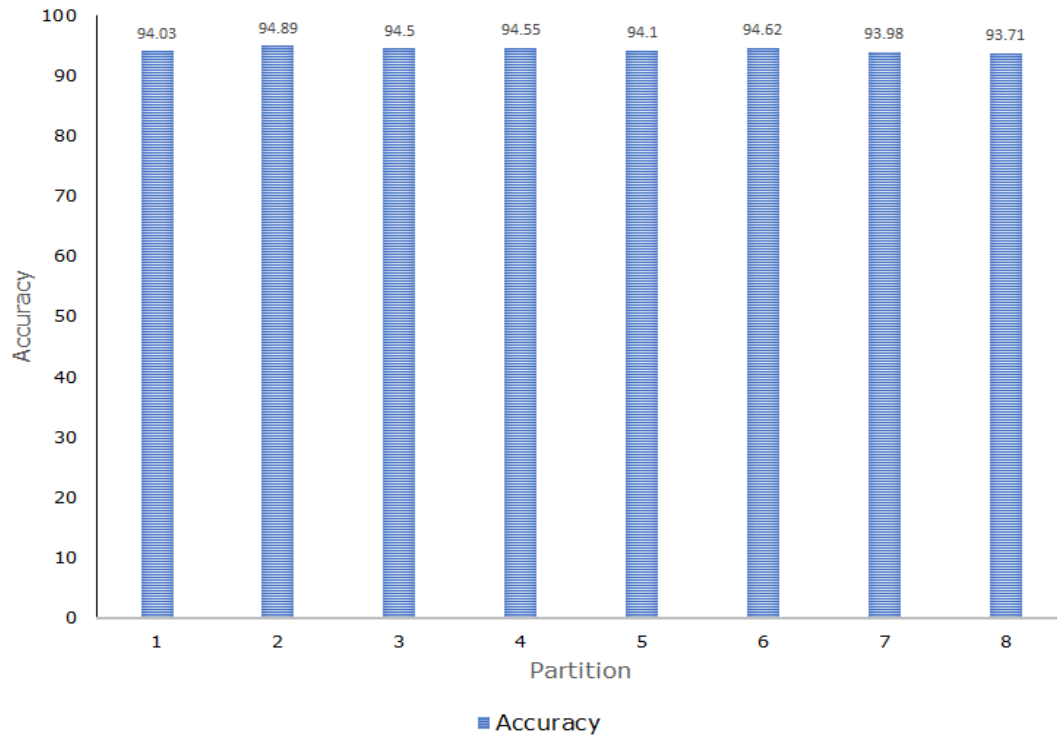


Figure 5.11: Mnist Model: 8-Partitions Accuracy

(C) Mnist Model: 20-Partitions Results:

The results of evaluating the performance of our framework using Mnist model with data parallelism of size 20 and repartition option set to True is shown in Table 5.11. For models evaluation, data size, BPNN model execution time, MLR model execution time, framework total execution time, and framework accuracy for each partition are computed.

Table 5.11: Mnist Model: 20-Partitions Experiment Results

PID	Data size	BPNN E. time	MLR E. Time	Total E. Time	Accuracy
0	3000	449.5605	0.041814	449.60233	90.38
1	2996	504.14792	0.037349	504.18527	90.69
2	3000	510.8083	0.037149	510.84546	90.60
3	2995	526.6572	0.026876	526.6841	90.67
4	2987	501.9434	0.040785	501.9842	92.63
5	3013	514.8201	0.024984	514.84515	91.33
6	2994	501.64133	0.037749	501.67908	90.27
7	2991	531.69415	0.03307	531.7272	90.42
8	3007	453.71613	0.037766	453.7539	90.59
9	2994	516.5516	0.025814	516.5774	90.90
10	3014	328.2883	0.06679	328.3551	91.21
11	3003	454.54425	0.040927	454.58517	90.59
12	2990	448.4523	0.044744	448.49704	90.47
13	3004	526.1988	0.019451	526.21826	90.87
14	3003	502.39572	0.029512	502.42523	91.24
15	3003	401.88535	0.065201	401.95056	90.53
16	2998	523.19684	0.026287	523.22314	90.88
17	3007	513.1345	0.026183	513.16064	90.53
18	2999	517.6213	0.024409	517.6457	90.70
19	3002	311.3553	0.085796	311.44107	89.16

Our framework uses Spark hash partition method which attempts to load an equal amount of data in each partition. For example, partition 0 has data of size 3000 rows, partition 1 has a data size of 2996 rows, ... etc (see Figure 5.12).

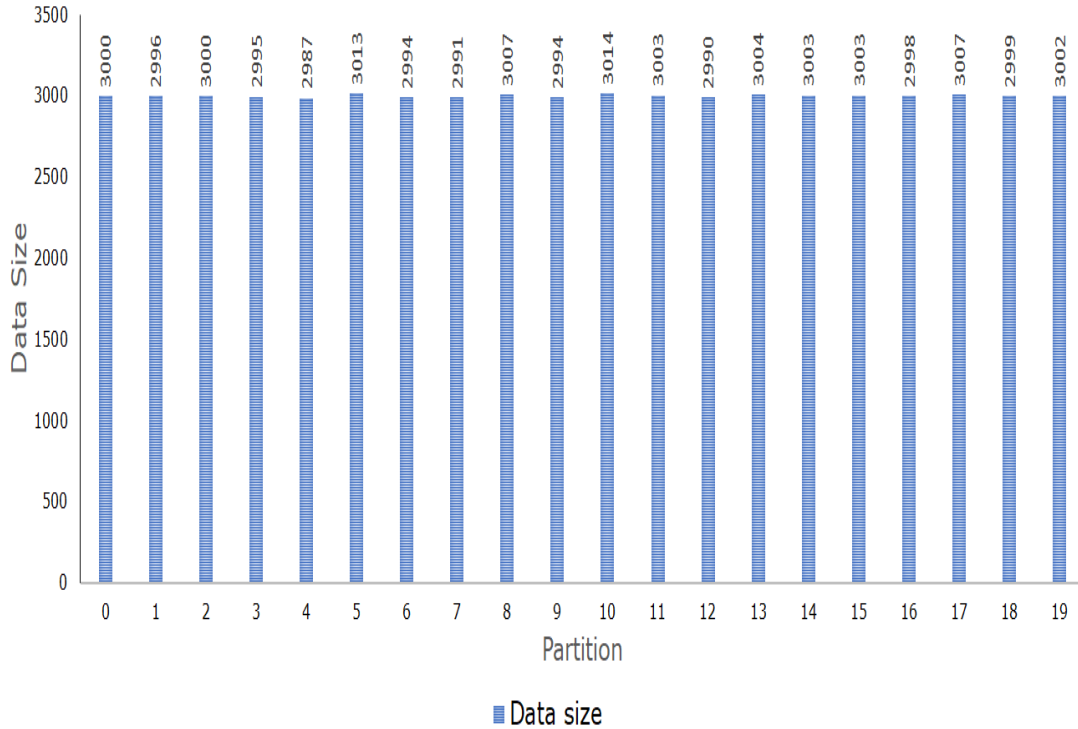


Figure 5.12: Mnist Model: 20-Partitions Data Size (no of rows)

To study the execution time of MLR and BPNN models using Mnist dataset with data parallelism of size 20, the execution time of MLR model is compared to the execution time of BPNN model for each partition using the Mnist dataset. The final result shows that the execution time of MLR model is very low compared to the execution time of BPNN model. For example, For 80 epochs the execution time for MLR model of partition 0 is 0.037066 msec compared to 278401 msec for BPNN model. For more details see Figure 5.13.

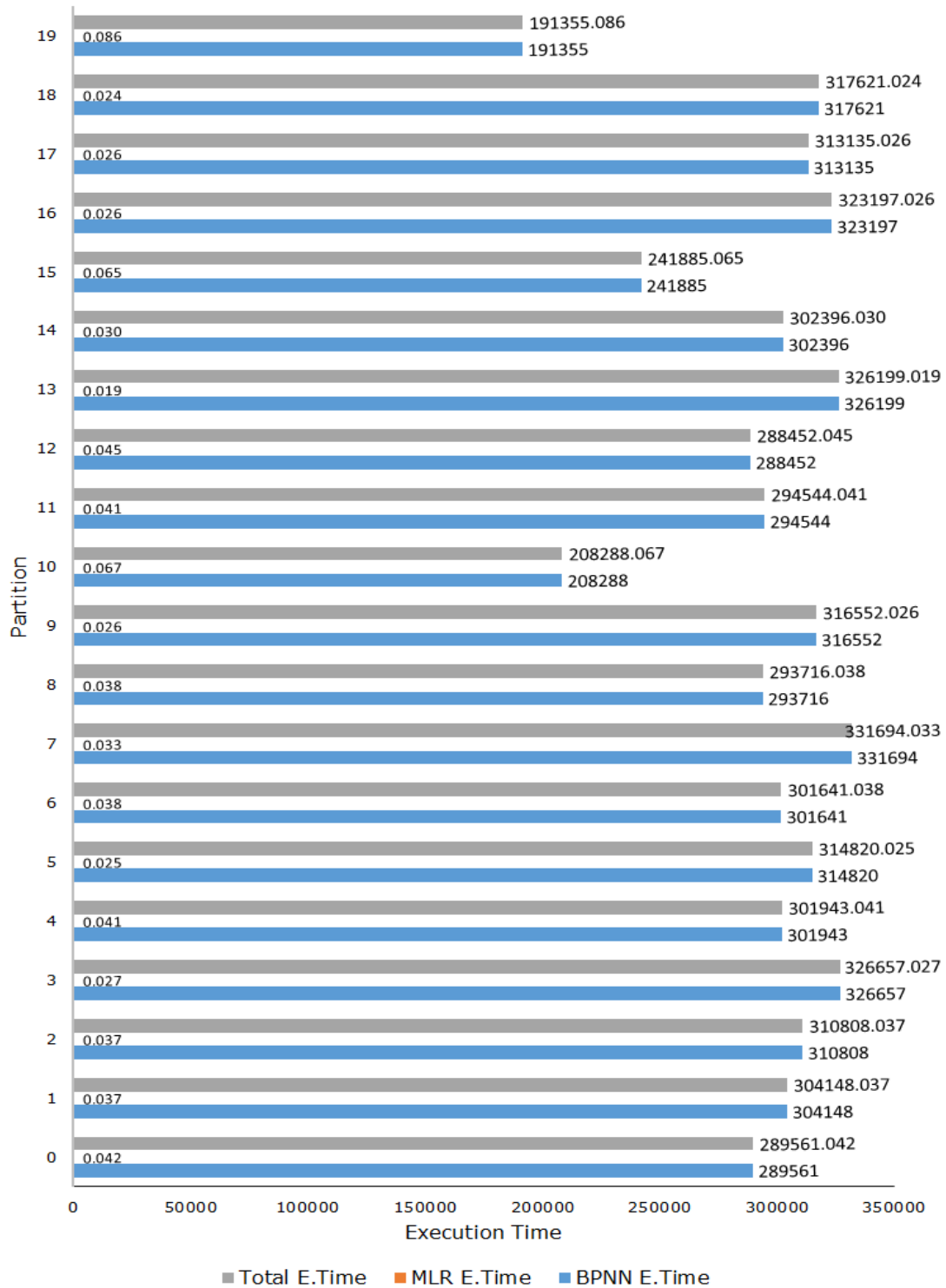


Figure 5.13: Mnist Model: 20-Partitions Elapsed Time (msecs)

To study the accuracy obtained by our framework using Mnist dataset with data parallelism of size 20, the accuracy of BPNN model is compared to the accuracy of the final output of the framework (BPNN + MLR) for each partition using the Mnist dataset. The comparison results show that, the employment of the MLR model increase the final accuracy of our framework by 2% - 3% compared to the accuracy obtained by BPNN model only (see Figure 5.14).

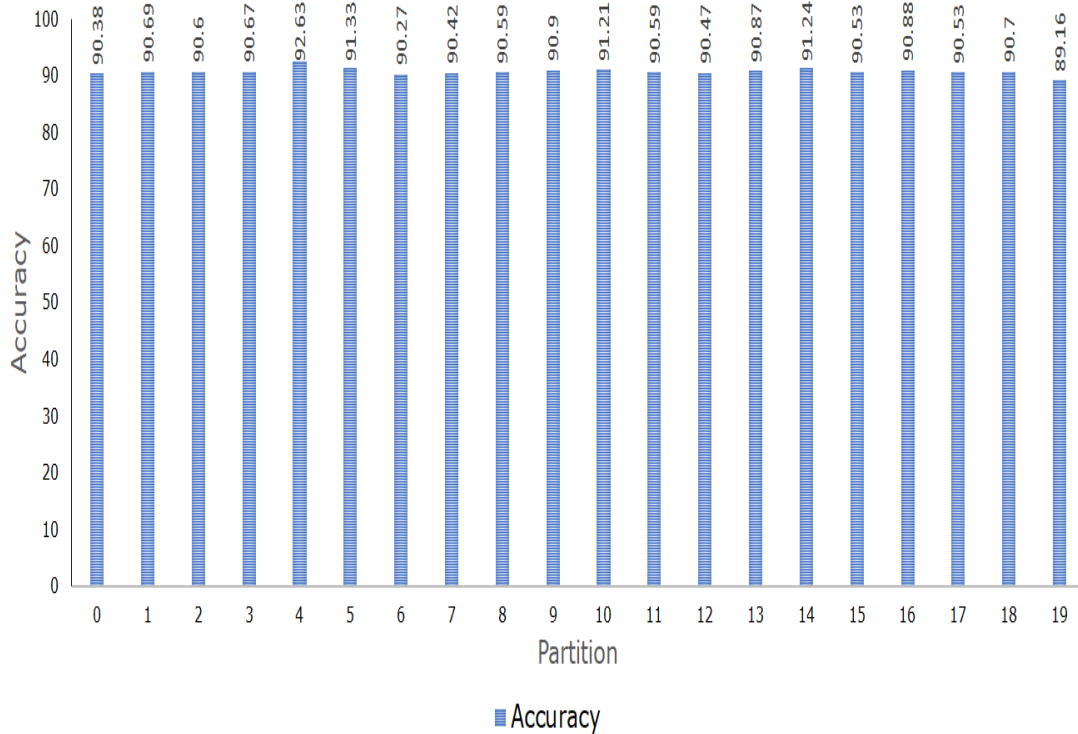


Figure 5.14: Mnist Model: 20-Partitions Accuracy

(D) Mnist Model: 40-Partitions Results:

To evaluate our framework models, we compute the data size, BPNN model execution time, MLR model execution time, framework total execution time, and framework accuracy using Mnist dataset. The results of evaluating the performance of our framework using Mnist model with data parallelism of size 40 and

repartition option set to True are shown in Table 5.13.

Table 5.12: Mnist Model: 8-Partitions Experiment Results - 1

PID	Data size	BPNN E. time	MLR E. Time	Total E. Time	Accuracy
0	1500	438.40128	0.037066	438.43832	87.45
1	1500	447.55243	0.030421	447.58286	88.20
2	1505	436.09845	0.067311	436.16574	87.08
3	1491	450.03778	0.024003	450.06177	87.96
4	1504	420.82083	0.185372	421.0062	87.95
5	1496	335.67252	0.08609	335.7586	87.97
6	1495	452.73676	0.026059	452.7628	88.57
7	1500	439.2488	0.035932	439.28476	87.82
8	1489	447.45917	0.033579	447.49274	88.85
9	1498	417.4352	0.058094	417.49332	89.23
10	1506	409.705	0.056274	409.76126	87.87
11	1507	410.8439	0.101399	410.9453	89.55
12	1502	251.37236	0.11748	251.48984	88.40
13	1492	406.21255	0.100754	406.3133	87.17
14	1499	406.92175	0.105456	407.02722	88.39
15	1492	409.8996	0.155266	410.05484	88.64
16	1499	441.37222	0.031766	441.404	87.99
17	1508	458.81604	0.039578	458.85562	86.06
18	1494	408.67935	0.101665	408.781	87.45
19	1500	433.35034	0.246572	433.59692	88.31
20	1510	437.19513	0.03474	437.2299	88.60
21	1504	257.54373	0.10122	257.64496	88.15
22	1502	436.98187	0.097225	437.0791	87.84
23	1501	355.05017	0.060227	355.1104	87.66
24	1488	421.89145	0.127192	422.01865	87.44

Table 5.13: Mnist Model: 40-Partitions Experiment Results - 2

PID	Data size	BPNN E. time	LR E. Time	Total E. Time	Accuracy
25	1502	230.3202	0.072398	230.39261	88.73
26	1499	454.54895	0.070694	454.61963	88.62
27	1505	433.78244	0.040688	433.82315	88.17
28	1498	447.09134	0.023049	447.11438	87.5
29	1505	416.13266	0.048166	416.18085	88.58
30	1507	413.69913	0.043712	413.74283	88.20
31	1496	403.07788	0.044641	403.12253	87.66
32	1493	449.4208	0.029532	449.45035	87.69
33	1505	217.83989	0.059265	217.89915	88.81
34	1503	221.04248	0.168456	221.21094	87.70
35	1504	411.69122	0.046329	411.73755	88.27
36	1501	251.72954	0.31963	252.04916	86.79
37	1498	502.87888	0.033761	502.91266	87.45
38	1506	418.8834	0.037127	418.9205	87.81
39	1496	434.25623	0.037624	434.29385	85.13

Our framework partitioning process using Mnist dataset with parallelism of size 40, generates partitions of almost equal sizes. That happens because it uses Spark hash partition method which attempts to load an equal amount of data in each partition. For example, partition 0 has a data size of 1500 rows, partition 1 has a data size of 1500 rows, ... etc (see Figure 5.15).

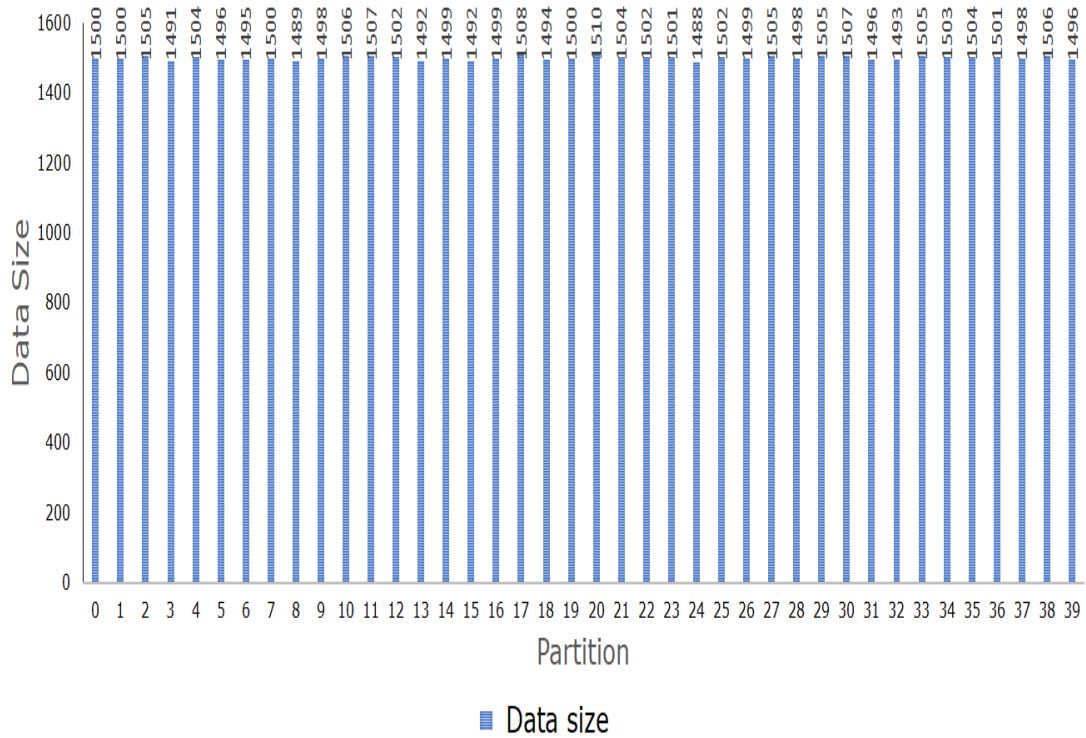
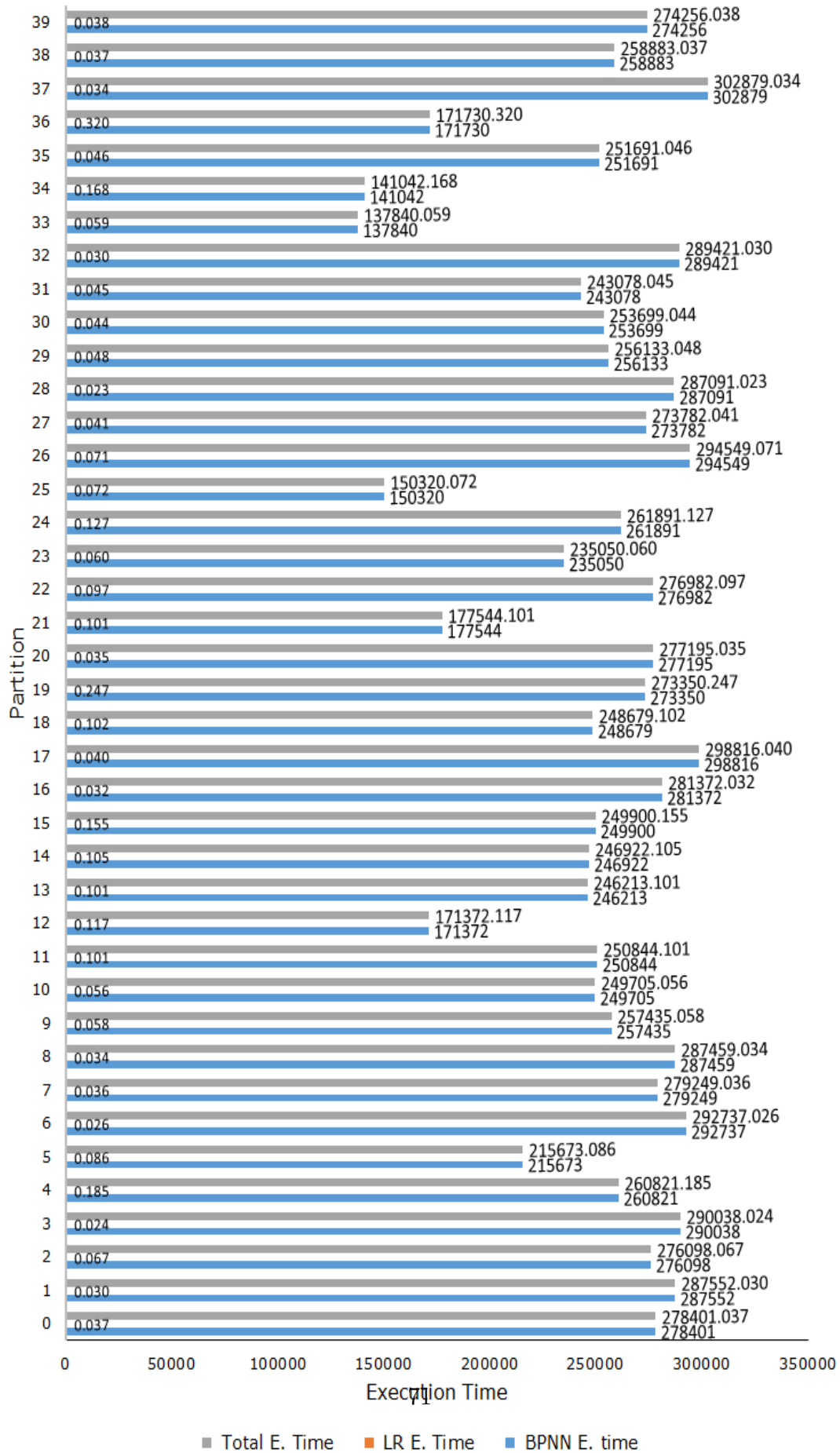


Figure 5.15: Mnist Model: 40-Partitions Data Size (no of rows)

To study the execution time of MLR and BPNN models using Mnist dataset with data parallelism of size 40, the execution time of MLR model is compared to the execution time of BPNN model for each partition using Mnist dataset. The final result shows that the execution time of MLR model is very low compared to the execution time of BPNN model. For example, for 80 epochs the execution time for MLR model of partition 0 is 0.037066 msec compared to 278401 msec for BPNN model. For more details see Figure 5.16.



To study the accuracy obtained by our framework using Mnist dataset with data parallelism of size 40, the accuracy of BPNN model is compared to the accuracy of the final output of the framework (BPNN + MLR) for each partition using Mnist dataset. The comparison results show that, the employment of the MLR model increase the final accuracy of our framework by 2% - 3% compared to the accuracy obtained by BPNN model only (see Figure 5.17).

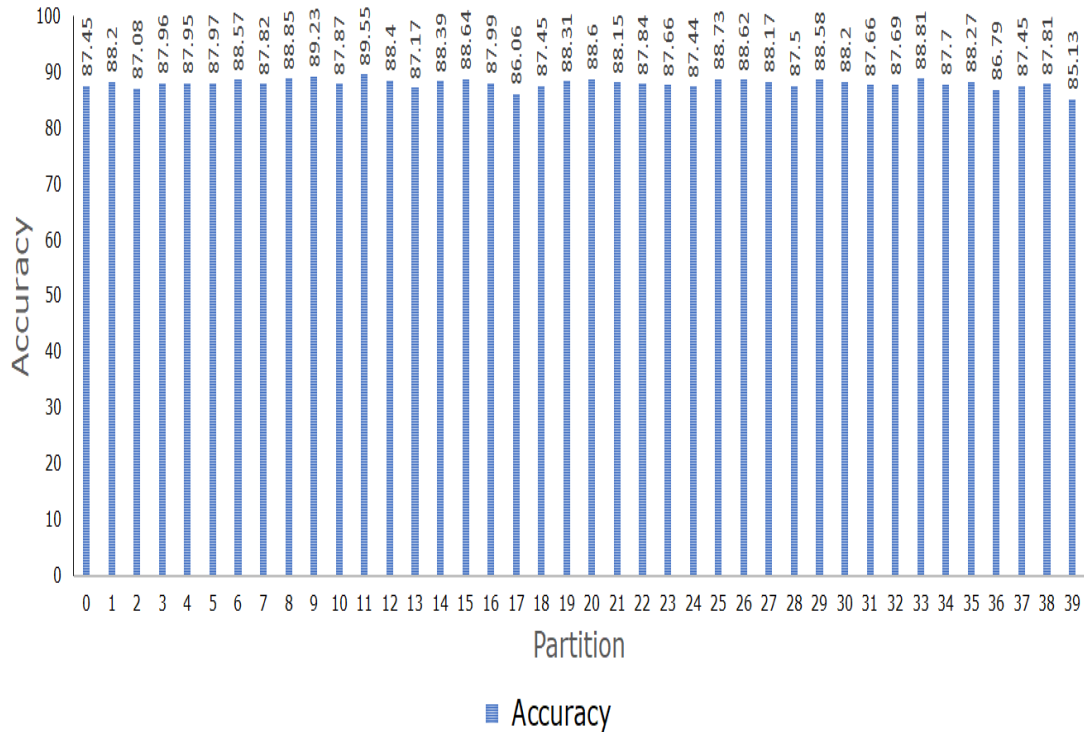


Figure 5.17: Mnist Model: 40-Partitions Accuracy

(E) Mnist Model: Result Analysis:

According to our framework experimental results, Mnist model Scalability is very good. The training execution time and accuracy have been monitor while the cluster is scaling up. For example, for 80 training epochs, the execution time is decreased from 650.21 seconds for 4-partitions to 221.25 seconds for 40-partition.

For more details (see Table 5.14).

Table 5.14: Mnist Model; Scaleup and Speedup

No of Partitions	E. Time (Secs)
4	655.21
8	410.03
20	301.03
40	221.25

Mnist model gains 65.97% speedup as a result of scaling up from 4 to 40 partitions (see Figure 5.18 and Figure 5.19).

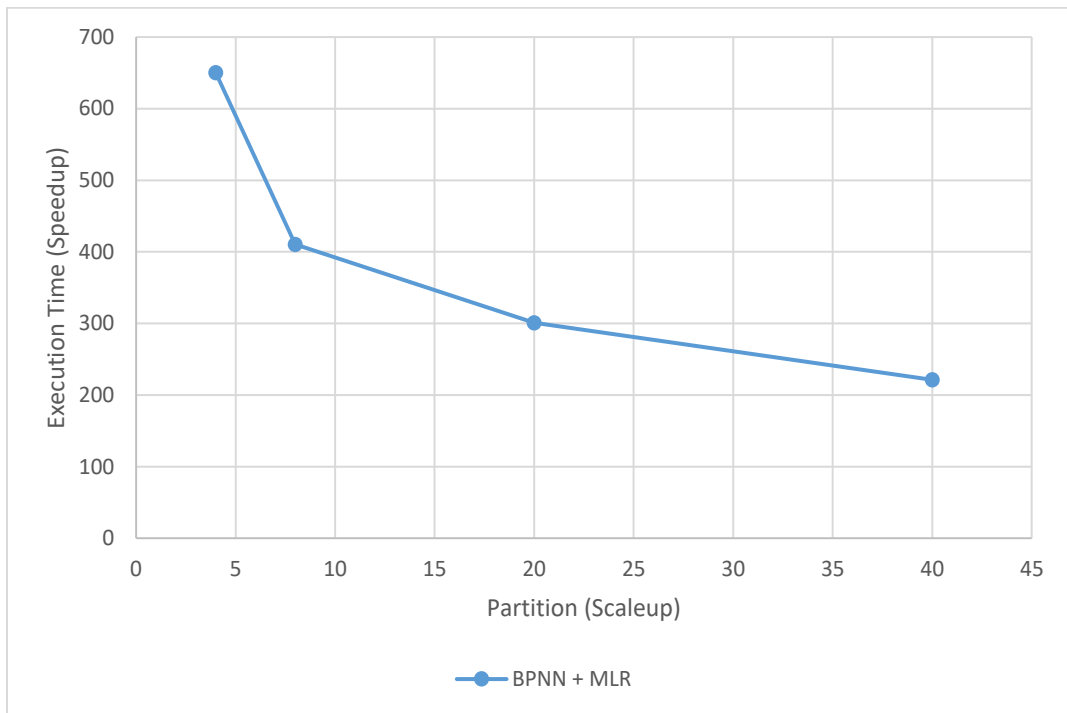


Figure 5.18: Mnist Model: Scaleup vs Speedup (1)

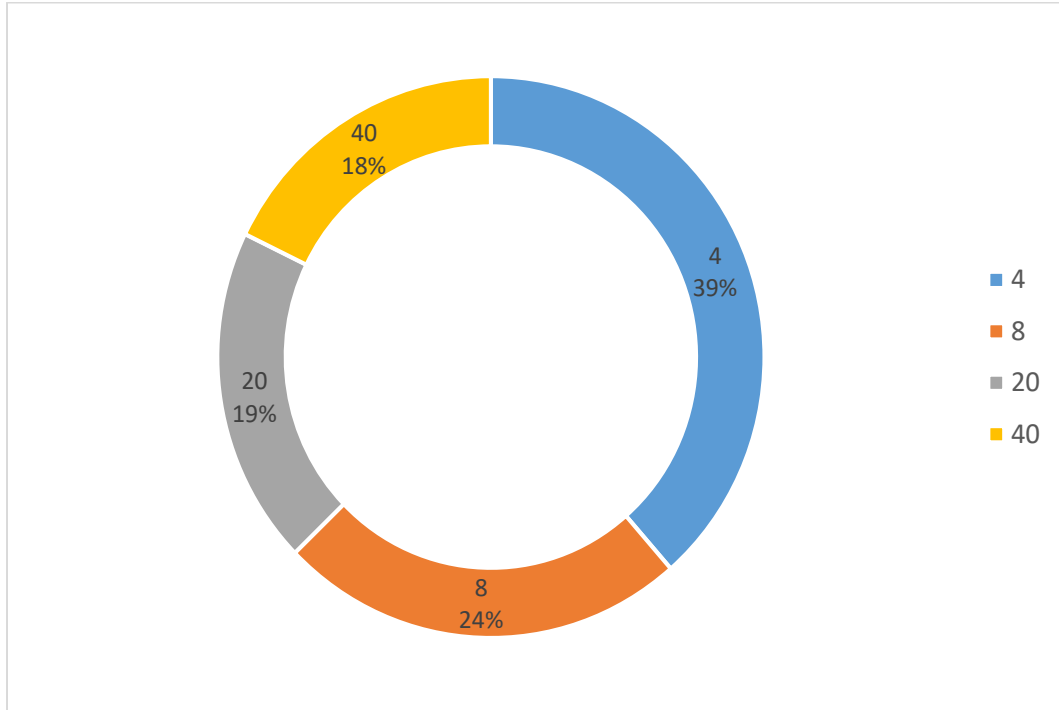


Figure 5.19: Mnist Model: Scaleup vs Speedup (2)

The implementation of the parallel multivariate linear regression algorithm improves our model accuracy by 3% - 4% (see Figure 5.20).

Table 5.15: Mnist Model; Scaleup and Accuracy

No of Partitions	Accuracy
4	97,83
8	96,51
20	94,85
40	92,78

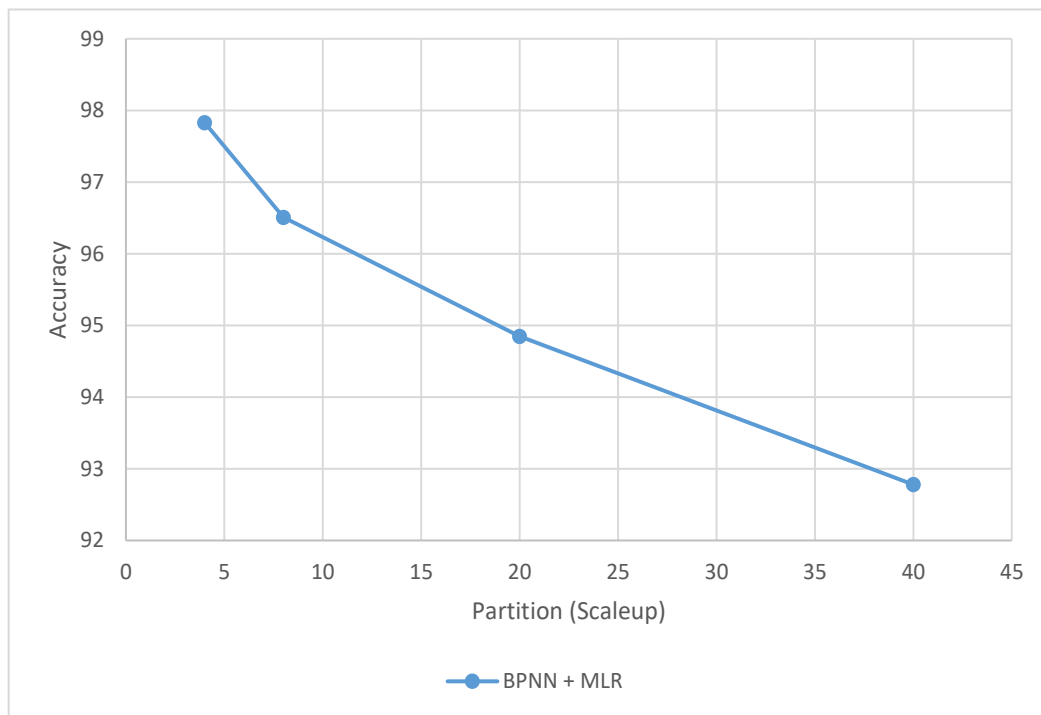


Figure 5.20: Mnist Model: Scaleup and Accuracy

Since the Mnist model is small in size (i.e., 60,000 rows of size 47MB) and data can fit in one machine memory, then using one machine with GPU can give better results than parallel computing.

5.4.2 Higgs Model

The structure of the BPNN Higgs model has an input layer, 3 hidden layers, and an output layer of sizes [28, 1024, 1024, 1024, 1]. The evaluation process uses the Higgs dataset which has 10,000,000 rows of size 280 MB. The framework accuracy and execution time are evaluated using different data parallelism sizes which vary between 4, 8, 20, and 40. Table 5.16 describes the important parameters for this model.

Table 5.16: Higgs Model Parameters

Parameter Type	Parameter Value
Model Size	[28, 1024, 1024, 1024, 1]
No of Layers	5
No. of Hidden Layers	3
Input Layer size	28
Output Layer size	1
Total Parameter size	2.08M

Higgs Model: Result Analysis:

While increasing the data parallelism size (i.e., number of partitions) of our framework, the Higgs model speed increases accordingly. The framework gains 63.70% speedup as a result of scaling up from 4 to 40 partitions (see Table 5.17).

Table 5.17: Higgs Model; Scaleup and Speedup

No of Partitions	E. Time (Secs)
4	15690.22
8	10447.32
20	6434.3
40	5695.34

The data size used to train Higgs model is larger than Mnist data size. Since Spark and Hadoop clusters divide data into chunks of 128 MB, at least 3 partitions are required to store Higgs data. However, too many logical partitions slow down the model and may prevent convergence (see Figure 5.21 and Figure 5.22).

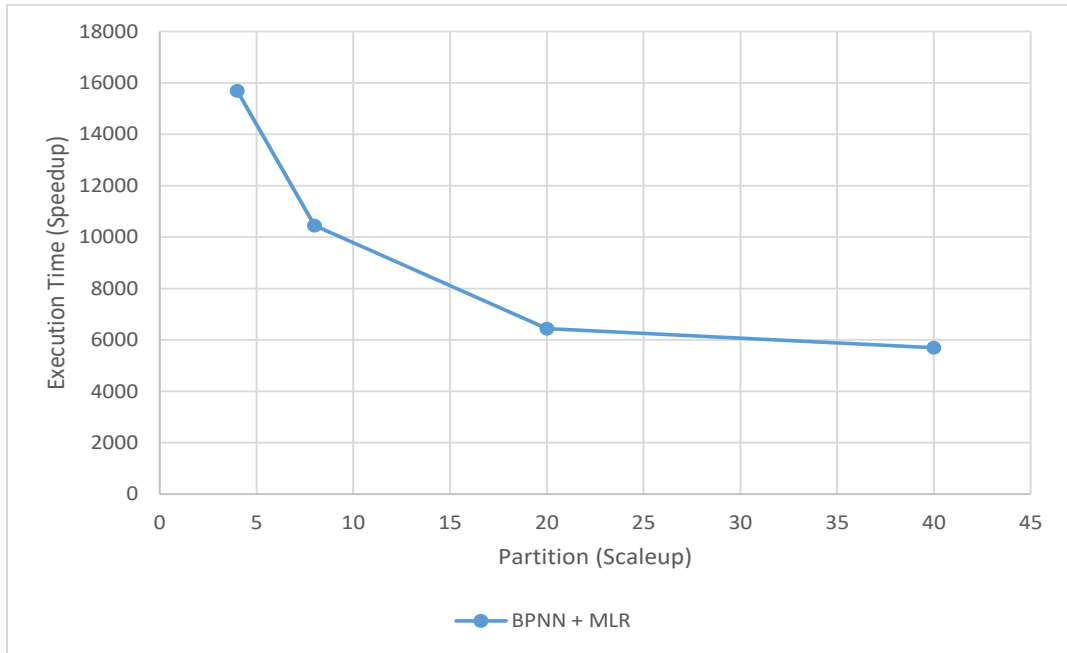


Figure 5.21: Higgs Model: Scaleup vs Speedup (1)

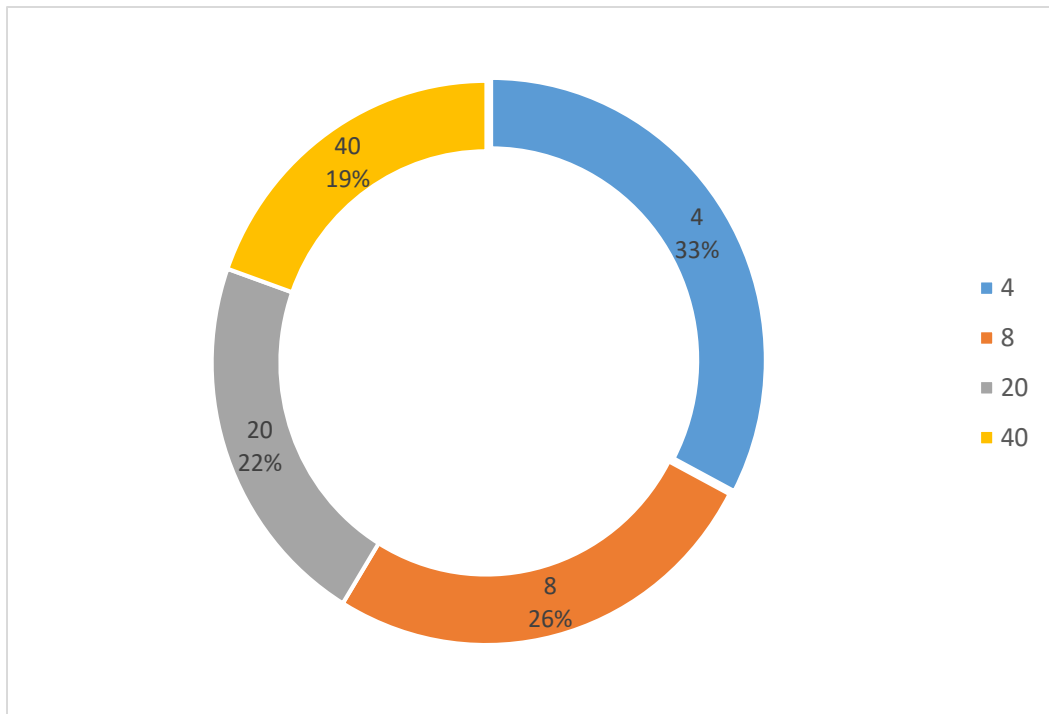


Figure 5.22: Higgs Model: Scaleup vs Speedup (2)

The implementation of the parallel multivariate linear regression algorithm improve our model accuracy by 3% - 5% (see Figure 5.23).

Table 5.18: Higgs Model; Scaleup and Accuracy

No of Partitions	Accuracy
4	92.6
8	91.01
20	88.05
40	85.55

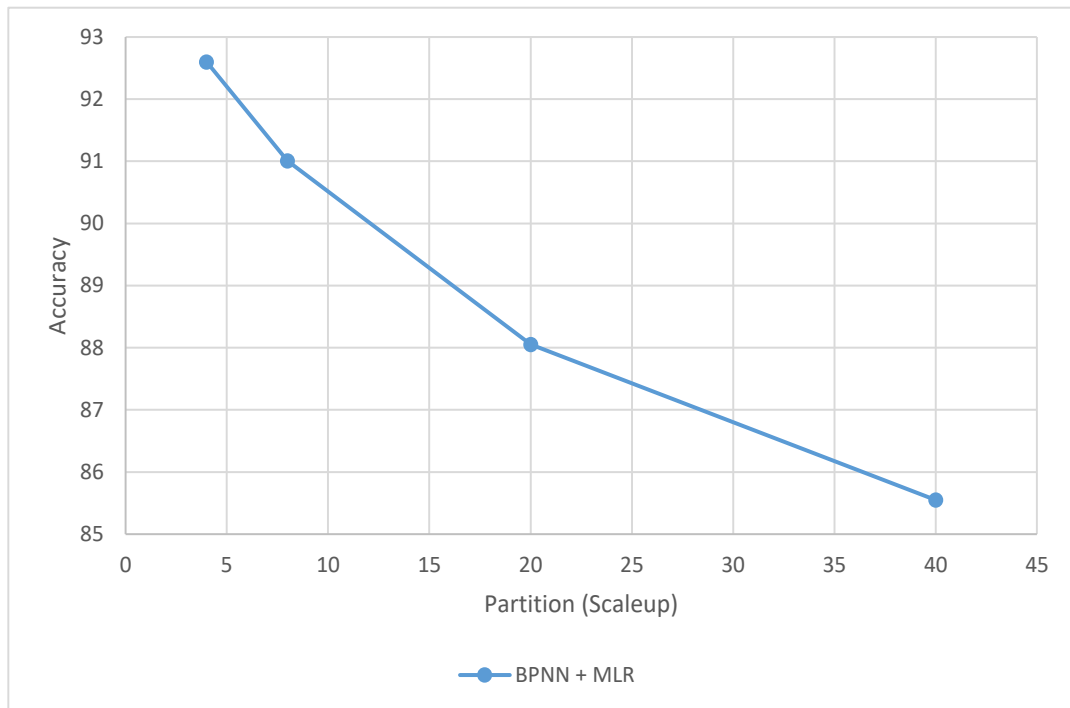


Figure 5.23: Higgs Model: Scaleup vs Accuracy

5.4.3 Molecular Model

The structure of the BPNN Molecular model has an input layer, 3 hidden layers, and an output layer of sizes [2871, 1024, 1024, 1024, 15]. The evaluation process uses the Molecular dataset which has 150,000 rows of size 430MB. The model accuracy and execution time are evaluated using different data parallelism sizes vary between 4, 8, 20, and 40. Table 5.19 describes the important parameters for this model.

Table 5.19: Molecular Model Parameters

Parameter Type	Parameter Value
Model Size	[2871, 1024, 1024, 1024, 15]
No of Layers	5
No. of Hidden Layers	3
Input Layer size	2871
Output Layer size	15
Total Parameter size	5.05M

Molecular Model: Result Analysis:

Molecular model gains 60.29% speedup as a result of scaling up from 4 to 40 partitions. For 80 training epochs, the execution time decreased from 38494.93 seconds for 4-partitions to 15287.74 seconds for 40-partitions. For more details (see Table 5.20).

Table 5.20: Molecular Model; Scaleup and Speedup

No of Partitions	E. Time
4	38494.93
8	24470.11
20	17947.82
40	15287.74

Scaling up with too many partitions reduce data parallelism gains. For example, scaling up with data parallelism size greater than 20 decreases speedup and slow convergence speed (See Figure 5.24 and Figure 5.25).

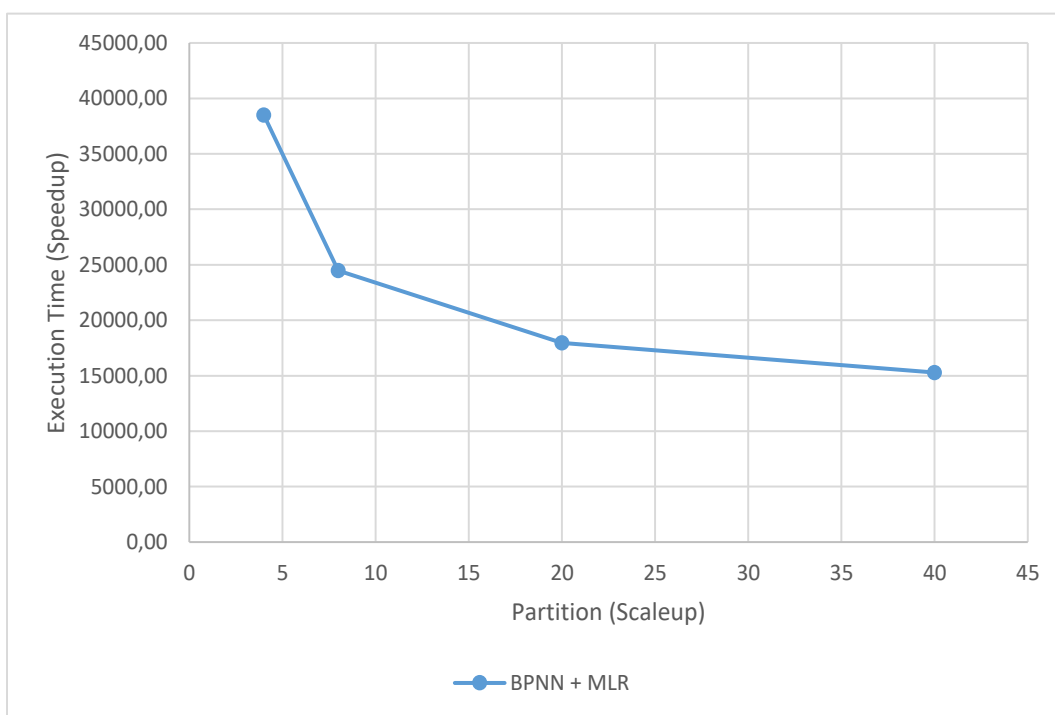


Figure 5.24: Molecular Model: Scaleup vs Speedup (1)

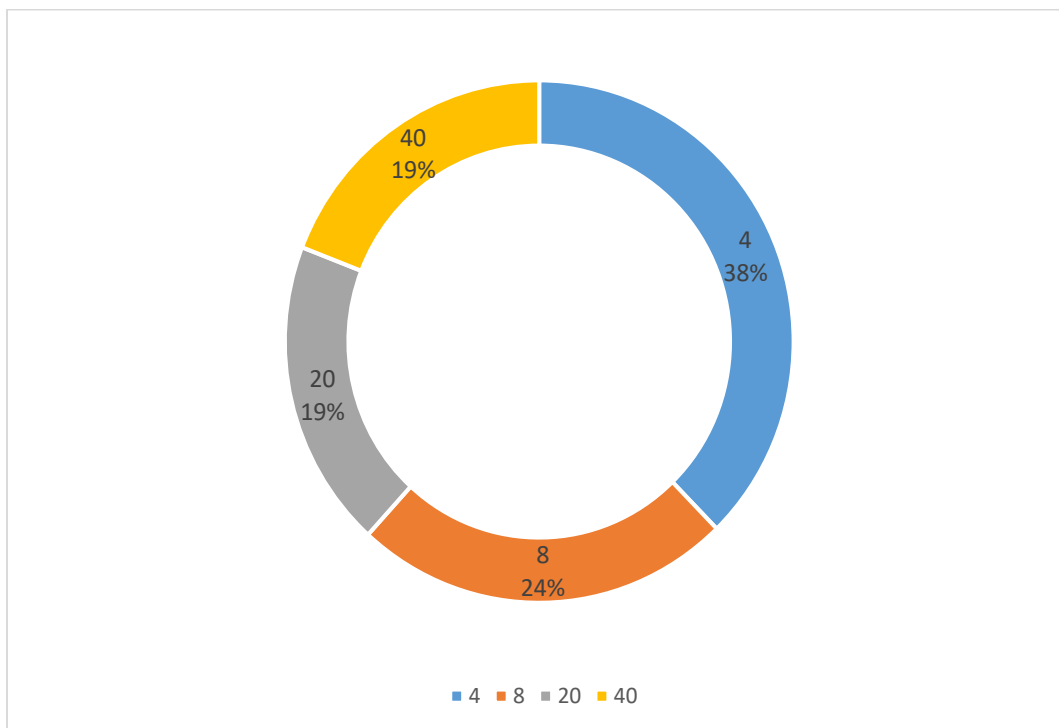


Figure 5.25: Molecular Model: Scaleup vs Speedup (2)

The implementation of the parallel multivariate linear regression algorithm improve our model accuracy by 3% - 4.6% (See Figure 5.26).

Table 5.21: Molecular Model; Scaleup and Accuracy

No of Partitions	Accuracy
4	91.6
8	89.12
20	86.22
40	83.01

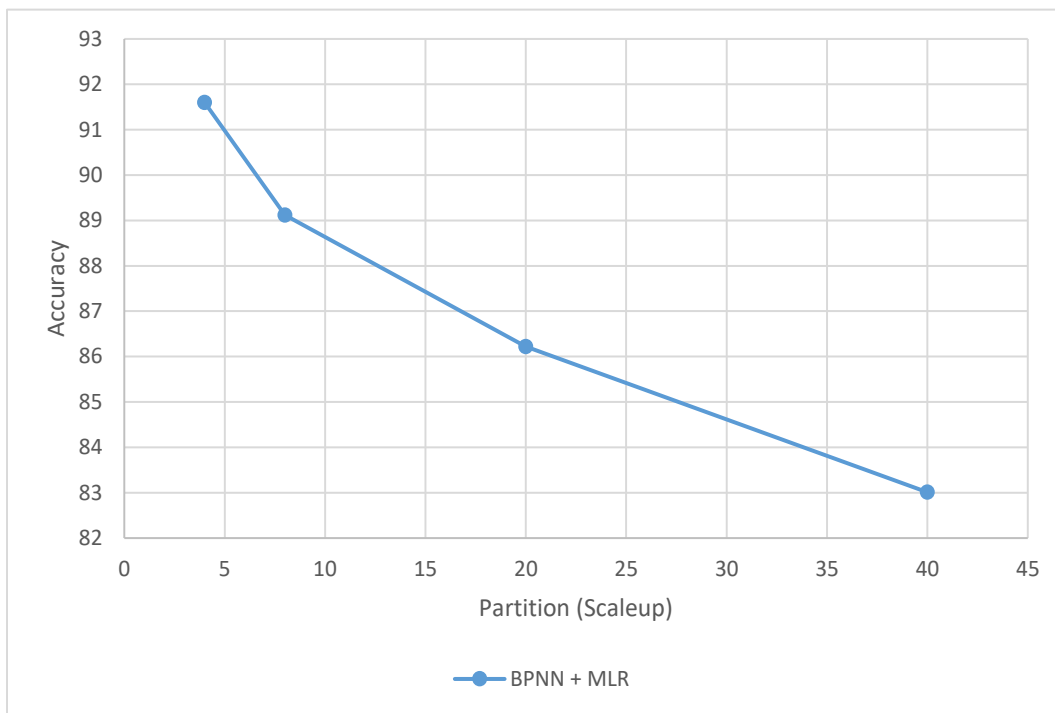


Figure 5.26: Molecular Model: Scaleup and Accuracy

5.5 Framework Evaluation Results Analysis

According to the experimental results our framework scaleup is very good (see Figure 5.27 and Figure 5.28). Three models were built to evaluate the framework including Mnist, Higgs, and Molecular. The speed and accuracy are evaluated in a variety of configurations such as datasets size, data parallelism size, and model size. Result analysis for each model can be found in sections (5.4.1, 5.4.2, 5.4.3).

Our framework evaluation results can be summarized as follows:

- (1) Excluding models with small data size all models can benefit from data parallelism.
- (2) Too many data parallelism sizes can lead to network issues overhead and

slow of convergence speed.

(3) Slow down in training speed as model size increases.

(4) Training performance can scale up with data parallelism size to a factor limited by the network traffic overhead.

(5) The implementation of the parallel multivariate linear regression (MVLR) algorithm improves our framework accuracy.

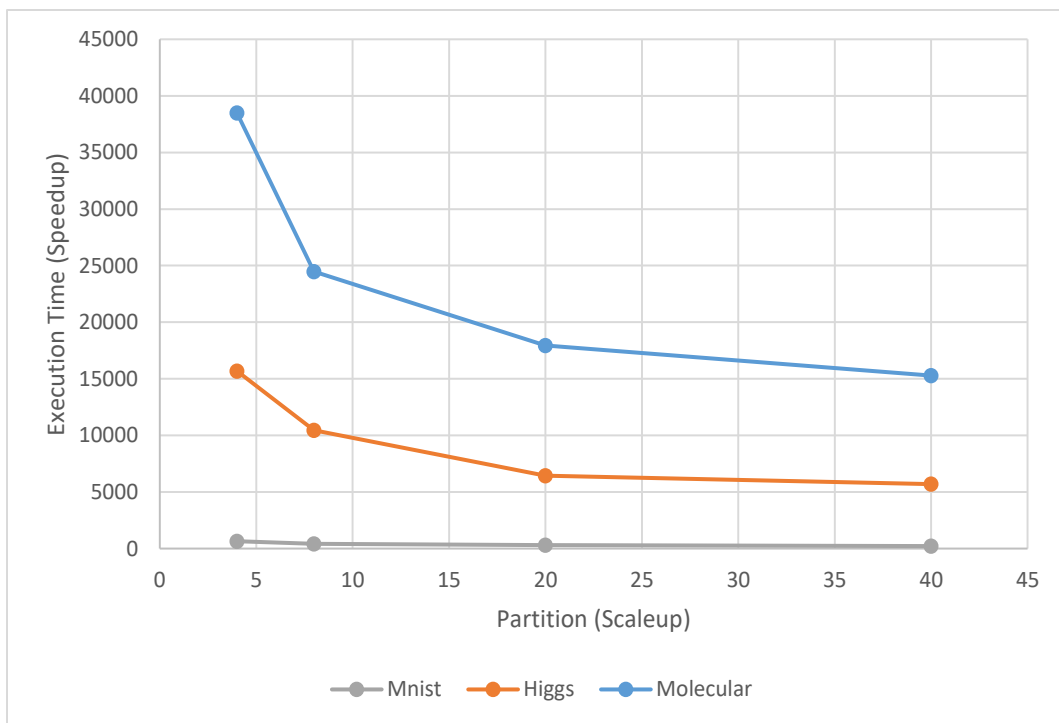


Figure 5.27: Framework: Scaleup vs Speedup

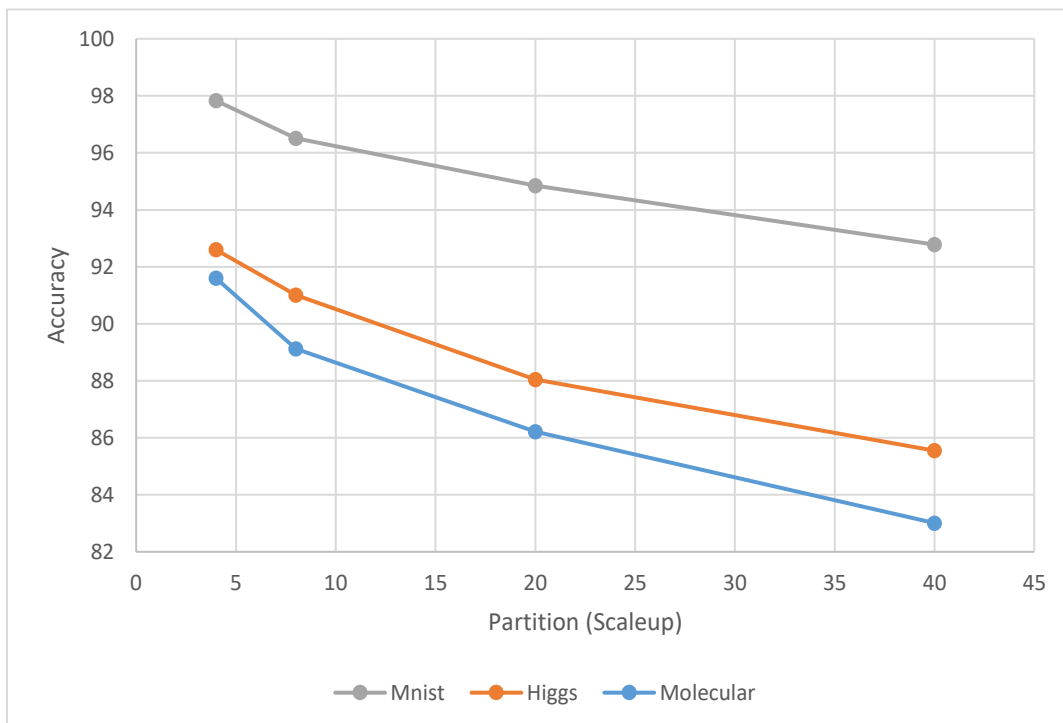


Figure 5.28: Framework: Scaleup vs Accuracy

5.6 Frameworks Comparison

DistBelief is a distributed computation of neural networks proposed by Dean et al. in [17]. The computation takes place on worker nodes. The framework manages communication, synchronization, and data transfer between cluster workers. The architecture of the framework is shown in Figure 5.29.

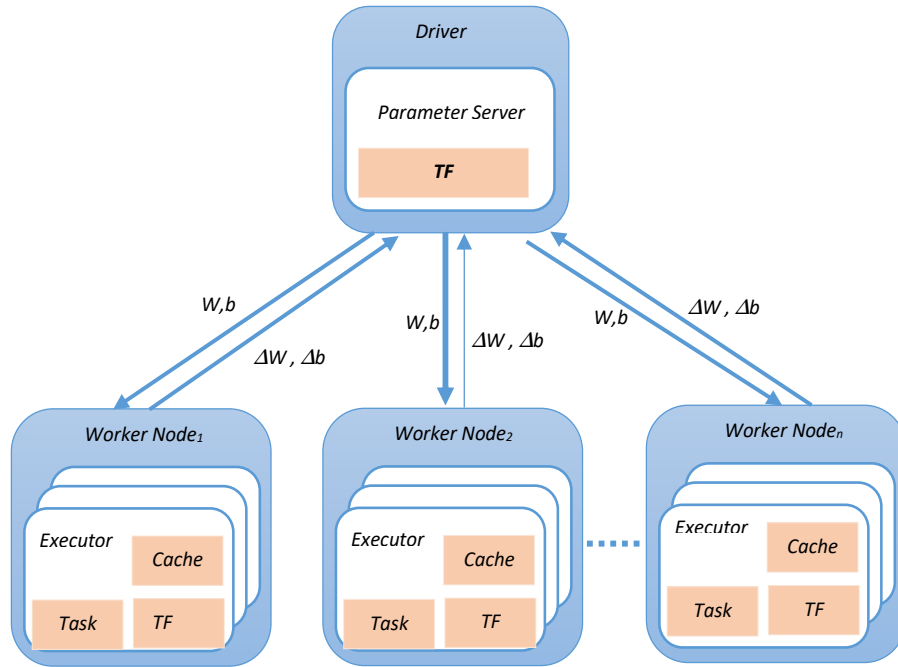


Figure 5.29: DistBelief Framework Architecture

Under the same environment and the same configuration parameters our framework execution time and accuracy are compared to DistBelief framework. The models Mnist, Higgs, and Molecular are developed to evaluate both frameworks. Two types of implementations are performed including session implementation and epoch implementation. For session implementation (BPNN + MLR) the synchronization process is performed at the end of the training session, while the synchronization process is performed at the end of each epoch for the epoch implementation (BPNN + MLR (epoch)). The accuracy and training execution time are evaluated and compared using different data parallelism sizes (i.e., number of partitions) vary between 4, 8, 20, and 40.

5.6.1 Frameworks Comparison: Mnist Model

The experiment results of our framework using Mnist model show that, while scaling up data parallelism size from 4 to 40, our framework gains 65.97% and 60.79% speedup compared to 54.26% gained by DistBelief framework (see Figure 5.30 and Figure 5.31).

Model/Partitions	Execution Time (Sec)			Accuracy		
	BPNN + MLR	BPNN + MLR (Epoch)	DistBelief	BPNN + MLR	BPNN + MLR (Epoch)	DistBelief
4	650,21	740,52	897,01	97,83	98,2	95,9
8	410,3	480,11	695,33	96,51	97,17	94,2
20	301,03	351,21	470,41	94,85	95,51	92,1
40	221,25	290,33	410,25	92,78	93,2	89,6
Speed Gained	65,97%	60,79%	54,26%			

Figure 5.30: Frameworks Comparison: Mnist Model

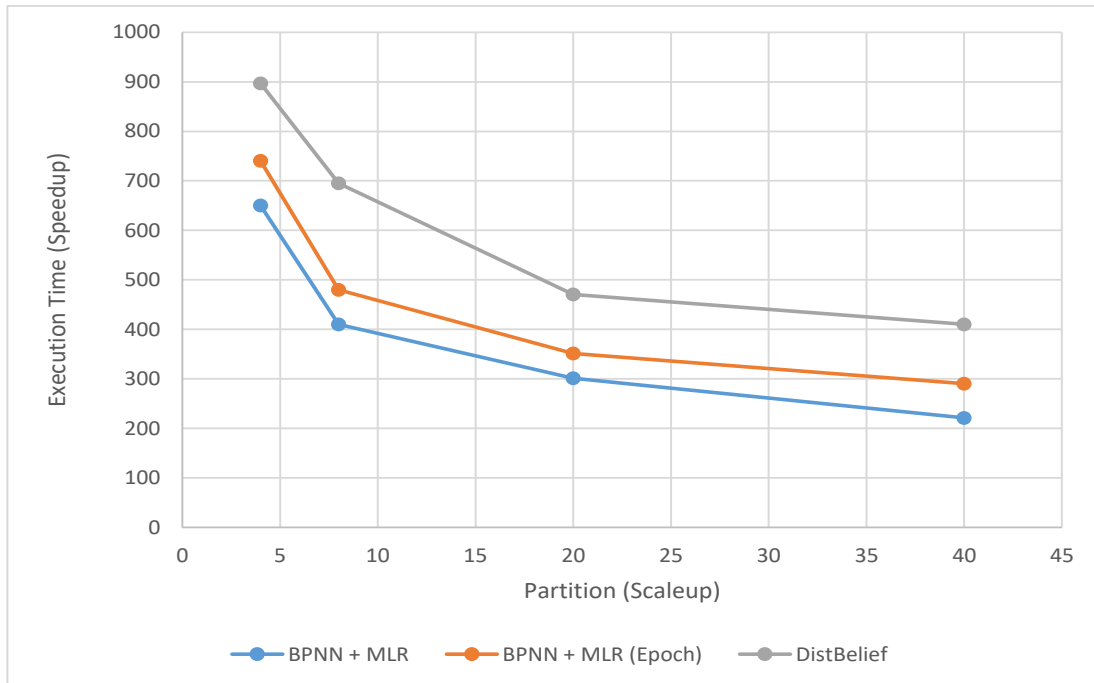


Figure 5.31: Frameworks Comparison: Mnist Model Scaleup vs Speedup

While scaling up the accuracy of our framework and DistBelief framework is decreased. However, the implementation of parallel MLR increases our framework accuracy by 3%-4%. The results are shown in Figure 5.32.

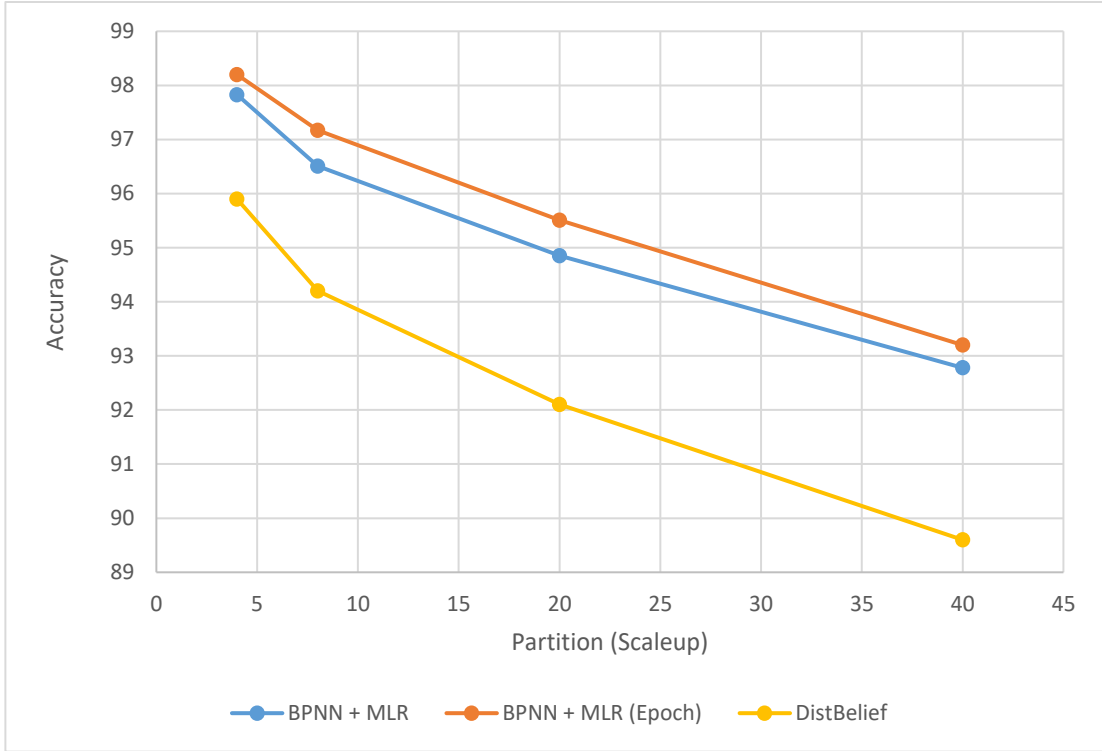


Figure 5.32: Frameworks Comparison: Mnist Model Scaleup vs Accuracy

Since one partition is enough to store Mnist model dataset using one machine with GPU may give better results than parallel computing. This point has been added to our future works due to the lack of GPU environment.

5.6.2 Frameworks Comparison: Higgs Model

The experiment results of our framework using Higgs model show that, while scaling up data parallelism size from 4 to 40, our framework gains 63.70% and

56.74% speedup compared to 51.66% gained by DistBelief framework (see Figure 5.33 and Figure 5.34).

Model/Partitions	Execution Time (Sec)			Accuracy		
	BPNN + MLR	BPNN + MLR (Epoch)	DistBelief	BPNN + MLR	BPNN + MLR (Epoch)	DistBelief
4	15690,22	18199,85	26270,06	92,6	93,9	90,25
8	10447,32	13956,94	19848,78	91,01	92,5	88,51
20	6434,3	8453,55	12727,61	88,05	89,5	86,02
40	5695,34	7873,25	12700,22	85,55	86,81	83,35
Speed Gained	63,70%	56,74%	51,66%			

Figure 5.33: Frameworks Comparison: Higgs Model

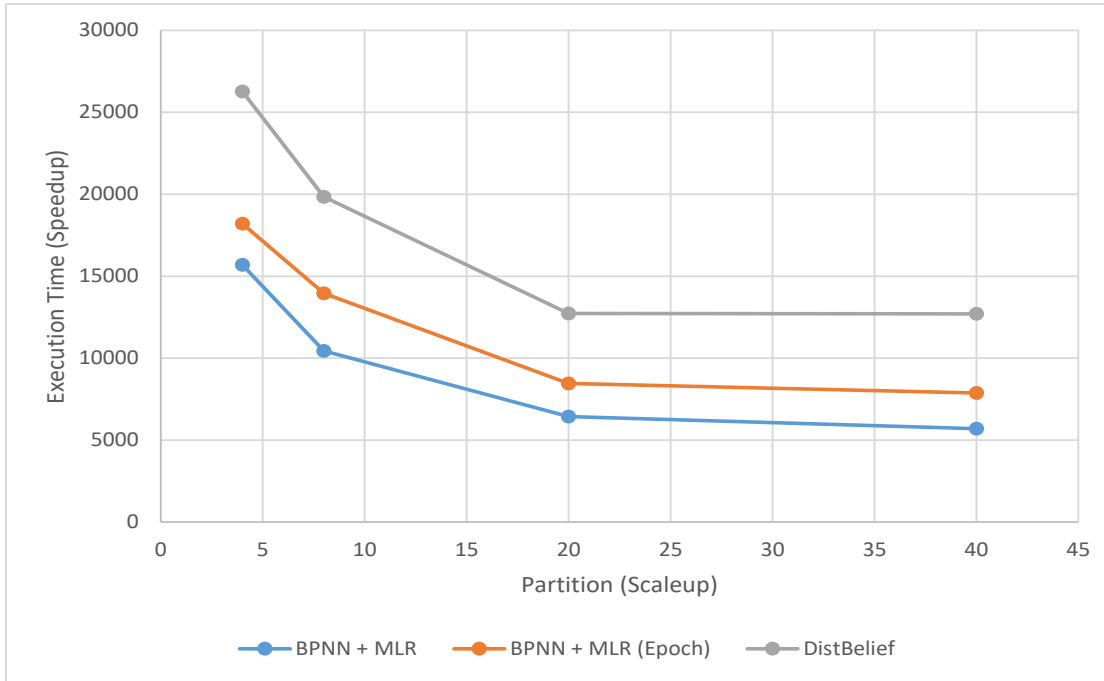


Figure 5.34: Frameworks Comparison: Higgs Model Scaleup vs Speedup

While scaling up the accuracy of our framework and DistBelief framework is decreased. However, the implementation of parallel MLR increases our framework accuracy by 3%-5% (see Figure 5.35).

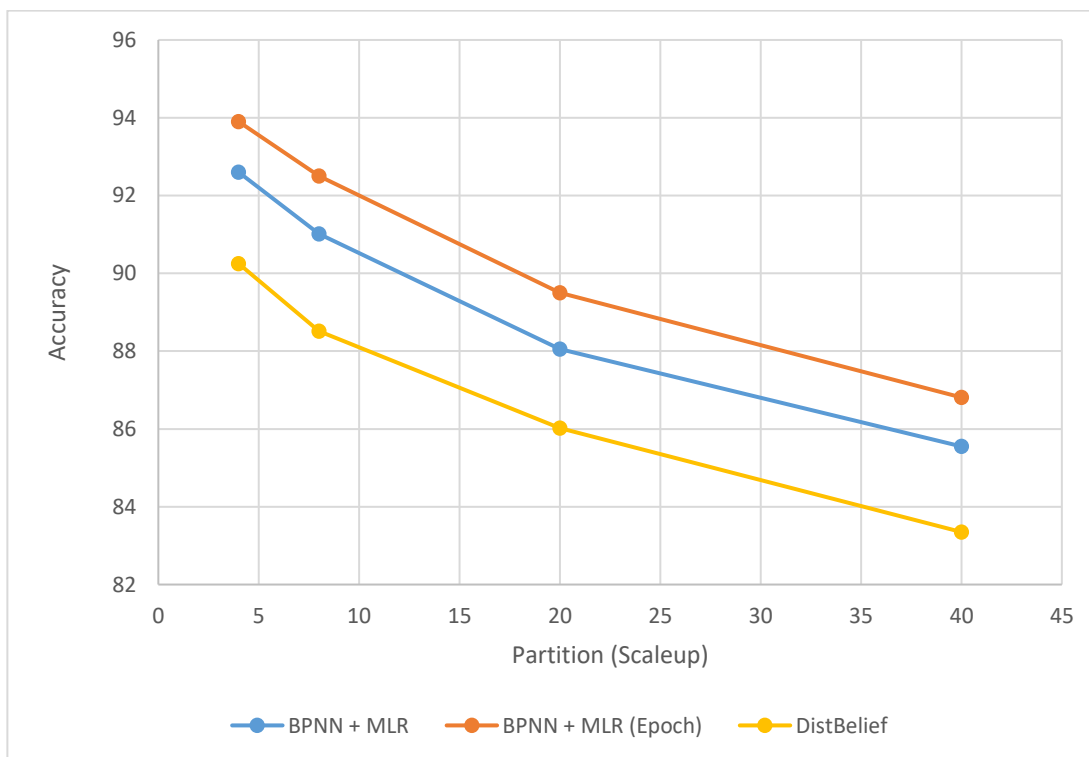


Figure 5.35: Frameworks Comparison: Higgs Model Scaleup vs Accuracy

At least 3 partitions are required to store Higgs dataset. However, too many logical partitions slow convergence speed due to the network traffic overheads since the model parameter size is 2.08MB.

5.6.3 Frameworks Comparison: Molecular Model

The experiment results of our framework using Molecular model show that, while scaling up data parallelism size from 4 to 40, our framework gains 60.29% and 52.91% speedup compared to 48.46% gained by DistBelief framework (see Figure 5.36 and Figure 5.37).

Model/Partitions	Execution Time (Sec)			Accuracy		
	BPNN + MLR	BPNN + MLR (Epoch)	DistBelief	BPNN + MLR	BPNN + MLR (Epoch)	DistBelief
4	38494,93	41639,82	55044,07	91,6	92,21	89,01
8	24470,11	28614,99	38618,54	89,12	90,01	86,92
20	17947,82	21237,59	30099,80	86,22	87,55	83,71
40	15287,74	19607,90	31672,24	83,01	84,02	81,11
Speed Gained	60,29%	52,91%	42,46%			

Figure 5.36: Frameworks Comparison: Molecular Model

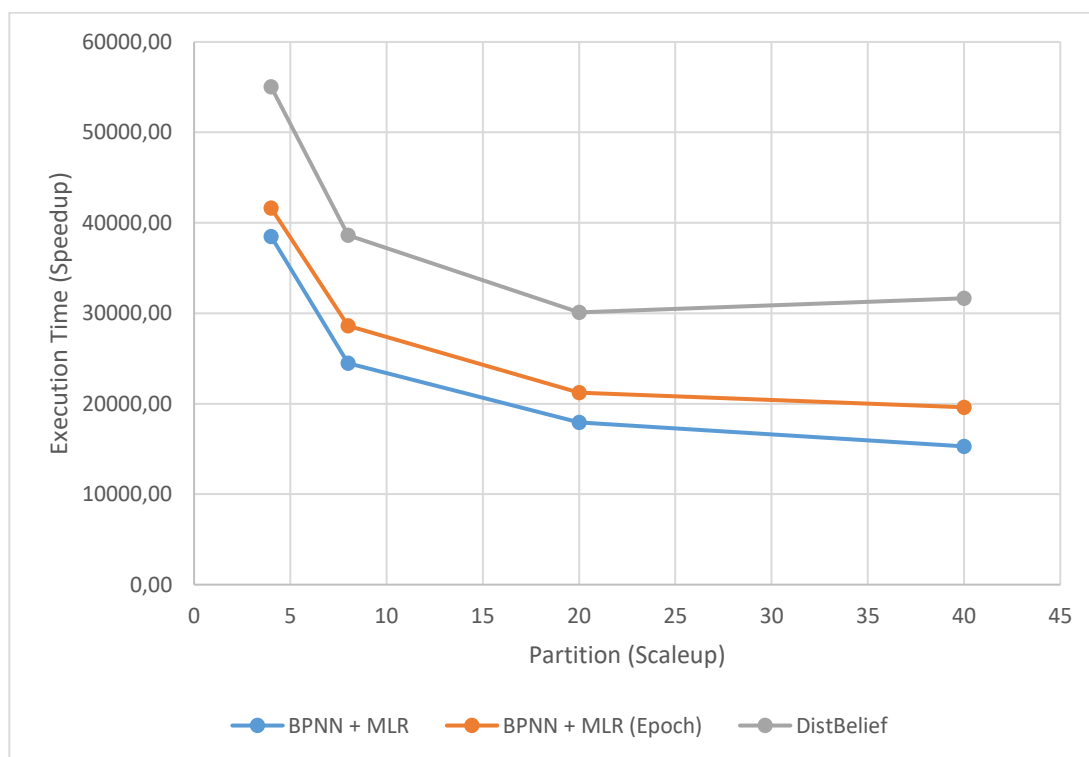


Figure 5.37: Frameworks Comparison: Molecular Model Scaleup vs Speedup

While scaling up the accuracy of our framework and DistBelief framework is decreased. However, the implementation of parallel MLR increases our framework accuracy by 3%-4.6% (see Figure 5.38).

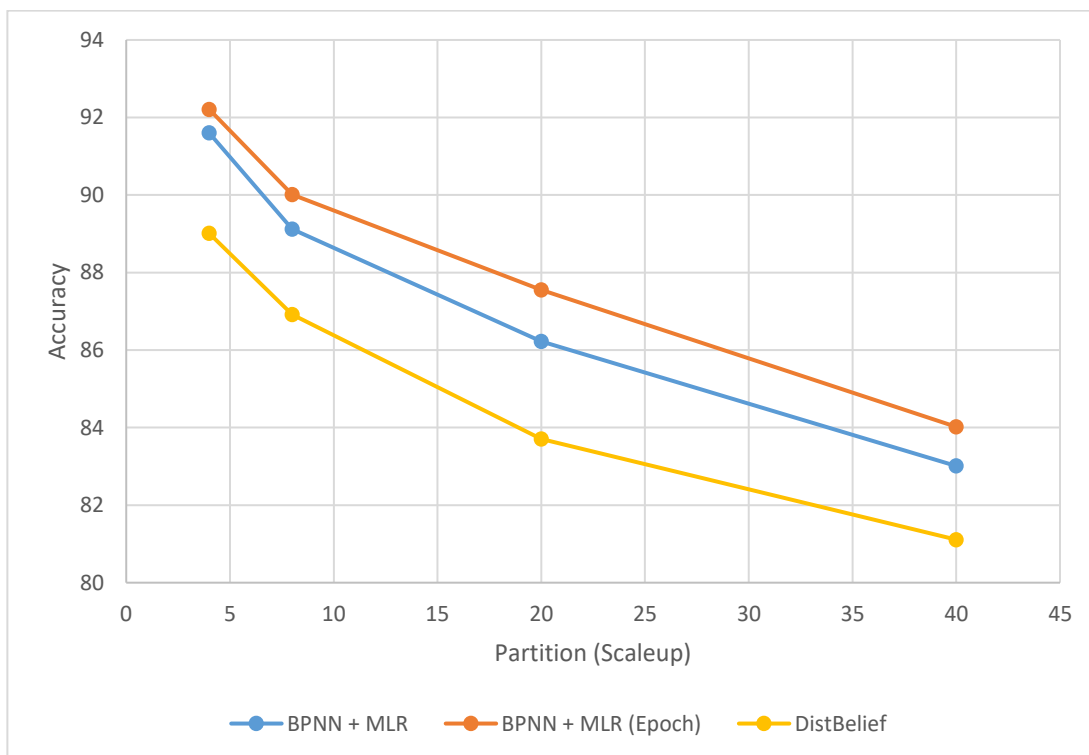


Figure 5.38: Frameworks Comparison: Molecular Model: Scaleup vs Accuracy

At least 4 partitions are required to store Molecular dataset. However, too many logical partitions slowdown the model due to the network traffics overheads since the model parameter size is 5.05MB.

CHAPTER 6

CONCLUSIONS & FUTURE WORK

BPNN algorithm is widely used to solve many problems in different domains. With the massive amount of data collected nowadays the performance of the traditional sequential versions of these algorithms are unable to handle this huge amount of data. People easily spend hours, weeks or even months or maybe years to train BPNN. A natural solution is parallel and distributed computing. Existing parallel BPNN solutions make the trade-off between efficiency and accuracy. The main drawback with these solutions is that they are computationally expensive and the training phase is a bottleneck. Therefore, a different approach of a solution is needed to improve BPNN algorithm efficiency and accuracy at the same time.

The main goal of the proposed framework is to improve BPNN performance in term of accuracy and execution time in the context of massive data. It contains two components. The first component partitions data and distributes the com-

putation of BPNN algorithm between cluster node, while the second component implements parallel multiple linear algorithms using linear algebra QR matrix decomposition method and predicts the final output. Our framework is unique and not like existing solutions. First, it benefits from the fast in-memory processing power provided by the Spark cluster. Second, it is able to handle the extensive computations of BPNN algorithm using the concept of mini-batches, while the employment of Tensorflow framework accelerates matrices computations and permits the framework to easily run in GPU environment. Third, the implementation of the parallel multiple linear regression algorithm improves our framework accuracy.

The experimental results show that our framework outperforms DistBelief frameworks in term of accuracy and execution time. Using our framework with data parallelism sizes vary between 4, 8, 20, and 40. Small models like Mnist model gained 66% speedup, while these type of models gained 54% speedup using DistBelief framework. Larger models with larger datasets like Molecular model gained 63% speedup compared to 52% speedup gained using DistBelief framework, while models with larger model parameters size gained 60% speedup compared to 42% speedup gained using DistBelief framework. DistBelief framework slowdown when scaling above 40 data parallelism size, while our framework is stable. In addition, the implementation of the parallel multiple linear regression algorithm improves our model accuracy by 3-5% compared to DistBelief framework. In general, our framework scaleup very well as it requires less computation time and less network overhead compared to other models.

Our framework is evaluated using three physical servers each of 24 cores, 2.0 GHz speed, and 2 network card of 100.0 MBPS. These resources are shared among 28 virtual machines which are used to build our Hadoop cluster. The maximum data parallelism size used to evaluate our framework was limited to 40 only. This was due to the limitations on the hardware available used for the training phase. However, a real cluster environment with more physical nodes and GPU support is required for performing all needed experiments to evaluate the performance of our framework.

As future work, it is interesting to evaluate our framework on GPU, TPU environments since our framework evaluation models are capable to run in such type of environments. It is also interesting to find a method of implementing our framework using a tensor decomposition optimization technique which may help to deal with the extensive computation of BPNN algorithm efficiently. Our framework can also benefit from the services provided by Apache Ignite framework, a peer to peer grid computing framework which provides a centric distributed shared memory that can be used to synchronized Spark Workers.

REFERENCES

- [1] Apache, *Apache Spark TM- Lightning-Fast Cluster Computing*, 2018 (accessed May 9, 2018). [Online]. Available: <http://spark.apache.org/>
- [2] A. Oussous, F.-Z. Benjelloun, A. A. Lahcen, and S. Belfkih, “Big data technologies: A survey,” *Journal of King Saud University - Computer and Information Sciences*, 2017. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1319157817300034>
- [3] P. Yadav and S. Vishwakarma, “Application of internet of things and big data towards a smart city,” in *2018 3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU)*, Feb 2018, pp. 1–5.
- [4] H. Kaur and A. S. Kushwaha, “A review on integration of big data and iot,” in *2018 4th International Conference on Computing Sciences (ICCS)*, Aug 2018, pp. 200–203.
- [5] Y. Liu, J. Yang, Y. Huang, L. Xu, S. Li, and M. Qi, “Mapreduce based parallel neural networks in enabling large scale machine learning,” *CoRR*,

vol. abs/1410.4453, 2015. [Online]. Available: <http://arxiv.org/abs/1410.4453>

- [6] V. Sunderam, “Current trends in high performance parallel and distributed computing,” in *Proceedings International Parallel and Distributed Processing Symposium*, April 2003, pp. 1 pp.–.
- [7] O. Delannoy and S. Petiton, “A peer to peer computing framework: design and performance evaluation of yml,” in *Third International Symposium on Parallel and Distributed Computing/Third International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks*, July 2004, pp. 362–369.
- [8] T. Mattson, B. Sanders, and B. Massingill, “Patterns for parallel programming,” 09 2004.
- [9] K. Mivule, B. Harvey, C. Cobb, and H. El-Sayed, “A review of cuda, mapreduce, and pthreads parallel computing models,” *CoRR*, vol. abs/1410.4453, 2014. [Online]. Available: <http://arxiv.org/abs/1410.4453>
- [10] Y. Wang, H. An, Z. Liu, T. Liu, and D. Zhao, “Parallelizing back propagation neural network on speculative multicores,” in *2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS)*, Dec 2016, pp. 902–907.

- [11] Z. Chenjie and R. Ruonan, “The improved bp algorithm based on mapreduce and genetic algorithm,” in *2012 International Conference on Computer Science and Service System*, Aug 2012, pp. 1567–1570.
- [12] B. B. Kivilcim, I. O. Ertugrul, and F. T. Yarman-Vural, “Modeling brain networks with artificial neural networks,” *CoRR*, vol. abs/1807.08368, 2018. [Online]. Available: <http://arxiv.org/abs/1807.08368>
- [13] C. Cheng, X. Cheng, N. Dai, X. Jiang, Y. Sun, and W. Li, “Prediction of facial deformation after complete denture prosthesis using bp neural network,” *Computers in Biology and Medicine*, vol. 66, pp. 103 – 112, 2015. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0010482515002991>
- [14] A. B. M. Moniruzzaman and S. A. Hossain, “Nosql database: New era of databases for big data analytics - classification, characteristics and comparison,” *CoRR*, vol. abs/1307.0191, 2013. [Online]. Available: <http://arxiv.org/abs/1307.0191>
- [15] Y. Liu, W. Jing, and L. Xu, “Parallelizing Backpropagation Neural Network Using MapReduce and Cascading Mode,” *Computational Intelligence and Neuroscience*, vol. 2016, no. 2842780, p. 11, 2016.
- [16] G. Ren, Q. Hua, P. Deng, and C. Yang, “Fp-mrbp: Fine-grained parallel mapreduce back propagation algorithm,” in *Artificial Neural Networks and Machine Learning – ICANN 2017*, A. Lintas, S. Rovetta, P. F. Verschure,

and A. E. Villa, Eds. Cham: Springer International Publishing, 2017, pp. 680–687.

- [17] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Y. Ng, “Large scale distributed deep networks,” in *NIPS*, 2012.
- [18] S. Sagiroglu and D. Sinanc, “Big data: A review,” in *2013 International Conference on Collaboration Technologies and Systems (CTS)*, May 2013, pp. 42–47.
- [19] A. B. M. Moniruzzaman and S. A. Hossain, “Nosql database: New era of databases for big data analytics - classification, characteristics and comparison,” *CoRR*, vol. abs/1307.0191, 2013. [Online]. Available: <http://arxiv.org/abs/1307.0191>
- [20] E. A. Mohammed, B. H. Far, and C. Naugler, “Applications of the mapreduce programming framework to clinical big data analysis: current landscape and future trends,” in *BioData Mining*, 2014.
- [21] A. Oussous, F.-Z. Benjelloun, A. A. Lahcen, and S. Belfkih, “Big data technologies: A survey,” *Journal of King Saud University - Computer and Information Sciences*, vol. 30, no. 4, pp. 431 – 448, 2018. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1319157817300034>

- [22] J. Dean and S. Ghemawat, “Mapreduce: Simplified data processing on large clusters,” *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008. [Online]. Available: <http://doi.acm.org/10.1145/1327452.1327492>
- [23] G. Yang, “The application of mapreduce in the cloud computing,” in *2011 2nd International Symposium on Intelligence Information Processing and Trusted Computing*, Oct 2011, pp. 154–156.
- [24] M. Afzali, N. Singh, and S. Kumar, “Hadoop-mapreduce: A platform for mining large datasets,” in *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, March 2016, pp. 1856–1860.
- [25] M. R. M. VeeraManickam, M. Mohanapriya, B. K. Pandey, S. Akhade, S. A. Kale, R. Patil, and M. Vigneshwar, “Map-reduce framework based cluster architecture for academic student’s performance prediction using cumulative dragonfly based neural network,” *Cluster Computing*, Feb 2018. [Online]. Available: <https://doi.org/10.1007/s10586-017-1553-5>
- [26] Y. Zhang, T. Cao, S. Li, X. Tian, L. Yuan, H. Jia, and A. V. Vasilakos, “Parallel processing systems for big data: A survey,” *Proceedings of the IEEE*, vol. 104, no. 11, pp. 2114–2136, Nov 2016.
- [27] J. Weets, M. K. Kakhani, and A. Kumar, “Limitations and challenges of hdfs and mapreduce,” in *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*, Oct 2015, pp. 545–549.

- [28] V. S. Jonnalagadda, P. Srikanth, K. Thumati, S. H. Nallamala, and A. Professors, “A review study of apache spark in big data processing,” *International Journal of Computer Science Trends and Technology (IJCST)*, vol. 4, pp. 93–98, March 2016.
- [29] K. Singh and R. Kaur, “Hadoop: Addressing challenges of big data,” in *2014 IEEE International Advance Computing Conference (IACC)*, Feb 2014, pp. 686–689.
- [30] V. A. Ayma and R. S. Ferreira, “Classification algorithms for big data analysis, a map reduce approach,” 2015.
- [31] Apache, *Apache Hadoop*, 2018 (accessed May 9, 2018). [Online]. Available: <http://hadoop.apache.org/>
- [32] —, *The Hadoop ecosystem table*, 2018 (accessed May 9, 2018). [Online]. Available: <https://hadooecosystemtable.github.io/>
- [33] —, *Apache mahout*, 2018 (accessed May 9, 2018). [Online]. Available: <http://mahout.apache.org>
- [34] K. K. Pathrikar¹ and A. A. Dudhgaonkar², “Review on apache spark technology,” *International Journal of Computer Science Trends and Technology (IJCST)*, vol. 4, 2017.
- [35] A.-K. Koliopoulos, P. Yiapanis, F. Tekiner, G. Nenadic, and J. Keane, “A parallel distributed weka framework for big data mining using

- spark,” *CoRR*, vol. abs/1410.4453, 2015. [Online]. Available: <http://arxiv.org/abs/1410.4453>
- [36] Y. Liu, L. Xu, and M. Li, “The parallelization of back propagation neural network in mapreduce and spark,” *International Journal of Parallel Programming*, vol. 45, no. 4, pp. 760–779, Aug 2017. [Online]. Available: <https://doi.org/10.1007/s10766-016-0401-1>
- [37] N. Pentreath, *Machine Learning with Spark*, ser. Community experience distilled. Packt Publishing, 2015. [Online]. Available: <https://books.google.fr/books?id=syPHBgAAQBAJ>
- [38] Apache, *Spark Community*, 2018 (accessed May 9, 2018). [Online]. Available: <https://spark.apache.org/community.html>
- [39] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, D. Xin, R. Xin, M. J. Franklin, R. Zadeh, M. Zaharia, and A. Talwalkar, “Mllib: Machine learning in apache spark,” *Journal of Machine Learning Research*, vol. 17, no. 34, pp. 1–7, 2016. [Online]. Available: <http://jmlr.org/papers/v17/15-237.html>
- [40] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauly, M. J. Franklin, S. Shenker, and I. Stoica, “Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing,” in *Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*. San Jose, CA: USENIX, 2012, pp.

- 15–28. [Online]. Available: <https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/zaharia>
- [41] K. Aziz, D. Zaidouni, e. Y. Bellafkih, Mostafa”, M. Lazaar, M. Al Achhab, and N. Enneya, “Big data optimisation among rdds persistence in apache spark,” in *Big Data, Cloud and Applications*. Cham: Springer International Publishing, 2018, pp. 29–40.
- [42] H. Kim, J. Park, J. Jang, and S. Yoon, “Deepspark: Spark-based deep learning supporting asynchronous updates and caffe compatibility,” 02 2016.
- [43] S. Tang, B. He, C. Yu, Y. Li, and K. Li, “A survey on spark ecosystem for big data processing,” *CoRR*, vol. abs/1811.08834, 2018. [Online]. Available: <http://arxiv.org/abs/1811.08834>
- [44] R. Bosagh Zadeh, X. Meng, A. Ulanov, B. Yavuz, L. Pu, S. Venkataraman, E. Sparks, A. Staple, and M. Zaharia, “Matrix computations and optimization in apache spark,” in *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’16. New York, NY, USA: ACM, 2016, pp. 31–38. [Online]. Available: <http://doi.acm.org/10.1145/2939672.2939675>
- [45] A. G. Shoro and T. R. Soomro, “Big data analysis: Ap spark perspective,” *Global Journal of Computer Science and Technology*, vol. 15, 2015.
- [46] S. Salloum, R. Dautov, X. Chen, P. X. Peng, and J. Z. Huang, “Big data analytics on apache spark,” *International Journal of Data Science*

- and Analytics*, vol. 1, no. 3, pp. 145–164, Nov 2016. [Online]. Available: <https://doi.org/10.1007/s41060-016-0027-9>
- [47] M. Lovri, J. M. Molero, and R. Kern, “Pyspark and rdkit: Moving towards big data in cheminformatics,” *Molecular Informatics*, vol. 0, no. 0, March, 2019. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/minf.201800082>
- [48] G. Luo, X. Luo, T. F. Gooch, L. Tian, and K. Qin, “A parallel dbscan algorithm based on spark,” in *2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom) (BDCloud-SocialCom-SustainCom)*, Oct 2016, pp. 548–553.
- [49] T. Zurek, “Optimal interval partitioning for temporal databases,” in *Proceedings of the Third Basque International Workshop on Information Technology - BIWIT'97 - Data Management Systems*, July 1997, pp. 140–147.
- [50] S. Mishra and A. C. Suman, “An efficient method of partitioning high volumes of multidimensional data for parallel clustering algorithms,” *CoRR*, vol. abs/1609.06221, 2016.
- [51] A. H. Atashkar, N. Ghadiri, and M. Joodaki, “Linked data partitioning for rdf processing on apache spark,” in *2017 3th International Conference on Web Research (ICWR)*, April 2017, pp. 73–77.

- [52] A. Gounaris, G. Kougka, R. Tous, C. T. Montes, and J. Torres, “Dynamic configuration of partitioning in spark applications,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 7, pp. 1891–1904, July 2017.
- [53] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauly, M. J. Franklin, S. Shenker, and I. Stoica, “Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing,” in *Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*. San Jose, CA: USENIX, 2012, pp. 15–28. [Online]. Available: <https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/zaharia>
- [54] S. S. 2016, *Distributed TensorFlow on Spark First presented at the 2016 Spark Summit East*, 2018 (accessed Nov 9, 2018). [Online]. Available: <https://github.com/adatao/tensorspark>
- [55] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. J. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Józefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. G. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. A. Tucker, V. Vanhoucke, V. Vasudevan, F. B. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “Tensorflow: Large-scale machine learning on heterogeneous distributed systems,” *CoRR*, vol. abs/1603.04467, 2016. [Online]. Available: <http://arxiv.org/abs/1603.04467>

- [56] W. 2018, *TensorFlow*, 2018 (accessed Nov 9, 2018). [Online]. Available: <https://en.wikipedia.org/wiki/TensorFlow/>
- [57] *Google Open-Sources The Machine Learning Tech Behind Google Photos Search, Smart Reply And Mor.*
- [58] G. 2015, *Distributed TensorFlow*, 2018 (accessed Nov 9, 2018). [Online]. Available: <https://www.tensorflow.org/>
- [59] D. 2016, *TensorFrames (TensorFlow on Spark DataFrames) lets you manipulate Apache Spark's DataFrames with TensorFlow programs*, 2018 (accessed May 9, 2018). [Online]. Available: <https://github.com/databricks/tensorframes/>
- [60] WikiPedia, "Artificial neural network," 2019, (accessed Mar 31, 2019). [Online]. Available: https://en.wikipedia.org/wiki/Artificial_neural_network
- [61] J.-T. Tsai, J.-H. Chou, and T.-K. Liu, "Tuning the structure and parameters of a neural network by using hybrid taguchi-genetic algorithm," *IEEE Transactions on Neural Networks*, vol. 17, no. 1, pp. 69–80, Jan 2006.
- [62] Y. Fu and T. Chai, "Neural-network-based nonlinear adaptive dynamical decoupling control," *IEEE Transactions on Neural Networks*, vol. 18, no. 3, pp. 921–925, May 2007.

- [63] J. Peng, K. Li, and G. W. Irwin, “A new jacobian matrix for optimal learning of single-layer neural networks,” *IEEE Transactions on Neural Networks*, vol. 19, no. 1, pp. 119–129, Jan 2008.
- [64] J. Cao, H. Cui, H. Shi, and L. Jiao, “Big data: A parallel particle swarm optimization-back-propagation neural network algorithm based on mapreduce,” *PLOS ONE*, vol. 11, no. 6, pp. 1–17, 06 2016. [Online]. Available: <https://doi.org/10.1371/journal.pone.0157551>
- [65] T. Kamio, S. Tanaka, and M. Morisue, “Backpropagation algorithm for logic oriented neural networks,” in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, vol. 2, July 2000, pp. 123–128 vol.2.
- [66] P. Werbos and J. Paul, “Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.” 1974.
- [67] B. Buscema, “Back propagation neural networks,” pp. 233–70, 02 1998.
- [68] F. C. Chen, “Back-propagation neural network for nonlinear self-tuning adaptive control,” 10 1989, pp. 274 – 279.
- [69] N. Archer and S. Wang, “Application of the back propagation neural network algorithm with monotonicity constraints for twogroup classification problems*,” *Decision Sciences*, vol. 24, pp. 60 – 75, 06 2007.

- [70] Z. hai Guo, J. Wu, H. yan Lu, and J. zhou Wang, “A case study on a hybrid wind speed forecasting method using bp neural network,” *Knowledge-Based Systems*, vol. 24, no. 7, pp. 1048 – 1056, 2011. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0950705111000852>
- [71] F. Zhang, P. Li, Z.-G. Hou, Z. Lu, Y. Chen, Q. Li, and M. Tan, “semg-based continuous estimation of joint angles of human legs by using bp neural network,” *Neurocomputing*, vol. 78, no. 1, pp. 139 – 148, 2012, selected papers from the 8th International Symposium on Neural Networks (ISNN 2011). [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0925231211004784>
- [72] R. Hecht-Nielsen, “Neural networks for perception (vol. 2),” H. Wechsler, Ed. Orlando, FL, USA: Harcourt Brace & Co., 1992, ch. Theory of the Backpropagation Neural Network, pp. 65–93. [Online]. Available: <http://dl.acm.org/citation.cfm?id=140639.140643>
- [73] F. M. Silva and L. B. Almeida, “Speeding up backpropagation,” in *Advanced Neural Computers*, R. ECKMILLER, Ed. Amsterdam: North-Holland, 1990, pp. 151 – 158. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/B9780444884008500224>
- [74] R. Rojas, *Neural Networks*. New York, NY: Springer New York, 1996.
- [75] A. B. H and V. N, “Privacy preserving back-propagation neural network learning made practical with cloud computing,” *IEEE Trans. Parallel*

- Distrib. Syst.*, vol. 25, no. 1, pp. 212–221, Jan. 2014. [Online]. Available: <http://dx.doi.org/10.1109/TPDS.2013.18>
- [76] J. A. Cruz-Lopez, V. Boyer, and D. El-Baz, “Training many neural networks in parallel via back-propagation,” *International Journal of Parallel Programming*, vol. 45, no. 4, pp. 760–779, Aug 2017. [Online]. Available: <https://doi.org/10.1007/s10766-016-0401-1>
- [77] S. Scardapane and P. DiLorenzo, “A framework for parallel and distributed training of neural networks,” *Neural Networks*, vol. 91, pp. 42 – 54, 2017. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0893608017300849>
- [78] D. J. Wasem, *Mining of Massive Datasets*. USA: CreateSpace Independent Publishing Platform, 2014.
- [79] J. D. Jobson, *Multiple Linear Regression*. New York, NY: Springer New York, 1991. [Online]. Available: https://doi.org/10.1007/978-1-4612-0955-3_4
- [80] G. K. Uyanik and N. Guer, “A study on multiple linear regression analysis,” *4th International Conference in New Horizon in Education*, pp. 234–240, 05 2013.
- [81] O. Brnlund and T. Johnsen, “Qr-factorization of partitioned matrices: Solution of large systems of linear equations with non-definite coefficient matrices,” *Computer Methods in Applied Mechanics and Engineering*,

- vol. 3, no. 2, pp. 153 – 172, 1974. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/0045782574900231>
- [82] P. G. Constantine and D. F. Gleich, “Tall and skinny qr factorizations in mapreduce architectures,” in *Proceedings of the Second International Workshop on MapReduce and Its Applications*, ser. MapReduce ’11. New York, NY, USA: ACM, 2011, pp. 43–50. [Online]. Available: <http://doi.acm.org/10.1145/1996092.1996103>
- [83] A. R. Benson, D. F. Gleich, and J. Demmel, “Direct qr factorizations for tall-and-skinny matrices in mapreduce architectures,” *2013 IEEE International Conference on Big Data*, pp. 264–272, 2013.
- [84] G. Golub and F. Uhlig, “The qr algorithm: 50 years later its genesis by john francis and vera kublanovskaya and subsequent developments,” *IMA Journal of Numerical Analysis*, vol. 29, no. 3, pp. 467–485, 2009. [Online]. Available: <http://dx.doi.org/10.1093/imanum/drp012>
- [85] A. Frank, D. Fabregat-Traver, and P. Bientinesi, “Large-scale linear regression: Development of high-performance routines,” *Applied Mathematics and Computation*, vol. 275, pp. 411 – 421, 2016. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0096300315015805>
- [86] M. R. Adjout and F. Boufares, “A massively parallel processing for the multiple linear regression,” in *2014 Tenth International Conference on Signal-Image Technology and Internet-Based Systems*, Nov 2014, pp. 666–671.

- [87] Z. Liu, H. Li, and G. Miao, “Mapreduce-based backpropagation neural network over large scale mobile data,” in *2010 Sixth International Conference on Natural Computation*, vol. 4, Aug 2010, pp. 1726–1730.
- [88] M. Zinkevich, M. Weimer, L. Li, and A. J. Smola, “Parallelized stochastic gradient descent,” in *Advances in Neural Information Processing Systems 23*, J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, Eds. Curran Associates, Inc., 2010, pp. 2595–2603. [Online]. Available: <http://papers.nips.cc/paper/4006-parallelized-stochastic-gradient-descent.pdf>
- [89] Q. He, T. Shang, F. Zhuang, and Z. Shi, “Parallel extreme learning machine for regression based on mapreduce,” *Neurocomputing*, vol. 102, pp. 52 – 58, 2013, advances in Extreme Learning Machines (ELM 2011). [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0925231212004213>
- [90] C. Dobre and F. Xhafa, “Parallel programming paradigms and frameworks in big data era,” *Int. J. Parallel Program.*, vol. 42, no. 5, pp. 710–738, Oct. 2014. [Online]. Available: <http://dx.doi.org/10.1007/s10766-013-0272-7>
- [91] C. Ren, N. An, J. Wang, L. Li, B. Hu, and D. Shang, “Optimal parameters selection for bp neural network based on particle swarm optimization: A case study of wind speed forecasting,” *Knowledge-*

- Based Systems*, vol. 56, pp. 226 – 239, 2014. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0950705113003730>
- [92] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, “Do we need hundreds of classifiers to solve real world classification problems?” *Journal of Machine Learning Research*, vol. 15, pp. 3133–3181, 2014. [Online]. Available: <http://jmlr.org/papers/v15/delgado14a.html>
- [93] A. Suliman and Y. Zhang, “A review on backpropagation neural networks in the application of remote sensing image classification,” 2015.
- [94] J. Zheng, Q. Ma, and W. Zhou, “Performance comparison of full-batch bp and mini-batch bp algorithm on spark framework,” in *2016 8th International Conference on Wireless Communications Signal Processing (WCSP)*, Oct 2016, pp. 1–5.
- [95] L. Wang, J. Tao, R. Ranjan, H. Marten, A. Streit, J. Chen, and D. Chen, “G-hadoop: Mapreduce across distributed data centers for data-intensive computing,” *Future Generation Computer Systems*, vol. 29, no. 3, pp. 739 – 750, 2013, special Section: Recent Developments in High Performance Computing and Security. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0167739X12001744>
- [96] J. Cao, L. Chen, M. Wang, H. Shi, and Y. Tian, “A parallel adaboost-backpropagation neural network for massive image dataset classification,” *Scientific Reports*, vol. 6, p. 38201, 12 2016.

- [97] Y. Zhang, Q. Gao, L. Gao, and C. Wang, “imapreduce: A distributed computing framework for iterative computation,” *Journal of Grid Computing*, vol. 10, no. 1, pp. 47–68, Mar 2012. [Online]. Available: <https://doi.org/10.1007/s10723-012-9204-9>
- [98] Y. Zhang, S. Chen, Q. Wang, and G. Yu, “i2mapreduce: Incremental mapreduce for mining evolving big data,” pp. 1482–1483, 05 2016.
- [99] I. Mustapha, S. Hasan, S. M. Shamsuddin, N. Lopes, and W. Yee Leng, “Gpu-based multiple back propagation for big data problems,” *International Journal of Advances in Soft Computing and its Applications*, vol. 8, 04 2016.
- [100] B. D. Haeffele and R. Vidal, “Global optimality in neural network training,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
- [101] E.-N. Ibrahim and Z. Mohamed, “Improving error back propagation algorithm by using cross entropy error function and adaptive learning rate,” 2017.
- [102] L. M. B. Sulaiman, S. Hasan, and S. M. Shamsuddin, “Multi back propagation (mbp) algorithm for big data application,” vol. 2, 2017.
- [103] R. Venkadachalam, P. Baskaran, A. Krishnamoorthy, D. Damodaran, and P. Sadasivam, “Back propagation neural network based big data analytics for a stock market challenge,” *Communication in Statistics- Theory and Methods*, pp. 1–21, 11 2018.

- [104] N. Courant Institute, “The mnist database,” 2019, (accessed Mar 31, 2019). [Online]. Available: <http://yann.lecun.com/exdb/mnist/>
- [105] C. for Machine Learning and I. Systems, “Higgs data set,” 2019, (accessed Mar 31, 2019). [Online]. Available: <https://archive.ics.uci.edu/ml/datasets/HIGGS>
- [106] P. Baldi, P. Sadowski, and D. O. Whiteson, “Searching for exotic particles in high-energy physics with deep learning.” *Nature communications*, vol. 5, p. 4308, 2014.
- [107] Merck, “Merck molecular activity challenge,” 2019, (accessed Mar 31, 2019). [Online]. Available: <https://www.kaggle.com/c/MerckActivity>

Vitae

- Name: Ismail Gasim A. Abdelgader
- Nationality: Sudanese
- Date of Birth: 01-Jan-1970
- Email: *iabdelgader@sfhd.med.sa*
- Permanent Address: Suadn, Khartoum
- Permanent Address: Suadn, Khartoum
- Educational Qualifications:
 - MS in Computer Science (IP) (King Fahad University of Petroleum & Minerals, KSA).
 - BS (Honors) in Computer Science (University of Khartoum, Sudan).
- Professional Certifications:
 - Oracle Certified Professional (OCP Database Track 2002).
 - Java Programmer (SCJP Track: Java 2 Platform Edition, 2004).
- Courses & Training:
 - Attended a Tutorial on TCPIP Based Networking (18th Annual Computer Exhibition, 2000, KFUPM, KSA).

- Attended a Tutorial on Web Development (18th Annual Computer Exhibition, 2000, KFUPM, KSA).
- Attended a Tutorial on Educational computer techniques (Ministry of Education 1998, KSA).
- Summary of Experiences:
 - 14+ years highly experienced, talented and dependable Senior Oracle database administrator).
 - Intensive work in implementing Oracle RAC, ASM, RMAN, Data Pump, Performance Tuning skills, Data Guard, Cloning.
 - Ensure high availability by managing & implementation of standby databases and configuration of data guard.
 - Performance tuning advanced replication, migration and troubleshooting.
 - Oracle database upgrade & applying critical Patching update (CPU).