Optimization of machine learning training workflows on Apache Spark cluster

Marios Marinos
Computer Science TU Delft
TU Delft
Delft, Netherlands
M.Marinos@student.tudelft.nl

Damian Voorhout
Computer Science TU Delft
TU Delft
Delft, Netherlands
D.M.Voorhout@student.tudelft.nl

Nick-Andian Tehrany
Computer Science TU Delft
TU Delft
Delft, Netherlands
N.Tehrany@student.tudelft.nl

***Abstract*—With artificial intelligence and machine learning becoming an ever more fundamental part of society, comes the need to constantly improve performance and required training times of these models. To this end, distributed deep learning offers the building foundation to expedite training times of models, while maintaining high accuracy. These platforms provide endless configurations of system- and hyper- parameters, making the finding of optimal parameter setups, that deliver low training time with high accuracy, a challenging effort. To this end, we propose a model to analyze the effects of a number of possible parameter setups on the overall performance of the model, taking into consideration the required training time, normalized with the achieved accuracy. Additionally, we construct a queuing model to calculate response times of individual jobs, as well as the number of jobs in the system. Based on the findings from said models, we optimize the parameters to achieve the lowest cost. Results reveal that increasing the learning rate leads to logarithmically lower overall cost. Furthermore, the queuing model reveals an increased number of jobs in the system, as well as a decrease in job time, when a higher learning rate is utilized.**

***Index Terms*—Apache Spark, BigDL, Distributed Deep Learning, Design of Experiments, Queuing Theory**

I. Introduction

In the modern age, big data and artificial intelligence are becoming an ever increasing part of daily life with smart personal assistants, such as Amazon Alexa, and self-driving cars [1]. At the center is the development of deep learning models, such as deep neural networks (DNNs), for tasks including natural language processing, pattern- and image-recognition, and medical diagnosis. The increasing demand for said models requires constant improving of their performance for better accuracy and faster execution. Obstacles arise, as training of such models requires large amounts of data [2], and the training of the model can require several hours to multiple days. Contrary to popular belief, the root of this issue comes from available algorithms, as opposed to the computational power of available hardware, as it is difficult to fully leverage the capacities of supercomputers, and hardware in general, for training of DNN's [3].

Aiming to overcome this obstacle is the development of distributed cluster frameworks, which provide the possibility of distributed deep learning. Some of the popular distributed cluster platforms used for deep learning are Apache Spark [4], Tensorflow [5], and Apache MXNet [6]. The goal of said distributed deep learning platforms is to parallelize the training process of models, by executing training in a multi-node environment, where each node is assigned part of the work from the model. Benchmarks of commonly used distributed machine learning platforms show a linear speedup in training times [7] over non-distributed deep learning. With such development, comes the issue of how to optimally arrange distributed systems in order to achieve the lowest training time, while maintaining high accuracy.

This calls for efforts to minimize the cost of these systems, by identifying the trade-offs between the time it takes for training of these models, and the achieved accuracy of the model. Many factors play a role in this, ranging from the system parameters of how many nodes are used and what computational power each node has, to the hyper parameters of the code that is being run on said training model. The goal is to maintain high accuracy of models, but minimize training time and overall cost of model training.

We propose a predictive model for analyzing the affects of different factors within the system- and hyper- parameter setup on the overall cost of the system. To this end, we additionally construct a cost function given as,

$$Cost = \frac{T}{(A * 100)^2}$$

where T depicts the required training time of the model, and A the achieved accuracy. This takes into consideration the total training time and achieved accuracy, giving a normalized cost from the the time normalized by the squared accuracy. The aim is to minimize model training times in a distributed setting, while maintaining high accuracy, resulting in an overall low calculated cost from the cost function. Additionally, we present a queuing model, which calculates the mean response time of jobs and the average number of jobs in the system, with different numbers of worker nodes. All source code, from retrieved results of the experiments to construction of the different models, is openly available[1].

Results of the predictive model and the integrated cost function show that the most influential factor is the model's learning rate, followed by the interaction of the learning

[1]https://github.com/MariosMarinos/Big-DL

rate with the batch size. Knowing said important factors, optimizations consisted of continuously increasing the learning rate until the cost function no longer decreased. Results reveal that the optimal learning rate, resulting in the lowest overall cost, was achieved with a value of 9.011.

II. Background

Adapting system- and hyper- parameters is a challenging problem, as there are several different parameters to adjust. The goal is to find parameters which decrease required time of training, while maintaining high accuracy. Related work has suggested several important hyper parameters, which affect the overall complexity of the model to be trained, as well as minimizing training times of the model [8]–[10]. Based on the related work, the following hyper parameters were chosen,

- N_{epoch}
- $learning\ rate$
- $batch\ size$

Intuitively, one would assume that all these factors have a major impact on the training times, as running less epochs will take less time, a higher learning rate increases the required steps in the model's loss function to find the minimal loss, and a higher batch size will also require less training time. These are part of the reasoning for including these factors, alongside the related work showing similar findings, but the affects of said factors on the cost function are to be determined by our models.

Optimization of system parameters depends on distributability of the model and the possible additional costs of distributing the model. Related work has investigated effects of several factors, with findings on the most influential ones [11], [12], from which the following factors were selected for our experiments,

- N_{nodes}
- $N_{executors}$
- $N_{executor\_cores}$

The number of nodes represents the number of workers, amongst which the training of the model will be split up. The number of executors depicts the the number of executors running tasks in each worker node, and the executor_cores presents the number of cores that each executor is running. Naturally, assuming the higher number of workers there are, the faster the training will be is an invalid assumption, as this depends on how much of the training can be run in parallel. Scheduling and distributing of jobs to the workers will also add overhead, which increases with the number of workers.

With a total of 6 factors presented in our model, 3 factors for the system parameters and 3 factors for the model hyper parameters, the goal is to build a model to analyze affects on the cost function by altercations on each factor. Additionally, examining to what extent distribution of training amongst the workers is beneficial in performance.

III. Experiments

A. Design

To determine the effects of a set of factors on a target variable, a test design, or test suite, is required. One way of doing so is by employing a "One-Factor-At-a-Time" (OFAT) strategy, where only a single factor is altered during each consecutive experiment. However, given the complexity of the problem statement, an OFAT strategy would lead to many experiments. Furthermore, OFAT does not lend itself well to estimating the interaction between factors. It is however reasonable to assume there are indeed factors that influence each other given the selected factors for this research. This makes OFAT an unsuitable design strategy. Conversely, factorial design does allow the detection of factor interactions. Similar to OFAT, Full factorial designs also require a substantial amount of experiments that need to be performed to obtain the most accurate results. Fractional factorial reduces the amount of experiments that are needed, but confounds interactions. This means we can not be sure which factors, or combinations of factors, are affecting an outcome. In practice this is often irrelevant as the primary factors are not confounded when only high level confoundings take place. We can ensure this is the case by making use of a fractional factorial design with high resolution.

Fractional factorial designs can be formulated as follows: $n = I^{k-p}$. Where I is the number of levels of each factor, k the number of factors, p the number of interactions that are confounded and n the number of required experiments. Clearly, we can reduce the number of experiments by increasing p, but this leads to more confoundings. To keep the number of confoundings to a minimum and thus creating a more insightful model, p is chosen to be 1. Also, as was discussed in the section II, there are a total of 6 factors which will be examined in this design, each being able to take 2 different values, or levels. This leads to a 2^{6-1} design and 32 required experiments. As was mentioned, to reduce the number of low level confoundings, designs with high resolution are desirable. As designs are not unique, many designs with high resolutions exist for a given k and p. Table I shows the used fractional 2^{6-1} design with the relationship $I = ABCDEF$, which can be found in [13].

Finally, preliminary testing showed high variance in the outcome for certain factor level combinations. Therefor 10 replications are performed for each experiment.

B. Factors and levels

The 2^{6-1} fractional factorial model considers only 2 levels for each factor, a low level and a high level, as represented by the -1 and $+1$ in table I respectively. Levels need to be chosen in such a way that a level has a significantly different effect on the outcome than the other level. At the same time, the

	A	B	C	D	E	F
1	-1	-1	-1	-1	-1	-1
2	+1	-1	-1	-1	-1	+1
3	-1	+1	-1	-1	-1	+1
4	+1	+1	-1	-1	-1	-1
5	-1	-1	+1	-1	-1	+1
6	+1	-1	+1	-1	-1	-1
7	-1	+1	+1	-1	-1	-1
8	+1	+1	+1	-1	-1	+1
9	-1	-1	-1	+1	-1	+1
10	+1	-1	-1	+1	-1	-1
11	-1	+1	-1	+1	-1	-1
12	+1	+1	-1	+1	-1	+1
13	-1	-1	+1	+1	-1	-1
14	+1	-1	+1	+1	-1	+1
15	-1	+1	+1	+1	-1	+1
16	+1	+1	+1	+1	-1	-1
17	-1	-1	-1	-1	+1	+1
18	+1	-1	-1	-1	+1	-1
19	-1	+1	-1	-1	+1	-1
20	+1	+1	-1	-1	+1	+1
21	-1	-1	+1	-1	+1	-1
22	+1	-1	+1	-1	+1	+1
23	-1	+1	+1	-1	+1	+1
24	+1	+1	+1	-1	+1	-1
25	-1	-1	-1	+1	+1	-1
26	+1	-1	-1	+1	+1	+1
27	-1	+1	-1	+1	+1	+1
28	+1	+1	-1	+1	+1	-1
29	-1	-1	+1	+1	+1	+1
30	+1	-1	+1	+1	+1	-1
31	-1	+1	+1	+1	+1	-1
32	+1	+1	+1	+1	+1	+1

TABLE I
SIGN TABLE FOR THE 2^{6-1} FRACTIONAL FACTORIAL DESIGN. -1 AND $+1$ REPRESENT LOW LEVELS AND HIGH LEVELS RESPECTIVELY.

levels are bounded by hardware and software considerations. The factor levels are as follows:

$$N_{nodes} \in \{1, 4\}$$
$$N_{executors} \in \{1, 4\}$$
$$N_{executor\_cores} \in \{1, 4\}$$
$$N_{epoch} \in \{1, 3\}$$
$$learning\ rate \in \{0.001, 0.01\}$$
$$batch\ size \in \{64, 256\}$$

The low level of the N_{nodes}, $N_{executors}$ and $N_{executor\_cores}$ factors are all bounded by a minimum of 1. Their maximum is set to 4, both to create a sizable gap between the high and low levels but also to limit the hardware requirements; each node requires a new instance. The hardware details are discussed in further detail in the next section.

The experiments are run using the MNIST dataset[2]. Each job represents the training of a classifier on this dataset, which is known to be among the easiest classification problems. As we are interested in the relationship between the time each job takes and what accuracy the job achieves, it makes sense to keep the N_{epochs} relatively low for both the high and low levels. The expectation is that the learner does not need many more epochs than 3 to achieve a decent classification accuracy, which is what preliminary tests also concluded. This is the reason the high level is set to 3 and the low level to 1 as this is the minimum number of epochs a learner can perform.

The levels of the learning rate are harder to determine. Its effect highly depends on the dataset, type of learning task and other factors. Experience and literature tells us that common learning rate values are somewhere around 0.001. As was previously mentioned, the learning task is a relatively easy one, and few epochs are used during training. This is why we can expect an increase in learning rate to lower the time needed to complete a task significantly. Therefore, it is sensible to set the high level of the learning rate an order of magnitude higher than the low level, which is set at the common value of 0.001.

Similar to the learning rate, the batch size can take on a wide range of values, from 1 to however much your resources can handle in terms of memory usage. However, in practice a batch size of 1 is rarely used in the context of ordinary classification tasks on simple datasets such as MNIST. To keep the batch size range in a more realistic range, it was decided to set the low level of $batch\ size$ to be 64. The high level is set to a somewhat modest 256, as the computational resources that were used in performing the experiments did not employ graphical processing units, only central processing units.

C. Hardware

All 32 experiments were run on the servers of Google Cloud[3], which works with virtual machine instances, meaning each instance has its own environment, boot disk and dedicated hardware. Furthermore, Google Cloud allows you to pick a hardware configuration from a list of options varying in available memory, number of processor cores and processor core speeds. Each test ran on one or four instances depending on the number of nodes the test required, each with a "e2-standard-4" hardware setting, meaning it has 4 virtual central processing cores and 16 GB of memory available. The central processing unit that Google Cloud provides in these instances are 2 core E2-Xeon processors with hyper-threading[4]. The generation of the processors are not directly provided and could be any of the following: Skylake, Broadwell and Haskell. As the E2 processors are considered general purpose, which one Google Cloud assigns to the created instances depends on availability.

Besides the 4 "e2-standard-4" instances used, a single "e2-standard-16" instance was employed for any experiment that has a configuration as follows: $N_{nodes} = 1$, $N_{executors} = 4$, $N_{executor\_cores} = 4$ and the other factors can take on any level. An example of such an experiment can be seen in row 7 of table II. These type of configurations require 16 (virtual) cores in a processor as each executor uses 4 cores. In the case where instead there is an experiment with the level of N_{nodes} set to high, one can get away with the 4 "e2-standard-4" instances as each instance will be able to handle a single executor which

[2]http://yann.lecun.com/exdb/mnist/

[3]https://cloud.google.com/

[4]https://cloud.google.com/compute/docs/machine-types#e2_machine_types

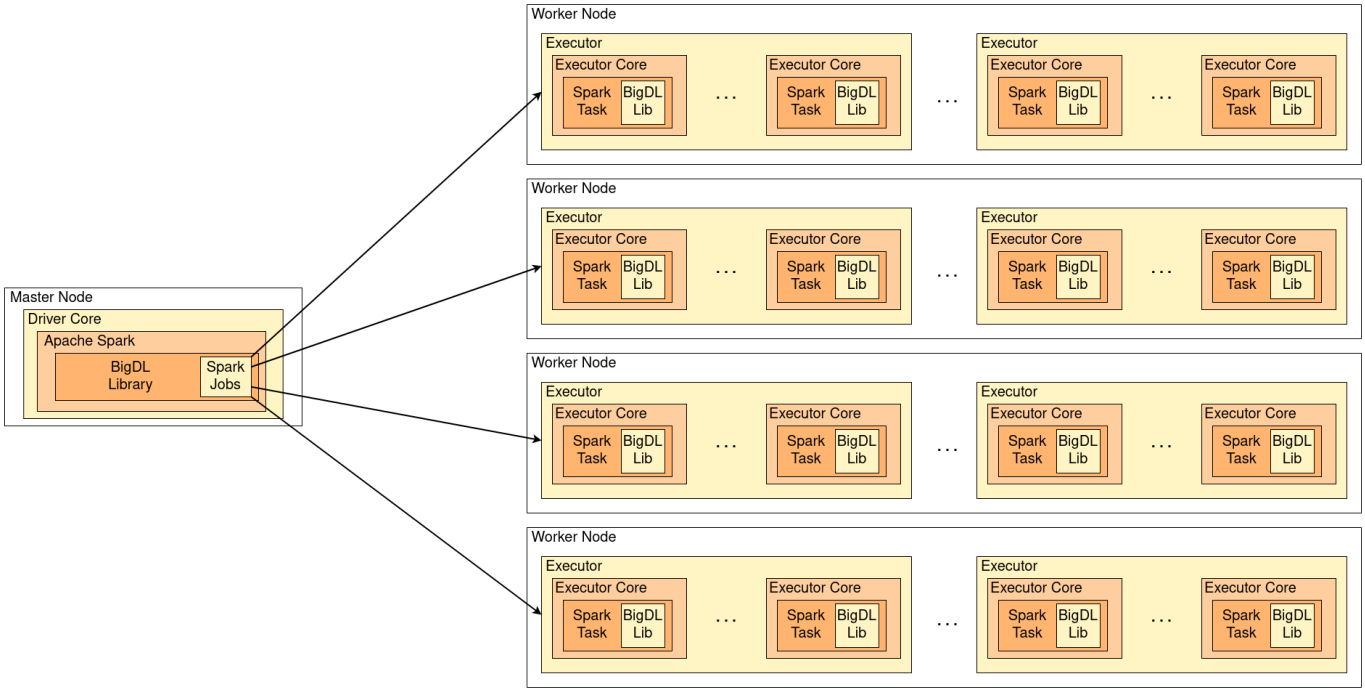

Fig. 1. Overview of the Apache Spark system setup that was used to perform the experiments.

makes use of 4 cores. An example of such an experiment can be found in row 16 of table II. The ”e2-standard-16” has 16 virtual cores available, as well as 64 GB of memory.

The instances mentioned so far are all instances for the worker nodes. The master node runs on its own separate instance, which is also an ”e2-standard-4” instance.

D. Software

Each experiment was run using Apache Spark[5]. Spark interacts with the cluster context and the executors on the worker nodes directly. The driver node, or master, assigns the jobs to the executors which then have full responsibility over that job. Each executor can split up the job into tasks and distribute them over each executor core. The results of each taks on the executor core is returned to the master node and consequently logged.

Besides Spark, the experiments are conducted using BigDL[6] which runs on top of Spark and provides machine learning APIs to it. This enables Spark to create, distribute, and interact with machine learning workflows. The full setup can be seen in figure 1.

For each job 2 output files are created, an .err file which contains any errors that may have occurred during the execution and an .out file, which contains all logs of the process. If any errors occurred, the .out file will contain the exceptions as well as any other logs created during the process. Assuming no errors occurred, the logs contain the information of each processed batch, as well as a summary for each processed epoch. This includes a wall clock which keeps track of the time that has passed for each subtask, as well as the accuracy the classifier has achieved on the given test set. An automated script reads each .out file, and given that it contains no errors, will extract the final wall clock and accuracy of the completed job. It uses these metrics to compute the cost of that particular job. The script is only executed when there are 10 or more .out files present to keep the variance in check.

[5]https://spark.apache.org/

[6]https://bigdl-project.github.io/0.11.0/

E. Results

The experiments were run with $\lambda = 0.01$, using the MNIST dataset and the provided bi-rnn network. The results of all experiments can be found in table II. It contains 10 replications per experiment for each setting of the factors according to the sign table I.

IV. Predictive Models

A. ANOVA Model

After having done the experiments with the 2^{6-1} fractional factorial that was described in section III, there is a need to decide on which predictive model should be used in order to find out which factors are most significant. By deciding to use an ANOVA model, there are five assumptions that need to be taken into account to get valid results.

- The factorial ANOVA requires the dependent variable in the analysis to be of metric measurement level, such as ratio or interval data, whereas the independent variables can be nominal.
- Observations are sampled independently from each other.

	N_{nodes}	$N_{executors}$	$N_{executor\_cores}$	N_{epochs}	$learning\_rate$	$batch\_size$	$cost$
1	1	1	1	1	0,001	64	0.044,0.019,0.034,0.038,0.032,0.025,0.025,0.033,0.034,0.042
2	4	1	1	1	0,001	256	0.176,0.192,0.061,0.057,0.123,0.151,0.166,0.177,0.277,0.188
3	1	4	1	1	0,001	256	0.173,0.218,0.056,0.127,0.253,0.107,0.171,0.135,0.273,0.166
4	4	4	1	1	0,001	64	0.055,0.057,0.019,0.025,0.034,0.047,0.019,0.025,0.035,0.028
5	1	1	4	1	0,001	256	0.095,0.035,0.146,0.057,0.071,0.058,0.118,0.107,0.112,0.134
6	4	1	4	1	0,001	64	0.03,0.023,0.022,0.025,0.018,0.021,0.025,0.028,0.026,0.018
7	1	4	4	1	0,001	64	0.043,0.042,0.024,0.02,0.026,0.035,0.017,0.059,0.034,0.037
8	4	4	4	1	0,001	256	0.037,0.091,0.223,0.029,0.129,0.136,0.082,0.545,0.062,0.074
9	1	1	1	3	0,001	256	0.08,0.046,0.035,0.047,0.107,0.078,0.158,0.094,0.072,0.063
10	4	1	1	3	0,001	64	0.034,0.03,0.037,0.031,0.036,0.034,0.032,0.036,0.042,0.038
11	1	4	1	3	0,001	64	0.062,0.058,0.05,0.052,0.053,0.05,0.056,0.048,0.056,0.057
12	4	4	1	3	0,001	256	0.012,0.011,0.012,0.015,0.014,0.014,0.013,0.014,0.013,0.014
13	1	1	4	3	0,001	64	0.039,0.024,0.026,0.029,0.026,0.022,0.023,0.027,0.023,0.021
14	4	1	4	3	0,001	256	0.012,0.012,0.009,0.011,0.009,0.01,0.009,0.011,0.009,0.011
15	1	4	4	3	0,001	256	0.04,0.072,0.029,0.058,0.055,0.056,0.028,0.063,0.115,0.029
16	4	4	4	3	0,001	64	0.027,0.017,0.015,0.019,0.014,0.03,0.026,0.011,0.017,0.021
17	1	1	1	1	0.01	256	0.006,0.006,0.007,0.006,0.006,0.007,0.005,0.006,0.008,0.006
18	4	1	1	1	0.01	64	0.043,0.035,0.027,0.058,0.056,0.017,0.032,0.031,0.024,0.044
19	1	4	1	1	0.01	64	0.022,0.018,0.019,0.024,0.022,0.023,0.021,0.023,0.019,0.017
20	4	4	1	1	0.01	256	0.019,0.021,0.02,0.021,0.02,0.021,0.021,0.02,0.022,0.022
21	1	1	4	1	0.01	64	0.008,0.007,0.008,0.007,0.008,0.008,0.008,0.009,0.008,0.008
22	4	1	4	1	0.01	256	0.014,0.014,0.011,0.01,0.009,0.01,0.011,0.013,0.011,0.011
23	1	4	4	1	0.01	256	0.004,0.005,0.006,0.005,0.005,0.005,0.005,0.005,0.007,0.005
24	4	4	4	1	0.01	64	0.015,0.009,0.014,0.014,0.016,0.015,0.009,0.015,0.012,0.015
25	1	1	1	3	0.01	64	0.019,0.02,0.02,0.02,0.019,0.02,0.019,0.019,0.019,0.019
26	4	1	1	3	0.01	256	0.011,0.011,0.011,0.011,0.011,0.012,0.011,0.011,0.011,0.012
27	1	4	1	3	0.01	256	0.011,0.011,0.012,0.01,0.011,0.011,0.012,0.011,0.011,0.011
28	4	4	1	3	0.01	64	0.037,0.037,0.036,0.037,0.036,0.036,0.036,0.037,0.037,0.037
29	1	1	4	3	0.01	256	0.008,0.008,0.008,0.008,0.008,0.008,0.008,0.008,0.008,0.008
30	4	1	4	3	0.01	64	0.018,0.018,0.018,0.018,0.019,0.018,0.018,0.019,0.018,0.018
31	1	4	4	3	0.01	64	0.035,0.035,0.035,0.037,0.036,0.036,0.036,0.036,0.034,0.036
32	4	4	4	3	0.01	256	0.012,0.011,0.009,0.013,0.009,0.014,0.014,0.009,0.008,0.013

TABLE II
SHOWS THE RESULTS OF THE 32 EXPERIMENTS WHERE EACH EXPERIMENT SETTING IS DETERMINED BY THE SIGN TABLE OF TABLE I. EACH EXPERIMENT IS REPLICATED 10 TIMES.

- Residuals, also called error values, are normally distributed. This property can be validated using the Shapiro-Wilks test.
- Homogeneity of variances: Variances are equal between different levels. This property can be validated using the Levene or Bartlett Test.
- There should be no multicollinearity among the factors of the model. Multicollinearity occurs when the independent variables are inter-correlated and not independent from each other.

In addition to that, the model does not take into account every interaction between the factors, instead all single factors are considered and some of the interactions, as can be seen in figure 2. To get an intuition on how good the model fits the data, taking a look at the R^2 measure is a good idea. R-squared is a goodness-of-fit measure for linear regression models. This statistic indicates the percentage of the variance in the dependent variable that the independent variables explain collectively. For instance, small R-squared values are not always a problem, and high R-squared values do not necessarily signify an accurate model. The presented model has a R-squared value of 40% which signifies that it is likely the model does not fit the data very well. In Figure 2 it can be seen that for a significance level of $\alpha = 0.05$ the factors that have $p < 0.05$ can be considered as statistically significant. These are learning rate, batch size and all of the interactions that are considered. That means, with 95% confidence, learning rate, batch size and the interactions have a strong influence on the outcome of the cost function. In the same figure, there is a column called "Percentage_Variation_Explained" from which we can conclude that indeed, learning rate is the most significant factor with a 13% contribution to the total variation of the model. Second is the interaction between learning rate and batch size with 11%. There is a significant percentage variation of the model that can not be explained, the residuals, which presumably roots from the fact that some of the initial assumptions of the proposed ANOVA model are wrong.

As there is reason to assume the model is inaccurate, it is paramount to check whether the model violates any of the assumptions. Starting with the assumption of normality of residuals, in figure 3 the distribution of the residuals is displayed in a Q-Q plot. The residuals seem to mostly follow the normal distribution, except for a single outlier. The Sharipo-Wilk test can also be used to check the normality of residuals with the following null hypothesis: The residuals from the model are drawn from the normal distribution. From table III it is clear that the p-value is not significant, hence, the null hypothesis cannot be rejected, concluding that it is

	df	sum_sq	mean_sq	F	PR(>F)	Percentage_Variation_Explained
N_nodes	1.0	0.000835	0.000835	0.811084	3.685497e-01	0.168076
N_Executors	1.0	0.002662	0.002662	2.586327	1.088819e-01	0.535950
NumberExecutorCores	1.0	0.000555	0.000555	0.539326	4.633077e-01	0.111761
Max_epochs	1.0	0.003148	0.003148	3.058204	8.139210e-02	0.633734
learning_rate	1.0	0.065909	0.065909	64.030983	2.993076e-14	13.268775
Max_epochs:learning_rate	1.0	0.013646	0.013646	13.257446	3.216297e-04	2.747265
Batch_Size	1.0	0.007529	0.007529	7.314456	7.245951e-03	1.515733
Max_epochs:Batch_Size	1.0	0.023049	0.023049	22.392219	3.483304e-06	4.640212
learning_rate:Batch_Size	1.0	0.055835	0.055835	54.244237	1.858615e-12	11.240724
Max_epochs:learning_rate:Batch_Size	1.0	0.026078	0.026078	25.334631	8.489754e-07	5.249951
Residual	289.0	0.297477	0.001029	NaN	NaN	59.887820

Fig. 2. Results of the ANOVA modeling, it shows the significance each factor has on the outcome of the cost function.

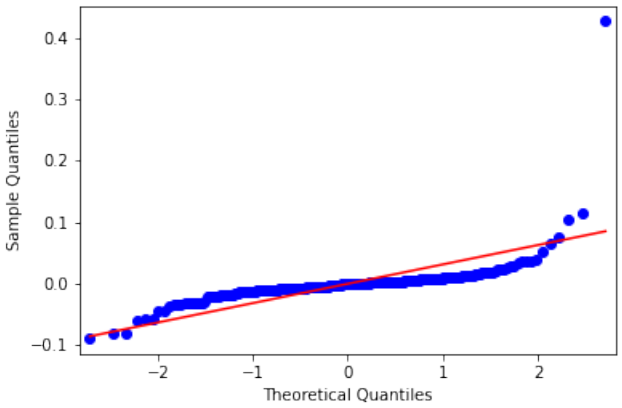

Fig. 3. Q-Q plot of the model's residuals and standardized line.

W	P-Value
0.4723438620567322	7.447296649928366e-29

TABLE III
SHARIPO-WILK TEST FOR NORMALITY OF RESIDUALS.

likely the residuals are drawn from the normal distribution.

To check the Homogeneity of variance assumption the Bartlett test is used if the data are normally distributed, otherwise the Levene test is preferred. In that case, the data are normally distributed, thus, the Bartlett test fits better.

W	P-Value
12919.024882636862	0.0

TABLE IV
BARTLETT'S TEST FOR HOMOGENEITY OF VARIANCES.

The Bartlett's test null hypothesis is: Samples from factors have equal variances. The p-value from table IV is not significant, thus the null hypothesis is rejected and it is safe to assume that the factors have equal variances.

Finally, the assumption of Multicollinearity should be checked to see whether their exists a linear relationship between the factors. A common convention is that the VIF-factor should at most be 10 [14]. A high VIF-Factor value means that there is a strong linear relationship between the independent variables, i.e. the factors. Therefore, in table V it can be seen that N_{epochs} and $learning\_rate$ have a high VIF value, which means that these independent variables can be predicted by other independent variables in the dataset. So, the Multicollinearity assumption is violated and could explain why the model does not fit the data very well.

Features	VIF Factors
$Intercept$	53.00742210464434
N_{nodes}	1.0044642857142865
$N_{executors}$	1.0044642857142851
$N_{executor\_cores}$	1.0666666666666662
N_{epochs}	9.08568815729309
$learning\_rate$	18.162962962962965
N_{epochs}:$learning\_rate$	23.67992684042065
$batch\_Size$	17.56378600823049
N_{epochs}:$batch\_Size$	24.788608008361116
$learning\_rate$:$batch\_Size$	32.26149323927107
N_{epochs}:$learning\_rate$:$batch\_Size$	33.942282317591015

TABLE V
VIF-FACTORS FOR EACH INDEPENDENT VARIABLE AND THEIR INTERACTIONS.

Finally, as can be seen in figure 4, the learning rate has the largest impact on the outcome of the cost function, which is quite understandable as by increasing the learning rate a decent accuracy can be achieved in a small amount of time. The other factors seems to be irrelevant, which is also to be expected, as far as the cost function is considered, because the experiments were only run using 1 epoch and 3 epochs, so the system parameters (Number of Nodes, Number of Executors, Number of Executor Cores) do not get to affect the training time, with a very slight negative impact of the max epochs and slight positive impact from the intercept.

B. Queuing Model

After creating the ANOVA model and getting the results, there is a need for creating a queuing model in order to get an estimation of how fast a system can process the jobs and also how many jobs are in a system on average. To calculate these metrics we need to first define the system. For simplicity, two systems were considered, a M/M/1 and a M/M/4 queuing

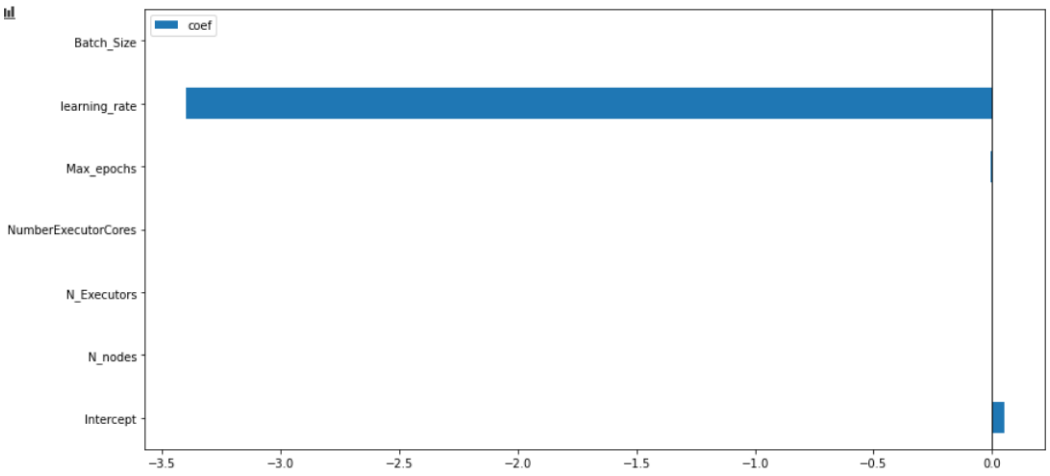

Fig. 4. Coefficient of each factor.

systems. Now, the models are oversimplified from the point of view of real life servers. Only the nodes are considered as different servers and not the number of executors, number of executor cores, etc. In addition to that, an assumption is made that the jobs arrivals and the mean service times from the jobs in the system are derived from a Poisson distribution. Taking everything into account, the calculation of the μ was based on the average cost of all performed experiments, which is time, normalized by squared accuracy. Finally, we assume $\rho = 0.8$, as this provides a realistic interpretation of a busy server. Below, for both systems, the derivations can be found for the average number of jobs in the system $E[N]$ and the average response time of the system $E[T]$. Again, note that "time" in this case refers to job time normalized by the resulting accuracy.

$$M/M/1\ Queue : \lambda = \rho * \mu \Rightarrow 0.8 * 3.085 = 2.468$$
$$E[N] = \frac{\rho}{1-\rho} = \frac{0.8}{1-0.8} = 4$$
$$E[T] = \frac{E[N]}{\lambda} = \frac{4}{0.259} = 1.620$$
$$M/M/4\ Queue : \lambda = \rho * \mu * n \Rightarrow 0.8 * 3.439 * 4 = 0.930$$
$$E[T] = \frac{1}{\lambda} * p_Q * \frac{\rho}{1-\rho} + \frac{1}{\mu} = 11.005$$
$$E[N] = \lambda * E[T] = 5.584$$

V. Optimization Strategy

The models suggest learning rate is the most influential factor on the outcome of the experiments. However, analysis also showed that the models can not explain the results perfectly. To see the effects of altering the learning rate in practice, heuristic optimization is applied. The optimization employs exponential increase, linear back down. The learning rate starts at 0.002 and will increase exponentially by a factor of 2 each time the cost goes down of the performed experiment in comparison to the last experiment. If the cost stays the same or goes up, we return to the learning rate of the previous experiment and instead increase the learning rate by 10% of this learning rate as long as the cost decreases, otherwise terminate. The optimization uses approximately 5 replications to determine if a certain learning rate is more effective or not. During the optimization the following settings were used: $N_{nodes} = 1$, $N_{executors} = 1$, $N_{executor\_cores} = 1$, $batch\_size = 256$ and $N_{epochs} = 1$. The results of the experiment can be seen in figure 5. It should be noted that the cost function used in this graph equals $cost = \frac{time}{accuracy^2}$, which is functionally similar to the one used before but emphasizes the cost to be more like "normalized time" than just a cost function. The graph explains why the learning rate is such an impactful factor when it comes to determining the outcome of the cost function. Lower learning rates lead to low accuracy as the learner does not have enough time to learn much. Conversely, the learning rate can be increased at will to get better and better results. Other factors become irrelevant in the face of the learning rate because of this. This illustrates that the chosen levels of

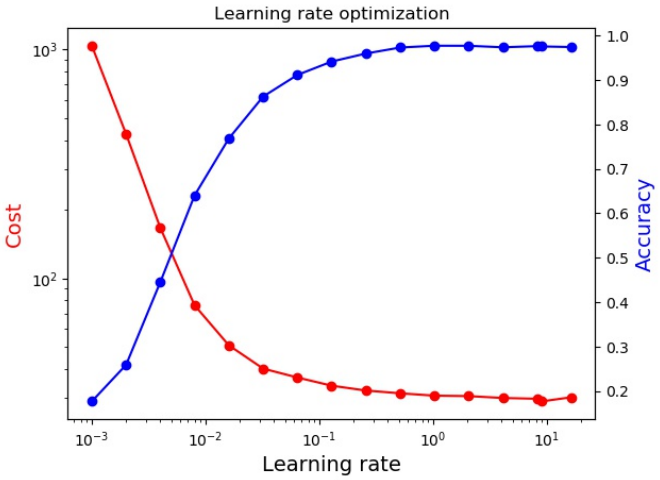

Fig. 5. Results of the optimization of the learning rate. The lowest cost function is obtained at $learning\ rate = 9.011$. Both axis are in log-scale.

N_{epochs} are too low. The learner will in both instances not be able to learn much in the given number of epochs and so simply benefits from an increased learning rate to get a somewhat decent accuracy. The lowest cost obtained by the optimization script is with a learning rate of 9.011, although it should be noted that this likely due to variance; increasing the learning rate will logarithmically lead to better and better results.

Finally, as the optimization of the learning rate leads to a lower cost, the response time of the system will also decrease. Previously, μ was estimated to be 3.085 given the results of the experiments. The μ of the optimized system, including the levels of the factors used for the optimization, is $\frac{1}{\mu} = \frac{29.5}{(0.976*100)^2} = 0.003$ so $\mu = \frac{1}{0.003} = 333.333$. Clearly, this μ is obtained under different circumstances than the non-optimized μ and should be taken with a grain of salt. However, even taking into consideration those differences, the improvement is substantial. The improved response time for a $M/M/1$ server that follows from this is shown below.

$$\lambda = \rho * \mu = 0.8 * 333.333 = 266.667$$
$$E[N] = \frac{\rho}{1-\rho} = \frac{0.8}{1-0.8} = 4$$
$$E[T] = \frac{E[N]}{\lambda} = \frac{4}{333.333} = 0.015$$

VI. Conclusion

To get the most out of a server cluster in the context of machine learning, a balance needs to be struck between the time it takes to train a learner and the accuracy it can achieve. We can see a better balance can be struck by finding and tuning the relevant parameters of a learning system. Using fractional factorial experiment designs, ANOVA modeling, queuing theory and heuristic optimization, these parameters can be highlighted and tuned accordingly. This research showed that even when applying basic models and making generous assumptions, meaningful results can be obtained. From our findings it can be stated that the expected response time of a $M/M/1$ server can be improved significantly if the learning rate is adjusted correctly.

As was already mentioned, while the results are promising, caution needs to be taken with regards to their validity. We therefore propose the following suggestions and future work recommendations that could improve the results of this research. First, more meaningful results could have been obtained if datasets were considered which are harder for learners to learn; where a decent accuracy could not be obtained in a small number of epochs, regardless of the learning rate. Also, a more nuanced cost function that penalizes lack of accuracy to a greater degree could lead to more relevant results. Finally, as was stated in section III-C, there is no guarantee the experiments were run on the exact same hardware, which could lead to inconsistencies in the results.

References

[1] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," 2016.

[2] A. Halevy, P. Norvig, and F. Pereira, "The unreasonable effectiveness of data," *IEEE Intelligent Systems*, vol. 24, pp. 8–12, 2009. [Online]. Available: http://www.computer.org/portal/cms_docs_intelligent/intelligent/homepage/2009/x2exp.pdf

[3] Y. You, Z. Zhang, C.-J. Hsieh, J. Demmel, and K. Keutzer, "Imagenet training in minutes," in *Proceedings of the 47th International Conference on Parallel Processing*, 2018, pp. 1–10.

[4] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing*, ser. HotCloud'10. USA: USENIX Association, 2010, p. 10.

[5] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," 2016.

[6] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, "Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems," 2015.

[7] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer, "A survey on distributed machine learning," 2019.

[8] B. Reagen, J. M. Hernández-Lobato, R. Adolf, M. Gelbart, P. Whatmough, G. Wei, and D. Brooks, "A case for efficient accelerator design space exploration via bayesian optimization," in *2017 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, 2017, pp. 1–6.

[9] S. C. Smithson, Guang Yang, W. J. Gross, and B. H. Meyer, "Neural networks designing neural networks: Multi-objective hyper-parameter optimization," in *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2016, pp. 1–8.

[10] Z. Chao, S. Shi, H. Gao, J. Luo, and H. Wang, "A gray-box performance model for apache spark," *Future Generation Computer Systems*, vol. 89, pp. 58 – 67, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X17323233

[11] M. Bilal and M. Canini, *Towards Automatic Parameter Tuning of Stream Processing Systems*. New York, NY, USA: Association for Computing Machinery, 2017, p. 189–200. [Online]. Available: https://doi.org/10.1145/3127479.3127492

[12] A. Fekry, L. Carata, T. Pasquier, A. Rice, and A. Hopper, "Tuneful: An online significance-aware configuration tuner for big data analytics," 2020.

[13] G. E. Box, W. H. Hunter, S. Hunter *et al.*, *Statistics for experimenters*. John Wiley and sons New York, 1978, vol. 664.

[14] F. Graybill and H. Iyer, *Regression Analysis: Concepts and Applications*, ser. An Alexander Kugushev book. Duxbury Press, 1994. [Online]. Available: https://books.google.nl/books?id=nAbvAAAAMAAJ