Convergence and model performance analysis using asynchronous distributed SGD

Arthur Lamour, Nicolas Reategui, Robin Junod
Department of Computer Science, EPFL Lausanne, Switzerland [GitHub]

***Abstract*—As large machine learning models are becoming part of our daily life, with some models composed of 175 billion parameters [1], the need for distributed training paradigms is crucial to achieving reasonable training times. There are different distributed architectures such as connected worker networks or Master-Slave servers which can be used for this purpose. However, implementing distributed training poses several challenges, including reduced accuracy and lack of convergence. In this study, we aim to investigate these challenges and their impact on model performance.**

I. Introduction

The optimization of deep learning models has been a fundamental research area in machine learning, with Stochastic Gradient Descent (SGD) being one of the most widely used optimization algorithms. In the context of centralized asynchronous SGD (aSGD) [2] multiple workers asynchronously compute and update the gradients to the parameter server based on their mini-batches of data. If workers wait for each other at each mini-batch update, this variant is called centralized synchronous SGD (sSGD), the gradients are aggregated before updating the global model parameters. This approach allows for parallel and distributed training, where workers can operate independently and asynchronously, communicating with the central server to exchange information and synchronize the model updates. The objective of this study is to analyze and present the benefits and challenges associated with utilizing such a computational method for stochastic gradient descent in neural networks. This study primarily addresses the following key research questions:

- How do varying delays impact the convergence of distributed aSGD?
- Does the partitioning of data among workers enhance convergence during the training process? Is it beneficial to distribute labels among workers as well?
- What is the influence of momentum in aSGD? Does it serve as a regularization factor?

II. Model and Method

The experiments made and the results presented have been computed using the simulation of a centralized aSGD algorithm that can be found on our public [GitHub]. This repository can generate a great range of experiments concerning aSGD, sSGD and non-distributed SGD, with hyperparameters that can be easily tuned to study their influences. Additionally, the repository includes implemented plotting scripts that facilitate the visualization of experiment results. These scripts enable the visualization of various aspects, such as the loss landscape [3], the temporal evolution of the loss, training and validation loss curves, accuracy metrics, computational speed, and more. One thing that must be noted is that the computational distribution among workers is performed on a single machine, utilizing the CPU resources.

A. Main model

Neural network: The main network used in this study is the LeNet5 [4]. LeNet5 is a well-known, yet simple, convolutional neural network that utilizes convolutions, pooling, and fully connected layers. It gained prominence for its successful performance on the MNIST dataset for handwritten digit recognition. Furthermore, its architectural design has been influential in inspiring subsequent networks like AlexNet and VGG [5].

Dataset: The FashionMNIST dataset [6] is a collection of 70'000 grayscale images in 10 categories. The images are of low resolution (28x28 pixels) and each represents a specific article of clothing or a fashion accessory. This dataset is widely used in the field of ML, often as a benchmark.

B. Experiments and method

Delays: We applied a constant delay $d \in [0, 30, 60, 90]$ ms either to all the workers or a single worker. Note that apart from the explicitly induced delay, workers might slow down due to other processes running on the CPUs. The last tables V, VI, VII were generated using a Gaussian distributed delay across all workers, i.e. during the simulation the workers would independently receive delays extracted from a Gaussian distribution.

Data partitioning: For both the aSGD and sSGD we applied either one of the following data partitioning strategies during training: 1) No partitioning 2) *Split Dataset* : Random distribution of the dataset among workers. 3) *Split Labels* Division of the dataset where specific labels are assigned to each worker.

Momentum: The momentum used in this study has been implemented with the parameters of the SGD optimizer from the Pytorch package. Its algorithm uses the Nesterov Accelerated Gradient (NAG), based on the formula from [7]. It presents a superior rate of convergence under certain conditions, specifically for smooth, not strongly convex functions. NAG can achieve a convergence rate of $O(1/T^2)$, which surpasses the $O(1/T)$ of standard gradient descent.

$$v_{t+1} = \mu v_t - \varepsilon \bigtriangledown f(\Theta_t + \mu v_t) \tag{1}$$

$$\Theta_{t+1} = \Theta_t + v_{t+1} \tag{2}$$

This method differs from the classical momentum method as the gradient is computed at $\Theta_t + \mu v_t$ and not at Θ_t.

Loss landscape visualization: Given that gradient descent operates in a dimension exceeding 60'000 (LeNet5), direct visualization is impractical. One effective approach to represent the loss landscape is by selecting two meaningful vectors to explore it [3]. While these vectors could be chosen at random, we have opted for a more systematic approach. We performed Principal Component Analysis (PCA) on the model parameters evolution during training, retaining the two most significant parameter vectors. We then built a 2D grid around the trajectory and used the inverse PCA to return to the model's multidimensional parameter space for which we computed the loss using the test set.

Delay compensation: In order to dampen the effect produced by introducing delays amongst the workers it is possible to modify the gradient update in order to compensate for it. In [8], researchers modified the gradient updated as $w_{t+1} \leftarrow w_t - \eta \cdot (g_m + \lambda_t \cdot g_m \cdot g_m(w_t - w_{\text{bak}}(m)))$, where $w_{\text{bak}}(m)$ is a worker-specific backup model saved by the central parameter server. The algorithm is described in full in 1, the results are present in IV

III. Results

A. Delay effect on aSGD accuracy

Table I
Classification report

Class n°	0	1	**2**	3	4	5	6	7	8	9
Precision %	99	99	**11**	80	20	93	24	97	99	78
Recall %	12	73	**2**	25	**100**	94	7	71	35	99
f1-score %	21	84	**3**	38	33	94	11	82	52	87

TABLE 1: Classification report performed on the test set showing the accuracies in % for each of the classes resulting from the training of 10 workers with 90 ms delay applied to the first worker. Note that the number of training points is the same for each of the classes.

B. Effect of momentum with aSGD

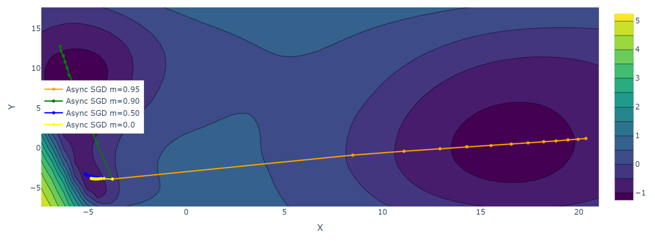

Figure 2. Top view of the figure 1

Table II
Comparison of test accuracies (in %) for sSGD and aSGD

Workers		2		5		10	
sSGD		87.5	88.7	86.1	87.5	84.8	85
aSGD	0 ms	89	85.5	88.5	76.4	84.6	85
	30 ms	88.97	82.04	88.54	75.25	86.92	63.5
	60 ms	89.27	87.82	88.57	75.51	**NC**	60.83
	90 ms	88.49	88.86	88.53	80.4	83.97	68.12
	30 ms	89.26	60.8	87.8	39.37	87.24	65.27
	60 ms	88.7	61.95	88.15	58.28	85.4	63.53
	90 ms	89.47	59.62	87.79	69.74	77.54	51.91

TABLE 2: Each worker column is divided in two, the first sub-column corresponds to textitSplit Dataset strategy and the second sub-column to textitSplit Labels strategy II-B. The first rows of delayed aSGDs results from applying a delay to all the workers. In the next ones, the delay is applied to only one worker. For all these experiments we use a momentum of $m = 0.9$. (NC = No Convergence)

Table III
Interplay between momentum and delays

2 workers			5 workers			10 workers		
0.0	0.5	0.99	0.0	0.5	0.99	0.0	0.5	0.99
53.58	54.1	37.47	54.85	52.54	**NC**	63.97	68.42	**NC**
55.45	56.43	34.89	33.38	38.79	**NC**	63.44	77.97	**NC**
44.75	54.91	34.03	30.92	51.55	**NC**	63.11	46.07	**NC**

TABLE : 3 For this set of experiments we used 3 momentums $m \in [0, 0.5, 0.99]$ and 3 delays $d \in [\ 30\ ,\ 60\ ,\ 90\ ]$ ms corresponding to each of the rows respectively. We slowed down only one of the workers and assigned specific labels to each of the workers. Test accuracies of 10 % result from non-convergence of the model, it outputs the same label regardless of the input.(NC = No Convergence)

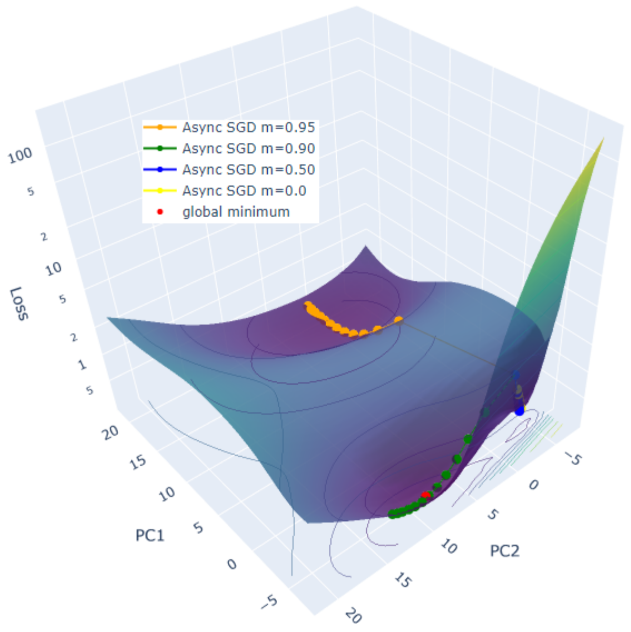

Figure 1. Momentum influence on convergence, loss landscape and convergence trajectories. The PCA was computed on all the trajectories points to determine the vectors PC1 and PC2.

C. *Delay compensation for aSGD*

Table IV
DELAY COMPENSATION

Workers	2		5		10	
λ	2	5	2	5	2	5
30 ms	59.74	56.16	59.42	50.37	64.0.2	**NC**
60 ms	56.99	58.66	43.02	74.12	71.21	67.5
90 ms	60.46	63.12	34.42	62.04	**NC**	53.58

TABLE: 4 Test accuracies for aSGD with different delays applied to a single worker using delay compensation. Here we applied for a strong compensation by setting $\lambda \in 2, 5$. In red are the values which showed an increase with respect to the experiment where only one worker was slowed down in II.(NC = No Convergence)

IV. DISCUSSION

sSGD and aSGD: As we can see on the control experiments shown in the first two rows of II, using sSGD, which ensures update synchronization, does not differ greatly from aSGD with no delays. As the experiments is performed on the CPU of a one computer, the sSGD and aSGD, with no artificial delay implemented, don't show significant differences. This suggests that the workers are operating at nearly equal speeds when no delay is implemented.

Data partitioning and Delays: Interestingly, the *Split Labels* has a negative impact on the accuracy whereas the *Split Dataset* enables some compensation as the data is uniformly distributed across the workers. Moreover, introducing a delay on only one worker instead of the entire cluster results in a decrease in accuracy. This can be attributed to the influence of outdated gradients from the "slow" worker, which negatively impact the training process.

We observe an average decrease in accuracy between label and dataset splitting of 27.8% with standard deviation (std) of 8%. In the experiments where only one worker is slowed down, we observe a decrease in accuracy of approximately 11% with a std of 6% compared to the scenario where all workers are slowed down.

This clearly highlights that the *Split Labels* strategy is not a viable option and demonstrates that slow workers have a negative impact on the training process.

It was also observed during training of more complex model 6 that increasing the number of workers in 0-delayed aSGD can lead to gradient vanishing whereas there is no significant change for sSGD. This might be explained by the regularization effect of the averaging performed after collecting all the updates from the workers in sSGD. The convergence could also be impacted by the initial conditions and non-explicit delays induced by concurrent processes in the CPU.

Delay compensation: Finally, IV shows that modifying the distributed framework in order to compensate for slowed-down workers can improve accuracy 7% and 3% overall for $\lambda = 2, 5$ respectively; 16% if considering only cases where accuracy improved. This proves that delay compensation is actually a feasible solution for the problems that arise with outdated gradients.

Momentum: The aSGD model was tested with five different momentums. Out of these, four momentums led to convergence, while the momentum of 0.99 caused divergence. The experiments were conducted using identical initial parameters for all model trainings. Notably, when a momentum of 0.5 was applied, the convergence was nearly identical to that without momentum. On the other hand, a momentum of 0.95 resulted in the model converging to a different local minimum. Interestingly, the momentum of 0.9 led to convergence at the most optimal local minima among all the experiments. We can see that the momentum has a big influence on the convergence in async SGD. It can explore a wider range of the loss landscape and potentially converge to a better local minimum. On the other hand, if its value is too large, the model will diverge. It doesn't serve directly as a regularization factor like L1 or L2[9]. As we can see on III, increasing the momentum when applying delays may impact the accuracy -although this seems not to be the case for all worker sizes- as well as leading to a decrease in the stability of the convergence shown by the random classification performed by clusters of sizes 5 and 10 for any delay. Furthermore, using a momentum of 0.5 instead of 0.0 seems to improve accuracy as it may prevent converging in local minima. Analyzing together III and II, we can see that using the momentum of 0.9 leads to optimal solutions for most delays and worker sizes. This value was actually obtained after running hyperparameter tuning on the classic non-distributed model.

V. SUMMARY

In this study, we were able to confirm that asynchronous SGD implementations can be subject to delay amongst workers which seriously deteriorates the model performance, especially in large clusters. Randomly assigning the data points to different workers instead of assigning specific classes helps overcome the decrease in accuracy. Furthermore, momentum helps escape local minima but has to be used with care in large worker clusters as this would exacerbate outdated gradient-related issues. It is possible to counter to a certain degree, the delay by adapting the update algorithm as in 1. Nevertheless, the λ variance parameter is another hyperparameter that could have to be adapted to the model architecture and data structure as well as the delay distribution.

ACKNOWLEDGEMENTS

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and et al. Language models are few-shot learners, Jul 2020. URL https://doi.org/10.48550/arXiv.2005.14165.

[2] Xiangru Lian, Wei Zhang, Ce Zhang, and Ji Liu. Asynchronous decentralized parallel stochastic gradient descent. In International Conference on Machine Learning, pages 3043–3052. PMLR, 2018.

[3] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. Advances in neural information processing systems, 31, 2018.

[4] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In Proceedings of the IEEE, volume 86, pages 2278–2324. IEEE, 1998.

[5] Mohammed Kayed, Ahmed Anter, and Hadeer Mohamed. Classification of garments from fashion mnist dataset using cnn lenet-5 architecture. In 2020 international conference on innovative trends in communication and computer engineering (ITCE), pages 238–243. IEEE, 2020.

[6] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747, 2017.

[7] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In International conference on machine learning, pages 1139–1147. PMLR, 2013.

[8] Shuxin Zheng, Qi Meng, Taifeng Wang, Wei Chen, Nenghai Yu, Zhi-Ming Ma, and Tie-Yan Liu. Asynchronous stochastic gradient descent with delay compensation, Feb 2020. URL https://arxiv.org/abs/1609.08326.

[9] Bohan Wang, Qi Meng, Huishuai Zhang, Ruoyu Sun, Wei Chen, Zhi-Ming Ma, and Tie-Yan Liu. Does momentum change the implicit regularization on separable data? Advances in Neural Information Processing Systems, 35:26764–26776, 2022.

Annex

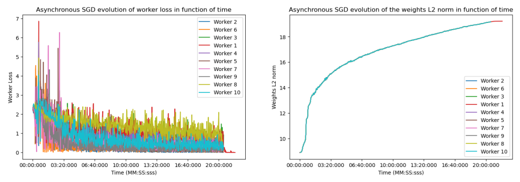

Figure 3. Loss and weights norm during training for aSGD with 90 ms delay applied to a single worker

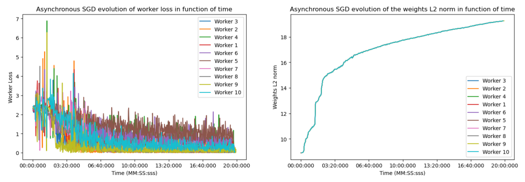

Figure 4. Loss and weights norm during training for aSGD with no delay

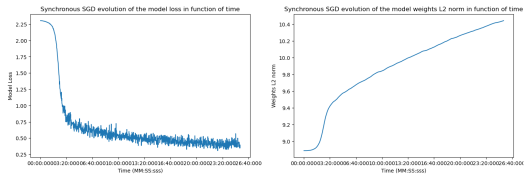

Figure 5. Loss and weights norm during training for sSGD

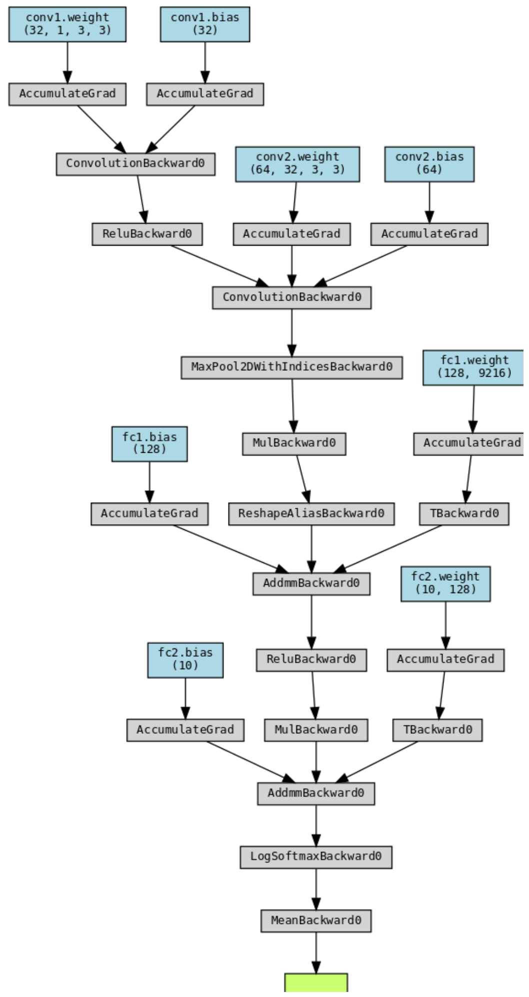

Figure 6. Large alternative model for Fashion MNIST classification

Algorithm 1 Variance Control Algorithm

```
1: **Input:** Learning rate $\eta$, variance control parameter $\lambda_t$
2: **Initialize:** $t = 0$, $w_0$ is initialized randomly, $w_{\text{bak}}(m) = w_0$, $m \in \{1, 2, \ldots, M\}$
3: **repeat**
4:     **if** receive "$g_m$" **then**
5:         $w_{t+1} \leftarrow w_t - \eta \cdot (g_m + \lambda_t \cdot g_m \cdot g_m \cdot (w_t - w_{\text{bak}}(m)))$
6:         $t \leftarrow t + 1$
7:     **else if** receive "pull request" **then**
8:         $w_{\text{bak}}(m) \leftarrow w_t$
9:         Send $w_t$ back to worker $m$.
10: **until** forever
```

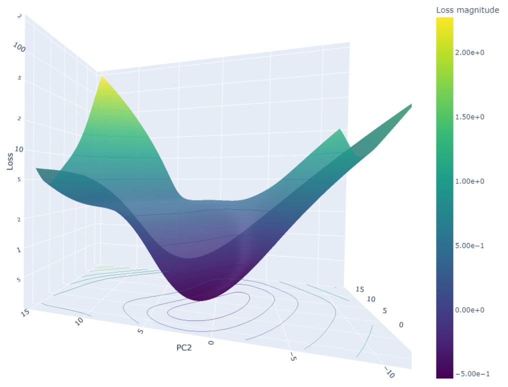

Figure 7. Loss Landscape of leNet5 model using Fashion MNIST

Table V
INTEPLAY BETWEEN MOMENTUM, GAUSSIAN DELAY, AND COMPENSATION

Workers	Momentum	Delay	Mode	λ	Test accuracy
11	00	30 ms	dataset	0	83.41
11	00	30 ms	dataset	2	84.66
11	00	30 ms	labels	0	64.08
11	00	30 ms	labels	2	65.67
11	05	30 ms	dataset	0	86.60
11	05	30 ms	dataset	2	86.74
11	05	30 ms	labels	0	76.08
11	05	30 ms	labels	2	71.36
11	09	30 ms	dataset	0	85.55
11	09	30 ms	dataset	2	86.89
11	09	30 ms	labels	0	10.00
11	09	30 ms	labels	2	63.77
11	099	30 ms	dataset	0	10.00
11	099	30 ms	dataset	2	10.00
11	099	30 ms	labels	0	10.00
11	099	30 ms	labels	2	10.00
11	00	60 ms	dataset	0	82.24
11	00	60 ms	dataset	2	83.02
11	00	60 ms	labels	0	69.48
11	00	60 ms	labels	2	63.50
11	05	60 ms	dataset	0	87.13
11	05	60 ms	dataset	2	86.22
11	05	60 ms	labels	0	73.90
11	05	60 ms	labels	2	76.76
11	09	60 ms	dataset	0	85.34
11	09	60 ms	dataset	2	50.07
11	09	60 ms	labels	0	10.00
11	09	60 ms	labels	2	63.40
11	099	60 ms	dataset	0	10.00
11	099	60 ms	dataset	2	10.00
11	099	60 ms	labels	0	10.00
11	099	60 ms	labels	2	10.00
11	00	90 ms	dataset	0	84.50
11	00	90 ms	dataset	2	84.53
11	00	90 ms	labels	0	58.17
11	00	90 ms	labels	2	69.41
11	05	90 ms	dataset	0	86.08
11	05	90 ms	dataset	2	85.66
11	05	90 ms	labels	0	78.50
11	05	90 ms	labels	2	80.94
11	09	90 ms	dataset	0	87.36
11	09	90 ms	dataset	2	10.00
11	09	90 ms	labels	0	69.12
11	09	90 ms	labels	2	69.27
11	099	90 ms	dataset	0	10.00
11	099	90 ms	dataset	2	10.00
11	099	90 ms	labels	0	10.00
11	099	90 ms	labels	2	10.00

Table VI
INTEPLAY BETWEEN MOMENTUM, GAUSSIAN DELAY, AND COMPENSATION

Workers	Momentum	Delay	Mode	λ	Test accuracy
3	00	30 ms	dataset	0	83.60
3	00	30 ms	dataset	2	84.14
3	00	30 ms	labels	0	83.13
3	00	30 ms	labels	2	80.80
3	05	30 ms	dataset	0	86.21
3	05	30 ms	dataset	2	86.85
3	05	30 ms	labels	0	86.66
3	05	30 ms	labels	2	81.89
3	09	30 ms	dataset	0	88.72
3	09	30 ms	dataset	2	89.51
3	09	30 ms	labels	0	88.46
3	09	30 ms	labels	2	84.77
3	099	30 ms	dataset	0	10.00
3	099	30 ms	dataset	2	10.00
3	099	30 ms	labels	0	56.49
3	099	30 ms	labels	2	58.84
3	00	60 ms	dataset	0	83.77
3	00	60 ms	dataset	2	84.33
3	00	60 ms	labels	0	81.01
3	00	60 ms	labels	2	81.14
3	05	60 ms	dataset	0	86.48
3	05	60 ms	dataset	2	86.28
3	05	60 ms	labels	0	79.62
3	05	60 ms	labels	2	83.66
3	09	60 ms	dataset	0	89.25
3	09	60 ms	dataset	2	88.56
3	09	60 ms	labels	0	89.00
3	09	60 ms	labels	2	88.77
3	099	60 ms	dataset	0	10.00
3	099	60 ms	dataset	2	10.00
3	099	60 ms	labels	0	10.00
3	099	60 ms	labels	2	52.68
3	00	90 ms	dataset	0	85.14
3	00	90 ms	dataset	2	83.41
3	00	90 ms	labels	0	83.23
3	00	90 ms	labels	2	85.71
3	05	90 ms	dataset	0	86.79
3	05	90 ms	dataset	2	86.42
3	05	90 ms	labels	0	87.47
3	05	90 ms	labels	2	84.51
3	09	90 ms	dataset	0	88.12
3	09	90 ms	dataset	2	88.73
3	09	90 ms	labels	0	88.56
3	09	90 ms	labels	2	85.46
3	099	90 ms	dataset	0	58.83
3	099	90 ms	dataset	2	54.58
3	099	90 ms	labels	0	53.47
3	099	90 ms	labels	2	10.00

Table VII
INTEPLAY BETWEEN MOMENTUM, GAUSSIAN DELAY, AND COMPENSATION

Workers	Momentum	Delay	Mode	λ	Test accuracy
6	00	30 ms	dataset	0	83.78
6	00	30 ms	dataset	2	84.69
6	00	30 ms	labels	0	71.94
6	00	30 ms	labels	2	55.61
6	05	30 ms	dataset	0	86.57
6	05	30 ms	dataset	2	86.83
6	05	30 ms	labels	0	71.01
6	05	30 ms	labels	2	75.00
6	09	30 ms	dataset	0	87.97
6	09	30 ms	dataset	2	88.46
6	09	30 ms	labels	0	71.08
6	09	30 ms	labels	2	62.71
6	099	30 ms	dataset	0	10.00
6	099	30 ms	dataset	2	10.00
6	099	30 ms	labels	0	10.00
6	099	30 ms	labels	2	10.00
6	00	60 ms	dataset	0	84.39
6	00	60 ms	dataset	2	85.23
6	00	60 ms	labels	0	72.17
6	00	60 ms	labels	2	71.55
6	05	60 ms	dataset	0	87.79
6	05	60 ms	dataset	2	86.42
6	05	60 ms	labels	0	78.46
6	05	60 ms	labels	2	69.98
6	09	60 ms	dataset	0	88.35
6	09	60 ms	dataset	2	87.85
6	09	60 ms	labels	0	71.39
6	09	60 ms	labels	2	64.45
6	099	60 ms	dataset	0	10.00
6	099	60 ms	dataset	2	10.00
6	099	60 ms	labels	0	10.00
6	099	60 ms	labels	2	10.00
6	00	90 ms	dataset	0	83.71
6	00	90 ms	dataset	2	84.78
6	00	90 ms	labels	0	71.14
6	00	90 ms	labels	2	74.58
6	05	90 ms	dataset	0	86.36
6	05	90 ms	dataset	2	87.47
6	05	90 ms	labels	0	78.37
6	05	90 ms	labels	2	77.00
6	09	90 ms	dataset	0	88.43
6	09	90 ms	dataset	2	88.50
6	09	90 ms	labels	0	75.74
6	09	90 ms	labels	2	67.78
6	099	90 ms	dataset	0	10.00
6	099	90 ms	dataset	2	10.00
6	099	90 ms	labels	0	10.00
6	099	90 ms	labels	2	10.00