UDC – 004.725:004.852

RESEARCH OF MODEL FOR INCREASE LEARNING SPEED OF A NEURAL NETWORK IN THE LINUX OPERATING SYSTEM

Timur V. Jamgharyan
National Polytechnic University of Armenia
105, Teryan St, 0009, Yerevan
e-mail: t.jamharyan@polytechnic.am
ORCID ID: 0000-0002-9661-1468
Republic of Armenia

https://doi.org/10.56243/18294898-2025.2-99

Abstract

The paper presents the effect of changing the duration of the quantum of processor time on the efficiency of neural network training in the Linux environment. The speed of training of a convolutional neural network is considered with different parameters of the computing cluster. The obtained results demonstrate the possibilities of optimizing neural network training in a multitasking environment, increasing the overall computational efficiency.

A detailed analysis of the relationship between resource allocation strategies and training speed is carried out, and recommendations for configuring Linux systems for working with neural networks are proposed.

Keywords: batching, completely fair scheduler, mathplotlib, hyperparameter, convolutional neural network, failover cluster, operating environment, processor time quantum.

Introduction

The training neural networks on computing clusters without the use of GPU (Graphics Processing Unit), an important condition for increasing the speed of training is the choice of the operating system (OS) and its configuration. One of the configuration parameters is the size of the quantum of processor time. In Linux OS, this parameter is controlled by the task scheduler, which supports various algorithms (O(1) scheduler, CFS, EEVDF)[1] [1] and resource management mechanisms such as *cgroups*[2] [2]. Optimizing the size of a quantum of processor time is especially relevant in multitasking computing environments, for example, in distributed

[1] O (1) scheduler - a kernel scheduling algorithm that can schedule processes for a constant period of time, regardless of how many processes are running in the operating system.
CFS (Completely Fair Scheduler) - a kernel scheduling algorithm that uses a red-black binary search tree based on the waiting time for a single process to execute.
EEVDF (Earliest eligible virtual deadline first) - a priority distribution algorithm for limited real-time systems.

[2] cgroups - is an OS kernel mechanism that allows you to limit the use, keep track of, and isolate the consumption of system resources (processor, memory, disk I/O, network, etc.) at the process collection level.

training of artificial neural networks. By default, most modern Linux distributions, such as Ubuntu, Debian, CentOS, openSUSE, Red Hat, and Fedora, use the CFS scheduler, which dynamically manages time slices based on process priorities (some Linux distributions with kernel versions higher than 6.6 may also use the EEVDF algorithm).

CFS does not have a fixed CPU time slice value, but instead calculates it based on various factors, including system load and CPU (Central Processing Unit) task priorities. In early versions of the Linux kernel, the CPU time slice was fixed at 100 or 120 milliseconds, which is comparable to server versions of Windows [3]. CPU time slice values for various Linux distributions are presented in Tab. 1.

Table 1

The value of the quantum of processor time in different Linux distributions

OS	Scheduler	Quantum of processor time (ms)		Note
		Server OS version	Client OS version	
Ubuntu	CFS	10-100	4-10	Use *cgroups* for priority management
Debian	CFS	10-100	4-10	Similar to Ubuntu
openSUSE	CFS/EEVDF	10-100 4-10	6-15	Optimized for server and desktop solutions
Red Hat	CFS/EEVDF	10-100 4-10	6-15	Use *cgroups* and *rtprio* to manage priorities
Fedora	CFS	10-100 6-10	10-12	It is possible to change the planning mechanism

Linux OS is the main platform for deploying software for training artificial neural networks [4], an urgent task is to determine both the OS parameters and the optimal value of the processor time quantum to increase the speed of training/reaction in parallel computing.

A separate task is the task of synchronizing the CFS of the cluster nodes, since different loads on the OS nodes generate different CFS values of a single node from the cluster, which ultimately reduces the overall performance of the computing cluster. Machine learning (ML) researchers offer various solutions to the problem of choosing the optimal value of the processor time quantum [5-10], each of which has certain limitations.

However, the method of accelerating the training of neural networks by centralized change of the processor time quantum and synchronous batching of the code during instrumental tuning of the Linux OS has not been considered. The closest research to this one is [10], however, this study was conducted on GPU. Also the research [10] was not carried instrumental tuning of the OS out due to the proprietary nature of the source code.The novelty of the research lies in the centralized, synchronous for all nodes change of the quantum of processor time and batching of the code, with the instrumental adjustment of the OS.

Conflict Setting

It is necessary to determine the optimal value of the processor time quantum for Linux OS when training a convolutional neural network.

Discussion

Synchronously for all cluster nodes, by changing the value of the processor time quantum, determine the most effective value for the learning speed.

Boundary condition

- training was carried out only on the basis of CPU,
- at a given point in time, only one neural network is trained,
- Ubuntu Server OS was used as the operating environment for the study. Other Linux OS distributions were not considered,
- the script for changing the quantum of processor time was installed on all nodes of the cluster and synchronized with the quorum node.

Experimental procedures

The researchs used a computing cluster[3] consisting of 12+1 nodes united in a network (Fig. 1) running Ubuntu Server 22.04 OS [11]. TensorFlow [12] was used as a ML framework. Computing resources were managed by centralized changes to:

- machine learning library process priorities,
- system timer (processor time quantum) of cluster nodes,
- the number of CPU cores of cluster nodes involved in computations.

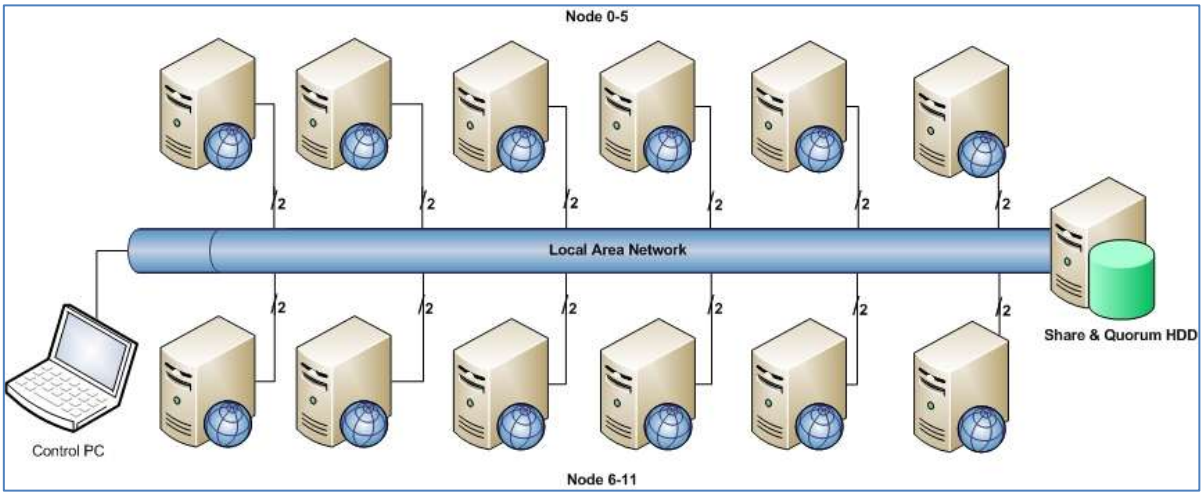

Fig. 1 Diagram of the computing cluster

The change of the quantum of processor time was performed using a script. Fig. 2-9 shows the configuration values of the Ubuntu Server 22.04 OS for different operating modes of a single node.

The following parameters were subject to adjustment:

[3] Single computer (node) parameters: CPU-i9, RAM 16Gb

• the number of CPU cores. For training, 6 physical CPU cores out of 10 on each node were used (Fig. 2),

• the values of the system timer (Fig. 3-6). The value of the quantum of processor time in this version (build) of the Ubuntu 22.04 OS is related to the value of the system timer in the ratio N/10 (N is the number of ticks of the system timer). The change of the value of the system timer was verified by the getconf CLK_TCK command (Fig. 7),

• the execution priority PPID (Parent Process IDentificator) of the Tensor Flow software (Fig. 8, 9) [13, 14].

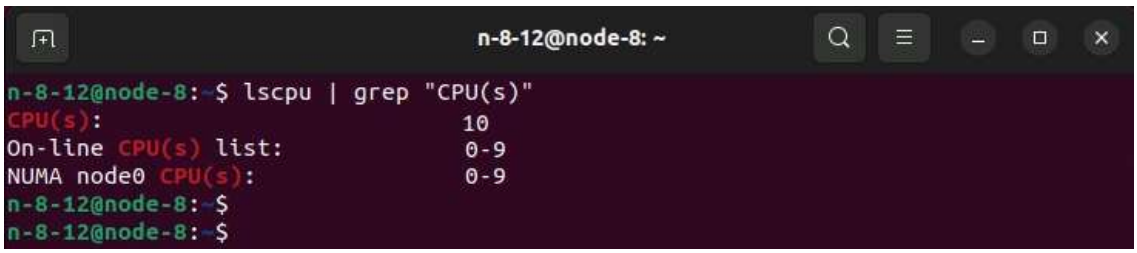

Fig. 2 Checking available CPU cores on a single node

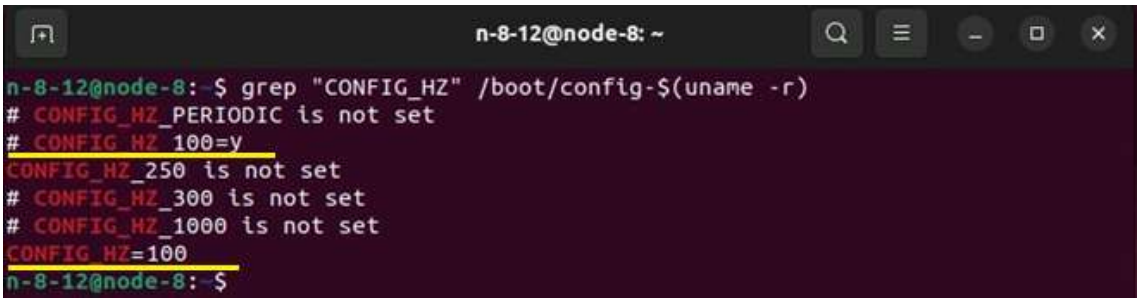

Fig. 3 System timer value 10 ms (default value)

Fig. 4 System timer value 25 ms

Fig. 5 System timer value 30 ms

Fig. 6 System timer value 100 ms

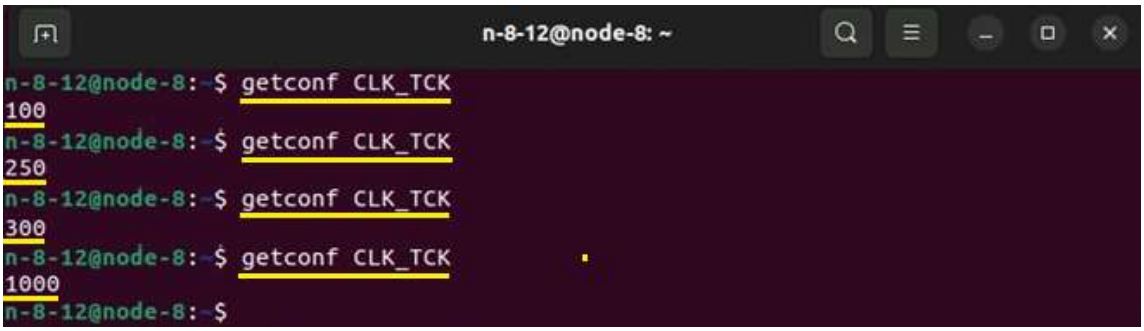

Fig. 7 Verifying the change of system timer values

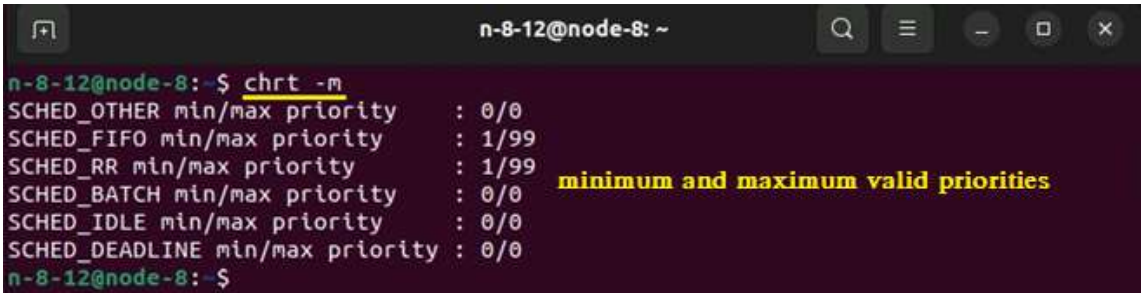

Fig. 8 Process execution priority thresholds

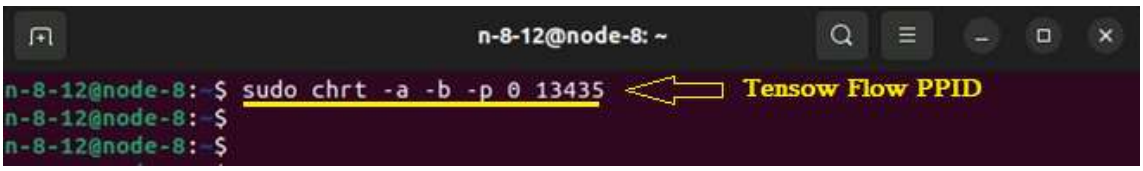

Fig. 9 Changing the execution priority of the Tensor Flow parent process

Training was carried out in two stages:

- training without synchronizing the quantum of processor time in the cluster nodes, with the batching change script disabled and without additional OS configuration (Tab. 2). Increasing the priority of the Tensor Flow process, fixing the number of CPU cores, and dynamically changing the system timer in the nodes were not performed,
- training with synchronized values of the quantum of processor time (10, 25, 30, 100 ms) in the cluster nodes, with the batching change script enabled and the OS configured for the training task (Tab. 3).

Research Results

The research results of the change in the learning rate with different values of the

quantum of processor time are presented in Tab. 2-3.

Table 2

Learning rate of neural networks without synchronization of the quantum of processor time and the disabled script for changing batching

Ubuntu Server 22.04	Value of quantum of processor time (ms). Based on CFS algorithm	Convolutional Neural Network Learning Rate Increment (%) / Accuracy (%)			The value of the quantum of processor time (ms). Fixed values	Convolutional Neural Network Learning Rate Increment (%) / Accuracy (%)		
		Number of layers				Number of layers		
		3	5	7		3	5	7
		0.77/97	0.61/94	0.38/93	100	0.62/96	0.7/92	0.4/97
		0.96/94	0.74/90	0.54/88	30	0.74/98	1.0/92	0.8/93
		1.57/871	1.12/84	0.8/81	25	0.81/90	1.6/88	0.82/87
		1.74/85	0.41/80	0.24/88	10	1.3/95	1.9/90	1.2/87

Table 3

Neural network training speed with synchronized values of the processor time quantum and the enabled batching change script

Ubuntu Server 22.04	The value of the quantum of processor time (ms). Fixed values	Convolutional Neural Network Learning Rate Increment (%) / Accuracy (%)		
		Number of layers		
		3	5	7
	100	2.1/96	2.2/93	1.1/91
	30	2.8/98	1.0/98	1.2/92
	25	3.87/95	2.82/95	1.8/95
	10	2.5/92	2.1/92	1.4/90

The graphs of the dependence of the learning speed and reliability on the quantum of processor time for different numbers of layers of the neural network are shown in Fig. 10, 11. The reliability of the output values of the trained neural network was checked by comparing it with the a priori known output values of the neural network trained in the study [15] (reference network). The comparison was carried out using the *ssdeep* software [16].

Based on the conducted research, the following conclusions can be made:

- optimal balance is achieved with a fixed quantum of 10-30 ms,
- the best balance of speed and accuracy is observed with a quantum of 25-30 ms,
- CFS quantum provides more stable operation, but with less acceleration,
- a fixed value of the processor time quantum gives a higher speed increase.

T.V. Jamgharyan

RESEARCH OF MODEL FOR INCREASE LEARNING SPEED OF A NEURAL NETWORK IN THE LINUX OPERATING SYSTEM

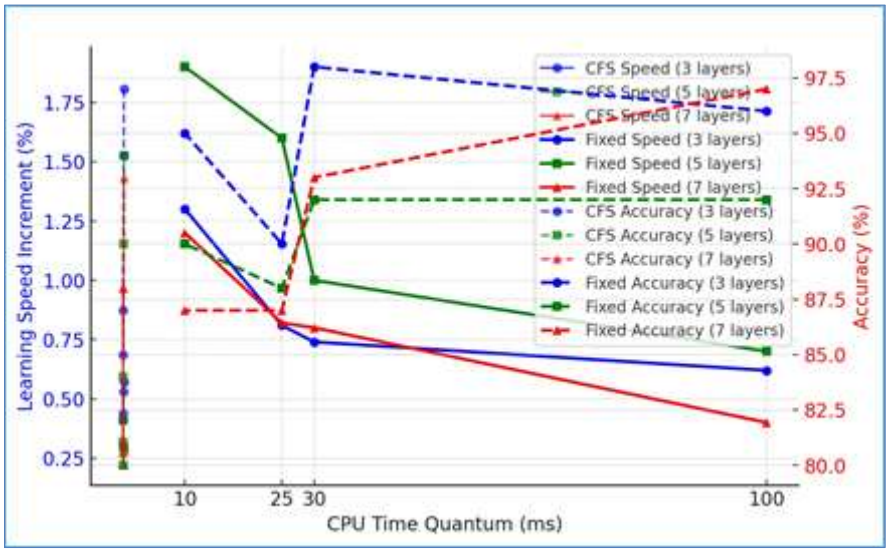

Fig. 10 Learning speed using CFS

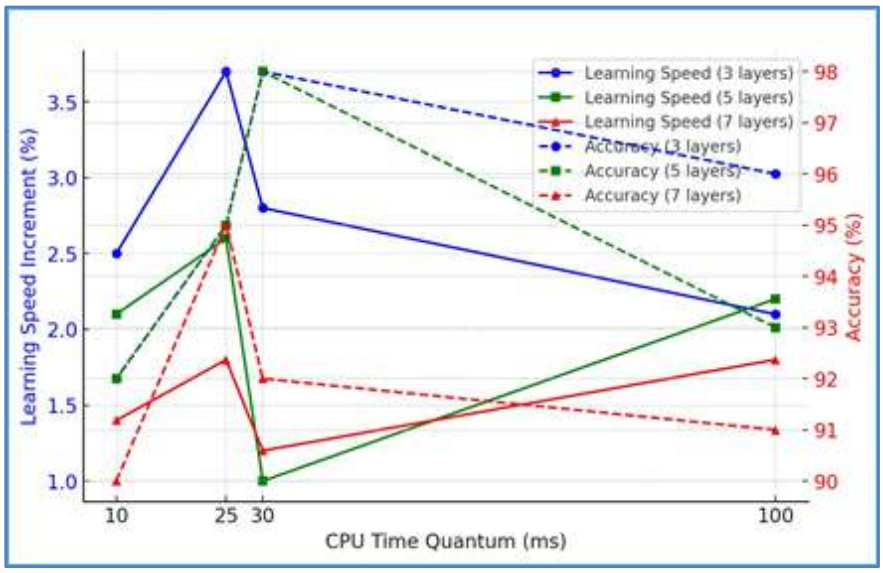

Fig. 11 Learning speed for a fixed value of the quantum of processor time

Conclusion

The research considers a model for increasing the training speed of a convolutional neural network based on a hardware cluster running under Linux OS. It is determined that setting a fixed value of the processor time quantum and instrumental adjustment of other parameters (increasing the priority of the neural network process, adjusting OS parameters, centralized synchronization, using batching) allows increasing both the reliability and the training speed of a convolutional neural network by an average of (2.5÷3)% relative to the standard CFS scheduler, depending on the network depth.

With an increase in the number of neural network layers, in order to reduce fluctuations in the time of interprocessor exchange in the cluster, it is necessary to increase the processor

time quantum, which reduces the speed and/or reliability. Using cgroups to control CPU resources allows only a partial solution to this problem under multi-threaded load. In all cases, even with limited resources, it is possible to achieve acceleration of neural network training.

References

1. Both David, Using and Administering Linux:Volume 1: Zero to SysAdmin:Getting Started, Second Edition, APRESS, 2023, pp. 667
2. Both David, Using and Administering Linux:Volume 3, Second Edition, APRESS, 2023, pp. 536
3. Russinovich M., Yosifofovich P., et al., Windows Internals, Part 1: System architecture, processes, threads, memory management, and more, 7th Edition., The Microsoft Press, 2018, pp. 945
4. Goodfellow Y., Bengo Y., Courville A., Deep Leaning, MIT Press, 2017, pp. 800
5. Gao H., et al, «IFQ-Net: Integrated Fixed-point Quantization Networks for Embedded Vision». https://doi.org/10.48550/arXiv.1911.08076
6. Bertsimas D., et al, «Deep Trees for (Un)structured Data: Tractability, Performance, and Interpetability». https://doi.org/10.48550/arXiv.2410.21595
7. Wang Z., et al, «Dumpy OS: A Data-Adaptive Multi-ary Index for Scalable Data Series Similarity Search». https://doi.org/10.48550/arXiv.2501.08293
8. Cheng C., et al, «Dynamic Optimization of Storage Systems Using Reinforcement Learning Techniques». https://doi.org/10.48550/arXiv.2501.00068
9. Breuel T., «The Effects of Hyperparameters on SGD Training of Neural Networks». https://doi.org/10.48550/arXiv.1508.02788
10. Ryu M., Byeon G., Kim K., «A GPU-Accelerated Distributed Algorithm for Optimal Power Flow in Distribution Systems». https://doi.org/10.48550/arXiv.2501.08293
11. Ubuntu operating systems official download page https://ubuntu.com/download/server , the resource is available on 17.03.2025
12. Tensor Flow software download official page. https://www.tensorflow.org/install?hl=ru , the resource is available on 17.03.2025
13. Tevault D., Mastering Linux Security and Hardening, Packt, 2023, pp. 619.
14. Nemeth E., Snyder G., Hein T., Whaley B., Unix and Linux Administration Handbook, Prentice Hall, 2012, pp. 1303
15. Jamgharyan T., Iskandaryan V., Khemchyan A., (2024). Obfuscated Malware Detection Model. *Mathematical Problems of Computer Science*, *62*, 72–81. https://doi.org/10.51408/1963-0122
16. ssdeep software download official page, https://ssdeep-project.github.io/ssdeep/index.html , the resource is available on 17.03.2025

References

1. Both David, Using and Administering Linux:Volume 1: Zero to SysAdmin:Getting Started, Second Edition, APRESS, 2023, pp. 667
2. Both David, Using and Administering Linux:Volume 3, Second Edition, APRESS, 2023, pp. 536

3. Russinovich M., Yosifofovich P., et al., Windows Internals, Part 1: System architecture, processes, threads, memory management, and more, 7th Edition., The Microsoft Press, 2018, pp. 945
4. Goodfellow Y., Bengo Y., Courville A., Deep Leaning, MIT Press, 2017, pp. 800
5. Gao H., et al, «IFQ-Net: Integrated Fixed-point Quantization Networks for Embedded Vision». https://doi.org/10.48550/arXiv.1911.08076
6. Bertsimas D., et al, «Deep Trees for (Un)structured Data: Tractability, Performance, and Interpetability». https://doi.org/10.48550/arXiv.2410.21595
7. Wang Z., et al, «Dumpy OS: A Data-Adaptive Multi-ary Index for Scalable Data Series Similarity Search». https://doi.org/10.48550/arXiv.2501.08293
8. Cheng C., et al, «Dynamic Optimization of Storage Systems Using Reinforcement Learning Techniques». https://doi.org/10.48550/arXiv.2501.00068
9. Breuel T., «The Effects of Hyperparameters on SGD Training of Neural Networks». https://doi.org/10.48550/arXiv.1508.02788
10. Ryu M., Byeon G., Kim K., «A GPU-Accelerated Distributed Algorithm for Optimal Power Flow in Distribution Systems». https://doi.org/10.48550/arXiv.2501.08293
11. Официальная страница загрузки операционной системы Ubuntu, https://ubuntu.com/download/server , the resource is available on 17.03.2025
12. Официальная страница ПО Tensor Flow. https://www.tensorflow.org/install?hl=ru , the resource is available on 17.03.2025
13. Tevault D., Mastering Linux Security and Hardening, Packt, 2023, pp. 619.
14. Nemeth E., Snyder G., Hein T., Whaley B., Unix and Linux Administration Handbook, Prentice Hall, 2012, pp. 1303
15. Jamgharyan T., Iskandaryan V., Khemchyan A., (2024). Obfuscated Malware Detection Model. *Mathematical Problems of Computer Science*, *62*, 72–81. https://doi.org/10.51408/1963-0122
16. Официальная страница загрузки ПО ssdeep, https://ssdeep-project.github.io/ssdeep/index.html , the resource is available on 17.03.2025

ՆԵՅՐՈՆԱՅԻՆ ՑԱՆՑԻ ՈՒՍՈՒՑՄԱՆ ԱՐԱԳՈՒԹՅԱՆ ԲԱՐՁՐԱՑՄԱՆ ՄՈԴԵԼԻ ՀԵՏԱԶՈՏՈՒԹՅՈՒՆ LINUX ՕՊԵՐԱՑԻՈՆ ՀԱՄԱԿԱՐԳՈՒՄ

Թ.Վ. Ջամղարյան

Հայաստանի ազգային պոլիտեխնիկական համալսարան

Հոդվածում ներկայացված է պրոցեսորի ժամանակի քվանտի տևողության փոփոխման ազդեցությունը Linux միջավայրում փաթույթային նեյրոնային ցանցի ուսուցման արագության ու ճշգրտության վրա: Ստացված արդյունքները ցույց են տալիս Linux օպերացիոն համակարգի միջավայրում փաթույթային նեյրոնային ցանցի ուսուցման լավարկման հնարավորությունը՝ բարձրացնելով ընդհանուր հաշվարկային արդյունավետությունը:

T.V. Jamgharyan

RESEARCH OF MODEL FOR INCREASE LEARNING SPEED OF A NEURAL NETWORK IN THE LINUX OPERATING SYSTEM

Կատարվել է ռեսուրսների բաշխման ռազմավարությունների և ուսուցման արագության միջև կապի մանրամասն վերլուծությունը, որը թույլ տալիս ստանալ ավելի ճշգրիտ արդյունքները նեյրոնային ցանցերը ուսուցանելու ժամանակ:

Բանալի բառեր. հիպերպարամետր, օպերացիոն միջավայր, պրոցեսորի ժամանակի քվանտ, փաթույթային նեյրոնային ցանց, batching, completely fair scheduler, mathplotlib, failover cluster.

ИССЛЕДОВАНИЕ МОДЕЛИ ПОВЫШЕНИЯ СКОРОСТИ ОБУЧЕНИЯ НЕЙРОННОЙ СЕТИ В ОПЕРАЦИОННОЙ СИСТЕМЕ LINUX

Т.В. Джамгарян
Национальный политехнический университет Армении

В статье исследуется влияние изменения длительности кванта процессорного времени на эффективность обучения искусственных нейронных сетей в среде ОС Linux. Рассматривается скорость обучения свёрточной нейронной сети при различных параметрах вычислительного кластера. Полученные результаты демонстрируют возможности оптимизации обучения искусственных нейронных сетей в многозадачной среде, повышая общую вычислительную эффективность.

Проведен детальный анализ взаимосвязи между стратегиями распределения ресурсов и скоростью обучения, а также предложены рекомендации по настройке ОС Linux систем для работы с искусственными нейронными сетями.

Ключевые слова: квант процессорного времени, свёрточная нейронная сеть, batching, completely fair scheduler, mathplotlib, failover cluster.

Submitted on 14.11.2024
Sent for review on 20.11.2024
Guaranteed for printing on 26.07.2025