# Computer Architecture Execution Time Optimization Using Swarm in Machine Learning

**Sarah AlBarakati[†], Sally AlQarni[†], Rehab K. Qarout*[†] and Kaouther Laabidi[††],**

[†]Department of Computer Science & Artificial Intelligence, University of Jeddah
[††]Department of Computer Engineering and Networks, University of Jeddah

**Summary**
Computer architecture serves as a link between application requirements and underlying technology capabilities such as technical, mathematical, medical, and business applications' computational and storage demands are constantly increasing. Machine learning these days grown and used in many fields and it performed better than traditional computing in applications that need to be implemented by using mathematical algorithms. A mathematical algorithm requires more extensive and quicker calculations, higher computer architecture specification, and takes longer execution time. Therefore, there is a need to improve the use of computer hardware such as CPU, memory, etc. optimization has a main role to reduce the execution time and improve the utilization of computer recourses. And for the importance of execution time in implementing machine learning supervised module linear regression, in this paper we focus on optimizing machine learning algorithms, for this purpose we write a (Diabetes prediction program) and applying on it a Practical Swarm Optimization (PSO) to reduce the execution time and improve the utilization of computer resources. Finally, a massive improvement in execution time were observed.

***Keywords:*** *computer architecture, Machine learning, optimization, execution time, PSO, medical.*

## 1. Introduction

Machine learning (ML) recently grown at a special rate, attracting many researchers and practitioners. It has become one of the most popular study areas, applying it in many fields like machine translation, speech recognition, image recognition, recommendation systems, and others [5]. A ML model is trained to generate predictions or decisions without explicit programming in each application by detecting embedded patterns or relationships in the data. Particularly in tasks/applications where relationships are too complicated to analyze using analytical methods, ML models can perform well [1].

Recently machine learning models, most of which are deep neural networks (DNNs) and their variants (e.g., multi-layer perceptron, convolutional neural networks, and recurrent neural networks) already have high memory and computational resource requirements. Since people are searching for better artificial intelligence, there is a trend toward larger, more expressive, and more complicated models. With Moore's Law's gains declining, this trend encourages advancements in computer architecture/systems, enabling faster and more energy-efficient ML model implementations Improvements at various levels of system and architecture designs [2].

Optimization is also regarded as one of the crucial components of machine learning. Most machine learning methods work by constructing an optimization model and learning the parameters in the objective function from the input data. The effectiveness and efficiency of numerical optimization algorithms significantly impact the popularization and implementation of machine- learning models in the age of massive data.[5] Traditionally, architectural and system optimizations are performed to accelerate the execution and enhance the performance of ML models, it evident that advances in processing capabilities, such as better use of parallelism, data reuse, and sparsity, play a role in ML revolutions.

There has recently been evidence of using machine learning to improve system designs, revealing attractive potential [2]. While computer architects have been using GPUs and custom hardware to accelerate the performance of machine learning algorithms, there have been few implementations utilizing these algorithms to improve computer system performance. However, the work that has been done has yielded very encouraging results [3].
Machine learning's growing popularity, and the desire to conduct more extensive and quicker calculations, have prompted the development of hardware accelerators that can compete with GPUs while using significantly less energy, particularly for deep convolution networks (CNNs). There has been limited research using machine learning algorithms to improve computer performance since computer architects have focused intensely on specialized hardware for machine learning [3].

In this paper we implement a Machine learning algorithm named as (Diabetes prediction program) in python programming language and it is a linear regression supervised learning model. We implemented the program, and the execution time was calculated, then the optimization algorithm Practical swarm optimization (PSO) has been merged into the same code, also the execution time was calculated. By comparing the execution time in two cases, the result is a huge optimization in reducing the execution

Manuscript received October 5, 2023
Manuscript revised October 20, 2023
https://doi.org/10.22937/IJCSNS.2023.23.10.7

time while maintaining the same accuracy of the prediction result.

## 2. BACKGROUND

A. Computer Architecture

The design of computer systems, including all subsystems such as the CPU, memory, and I/O systems, is known as computer architecture. these components are crucial to the entire system's operation and performance [6].

B. Computer system performance

A computer system's performance relies on both the architecture and its implementation. It was evident that both the architecture and implementation must be optimized [6].

C. Machine Learning

Machine learning is the study of feeding data and information to computers without explicitly programming them to learn and act like humans [7]. Figure 1 is a schematic that shows the machine learning process. Machine learning has been used to solve problems in different industry domains, including banking and financial services, insurance, healthcare, and life sciences. Before beginning to solve any problem using machine learning, it is necessary to determine whether the problem is appropriate to use [7].

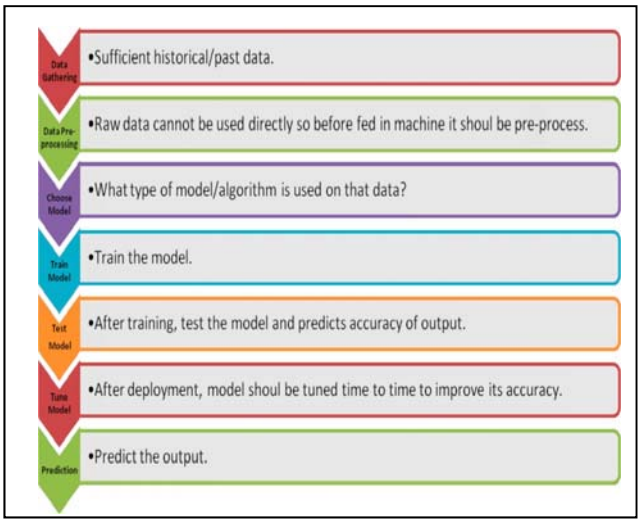


Figure 1: Machine learning process [7]

D. Machine Learning Methods

Machine learning has two prominent model families:

1) Deep Learning
2) Classical Machine Learning

a. Deep learning (DL) refers to a class of machine learning (ML) models made up of artificial neural networks (ANNs). ANNs were inspired by the structure and function of the human brain. Using various layers, they learn a hierarchy of parametric features (e.g., fully connected, ReLU). A technique known as back-propagation is used to train all parameters. Training a deep learning model incurs massive costs: they typically need many GPUs for reasonable runtime, massive, labelled datasets, and complex hyper-parameter tuning.

b. Classical Machine Learning. ML model families such as generalized linear models, decision tree models, and Bayesian models are widely used in many applications that work with structured data. These model families are commonly referred to collectively as classical machine learning.

We also identify three different learning paradigms for the two families of the ML model above, as shown in figure 2:

- Supervised Learning.
- Unsupervised Learning.
- Reinforcement Learning.

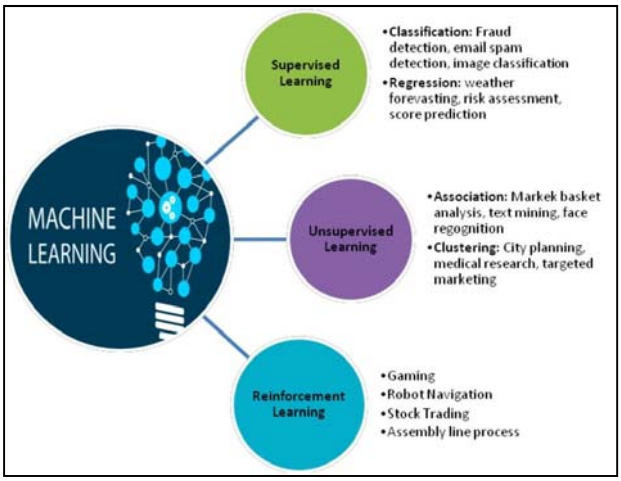


Figure 2: Types of machine learning [7]

The machine learning model that we used in our experiment in this paper is Supervised learning, and one of the important types of supervised learning is a linear regression, as explained below:

- Supervised Learning. In supervised learning, training data consists of inputs (also known as features) and output pairs. The goal of the ML model is to learn a predictive function that takes unseen inputs and predicts the output value to minimize the discrepancy between the predicted and actual values corresponding to the unseen inputs.

1. Linear regression:

Linear regression is One of the most basic and widely used supervised machine learning methods. It is a method for performing predictive analysis that is based on mathematics. Linear regression allows for projections of continuous/real or mathematical variables. It takes one or more independent variables and predicts the values of dependent variables.

- Unsupervised Learning. In unsupervised learning, training data contains only inputs, not explicit outputs. The goal of the ML model is to learn a function that can identify the latent structure of an input. Given an unseen test input, the use of the trained unsupervised ML model can be seen to predict the structural characteristics of the input.
- Reinforcement Learning. The purpose of reinforcement learning methods is to learn a function that takes input states and produces actions that maximize the overall cumulative reward (not the immediate rewards as supervised learning).

E. Optimization and Machine Learning

The heart of machine learning is optimization. Most machine learning issues reduce optimization problems. Consider a machine learning analyst working to solve an issue using some set of data. The modeler formulates the problem by choosing an appropriate model family and manipulating the data into a modelling-friendly format. The model is then trained by solving a core optimization problem that optimizes the model's variables or parameters related to the chosen loss function and perhaps some regularization function. The fundamental optimization problem may be solved several times during the model selection and validation process. Through these important optimization problems, the research area of mathematical programming intersects with machine learning. On the one hand, mathematical programming theory defines what makes a solution optimal – the optimality conditions. On the other hand, machine learning researchers can use mathematical programming algorithms to train large families of models [8].

From the point of view of machine learning, and optimization algorithm should have the following properties [8]:

1. Excellent generalization.
2. Scalability to large-scale issues.
3. In terms of execution time and memory requirements, it must be a good performance in practice.
4. Algorithm implementation should be simple and easy.
5. Exploitation of the structure of the problem
6. Rapid convergence to a model's approximate solution.
7. For the class of machine learning models' robustness and numerical stability were attempted.
8. Convergence and complexity are theoretically known.

a. Particle Swarm Optimization:

Kennedy proposed swarm modeling to simulate the social behavior of fish and birds, and Kennedy and Eberhart presented the optimization algorithm as an optimization technique in 1995. PSO has particles that represent candidate solutions to the problem, each particle searches for the best solution in the search space, and each particle or candidate solution has a position and velocity. Based on its inertia, own experience, and knowledge obtained from other particles in the swarm, a particle adjusts its velocity and position to discover the best solution to the problem [13].

The particles update its position and velocity according to the following Equation, this equation is a built-in Pyswarm library [13]:

$$v_i^{k+1} = wv_i^k + c_1 rand_2 \times (pbest_i - s_i^k) + (gbest_i - s_i^k)$$

Where:

| | | |
|---|---|---|
| $v_i^{k+1}$ | = | Velocity of agent i at iteration k, |
| w | = | Weighting function, |
| cj | = | Weighting factor, |
| rand | = | Random number between 0 and 1, |
| $S_i^k$ | = | Current position of agent iteration k, |
| pbesti | = | Pbest of agent i, |
| gbesti | = | gbest of the group. |

The weighting function used in Equation 1:

$$w = w_{max} - \frac{w_{max} - w_{min}}{iter_{max}} \times iter$$

Where:

| | |
|---|---|
| Wmax= | Initial weight, |
| Wmin= | Final weight, |
| itermax= | Maximum iteration number, |
| Iter = | Current iteration number. |

Figure 3. PSO Equation [13]

## 3. RELATED WORK

Daniel et al. [3], described a method for highlighting when, why, and how to use machine learning models to improve system performance, A relevant example demonstrates the ability of machine learning-based cross core IPC predictors to enable CPU schedulers to optimize system throughput. They described a data generation process for every execution quantum and parameter engineering. They have evaluated a group of popular machine learning models, including stochastic gradient descent based linear regression, decision trees, random

forests, artificial neural networks, and k-nearest neighbours. The inner workings of the algorithms, computational and memory complexities, and a process to fine-tune and evaluate the models were then discussed. The random forest narrowly produces the lowest root mean squared error in its testing predictions after comparing the results of the predictors. Finally, they discussed how a mechanism such as a scheduler for heterogeneous systems might use the predictor to improve overall system performance. In the future, reinforcement learning could be a worthwhile alternative to explore when applying machine learning to improve scheduling.

Lizhong et al. [1] reviewed machine learning in many individual components, including memory systems, branch predictors, networks-on-chip, GPUs, system-wide simulation, and runtime optimization. They analyzed current practice to highlight useful design strategies and determine opportunities for future work based on optimized implementation strategies, appropriate expansions to existing work, and ambitious long-term potential. These strategies and techniques, when combined, promise a bright future for increasingly automated architectural design. They concluded that model optimization opportunities such as pruning, and quantization provide comprehensive benefits by enabling more practical implementation. Similarly, there are several chances to expand existing work using ever-more-powerful machine learning models, enabling finer granularity and system-wide implementation. Finally, ML can be applied to learn hierarchical or abstract representations to characterize complete system behaviour based on both high- and low-level details, enabling it to be applied to entirely new aspects of architecture. These extensive opportunities, and yet to be imagined possibilities, may someday close the loop on highly (or even completely) automated architectural design.

NAN WU and YUAN XIE. [2] presented a comprehensive review of work that uses machine learning for system design, divided into two categories: ML-based modelling, which includes predicting performance metrics or other criteria of interest, and ML-based design methodology, which uses machine learning directly as a design tool. They discussed existing studies based on their goal level of the system, which ranged from the circuit level to the architecture/system level, for ML-based modelling. They used a bottom-up approach to review current work for ML-based design methodology, with a focus on (micro-)architecture design (memory, branch prediction, NoC), coordination between architecture/system and workload (data center management, and security, resource allocation and management), compiler, and design automation. They provided a future vision of prospects and directions for applying machine learning to computer architecture and systems, which might lead to a brighter and more promising future.

H. Krishnaveni et al. [9] presented the importance of enhancing resource utilization and improvise the overall performance of advanced cloud computing applications by efficient task scheduling. This scheduling is critical for achieving a high- performance schedule in a heterogeneous-computing system. Existing scheduling methods such as Min-Min, Suffrage, and Enhanced Min-Min were solely concerned with minimizing the makespan and ignored other factors such as resource utilization and load balancing. The Execution Time Based Suffrage Method (ETSA) is an efficient algorithm that considers the parameters makespan and resource consumption while scheduling activities. It has been written in Java using the Eclipse IDE, and a set of ETC matrices are used in testing to assess the suggested approach. The ETSA outperforms previous algorithms in terms of timeliness and resource utilization.

Artem M et al. [10] highlighted the complexity and stochastic features of the process components, and their runtime, and a solution is proposed that takes these factors into account. The proposed method in this research addresses issues at various levels, from a task to a process, error measurement and the theory behind the estimation algorithm. The suggested makespan estimation technique may be readily implemented as a standalone module for a wide range of schedulers. Combining task estimates into the overall workflow makespan used a dual stochastic representation, characteristic/distribution function.They also proposed workflow reductions, which are operations on a workflow graph that do not reduce the accuracy of the estimates but simplify the graph structure, thereby improving the algorithm's performance. Another noteworthy aspect of their work is that they include the described estimation schema into an earlier established scheduling algorithm, GAHEFT, and use the CLAVIRE platform to empirically evaluate the enhanced solution's performance in the natural environment. In all circumstances, the suggested GAHEFT and eGAHEFT algorithms outperformed the MinMin algorithm by 30–48 percent, according to the results of the experimental study. Both estimation approaches were used to analyze all the obtained outcome schedules. The eGAHEFT performed worse than GAHEFT, according to quantile estimation. The EDF estimate approach and real-world testing, on the other hand, revealed that the eGAHEFT solution was superior to the GAHEFT solution. Thus, simply by employing a better method for making span estimate, the efficiency of the meta-heuristic algorithm was raised by 6.4 percent without any changes to the meta-heuristic algorithm itself.

Shiliang et al. [11] introduced and summarized commonly used optimization methods from a machine learning perspective and their applications in many machine learning domains. They initially discussed the theoretical foundations of optimization methods from first-order, high-order, and derivative-free perspectives and recent research advance. Then, in the supplemental material, they discussed

applying optimization algorithms in various machine-learning scenarios and how to improve their performance. Finally, they talked about some obstacles and unsolved issues in machine-learning optimization methods.

Tanash, M et al. [12] designed a supervised machine learning model and implemented it into the Slurm resource manager simulator to calculate the number of memory resources (Memory) needed and it's time to complete the computation. Different machine learning algorithms are used in their model. Their objective is to use Slurm to integrate and test the suggested supervised machine learning model. They evaluated the performance and accuracy of their integrated model by using over 10000 tasks from our HPC log files. Their work aims to improve Slurm's performance by predicting the number of required jobs memory resources and the time required for each job to optimize the HPC system's utilization using their integrated supervised machine learning model. Their results indicate that their model reduces computational response times (from five days to ten hours for significant works), significantly increases HPC system utilization, and decreases the average waiting time for submitted jobs for larger jobs.

## 4. METHODOLOGY

Machine learning is widely utilized in different fields to address complex issues that are difficult to solve with traditional computer methods. The goal of machine learning is to gain knowledge from data. There have been numerous experiments on how to make robots learn without being explicitly programmed. Many mathematicians and programmers use a variety of ways to solve these problems, which involves large data sets.

This led to increasing the execution time and the need to optimize the performance. Testing, as a part of the software engineering process, usually requires approximately 40-50 percent of the development efforts in software firms ]16[. It is critical for any software to assess its quality and ability to satisfy requirements, which may be accomplished through testing this software [13] .

Also, software community aims to deliver high quality software to customers, to ensure that the software will run perfect with no delays in execution time, as this is the aim of this research is to calculate the execution time of the program "Diabetes prediction" which is Linear regression Machine learning software written in python programming language. Then make an optimization to a Linear Regression Diabetes program, this could be achieved by using the Particle Swarm Optimization (PSO) techniques, as each population in each iteration searches for best execution time through particles in this population, and finally compares the best solution to produce the best execution time. We chose this optimization method because when compared to other methods, PSO has been shown to produce superior results in a faster and less expensive manner. It's also possible to parallelize it. Furthermore, it does not consider the gradient of the problem to be solved. To put it another way, unlike traditional optimization methods, PSO does not require a differentiable problem. Also, when compared to mathematical algorithms and other heuristic optimization techniques, the PSO algorithm has the following advantages: simple concept, simple implementation, robustness to control parameters, and computational efficiency.

The computer specifications that were used to implement the two programs, as shown in figure 4 and 5.

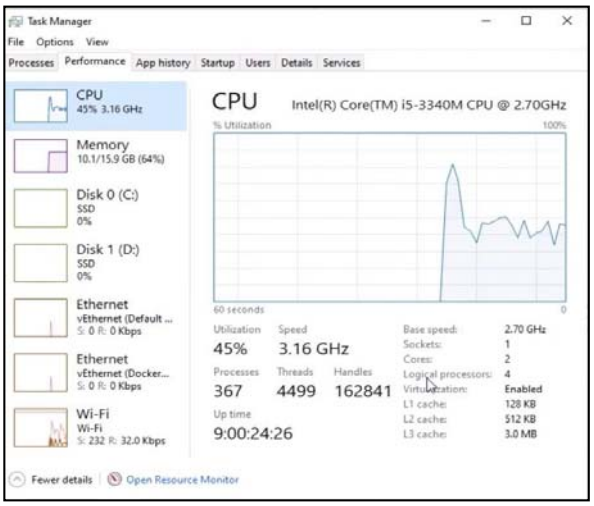


Figure 4. CPU specifications

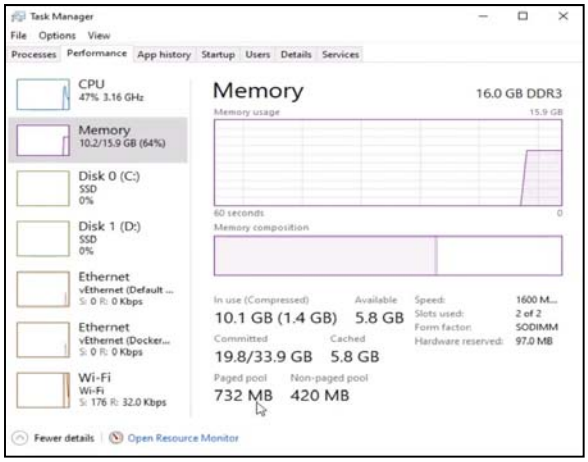


Figure 5. Memory specifications

Diabetes prediction program are described below in the following part:

Diabetes prediction Program using linear regression:
Is a Linear Regression code written in Python by using Anaconda Jypter toolkit. It predicts if a patient has diabetes or not from their multiple risk factor (features) of DM such as (HeartDiseaseorAttack,BMI,HighChol,..). We used five libraries in this program: Numpy and Pandas to work with a dataset to predict Diabetes and Sklearn for implementing machine learning functions and PSO for optimizing the machine learning code and finally the time library for measuring the execution time of the code and we use a diabetesdataset [14].

A. Diabetes dataset
It contains 22 columns which represent the risk factor (features) of DM. while the first column Diabetes_binary represents if a patient has diabetes, then its value =0 if not its value=1. and the number of rows that represent the number of patients
are 70,693 rows while the first row is to identify the name of columns, as shown in the figure 6 below.

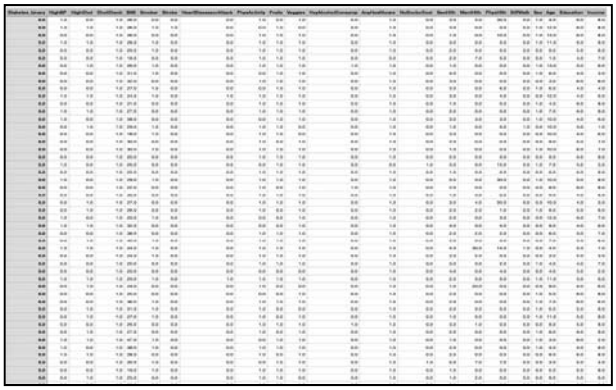

Figure 6 dataset diabetes

B. Implementation:

Step 1: Implementing the Diabetes program and we named it (LinearRegressionDiabetes.ipynb) and the program as explained before is to predict if a patient has diabetes or not based on some features.And measuring the execution time for the program Before Optimization. and the program executed in 0.103 seconds and the compute performance accuracy is 75 %, as shown in the figure 7,8 below.

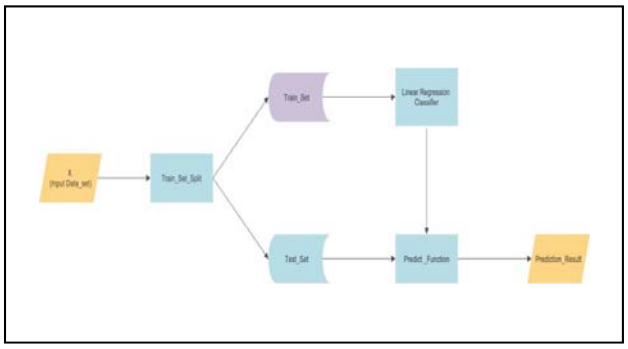


Figure 7 . Workflow Diagram for Diabets Predection Program without PSO

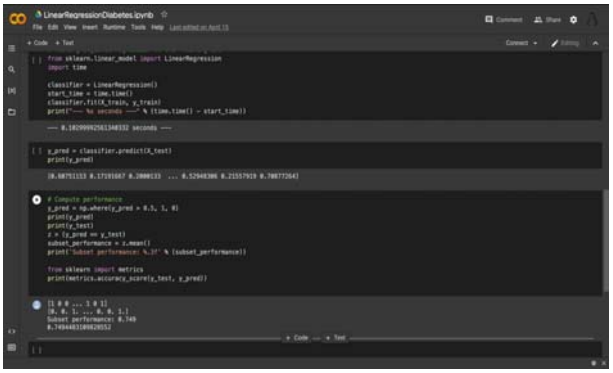

Figure 8 . execution time and accuracy of Diabetes program before using optimazation method

Step 2: applying PSO method [15] on the same code from step 1 and we named it (LinearRegressionDiabetesPSO.ipynb) and it is for optimizing the execution time for linear regression code in step 1 .and the program executed in 0.033 seconds and the compute performance has given the same accuracy of the step 1 code is 75 %, as shown in the figure 9,10 below.

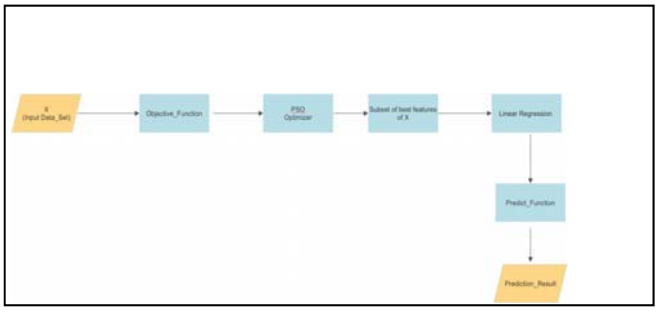


Figure.9 Workflow Diagram for Diabets Predection Program with PSO

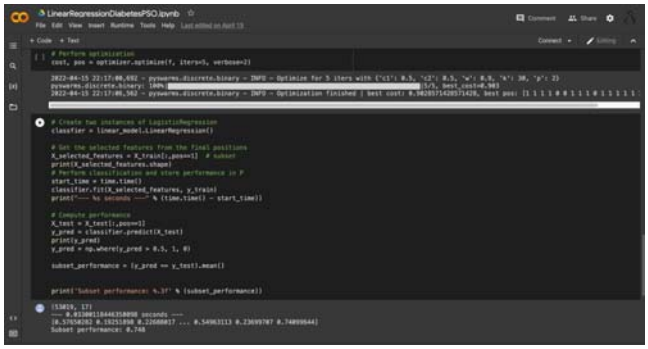

Figure 10. execution time and accuracy of Diabetes program after using optimization method

Step 3: comparing the results from step 1 and step 2 and finding the optimal execution time. The execution time is optimized by approximately 89% after using Particle Swarm Optimization.

Our proposed scheme shows the research paper general work frame and the results of the experiment, as illustrated below figure.11.

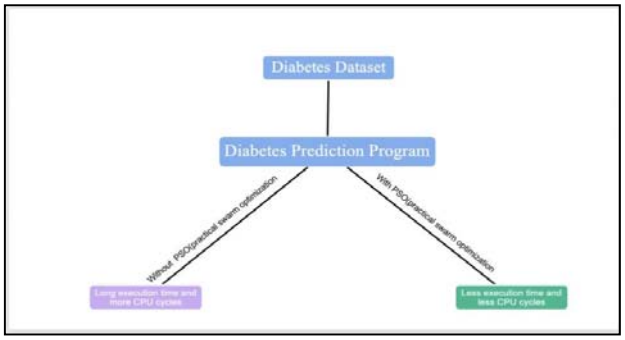


Figure 11. The proposed model.

## 5. RESULT

This paper presented the results of LinearRegressionDiabetes, LinearRegressionDiabetesPSO. In LinearRegressionDiabetesPSO, the results show a massive decreasing in execution time by 89% while using the same computing resources the summarization of the result is as Illustrated in table I

Table I: Comparsion Between Two Programs In Execution Time And Accuracy

| Program | Execution time | Accuracy |
|---|---|---|
| **LinearRegressionDiabetes names as (Diabetes)** | 103 milliseconds | 75% |
| **LinearRegressionDiabetesPSOnnames as (DiabetesPSO)** | 33 milliseconds | 75% |

Program Execution time Accuracy
LinearRegressionDiabetes names as (Diabetes) 103 milliseconds 75% LinearRegressionDiabetesPSOnnames as (DiabetesPSO) 33 milliseconds 75%

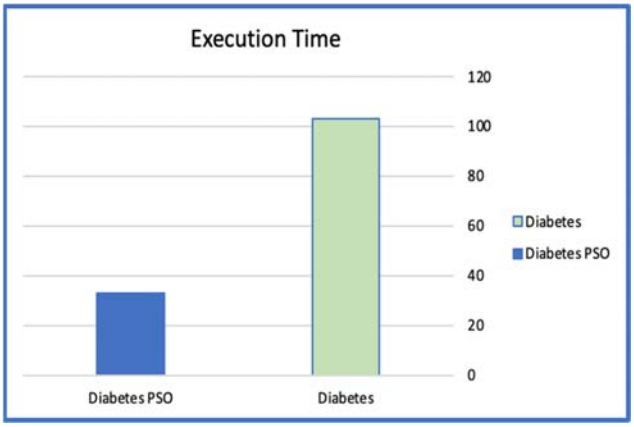


Figure 12. The diffrence of Execution time in both programs

## 6. CONCLUSION

Computer architectures describe how a computer's hardware components are connected, as well as the data transfer and processing methods used. Different computer architecture configurations have been developed to increase data processing by speeding up data movement. The research presents the application of PSO and its impact in reducing the execution time of machine learning method based on a huge dataset.

The proposed research described the basic concepts of Machine learning and PSO and their impact on the computer architecture and its performance, by calculating execution time for software modules when using PSO and without using it in our proposed (diabetes prediction

program) and it showed how it was useful in finding the optimal solution to the problem. then Comparative study is done between both the algorithms where PSO can be useful, this paper shows that PSO algorithm is more efficient in speed and time compared with same algorithm before applying it when the execution time was calculated.

## References


[1] Penney, D. D., & Chen, L. (2019). A survey of machine learning applied to computer architecture design. arXiv preprint arXiv:1909.12373.

[2] Wu, N., & Xie, Y. (2022). A survey of machine learning for computer architecture and systems. ACM Computing Surveys (CSUR), 55(3), 1-39.

[3] Nemirovsky, D., Arkose, T., Markovic, N., Nemirovsky, M., Unsal, O., Cristal, A., & Valero, M. (2018). A general guide to applying machine learning to computer architecture. Supercomputing Frontiers and Innovations, 5(1), 95-115.

[4] N, S. T. (2020R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.

[5] S. Sun, Z. Cao, H. Zhu and J. Zhao, "A Survey of Optimization Methods From a Machine Learning Perspective," in IEEE Transactions on Cybernetics, vol. 50, no. 8, pp. 3668-3681, Aug. 2020, doi: 10.1109/TCYB.2019.2950779.

[6] Dumas II, J. D. (2018). Computer architecture: Fundamentals and principles of computer design. CRC Press.

[7] Himani Maheshwari, Pooja Goswami, Isha Rana, (2019). A Comparative Study of Different Machine Learning Tools. International Journal of Computer Sciences and Engineering, 7(4), 184-190.

[8] Bennett, K. P., & Parrado-Hernández, E. (2006). The interplay of optimization and machine learning research. The Journal of Machine Learning Research, 7, 1265-1281.

[9] algorithm for static task scheduling in cloud. In Advances in Big Data and Cloud Computing (pp. 61-70). Springer, Singapore.

[10] Chirkin, A. M., Belloum, A. S., Kovalchuk, S. V., Makkes, M. X., Melnik, M. A., Visheratin, A. A., & Nasonov, D. A. (2017). Execution time estimation for workflow scheduling. Future generation computer systems, 75, 376-387.

[11] Sun, S., Cao, Z., Zhu, H., & Zhao, J. (2019). A survey of optimization methods from a machine learning perspective. IEEE transactions on cybernetics, 50(8), 3668-3681.

[12] Tanash, M., Dunn, B., Andresen, D., Hsu, W., Yang, H., & Okanlawon, A. (2019). Improving HPC system performance by predicting job resources via supervised machine learning. In Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (learning) (pp. 1-8).

[13] Darwish, N. R., Mohamed, A. A., & Zohdy, B. S. (2016). Applying swarm optimization techniques to calculate execution time for software modules. IJARAI, 5(3), 12-17.

[14] https://www.kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset

[15] https://pyswarms.readthedocs.io/en/development/examples/feature_subset_selection.html

[16] Jovanovic and Irena,”Software TestingMethods and Techniques,” May26, 2008.