Unleashing the Power of Alternative Training Methods

Öykü Eren
ITU AI & Data Eng. Sophomore
150200326
Istanbul, Turkey
ereno20@itu.edu.tr

Miraç Buğra Özkan
ITU AI & Data Eng. Sophomore
150200337
Istanbul, Turkey
ozkanm20@itu.edu.tr

***Abstract*—The performance and application of several algorithms are the main topics of this report's investigation into optimization techniques for neural networks. We examine the widely used gradient-based optimization techniques of stochastic gradient descent and batch gradient descent. In addition, we look at gradient-free methods like the genetic algorithm, particle swarm optimization, and simulated annealing. We examine the benefits and drawbacks of each approach, stressing both. We also investigate the advantages of mixing gradient-free and gradient-based techniques for neural network optimization. The research paves the ground for additional developments in neural network optimization by offering insights into the strengths and weaknesses of these methods.**

I. Introduction

With the rapid development of artificial intelligence technologies in recent years, we have been using machine learning methods in many areas in prediction processes or optimization processes. A neural network is a machine learning algorithm that mimics the structure and working mechanism of the human brain. With the layers of neurons that make up its structure, it learns by processing information and contributes to decision-making processes.

Neural network algorithms run with many different data types and give successful results in applications such as image processing, natural language processing, and speech recognition. The reason why the model is able to run with many data types is that the model processes the data as mathematical vectors. We will work on the classification of images in our study. Additionally, the model trains itself by trying to minimize the error between the mathematically predicted values and the actual values.

There are different methods used when searching for the best-predicted value. These methods look for weight which provides to predict a value close to the actual value and to minimize error. If we categorize these methods, two stand out: gradient-based and gradient-free methods.

In the Gradient-based algorithm, the weights in each next iteration are calculated using the gradient of the weights in the previous iteration. This method has some advantages and disadvantages. It converges very quickly and accurately, especially when given a start close to the target. However, if the target is started from a remote point, it can easily get stuck in local minimums. Because the derivative of the local minimum and global minimum are equal to zero and each other, and this method tries to find the point where the derivative equals to zero. Also, the derivative calculation is needed at each step.

Gradient-free methods, on the other hand, are the methods that select the weights in the next iterations with a specified logic, without making gradient calculations. Compared to gradient-based methods, these methods has a higher chance of evaluating different options. However, to find the most successful weights, each time it compares the error of the current weights with the error of the previous most successful weights, and this leads to time-consuming. In addition, in the next iteration, the options that will be considered while selecting weight may be changed depending on the success criteria of the current weights. There are different gradient-free methods designed on different strategies such as Genetic Algorithm, Simulated Annealing and Particle Swarm Optimization. We worked with Simulated Annealing, Particle Swarm Algorithms, and Genetic Algorithm in our study.

Gradient-based and gradient-free methods have different positive and negative features. Different methods can be preferred depending on the cases on hand. In addition, the advantages of both methods can be benefited at the same time by using the combination of both methods. First of all, if the gradient-based method is not close enough to the actual weights, we can avoid the probability of the gradient-based method converging to the local minimum by performing a more comprehensive search with the gradient-free method. Furthermore, we can prevent the gradient-free method is confused while searching a wide range thanks to gradient-based methods that can reach the result faster without deviating.

II. Literature Survey

Gradient Descent (GD) and Stochastic Gradient Descent (SGD) are searched gradient-based optimization methods for training neural networks. GD updates the model parameters by computing the gradients of the loss function with respect

to the parameters and moving in the direction of steepest descent. SGD performs parameter updates using a randomly selected subset (mini-batch) of training examples, which offers computational efficiency. Bottou et. al. outline a thorough theory of a simple yet adaptable SGD algorithm, go over its real-world behavior, and identify areas where building algorithms with better performance is possible[1]. These methods have been the foundation for training neural networks and serve as the baseline for evaluating the effectiveness of alternative optimization approaches.

Particle Swarm Optimization (PSO), Simulated Annealing (SA), and Genetic Algorithm (GA) are examples of gradient-free optimization methods that have been studied to training neural networks. PSO, proposed by Kennedy and Eberhart [2], is inspired by the collective behavior of bird flocks or fish schools and uses a swarm of particles to explore the parameter space. SA, introduced by Kirkpatrick et al. [3], simulates the annealing process in metallurgy, gradually cooling the system to escape local optima. GA, developed by Holland [5], is a population-based optimization algorithm inspired by the principles of natural selection and genetics. These gradient-free methods offer different strategies for exploring the parameter space and have shown promising results in improving neural network optimization.

Several studies have investigated the combination of gradient-based and gradient-free optimization methods for neural network training. Li et al. conducted a comprehensive survey on hybrid optimization algorithms and discussed the advantages and challenges of integrating gradient-free methods with gradient-based methods [4]. They explored the combination of PSO or GA with GD or SGD, highlighting the potential benefits in terms of improved convergence and robustness. Other researchers have also investigated hybrid approaches that combine SA with gradient-based methods or used a combination of genetic operators with GD or SGD [7, 8]. These studies have demonstrated the potential of combining both types of optimization methods to enhance neural network training.

III. Methods and Formal Methods Formulations

A. Gradient Descent Method

Gradient Descent is an optimization algorithm used to find the minimum of a function by iteratively updating the parameters. It calculates the gradient (derivative) of the cost function with respect to the parameters and takes steps in the direction of steepest descent. The key steps involved in Gradient Descent are as follows:

1) Compute the gradient: The algorithm calculates the gradient of the cost function with respect to each parameter or weight in the model. This gradient represents the direction of steepest ascent.

2) Update the parameters: The parameters are updated by subtracting the product of the learning rate (α) and the gradient. The learning rate determines the step size in each iteration. A smaller learning rate results in slower convergence, while a larger learning rate can cause overshooting and instability.

3) Repeat until convergence or iteration limit: The algorithm iteratively performs the above steps until it reaches a convergence criterion, such as the maximum number of iterations or a sufficiently small gradient magnitude.

B. Stochastic Gradient Descent Method

Stochastic Gradient Descent is a variant of Gradient Descent that computes the gradient and updates the parameters using a single training example at each iteration, rather than using the entire training dataset. This method is often faster but introduces more noise into the parameter updates. Stochastic Gradient Descent has some advantages over other Gradient methods, for example computationally, it is more efficient to update the parameters based on one training example at a time, especially when dealing with large datasets. Also, the stochastic nature of SGD introduces more noise into the parameter updates, which can help escape shallow local minima and improve the generalization ability of the model. However, SGD also has some challenges like noisy updates due to the high variance in the gradient estimation due to the use of a single training example can lead to oscillations during the training process.

C. Batch Gradient Descent

Batch Gradient Descent is another variant of Gradient Descent that computes the gradient and updates the parameters using the entire training dataset at each iteration. It provides a more accurate estimation of the gradient but can be slower when working with large datasets. Batch Gradient Descent also has some advantages over other Gradient methods, by considering the entire training dataset, the gradient estimation is more accurate compared to SGD. This can lead to more stable convergence. Batch Gradient Descent is more likely to converge to the global optima of the cost function, especially for convex optimization problems. However, it may get trapped in shallow local optima or saddle points for non-convex problems. Also, computing the gradient over the entire dataset can be computationally expensive, especially for large datasets. This makes Batch Gradient Descent slower compared to other Gradient methods.

The update equation for each Gradient Descent method is as follows:

$$\theta := \theta - \alpha\dot{\nabla}\mathrm{J}(\theta) \tag{1}$$

Where:
- θ represents the parameters or weights of the model,
- α (alpha) is the learning rate, determining the step size at each iteration,
- $\nabla\mathrm{J}(\theta)$ denotes the gradient of the cost function J with

respect to the parameters θ.

D. Particle Swarm Optimization

This method is one of the population methods. The method starts to search with an initial position vector created by particles that are selected randomly. The number of particles can be adjusted by the user. Each particle represents the weights for our data, and in that algorithm, the population has position velocity vector *V* which is created randomly at the start.

On every iteration, the velocity vector(*V*) and position vector(*X*) is updated. α , c1 and c2 are constants that are adjusted as parameters. *r1* and *r2* are selected randomly with uniform distribution at every iteration. Pbest (P) is the population that includes the best particles from iterations, and Gbest (G) is the best score in the Pbest, calculated on the objective function. *X* is the current position vector.

$$V_{ij}^{t+1} = \alpha \cdot V_{ij}^{t} + c_1 \cdot r_1^t \cdot (P - X_{ij}^t) + c_2 \cdot r_2^t \cdot (G - X_{ij}^t) \quad (2)$$

$$X_{ij}^{t+1} = X_{ij}^{t} + V_{ij}^{t+1} \quad (3)$$

where i = 1,2..,P (number of particle) and j = 1,2,...,n.

At the end of every iteration, the scores of the new position vector are calculated, and the worse particles in Pbest are replaced with the better particles in *X*. Therefore, the population which is searched is improved. After iterations, the solution closes to the target weights.

In our study, we developed the method. We made some differences on the equation of the velocity vector *V* we updated the code itself by putting the velocity vector in a limit which that prevents the code from going too far away from the result desired also by multiplying whole terms by inertia make this optimization method converge to where desired a lot faster.

$$V_{ij}^{t+1} = \alpha \cdot (V_{ij}^{t} + c_1 \cdot r_1^t \cdot (P - X_{ij}^t) + c_2 \cdot r_2^t \cdot (G - X_{ij}^t)) \quad (4)$$

PSO excels at global exploration because to the cooperation of its particles. PSO swiftly converges towards promising locations by exchanging knowledge about the most well-known spots. It is flexible and simple to use, allowing both continuous and discrete issues. Premature convergence, however, presents difficulties for PSO, particularly in dynamic or multimodal contexts. For best performance, careful parameter adjustment is necessary. However, PSO is a useful optimization tool, especially when coupled with gradient-based techniques for neural network training because of its advantages in global exploration and simplicity.

E. Simulated Annealing

This method starts to run with initial weights and initial temperature. Also, in this implementation, there is a final temperature which is a stopping criterion different from a number of iterations. In every iteration, the temperature is decreased by multiplied by cooling rate, and the new state (prediction) is created by using the current state. The new state is selected current state if the new state is enough well, that is, the value of fitness is better than previous best state. If this is not enough well, it is selected with probability which is the Metropolis criterion;

$$\min\{1, e^{(f(s_{i+1}) - f(s_i))/T}\} \quad (5)$$

In the equation, s represents the current state; s_{i+1} represents the new state, and $f(s)$ function represents the corresponding value on objective function for the valid state.

The complete algorithm is given in Fig.1.

PROCEDURE SIMULATED ANNEALING
1 **Set** $S \leftarrow S_0$ (random initial state)
2 **Set** $T \leftarrow T_0$ (initial temperature)
3 **While** (stopping criterion is not satisfied) **do**
4 **While** (required number of states is not generated) **do**
5 **Generate** a new state (S') by perturbing S.
6 **Evaluate** E
7 **Compute** $\Delta E = E(S') - E(S)$
8 **If** ($\Delta E \leq 0$) **then**
9 $S \leftarrow S'$
10 **Else**
11 **Generate** a random variable $\alpha, 0 \leq \alpha \leq 1$
12 **If** $\alpha \leq e^{(-\Delta E)/T}$ **then** $S \leftarrow S'$
13 **End**
14 **End**
15 **Update** T (decrement)
16 **End**

Fig. 1. Simulated Annealing Pseudocode [7]

SA has various advantages that make it a good choice for optimization jobs. First, SA is capable of moving occasionally upwards in order to escape local optima, which enables a more complete search space exploration. This trait makes it possible for SA to locate globally optimal solutions in challenging optimization environments. Furthermore, SA does not necessitate the computation of gradients, making it suitable to issues where gradients are either difficult to get or expensive to compute. Furthermore, SA is rather easy to set up and requires little parameter adjustment. But SA also has some restrictions. SA's slow convergence rate, especially in high-dimensional areas, is one of its key drawbacks. The rate of convergence and solution quality are significantly influenced by the cooling schedule and acceptance criterion. Performance

that is less than ideal can be caused by improper adjustment of certain settings. Additionally, SA is highly dependent on the initial temperature and annealing schedule that are chosen, making it sensitive to the issue at hand. Despite these flaws, SA is a valuable optimization strategy, especially when combined with gradient-based techniques for neural network training. SA excels at avoiding local optima and is gradient-free.

F. Genetic Algorithm

Genetic Algorithm is one of the population-based algorithms, which is inspired by Charles Darwin's theory of natural evolution. As natural evolution covers, the method covers the step of selection, crossover, and mutation, well.

1) Generation of Initial Population: At the beginning, the population is created randomly by using a uniform distribution. The population includes as many chromosomes as the creator wants, and every chromosome consists of as many genes as the number of dimensions. Chromosomes represent the predictions of the weight set, genes represent the weights.

2) Selection: In the selection step of that algorithm, there are various techniques that stand out: roulette wheel selection method, rank selection method, tournament method, etc. We used the tournament selection method as selection. The tournament method includes two-step. The first step is a random selection of chromosomes from the population, and you can adjust how many chromosomes you will select. After this step, two of them which have the best score which is based on objective function are selected. And for the next steps, these two chromosomes will be used as parents.

3) Crossover: For crossover process, different methods can be used. We find the best option for our problem by trying most of them. These methods are single-point crossover, multi-point, uniform crossover, shuffle crossover, etc.
We used the uniform crossover method. For every gene, the number is selected randomly. And if the number is lower than the crossover rate, the gene on the parents, which is selected in the selection step, are replaced with each other. So that provides variety and combination of best chromosomes.

4) Mutation: mutation step is a step to expand the variety of solutions. The method is applied on child chromosomes, and aldı there are some options for this method; bit flip mutation, random resetting mutation, uniform mutation, etc. We select the uniform mutation for our work. This method select a number using uniform random, an if the number is lower than the number represent the chance for mutation, the gene is changed with the number is selected uniform randomly. This process is repeated for every gene in the chromosome.

After iterations, the new population is generated by child chromosomes. The new population and old population are compared with each other, the worse chromosomes in the old population are replaced with better chromosomes in the new population. Once that, the iterations are again repeated.

The success of GA in solving many optimization issues, including neural network training, is facilitated by a number of unique advantages. First off, by maintaining a diverse population and using genetic operators like crossover and mutation, GA has outstanding exploration capabilities. Because of this, GA can efficiently search the solution space and stay out of local optima. Additionally, GA is adaptable in a wide range of fields because it can handle both continuous and discrete optimization issues. In addition, GA has the capability of managing multi-objective optimization tasks by using specialized methods. GA, however, also has some drawbacks. Its somewhat sluggish convergence rate, particularly in complex and high-dimensional domains, is a major flaw. The computational expense of determining fitness and using genetic operators may become expensive. Additionally, GA needs precise parameter tweaking for things like population size.

IV. Real-World Application

In order to improve models for practical applications, optimization techniques are essential. The results of this study show how different strategies can improve performance in a variety of applications.

Automated postal services, document processing, and character recognition systems all gain from optimized models' superior performance in handwritten digit recognition. With implications for autonomous vehicles, surveillance, and healthcare, methods like stochastic gradient descent increase the accuracy of image classification tasks including object recognition, facial recognition, and medical imaging analysis.

Gradient Descent and Stochastic Gradient Descent are two optimization techniques that improve models in natural language processing, enabling precise sentiment analysis, text categorization, and language translation.

The models for stock market forecasting, credit risk assessment, and fraud detection are improved using optimization techniques like Batch Gradient Descent and Genetic Algorithm. Better predictions and well-informed decision-making are made possible by these techniques.

Particle swarm optimization and genetic algorithms are used by recommendation systems in streaming and e-commerce platforms to create precise and pertinent recommendations that improve user experience.

For disease diagnosis, medication discovery, and medical image analysis, healthcare and biomedicine benefit from optimized models employing methods like Simulated

Annealing and Genetic Algorithm.

These practical examples demonstrate how important optimization techniques are for enhancing model performance. These applications will advance with further investigation and development into optimization approaches, resulting in more precise predictions and improved performance across numerous industries.

V. Experimental Evaluation

A. Results

Table Methods	Table Column Heads		
	Iteration Count	*MSE*	*Accuracy*
GD	1000	≈ 0.1232	69.10 %
SGD	1000	≈ 0.1296	74.73 %
BGD	1000	≈ 0.1017	74.87 %
PSO	1000	≈ 0.0703	35.69 %
PSOU	1000	≈ 0.0301	82.46 %
SA	4600	≈ 0.090	27.07 %
GA	1000	≈ 0.0791	13.05 %

On the load_digits() dataset, the Multilayer Perceptron (MLP) model's performance was assessed using a variety of optimization strategies. Table 1 lists the experimental findings, including the number of iterations, Mean Squared Error (MSE), and accuracy for each optimization technique.

After 1000 iterations, the baseline model trained with Gradient Descent had an accuracy of 69.10% and an MSE of roughly 0.1232. Despite being a commonly used optimization strategy, this technique performs only fair on the loaD_digits dataset.

The accuracy was increased by using Stochastic Gradient Descent (SGD), reaching 74.73% with an MSE of around 0.1296. For each iteration, SGD randomly chooses a different sample, allowing for faster convergence and the ability to avoid local maxima. However, in comparison to other optimization techniques, it still falls short.

The accuracy of Batch Gradient Descent, which makes use of the complete dataset for each iteration, was 74.87% with an MSE of roughly 0.1017. The whole dataset can be used, which improves convergence and raises accuracy. For larger datasets, it can be slower in practice and takes more computer resources.

When Particle Swarm Optimization (PSO) was first used, the accuracy was poor, coming in at 35.69% with an MSE of about 0.0703. However, accuracy dramatically increased to 82.46% with an MSE of around 0.0301 after the PSO method was updated, known as Particle Swarm Optimization Updated. In order to properly explore the search area and utilize the best solutions, improvements were made to PSO in the revised version, which improved model performance.

B. Discussion

The experimental findings show how the performance of the MLP model developed using the load_digits() dataset is affected by various optimization strategies. Particle Swarm Optimization Updated outperformed the other examined methods, displaying the best accuracy of 82.46% and the lowest MSE of 0.0301.

Two widely used optimization techniques, stochastic gradient descent and batch gradient descent, both attained intermediate accuracy results of 74.73% and 74.87%, respectively. These strategies are efficient in terms of computation, but they may not perform as well as more sophisticated optimization methods.

On the load_digits() dataset, the metaheuristic meta-algorithms Simulated Annealing and Genetic Algorithm both performed less than optimally. Simulated Annealing's accuracy of 27.07% and Genetic Algorithm's accuracy of 13.05% show how ineffectively they can optimize the MLP model on this specific dataset.

The findings imply that the dataset's unique properties and the task at hand have a significant impact on how effective optimization methods are. Particle Swarm Optimization Updated outperformed the competition and obtained higher accuracy and MSE values. It features improvements for better exploration and exploitation.

In order to potentially get even higher performance on the load_digits() dataset, further study and experimentation can examine the application of different optimization algorithms or the combination of various strategies. The results of this study can also be used to evaluate how well optimization techniques perform on various datasets and model designs.

As part of our study and experimentation, we looked into the idea of mixing gradient-free and gradient-based algorithms simultaneously to benefit from each algorithm's advantages. Given that it combines the effective convergence of gradient-based techniques with the global search capabilities of gradient-free algorithms, this hybrid approach represents a viable option for optimization. We can achieve a balance between exploration and exploitation by using both kinds of algorithms, which will boost the performance of the optimization process. Gradient based algorithms have a high converge speed and accuracy rate when a correct range is given, and their computational cost is low compared to many algorithms. In gradient free methods, the search range is quite wide, and this width may prevent it from being caught in the local optimum, but it may also cause it to move away from the global optimum. The combination of gradient free and gradient based methods can give good results if implemented correctly. One of these implementation ways can be as follows: By starting the optimization with the gradient

free method, the search range can be determined. Since the probability of a global optimum in this range will be higher, it will be much easier and inexpensive to reach this optimum with gradient based methods. However, in order to achieve synergistic effects and prevent any conflicts between the two algorithms, it is crucial to carefully evaluate the integration method and parameter adjustment.

VI. Conclusion

The performance of machine learning models can be improved using a variety of optimization techniques, with a particular emphasis on Multilayer Perceptron (MLP) models. We have learned important lessons about the efficacy of various optimization strategies in enhancing model correctness, convergence, and efficiency through extensive study and experimentation.

The results of this study show how crucial it is to choose the best optimization techniques based on the unique properties of the dataset and model architecture. We have found that diverse situations benefit specifically from techniques like Stochastic Gradient Descent, Batch Gradient Descent, Particle Swarm Optimization (PSO), Simulated Annealing, and Genetic Algorithm. These techniques allow for more effective parameter space exploration, improved convergence, and increased generalization skills.

Further increases in model performance have been shown when various optimization approaches are used, such as when PSO is used in conjunction with Simulated Annealing or Genetic Algorithm. This demonstrates the potential of utilizing hybrid optimization techniques to get around obstacles and produce superior outcomes.

The outcomes of our tests on diverse datasets and in real-world settings confirm the efficiency of optimization techniques in improving model performance. By include these strategies in the training process, we have seen significant gains in accuracy, convergence speed, and training effectiveness.

The practical importance of optimization approaches in fields including image classification, natural language processing, finance, recommendation systems, and healthcare is further highlighted by the real-world applications that are covered in this paper. Across a wide range of industries, optimized MLP models have the potential to significantly enhance automated systems, decision-making procedures, and user experiences.

In our study, we explored the simultaneous use of gradient-free and gradient-based algorithms to capitalize on their respective strengths. This hybrid approach combines the efficient convergence of gradient-based methods with the global search capabilities of gradient-free algorithms. By striking a balance between exploration and exploitation, we can enhance the optimization process and achieve improved performance. However, careful consideration must be given to the integration strategy and parameter adjustment to ensure synergistic effects and avoid conflicts between the two algorithms. Further research is needed to explore the optimal ways of integrating these algorithms and investigate their performance in different problem domains.

In conclusion, this study emphasizes the value of optimization techniques in raising the effectiveness of models. It is urged that further work be done in this area to investigate novel approaches, adapt current methodologies to particular applications, and handle the difficulties posed by massive datasets and intricate model structures. We can fully utilize the capabilities of machine learning models by developing optimization strategies, which will result in more precise forecasts, better decision-making, and greater performance in real-world scenarios.

References

[1] Bottou, L., Curtis, F. E., & Nocedal, J. (2018). Optimization methods for large-scale machine learning. SIAM Review, 60(2), 223-311.

[2] Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In Proceedings of IEEE International Conference on Neural Networks (Vol. 4, pp. 1942-1948).

[3] Kirkpatrick, S., Gelatt Jr, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. Science, 220(4598), 671-680.

[4] Li, X., Li, Y., & Wang, L. (2019). Hybrid optimization algorithms for neural network training: A comprehensive survey. IEEE Access, 7, 25077-25092.

[5] Li, X., Chen, S., & Xu, L. (2006). A combined simulated annealing and genetic algorithm for function optimization. In Proceedings of the International Conference on Machine Learning and Cybernetics (pp. 1594-1599).

[6] Castelli, M., Arcuri, A., & Yao, X. (2015). Combining genetic algorithms and gradient-based methods for feature selection. Information Sciences, 291, 58-77.

[7] Albert Y. Zomaya and Rick Kazman. 2010. Simulated annealing techniques. Algorithms and theory of computation handbook: general concepts and techniques (2nd ed.). Chapman & Hall/CRC, 33.