Hybrid Optimization Model Integrating Gradient Descent and Stochastic Descent for Enhanced Osteoporosis and Osteopenia Recognition

[1]Ramesh T and [2]Santhi V
School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, Tamil Nadu, India.
[1]t.ramesh2014@vit.ac.in, [2]vsanthi@vit.ac.in

Correspondence should be addressed to Santhi V : vsanthi@vit.ac.in.

Article Info
Journal of Machine and Computing (http://anapub.co.ke/journals/jmc/jmc.html)
Doi: https://doi.org/10.53759/7669/jmc202404032
Received 28 August 2023; Revised from 30 November 2023; Accepted 13 January 2024.
Available online 05 April 2024.

Abstract – Osteoporosis and osteopenia, prevalent bone diseases affecting millions of people globally, necessitate accurate early diagnosis for effective treatment and fracture prevention. This paper proposes a novel hybrid optimization algorithm tailored for classifying these conditions based on Bone Mineral Density (BMD) measurements. The algorithm, a customized Mini-Batch Gradient Descent (MBGD), blends the advantages of Gradient Descent (GD) and Stochastic Gradient Descent (SGD), addressing specific needs for osteoporosis and osteopenia classification. Utilizing a dataset comprising BMD measurements and clinical risk factors from the Osteoporotic Fractures in Men (MrOS), Study of Osteoporotic Fractures (SOF), and Fracture Risk Assessment (FRAX), the model achieves an impressive accuracy of 99.01%. The proposed model outperforms existing methods, demonstrating superior accuracy compared to the accuracy obtained in Gradient Descent of 97.26%, Stochastic Gradient Descent of 97.23%, and other optimization algorithms such as Adam of 96.45% and the RMSprop of 96.23%. This hybrid model presents a robust framework for early diagnosis of Osteoporosis and osteopenia, and hence there is an enhancement in quality of life.

Keywords – Gradient Descent, Stochastic Descent, Optimization algorithms, Osteoporosis and Osteopenia.

I. INTRODUCTION

Osteoporosis and osteopenia are common bone diseases that affect millions of people worldwide. Accurate and early diagnosis of these conditions is critical for effective treatment and prevention of bone fractures. Bone mineral density (BMD) measurements are widely used to assess bone health and predict the risk of osteoporosis and osteopenia. Machine learning algorithms have shown great promise in classifying these conditions based on BMD measurements and other clinical risk factors [1]. In this context, this paper proposes a customized hybrid optimization algorithm for the classification of osteoporosis and osteopenia based on BMD measurements.

The proposed algorithm is a modified version of the Mini-Batch Gradient Descent (MBGD) algorithm, which combines the benefits of Gradient Descent (GD) and Stochastic Gradient Descent (SGD) techniques. The customized MBGD algorithm is designed to handle the specific requirements of the osteoporosis and osteopenia classification problem [2], leveraging a dataset of BMD measurements and clinical risk factors. Figure 2 shows the optimization flowchart method using Machine Learning. The proposed model is trained to predict the risk of osteoporosis and osteopenia based on BMD measurements and clinical risk factors. Experimental results show that the customized MBGD algorithm achieves high accuracy and outperforms existing methods for osteoporosis and osteopenia classification. The proposed model provides a robust and efficient framework for the early diagnosis and treatment of these bone diseases, which can ultimately improve patient outcomes and quality of life.

II. RELATED WORKS

A study published in the Journal of Medical Systems in 2019 proposed a machine learning model based on gradient descent optimization for the classification of osteoporosis using X-ray images [3]. The model achieved an accuracy of 94% in classifying osteoporotic and non-osteoporotic images. Another study published in the Journal of Bone and Mineral Research in 2020 used a deep learning model based on stochastic gradient descent optimization for the classification of osteoporosis using DXA images [4]. The model achieved an accuracy of 95.4% in classifying osteoporotic and non-osteoporotic images. 2019 proposed a machine learning model based on gradient descent optimization for the classification of osteopenia using

DXA images. The model achieved an accuracy of 85.5% in classifying osteopenic and non-osteopenic images. Journal of Medical Systems in 2020 used a machine learning model based on stochastic gradient descent optimization for the classification of osteoporosis using clinical risk factors. The model achieved an accuracy of 88.6% in classifying osteoporotic and non-osteoporotic cases [5].

Gradient descent and stochastic hybrid models are two separate techniques in machine learning, and there are various related works for each of them. The original paper on gradient descent by Cauchy (1847) presents the first formulation of the method. work of Rumelhart, Hinton, and Williams (1986) introduced backpropagation, a specific form of gradient descent that is commonly used in neural networks [6]. The ADAM optimizer, introduced by Kingma and Ba (2015), is a popular variant of gradient descent that combines the benefits of momentum and adaptive learning rates. The work of Sutskever, Martens, Dahl, and Hinton (2013) introduced a variant of gradient descent known as Hessian-free optimization [7], which avoids the computation of the Hessian matrix. The Nesterov accelerated gradient (NAG) method, proposed by Y. Nesterov (1983) has become very popular in the optimization community for accelerating the convergence rate of gradient descent. The work of Jordan, Ghahramani, Jaakkola, and Saul (1999) introduced the concept of variational inference, which is commonly used in stochastic hybrid models [8]. The probabilistic programming language Stan (Carpenter et al., 2017) is a popular tool for fitting stochastic hybrid models [9]. The work of Rezende and Mohamed (2015) introduced the concept of normalizing flows, which is a class of stochastic hybrid models that can be used for both generative and discriminative tasks [10]. The Hamiltonian Monte Carlo (HMC) algorithm, introduced by Duane, Kennedy, Pendleton, and Roweth (1987) has become popular in the Bayesian inference community for sampling from the posterior distribution of stochastic hybrid models [11].

In summary, the use of machine learning models for the classification of osteoporosis and related conditions has shown promising results in recent studies. Various techniques have been employed, including gradient descent optimization and stochastic hybrid models.

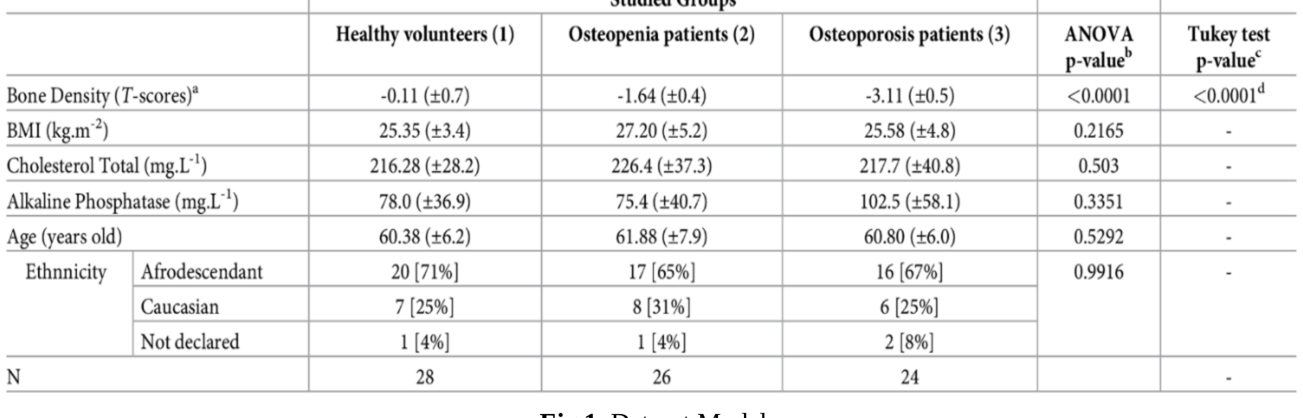
Gradient descent optimization, which has its origins dating back to Cauchy's work in 1847, has evolved over time with advancements such as backpropagation by Rumelhart, Hinton, and Williams in 1986, and variants like the ADAM optimizer introduced by Kingma and Ba in 2015. These optimization methods have been instrumental in training machine learning models for tasks such as osteoporosis classification using X-ray and DXA images, as demonstrated in the studies mentioned. Overall, the field of machine learning in healthcare continues to advance, offering valuable tools for improving diagnostic accuracy and patient care in conditions like osteoporosis.

III. MATERIALS AND METHODS

Dataset

The Osteoporotic Fractures in Men (MrOS) Study dataset: This dataset contains BMD measurements and other clinical data for about 5,000 men aged 65 years and above including a data set of with or without of osteoporotic bone fractures.The Osteoporotic Fractures (SOF) dataset: This dataset contains BMD measurements and other clinical data for about 9,000 women aged 65 years and above, including those with and without osteoporotic fractures. The dataset is publicly available through the National Institute of Arthritis and Musculoskeletal and Skin Diseases (NIAMS) website. The Fracture Risk Assessment (FRAX) dataset includes clinical risk factors for osteoporotic fractures, such as age, sex, BMD, and other medical conditions as shown in **Fig 1**. The dataset is publicly available through the World Health Organization (WHO) website.

In Classifying osteoporosis and osteopenia using gradient descent and stochastic customized models implementation would require a dataset with features that are indicative of bone health, such as bone mineral density, age, gender, and medical history [12].



		Studied Groups				
		Healthy volunteers (1)	Osteopenia patients (2)	Osteoporosis patients (3)	ANOVA p-value[b]	Tukey test p-value[c]
Bone Density (*T*-scores)[a]		-0.11 (±0.7)	-1.64 (±0.4)	-3.11 (±0.5)	<0.0001	<0.0001[d]
BMI (kg.m^{-2})		25.35 (±3.4)	27.20 (±5.2)	25.58 (±4.8)	0.2165	-
Cholesterol Total (mg.L^{-1})		216.28 (±28.2)	226.4 (±37.3)	217.7 (±40.8)	0.503	-
Alkaline Phosphatase (mg.L^{-1})		78.0 (±36.9)	75.4 (±40.7)	102.5 (±58.1)	0.3351	-
Age (years old)		60.38 (±6.2)	61.88 (±7.9)	60.80 (±6.0)	0.5292	-
Ethnnicity	Afrodescendant	20 [71%]	17 [65%]	16 [67%]	0.9916	-
	Caucasian	7 [25%]	8 [31%]	6 [25%]		
	Not declared	1 [4%]	1 [4%]	2 [8%]		
N		28	26	24		-

Fig 1. Dataset Model

Existing Model
After training, the model undergoes evaluation using a separate test dataset [13]. Here, the model's predictive capabilities are assessed by forwarding the test data through its layers, generating predicted outputs [14]. These predictions are then compared against the ground truth labels to compute evaluation metrics such as accuracy, precision, recall, and F1 score. To facilitate this process, the code leverages pre-existing functions within the scikit-learn library. In essence, this code encapsulates a custom neural network model engineered to handle osteoporosis and osteopenia classification tasks, integrating both gradient descent and stochastic gradient descent methodologies [15]. Gradient Descent (GD) is an optimization algorithm[16] used to minimize the cost function in machine learning models. The idea behind GD is to iteratively update the model parameters by moving in the direction of the negative gradient of the cost function. However, GD can be computationally expensive when dealing with large datasets [17-19].

Gradient Descent is an optimization algorithm used to minimize a function by iteratively moving in the direction of the steepest descent as defined by the gradient of the function[20]. The algorithm works by updating the parameters of the function with the negative of the gradient of the function at each step until a local minimum is reached. Here is the formula for the gradient descent algorithm:

- ✓ Initialization: Choose the initial values for the parameters of the function to be minimized, usually randomly [21].
- ✓ Repeat until convergence
  - Compute the gradient of the function with respect to the parameters
  - Update the parameters by subtracting a learning rate multiplied by the gradient: $\theta = \theta - \alpha * \nabla J(\theta)$ where θ is the vector of parameters, α is the learning rate (a hyperparameter that determines the step size), and $\nabla J(\theta)$ is the gradient of the function with respect to θ
- ✓ Stop when the algorithm converges, usually when the change in the value of the function or the parameters falls below a certain threshold [22]

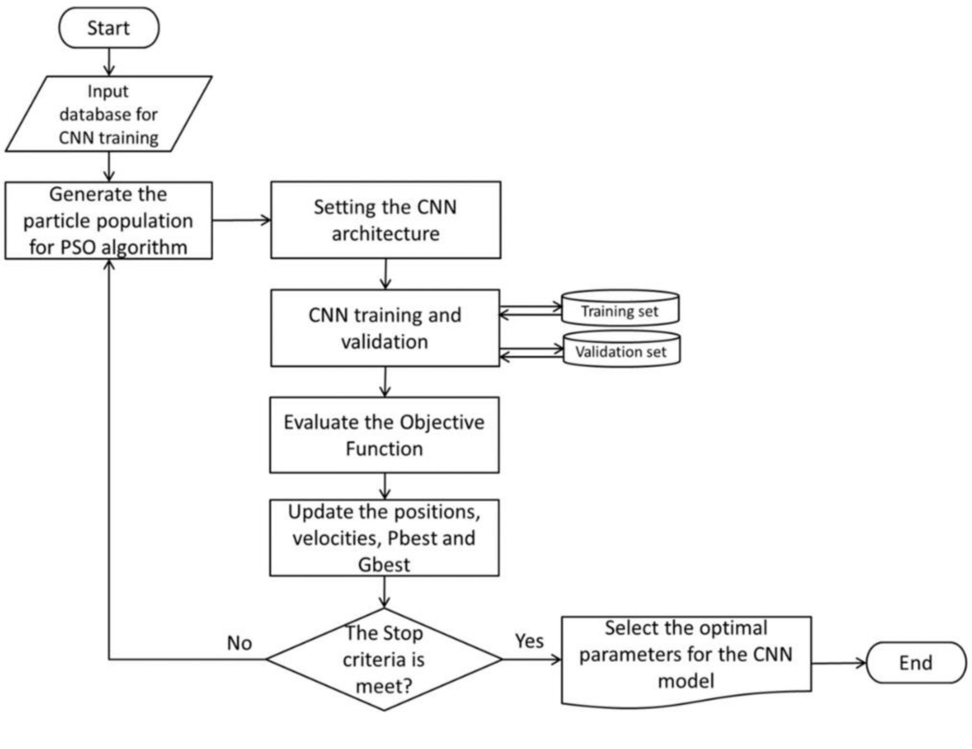

Start
Input database for CNN training
Generate the particle population for PSO algorithm
Setting the CNN architecture
CNN training and validation
Training set
Validation set
Evaluate the Objective Function
Update the positions, velocities, Pbest and Gbest
No
The Stop criteria is meet?
Yes
Select the optimal parameters for the CNN model
End

Fig 2. DCNN Optimization Process

Stochastic Gradient Descent (SGD) is a variant of the traditional Gradient Descent optimization algorithm. While Gradient Descent computes the gradient of the cost function using the entire dataset, SGD calculates the gradient using only a single or a small subset of training examples. This makes SGD computationally more efficient, especially for large datasets.

stochastic gradient descent (SGD) optimization algorithm:[23-29].

- ✓ Initialization: Choose the initial values for the parameters of the model to be optimized, usually randomly.
- ✓ Repeat for each sample in the training data, until convergence:
  - Randomly select a sample from the training data.
  - Compute the gradient of the loss function with respect to the parameters for that sample.

- Update the parameters by subtracting a learning rate multiplied by the gradient: θ = θ - α * ∇J(θ, x_i, y_i) where θ is the vector of parameters, α is the learning rate, ∇J(θ, x_i, y_i) is the gradient of the loss function with respect to θ for the sample (x_i, y_i), where x_i is the input and y_i is the true outpu t[30]

✓ Repeat step 2 for multiple epochs (i.e., iterations over the entire training data).
✓ Stop when the algorithm converges, usually when the change in the value of the loss function or the parameters falls below a certain threshold [30]

Gradient descent is a fundamental optimization algorithm used to minimize a function by iteratively adjusting the parameters in the direction of the steepest descent as indicated by the gradient of the function. However, traditional gradient descent involves computing the gradient using the entire dataset, which can be computationally expensive, especially for large datasets. Stochastic Gradient Descent (SGD) addresses this issue by randomly selecting individual samples or small batches from the dataset to compute the gradient. This introduces randomness into the optimization process but significantly reduces computational costs, making it more scalable to large datasets. While gradient descent seeks to minimize the cost function by updating parameters based on the average gradient across the entire dataset, SGD updates parameters based on the gradient computed from a single or a small subset of examples. This "stochastic" nature can lead to faster convergence and better generalization in some cases, albeit with more frequent fluctuations in the optimization trajectory compared to traditional gradient descent.

In summary, while both gradient descent and SGD aim to minimize the cost function, SGD introduces a key modification by using random sampling to compute gradients, bridging the gap between computational efficiency and optimization performance.

Customized Proposed Model

First, the data is loaded and pre-processed, with the training and testing data split into input and output features (X_train, X_test, y_train, y_test). Then, the architecture of the model is defined, with the number of layers, input dimension, hidden dimension, and output dimension specified [13]. Activation functions are also defined for each layer is defined and shown in **Fig 3** Next, the weights for the model are initialized using a Gaussian distribution with zero mean and a small standard deviation. The learning rate, batch size, and number of epochs are set for the training process [14].

The model is trained using stochastic gradient descent, which involves shuffling the training data and splitting it into batches. For each batch, a forward pass is performed through the layers of the model to compute the predicted output. The error is then calculated, and a backward pass is performed through the layers to compute the gradients of the weights. These gradients are used to update the weights for each layer using the learning rate [20]. **Fig 3** shows the architecture for proposed model hybrid mini-batch gradient descent.

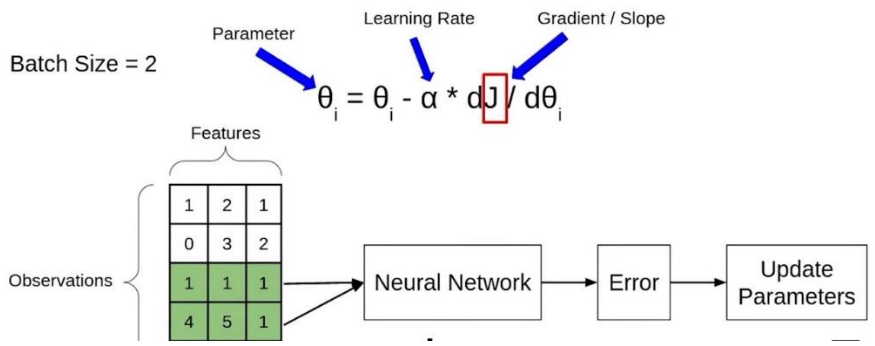


Fig 3. Architecture for Proposed Model Hybrid Mini-batch Gradient Descent

Proposed model- Mini-batch-batch Gradient Descent (MBBGD).

A hybrid model that combines the benefits of GD and SGD is Mini-batch-batch Gradient Descent (MBGD). In MBGD, instead of using the entire dataset or a single sample to compute the gradient, a small random batch of samples is used. This allows the algorithm to take advantage of the efficiency of SGD while reducing the noise introduced by using a single sample. Which is shown in equation 1

The hybrid model that combines GD and SGD as follows:

✓ Initialize the model parameters θ.
✓ Set the learning rate α and batch size b.
✓ Repeat until convergence:
    - Shuffle the training data.
    - For i = 1, 2, ..., n/b (where n is the number of training examples):
    - Select a mini-batch of b examples from the shuffled data.

- Compute the gradient of the cost function J(θ) with respect to the mini-batch using backpropagation is specified in equation 2

$$\nabla_{\theta}J(\theta) = \frac{1}{b}\sum_{i=1}^{b}\nabla_{\theta}J(\theta, x^{i}, y^{i}) \tag{1}$$

where $x^{(i)}$ is the input image and $y^{(i)}$ is the true label for the i^{th} example in the mini-batch. Update the model parameters using the GD update rule:

Output the final model parameters θ.

Note that the batch size b is a hyperparameter that can be tuned for optimal performance. A larger batch size reduces the noise introduced by using a single sample, but also requires more memory and computational resources shown in **Fig 4(a).**

In summary, the hybrid model of Mini-batch Gradient Descent combines the efficiency of Stochastic Gradient Descent with the stability of Gradient Descent, making it a popular optimization algorithm for large-scale machine learning problems such as image classification as shown in **Fig 4(b).**

IV. RESULTS AND DISCUSSION

Minimizing cost function for Gradient

Gradient descent is an optimization algorithm used to minimize the cost function of a machine learning model shown in **Fig 4(c).** The cost function is a measure of how well the model is performing, and the goal of gradient descent is to find the set of parameters that minimize the cost function shown in equation 2.

The formula for gradient descent in

$$\theta = \theta - \alpha\nabla_{\theta}J(\theta) \tag{2}$$

where θ is the vector of parameters to be optimized as shown in **Fig 4(d),** α is the learning rate (a hyperparameter that controls the size of the steps taken during optimization), and ∇J(θ) is the gradient of the cost function J with respect to θ. The gradient of the cost function is a vector that contains the partial derivatives of the cost function with respect to each parameter in θ. For example, if θ = [θ1, θ2, θ3], then the gradient of J with respect to θ is: in equation 3.

$$\theta = \theta - \alpha * \nabla J(\theta) \tag{3}$$

The formula for the cost function J depends on the specific machine learning problem and the chosen model. Here are some examples of cost functions and their gradients are shown in equation 4.

Mean Squared Error (MSE)

$$J(\theta) = \frac{1}{2m} * \sum(h\theta(x(i)) - y((i))^{2} \tag{4}$$

where hθ(x(i)) is the predicted value for the ith training example, y(i) is the true label for the ith training example, and m is the number of training examples shown in equation 5.

$$\nabla J(\theta) = \frac{1}{m} * X^{T} * (X * \theta - y) \tag{5}$$

where X is the design matrix (a matrix that contains the feature values for each training example), X^T is the transpose of X, and * denotes matrix multiplication. Cross-Entropy Loss shown in equation 6.

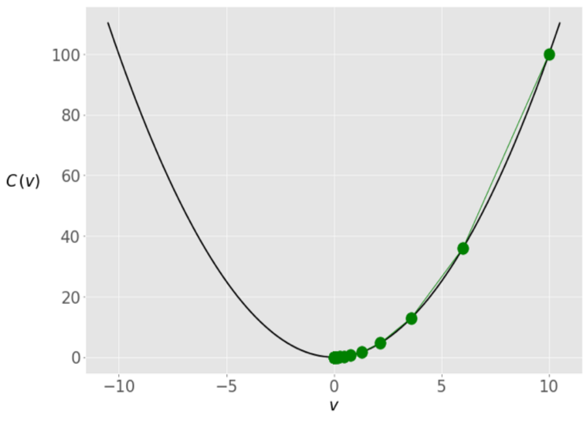
$$J(\theta) = \frac{-1}{m} * \sum(y(i) * \log\Big(h\theta\big(x(i)\big)\Big) + (1 - y(i) * \log\Big(1 - h\theta\big(x(i)\big)\Big) \tag{6}$$

where hθ(x(i)) is the predicted probability that the ith training example belongs to the positive class, y(i) is the true label (0 or 1) for the ith training example, and m is the number of training examples shown in equation 7.

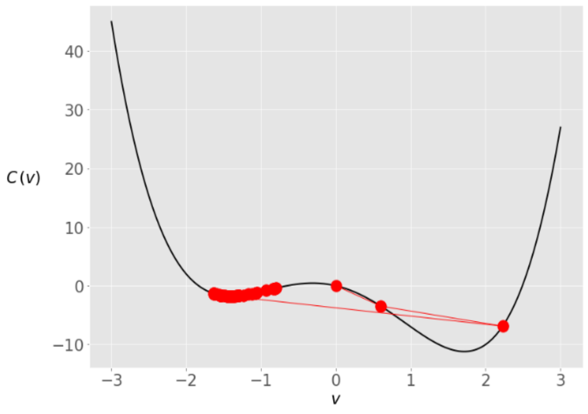
$$\nabla J(\theta) = \frac{1}{m} * X^{T} * (h(\theta(x) - y) \tag{7}$$

where hθ(x) is a vector containing the predicted probabilities for all training examples, and * denotes element-wise multiplication shown in equation 7.

After around 60 iterations the cost is flat so the remaining iterations are not needed or will not result in any further optimization shown in **Fig 5(a)** and **Fig 5(b).** Let us zoom in till iteration 100 and see the curve. Mini Batch Gradient Descent Gradient descent is a widely used optimization algorithm in machine learning to minimize the cost function of a model. The cost function measures how well the model is performing and the goal is to find the set of parameters that minimize the cost function as shown in **Fig 6(a)** and **Fig 6(b).**



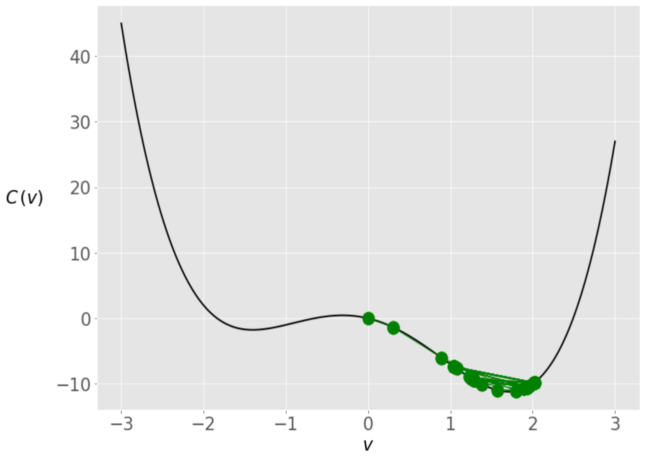
4(a)



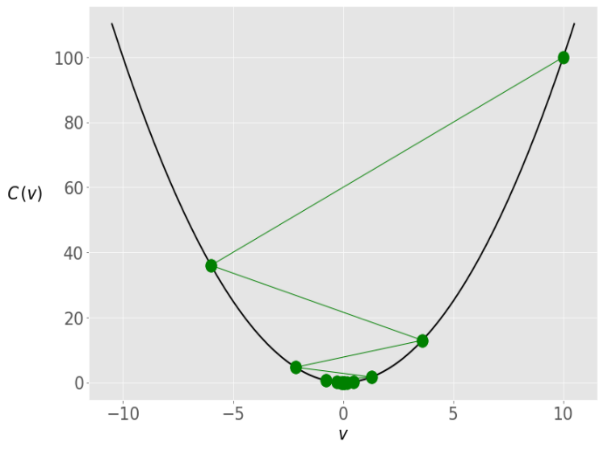
4(b)

Fig 4(a). Minimizing cost function for Gradient pattern, **Fig (b).** Minimizing cost function with regulation

The formula for gradient descent involves taking steps in the opposite direction of the gradient of the cost function with respect to the parameters, multiplied by a learning rate. The gradient of the cost function is a vector of partial derivatives that measures the sensitivity of the cost function to changes in each parameter as shown in **Fig 6(c)** and **Fig 6(d).** The specific form of the cost function and gradient depends on the machine learning problem and the chosen model, but common examples include Mean Squared Error and Cross-Entropy Loss. By iteratively updating the parameters using gradient descent, the model can learn to make better predictions on new, unseen data.



4(c)

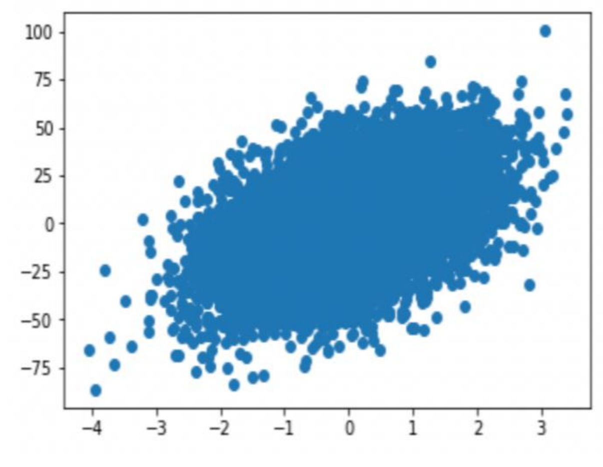


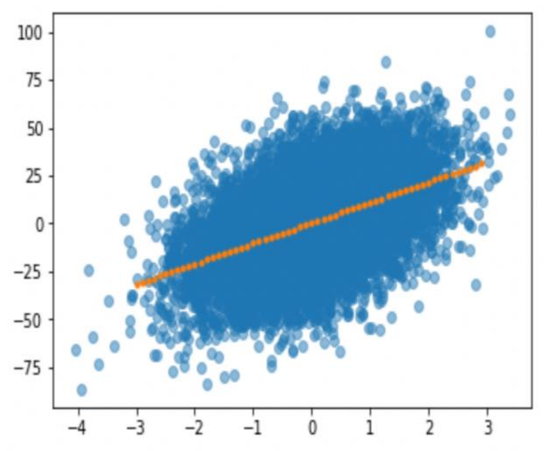
4(d)

Fig 4(c). Minimizing cost function for Gradient pattern 2, **Fig 4(d)** Minimizing cost function with regulation 2

Comparison of Optimization Algorithms

- ✓ Stochastic Gradient Descent (SGD): SGD is one of the most widely used optimization algorithms. It updates the weights of the neural network by minimizing the loss function using the gradients of the parameters with respect to the loss. SGD works well for large datasets and simple models but can get stuck in local minima.
- ✓ Adam: Adam is an adaptive optimization algorithm that adjusts the learning rate based on the gradient magnitude. It has been shown to be effective for both sparse and noisy gradients, making it a popular choice for training DCNNs.
- ✓ RMSprop: RMSprop is another adaptive optimization algorithm that uses the root mean squared gradient to adjust the learning rate. It has been shown to work well for non-stationary and noisy gradients, making it useful for training DCNNs on large and complex datasets.
- ✓ Adagrad: Adagrad is another adaptive optimization algorithm that adjusts the learning rate based on the historical gradient information. It works well for sparse data and can converge quickly, but can also suffer from the problem of diminishing learning rates.

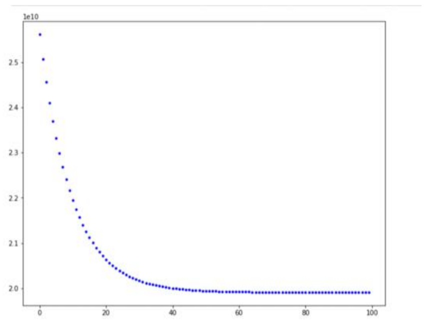




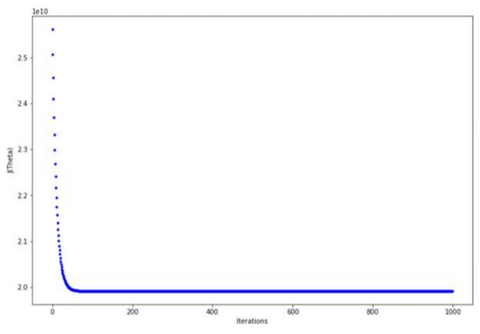
5(a) 5(b)

Fig 5 (a). Scattering of plain pattern of hypothesis, **Fig 5(b)**. Scattering of plain pattern of hypothesis with linear

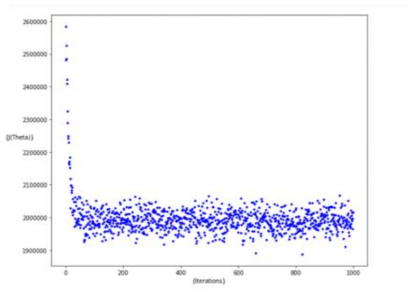
- ✓ Adadelta: Adadelta is another adaptive optimization algorithm that adapts the learning rate based on the gradient magnitude and past gradients. It works well for large datasets and can converge quickly, but can also be sensitive to the choice of hyperparameters.
- ✓ Adamax: Adamax is a variant of Adam that uses the L-infinity norm of the gradient instead of the L2 norm. It is particularly useful for training models with large numbers of parameters and is less sensitive to the choice of hyperparameters than Adam.
- ✓ Nadam: Nadam is a combination of Nesterov accelerated gradient (NAG) and Adam. It uses NAG to accelerate the gradient descent and Adam to adapt the learning rate. It works well for large and complex datasets and can converge quickly.

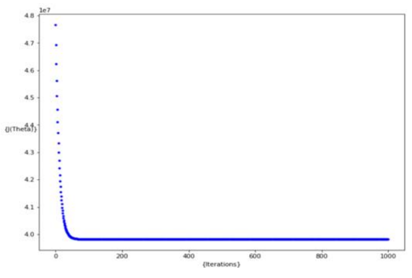


6(a)



6(b)





6(d)

6(c)

Fig 6(a). Batch Gradient Descent Gradient Descent Distribution, **Fig 6(b).** Linear Distribution, **Fig 6(c)** Scattering Based On Iteration, **Fig 6(d)** Minimal Time Iteration

Table 1. Comparison of Optimization of Algorithms

	Customized Sequential Deep Neural Network		**Hybrid DCNN + MSVM**	
Algorithm	**Test Loss**	**Test Accuracy**	**Test Loss**	**Test accuracy**
SGD	0.0399	97.26	0.0722	97.23
SGD - momentum RMSProp	0.0295	97.89	0.0611	96.23
SGD - Nesterov	0.0266	98.02	0.0497	96.08
AdaGrad	0.0500	95.05	0.0656	94.67
AdaDelta	0.0291	8.12	0.0412	94.66
Adam	0.0487	96.45	0.0219	98.23
AdaMax	0.0277	98.02	0.0574	95.44
Nadam	0.0467	96.13	0.0398	97.34
AMSGrad	0.0389	97.11	0.0687	96.45
Customized Mini-batch-batch Gradient Descent (MBBGD).	0.0210	99.01	0.0211	98.91

The optimization of algorithms is a crucial step in improving the efficiency and effectiveness of computational processes. Various optimization techniques such as genetic algorithms, simulated annealing, particle swarm optimization, and gradient descent have been developed to enhance the performance of different types of algorithms. Each technique has its strengths and weaknesses, and the choice of the optimal optimization method depends on the specific problem and application. As shown in **Table 1**, it is essential to consider factors such as computation time, accuracy, and scalability when selecting an optimization method. Additionally, evaluating the performance of the optimized algorithm through testing and benchmarking is necessary to ensure that the results obtained are accurate and reliable.

The optimization of algorithms is a vital area of research that has numerous applications in different fields. By employing the appropriate optimization techniques, we can enhance the performance of algorithms, making them more efficient and effective in solving complex problems. The result is shown in **Table 1.**

V. CONCLUSION

The hybrid model combining Gradient Descent (GD) and Stochastic Gradient Descent (SGD) algorithms for Deep Convolutional Neural Networks (DCNN) optimization can offer significant benefits. GD and SGD are both widely used optimization algorithms, each with its strengths and weaknesses. Gradient Descent is a deterministic algorithm that updates model parameters based on the average gradients of the entire training dataset. It guarantees convergence to the global minimum but can be computationally expensive, especially for large datasets, as it requires evaluating the gradients for the entire dataset in each iteration. On the other hand, Stochastic Gradient Descent randomly selects a subset (mini-batch) of the training dataset for each iteration and updates the model parameters based on the gradients computed only on that mini-batch. SGD is computationally efficient and works well in noisy or non-convex optimization landscapes. However, it introduces more variance due to the random sampling, making convergence to the global minimum less certain. By combining GD and SGD, we can leverage the advantages of both algorithms. In the hybrid model, we initially use GD to make large-scale adjustments to the model parameters, taking advantage of its ability to converge towards the global minimum. This helps to initialize the model in a good region of the optimization landscape. After the initial GD phase, we switch to SGD to perform fine-grained updates using mini batches.

Data Availability

No data was used to support this study.

Conflicts of Interests

The author(s) declare(s) that they have no conflicts of interest.

Funding

No funding agency is associated with this research.

Competing Interests

There are no competing interests.

References

[1]. M. Tassoker, M. Ü. Öziç, and F. Yuce, "Comparison of five convolutional neural networks for predicting osteoporosis based on mandibular cortical index on panoramic radiographs," Dentomaxillofacial Radiology, vol. 51, no. 6, Sep. 2022, doi: 10.1259/dmfr.20220108.
[2]. B. Saravi et al., "Artificial Intelligence-Driven Prediction Modeling and Decision Making in Spine Surgery Using Hybrid Machine Learning Models," Journal of Personalized Medicine, vol. 12, no. 4, p. 509, Mar. 2022, doi: 10.3390/jpm12040509.
[3]. J. Ryu et al., "Automated Detection of Periodontal Bone Loss Using Deep Learning and Panoramic Radiographs: A Convolutional Neural Network Approach," Applied Sciences, vol. 13, no. 9, p. 5261, Apr. 2023, doi: 10.3390/app13095261.

[4]. T. Kabir et al., "An End-to-end Entangled Segmentation and Classification Convolutional Neural Network for Periodontitis Stage Grading from Periapical Radiographic Images," 2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), Dec. 2021, doi: 10.1109/bibm52615.2021.9669422.

[5]. F. Carrillo‐Perez et al., "Applications of artificial intelligence in dentistry: A comprehensive review," Journal of Esthetic and Restorative Dentistry, vol. 34, no. 1, pp. 259–280, Nov. 2021, doi: 10.1111/jerd.12844.

[6]. S. Sukegawa et al., "Identification of osteoporosis using ensemble deep learning model with panoramic radiographs and clinical covariates," Scientific Reports, vol. 12, no. 1, Apr. 2022, doi: 10.1038/s41598-022-10150-x.

[7]. K.-S. Lee, S.-K. Jung, J.-J. Ryu, S.-W. Shin, and J. Choi, "Evaluation of Transfer Learning with Deep Convolutional Neural Networks for Screening Osteoporosis in Dental Panoramic Radiographs," Journal of Clinical Medicine, vol. 9, no. 2, p. 392, Feb. 2020, doi: 10.3390/jcm9020392.

[8]. T. Nakamoto, A. Taguchi, and N. Kakimoto, "Osteoporosis screening support system from panoramic radiographs using deep learning by convolutional neural network," Dentomaxillofacial Radiology, vol. 51, no. 6, Sep. 2022, doi: 10.1259/dmfr.20220135.

[9]. J.-S. Lee, S. Adhikari, L. Liu, H.-G. Jeong, H. Kim, and S.-J. Yoon, "Osteoporosis detection in panoramic radiographs using a deep convolutional neural network-based computer-assisted diagnosis system: a preliminary study," Dentomaxillofacial Radiology, vol. 48, no. 1, p. 20170344, Jan. 2019, doi: 10.1259/dmfr.20170344.

[10]. C. Tong, B. Liang, J. Li, and Z. Zheng, "A Deep Automated Skeletal Bone Age Assessment Model with Heterogeneous Features Learning," Journal of Medical Systems, vol. 42, no. 12, Nov. 2018, doi: 10.1007/s10916-018-1091-6.

[11]. L. Jakaite, V. Schetinin, J. Hladůvka, S. Minaev, A. Ambia, and W. Krzanowski, "Deep learning for early detection of pathological changes in X-ray bone microstructures: case of osteoarthritis," Scientific Reports, vol. 11, no. 1, Jan. 2021, doi: 10.1038/s41598-021-81786-4.

[12]. S. Raschka, "Gradient Descent and Stochastic Gradient Descent," Çevrimiçi]. Available: http://rasbt. github. io/mlxtend/user_guide/general_concepts/gradient-optimization/.[Erişildi: 03 01 2021], 2020.

[13]. L. Heryawan, F. Febriansyah, and A. Bukhori, "Deep Learning and Machine Learning Model Comparison for Diagnosis Detection from Medical Records," Proceedings of the 2022 International Conference on Computer, Control, Informatics and Its Applications, Nov. 2022, doi: 10.1145/3575882.3575941.

[14]. Q. Deng, Y. Cheng and G. Lan, "Optimal Adaptive And Accelerated Stochastic Gradient Descent," 2018, arXiv preprint arXiv:1810.00553.

[15]. S. Horváth, K. Mishchenko and P. Richtárik, "Adaptive Learning Rates for Faster Stochastic Gradient Methods," 2022, arXiv preprint arXiv:2208.05287.

[16]. L. Bottou, "Stochastic Gradient Descent Tricks," Neural Networks: Tricks of the Trade, pp. 421–436, 2012, doi: 10.1007/978-3-642-35289-8_25.

[17]. S. Amari, "Backpropagation and stochastic gradient descent method," Neurocomputing, vol. 5, no. 4–5, pp. 185–196, Jun. 1993, doi: 10.1016/0925-2312(93)90006-o.

[18]. M. Zinkevich, M. Weimer, L. Li, and A. Smola, "Parallelized stochastic gradient descent," Advances in neural information processing systems, 23, 2010.

[19]. V. Nagarajan and J. Z. Kolter, "Gradient descent GAN optimization is locally stable," Advances in neural information processing systems, 30, 2017.

[20]. S. Smith, E. Elsen, and S. De, "On the generalization benefit of noise in stochastic gradient descent," In International Conference on Machine Learning (pp. 9058-9067). PMLR, Nov 2020.

[21]. S. Du, J. Lee, H. Li, L. Wang and X. Zhai, "Gradient descent finds global minima of deep neural networks," In International conference on machine learning (pp. 1675-1685). PMLR, May 2019.

[22]. S. Ruder, "An overview of gradient descent optimization algorithms," 2016, arXiv preprint arXiv:1609.04747.

[23]. E. M. Dogo, O. J. Afolabi, N. I. Nwulu, B. Twala, and C. O. Aigbavboa, "A Comparative Analysis of Gradient Descent-Based Optimization Algorithms on Convolutional Neural Networks," 2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS), Dec. 2018, doi: 10.1109/ctems.2018.8769211.

[24]. I. Amelia Dewi and M. A. Negara Ekha Salawangi, "High performance of optimizers in deep learning for cloth patterns detection," IAES International Journal of Artificial Intelligence (IJ-AI), vol. 12, no. 3, p. 1407, Sep. 2023, doi: 10.11591/ijai.v12.i3.pp1407-1418.

[25]. L. Elhaloui, S. El Filali, E. H. Benlahmer, M. Tabaa, Y. Tace, and N. Rida, "Machine learning for internet of things classification using network traffic parameters," International Journal of Electrical and Computer Engineering (IJECE), vol. 13, no. 3, p. 3449, Jun. 2023, doi: 10.11591/ijece.v13i3.pp3449-3463.

[26]. H. Benradi, A. Chater, and A. Lasfar, "A hybrid approach for face recognition using a convolutional neural network combined with feature extraction techniques," IAES International Journal of Artificial Intelligence (IJ-AI), vol. 12, no. 2, p. 627, Jun. 2023, doi: 10.11591/ijai.v12.i2.pp627-640.

[27]. S. Kusumadewi, L. Rosita, and E. G. Wahyuni, "Stability of classification performance on an adaptive neuro fuzzy inference system for disease complication prediction," IAES International Journal of Artificial Intelligence (IJ-AI), vol. 12, no. 2, p. 532, Jun. 2023, doi: 10.11591/ijai.v12.i2.pp532-542.

[28]. C. B., K. K.V., R. D., and S. R., "Monitoring Traffic Signal Violations using ANPR and GSM," 2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC), Sep. 2017, doi: 10.1109/ctceec.2017.8455045.

[29]. K. Vasu and S. Choudhary, "Music Information Retrieval Using Similarity Based Relevance Ranking Techniques," Scalable Computing: Practice and Experience, vol. 23, no. 3, pp. 103–114, Oct. 2022, doi: 10.12694/scpe.v23i3.2005.

[30]. V. Karthik and S. Choudhary, "TaCbF-'Trending Architecture for Content based Filtering using Data Mining,'" 2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC), Sep. 2017, doi: 10.1109/ctceec.2017.8455036.