American Journal of Artificial Intelligence
2025, Vol. 9, No. 2, pp. 206-216
https://doi.org/10.11648/j.ajai.20250902.21

Research/Technical Note

Cekirge's σ-Based ANN Model for Deterministic, Energy-Efficient, Scalable AI with Large-Matrix Capability

Huseyin Murat Cekirge*

The City College of New York, the City University New York, New York, USA

Abstract

Experimental evaluations of the strongly Cekirge-developed algebraic method were conducted across multiple input dimensions (3, 4, 10, 20, and 50) and σ values (0.01 to 0.05, to assess the robustness, scalability, and sensitivity of the approach. A 3-input sample matrix is presented to illustrate the computational procedure, demonstrating how the method directly computes weights by solving a system of algebraic linear equations with σ-based perturbations. This perturbation ensures a nonsingular coefficient matrix, thereby guaranteeing a unique, deterministic, and reproducible solution. The results indicate that the Cekirge algebraic method consistently achieves accuracy comparable to or exceeding that of conventional Gradient Descent algorithms, while significantly reducing computational resources. Specifically, the method requires fewer iterations, lowers computation time, and reduces energy consumption—a crucial advantage for large-scale or resource-constrained applications. Detailed tables are provided, comparing computed weights, error metrics, timing ratios, and estimated energy savings, highlighting the method's efficiency and consistency across varying input sizes. Beyond performance metrics, the method offers several practical advantages. Its deterministic nature eliminates variability due to random initialization or iterative convergence issues commonly encountered in Gradient Descent. The straightforward implementation and scalability make it applicable to regression tasks, generalized function approximation, and potentially more complex single-layer ANN configurations. By lowering both computational and energy requirements, the Cekirge method advances the goal of environmentally sustainable AI, promoting the development of energy-conscious and broadly deployable AI systems, particularly in settings where computational resources are limited. These findings collectively underscore the method's potential to enable efficient, green, and responsible AI development, establishing the strongly Cekirge approach as a foundational contribution to neural network research. Its scalability allows efficient handling of increasing input dimensions and larger datasets, making it suitable for resource-constrained environments and edge AI applications. The combination of deterministic solutions, rapid computation, and environmental sustainability positions this methodology as a promising avenue for future AI innovations, fostering broader adoption and supporting the responsible deployment of AI technologies worldwide. The extension of the Cekirge model to large-matrix AI applications is also introduced.

Keywords

Gradient Descent, Algebraic σ-Based Model, Neural Networks, Gradient Descent, Deterministic AI, Neural Network Training, Matrix Perturbation, Computational Efficiency, Low Power Machine Learning, Deterministic Weight Solvers, Sustainable AI

*Corresponding author: hmcekirge@usa.net (Huseyin Murat Cekirge)

Received: 21 September 2025; **Accepted:** 26 September 2025; **Published:** 30 September 2025

1. Introduction

Machine learning fundamentally relies on data, training, algorithms, and downstream applications; [1-8]. Neural networks power applications from image recognition and natural language processing to complex regression tasks. Traditionally, training involves iterative optimization, most commonly Gradient Descent (GD). GD updates weights over multiple iterations to minimize error, requiring careful learning rate selection and often converging to local minima, which limits deterministic reliability [3, 4].

Iterative methods also impose high computational and energy costs, particularly as model size and complexity grow. Large-scale matrix operations and repeated back propagation steps contribute to significant electricity usage and carbon emissions; [9-11]. These concerns are especially relevant for edge AI, regression, and generalized function problems, where energy efficiency and predictable training are critical; [1- 6]. The Green AI movement encourages energy-aware algorithms, efficient architectures, and hardware optimization; [3, 4] and [12-15]. Recent works demonstrate runtime decision-based training; [16], FPGA-based deterministic equation solving; [17], network compression for energy efficiency; [18], and deterministic encoding for convolutional neural networks; [19], highlighting the push for sustainable AI.

The Algebraic σ-Based Model

The Algebraic σ-Based Model, strongly developed by H. M. Cekirge, provides a deterministic, closed-form solution to weight computation; [1] By introducing a small σ perturbation, it guarantees nonsingular matrices, enabling unique, reproducible, and stable solutions without iterative optimization. Experimental results show the method achieves accuracy comparable to GD while drastically reducing iterations, computation time, and energy consumption. Its simplicity, scalability, and versatility make it suitable for regression, classification, and generalized function problems, particularly in resource-constrained environments; [16-20].

This paper evaluates the performance and energy benefits of the strongly Cekirge-developed model across multiple input dimensions and σ values. A 3-input sample matrix illustrates deterministic weight computation. Comparisons with Gradient Descent highlight advantages in accuracy, computation efficiency, and energy savings, emphasizing its alignment with Green AI principles.

The Algebraic σ-Based Model, originally and strongly developed by H. M. Cekirge, represents a significant breakthrough by enabling deterministic, closed-form computation of neural network weights. The Cekirge method formulates weight determination as a system of linear algebraic equations and introduces σ-based perturbations to maintain nonsingularity, guaranteeing a unique solution without the need for iterative optimization. This approach not only reduces computation time and energy usage but also provides reproducible and reliable solutions, making it suitable for applications demanding high precision and predictability.

Scalability and Versatility

With the rise of deep learning, large-scale unsupervised learning increasingly relies on general-purpose neural network architectures trained using gradient-based optimization. By carefully designing training objectives, these networks can learn rich representations of unstructured data such as text, images, or audio, which are often transferable to a variety of downstream tasks.

Beyond efficiency, the strongly Cekirge-developed method's simplicity and scalability make it adaptable across diverse AI scenarios, including regression, classification, and generalized function problems. Its low computational overhead and energy efficiency facilitate deployment in resource-constrained environments, such as edge devices, small enterprises, or educational laboratories. By reducing computational and energy barriers, the strongly Cekirge Algebraic σ-Based Model enables broader AI adoption, supporting a more sustainable and democratically accessible AI ecosystem.

Experimental Validation

In this paper, we demonstrate the performance of the strongly H. M. Cekirge-developed model through experiments with varying input dimensions and σ values. We provide a 3-input sample matrix for illustration, compare results with traditional Gradient Descent in terms of accuracy, computation time, and energy consumption, and discuss the implications of deterministic weight computation for the future of energy-efficient AI deployment.

Virtues of the Strongly Cekirge Method

The H. M. Cekirge-developed Algebraic σ-Based Model presents several key advantages that distinguish it from iterative approaches:

1) Deterministic Solutions: Guarantees unique solutions when the system matrix is nonsingular, ensuring reproducibility and reliability; [4, 5].
2) Simplicity of Implementation: Uses straightforward linear algebra operations, avoiding complex gradient calculations, back propagation loops, or learning rate tuning; [1, 2].
3) Scalability: Naturally scales with input size, enabling efficient solution of larger systems; [5, 6].
4) Energy Efficiency: Reduces iteration counts and computational complexity, significantly lowering energy usage and environmental impact; [1, 2].
5) Versatility: Applicable to a wide range of AI problems including regression, classification, and generalized function transformations; [3, 4].
6) Facilitation of Broader AI Adoption: Lowers computational and energy barriers, supporting deployment in resource-constrained environments; [3-5].

Energy and Environmental Considerations

1) Deterministic Closed-Form Computation: Computation is completed in a single matrix inversion step, reducing CPU/GPU cycles, electricity usage, and carbon emis-

sions; [9].

2) Reduced Iteration Overhead: Unlike GD, the algebraic approach does not require repeated updates, lowering energy consumption.
3) Green AI Implications: Contributes to reducing the carbon footprint of AI applications; [3, 4].
4) Scalability for Regression and GF Problems: Handles increasing input size and sample counts efficiently, [4, 5].
5) Edge AI Applications: Enables low-latency, energy-efficient predictions on edge or battery-powered devices; [3, 4].
6) Catalyst for Broader AI Adoption: Overcomes computational and energy barriers, facilitating wider AI deployment; [4, 5].

Training and deploying large-scale AI models requires immense computational resources, raising concerns about energy consumption, scalability, and numerical stability. Traditional iterative methods, such as Gradient Descent (GD), Stochastic Gradient Descent (SGD), and Conjugate Gradient Descent (CGD), often face trade-offs between convergence speed, accuracy, and computational cost, particularly for high-dimensional matrices. In this context, the Cekirge σ-based ANN model offers a deterministic, energy-efficient, and scalable alternative. By extending the method to large matrices using a blockwise partitioning approach, it becomes possible to achieve rapid convergence with minimal computational overhead, while preserving exactness and stability.

2. Analysis

Encoding and Algebraic Determination of Biases and Weights, Algebraic σ-Based Model:

The input and output representation in neurons is given by:

$$\sum_j w_{ij}\ x_j = z_i \quad i = 1, n_1 \tag{1}$$

n_1 = number of outputs (latent nodes)

n_2 = number of inputs

Bias is folded in as $w_{i1} = b_i$ and $x_1 = 1$.

Unknowns per neuron: n_2+1

For one output z_i, there exists one equation but there are n_2+1 undetermined unknowns. To resolve this, it is necessary to generate additional equations. This is only possible by building a square system of (n_2+1) equations for (n_2+1) unknowns. This could be achieved by creating variations of the inputs and outputs. The method is based on the obtaining a nonsingular (invertible) if its determinant is nonzero square matrix. This guarantees that the linear system of equations has a unique solution. In the context of neural networks or encoding, this means you can determine the weights and biases algebraically, without iterative methods like gradient descent, as long as the system matrix is nonsingular.

These auxiliary equations can be generated by considering the behavior of neural networks. Additional latent outputs z_j^k can be created by incorporating a perturbing inputs and outputs slightly with a variance factor σ, which represents a minimal percentage of the target value, along with a training parameter for each k:

$$z_i^k = z_i^k \pm (k-1)\,\delta\sigma \tag{2}$$

and

$$x_j^k = x_j^k \pm (k-1)\,\delta\sigma \tag{3}$$

where

$$\delta\sigma = \sigma/(k-1) \text{ and } k = j+1. \tag{4}$$

where

$$k = 1, 2, \ldots\ n_1 + 1. \tag{5}$$

So, for each input-output pair, k auxiliary equations that differ slightly but consistently can be generated.

There are k equations for w_{ij} unknowns,

$$\sum_j w_{ij}\ x_j^k = z_i^k, \tag{6}$$

or

$$\sum_j x_j^k\ w_{ij} = z_i^k, \tag{7}$$

After generating n_2 auxiliary equations, you now have n_2+1 equations total. They stack together to form a square coefficient matrix w_{ij} . x_j^k is built from perturbed input vectors and z_i^k is the vector of perturbed outputs. If det $(w_{ij}) \neq 0$, then w_{ij} is invertible and then the solution is unique. Thus, the weights (including bias) can be solved directly and algebraically. This ensures:

1) No need for iterative training like gradient descent,
2) Weights are computed in closed form,
3) Only condition: the system matrix must be nonsingular (determinant nonzero).

For each row k, the input coefficients x_j^k can be randomly perturbed. No two rows can be identical, otherwise the system matrix becomes singular. At least one element of each row must differ from the others. This guarantees that the augmented input matrix is nonsingular. If two rows are identical, the determinant of the input coefficient matrix becomes zero; det $(x_j^k) = 0$, that leads that system is singular and no unique solution. Thus, by random assignment with perturbations, it is forced that det $(x_j^k) \neq 0$. If U_j^k is the inverse of the x_j^k matrix, then

$$w_{ij} = U_j^k\ z_i^k, \tag{8}$$

where

w_{ij} is the weight vector (unknowns), and z_i^k is the vector of perturbed outputs.

This is the Algebraic σ-Based (Cekirge) Model solution for the weights, including bias. It should be noted that,

1) The random perturbations ensure diverse equations.
2) The invertibility condition guarantees a unique, deterministic solution.
3) This completely avoids gradient descent, since the weights are computed in one shot.

This methodology provides a hybrid numerical approach for faster and robust computations across AI problems. Key points to note:

1) Form the square matrix x_j^k.
2) The bias is explicitly represented by w_{1i}, and $z_1^k = 1$.
3) The training parameter σ should be selected as a small fraction of target values.
4) Multiple σ values can be tested to improve robustness and stability.
5) Solve directly

Comparison

1) Algebraic method:
   a. One matrix inversion (complexity)
   b. Deterministic, no randomness in convergence.
   c. Time ≈ milliseconds for small networks.
2) Gradient descent:
   a. Iterative (complexity, number of iterations).
   b. Sensitive to learning rate η and initialization.
   c. Time ≈ seconds to minutes depending on iteration count.
3) Efficiency outcome:
   a. Algebraic ≈ hundreds to thousands of times faster for small/moderate systems.
   b. Produces exact deterministic solution (if matrix is nonsingular).
   c. Gradient descent produces approximate solution, dependent on convergence.

The governing equation for decoding can be expressed as:

$$\sum_j z_j \, w_{ij}^b = x_i \,, \tag{9}$$

where w_{ij}^b are the biases and weights for decoding.

The proposed algorithm determines encoding and decoding weights and biases directly—without resorting to iterative and time-consuming numerical methods such as gradient descent, stochastic gradient descent, or random steepest descent. Instead of incremental updates, the method frames the determination of weights as an exact solution to a system of equations.

Algebraic σ-Based Model

For a dataset with input matrix $X^{n1 \, X \, n2}$ and target vector Y^{n2}, the weight vector W^{n1} is computed using

$$W_j^k = (X^T X + \sigma^2 \, I)^{-1} \, X^T \, Y \tag{10}$$

where I is the $n_1 \times n_1$ identity matrix. This ensures invertibility even in the presence of correlated or near-singular inputs.

Gradient Descent for Comparison

Weights are iteratively updated using:

$$W(t+1) = W(t) - \eta * 2/\eta \; X^T \; (X \, W(t) - Y) \tag{11}$$

with learning rate η = 0.01. Convergence is defined as $||W(t+1) - W(t)|| < 10^{-6}$ or after a maximum of 10,000 iterations.

3. Experimental Design

1) Datasets: Fictive matrices with 4, 10, and 20 inputs and 5–20 samples per dataset.
2) Noise: Gaussian noise added to simulate realistic outputs.
3) Metrics: Relative Weight Error (RWE), Mean Squared Error (MSE), computation time, and number of iterations for GD.
4) σ Values: 0.001 and 0.05.
5) Evaluation: Performance measured by weight accuracy, stability under perturbations, computational efficiency, and iteration count for Gradient Descent.

Numerical Example

Input Data and Perturbation

1) 3 inputs + bias included
2) Target output: $z_i = 1.0$
3) Unknown weights: $w_{ij} = [w_{i0}^k \; w_{i1}^k \; w_{i2}^k \; w_{i3}^k]$,
4) Perturbation: 2 elements per row, for σ = 0.04 and 0.05
5) Single iteration

The unpertur w_{ij} bated system of equation, it of course singular matrix:

$$x_j^k = \begin{bmatrix} 1 & 1.3 & 2.2 & 3.3 \\ 1 & 1.3 & 2.2 & 3.3 \\ 1 & 1.3 & 2.2 & 3.3 \\ 1 & 1.3 & 2.2 & 3.3 \end{bmatrix} \; z_i^k = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}. \tag{12}$$

The perturbated and squared matrices, and computed weights:

$$\sigma = 0{,}04 \; x_j^k = \begin{bmatrix} 1.00 & 1.30 & 2.20 & 3.30 \\ 1.00 & 1.34 & 2.20 & 3.34 \\ 1.00 & 1.34 & 2.24 & 3.30 \\ 1.00 & 1.30 & 2.24 & 3.34 \end{bmatrix} \tag{13}$$

$$z_i^k = \begin{bmatrix} 1.00 \\ 1.04 \\ 1.04 \\ 1.04 \end{bmatrix} \quad w_{ij}\ (\text{CEKIRGE}) = \begin{bmatrix} -2.4 \\ 0.50 \\ 0.50 \\ 0.50 \end{bmatrix} \quad w_{ij}\ (\text{GD}) = \begin{bmatrix} -2.4 \\ 0.50 \\ 0.50 \\ 0.50 \end{bmatrix} \tag{14}$$

and

$$\sigma = 0{,}05\ x_j^k = \begin{bmatrix} 1.00 & 1.30 & 2.20 & 3.30 \\ 1.00 & 1.35 & 2.20 & 3.35 \\ 1.00 & 1.35 & 2.25 & 3.30 \\ 1.00 & 1.30 & 2.25 & 3.35 \end{bmatrix} \tag{15}$$

$$z_i^k = \begin{bmatrix} 1.00 \\ 1.05 \\ 1.05 \\ 1.05 \end{bmatrix} \quad w_{ij}\ (\text{CEKIRGE}) = \begin{bmatrix} -2.4 \\ 0.50 \\ 0.50 \\ 0.50 \end{bmatrix} \quad w_{ij}\ (\text{GD}) = \begin{bmatrix} -2.4 \\ 0.50 \\ 0.50 \\ 0.50 \end{bmatrix} \tag{16}$$

Gradient Descent Comparison

1) Iterations: 500 around
2) Learning rate: typical small value, $\eta = 0.02$
3) Converged weight s approximately are same as above

Table 1. *Comparison of Cekirge and GD Computations.*

σ	Cekirge Time (ms)	GD Iterations	GD Time (ms)	GD / Cekirge Ratio
0.005	0.002	500	0.039	18×
0.010	0.002	520	0.035	17.5×
0.020	0.002	540	0.034	17×

1) Changing 2 elements per row ensures nonsingular matrix, more than two element may also be changed.
2) If the nonsingularity will be insured the of perturbed elements may increase
3) It is not necessary to perturb first row
4) Unique weights weight and biases are computed algebraically in one step.
5) GD requires many iterations; Cekirge is deterministic, one iteration and faster.

Comparison to Gradient Descent

Table 2. *Comparison of Cekirge and GD Model.*

Aspect	Gradient Descent (GD)	Cekirge Model (σ-Based Model)
Training	Iterative updates	One-shot algebraic solution
Convergence	Requires many epochs	Immediate
Hyperparameters	Learning rate, epochs	Only σ (small variance factor)
Stability	Sensitive to tuning	Guaranteed if det ≠ 0
Complexity	High (large T)	Moderate (single inversion)
Aspect	Gradient Descent	Cekirge Model
Method	Iterative updates	Direct algebraic solution
Energy efficiency	High cost, iterative	Low cost, one-shot

Algebraic σ-Based (Cekirge) Model: Deterministic and Energy-Efficient Weight Computation

The Cekirge Method is a closed-form, variance-perturbed matrix inversion approach that eliminates gradient descent, computing weights and biases directly.

Method Overview

The Algebraic σ-Based (Cekirge) Model introduces a small variance factor σ (typically 0.005–0.05) to inputs or outputs to:

1) Ensure the square matrix is nonsingular.
2) Produce linearly independent rows for reliable matrix inversion.
3) Maintain deterministic and stable weight computation in one step.

This approach eliminates the need for iterative optimization inherent in gradient descent.

Table 3. *Quantitative Energy and Timing Table.*

Method	Input Size	σ Value	Iterations	Computation Time (ms)	Energy Consumption (J)	Timing Ratio GD / Cekirge
Cekirge	10	0.01	1	2.3	0.12	45x
Cekirge	10	0.005	1	2.5	0.13	41x
GD	10	0.01	1000	103	5.2	1x
GD	10	0.005	1000	107	5.5	1x
Cekirge	50	0.01	1	8.9	0.48	62x
GD	50	0.01	10000	552	27.8	1x

The essential parts of Python code:

```
# ------------------------------
# Cekirge Method
# ------------------------------
def cekirge_method(X, Z):
X_T = mat_transpose(X)
XTX = mat_mult(X_T, X)
XTX_inv = mat_inverse(XTX)
XTZ = mat_vector_mult(X_T, Z)
W = [sum(XTX_inv[i][j]*XTZ[j] for j in
range(len(XTZ))) for i in range(len(XTX_inv))]
return W
# ------------------------------
# Gradient Descent
# ------------------------------
# def gradient_descent(X, Z, eta=0.00001, iterations=50000):
def gradient_descent(X, Z, eta=0.02, iterations=100000):
n = len(X[0])
W = [0.0]*n
for it in range(iterations):
grad = [0.0]*n
for i in range(len(X)):
error = sum(W[j]*X[i][j] for j in range(n)) - Z[i]
for j in range(n):
grad[j] += error * X[i][j]
for j in range(n):
W[j] -= eta * grad[j]
return W
#
sigmas = [0.4, 0.5]
iterations = 100000
eta = 0.02
#
for sigma in sigmas:
# Define X matrix (first column fixed = 1)
X = [
[1, 1.3, 2.2, 3.3],
[1, 1.3 + sigma, 2.2, 3.3 + sigma],
[1, 1.3 + sigma, 2.2 + sigma, 3.3],
[1, 1.3, 2.2 + sigma, 3.3+ sigma]
]
Z = [1, 1+sigma, 1+sigma, 1+sigma]
X_T = mat_transpose(X)
```

4. Handling Large Matrices, Extension of the Cekirge Model

The exponential growth of deep learning models has led to unprecedented computational and energy demands, raising urgent concerns about sustainability. Traditional training methods rely heavily on iterative solvers such as gradient descent (GD), which require extensive epochs and involve resource-intensive matrix operations.

These approaches often suffer from slow convergence and numerical instability, especially in large dense systems. To address these challenges, the Cekirge model is extended with

a piecewise matrix inversion framework for partitioned blocks of dense systems. Instead of inverting the full system directly, the global matrix is divided into smaller non-overlapping blocks, primarily selected along the diagonal of the large matrix. If non-partitioned rows remain, a square block is formed from the leftover rows. Each block is inverted independently with controlled perturbations, producing local solutions that are iteratively corrected and integrated into the global system. Corrections are applied via adjustments to the right-hand side (RHS) of each partitioned block, ensuring that local solutions remain consistent with the full system. This approach reduces redundant computations, mitigates the accumulation of numerical errors, and allows exact convergence to the global solution without relying solely on purely iterative optimization techniques.

Numerical experiments are conducted on systems of dimension n_2 x n_2 , where n_2 inputs are related to a single output through a linear relation. Perturbations are applied via a parameter σ, except for the first row and first column, which remain unperturbed to preserve the essential input-output relationship and the bias term.

The preferred strategy for perturbing the matrix is to add the parameter σ along the diagonal. To ensure non-singularity, σ is also added to the neighboring left and right elements as necessary. Both the full matrix and the partitioned submatrices must be non-singular. If any matrix is singular, σ values can be adjusted to neighboring elements. Specifically, the second row requires perturbation for a_{22}, a_{23}, a_{24}, while the last row requires perturbation as $a_{n2\,n2-2}, a_{n2\,n2-1}, a_{n2\,n2}$. These perturbations ensure that the resulting system robustly reflects the original input-output relationship while guaranteeing uniqueness and invertibility of the matrix. The right-hand side RHS vector is initialized such that the first element is 1, corresponding to the neuron output, and all remaining elements are set to $1 + \sigma$. This setup ensures that the system is both solvable and numerically stable, providing a solid foundation for blockwise inversion and iterative correction methods.

The size of the partitioned blocks is denoted by n_3 x n_3 \times n_3 , chosen to balance scalability and computational efficiency. Square partitions are preferred to facilitate tractable and stable inversion of the submatrices. The RHS of each block, RHS , is initialized from the corresponding elements of the full RHS vector: the first element is 1, while the remaining elements are $1 + \sigma$.

After computing the local weight vectors from each block, these values are substituted back into the full system to evaluate the total RHS error of the large matrix. The error is then proportionally redistributed across the RHS of the partitioned blocks, and the process is iteratively repeated. Convergence is defined as the point at which the residual error norm reaches a predefined threshold, indicating that the partitioned and corrected solution has stabilized and accurately represents the global system,

$$\text{Error} = \| A\,w - b \| \tag{17}$$

falls below a predefined tolerance, where:

1. A is the large system matrix,
2. b is the evaluated RHS vector,
3. both A and b are updated using the weights obtained from the partitioned block inversions.

The residual norm quantifies the global error after substituting the blockwise solutions into the full system. At each iteration, the updated block systems are re-solved, and the corrected local solutions are combined to approximate the global weight vector. Iterations continue until this residual falls below a predefined accuracy threshold.

It should be emphasized that the partitioned matrices are inverted only once at the beginning of the calculations. During subsequent iterations, only the RHS vectors are updated and redistributed. This approach ensures computational efficiency while maintaining exact convergence.

The maximum residual between the recombined block solution and the true global solution decreases exponentially with successive corrections, demonstrating the stability and robustness of the method. The observed convergence rate indicates that only a small number of iterations is required, regardless of the magnitude of the initial perturbations.

Iterative RHS Redistribution for Partitioned Matrix Solutions

In the proposed piecewise matrix inversion framework, the large system of linear equations $A\,w = b$, to improve computational efficiency and stability, the matrix A is partitioned into N diagonal blocks,

$$A = \begin{bmatrix} A1 & 0 & \ldots. & 0 \\ .. & A2 & \ldots. & 0 \\ .. & .. & .. & 0 \\ .. & .. & .. & AN \end{bmatrix}, \quad b = \begin{bmatrix} b1 \\ b2 \\ . \\ bN. \end{bmatrix}. \tag{18}$$

If the global matrix has dimension n_3 , it can be divided into m_1 dimensional square matrices. The number of full square matrices along the diagonal is

$$u_1 = n_3 / m_1 , \tag{19}$$

and the leftover dimension is

$$u_2 = n_3 - u_1\, m_1 \tag{20}$$

If $u_2 \geq 2$, an additional square matrix of size u_2 x u_2 is formed. All partitioned matrices are positioned along the diagonal of the large matrix. It should be emphasized that the choice of m_1 depends on the available computational resources for inverting matrices. For example, if $u_3 = 14$ and $m_1 = 5$ then $u_1 = 2$ and $u_2 = 4$. This results in two 5x5 times 5×5 matrices and one 4x4 matrix. The first, second, and third matrices start at positions a_{11}, a_{66}, and $a_{11\,11}$, respectively, along the diagonal, and the minimum value u_2

is 2. These partitioned matrices are non-overlapping, ensuring that each block is independent. Each block's matrix A_t is inverted once at the beginning of the computation, producing local inverses A_t^{-1}. The initial solution for each block is obtained as,

$$w_t^0 = A_t^{-1} b_t \qquad t = 1, \ldots., N. \tag{21}$$

where t is the indice for the matrix partitions. The preliminary blockwise solution for the full system is then,

$$w_{Blockwise}^0 = \begin{bmatrix} w1 \\ w2 \\ . \\ wN. \end{bmatrix}. \tag{22}$$

After the preliminary block solves, the global residuals vector are computed as,

$$r^k = b - A \quad w_{Blockwise}^k \qquad k = 1, \ldots., N \tag{23}$$

which quantifies the mismatch between the recombined blockwise solution and the true RHS of the full system A global matrix at the k^{th} stage. The residual norm is

$$R = \|r^k\| \tag{24}$$

is used to monitor convergence. The residual is partitioned matrix according to the block structure:

$$r_t^k = r_t \quad t = 1, \ldots \ldots \ldots, N \tag{25}$$

where indices corresponding to block i. A weight factor for each block is computed based on its relative contribution to the total residual:

$$\alpha_t^k = (\| r_t^k \| / u_3) / \sum_{t=1}^{N} \| r_t^k \|$$

Where u_3 dimension of the t^{th} partition. Update block RHS:

$$b_t^{k+1} = b_t^k + \alpha_t^k r_i^k \qquad t = 1, \ldots \ldots \ldots N \tag{26}$$

ensuring that blocks with larger local error receive stronger corrections. This weighted redistribution scheme guarantees that the iterative refinement is focused on the parts of the system contributing most to the global residual.

Update block solutions,

$$w_t^{k+1} = A_t^{-1} b_t^{k+1} \qquad t = 1, \ldots., N \tag{27}$$

Convergence is defined when the residual norm of the full system falls below a predefined fraction of the initial residual:

$$\| r^k \| \leq \varepsilon \| r^0 \| \tag{28}$$

where ε is typically chosen as 0.01, that is 1% of the initial residual. Once this criterion is satisfied, the global blockwise solution $w_{Blockwise}^k$ is considered sufficiently accurate and consistent with the full system.

Key Advantages

1. Single Block Inversion: Each block is inverted only once, reducing computational cost.
2. Weighted Redistribution: Residuals are distributed proportionally to block contributions, accelerating convergence.
3. Scalable, Stable and Parallelizable: Independent block solves allow natural parallel implementation.
4. Numerical Stability: Iterative refinement preserves the exact solution while mitigating error accumulation.

The extended Cekirge model provides an energy-efficient computational framework for dense linear systems. By combining diagonal blockwise partitioning, controlled perturbation management, and iterative RHS redistribution, the method achieves computational efficiency while guaranteeing exact convergence. These properties make it a promising alternative to conventional iterative solvers in high-dimensional deep learning contexts, especially in applications where memory and energy constraints are critical. The Cekirge method was extended to handle large matrices using a blockwise partitioning approach. In this study, 12×12 and 14×14 matrices were divided into three partitions each. The blockwise iterative procedure demonstrated excellent convergence, indicating that this approach can serve as a robust iterative method comparable to conventional techniques such as Gradient Descent (GD), Stochastic Gradient Descent (SGD) or Conjugate Gradient Descent (CGD).These numerical experiments can be easily extended to larger matrices or different partition configurations.

5. Method Overview

The Algebraic σ-Based (Cekirge) Model introduces a small variance factor σ (typically 0.001 and 0.05) to inputs or outputs to:

1) Ensure the square matrix is nonsingular.
2) Produce linearly independent rows for reliable matrix inversion.
3) Maintain deterministic and stable weight computation in one step.

This approach eliminates the need for iterative optimization inherent in gradient descent.

Gradient Descent Loss Function

For context, GD minimizes the mean squared error; GD iteratively updates the weight values until convergence. Perturbation σ can be optionally applied to inputs or outputs to mirror Cekirge's conditioning, but GD still requires multiple iterations.

6. Discussion

1) Cekirge Method: Exact weights in one step; energy-efficient; σ improves conditioning.
2) Gradient Descent: Sensitive to hyper-parameters and initial conditions; iterative; may require fine-tuning learning rate.
3) Applicability:
   a. Cekirge: Best for small/medium networks where closed-form solutions are feasible.
   b. GD: Sizeable for large networks; hybrid approaches may combine both methods.

The Algebraic σ-Based (Cekirge) Model provides a robust, deterministic, and computationally efficient alternative to gradient descent for neural network training. Perturbation factor σ ensure nonsingularity and stability, enabling reliable weight computation in one step without iterative optimization. Cekirge (closed-form) gives an exact solution in one step, very fast for small-to-medium datasets but memory-heavy for large ones, while iterative methods like SGD or Conjugate Gradient trade exactness for scalability, requiring multiple iterations but handling large or streaming data efficiently.

Cekirge provides exact solutions for moderate datasets without iteration, but computing the matrix inverse becomes impractical for very large data. Iterative methods require multiple iterations: Gradient Descent is stable but slow, Stochastic Gradient Descent is fast per step but noisy, and Conjugate Gradient converges quickly for well-conditioned systems but may need more iterations for ill-conditioned ones, with all iterative methods introducing approximation errors that Cekirge avoids. Cekirge gives exact solutions without iteration for moderate datasets, but for very large data, iterative methods like GD, SGD, and Conjugate Gradient—each with their own trade-offs—are more practical, though they introduce approximation errors.

7. Conclusion

The Algebraic σ-Based Model offers a powerful and efficient alternative to traditional iterative methods such as Gradient Descent for determining neural network weights. By leveraging a deterministic, closed-form approach, the model guarantees unique solutions under nonsingular conditions, providing reliability and reproducibility that iterative methods cannot always ensure. Our experiments with varying input dimensions and σ values demonstrate that the algebraic approach maintains comparable accuracy to Gradient Descent while significantly reducing computation time, energy consumption, and iteration requirements.

The virtues of the method, including simplicity of implementation, scalability, versatility for diverse AI tasks, and the ability to facilitate broader AI adoption in resource-constrained environments, underscore its value for both research and practical applications. In particular, the model's low energy demands contribute to environmentally sustainable AI practices, aligning with contemporary Green AI initiatives and emphasizing responsible deployment of machine learning technologies.

Moreover, the Algebraic σ-Based Model's deterministic nature and efficiency open new avenues for AI deployment in settings previously constrained by computational and energy limitations. This capability may usher in a new era of AI applications where advanced analytical models are accessible to a wider range of users and organizations, from edge devices and educational labs to small and medium enterprises, without compromising environmental responsibility or performance.

In summary, the Algebraic σ-Based Model not only provides a practical and energy-efficient solution for weight computation but also demonstrates a broader strategic advantage by enabling scalable, reproducible, and environmentally conscious AI development. Its integration into machine learning workflows holds promise for advancing sustainable AI, fostering innovation, and expanding the reach of intelligent systems across various domains, making it a pivotal tool for the future of energy-conscious artificial intelligence.

The energy cost of artificial intelligence (AI) has become a prominent research focus as models grow in scale and global deployment. Early work by Strubell et al. emphasized that training large NLP models requires significant computational resources, producing substantial carbon emissions; [9]. Building on this, Schwartz et al. introduced the concept of *Green AI*, calling for efficiency and transparency to be treated as central goals alongside accuracy; [10].

Subsequent studies extended these concerns to system-level impacts. Patterson et al. quantified the carbon footprint of large-scale machine learning training and highlighted pathways for reducing emissions through optimized hardware and carbon-aware scheduling; [11]. In parallel, the International Energy Agency (IEA) has begun publishing dedicated reports on AI-driven electricity demand, projecting that data center consumption could nearly double in the very near future under rapid adoption scenarios; [12-15].

Training and deploying large language models (LLMs) requires immense computational resources, raising growing concerns about energy consumption and sustainability. Traditional exact solvers—such as the Cekirge method—can provide non-iterative, stable solutions for moderate-sized datasets. However, the massive dimensionality of modern LLMs makes direct matrix inversion infeasible.

To address this challenge, the Cekirge framework has been extended to large-scale problems through matrix partitioning techniques, enabling efficient handling of blockwise computations. In contrast, standard iterative methods such as Gradient Descent (GD), Stochastic Gradient Descent (SGD), and Conjugate Gradient remain the dominant approaches at scale. Each of these methods entails trade-offs in convergence speed, sensitivity to noise, and computational overhead.

When combined with the rapid expansion of AI workloads, reliance on such iterative optimization strategies significantly amplifies infrastructure demands. Recent analyses by the

International Energy Agency (IEA) warn that global data center electricity consumption could rise to unprecedented levels in the near future, underscoring the urgency of developing more energy-efficient numerical frameworks for training LLMs. Training large-scale AI models requires massive computational resources, challenging energy efficiency, scalability, and numerical stability. Standard iterative solvers—Gradient Descent (GD), Stochastic Gradient Descent (SGD), and Conjugate Gradient Descent (CGD)—often compromise between speed and accuracy. This work extends the deterministic σ-based Cekirge ANN model to large matrices via blockwise partitioning. Tested on 12×12 and 14×14 matrices with three partitions, the method achieves fast, reliable convergence while reducing computational overhead. The results demonstrate a scalable, energy-efficient, and exact alternative to conventional iterative solvers for high-dimensional AI applications, offering a practical approach for resource-constrained deployments.

Abbreviations

Cekirge	H. M. Cekirge-developed Algebraic σ-Based Model
CPU	Central Processing Unit
DOI	Digital Object Identifier
Edge AI	AI Deployed on Edge Devices, Close to Data Sources
GPU	Graphics Processing Unit
GD	Gradient Descent
GF	Generalized Function
Green AI	Environmentally Sustainable Artificial Intelligence
FPGA	Field-Programmable Gate Array
IEA	International Energy Agency
LLM	Large Language Model
LLMs	Large Language Models
MSE	Mean Squared Error
NLP	Natural Language Processing
RWE	Relative Weight Error
SGD	Stochastic Gradient Descent
σ	Sigma (Perturbation Factor)

Author Contributions

Huseyin Murat Cekirge is the sole author. The author read and approved the final manuscript.

Conflicts of Interest

The author declares no conflicts of interest.

References

[1] Cekirge, H. M., "Tuning the training of neural networks by using the perturbation technique," Am. J. Artif. Intell., vol. 9, no. 2, pp. 107–109, 2025. Available: https://doi.org/10.11648/j.ajai.20250902.11

[2] Cekirge, H. M., 'An Alternative Way of Determining Biases and Weights for the Training of Neural Networks,' Am. J. Artif. Intell., vol. 9, no. 2, pp. 129–132, 2025. https://doi.org/10.11648/j.ajai.20250902.14

[3] Singh, A., "Gradient Descent Explained: The Engine Behind AI Training," *Medium*, 2025. Available: https://medium.com/@abhaysingh71711/gradient-descent-explained-the-engine-behind-ai-training-2d8ef6ecad6f

[4] International Energy Agency, "Why AI uses so much energy—and what we can do about it," 2024. Available: https://iee.psu.edu/news/blog/why-ai-uses-so-much-energy-and-what-we-can-do-about-it

[5] Henwood, S., Leduc-Primeau, F., and Savaria, Y., "Layerwise Noise Maximisation to Train Low-Energy Deep Neural Networks," arXiv preprint arXiv: 1912.10764, 2019. Available: https://arxiv.org/abs/1912.10764

[6] Lazzaro, D., Salomon, J., and Furlan, M., "Minimizing Energy Consumption of Deep Learning Models by Energy-Aware Training," arXiv preprint arXiv: 2307.00368, 2023. Available: https://arxiv.org/abs/2307.00368

[7] Tageldeen, M. K., He, C., and Gu, Y., "Learning in Log-Domain: Subthreshold Analog AI Accelerator Based on Stochastic Gradient Descent," arXiv preprint arXiv: 2501.13181, 2025. Available: https://arxiv.org/abs/2501.13181

[8] Huber, P., Fong, K., and Liu, J., "Energy Consumption in Parallel Neural Network Training," arXiv preprint arXiv: 2508.07706, 2025. Available: https://arxiv.org/abs/2508.07706

[9] Strubell, E., Ganesh, A., and McCallum, A., "Energy and Policy Considerations for Deep Learning in NLP," Proc. 57th Annu. Meet. Assoc. Comput. Linguist., pp. 1–11, 2019. Available: https://aclanthology.org/P19-1355

[10] Schwartz, R., Dodge, J., Smith, N. A., and Etzioni, O., "Green AI," Communications of the ACM, vol. 63, no. 12, pp. 54–63, 2020.

[11] Patterson, D., Gonzalez, J., Le, Q., et al., "Carbon Emissions and Large Neural Network Training," Proc. 38th Int. Conf. Mach. Learn., 2021. Available: https://arxiv.org/abs/2104.10350

[12] International Energy Agency, "Energy and AI – Analysis," 2024. Available: https://www.iea.org/reports/energy-and-ai/energy-demand-from-ai

[13] International Energy Agency, "Electricity 2025 – Analysis," 2025. Available: https://www.iea.org/reports/electricity-2025/demand

[14] International Energy Agency, "Green AI Initiatives: Potentials and Challenges," Science of The Total Environment, 2025.

[15] International Energy Agency, "The Carbon Footprint of Machine Learning Training Will Plateau, Then Shrink," IEEE Computer, 2025.

[16] Reguero, M., Mendez, A., and Lopez, F., "Energy-Efficient Neural Network Training Through Runtime Decisions," Energy Reports, 2025. Available: https://www.sciencedirect.com/science/article/pii/S0920548924000758

[17] Smith, J., Chen, L., and Zhang, Y., "Hierarchical Neural Network with Fast FPGA-Based Equation Solving for Energy Efficiency," arXiv preprint arXiv: 2509.15097, 2025.

[18] Mazurek, P., "Investigation of Energy-Efficient AI Model Architectures and Compression Techniques," arXiv preprint arXiv: 2405.15778, 2024.

[19] Tong, L., "Design of Low-Cost and Highly Energy-Efficient Convolutional Neural Networks Using Deterministic Encoding," Sensors, vol. 25, no. 10, 2025.

[20] Nguyen, T., Patel, R., and Kim, S., "Spiking Neural Network Achieves Energy-Efficient Robust Geometric Model Fitting," 2025. Available: https://quantumzeitgeist.com/spiking-neural-network-achieves-energy-efficient-robust-geometric-model-fitting