DS-LLM: LEVERAGING DYNAMICAL SYSTEMS TO ENHANCE BOTH TRAINING AND INFERENCE OF LARGE LANGUAGE MODELS.

Anonymous authors
Paper under double-blind review

ABSTRACT

The training of large language models (LLMs) faces critical computational cost challenges, hindering their scaling toward AGI and broader adoption. With model sizes doubling approximately every 3.4 months and training costs surging from \$64 million for GPT-4 in 2020 to \$191 million for Gemini Ultra in 2023, the economic strain is unsustainable. While optimizations like quantization provide incremental improvements, they fail to address the fundamental bottleneck. In this work, we propose DS-LLM, a novel framework leveraging dynamic system (DS)-based machines, which exploit Natural Annealing to instantaneously converge to minimum energy states, enabling orders-of-magnitude gains in efficiency. Unlike traditional methods, DS-LLM maps LLM components to optimization problems solvable via Hamiltonian configurations and utilizes DS machines' continuous electric current flow for hardware-native gradient descent during training. We mathematically demonstrate the equivalence between existing LLMs and DS-LLMs and offer a viable approach to build a DS-LLM from a trained conventional LLM. Evaluations using different sizes of models showcase a 1,238× speedup and a 116,563× energy reduction on training, a 127× speedup and a 37,545× energy reduction on inference, while maintaining consistent accuracy.

D-Q5

1 INTRODUCTION

Large language models are currently driving rapid advancements in AI, with their model sizes doubling approximately every 3.4 months. This exponential growth demands unprecedented computational power for both training and inference. The cost of training an LLM from scratch has surged from \$64 million for GPT-4 in 2020 to \$191 million for Gemini Ultra in 2023. Today, most cloud and HPC resources are dedicated to LLMs, leading to significant societal and environmental cost and energy concerns. Meanwhile, Moore's Law, the fundamental driver behind IT and AI development for decades, is losing momentum, exacerbating the situation. While improving traditional computing methods remains crucial, the need to pioneer new energy-efficient architectures—potentially supported by intelligence-carried designs—is becoming increasingly urgent to ensure the sustainable and affordable scaling of AI models, particularly LLMs.

Some emerging candidates include quantum computing (Kerenidis et al., 2024), optical computing (Anderson et al., 2024), and computing-in-memory (Tu et al., 2023), etc. These emerging systems hold great potential but each faces distinct technical challenges and generally requires further development. Is it feasible, in the near term, to rely on mature CMOS-based technology to train LLMs as quickly as CNNs, by reducing training time from 10 million hours to 10,000 hours, meanwhile cutting energy consumption from 20 terajoules to 200 megajoules?

Recently, Dynamical-System-based (DS) machines have emerged which incorporate an electrodynamics-based model that naturally converges to the minimum energy state by following the physical dynamics of electrons. By correctly setting the initial state, boundary conditions, and final state, various problems can be embedded into this "Natural Annealing" process. Because no external energy is theoretically required during such a process, DS machines are extremely energy efficient and have already demonstrated remarkable potential in some real-world applications. For example, solving complex optimization problems like MAX-CUT in (Hamerly et al., 2019), work-

A-Q1

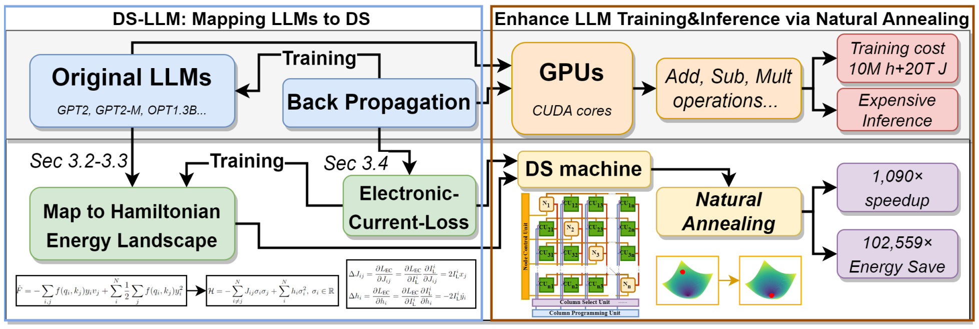
DS-LLM: Mapping LLMs to DS
Enhance LLM Training&Inference via Natural Annealing
Original LLMs
GPT2, GPT2-M, OPT1.3B...
Training
Back Propagation
GPUs
CUDA cores
Add, Sub, Mult operations...
Training cost 10M h+20T J
Expensive Inference
Sec 3.2-3.3
Training
Sec 3.4
Map to Hamiltonian Energy Landscape
Electronic-Current-Loss
DS machine
Natural Annealing
1,090× speedup
102,559× Energy Save
$\hat{F} = -\sum_{i,j} f(q_i,k_j) y_i v_j + \sum_i^N \frac{1}{2} \sum_j f(q_i,k_j) y_i^2$
$\mathcal{H} = -\sum_{i \neq j}^N J_{ij}\sigma_i\sigma_j + \sum_i^N h_i \sigma_i^2,\ \sigma_i \in \mathbb{R}$
$\Delta J_{ij} = \frac{\partial L_{EC}}{\partial J_{ij}} = \frac{\partial L_{EC}}{\partial I_L^i}\frac{\partial I_L^i}{\partial J_{ij}} = 2I_L^i x_j$
$\Delta h_i = \frac{\partial L_{EC}}{\partial h_i} = \frac{\partial L_{EC}}{\partial I_L^i}\frac{\partial I_L^i}{\partial h_i} = -2I_L^i \hat{y}_i$
Node Control Unit
Column Select Unit
Column Programming Unit

Figure 1: Overview of DS-LLM.

A-Q1

ing on graph learning problems including traffic predictions in (Pan et al., 2023), air quality, taxi demand, and pandemic progression in (Wu et al., 2024). DS machines are implemented through CMOS-compatible analog processors that operate at room temperature. Previous work on graph learning problems has shown over $1,000\times$ speedup and more than $100,000\times$ energy reduction, while achieving even better accuracy compared to state-of-the-art (SOTA) Graph Neural Networks (GNNs) on commercial GPUs.

B-W3

Then, is it feasible to leverage DS machines to accelerate LLMs? Unfortunately, no existing work provides a solution for mapping existing models like LLMs onto DS machines. The prior works like (Wu et al., 2024) ignore the architecture of mature neural networks and directly fit the data to the energy landscape of DS machine. For highly complex tasks like natural language processing (NLP), the energy landscape of a DS model would be much more intricate to fit the data distribution. Building and learning such an energy landscape could be extremely challenging-can we leverage existing, well-researched LLMs to help guide this process? In other words, can we equivalently perform neural networks like LLMs on DS machines using the powerful Natural Annealing process?

To this end, we propose the Dynamical System-based Large Language Model (DS-LLM), which serves as an algorithmic framework to bridge LLMs and DS machines. DS-LLM constructs the energy landscape of DS machines based on a reference LLM, enabling them to reproduce forward outputs and harness the power of Natural Annealing for LLM inference. For training, we introduce an Electric-Current-Loss based online training approach that allows LLM training on the same hardware. This method directly leverages the rapidly evolving electric current in DS machines, which naturally acts as gradients to continuously optimize parameters, thereby instantly shaping the energy landscape. The equivalence between DS-LLM and traditional LLMs, for both inference (Section 3.2-3.3) and training (Section 3.4), is mathematically proven and empirically verified through evaluation on models ranging from GPT2-124M to Llama2-7B. Experimental results showcase a 1,238$\times$ speedup and a 116,563$\times$ energy reduction on training, a 127$\times$ speedup and a 37,545$\times$ energy reduction on inference, while maintaining consistent accuracy.

To our knowledge, this is the first work to harness the computing power of DS machines to accelerate existing models, particularly LLMs. The main contributions of this work are summarized as follows:

- We propose DS-LLM, the first algorithm framework to bridge LLMs to existing DS machines, unlocking the remarkable power of Natural Annealing to equivalently perform LLMs on DS machines with theoretical and empirical proof.
- We propose an online continuous training method to achieve rapidly backpropagation on DS machines, significantly enhance the training speed and energy efficiency.

2 Background and Preliminary Knowledge

2.1 Dynamical-System-based Machine

(1) Hamiltonian and Natural Annealing: Dynamical systems implicitly incorporate an energy-based DS model and its energy function is called **Hamiltonian**, a fundamental concept in physics to describe the total energy of a system. The Hamiltonian of the backbone DS machine (Wu et al., 2024) is augmented from the Ising model which is a statistical physics model widely used in the modeling of interacting spins. The Hamiltonian is as below:

$$\mathcal{H} = -\sum_{i\neq j}^{N} J_{ij}\sigma_i\sigma_j + \sum_{i}^{N} h_i\sigma_i^2,\ \sigma_i \in \mathbb{R} \tag{1}$$

where σ are system spins, J are coupling parameters representing the correlations among spins, and h are spins' self-reaction intensity to external influence. While retaining the strengths of the Ising model, this new Hamiltonian lifts its binary constraint to support real values, achieving high performance on graph learning problems.

The computing power of DS machines stems from the process of **Natural Annealing**, an inherent characteristic of dynamical systems. In systems such as interacting spin models, the Hamiltonian naturally decreases due to spin interactions. From a physics perspective, this occurs because spins tend to settle into lower energy states, guiding the system toward optimal solutions. To harness Natural Annealing, the parameters J and h are programmed based on the target problem, shaping the Hamiltonian's energy landscape to align the desired outcomes with its minimum states. As the programmed DS machine initiates Natural Annealing from random initial conditions, the system rapidly converges to an energy minimum, with the spin dynamics stabilizing at the target results.

(2) DS machine Hardware: Since programming interacting spins can be prohibitively expensive, the backbone DS machine (Wu et al., 2024) is built on electronic dynamical systems, which are implemented using current CMOS technology.

As Fig. 2 shows, corresponding to equation 1, each spin σ_i is implemented as a nano-scale capacitor within a node unit (N_i), with its voltage representing spin value. Each capacitor is coupled with a programmable resistor, forming a feedback loop within the node unit that serves as the self-reaction parameters h. Capacitors from different node units ($N_i \& N_j$) are structurally connected by a programmable resistor in coupling unit CU_{ij} to perform the coupling parameters J. Natural Annealing in this system is driven by voltage imbalances across capacitors, which guide the natural movement of electrons toward equilibrium, propelling the system to evolve toward energy minima.

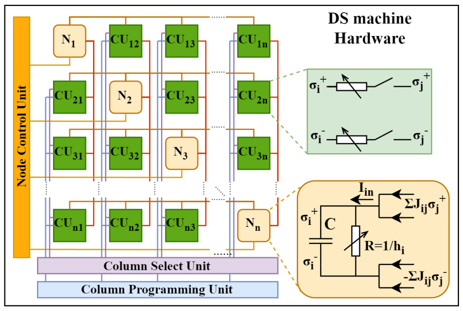

Figure 2: The backbone DS machine Hardware.

The convergence of the Natural Annealing, in other words, the spontaneous energy decrease of the Hamiltonian with time ($d\mathcal{H}/dt \leq 0$) can be theoretically guaranteed by Lyapunov stability analysis (Blaquiere, 2012). As Fig. 2 shows, for each spin σ_i, its electrodynamics behavior is designed as:

$$\frac{d\sigma_i}{dt} = \sum_{i\neq j}^{N} J_{ij}\sigma_j - h_i\sigma_i = -\frac{1}{2}\frac{\partial \mathcal{H}}{\partial \sigma_i} \tag{2}$$

Then, following the chain rule we have:

$$\frac{d\mathcal{H}}{dt} = \sum_i \frac{\partial \mathcal{H}}{\partial \sigma_i}\frac{d\sigma_i}{dt} = -\frac{1}{2}\sum_i (\frac{\partial \mathcal{H}}{\partial \sigma_i})^2 \leq 0 \tag{3}$$

As the equations show, the system's electrodynamics inherently drive the Hamiltonian toward a local minimum. To escape local minima and achieve better solutions, several techniques can be employed such as spin-flipping and noise injection (Afoakwa et al., 2021).

2.2 Large Language Models

In this paper, LLMs refer to pre-trained Transformer models with billions of parameters, such as GPT, Llama, Gemini, and Claude, which are among the most popular models today. Since the Transformer model was introduced in (Vaswani, 2017), it has been successfully developed and has demonstrated impressive performance not only in natural language processing (NLP) tasks but also in areas such as computer vision (Wang et al., 2023) and audio processing (Ghosal et al., 2023).

While modern LLMs are evolving toward multi-modal capabilities to handle tasks with mixed information, this work focuses on the implementation of the classic Transformer model as an early-stage exploration.

A classic Transformer consists of both an encoder and a decoder, whereas mainstream commercial models like GPT are decoder-only. Despite this distinction, the key components of both encoder and decoder are similar: multi-head self-attention, feed-forward Multilayer Perceptrons (MLPs), linear projection layers, and layer normalization. In this work, we focus on the implementation of decoder-based models, such as GPT-2 and OPT, with methods that can be adapted to other Transformer variants.

While matrix multiplication remains a key computing demand in LLMs, like in CNNs, the bottleneck of computing is the attention layer. Multi-head self-attention contributes to the impressive performance of LLMs at the cost of extremely high computational requirements. Considering each token x_i as a row vector, the break down of multi-head self-attention of each head is:

$$q_i = x_i W_Q,\ k_i = x_i W_K,\ v_i = x_i W_V \tag{4}$$

$$f(q_i, k_j) = exp(\frac{q_i k_j^T}{\sqrt{d}}) \tag{5}$$

$$A_i = \sum_{j=1}^{N} \frac{f(q_i, k_j) v_j}{\sum_{j=1}^{N} f(q_i, k_j)} \tag{6}$$

where W_Q, W_K, and W_V are learned weight matrices, and q_i, k_i, v_i are the row vectors representing the *query*, *key*, and *value* for token x_i, respectively. Function $f(q_i, k_j)$ measures the similarity between query and key. Consequently, the computation in the attention layer involves a weighted sum, along with matrix multiplications and an exponential function. Based on the above decomposition, the next chapter will demonstrate how the Transformer model can be implemented on DS machines using Natural Annealing.

3 DS-LLM: Mapping LLM onto DS machine

3.1 Overview

In this chapter, we present a detailed, step-by-step guide on leveraging DS machines to enhance LLMs on both inference and training. As illustrated in Fig. 1, while traditional transformer models are executed on digital GPUs using CUDA and tensor cores, DS-LLM maps the original model into the energy landscape of DS machines, leveraging Natural Annealing for the forward pass. Additionally, whereas conventional transformers are trained offline using backpropagation, DS-LLM introduces an online **Electric-Current-Loss** based training method to implement rapid backpropagation. Thus, both inference and training of LLM are accelerated by harnessing the immense computing power of DS machines.

In the following subsections, we will first introduce the key method to map existing models to the energy landscape of DS machines, followed by a further extension of it to implement the entire transformer model with minor augmentations based on existing DS machines. Finally, we propose the Electric-Current-Loss training method and explain how it works as an online rapid back-propagation.

3.2 Mapping Method

The mapping of existing models to DS machines can follow two different approaches: one based on the Hamiltonian level and the other on the electrodynamics level. To illustrate, we begin with a single linear layer. Consider the input matrix $X \in \mathbb{R}^{n \times d_{in}}$, the output matrix $Y \in \mathbb{R}^{n \times d_{out}}$, and the weight matrix $W \in \mathbb{R}^{d_{out} \times d_{in}}$, along with an optional bias vector $b \in \mathbb{R}^{d_{out}}$. Since the bias term b can be incorporated into the matrix multiplication operation, we will omit it for simplicity. Thus, the function of this layer can be written as a single matrix multiplication $Y = XW^T$.

On the Hamiltonian level, to leverage the Natural Annealing process on DS machines, we must shape the energy landscape so that its minimum energy state corresponds to the desired output. To achieve this, we define a target function F that minimizes the squared Frobenius norm of the difference (or Euclidean distance) between the output state of the DS machine, Y_{DS}, and the desired output, $X_{\text{DS}}W^T$. This forms the following minimization problem:

$$F = \|Y_{\text{DS}} - X_{\text{DS}}W^T\|_F^2 = \sum_{i,l}(y_{il} - \sum_j w_{ij}x_{jl})^2 \tag{7}$$

where w, x, and y represent elements of the matrices W, X_{DS}, and Y_{DS}, respectively, and i,j, and l correspond to the dimensions n,d_{in}, and d_{out}. The target function F reaches its minimum, $F_{min} = 0$, when $Y_{\text{DS}} = X_{\text{DS}}W^T$. Since the absolute value of F_{min} is not crucial, and $\sum_j w_{ij}x_{jl}$ remains constant during inference, we can simplify the target function by eliminating its second-order term:

$$\hat{F} = \sum_{(i,l)} y_{il}^2 - 2\sum_{(i,l)}(y_{il}\sum_j^N w_{ij}x_{jl}) \tag{8}$$

After transforming the linear layer's computation into a minimization problem, we program the DS machine by aligning its Hamiltonian with this simplified target function, $\hat{F}$. As shown in equation 8, $\hat{F}$ is a special case of the Hamiltonian described in equation 1. In this case, the spins σ are divided into two groups: x_{ij}, representing the input, and y_{il}, representing the output. The self-reaction parameters h_i are set to 1 for y_{il}, while the coupling parameters J are assigned $2w_{ij}$ between corresponding spins x_{jl} and y_{il}, with all other elements set to 0. The value of h_i for x_{ij} does not affect the convergence in equation 3 because the input x_{ij} is fixed, resulting in $dx_{ij}/dt = 0$.

This configuration maps the target function $\hat{F}$ to the Hamiltonian of the DS machine, enabling the computation of this layer via Natural Annealing process. During this process, the Hamiltonian will continuously decrease until it reaches a minimum (which is also the minimum of $\hat{F}$), where $Y_{\text{DS}} = X_{\text{DS}}W^T$ naturally emerges.

On the electrodynamics level, we can arrive at a similar conclusion. As seen in equation 2, the electrodynamics behavior is governed by the coupling parameters J and self-reaction parameters h. According to Lyapunov stability analysis (Blaquiere, 2012), all spins should stabilize at a specific value when the system reaches a stable point, i.e., a local minimum. Hence, the electrodynamics of all spins must satisfy a boundary condition where $d\sigma/dt = 0$. Dividing the spins σ into two groups, as before—input x and output y —along with the boundary condition, we arrive at:

$$y_i = \frac{\sum_j^N J_{ij}x_j}{h_i} \tag{9}$$

In this scenario, the spin electrodynamics exhibit a solvable stable point, allowing us to directly program J and h to map the desired matrix multiplication results onto the spin dynamics. Evidently, this leads to the same mapping setup as in the Hamiltonian-level analysis.

It is a fortunate coincidence that the backbone DS machine is naturally compatible with matrix multiplication, one of the key computational demands in LLMs. However, when extending this mapping method to more general functions, such as other operations in LLMs, the backbone DS machine requires slight augmentation to support them.

3.3 TRANSFORMER IMPLEMENTATION

Based on the proposed mapping method, we then analyze the implementation of key components besides matrix multiplication in transformer, and finally the whole model.

Recall the decomposition of self-attention layer, except linear projection layer in equation 4 which is already implemented, there are still three key operations in attention layer left: a) Query-Key matrix multiplication; b) Exponential function; c) Weighted-Sum operation, where a and b are shown in equation 5, and c is shown in equation 6. Fig. 3 illustrates the implementation of these three key operations, with detailed explanations provided below. The previously introduced implementation of linear layer (also the matrix multiplication operation between weights and features) is also shown in the figure as a reference.

a) Query-Key Matrix Multiplication: Unlike the typical multiplication in a linear layer, this operation occurs between two feature matrices generated online, rather than between features and weights.

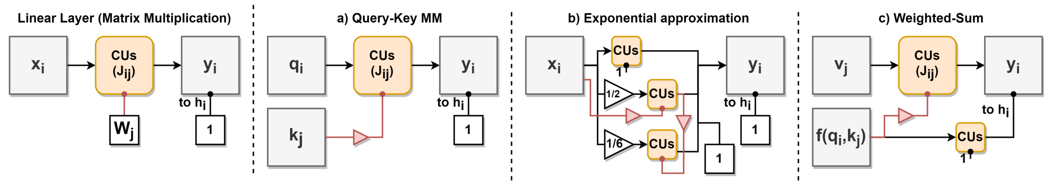

Figure 3: Implementation of key operations.

In the linear layer analysis, we assumed the weight matrix W is obtained offline and loaded onto programmable resistors in the coupling units. Fortunately, these resistors in the backbone DS machine are implemented using transistors, and programming them is accomplished by adjusting the voltage on their control ports—a common technique in CMOS design. Meanwhile, the spins in DS machines are represented by capacitors, where the voltage corresponds to the spin value. Thus, we can map the feature matrix K onto the programmable resistors by connecting the output voltage of the spins to the control ports of the resistors with necessary scaling circuits.

b) Exponential approximation: The exponential function is computationally expensive and requires a high-order Taylor expansion for approximation. Based on a pre-experiment, we explore the trade-off between model accuracy and hardware cost and select a 3rd-order Taylor expansion as an approximation. Details of this trade-off can be found in Appendix.

$$exp(x_i)_{\text{Taylor3}} = 1 + x + 1/2x^2 + 1/6x^3 \tag{10}$$

As Fig. 3 shows, both the second and third order terms are implemented in the same way as the Query-Key Matrix Multiplication. We will show the experiments results later in evaluation part to demonstrate that this approximation achieves comparable or even superior performance to the original exponential function.

c) Weighted-Sum: Similar to the matrix multiplication, we can build the target function:

$$F = \sum_i (y_i - \frac{\sum_j f(q_i,k_j)v_j}{\sum_j f(q_i,k_j)})^2 = \sum_i (\frac{\sum_j f(q_i,k_j)y_i - \sum_j f(q_i,k_j)v_j}{\sum_j f(q_i,k_j)})^2 \tag{11}$$

Since the probability of $\sum_j f(q_i,k_j)$ keeping zero is negligible in an evolving dynamical system, we multiply it on equation 11 and reduce the constant terms:

$$\hat{F} = -\sum_{i,j} f(q_i,k_j)y_i v_j + \sum_i^N \frac{1}{2}\sum_j f(q_i,k_j)y_i^2 \tag{12}$$

In this setup, we program $f(q_i,k_j)$ to J_{ij} like in the Query-Key Matrix Multiplication. Notice that $\frac{1}{2}\sum_j f(q_i,k_j)$ can be regarded as a matrix multiplication operation between $f(q_i,k_j)$ and an all-ones vector, which can be mapped to DS machine as a simplified linear layer. The results are connected to the programmable resistors in node units, where $h_i = \frac{1}{2}\sum_j f(q_i,k_j)$ for y_i, and set $h_i = 0$ for v_j. With this setup, the Weighted-Sum operation in equation 6 can also be mapped to the Hamiltonian described in equation 1.

At this point, we have most of the essential components of a transformer model. The other operations which have relatively lower computing demanding like LayerNorm and activation are handled by auxiliary functional units. Details of other operations can be found in the appendix.

Next, we demonstrate how to map the entire transformer model onto a DS machine. Assuming the original model consists of multiple layers, each decomposed into P operations, its function can be expressed as:

$$y = f^{(P)} \circ f^{(P-1)} \circ \cdots \circ f^{(2)} \circ f^{(1)}(x) \tag{13}$$

Since we have confirmed that the mapping method works for each computational component, we can now construct the general target function as follows:

$$\hat{F}^{(p)} = (x^{(p+1)})^2 - 2x^{(p+1)}f^{(p)}(x^{(p)}) \tag{14}$$

Here, we combine the mapping of all individual target functions by using the output of each lower layer as the input to the higher layer, where x^1 is the initial input and x^{P+1} is the final output. It's important to note that when combining them, the influence between spins is unidirectional—from lower to higher layers—i.e., $\partial\hat{F}^{(p+1)}/\partial x^{(p)} = 0$. Therefore, the Natural Annealing process converges in each operation, ensuring that $\hat{F}^{(P)}$ reaches its minimum when $x^{(P+1)} = f^{(P)} \circ f^{(P-1)} \circ \cdots \circ f^{(1)}(x^1)$.

In this setup, the entire transformer-based model can be mapped onto DS machines. With global natural annealing, the model achieves the desired output when the system reaches its global energy minimum.

3.4 Online Training with Electric-Current-Loss

After accelerating the forward pass of LLMs using Natural Annealing, we go further to enhance the training of LLMs. The core idea is that a perfectly trained DS machine should reach the energy minimum when its output spins match the ground truth from the training data. Based on our electrodynamics analysis of the mapping method, we propose an Electric-Current-Loss (ECL), leveraging the physical currents in DS machine to achieve rapid online training.

As illustrated in Fig. 4, for a linear layer, when Natural Annealing converges, for each output spin y_i, the total incoming electric current $I_{\text{in}}^i = \sum_j^N J_{ij}x_j$ at the node unit must balance the internal current $I_{\text{R}}^i = h_i y_i$ flowing through the resistor within the same node, ensuring that the capacitor voltage (representing the spin values) remains stable ($dy_i/dt = 0$). Referring to equation 9, if we map the ground truth output $\hat{y}_i$ onto the output spins and fix their values, we can define the internal current reference as $\hat{I_{\text{R}}^i} = h_i\hat{y}_i$. The difference between the incoming current I_{in}^i and the reference current $\hat{I_{\text{R}}^i}$ forms a new loss function, expressed as $L_{\text{EC}} = \sum_i^N (I_{\text{in}}^i - \hat{I_{\text{R}}^i})^2$. This difference is equal to the current through the sampling resistor r, $I_{\text{L}}^i = I_{\text{in}}^i - I_{\text{R}}^i$. We then update the parameters J_{ij} (notice h_i = 1 is constant) using gradient descent:

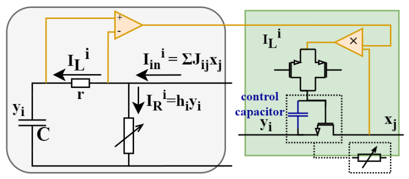

Figure 4: The feed back path of Electric-Current-Loss training.

$$\Delta J_{ij} = \frac{\partial L_{\text{EC}}}{\partial J_{ij}} = \frac{\partial L_{\text{EC}}}{\partial I_{\text{L}}^i}\frac{\partial I_{\text{L}}^i}{\partial J_{ij}} = 2I_{\text{L}}^i x_j \tag{15}$$

As depicted in Fig. 4, the multiplication of the loss current I_{L}^i by the spin values x_j or $\hat{y}_i$ can be implemented at the circuit level. The resulting current is then fed back to the control port of the programmable resistors in the coupling units (CUs) and nodes. Consequently, the parameters are updated as $J_{ij} \rightarrow J_{ij} - \Delta J_{ij}\Delta t$ by integrating the result current on the control capacitor of the programmable resistors over a time interval Δt.

We propose this training method, named Electric-Current-Loss (ECL) online training, which aims to minimize the loss function L_{EC}. Currently, this method only supports single-layer DS machines, where a ground truth $\hat{y}_i$ is available. It's worth noting that brain-inspired methods, such as predictive coding (Song et al., 2020), have been proposed to generate internal reference values for each layer, providing a lot of opportunities for extending ECL to multi-layer DS machines. However, in this early-stage research, we focus on conventional backpropagation training to provide a baseline on how to combine ECL with backpropagation to enable efficient LLM training on DS machines.

In the training of most LLMs, the loss in the output layer is typically computed using softmax with cross-entropy, yielding $\partial L/\partial x_i = y_i - \hat{y}_i$, where x_i represents the input logits. Referring to equation 9 and equation 14, after the DS machine completes the Natural Annealing process, the output spin value y_i should converge to the desired computational result $f^{(P)}(x^{(P)})$. If we map the ground truth $\hat{y}_i$ to the output spin of the final layer, we obtain:

$$\frac{\partial L}{\partial x_i} = y_i - \hat{y}_i = \frac{I_{\text{in}}^i - I_{\text{R}}^i}{h_i} = I_{\text{L}}^i \tag{16}$$

Thus, the gradients of the logits can be realized as an electric current using the ECL training method. As shown in Fig. 5, the gradients in other layers are calculated by the activations and the electric current loss from the later layer. Since the gradient of a matrix multiplication operation is itself a matrix multiplication, this calculation can also be mapped

B-Q2

on DS machines. For some special non-polynomial operations, the calculation will be handled by additional auxiliary circuits. Additionally, the mapping of forward process is directional and thus the connections and CUs from later layers to previous layers are not utilized. Therefore we can leverage these CUs for the calculation of gradients without incurring extra cost. The gradients are then integrated on the controlling capacitors, updating the parameters. In this way, we combine ECL with backpropagation, creating an online training method that gets rid of output readout and leverages the fast forward computing of DS machines.

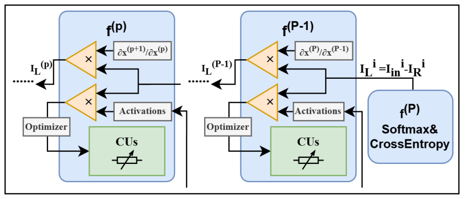
$f^{(p)}$
$\partial x^{(p+1)}/\partial x^{(p)}$
$I_L^{(p)}$
Activations
Optimizer
CUs
$f^{(P-1)}$
$\partial x^{(P)}/\partial x^{(P-1)}$
$I_L^{(P-1)}$
$I_L^i = I_{in}^i - I_R^i$
$f^{(P)}$
Softmax& CrossEntropy

Figure 5: ECL-based online backpropagation.

B-Q2

4 Evaluation

4.1 Model and Training Setup

Models: We apply DS-LLM to two open-source models of varying sizes: GPT-2 (base and medium) (Radford et al., 2019), OPT-1.3B and OPT-2.7B (Zhang et al., 2022), and Llama2-7B(Touvron et al., 2023).

Tasks and Datasets: For all three models, we fine-tune and evaluate on five datasets from the GLUE benchmark (Wang et al., 2019): SST-2 (Socher et al., 2013) for Single-Sentence Tasks, MRPC (Dolan & Brockett, 2005) and QQP (DataCanary et al., 2017) for Paraphrase Tasks, QNLI (Rajpurkar et al., 2016) and RTE (Dagan et al., 2006) for Inference Tasks. Additionally, we pre-train GPT-2-medium from scratch on OWT (Gokaslan & Cohen, 2019) to evaluate the impact of the DS-LLM attention approximation.

DS machine evaluation: Although the backbone DS machine chip has been manufactured and provides some public accessibility, this work requires additional augmentations. Therefore, we employ a CUDA-based Finite Element Analysis (FEA) software emulator to simulate the Natural Annealing process of DS machines. Upon publication, we plan to open-source the CUDA-based emulator to support open research and design. Additionally, the power consumption are evaluated using the Cadence Mixed-Signal Design Environment, with 45 nm CMOS technology.

Experiments Setup: The fine-tuning on the GLUE benchmark was conducted on 4 Nvidia A100 40GB GPUs. The global batch size was set to 32 for GPT-2 (124M), 16 for GPT-2-medium (355M) and OPT-1.3B, and 8 for OPT-2.7B and Llama2-7B. We fine-tuned all models for 2 epochs on QQP and 3 epochs on the other datasets. The optimizer used was AdamW with an initial learning rate of 2e-5, and all other parameters were kept as default, as provided by the Hugging Face Transformers library. For pre-training GPT-2-medium, we utilized 80 Nvidia A100 40GB GPUs with a global batch size of 480 and trained the model for 120,000 iterations. The optimizer was AdamW with a 6e-4 initial learning rate, and other parameters were also kept at default settings. The inference experiments are using the same global batch size as in the fine-tuning experiments.

4.2 Accuracy Comparison between DS-LLM and Original LLMs

In order to verify the accuracy loss of the proposed mapping method (DS-LLM) and training method (ECL), we fine-tune and evaluate the models on selected datasets to compare the accuracy across three model types: a) the original LLMs, trained offline and inferred on GPUs; b) DS-LLM models, trained offline on GPUs but inferred on DS machines; and c) DS-LLM-ECL models, trained online and inferred on DS machines. As shown in Table 1, the accuracy of DS-LLM models is comparable to that of the original LLMs, with some even outperforming them, demonstrating that the approximation loss during mapping is minimal and the mapping method is effective. The DS-LLM-ECL models also maintains good accuracy, validating the feasibility of the ECL online training method.

We further validate the training behavior and convergence speed of the proposed DS-LLM mapping method and the DS-LLM-ECL online training approach. The fine-tuning trajectories of the original LLMs, DS-LLM, and DS-LLM-ECL models are visualized in Fig. 6. Additionally, we present the training curves of the original LLMs and DS-LLM during GPT-2 Medium pretraining; unfortunately, we are unable to pretrain DS-LLM-ECL models due to computational limitations. Notably, both DS-

Table 1: Accuracy comparison (in Accuracy (%)): the higher the better.

C-W1

D-Q3

Task	Paraphrase Tasks		Inference Tasks		Single-Sentence Tasks
Dataset	MRPC	QQP	RTE	QNLI	SST2
GPT2	75.00	88.43	63.18	88.03	91.63
GPT2-DS	76.72	89.04	63.54	88.28	91.29
GPT2-DS-ECL	76.91	89.24	63.11	87.94	90.82
GPT2-M	79.66	90.57	68.59	91.05	93.58
GPT2-M-DS	79.17	90.55	70.40	90.33	93.35
GPT2-M-DS-ECL	79.31	90.09	70.41	89.73	93.06
OPT1.3B	84.07	90.94	77.26	91.09	92.43
OPT1.3B-DS	87.01	88.48	78.70	91.41	92.20
OPT1.3B-DS-ECL	86.57	88.59	78.05	91.45	91.81
OPT2.7B	86.52	91.03	82.33	93.15	94.08
OPT2.7B-DS	86.54	90.98	82.48	93.27	94.04
OPT2.7B-DS-ECL	86.74	90.67	82.44	93.41	94.25
Llama2-7B	90.01	91.10	88.45	95.75	96.58
Llama2-7B-DS	89.47	90.95	88.70	95.69	96.32
Llama2-7B-DS-ECL	89.56	91.08	88.57	95.73	95.79

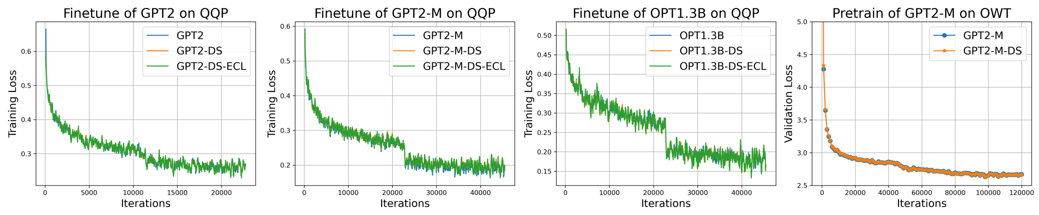

Figure 6: Visualization of training trajectories.

Table 2: Performance comparison on training time and energy consumption.

C-W1

D-Q3

Metric	Training time (s)					Energy Consumption (J)				
Dataset	MRPC	QQP	RTE	QNLI	SST2	MRPC	QQP	RTE	QNLI	SST2
GPT2	5.17E+1	2.12E+3	4.56E+1	8.32E+2	3.07E+2	4.14E+4	1.69E+6	3.65E+4	6.66E+5	2.46E+5
GPT2-DS-ECL	4.44E-2	4.37E+0	3.00E-2	1.26E+0	8.04E-1	3.77E-1	3.71E+1	2.55E-1	1.07E+1	6.83E+0
GPT2-M	1.62E+2	9.00E+3	1.55E+2	3.42E+3	1.32E+3	1.30E+5	7.20E+6	1.24E+5	2.74E+6	1.05E+6
GPT2-M-DS-ECL	9.25E-2	9.10E+0	6.25E-2	2.63E+0	1.68E+0	7.86E-1	7.74E+1	5.31E-1	2.23E+1	1.42E+1
OPT1.3B	1.78E+2	2.42E+4	1.77E+2	6.46E+3	2.83E+3	1.42E+5	1.94E+7	1.42E+5	5.17E+6	2.27E+6
OPT1.3B-DS-ECL	2.22E-1	2.18E+1	1.50E-1	6.30E+0	4.02E+0	1.89E+0	1.86E+2	1.28E+0	5.36E+1	3.42E+1
OPT2.7B	3.90E+2	7.34E+4	3.71E+2	2.02E+4	1.20E+4	3.12E+5	5.87E+7	2.97E+5	1.61E+7	9.59E+6
OPT2.7B-DS-ECL	4.81E-1	4.73E+1	3.25E-1	1.37E+1	8.71E+0	4.09E+0	4.02E+2	2.76E+0	1.16E+2	7.40E+1
Llama2-7B	1.55E+3	2.17E+5	1.49E+3	6.27E+4	3.46E+4	1.24E+6	1.73E+8	1.19E+6	5.02E+7	2.77E+7
Llama2-7B-DS-ECL	1.22E+0	1.20E+2	8.25E-1	3.47E+1	2.21E+1	1.04E+1	1.02E+3	7.01E+0	2.95E+2	1.88E+2

Table 3: Performance comparison on inference time and energy consumption.

C-W1

D-Q3

Metric	Inference time (s)					Energy Consumption (J)				
Dataset	MRPC	QQP	RTE	QNLI	SST2	MRPC	QQP	RTE	QNLI	SST2
GPT2	1.38E-1	1.17E+1	1.43E-1	1.84E+0	2.84E-1	1.10E+2	9.37E+3	1.14E+2	1.47E+3	2.27E+2
GPT2-DS	2.04E-3	4.69E-1	3.60E-3	6.48E-3	2.16E-3	5.51E-3	1.27E+0	9.72E-3	1.75E-2	5.83E-3
GPT2-M	4.46E-1	3.79E+1	3.23E-1	5.42E+0	8.75E-1	3.57E+2	3.03E+4	2.58E+2	4.34E+3	7.00E+2
GPT2-M-DS	4.25E-3	9.78E-1	7.50E-3	1.35E-2	4.50E-3	1.15E-2	2.64E+0	2.03E-2	3.65E-2	1.22E-2
OPT1.3B	6.28E-1	6.28E+1	6.86E-1	1.07E+1	1.23E+0	5.02E+2	5.03E+4	5.49E+2	8.56E+3	9.82E+2
OPT1.3B-DS	1.02E-2	2.35E+0	1.80E-2	3.24E-2	1.08E-2	2.75E-2	6.33E+0	4.86E-2	8.75E-2	2.92E-2
OPT2.7B	1.94E+0	2.05E+2	2.11E+0	3.24E+1	4.01E+0	1.55E+3	1.64E+5	1.69E+3	2.60E+4	3.21E+3
OPT2.7B-DS	2.21E-2	5.08E+0	3.90E-2	7.02E-2	2.34E-2	5.97E-2	1.37E+1	1.05E-1	1.90E-1	6.32E-2
Llama2-7B	6.35E+0	6.28E+2	6.98E+0	1.09E+2	1.35E+1	5.08E+3	5.02E+5	5.58E+3	8.76E+4	1.08E+4
Llama2-7B-DS	0.0561	12.903	0.099	0.1782	0.0594	1.51E-1	3.48E+1	2.67E-1	4.81E-1	1.60E-1

LLM and DS-LLM-ECL exhibit convergence curves that closely match those of the original models, demonstrating that the mapping and online training methods effectively replicate the performance of traditional LLMs on DS machines.

4.3 Performance of DS-LLM over LLMs on GPU

Table. 2 compares the total training time on the whole dataset and estimated energy consumption between the original model on GPU and DS-LLM-ECL on the DS machine. On average across all fine-tuning tasks, DS-LLM-ECL achieves a 1,238× speedup and a 116,563× reduction in energy consumption. The power consumption of the 4 A100 40GB GPUs is estimated at 800 watts, while the DS machine consumes approximately 8.5 watts.

B-Q3

Table. 3 shows the comparison of total inference time on the whole evaluation dataset and energy consumption between the original LLMs on GPU and DS-LLM on the DS machine. On average across all inference tasks, DS-LLM delivers a 127× speedup and a 37,545× reduction in energy consumption. During inference, the DS machine primarily performs Natural Annealing, consuming significantly less energy than during training, resulting in a power consumption of just 2.7 watts.

The performance of GPUs are evaluated based on naive implementation without special optimization technologies. There are lots of GPU optimization works like vLLM can obviously improve the performance of GPU.

D-Q4-2

Table 4: Comparison on Llama2-7B with low-precision CPU and edge devices.

Solutions	Token Generation Rate (tokens/s)	Energy Efficiency (tokens/KWh)
Low Precision CPUs(Shen et al., 2023)	45.4	1.63E+6
Cambricon-LLM(Yu et al., 2024)	3.55	3.60E+6
DS-LLM (this work)	3.03E+4	4.04E+10

B-W1

4.4 Comparison with low-precision implementations for edge devices

For a more comprehensive evaluation on inference, we also compared our DS-LLM with some low-precision implementations for edge devices, including accelerating LLM inference on low-precision CPU(Shen et al., 2023) and a new edge device Cambricon-LLM(Yu et al., 2024) on Llama2-7B model. The power of the CPU work is estimated as 100W, which is lower than any specific CPU they used. The results show that DS-LLM achieves orders of magnitudes higher token generation rate and energy efficiency than the references. This is because both low-precision CPUs and edge devices rely on digital computing paradigms that execute step-by-step, instruction-based operations. In contrast, DS machines operate without the need for micro-instructions, leveraging continuous natural annealing through analog currents. This enables DS machines to achieve exceptional speed and energy efficiency.

5 Related Work

The early-stage research on DS machines is rooted in the Ising model, which supports only binary spin values and primarily addresses binary optimization problems (Afoakwa et al., 2021). Our backbone DS machine was proposed in NPGL (Wu et al., 2024) and applied to graph learning problems. However, NPGL employs an individual learning method that ignores the architecture of Neural Networks, limiting its ability to leverage existing technologies effectively.

There are several variants of dynamical systems, such as optical (Inagaki et al., 2016) and oscillator-based (Lo et al., 2023) Ising machines. Although these approaches show promise, they have yet to be successfully integrated into machine learning applications. To the best of our knowledge, this work represents the first attempt to combine existing machine learning models, particularly Large Language Models (LLMs), with DS machines.

6 Conclusion

In this work, we introduce DS-LLM, the first algorithmic framework that bridges LLMs to existing DS machines, harnessing the power of Natural Annealing to efficiently execute LLMs on DS hardware. For training, we propose the ECL-based online training method, enabling DS-LLM to be trained on the same hardware used for inference. The mathematical equivalence between DS-LLM and traditional LLMs is proven and validated through experiments on models from GPT-2 to OPT-1.3B. Results demonstrate consistent accuracy while achieving a 1,238× speedup and 116,563× energy savings during training, along with a 127× speedup and 37,545× energy reduction in inference. In conclusion, DS-LLM presents a promising new solution for the community, with significant opportunities for further exploration in future studies.

7 Reproducibility Statement

To ensure the reproducibility of this work, we have provided detailed descriptions of the experimental settings in Section 4.1. The CUDA-based Finite Element Analysis emulator used in this study is adapted from the one used in (Afoakwa et al., 2021). We also plan to open-source this emulator upon publication.

REFERENCES

Richard Afoakwa, Yiqiao Zhang, Uday Kumar Reddy Vengalam, Zeljko Ignjatovic, and Michael Huang. Brim: Bistable resistively-coupled ising machine. In 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA), pp. 749–760. IEEE, 2021.

Maxwell Anderson, Shi-Yuan Ma, Tianyu Wang, Logan Wright, and Peter McMahon. Optical transformers. Transactions on Machine Learning Research, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=Xxw0edFFQC.

Austin Blaquiere. Nonlinear system analysis. Elsevier, 2012.

Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In Joaquin Quiñonero-Candela, Ido Dagan, Bernardo Magnini, and Florence d'Alché Buc (eds.), Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment, pp. 177–190, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-33428-6.

DataCanary, hilfialkaff, Lili Jiang, Meg Risdal, Nikhil Dandekar, and tomtung. Quora question pairs, 2017. URL https://kaggle.com/competitions/quora-question-pairs.

William B. Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In Proceedings of the Third International Workshop on Paraphrasing (IWP2005), 2005. URL https://aclanthology.org/I05-5002.

Deepanway Ghosal, Navonil Majumder, Ambuj Mehrish, and Soujanya Poria. Text-to-audio generation using instruction-tuned llm and latent diffusion model, 2023. URL https://arxiv.org/abs/2304.13731.

Aaron Gokaslan and Vanya Cohen. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019.

Ryan Hamerly, Takahiro Inagaki, Peter L. McMahon, Davide Venturelli, Alireza Marandi, Tatsuhiro Onodera, Edwin Ng, Carsten Langrock, Kensuke Inaba, Toshimori Honjo, Koji Enbutsu, Takeshi Umeki, Ryoichi Kasahara, Shoko Utsunomiya, Satoshi Kako, Ken ichi Kawarabayashi, Robert L. Byer, Martin M. Fejer, Hideo Mabuchi, Dirk Englund, Eleanor Rieffel, Hiroki Takesue, and Yoshihisa Yamamoto. Experimental investigation of performance differences between coherent ising machines and a quantum annealer. Science Advances, 5(5):eaau0823, 2019. doi: 10.1126/sciadv.aau0823.

Dan Hendrycks and Kevin Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units, 2017. URL https://openreview.net/forum?id=Bk0MRI5lg.

Yang Hu, Xinhan Lin, Huizheng Wang, Zhen He, Xingmao Yu, Jiahao Zhang, Qize Yang, Zheng Xu, Sihan Guan, Jiahao Fang, Haoran Shang, Xinru Tang, Xu Dai, Shaojun Wei, and Shouyi Yin. Wafer-scale computing: Advancements, challenges, and future perspectives [feature]. IEEE Circuits and Systems Magazine, 24(1):52–81, 2024. doi: 10.1109/MCAS.2024.3349669.

Takahiro Inagaki, Yoshitaka Haribara, Koji Igarashi, Tomohiro Sonobe, Shuhei Tamate, Toshimori Honjo, Alireza Marandi, Peter L. McMahon, Takeshi Umeki, Koji Enbutsu, Osamu Tadanaga, Hirokazu Takenouchi, Kazuyuki Aihara, Ken ichi Kawarabayashi, Kyo Inoue, Shoko Utsunomiya, and Hiroki Takesue. A coherent ising machine for 2000-node optimization problems. Science, 354(6312):603–606, 2016. doi: 10.1126/science.aah4243. URL https://www.science.org/doi/abs/10.1126/science.aah4243.

Iordanis Kerenidis, Natansh Mathur, Jonas Landman, Martin Strahm, Yun Yvonna Li, et al. Quantum vision transformers. Quantum, 8:1265, 2024.

Hao Lo, William Moy, Hanzhao Yu, Sachin Sapatnekar, and Chris H Kim. An ising solver chip based on coupled ring oscillators with a 48-node all-to-all connected array architecture. Nature Electronics, 6(10):771–778, 2023.

Guy Ohayon, Theo Adrai, Michael Elad, and Tomer Michaeli. Reasons for the superiority of stochastic estimators over deterministic ones: robustness, consistency and perceptual quality. In Proceedings of the 40th International Conference on Machine Learning, ICML'23. JMLR.org, 2023.

Zhenyu Pan, Anshujit Sharma, Jerry Yao-Chieh Hu, Zhuo Liu, Ang Li, Han Liu, Michael Huang, and Tony Geng. Ising-traffic: Using ising machine learning to predict traffic congestion under uncertainty. Proceedings of the AAAI Conference on Artificial Intelligence, 37(8):9354–9363, Jun. 2023. doi: 10.1609/aaai.v37i8.26121. URL https://ojs.aaai.org/index.php/AAAI/article/view/26121.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. OpenAI blog, 2019.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text, 2016. URL https://arxiv.org/abs/1606.05250.

Anshujit Sharma, Richard Afoakwa, Zeljko Ignjatovic, and Michael Huang. Increasing ising machine capacity with multi-chip architectures. In Proceedings of the 49th Annual International Symposium on Computer Architecture, ISCA '22, pp. 508–521, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450386104. doi: 10.1145/3470496.3527414. URL https://doi.org/10.1145/3470496.3527414.

Haihao Shen, Hanwen Chang, Bo Dong, Yu Luo, and Hengyu Meng. Efficient llm inference on cpus, 2023. URL https://arxiv.org/abs/2311.00502.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In David Yarowsky, Timothy Baldwin, Anna Korhonen, Karen Livescu, and Steven Bethard (eds.), Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pp. 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://aclanthology.org/D13-1170.

Ruibing Song, Chunshu Wu, Chuan Liu, Ang Li, Michael Huang, and Tony Tong Geng. DS-GL: Advancing graph learning via harnessing nature's power within scalable dynamical systems. In 2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA), pp. 45–57, 2024. doi: 10.1109/ISCA59077.2024.00014.

Yuhang Song, Thomas Lukasiewicz, Zhenghua Xu, and Rafal Bogacz. Can the brain do backpropagation? — exact implementation of backpropagation in predictive coding networks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), Advances in Neural Information Processing Systems, volume 33, pp. 22566–22579. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/fec87a37cdeec1c6ecf8181c0aa2d3bf-Paper.pdf.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

Fengbin Tu, Zihan Wu, Yiqi Wang, Weiwei Wu, Leibo Liu, Yang Hu, Shaojun Wei, and Shouyi Yin. Multcim: Digital computing-in-memory-based multimodal transformer accelerator with attention-token-bit hybrid sparsity. IEEE Journal of Solid-State Circuits, 2023.

A Vaswani. Attention is all you need. Advances in Neural Information Processing Systems, 2017.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In International Conference on Learning Representations, 2019. URL https://openreview.net/forum?id=rJ4km2R5t7.

Wenhai Wang, Zhe Chen, Xiaokang Chen, Jiannan Wu, Xizhou Zhu, Gang Zeng, Ping Luo, Tong Lu, Jie Zhou, Yu Qiao, and Jifeng Dai. Visionllm: Large language model is also an open-ended decoder for vision-centric tasks. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), Advances in Neural Information Processing Systems, volume 36, pp. 61501–61513. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/c1f7b1ed763e9c75e4db74b49b76db5f-Paper-Conference.pdf.

Christopher Wolters, Xiaoxuan Yang, Ulf Schlichtmann, and Toyotaro Suzumura. Memory is all you need: An overview of compute-in-memory architectures for accelerating large language model inference, 2024. URL https://arxiv.org/abs/2406.08413.

Chunshu Wu, Ruibing Song, Chuan Liu, Yunan Yang, Ang Li, Michael Huang, and Tong Geng. Extending power of nature from binary to real-valued graph learning in real world. In The Twelfth International Conference on Learning Representations, 2024. URL https://openreview.net/forum?id=qT7DXUmX7j.

Zhongkai Yu, Shengwen Liang, Tianyun Ma, Yunke Cai, Ziyuan Nan, Di Huang, Xinkai Song, Yifan Hao, Jie Zhang, Tian Zhi, Yongwei Zhao, Zidong Du, Xing Hu, Qi Guo, and Tianshi Chen. Cambricon-llm: A chiplet-based hybrid architecture for on-device inference of 70b llm, 2024. URL https://arxiv.org/abs/2409.15654.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. Opt: Open pre-trained transformer language models, 2022.

A Appendix

A.1 Discussion

As an early-stage research endeavor, DS-LLM seeks to introduce a promising new computing paradigm to address the growing computational demands of LLMs. Although still in its infancy, this section aims to discuss and analyze its potential, challenges, and future directions.

a) Feasibility: First, we want to highlight that there are no fundamental challenges to integrate DS machines into existing computing systems. DS machines, though architecturally distinct from digital processors, are built using CMOS-compatible technology. This compatibility ensures they can be integrated seamlessly into existing systems as co-processors (similar to TPUs or NPUs) via interfaces like PCIe. Theoretically, no fundamental hardware adaptations are required.

A-Q2

From a system perspective, though this work is still in early stage, we agree that integration to the existing computing infrastructure is very promising as future exploration. There are a lot of exploration space on developing software tool-chain like compilers, optimizing memory management, and pipelining tasks between DS machines and other processors. With all these works and software tools help, we believe DS machines are inherently feasible for integration into digital systems and work as a new type of co-processor like GPUs / TPUs / NPUs. Future work can explore hybrid use cases that combine CPUs, GPUs, and DS machines based on their unique strengths to achieve optimal performance.

Overall, while DS machines are not yet mature, their fundamental feasibility paves the way for exciting opportunities in integration with existing computing infrastructures.

B-Q1 C-W2 D-Q3

b) Scalability: While prior work like NP-GL was designed for small graphs with fewer than 1,000 nodes, the capacity of DS machines can be significantly expanded to handle much larger scales. First, DS machines have demonstrated linear complexity with respect to the number of nodes (Song et al., 2024), making them inherently scalable with increased chip area. For context, NP-GL occupies only about 5 mm^2, whereas modern GPUs like the H100 have a die size of 814 mm^2, and wafer-scale chips—such as those with up to 46,255 mm^2—are emerging (Hu et al., 2024). Based on linear complexity, a single-chip solution could theoretically support millions of nodes within a single DS machine.

Second, for even larger models or faster training, multi-chip approaches offer a viable path forward. Existing research has explored multi-chip solutions for DS machines (Sharma et al., 2022), where individual chips perform annealing with periodic synchronization. We are also investigating promising techniques such as deploying models across multiple DS machine chips to achieve pipeline parallelism.

Overall, the scalability of DS machines is theoretically well-founded, offering both single-chip and multi-chip solutions to meet the demands of increasingly large and complex models.

B-Q1

c) Model Flexibility Issues: This work focuses on classic operations in Transformer-based LLMs, acknowledging that modern LLMs may include different operations such as varied activation functions or embedding methods. Despite the diversity of LLM architectures, these operations can generally be categorized as either polynomial or non-polynomial. Polynomial operations, which can be broken down into basic addition and multiplication, are directly mappable to DS machines. Non-polynomial operations, such as exponential function, require transformation into polynomial approximations (e.g., via Taylor expansion) or the addition of auxiliary circuits, which may slightly increase latency depending on their complexity. Fortunately, most high-computational-demand operations, particularly in attention layers and FFNs, are polynomial or even linear. Thus, DS machines offer high flexibility and adaptability for various models.

C-W2

d) Stability: The stability of DS machines is inherently promising due to their fundamental characteristics. Compared to modern GPUs, DS machines consume significantly less power, which reduces the risk of overheating. Additionally, as DS machines rely on a stochastic annealing process rather than deterministic computation, they are naturally more robust to noise and non-ideal factors (Ohayon et al., 2023). We further evaluate this opinion in next subsection with experimental results.

D-Q2-1

e) Robustness on Circuit Non-idealities: For a comprehensive discussion, we address two types of Non-idealities: dynamic noise (e.g., thermal noise) and static offset (e.g., fabricated non-linearity and mismatch).

Table 5: Influence of dynamic noise on accuracy of DS-LLM.

Model	Noise Level	MRPC	QQP	RTE	QNLI	SST2
gpt2-DS-ECL	0	76.91	89.24	63.11	87.94	90.82
gpt2-DS-ECL	0.05	76.84	89.20	63.02	87.83	90.77
gpt2-DS-ECL	0.10	76.55	88.93	62.71	87.56	90.33
gpt2-DS-ECL	0.15	75.89	88.12	62.05	86.98	89.84
OPT1.3B-DS-ECL	0	86.57	88.59	78.05	91.45	91.81
OPT1.3B-DS-ECL	0.05	86.43	88.52	77.99	91.40	91.75
OPT1.3B-DS-ECL	0.10	86.15	88.31	77.63	91.22	91.43
OPT1.3B-DS-ECL	0.15	85.67	87.75	77.24	90.89	91.01

Dynamic noise: It has been proved that stochastic process like the natural annealing in DS machines is usually more robust to noise than deterministic process(Ohayon et al., 2023). We setup an experiment to evaluate the impact of varying noise levels on DS-LLM, where two models are fine-tuned on the downstream datasets based on our work with dynamic noise injected into the simulation. The dynamic noise was modeled as standard Gaussian noise with a standard deviation ranging from 0.05 to 0.15. The results demonstrate that DS machines exhibit high robustness to dynamic noise.

Static offset: Static offset is a common challenge in analog circuits, arising from factors like hardware non-linearity and mismatches during fabrication. Our proposed online training method addresses this effectively by training and performing inference on the same hardware device. This

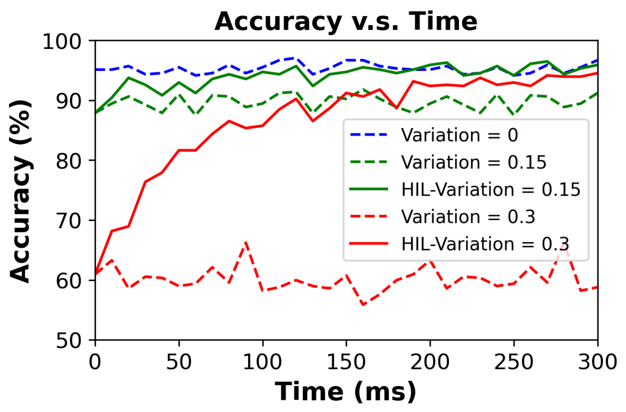

Figure 7: Accuracy recovery for a pre-trained GPT2 model during finetuning on DS machines.

ensures that the model inherently adapts to the hardware's biased patterns during training, resulting in accurate inference despite non-idealities.

Moreover, when deploying a pre-trained model on different devices, a promising solution is to finetune the model on the target hardware for a few batches. This allows the model to quickly adapt to the specific biased pattern of the new hardware. We evaluate this method by finetuning a pre-trained GPT2 model on DS machines with different offset level. The offset of DS machines is modeling as standard gaussian noise with standard variation from 0 to 0.3. As shown in Fig. 7, the dash line represents the inference accuracy without finetuning, while the solid line represents the inference accuracy with finetuning. The results illustrates the recovery of model accuracy during the finetuning process, underscoring the practicality and effectiveness of this approach.

C-W2

f) Precision Constraints: The precision of computation in DS machines is inherently continuous due to their analog nature. However, the precision of Analog-to-Digital Converters (ADCs) and Digital-to-Analog Converters (DACs) used in the system do affect the overall accuracy. Fortunately, ADCs and DACs are well-established technologies with numerous mature solutions that allow for various design trade-offs—for instance, achieving 16-bit precision with lower power consumption or 32-bit precision with higher power consumption. Such trade-offs align with and are theoretically compatible with existing quantization techniques, providing flexibility to balance precision and power efficiency based on the application requirements.

D-Q1-1

g) Novelty and differential from prior work: The prior work NPGL(Wu et al., 2024) is not a mapping of existing GNNs including those who have attention mechanism. It ignores the architecture of neural networks and directly fits the data with the Hamiltonian, which is a shallow fully connected model without any hierarchy or structure as shown in Eq. (1). The parameter size of this model is determined by the size of input data, making it fundamentally unable to support extremely complex tasks like NLP which cannot fit in the shallow Hamiltonian. Therefore, rather than directly fitting data into a shallow DS model as NPGL does, we propose a novel mapping method that enables the deployment of existing deep neural networks (NNs) to DS machines. This allows us to leverage well-established, mature model architectures, rather than working solely with the DS model itself. To the best of our knowledge, this is the first work to map deep NNs—designed for traditional computing architectures—onto DS machines, with LLMs chosen as a specific application.

D-Q1-2

D-Q4-4

h) Advantages over other emerging computing diagram: DS machines show high potential to satisfy the increasing LLM computing demanding, while there are also other emerging computing diagrams like quantum computing, optical computing and Computing-In-Memory works. We would like to briefly compare DS machines with other approach and highlight our key advantages.

Quantum computing: Quantum computing is a promising avenue but is still constrained by scalability issues and the need for complex error correction. As quantum systems scale, errors and noise increase, demanding advanced error-correction techniques that are not yet fully mature. Existing largest quantum computer from IBM has about 1100 qubits, which is too small to support LLM computing tasks. Meanwhile, due to the technology requirements, nowadays building and running quantum computers are still very expensive. In the contrast, DS machines are built on CMOS-compatible technologies, which are highly mature and the cost of its fabrication is at the same level of digital processors.

Optical Computing: The optical computing solutions also rely on special technology and is hard to be integrated with digital systems. Building complex and accurate optical circuits can be a big challenge and very expensive, making the stability and feasibility of optical computer a larger challenge than DS machines.

Computing-In-Memory (CIM): CIM technology is relatively feasible and CMOS compatible. Existing CIM works only supports inference, achieving up to 100s times acceleration and 2000 times energy reduction (Wolters et al., 2024), which is much lower than our solution especially on energy reduction. The insight behind this is that CIM still follows a traditional instruction-based paradigm, completing computations step by step. In contrast, DS machines are driven by physical processes, specifically natural annealing, which doesn't require step-by-step control. This allows DS machines to naturally perform complex tasks automatically without extra energy for controlling, achieving a much higher energy efficiency.

Table 6: Trade-off on Taylor Expansion order.

Taylor Expansion order	**Validation loss**	**Loss Drop**	**Hardware cost (units)**
Inf (baseline exponential)	3.25	0	-
1	3.35	0.10	1
3	3.28	0.03	6
5	3.27	0.02	15
7	3.27	0.02	28

D-Q2-2

i) Trade-off on Taylor Expansion: Before training the models, we conducted a pre-experiment by fine-tuning a pre-trained GPT-2 model on a small dataset (Shakespeare, 300k) with different orders of Taylor expansion. The results show that the 3rd-order expansion achieves sufficiently low validation loss, while the improvement from higher orders is marginal. In terms of hardware requirements, the resource cost scales approximately linearly with the order of each term. For instance, a polynomial like "$x^3 + x^2 + x$" requires around 6 resource units, while adding a term like "x^5" would demand an additional 5 units. Based on this trade-off, we selected the 3rd-order expansion as the most balanced design choice. For those prioritizing accuracy over hardware efficiency, higher-order expansions can be adopted and are compatible with our framework.

D-Q4-3

j) More inference metrics: For the inference comparison, we provide some more commonly used metrics to better show the performance of DS-LLM.

Table 7: Additional Inference Metrics on average of all datasets.

	Time to First Token (s)	Token Generation Rate (token/s)	Energy Efficiency (token/KWh)
GPT2	1.31E-4	7.62E+3	6.84E+7
GPT2-DS	1.20E-6	8.33E+5	1.11E+12
gpt2-m	3.92E-4	2.55E+3	2.38E+7
gpt2-m-DS	2.50E-6	4.00E+5	5.33E+11
OPT1.3B	6.85E-4	1.46E+3	1.37E+7
OPT1.3B-DS	6.00E-6	1.67E+5	2.22E+11
OPT2.7B	2.12E-3	4.72E+2	4.34E+6
OPT2.7B-DS	1.30E-5	7.69E+4	1.03E+11
Llama2-7B	7.08E-3	1.41E+2	1.35E+6
Llama2-7B-DS	3.30E-5	3.03E+4	4.04E+10

A.2 Implementation of other Operations

Due to the page limit, we only introduce the most important computing demanding operations in Section 3. Here we provide the analysis and implementation of other operations.

a) Activation Functions: The activation function used in GPT-2 at its initial publication was the ReLU function. However, many large language models (LLMs) have since transitioned to the Gaussian Error Linear Unit (GELU) (Hendrycks & Gimpel, 2017) for enhanced performance.

The ReLU function is straightforward to implement in hardware by simply turning off the output for spins with negative values. In contrast, the GELU function is more complex and is defined as follows:

$$\text{GELU}(x) = 0.5x\left(1 + \tanh\left(\frac{\sqrt{2/\pi}(x + 0.044715x^3)}{2}\right)\right) \tag{17}$$

The GELU function can be decomposed into matrix multiplication operations and the hyperbolic tangent function. To approximate the tanh function, we employ the same Taylor series expansion method used for the exponential function:

$$\tanh_{\text{Taylor}}(x) = x - \frac{x^3}{3} + \frac{x^5}{5} \tag{18}$$

Consequently, the implementation of the GELU function can be transformed into a series of matrix multiplication operations, similar to the approach introduced for the exponential function in Fig. 3 in Section 3.3.

b) Layer Normalization: Layer normalization is a crucial function in Large Language Models (LLMs). Unlike traditional Convolutional Neural Networks (CNNs), where normalization is performed in the batch direction and can be fixed during inference, layer normalization involves complex computations that cannot be avoided:

$$\mu = \frac{1}{H}\sum_{i=1}^{H} x_i \tag{19}$$

$$\sigma^2 = \frac{1}{H}\sum_{i=1}^{H}(x_i - \mu)^2 \tag{20}$$

$$\hat{x} = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} \tag{21}$$

$$y = \gamma\hat{x} + \beta \tag{22}$$

In this formulation, we can observe that the calculations of μ and y can be directly mapped to a series of matrix multiplication operations, as previously discussed. Additionally, reduction circuits, such as differential amplifiers, must be incorporated to handle the difference between x_i and μ. Furthermore, computing $\hat{x}$ requires an additional circuit to manage the division operation. Thus, the entire layer normalization operation can be effectively mapped to DS machines.