DS-LLM: Leveraging Dynamical Systems to Enhance Both Training and Inference of Large Language Models

Ruibing Song[1], Chuan Liu[1], Chunshu Wu[1], Ang Li[2], Dongfang Liu[3], Yingnian Wu[4], Tony (Tong) Geng[1]

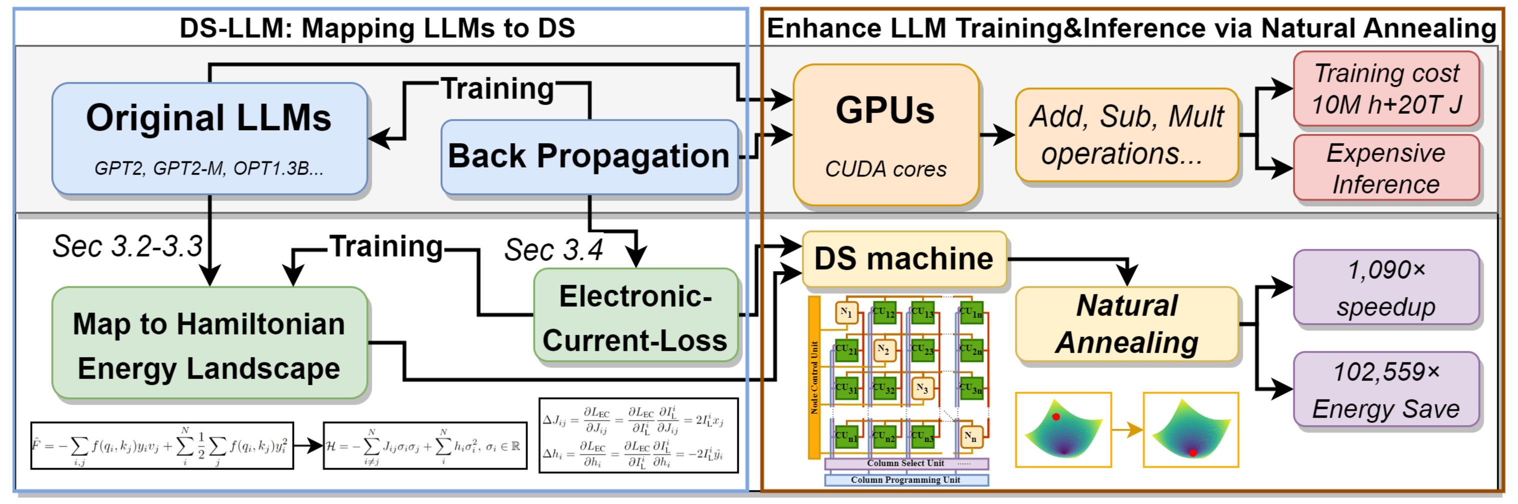
Department of Electrical and Computer Engineering, University of Rochester[1]; Physical & Computational Sciences Directorate, Pacific Northwestern National Laboratory[2]; Department of Computer Engineering, Rochester Institute of Technology[3]; Department of Statistics and Data Science, University of California, Los Angeles[4]

Motivation

- LLMs faces significant computational cost challenges:
  $64 million for GPT-4 in 2020 -> $191 million for Gemini Ultra in 2023
- Traditional techniques fail to address the fundamental bottleneck.
- Emerging techniques like Quantum computing, optical computing, and computing-in-memory are promising but facing significant technical barriers.
- Is it feasible to rely on mature CMOS-based technology to accelerate LLM training from 10 million hours to 10,000 hours while reducing energy consumption from 20 TJ to 200 MJ?

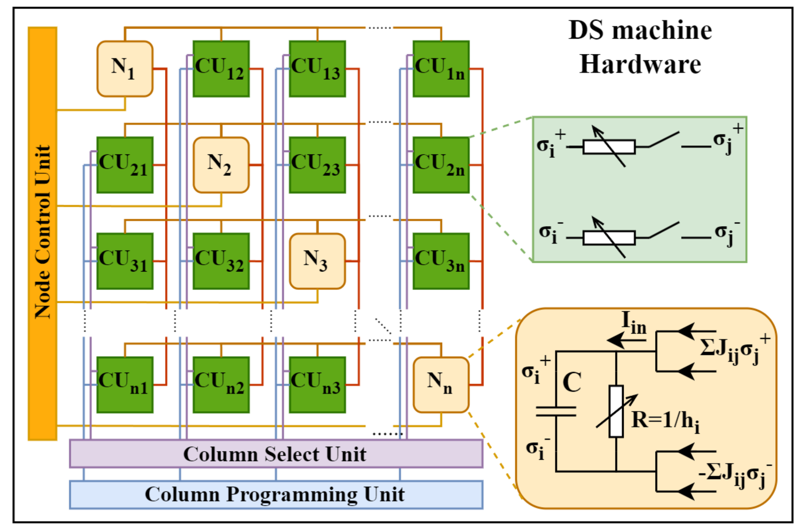
-> Yes! By training LLMs on DS-machines!

Introduction



- Mapping LLMs to dynamical system (DS)-based machines to leverage its amazing computing power and energy efficiency.
- For inference, DS-LLM maps LLM components to optimization problems solvable via Hamiltonian configurations that can be solved by DS-machines.
- For training, DS-LLM utilizes continuous electric current flow in DS-machines for hardware-native gradient descent during training.

Background



DS-machines operate based on the system energy aka Hamiltonian:

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

The electrodynamics is designed to inherently drive the Hamiltonian towards a local minimum.

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

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

Methodology

Inference Acceleration: Mapping existing LLM onto DS-machines

Step1: For a single linear layer

Approach 1: Hamiltonian level

- Shape the energy landscape -> map the minimum energy state to the desired output.
- Define a target function F to minimizes the difference between the output state of the DS-machine and the desired output:

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

- Align to the Hamiltonian of DS-machines:

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

- Map the linear layer to DS-machines. √

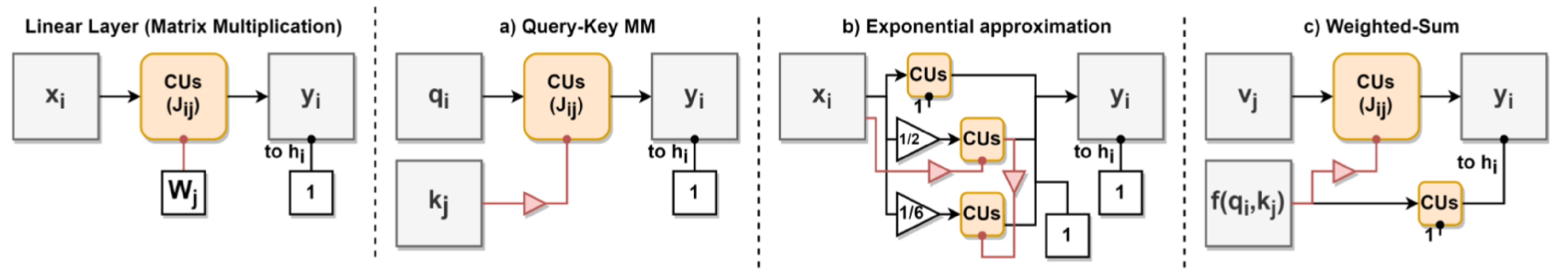
Approach 2: Electrodynamics level

- Guide the electrodynamics behavior by the coupling parameters J and self-reaction parameters h.
- Lyapunov stability analysis -> dσ/dt = 0 when DS-machines stop evolving.
- Dividing the spins σ into input x and output y:

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

- Directly program J and h to map the linear layer to DS-machines. √
- Equivalent to Hamiltonian level approach.

Step2: Extend the mapping approach to key operations in LLMs



Exponential approximation:

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

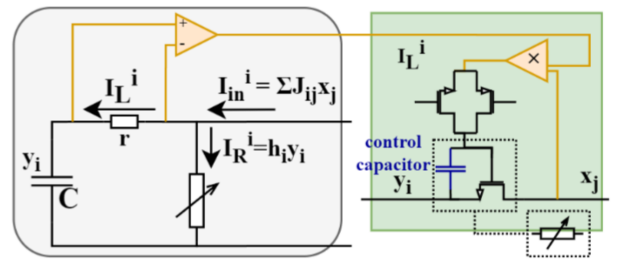
Weighted-Sum in Attention:

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

General solution: $\hat{F}^{(p)} = (x^{(p+1)})^2 - 2x^{(p+1)}f^{(p)}(x^{(p)})$ $\quad x^{(P+1)} = f^{(P)} \circ f^{(P-1)} \circ \cdots \circ f^{(1)}(x^{1})$

Training Acceleration: Online Training with Electric-Current-Loss

- **Macro-scale observation**: A well-trained DS-machine reaches its energy minimum when its output spins align with the ground truth from the training data.
- **Micro-scale**: The total incoming electric current $I_{in}^i = \Sigma J_{ij}x_j$ must balance the internal current $I_R^i = h_i y_i$ to ensure the voltage remains stable ($dy_i/dt = 0$).

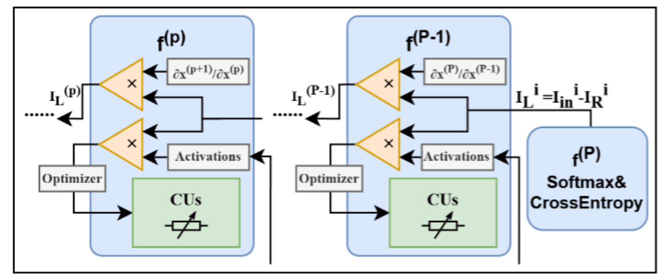


$$\frac{\partial L}{\partial x_i} = y_i - \hat{y}_i = \frac{I_{in}^i - I_R^i}{h_i} = I_L^i$$

- Map the ground truth output onto the output spins and fix their values.
- Define an electric-current-loss function equal to the current through the sampling resistor r, $I_L = I_{in} - I_R$.
- Update J_{ij} using gradient descent based on ECL.

$$\Delta J_{ij} = \frac{\partial L_{\mathrm{EC}}}{\partial J_{ij}} = \frac{\partial L_{\mathrm{EC}}}{\partial I_L^i}\frac{\partial I_L^i}{\partial J_{ij}} = 2I_L^i x_j$$

- The multiplication of the loss current I_L by the spin voltage x_j is implemented at circuit level.
- Feed back electric-current ΔJ_{ij} to coupling units.
- Parameters update: $J_{ij} \rightarrow J_{ij} - \Delta J_{ij}\Delta t$ by integrating the result current on over a time interval Δt.

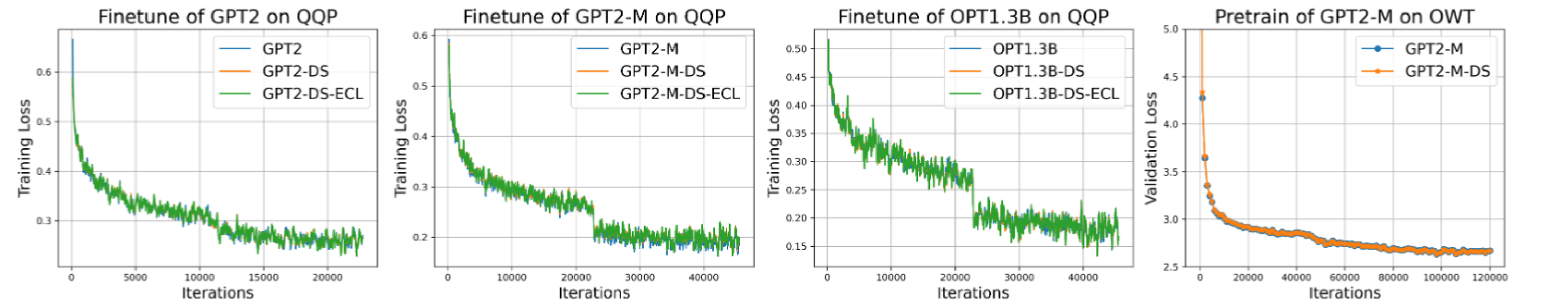


Combine ECL with BP

➔Enable online training
➔Eliminates output readout
➔Efficient training on DS-machines

Evaluations

Training trajectories visualization

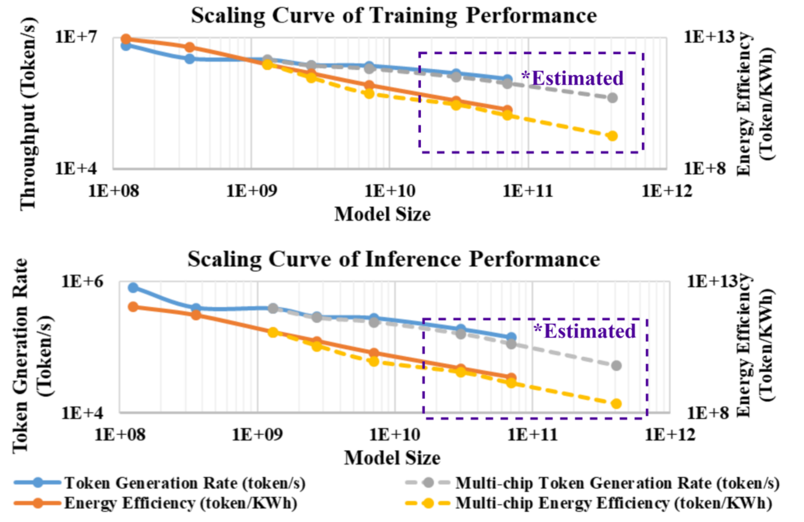


DS-LLM exhibit convergence curves that closely match original models on GPU, effectively replicating the performance of traditional LLMs on DS-machines.

Comparable Accuracy

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

Performance scaling with size



Hardware Performance Compared to GPU

Metric	Training		Inference		
	Throughput (token/s)	Energy Efficiency (token/KWh)	Time to First Token (s)	Token Generation Rate (token/s)	Energy Efficiency (token/KWh)
GPT2	1.37E+04	6.19E+07	6.46E-05	1.55E+04	1.39E+08
GPT2-DS	6.70E+06	8.93E+12	1.20E-06	8.33E+05	1.11E+12
gpt2-m	3.24E+03	1.46E+07	1.80E-04	5.55E+03	5.18E+07
gpt2-m-DS	3.27E+06	4.36E+12	2.48E-06	4.03E+05	5.38E+11
OPT1.3B	1.20E+03	5.41E+06	2.76E-04	3.62E+03	3.41E+07
OPT1.3B-DS	3.08E+06	9.56E+11	2.55E-06	3.92E+05	1.22E+11
OPT2.7B	3.97E+02	1.78E+06	8.12E-04	1.23E+03	1.13E+07
OPT2.7B-DS	2.33E+06	4.37E+11	3.41E-06	2.93E+05	5.50E+10
Llama2-7B	1.34E+02	6.05E+05	2.51E-03	3.98E+02	3.82E+06
Llama2-7B-DS	2.24E+06	1.56E+11	3.57E-06	2.80E+05	1.95E+10

Speedup:
Train: 5.3×10^3
Inference: 2.4×10^2

Energy Save
Train: 2.3×10^5
Inference: 6.4×10^3

Conclusion

- DS-LLM: the first algorithmic framework to bridge LLMs to DS-machines.
- The mathematical equivalence between DS-LLM and original LLMs is proven and validated through experiments on models from GPT-2 to Llama2-7B.
- Results demonstrate **consistent accuracy** while achieving a **5.3×10^3 speedup** for training and **2.4×10^2** for inference, along with a **reduction in energy consumption** of **2.3×10^5** during training and **6.4×10^3** during inference.
- DS-LLM presents a promising new solution for the community with significant opportunities for further exploration in future studies. We look forward to seeing more exciting developments emerge in this area!