Gradient-Free Training of Spiking Neural Networks via Low-Rank Evolution Strategies

Dhruv Patankar[1] and Sachit Ramesha Gowda[2]

[1]Shunya Research , dhruv@shunyaresearch.systems
[2]Shunya Research , sachit@shunyaresearch.systems

May 13, 2026

Abstract

Spiking Neural Networks (SNNs) offer compelling energy efficiency on neuromorphic hardware, yet their training remains challenging because the discrete spike threshold is non-differentiable. Surrogate-gradient methods sidestep this by approximating the derivative, but they impose backpropagation infrastructure that is incompatible with on-chip learning. Evolution Strategies (ES) are a natural gradient-free alternative, yet their computational cost scales with the number of parameters, making them impractical for large weight matrices.

We present a method for training SNNs using EGGROLL, a low-rank factorisation of ES perturbations that reduces per-generation memory from $\mathcal{O}(mn)$ to $\mathcal{O}(r(m+n))$. Combining EGGROLL with a Leaky Integrate-and-Fire SNN on N-MNIST, we demonstrate that gradient-free training achieves 79.21% test accuracy while reducing per-generation wall-clock time by 2.23× relative to full-rank ES. Our results demonstrate EGGROLL is viable for SNN training, with a clear accuracy-speed tradeoff, compatible with training on neuromorphic hardware without surrogate gradients.

1 Introduction

Energy Efficiency is a huge focus of modern AI research. A lot of research is focused on reducing the energy usage of AI architecture. Intel's Loihi [Davies et al., 2018] showed us that SNNs could be a potential solution on real hardware. The dominant paradaigm in current AI is the transformer architecture [Vaswani et al., 2017]. Transformers are trained on GPUs and use hundreds of watts in their training process. While the artificial neural networks with transformers show remarkable performance, they use a significant amount of energy. The human brain, in comparison, is extremely efficient, only using 20 watts. SNNs

attempt to capture this efficiency: neurons integrate inputs over time and fire only when a membrane potential threshold is crossed, enabling sparse, event-driven computation on dedicated neuromorphic chips such as Intel Loihi [Davies et al., 2018] and IBM TrueNorth [Merolla et al., 2014].

The training problem. Despite their efficiency during inference, SNNs are still quite difficult to train. The spiking generation function(Heavside step) has a zero gradient almost everywhere. This makes it impossible to do standard backpropagation, since it requires calculating the gradient. The common solution to this is using surrogate gradients [Neftci et al., 2019]. Surrogate gradients work by replacing the true gradient of a function with an approximation, which can be differentiated. This allows backpropagation to proceed through layers where it would otherwise be impossible. However, surrogate gradients still require autograd infrastructure, which is not compatible with on chip learning on neuromorphic hardware.

Evolution Strategies. ES [Salimans et al., 2017] is a group of loosely bio inspired training methods that are alternatives to backpropagation that do not require gradients to be calculated. In ES, weight vectors are randomly perturbed and then a fitness evaluation is done on these perturbations. However these methods are very computationally expensive, incurring $O(Pmn)$ memory for each generation, for a weight matrix of dimensions mn and population P, making it very challenging to scale on larger networks.

Our contribution. We integrate EGGROLL [Sarkar et al., 2025] — which replaces each full-rank perturbation with a low-rank product $\mathbf{AB}^\top$, $\mathbf{A} \in \mathbb{R}^{m\times r}$, $\mathbf{B} \in \mathbb{R}^{n\times r}$ — into the SNN training loop. This reduces per-generation cost to $\mathcal{O}(r(m+n))$ while preserving the gradient-free property.

Concretely, our contributions are:

1. we characterize EGGROLL's behavior on non-differentiable spike functions EGGROLL with SNNs, enabling gradient-free training without surrogate approximations (Section 3).

2. A rank ablation showing that accuracy remains relatively stable as r decreases, with a clear efficiency–accuracy Pareto frontier (Section 4).

3. Comparison against vanilla ES and surrogate-gradient BPTT on N-MNIST. N-MNIST is a native neuromorphic dataset, with no rate-coding approximation used, making this a more principled evaluation than the one done on a static MNIST dataset. (Section 4).

2 Background

2.1 N-MNIST Dataset

The N-MNIST dataset [Orchard et al., 2015] or Neuromorphic MNIST is a spiking version of the popular MNIST dataset that contains static images of hand drawn numbers. It consists of the same 60,000 training and 10,000 test samples as the original MNIST dataset. It was created by mounting the ATIS sensor on a motorized pan tilt unit and moving it while it recorded MNIST images on an LCD display. The sensor outputs asynchronous events with 2 polarities: ON events (higher brightness) and OFF events (lower brightness). This is a better fit for SNNs than the regular MNIST dataset because the data is already a spike train, so no rate coding approximation is required. Due to this, the SNN processes the actual sensor output instead of the surrogate encoding.

2.2 Spiking Neural Networks and the LIF Model

We model each neuron as a Leaky Integrate-and-Fire unit. At each discrete timestep t, the membrane potential V_t evolves as

$$V_t = \alpha V_{t-1} + \mathbf{w}^\top \mathbf{s}_{t-1}, \tag{1}$$

where $\alpha \in (0, 1)$ is the membrane decay constant (leak factor), $\mathbf{w}$ the synaptic weight vector, and $\mathbf{s}_{t-1}$ the binary spike vector from the previous layer. A spike is emitted when $V_t \geq V_{\text{th}}$, after which V_t resets to $V_{\text{reset}} = 0$. The network output is the spike count (firing rate) over T timesteps.

2.3 Surrogate Gradient Training

Because the spike function $\Theta(V - V_{\text{th}})$ is a Heaviside step, its true derivative is zero almost everywhere. Surrogate gradient methods [Neftci et al., 2019] substitute a smooth proxy $\hat{\Theta}'$ — commonly the fast sigmoid — only during the backward pass, leaving the forward pass unchanged. This recovers gradient flow at the cost of introducing an approximation bias and requiring full backpropagation-through-time (BPTT). Implementing this method requires autograd, which as mentioned earlier, is not compatible with on-chip learning, and makes it unsuitable for neuromorphic hardware.

2.4 OpenAI Evolution Strategies

Salimans et al. [2017] propose estimating the gradient of expected fitness $J(\boldsymbol{\theta})$ as

$$\nabla_{\boldsymbol{\theta}} J \approx \frac{1}{2P\sigma} \sum_{i=1}^{P} \big[F(\boldsymbol{\theta} + \sigma\boldsymbol{\varepsilon}_i) - F(\boldsymbol{\theta} - \sigma\boldsymbol{\varepsilon}_i)\big]\boldsymbol{\varepsilon}_i, \tag{2}$$

where $\boldsymbol{\varepsilon}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, P is the population size, and σ the perturbation scale. Antithetic (mirrored) sampling reduces variance with no extra sampling budget.

The estimate is passed to Adam [Kingma and Ba, 2014] as if it were a true gradient. The biggest issue with ES is that its memory cost is $\mathcal{O}(Pmn)$, making it slow for large models.

2.5 EGGROLL: Low-Rank Evolution Strategies

Sarkar et al. [2025] observe that sampling a full perturbation matrix $\mathbf{E} \in \mathbb{R}^{m\times n}$ is wasteful: for population P, the perturbation tensor requires $\mathcal{O}(Pmn)$ memory. EGGROLL instead draws

$$\mathbf{E}_i = \frac{1}{\sqrt{r}}\mathbf{A}_i\mathbf{B}_i^\top, \quad \mathbf{A}_i \in \mathbb{R}^{m\times r},\ \mathbf{B}_i \in \mathbb{R}^{n\times r}, \tag{3}$$

normalising by $\sqrt{r}$ so that each entry has unit variance regardless of rank, without which the variance scales by r, rendering the σ hyperparameter obsolete. Additionally, EGGROLL reconstructs noise on demand using a counter based deterministic random number generator(RNG), so that the perturbations do not have to be stored in memory. This trick helps EGGROLL work on the billion parameter scale. The gradient estimate is then reconstructed via the $(\mathrm{diag}(\mathbf{f})\mathbf{A})^\top\mathbf{B}$ formulation (§4.2 of Sarkar et al. [2025]), which never materialises individual perturbation matrices. Memory per generation drops to $\mathcal{O}(r(m+n))$.

3 Method

3.1 Network Architecture

We use a two-layer LIF network:

$$2312 \to 64 \xrightarrow{\text{LIF}} 10 \xrightarrow{\text{LIF}} \text{output}. \tag{4}$$

The input is of size 2312 since N-MNIST has a size of $34X34$ pixels, and 2 polarities for each pixel. Each LIF layer shares a common membrane decay β . Initial weights are sampled from $\mathcal{N}(0, 0.3^2)$. The output is the mean spike rate over T timesteps, converted to class probabilities via softmax for fitness evaluation. Biases are perturbed with independent 1-D Gaussian factors to maintain equivalence with full-rank ES when $r = \min(m, n)$. Unlike static-image SNNs, the forward pass feeds a different event frame to each timestep not the same image repeated T times. Event counts are clamped to $[0, 1]$ to prevent a single pixel dominating membrane potential.

3.2 EGGROLL Integration

Algorithm 1 details the training loop. Instead of backpropogating through the SNN the usual way, the algorithm starts with network weights and creates many perturbed copies of the network which are subsequently run on a minibatch. The performance of each copy is measured and that score is then converted into a gradient like signal which updates the weights with Adam. The key departure

Algorithm 1 EGGROLL training for an SNN

Require: rank r, population P, scale σ, learning rate η, generations G, timesteps T

1: Initialise weights $\boldsymbol{\theta} = \{\mathbf{W}_1, \mathbf{b}_1, \mathbf{W}_2, \mathbf{b}_2\}$ from $\mathcal{N}(0, 0.3^2)$
2: **for** $g = 1, \ldots, G$ **do**
3: Sample seed s; draw data mini-batch $(\mathbf{X}, \mathbf{y})$
4: Reconstruct $\mathbf{A}_i, \mathbf{B}_i, \mathbf{c}_i$ from s for $i = 1, \ldots, P$
5: **Batched forward:** evaluate all P perturbed nets on $\mathbf{X}$
6: Collect antithetic rewards r_i^+, r_i^- for each population member
7: Apply centered rank normalisation to $\{r_i^+, r_i^-\}$
8: Compute gradient $\hat{\nabla}$ via $(\mathrm{diag}(\mathbf{f})\mathbf{A})^\top \mathbf{B}$
9: Update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \cdot \mathrm{Adam}(\hat{\nabla})$
10: **end for**

from vanilla ES is the batched forward pass in Line 5: all P perturbed networks are evaluated simultaneously by expanding the data tensor along a population dimension, avoiding a sequential Python loop over population members. The factors $\mathbf{A}$, $\mathbf{B}$ are regenerated from a stored seed rather than cached, keeping GPU memory independent of P. Centered rank normalization is done to make the gradient scale unaffected by absolute fitness values across batches. Antithetic sampling is also used to reduce the variance.

3.3 Fitness Function

We use negative cross-entropy (log-likelihood) as the fitness signal, rather than raw classification accuracy:

$$F(\boldsymbol{\theta}) = -\frac{1}{B}\sum_{b=1}^{B} \log \frac{\exp(\hat{y}_{b,y_b})}{\sum_c \exp(\hat{y}_{b,c})}, \tag{5}$$

where $\hat{y}_{b,c}$ is the spike rate for class c on example b. Log-likelihood provides a smooth landscape for ES to navigate; raw accuracy (a step function over fitness values) gives no gradient signal between threshold crossings.

4 Experiments

4.1 Setup

Dataset. N-MNIST [Orchard et al., 2015]: 54,000 training / 6,000 validation / 10,000 test images, generator seed 0. Sensor size $34X34$, 2 polarities. No rate coding is used, the event frames are the spiking input.

Baselines.

- **Vanilla ES**: full-rank Gaussian perturbations, sequential evaluation, identical hyperparameters.
- **Surrogate-gradient BPTT**: fast-sigmoid surrogate, Adam optimiser, same architecture.

Compute. All experiments run on a single NVIDIA RTX 3070 Ti. Wall-clock times are averaged over three seeds.

4.2 Main Results

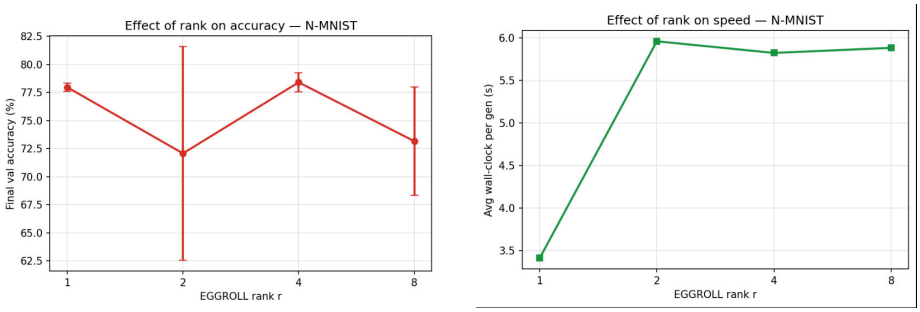
Table 1: Test accuracy and per-generation wall-clock time. Mean ± std over three seeds.

Method	Test Acc. (%)	Time / gen (s)	Speedup
Surrogate BPTT	94.03±0.28	31.69	0.41×
Vanilla ES	76.17±6.07	13.00	1.0×
EGGROLL $r = 1$	78.75±0.58	3.41	3.81×
EGGROLL $r = 2$	72.27±12.00	5.96	2.18×
EGGROLL $r = 4$	79.21±0.64	5.82	2.23×
EGGROLL $r = 8$	72.80±6.08	5.88	2.21×

The key takeaway from Table 1 is the significant speedup that EGGROLL provides to ES, for almost no tradeoff in accuracy. Even with a rank of 1, the difference in accuracy is very minute relative to the speedup provided by EGGROLL. Though Surrogate BPTT has a higher accuracy than ES, it is significantly slower.

4.3 Rank Ablation

Figure 1 plots validation accuracy and per-generation wall-clock time as functions of rank r.



(a) Accuracy vs. rank.

(b) Wall-clock vs. rank.

Figure 1: Rank ablation on N-MNIST.

The accuracy remains relatively stable across ranks, with the best performance occurring at $r = 1$ and $r = 4$, with larger variance at $r = 2$ and $r = 8$. Wall clock time is lowest at $r = 1$. It increases for $r = 2$, and remains relatively stable across the other 2 ranks.

4.4 Convergence Curves

Figure 2 plots the convergence of different ES implementations(Naive and with EGGROLL) across generations. Figure 3 plots the average wall clock time for vanilla ES and EGGROLL.

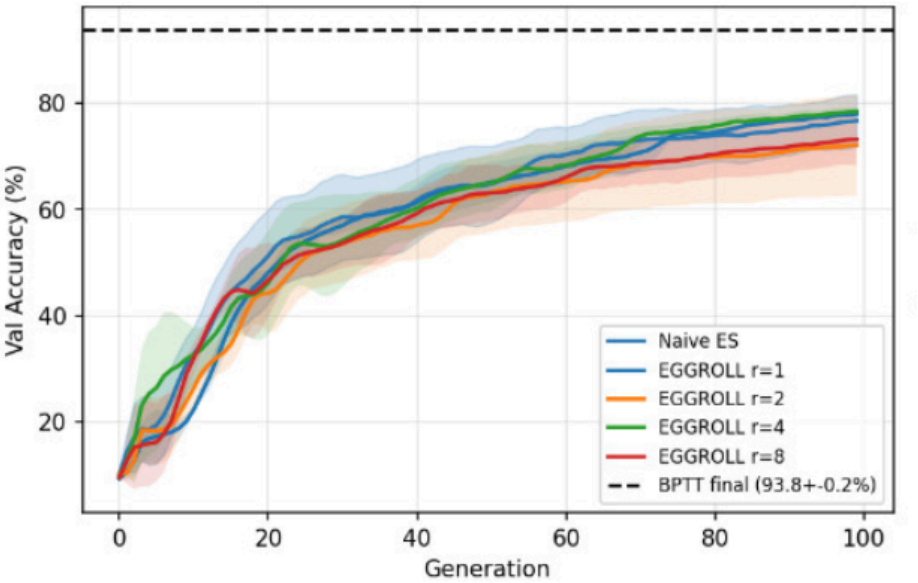


Figure 2: Validation accuracy over generations for all methods (mean ± std, three seeds). Dashed line: surrogate BPTT final accuracy.

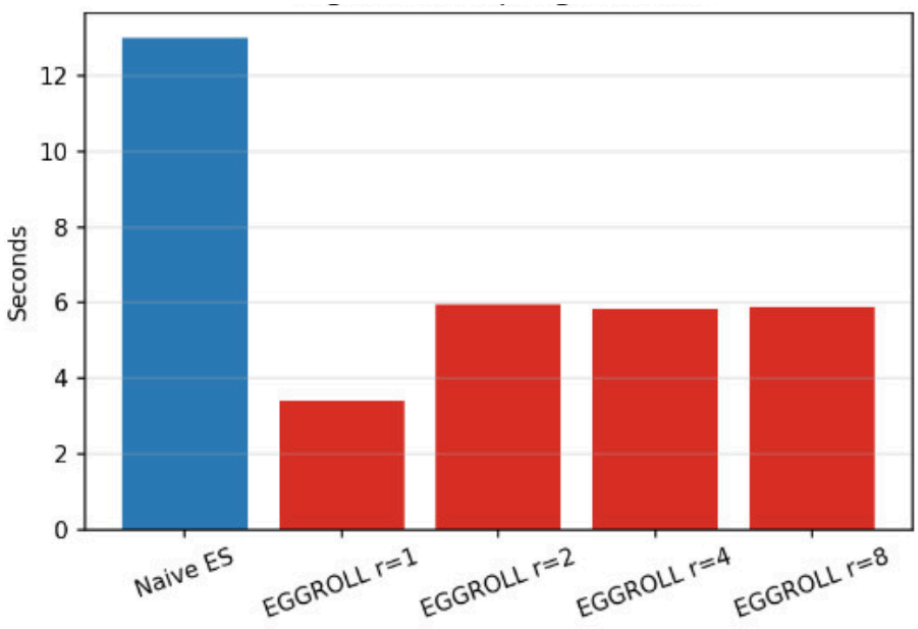


Figure 3: Wall clock time(in seconds) for EGGROLL implementations and vanilla ES

EGGROLL with $r = 4$ converges slightly faster than the other EGGROLL variants, with $r = 1$ slightly behind, but overall the difference in convergence speed is modest. The variance bands (mean $\pm$ std over three seeds) are generally narrow for EGGROLL with small ranks, indicating more stable convergence than several higher-rank or full-rank baselines. EGGROLL achieves substantially lower wall-clock training time due to its reduced per-generation cost. In particular EGGROLL reaches comparable validation accuracy in fewer seconds than vanilla ES despite similar generation-level convergence behavior.

5 Discussion

Accuracy gap. EGGROLL achieves 79.21% accuracy vs. 94.03% for surrogate BPTT. We attribute the gap to our limited compute budget as a result of which we could only run 100 generations for the ES training(naive and EGGROLL). Additionally with ES, the gradient estimate can be noisy because of stochastic perturbations and mini-batch variation. Due to our limited compute budget, we used a lightweight two-layer fully connected SNN, which keeps the parameter count modest, but limits our representational capacity relative to deeper architectures. Per-seed results (Appendix A) show occasional training collapse at $r = 2$, $r = 8$, suggesting sensitivity to initialization that warrants further study.

Rank and expressiveness. The rank ablation (Figure 1) shows that even $r = 1$ retains substantial accuracy, suggesting the useful gradient directions lie in a low-dimensional subspace. This is consistent with the flat minima hypothesis [Hochreiter and Schmidhuber, 1997] and with findings on intrinsic dimensionality in neural networks [Li et al., 2018]. The ablations are empirical evidence for this in SNNs.

Limitations. Our experiments are limited to N-MNIST with a two-layer, fully connected network with no convolutions, and 10 fixed time bins which may discard some temporal resolution. Additionally we have not tested this on real neuromorphic hardware, and have not measured the energy consumption. Scaling to CIFAR10-DVS [Li et al., 2017] and N-Caltech101 [Orchard et al., 2015] is the next step, since these are more complex than N-MNIST. Three seeds is insufficient for definitive rank claims; we report per-seed results in Appendix A for transparency.

6 Related Work

SNN training methods. Spike-Timing-Dependent Plasticity (STDP) is a local Hebbian rule that requires no global error signal [Bi and Poo, 1998] but struggles to match backprop accuracy. Surrogate gradients [Neftci et al., 2019, Bellec et al., 2020] are currently the dominant approach for deep SNNs, achieving state-of-the-art accuracy but requiring autograd. Conversion methods [Cao et al., 2015] train an ANN and convert weights, but incur long simulation times.

Evolution Strategies for neural networks. Salimans et al. [2017] showed ES competitive with RL on Atari. Such et al. [2017] demonstrated ES on deep networks via compact perturbation seeds. PEPG [Sehnke et al., 2010] and CMA-ES [Hansen, 2016] are closely related natural-gradient variants. EGGROLL [Sarkar et al., 2025] is, to our knowledge, the first explicit low-rank factorisation of ES perturbations shown to scale to large models; our work is the first to apply it to SNNs.

Neuromorphic hardware constraints. Loihi [Davies et al., 2018] and BrainScaleS [Schemmel et al., 2010] support on-chip learning but only with local rules or very limited precision. Gradient-free methods are a natural fit for these constraints, motivating this line of work.

EGGROLL Sarkar et al. [2025] showed that EGGROLL can work on language models and RNNs. As far as we know, our work is first application of EGGROLL on SNNs, that poses the challenge of non differentiable spikes.

7 Conclusion

Training on SNNs is challenging due to its discrete spike threshold. We have demonstrated that EGGROLL — low-rank Evolution Strategies — can train SNNs on N-MNIST without surrogate gradients or backpropagation, achieving 79.21% test accuracy at 2.23× the speed of vanilla ES. The rank $r = 4$ offers the best accuracy-per-second tradeoff in our setup.

Future work. The most immediate extension is scaling to more complex neuromorphic benchmarks, like N-Caltech101 [Orchard et al., 2015] and DVS-CIFAR10 [Orchard et al., 2015] with a convolutional SNN. Beyond accuracy, a key open question is whether EGGROLL-trained SNNs produce sparser spike trains than surrogate-gradient methods, which would directly translate to energy savings on neuromorphic hardware. Finally, combining EGGROLL with local plasticity rules for the early layers — reserving global ES updates for the readout layer — may offer the best of both worlds.

References

Guillaume Bellec, Franz Scherr, Anand Subramoney, et al. A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature Communications*, 11(1):3625, 2020.

Guo-qiang Bi and Mu-ming Poo. Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. *Journal of Neuroscience*, 18(24):10464–10472, 1998.

Yongqiang Cao, Yang Chen, and Deepak Khosla. Spiking deep convolutional neural networks for energy-efficient object recognition. volume 113, pages 54–66. Springer, 2015.

Mike Davies, Narayan Srinivasa, Tsung-Han Lin, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018.

Nikolaus Hansen. The cma evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural Computation*, 9 (1):1–42, 1997.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes. In *International Conference on Learning Representations*, 2018.

Hongmin Li, Hanchao Liu, Xiangyang Ji, Guoqi Li, and Luping Shi. Cifar10-dvs: An event-stream dataset for object classification. *Frontiers in Neuroscience*, Volume 11 - 2017, 2017. ISSN 1662-453X. doi: 10.3389/fnins.2017.00309. URL https://www.frontiersin.org/journals/neuroscience/articles/10.3389/fnins.2017.00309.

Paul A Merolla, John V Arthur, Rodrigo Alvarez-Icaza, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014.

Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks. *IEEE Signal Processing Magazine*, 36(6): 51–63, 2019.

Garrick Orchard, Ajinkya Jayawant, Gregory K. Cohen, and Nitish Thakor. Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in Neuroscience*, Volume 9 - 2015, 2015. ISSN 1662-453X. doi: 10.3389/fnins.2015.00437. URL https://www.frontiersin.org/journals/neuroscience/articles/10.3389/fnins.2015.00437.

Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

Soham Sarkar et al. Evolution strategies at the hyperscale. *arXiv preprint arXiv:2511.16652*, 2025.

Johannes Schemmel, Daniel Brüderle, Andreas Grübl, et al. A wafer-scale neuromorphic hardware system for large-scale neural modeling. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, pages 1947–1950. IEEE, 2010.

Frank Sehnke, Christian Osendorfer, Thomas Rückstieß, Alex Graves, Jan Peters, and Jürgen Schmidhuber. Parameter-exploring policy gradients. *Neural Networks*, 23(4):551–559, 2010. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2009.12.004. URL https://www.sciencedirect.com/science/article/pii/S0893608009003220. The 18th International Conference on Artificial Neural Networks, ICANN 2008.

Felipe Petroski Such, Vashisht Madhavan, Edoardo Conti, et al. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

A Variance Analysis Across Seeds

Table 2: Per-seed accuracy and average training time across three seeds.

Method	Test Acc. (%)	Time / gen (s)
Vanilla ES (seed 1)	80.66	12.41
Vanilla ES (seed 2)	78.59	13.15
Vanilla ES (seed 3)	69.26	13.43
EGGROLL $r = 1$ (seed 1)	79.27	3.04
EGGROLL $r = 1$ (seed 2)	78.13	3.59
EGGROLL $r = 1$ (seed 3)	78.86	3.60
EGGROLL $r = 2$ (seed 1)	79.15	6.06
EGGROLL $r = 2$ (seed 2)	79.25	5.94
EGGROLL $r = 2$ (seed 3)	58.42	5.88
EGGROLL $r = 4$ (seed 1)	79.94	5.90
EGGROLL $r = 4$ (seed 2)	78.98	5.82
EGGROLL $r = 4$ (seed 3)	78.72	5.76
EGGROLL $r = 8$ (seed 1)	79.78	5.75
EGGROLL $r = 8$ (seed 2)	69.94	5.64
EGGROLL $r = 8$ (seed 3)	68.68	6.27
Surrogate BPTT (seed 1)	94.35	32.09
Surrogate BPTT (seed 2)	93.88	31.67
Surrogate BPTT (seed 3)	93.86	31.32

B EGGROLL Gradient Derivation

Following Sarkar et al. [2025] §4.2, the gradient estimate for weight matrix $\mathbf{W}$ under low-rank perturbations is:

$$\begin{aligned}\hat{\nabla}_{\mathbf{W}} J &= \frac{1}{2P\sigma\sqrt{r}} \sum_{i=1}^{P} f_i \mathbf{A}_i \mathbf{B}_i^\top \\ &= \frac{1}{2P\sigma\sqrt{r}} \big(\mathrm{diag}(\mathbf{f})\, \mathbf{A}\big)^\top \mathbf{B}, \end{aligned} \tag{6}$$

where $\mathbf{A} \in \mathbb{R}^{P\times m\times r}$ and $\mathbf{B} \in \mathbb{R}^{P\times n\times r}$ are stacked factor tensors, and $f_i = r_i^+ - r_i^-$ is the antithetic rank difference. Equation (6) requires only two batched matrix multiplications, never materialising the $P \times m \times n$ perturbation tensor.