Allen Discovery Center
AT TUFTS UNIVERSITY

Diffusion Models are Evolutionary Algorithms

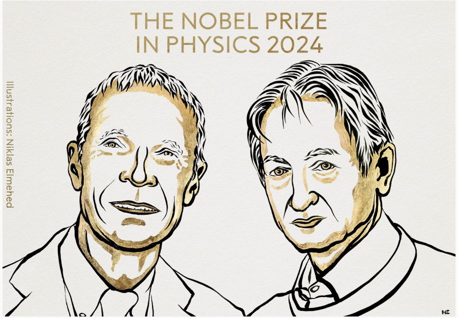
Yanbo Zhang[1], Benedikt Hartl[1,2], Hananel Hazan[1], Michael Levin[1,3]

[1]Allen Discovery Center at Tufts University
[2]Institute for Theoretical Physics, TU Wien, Austria
[3]Wyss Institute for Biologically Inspired Engineering at Harvard University

Intelligence and Learning

What is Intelligence?

- Ilya: Intelligence = compression
- Geoffrey Hinton: Intelligence = learning
  - Adaptive, reasoning, ...
  - self-organize



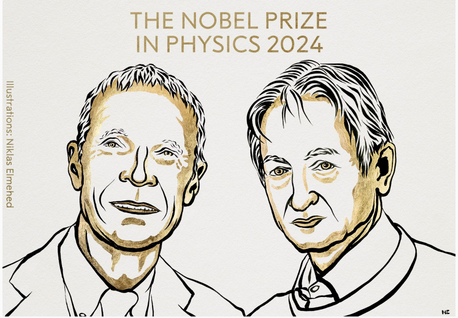
Intelligence of Evolution

- Adapting
- Immune systems, neurons, ...
- **Evolution itself can be seen as a kind of intelligence**

Intelligence and Learning

What is Intelligence?

- Ilya: Intelligence = compression
- Geoffrey Hinton: Intelligence = learning
  - Adaptive, reasoning, ...
  - self-organize



Evolution of Intelligence

- Machine Intelligence: diffusion model
- Iterative: denoise, optimize, adding noise

Diffusion Model

Quick review of Diffusion Models:

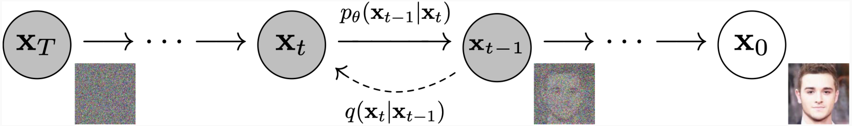


Figure 1: Example of Diffusion Models on image generation.

Diffusion and Training

Train a model to predict the added noise given $\boldsymbol{x}_t$ [1, 2].

$$\boldsymbol{x}_t = \sqrt{\alpha_t}\boldsymbol{x}_0 + \sqrt{1-\alpha_t}\boldsymbol{\epsilon},$$

and

$$\mathcal{L} = \mathbb{E}_{\boldsymbol{x}_0 \sim p_{\text{data}}, \boldsymbol{\epsilon} \sim \mathcal{N}(0,I)} \|\epsilon_\theta(\sqrt{\alpha_t}\boldsymbol{x}_0 + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}) - \boldsymbol{\epsilon}\|^2$$

Diffusion Model

Quick review of Diffusion Models:

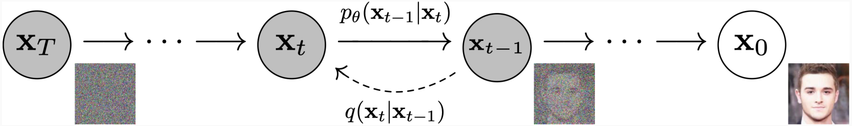


Figure 1: Example of Diffusion Models on image generation.

Denosing

Use the trained model to denoise from Gaussian distribution [2]:

$$\boldsymbol{x}_{t-1} = \sqrt{\alpha_{t-1}} \left(\frac{\boldsymbol{x}_t - \sqrt{1-\alpha_t}\epsilon_\theta(\boldsymbol{x}_t, t)}{\sqrt{\alpha_t}} \right) + \sqrt{1-\alpha_{t-1}-\sigma_t^2}\epsilon_\theta(\boldsymbol{x}_t, t) + \sigma_t \boldsymbol{w}$$

Similarity between evolution and diffusion

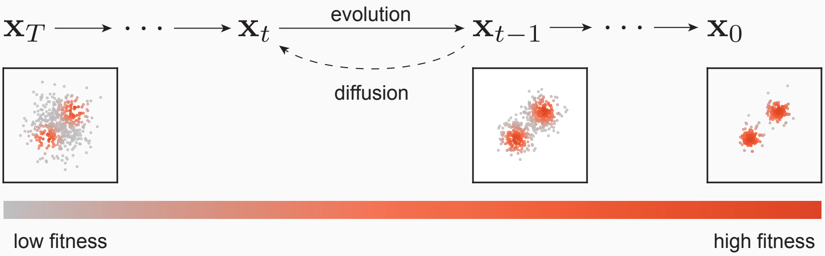
Diffusion and Evolution are both:

- directed: natural selection v.s. denoise
- randomness: mutation v.s. noise term
- optimization process

$$\boldsymbol{x}_{t-1} = \sqrt{\alpha_{t-1}}\hat{\boldsymbol{x}}_0 + \sqrt{1 - \alpha_{t-1} - \sigma_t^2}\hat{\boldsymbol{\epsilon}} + \sigma_t \boldsymbol{w}$$

Key idea

diffusion as reversed evolution, and denoising as evolution.



Connecting Diffusion to Evolution

Three problems need to be solved:

Mapping fitness to probability density

Higher fitness $f(x)$ corresponds to higher density $p(x)$, which requires a mapping function $g : \mathbb{R} \to \mathbb{R}^+$:

$$p(\boldsymbol{x}) = g[f(\boldsymbol{x})].$$

Connecting Diffusion to Evolution

Three problems need to be solved:

Mapping fitness to probability density

Higher fitness $f(x)$ corresponds to higher density $p(x)$, which requires a mapping function $g : \mathbb{R} \to \mathbb{R}^+$:

$$p(\boldsymbol{x}) = g[f(\boldsymbol{x})].$$

Denoise Model†

Diffusion Models have $\epsilon(x_t, t)$ to predict noise, evolutionary algorithm also need a predictive model:

$$p(\epsilon|x_t), etc.$$

Connecting Diffusion to Evolution

Three problems need to be solved:

Mapping fitness to probability density

Higher fitness $f(x)$ corresponds to higher density $p(x)$, which requires a mapping function $g : \mathbb{R} \to \mathbb{R}^+$:

$$p(\boldsymbol{x}) = g[f(\boldsymbol{x})].$$

Denoise Model†

Diffusion Models have $\epsilon(x_t, t)$ to predict noise, evolutionary algorithm also need a predictive model:

$$p(\epsilon|x_t), etc.$$

Determine iteration method

Directly use iteration method from diffusion models.

Diffusion models are doing early prediction

Most common understanding: diffusion models are trained to predict added noise: $p(\epsilon|x_t)$.

It is hard to analyze in this perspective. However, they are also predicting the origin directly:

$$\boldsymbol{x}_t = \sqrt{\alpha_t}\boldsymbol{x}_0 + \sqrt{1-\alpha_t}\boldsymbol{\epsilon} \Longleftrightarrow \hat{\boldsymbol{x}}_0 = \frac{\boldsymbol{x}_t - \sqrt{1-\alpha_t}\boldsymbol{\epsilon}}{\sqrt{\alpha_t}}$$

Hence,

$$\boldsymbol{x}_{t-1} = \sqrt{\alpha_{t-1}}\hat{\boldsymbol{x}}_0 + \sqrt{1-\alpha_{t-1}-\sigma_t^2}\frac{\boldsymbol{x}_t - \sqrt{\alpha_t}\hat{\boldsymbol{x}}_0}{\sqrt{1-\alpha_t}} + \sigma_t \boldsymbol{w}$$

Early prediction $p(x_0 = x|x_t)$

Diffusion models are making **early predictions**, then move toward it by small steps. The model is also predicting $p(x_0 = x|x_t)$.

Estimating high-fitness parameter

Apply Bayesian equation on early prediction:

$$p(\boldsymbol{x}_0 = \boldsymbol{x}|\boldsymbol{x}_t) = \frac{\overbrace{p(\boldsymbol{x}_t|\boldsymbol{x}_0 = \boldsymbol{x})}^{\mathcal{N}(\boldsymbol{x}_t;\sqrt{\alpha_t}\boldsymbol{x},1-\alpha_t)} \overbrace{p(\boldsymbol{x}_0 = \boldsymbol{x})}^{g[f(\boldsymbol{x})]}}{p(\boldsymbol{x}_t)}$$

Diffusion models are trained with MSE loss $\rightarrow$ We need average over $\boldsymbol{x}$:

$$\hat{\boldsymbol{x}}_0(\boldsymbol{x}_t, \boldsymbol{\alpha}, t) = \sum_{\boldsymbol{x}\sim p_{\text{eval}}(\boldsymbol{x})} p(\boldsymbol{x}_0 = \boldsymbol{x}|\boldsymbol{x}_t)\boldsymbol{x} = \frac{1}{Z}\sum_{\boldsymbol{x}\in \boldsymbol{X}_t} g[f(\boldsymbol{x})]\mathcal{N}(\boldsymbol{x}_t; \sqrt{\alpha_t}\boldsymbol{x}, 1-\alpha_t)\boldsymbol{x},$$

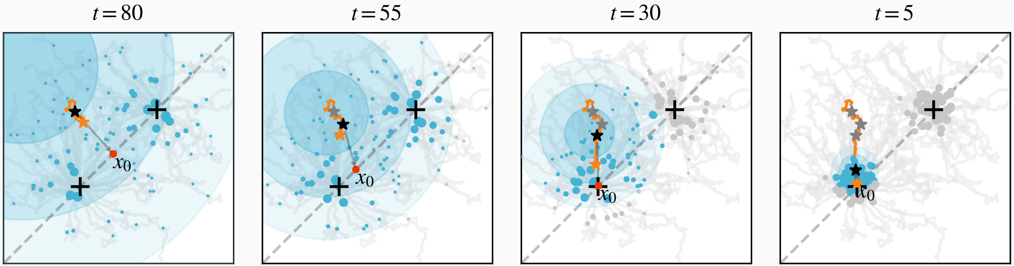
Take this back to diffusion models sampler, we got **Diffusion Evolution**!

$$\boldsymbol{x}_{t-1} = \sqrt{\alpha_{t-1}}\hat{\boldsymbol{x}}_0 + \sqrt{1-\alpha_{t-1}-\sigma_t^2}\,\frac{\boldsymbol{x}_t - \sqrt{\alpha_t}\hat{\boldsymbol{x}}_0}{\sqrt{1-\alpha_t}} + \sigma_t \boldsymbol{w}$$

Diffusion Evolution Algorithm

An example:

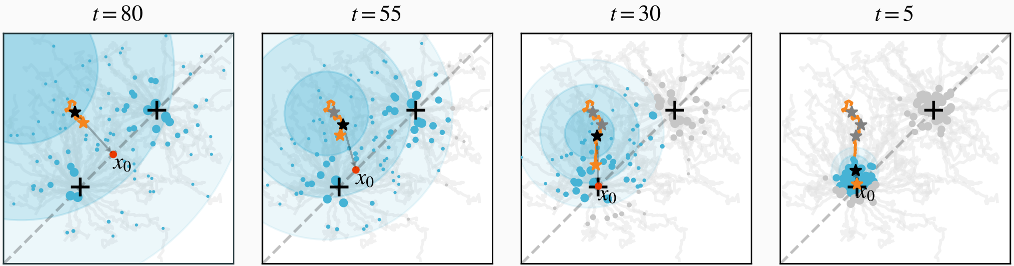
1. Random initialized population $\{\boldsymbol{x}_T^{(i)}\} \sim \mathcal{N}(0, I)$
2. Evaluate fitness and compute density $g[f(\boldsymbol{x})]$
3. Estimate $\hat{\boldsymbol{x}}_0^{(i)}$ for each individual
4. Move toward the $\hat{\boldsymbol{x}}_0^{(i)}$ plus mutation noise $\sigma_t \boldsymbol{w}$



Selection, Mutation, and Reproductive Isolation

$$\boldsymbol{x}_{t-1} = \underbrace{\sqrt{\alpha_{t-1}}\hat{\boldsymbol{x}}_0 + \sqrt{1-\alpha_{t-1}-\sigma_t^2}\,\frac{\boldsymbol{x}_t - \sqrt{\alpha_t}\hat{\boldsymbol{x}}_0}{\sqrt{1-\alpha_t}}}_{\text{directed evolution}} + \underbrace{\sigma_t \boldsymbol{w}}_{\text{undirected mutation}}$$

$$\hat{\boldsymbol{x}}_0(\boldsymbol{x}_t, \boldsymbol{\alpha}, t) = \frac{1}{Z}\sum_{\boldsymbol{x}\in\boldsymbol{X}_t} \underbrace{g[f(\boldsymbol{x})]}_{\text{selection}}\underbrace{\mathcal{N}(\boldsymbol{x}_t; \sqrt{\alpha_t}\boldsymbol{x}, 1-\alpha_t)}_{\text{reproductive isolation}}\boldsymbol{x}$$



Mapping Between Diffusion and Evolution

Diffusion		Evolution
MES loss	$\leftrightarrow$	Weighted average on $\hat{\boldsymbol{x}}_0$
$\boldsymbol{x}_t = \sqrt{\alpha_t}\boldsymbol{x}_0 + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}$	$\leftrightarrow$	Gaussian weight as "reproductive isolation"
DDPM/DDIM sampling	$\leftrightarrow$	Evolution iteration

Table 1: Diffusion models can be decomposed into three parts, associate to three parts in evolutionary algorithm respectively.

Experiments: parallel between diffusion and evolution

Latent Space Diffusion Evolution

Original Diffusion Evolution struggles on high-dimensional space:

$$\hat{\boldsymbol{x}}_0(\boldsymbol{x}_t, \boldsymbol{\alpha}, t) = \frac{1}{Z} \sum_{\boldsymbol{x} \in \boldsymbol{X}_t} g[f(\boldsymbol{x})] \quad \underbrace{\mathcal{N}(\boldsymbol{x}_t; \sqrt{\alpha_t}\boldsymbol{x}, 1 - \alpha_t)}_{\text{become more local at high-dimension}} \quad \boldsymbol{x}$$

Latent Space Diffusion Evolution

Evolve at lower-dimensional space may resolve this problem.

$$\begin{tikzcd}
\boldsymbol{z}_T \arrow[r] & \boldsymbol{z}_{T-1} \arrow[r] & \cdots \arrow[r] & \boldsymbol{z}_1 \arrow[r] & \boldsymbol{z}_0 \arrow[d, "\text{generate}", "\mathbb{R}^d \to \mathbb{R}^D"'] \\
\boldsymbol{x}_T \arrow[u, "\text{encode}", "\mathbb{R}^D \to \mathbb{R}^d"'] & & & & \boldsymbol{x}_0
\end{tikzcd}$$

Diffusion Evolution on high-dimensional space

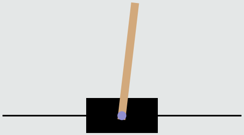
We train neural networks to solve Cart-pole system by Latent Space Diffusion Evolution.

Cart-pole system

Using the cart's survival time as the reward, the cart ends when:

- If cart exceed certain range;
- If pole falls down

Four inputs $(x, \theta, \dot{x}, \dot{\theta})$ and two possible control signals (move left and right).



Diffusion Evolution on high-dimensional space

We train neural networks to solve Cart-pole system by Latent Space Diffusion Evolution.

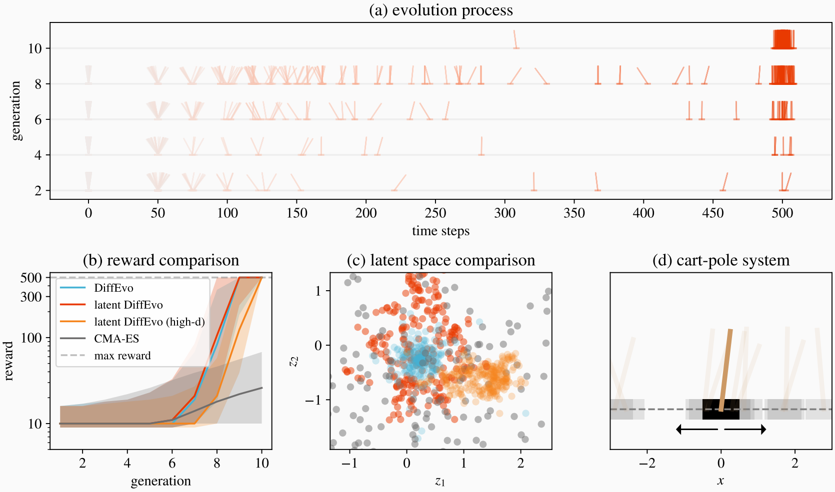
Neural network controller

- **Simple version:** Three layers, $(4, 8, 2)$, with 58 neurons;
- **Complex version:** Four layers, $(4, 128, 128, 2)$, with $17, 410$ neurons.

Both use ReLU activation.

Diffusion Evolution on high-dimensional space

We train neural networks to solve Cart-pole system by Latent Space Diffusion Evolution.



QD-Score

Table 2: QD-scores

	Diffusion Evolution	Latent Diffusion Evolution	CMA-ES	PEPG	Open-ES	MAP-Elite
Rosenbrock	35.4 (1.82)	23.4 (9.78)	1.00 (0.00)	1.25 (0.43)	0.73 (0.12)	**42.0** (0.36)
Beale	20.0 (0.94)	13.0 (4.50)	1.00 (0.00)	2.00 (0.00)	1.84 (0.72)	**23.7** (0.48)
Himmelblau	15.8 (1.18)	11.4 (3.99)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	**18.1** (0.42)
Ackley	28.3 (1.35)	16.7 (9.07)	13.4 (4.53)	1.13 (0.34)	2.14 (0.63)	**33.0** (0.53)
Rastrigin^{2}	35.2 (2.18)	17.8 (9.36)	18.9 (6.24)	2.30 (0.64)	3.99 (0.00)	**45.2** (0.93)
Rastrigin^{4}	10.6 (0.50)	**33.1** (6.66)	6.03 (2.30)	0.67 (0.12)	1.01 (0.00)	13.5 (0.22)
Rastrigin^{32}	0.96 (0.04)	**73.4** (2.10)	0.12 (0.03)	0.06 (0.01)	0.07 (0.01)	1.20 (0.02)
Rastrigin^{256}	0.11 (0.01)	**70.2** (0.62)	0.01 (0.00)	0.01 (0.00)	0.00 (0.00)	0.14 (0.00)

Code

Source code available:

https://github.com/Zhangyanbo/diffusion-evolution

ICLR 2025:

https://openreview.net/forum?id=xVefsBbG2O

email: Yanbo.Zhang@tufts.edu

Yanbo
Zhang

Benedikt
Hartl

Hananel
Hazan

Michael
Levin

References i

J. Ho, A. Jain, and P. Abbeel.
Denoising diffusion probabilistic models.
Advances in neural information processing systems, 33:6840–6851, 2020.

J. Song, C. Meng, and S. Ermon.
Denoising diffusion implicit models.
arXiv preprint arXiv:2010.02502, 2020.

Appendix I: Algorithm

Require: Population size N, parameter dimension D, fitness function f, density mapping function g, total evolution steps T, diffusion schedule $\boldsymbol{\alpha}$ and noise schedule $\boldsymbol{\sigma}$.

$[\boldsymbol{x}_T^{(1)}, \boldsymbol{x}_T^{(2)}, ..., \boldsymbol{x}_T^{(N)}] \leftarrow \mathcal{N}(0, I^{N\times D})$ ▷ Initialize population

for $t \in [T, T-1, ..., 2]$ **do**

  $\forall i \in [1, N] : Q_i \leftarrow g[f(\boldsymbol{x}_t^{(i)})]$ ▷ Fitness are cached to avoid repeated evaluations

  for $i \in [1, 2, .., N]$ **do**

$$Z \leftarrow \sum_{j=1}^{N} Q_j \mathcal{N}(\boldsymbol{x}_t^{(i)}; \sqrt{\alpha_t}\boldsymbol{x}_t^{(j)}, 1-\alpha_t)$$

$$\hat{\boldsymbol{x}}_0 \leftarrow \frac{1}{Z} \sum_{j=1}^{N} Q_j \mathcal{N}(\boldsymbol{x}_t^{(i)}; \sqrt{\alpha_t}\boldsymbol{x}_t^{(j)}, 1-\alpha_t)\boldsymbol{x}_t^{(j)}$$

$$\boldsymbol{w} \leftarrow \mathcal{N}(0, I^D)$$

$$\boldsymbol{x}_{t-1}^{(i)} \leftarrow \sqrt{\alpha_{t-1}}\hat{\boldsymbol{x}}_0 + \sqrt{1-\alpha_{t-1}-\sigma_t^2}\frac{\boldsymbol{x}_t^{(i)} - \sqrt{\alpha_t}\hat{\boldsymbol{x}}_0}{\sqrt{1-\alpha_t}} + \sigma_t \boldsymbol{w}$$

  end for

end for